Алгоритмическая теория игр

М. Вялый

Черновой вариант от 17 января 2020 г.

Текст составлен по материалам одноименного курса, прочитанного осенью 2019 г. в Санкт-Петербурге, Computer Science club. (страница курса `compsciclub.ru/courses/algo-game-theory/2019-autumn/about/`).
Это текущая версия невычитанного черновика, поэтому в нём возможны как содержательные ошибки и неточности, так и нестыковки между разными частями текста. Просьба о замеченных ошибках и неточностях сообщать автору (адрес `vyalyi`⟨стандартный суффикс gmail⟩).
В конце, на с. 120, приводится список изменений в разных версиях текста.

Оглавление

Предисловие **4**

1 Конечные игры: основные определения и результаты **5**
1.1 Игры на выигрыш на ациклическом графе 5
1.2 Игры на результат на ациклическом графе 9

2 Конечные игры и машины Тьюринга **13**
2.1 Модели вычислений и сложностные классы 14
2.1.1 Машины Тьюринга 14
2.1.2 Сложностные классы 15
2.1.3 Сводимости 16
2.2 Альтернирующие машины Тьюринга 17
2.3 Альтернация превращает время в память 18
2.4 Класс **PSPACE** и **PSPACE**-полные задачи 21

3 Беспристрастные игры **27**
3.1 Беспристрастные игры: оценка позиций и ним-функция 28
3.1.1 Ним 29
3.1.2 Ним-функция 30
3.1.3 Сумма игр 30
3.2 Игры вычитания 33
3.3 Многомерные игры вычитания 34

4 Игры на вычитания и клеточные автоматы **36**
4.1 Клеточные автоматы 36
4.2 Двоичные клеточные автоматы и модулярные игры 39
4.3 Избавление от остатков 42
4.4 Трудность многомерных игр вычитания 44
4.4.1 Теорема об иерархии 44
4.4.2 MT, которая работает на всех входах сразу 45
4.4.3 Доказательство теоремы о трудности игр вычитания 47

5 Бесконечные игры **49**
5.1 О детерминированности бесконечных игр 49
5.2 Игры достижимости 52
5.3 Игры чётности 54
5.4 Циклические игры 57
5.4.1 Каноническая форма циклической игры 58
5.4.2 Эквивалентные игры и потенциальные преобразования 59
5.5 Связь между играми чётности и циклическими играми 60

6 Теорема Колберга и позиционные стратегии в бесконечных играх **62**
6.1 Игры с затухающим платежом 62
6.2 От сжимающих отображений к нерастягивающим 66
6.3 Теорема о канонической игре из теоремы Колберга 68
6.4 Доказательство теоремы Колберга и следствий 68

7 Алгоритмические следствия из теоремы Колберга **72**
7.1 Языки, связанные с решением игр 72
7.2 Сводимости 74
7.3 Принадлежность классу $\mathbf{NP} \cap \text{co-}\mathbf{NP}$ 76
7.4 Псевдополиномиальные алгоритмы 77

8 Игры достижимости, игры чётности, игры с судьёй **80**
8.1 Игры с судьёй 80
8.2 Автомат с несколькими счётчиками 82
8.3 Абстрактные счётчики на упорядоченных деревьях 84
8.4 Полнота автомата на дереве сильно связных компонент стратегического графа 86
8.5 Сохранение полноты при расширении дерева 89

9 Универсальные деревья и квазиполиномиальные алгоритмы **91**
9.1 Почти оптимальная конструкция 92
9.2 Оценка размера 93
9.3 Собирая всё вместе 94
9.4 Нижние оценки на размер универсальных деревьев 95

10 Субэкспоненциальные алгоритмы для циклических игр **97**
10.1 Решение циклических игр: меры успешности стратегии 97
10.2 Вероятностный алгоритм поиска оптимальной стратегии 102
10.3 Оптимизация функций на многомерных боксах 104
10.4 Общий алгоритм оптимизации 110
10.5 Решение рекурренты 115

Литература **118**

Предисловие

Курс состоит из нескольких тем, относящихся к теории игр и алгоритмической сложности решения игр. Предварительного знания основных определений и фактов в теории игр не требуется: они будут, хотя бы кратко, рассказаны.

В начале будут рассказаны классические результаты о решении конечных игр и приведены примеры PSPACE-полных игр. Далее будет обсуждено (не-)существование выигрышных стратегий для бесконечных игр и дано доказательство существования особо простых стратегий (позиционные стратегии, стратегии с конечной памятью) для игр из нескольких интересных классов: игры чётности, циклические игры. Существование таких стратегий имеет важные следствия для алгоритмической сложности решения таких игр, помещая эту задачу в пересечение классов NP и co-NP.

Завершающей темой курса является изложение недавно придуманных квазиполиномиальных алгоритмов для решения игр чётности.

Для понимания курса желательно знакомство с основными понятиями теории сложности вычислений: машины Тьюринга, основные сложностные классы, понятие сводимости. Текст содержит лишь короткие напоминания по этим вопросам. Для более подробного ознакомления с теорией сложности рекомендуются стандартные учебники, например, учебники Сипсера [17] или Ароры и Барака [4]

Желательно также знакомство с базовыми алгоритмами (топологическая сортировка, построение кратчайших путей и т.п.).

Тема 1

Конечные игры: основные определения и результаты

Существует много разных игр, даже если ограничиться теми, которые представляют интерес с точки зрения алгоритмической сложности. Название курса слегка обманывает. Мы не будем рассматривать *все* вопросы, относящиеся к алгоритмической сложности решения игр.

Основной целью курса является рассказ о новых алгоритмах решения так называемых игр чётности (parity games, PG). Попутно будет рассказано и о связанных с ними играх, так называемых циклических играх (mean-payoff games, MPG) и играх с затухающим платежом (discounted payoff games, DPG).

Все эти игры примечательны тем, что каждая партия продолжается бесконечное время, однако у игроков всегда есть выигрышные стратегии достаточно простого вида.

Начнём мы всё-таки с игр, которые продолжаются конечное время. Для полноты изложения напомним основные определения и факты о конечных играх.

После этого мы обсудим классическую, хотя и побочную для данного курса тему: насколько трудно решать конечные игры в сжатом представлении.

1.1. Игры на выигрыш на ациклическом графе

Мы будем рассматривать только игры двух игроков с нулевой суммой. Последнее означает, что результат в любой партии противоположен для двух игроков: сколько один выиграл, столько другой проиграл. Игроков будем для краткости обозначать 0 и 1. Далее мы иногда будем давать им имена, которые обозначают цель игрока.

Общая картина такой игры состоит в следующем. Есть множество *позиций*, в каждый момент времени игра находится в одной из позиций. Каждая партия игры начинается с *начальной позиции*. В каждой позиции определено, какой из игроков делает ход. Ход приводит к изменению позиции. Некоторые позиции объявлены *терминальными*. В терминальной позиции партия игры заканчивается и указан

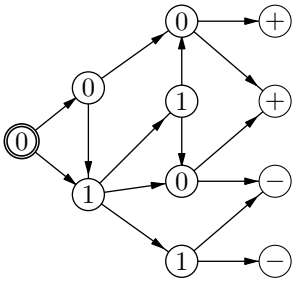


Рис. 1.1: Пример игрового поля

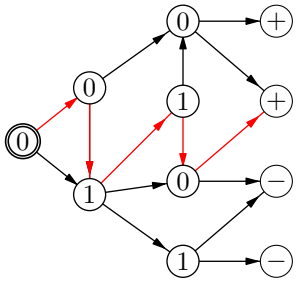


Рис. 1.2: Партия игры

результат игры: кто из игроков выиграл. В некоторых играх результат будет числом, указывающим, сколько именно выиграл игрок 1.

Множество позиций будет всегда конечным. Поэтому любая игра указанного вида задаётся ориентированным графом $G = (V, E)$ (в дальнейшем называем его *графом игры*), вершины которого обозначают позиции, а рёбра указывают возможные ходы. Позиции разбиты на два множества $V = V_0 \sqcup V_1$, в вершинах из V_0 ход делает игрок 0, в вершинах из V_1 — игрок 1. Среди вершин графа выделена также начальная позиция v_0. Каждой терминальной позиции (вершина исходящей степени 0) приписан результат. Пример такой игры показан на рисунке 1.1. Начальная позиция отмечена двойной линией. В терминальных позициях, помеченных знаком $+$, выигрывает игрок 1, а в помеченных знаком $-$ выигрывает игрок 0.

Такой набор данных, описывающих игру, будем называть *игровым полем* (arena).

Партия игры задаётся маршрутом на игровом поле, начинающимся в начальной позиции и либо заканчивающимся в одной из терминальных позиций, либо не заканчивающимся вовсе. На рисунке 1.2 изображена возможная партия игры. В этой партии выигрывает игрок 1.

Легко видеть, что если граф игры ациклический, то любая партия игры конечна. Пока ограничимся этим случаем.

Мы всюду предполагаем, что цель каждого из игроков состоит в том, чтобы выиграть. В таком предположении можно определить основное для теории игр понятие *выигрывающей стратегии*.

Стратегия — это правило, определяющее поведение игроков. Более формально, стратегия s_i игрока i — это функция, которая сопоставляет каждому маршруту τ по графу игры, начинающемуся в начальной позиции v_0 и заканчивающемуся в нетерминальной позиции $v \in V_i$, ребро $(v, s_i(\tau, v))$, то есть ход, который должен сделать игрок, *придерживающийся данной стратегии*.

Стратегия s_i называется *выигрывающей* для игрока i, если в каждой партии, в которой игрок i придерживается этой стратегии, он выигрывает (как бы ни играл противник).

Среди стратегий выделяются стратегии особенно простого вида: *позиционные* (memoryless). Позиционная стратегия определяет ход игрока только по текущей позиции, игнорируя все предыдущие ходы. То есть, позиционная стратегия игрока i —

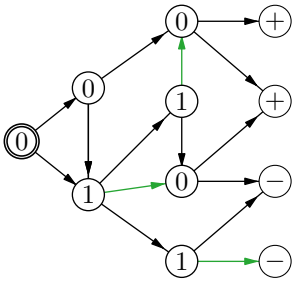


Рис. 1.3: Пример позиционной стратегии игрока 1

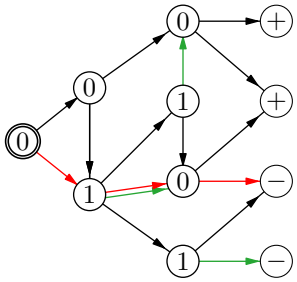


Рис. 1.4: Партия, в которой игрок 1 придерживается позиционной стратегии

это функция $s_i \colon V_i \to V$, удовлетворяющая ограничению: $(v, s_i(v)) \in E$ (то есть предписываемый стратегией ход возможен по правилам игры). На рисунке 1.3 приведён пример позиционной стратегии игрока 1, а на рисунке 1.4: пример партии, в которой игрок 1 придерживается данной стратегии. Рёбра, которые входят в стратегию, обозначены зелёным цветом. Из рисунка 1.4 видно, что стратегия на рисунке 1.3 не является выигрывающей для игрока 1, так как в данной партии выигрывает игрок 0.

Фундаментальный, хотя и очень простой, факт состоит в наличии позиционных выигрывающих стратегий у одного из игроков в рассматриваемом нами сейчас случае конечного ациклического графа.

Теорема 1.1. *Если граф игры ациклический, то у одного из игроков есть выигрывающая позиционная стратегия.*

Доказательство. Будем доказывать существование позиционной стратегии «разбором с конца» или «обратной индукцией». Этот приём основан на хорошо известном факте: вершины ациклического графа можно занумеровать так, чтобы каждое ребро вело из вершины с меньшим номером в вершину с большим номером. Такая нумерация называется «топологической сортировкой» или «линейным продолжением».

Упражнение 1. Докажите это утверждение.

Итак, зафиксируем такую нумерацию позиций графа игры $v_1, \ldots, v_n$, которая согласована с направлением рёбер: если $(v_i, v_j) \in E$, то $i < j$.

Определим пару позиционных стратегий s_0, s_1 и припишем каждой позиции результат игры w: 0 или 1. Позиция v_n необходимо терминальная, из неё не ведёт ни одного ребра. Результат в этой позиции совпадает с номером того игрока, который выигрывает в этой терминальной позиции.

Пусть стратегии s_0, s_1 и результат w уже определены для позиций с номерами больше t.

Если позиция v_t терминальная, то определять ход в этой позиции не нужно. Результат равен номеру игрока, который выигрывает в данной позиции.

Пусть в позиции v_t ход делает игрок 0. Определим результат игры в этой позиции, Если из v_t есть ход в позицию v_j, $j > t$, в которой результат равен 0, результат

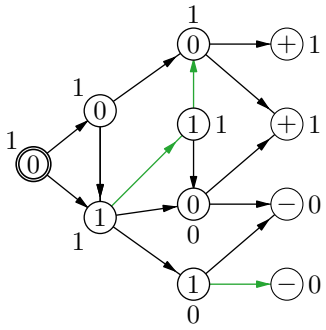


Рис. 1.5: Выигрывающая позиционная стратегия игрока 1 и результаты игры для всех позиций

$w(v_t) = 0$. Иначе результат равен 1. В первом случае определяем стратегию s_0 как $s_0(v_t) = v_j$. Во втором определяем стратегию s_0 произвольно.

Аналогично, с переменой игроков и значений результатов, определяем результат и стратегию в случае $v_t \in V_1$.

Теперь докажем, всё той же индукцией по убыванию номера в линейном упорядочении, что если партия начинается в позиции v_t, то игрок $w(v_t)$ выигрывает, если придерживается позиционной стратегии $s_{w(v_t)}$.

При $t = n$, как и для всех остальных терминальных вершин, это утверждение очевидно выполняется. Предположим, что оно доказано для всех номеров больше t. Пусть партия начинается в позиции v_t и вычисленный описанной выше процедурой результат равен $i = w(v_t)$.

Если $v_t \in V_i$, то по построению у игрока i есть ход $(v_t, s_i(v_t)) \in E$. По индуктивному предположению для позиции $s_i(v_t)$ утверждение уже доказано. Значит, если игрок делает ход из v_t в $s_i(v_t)$ и придерживается далее стратегии s_i, он выигрывает.

Если $v_t \notin V_{1-i}$, то по построению любой ход игрока $1 - i$ ведёт в позицию $j > t$, в которой результат также равен i. По индуктивному предположению для позиции v_j утверждение уже доказано. Значит, в любой партии, начинающейся в v_t игрок i выигрывает, если придерживается стратегии s_i. □

На рисунке 1.5 указана выигрывающая позиционная стратегия игрока 1.

Под *решением игры* мы понимаем ответ на вопрос «у кого из игроков есть выигрывающая стратегия в игре»?

Доказательство теоремы 1.1 даёт эффективный алгоритм решения игры. Под эффективным алгоритмом мы понимаем алгоритм, работающий за полиномиальное время от длины входа. В сущности, единственный нетривиальный шаг состоит в топологической сортировке, все дальнейшие действия, описанные в доказательстве, требуют времени $O(|E|)$. Полиномиальные алгоритмы топологической сортировки хорошо известны.

1.2. Игры на результат на ациклическом графе

Выше мы рассмотрели случай, когда результат игры бинарен: выигрыш одного из игроков. Практически так же анализируются и игры, в которых результатом является число (как мы договорились выше, выигрыш игрока 1; он же проигрыш игрока 0).

В простейшем случае результат игры определяется в терминальной позиции (терминальный платёж). Пример такой игры приведён на рисунке 1.6.

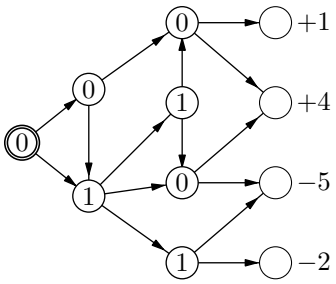


Рис. 1.6: Пример игрового поля, у терминалов указаны результаты игры

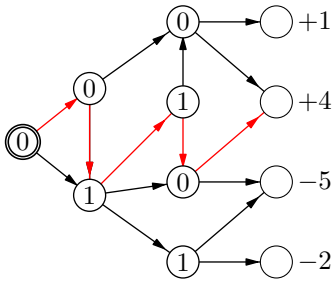


Рис. 1.7: Партия игры с результатом +4 (выигрыш игрока 1)

Когда результат игры является числом, понятие выигрывающей стратегии теряет смысл. Вместо этого появляется понятие стратегии, гарантирующей результат a: если игрок 1 (максимизатор) придерживается стратегии s, то результат в партии не меньше a; аналогично, если игрок 0 (минимизатор) придерживается стратегии s, то результат в партии не больше a.

Ценой игры называется такое число c, что у каждого из игроков существует стратегия, гарантирующая результат c.

Упражнение 2. Проверьте, что если цена игры существует, то определена однозначно.

Разбором с конца нетрудно доказать существование цены у любой игры с терминальными платежами на ациклическом графе.

Теорема 1.2. *Если граф игры ациклический и платежи терминальные, то в игре существует цена, а гарантирующие её стратегии можно выбрать позиционными.*

Доказательство. Аналогично доказательству в случае игр на выигрыш. Теперь при обратной индукции вычисляется не результат игры в каждой позиции, а цена позиции.

В терминальной позиции цена игры равна тому результату, которой приписан данной позиции.

Пусть стратегии s_0, s_1 и цена c уже определены для позиций с номерами больше t.

Если в позиции v_t ход делает игрок 0 (минимизатор), то ценой позиции является минимум $c(v_j)$ по тем позициям j, куда возможен ход из позиции v_t. Стратегия s_0 определяет $s_0(v_t)$ как одну из тех позиций, на которых достигается этот минимум.

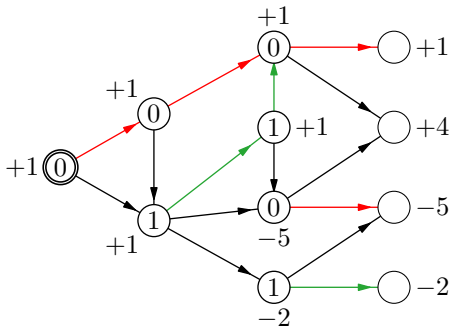


Рис. 1.8: Анализ игры

Если в позиции v_t ход делает игрок 1 (максимизатор), то ценой позиции является максимум $c(v_j)$ по тем позициям j, куда возможен ход из позиции v_t. Стратегия s_1 определяет $s_0(v_t)$ как одну из тех позиций, на которых достигается этот максимум.

Теперь докажем, индукцией по убыванию номера в линейном упорядочении, что если игра начинается в позиции v_t, то у игры есть цена $c(v_t)$ и её гарантируют позиционные стратегии s_0, s_1.

Действительно, для терминальных вершин это очевидно.

Если $v_t \in V_0$, то по построению у игрока 0 есть такой ход $(v_t, s_0(v_t)) \in E$, что $c(v_t) = c(s_0(v_t))$ нет хода в позицию с меньшим значением цены. По индуктивному предположению для позиции $s_0(v_t)$ это означает, что s_0 гарантирует цену не больше $c(v_t)$, если игра начинается с позиции v_t. Из индуктивного предположения также следует, что стратегия s_1 гарантирует для игрока 1 результат не меньше $c(v_j)$ в любой позиции $j > t$. Поэтому в позиции v_t стратегия s_1 гарантирует результат не меньше $c(v_t)$.

Аналогично рассматривается и случай $v_t \in V_0$. □

На рисунке указаны цены во всех позициях и отмечены гарантирующие их стратегии: зелёным — стратегия игрока 1, красным — стратегия игрока 0.

Далее мы будем рассматривать также игры с аддитивными платежами. В этом случае каждый ход (u, v) имеет стоимость $w(u, v)$ (прибавка к выигрышу; она попрежнему противоположна для двух игроков). Результат партии равен сумме стоимостей по всем рёбрам партии.

Пример такой игры приведён на рисунке 1.9.

Цена игры существует и в этом случае, и доказательство также основано на разборе с конца.

Теорема 1.3. *Если граф игры ациклический, а платежи аддитивные, то в игре существует цена, причём гарантирующие её стратегии можно выбрать позиционными.*

Доказательство. Практически дословно повторяется предыдущее доказательство.

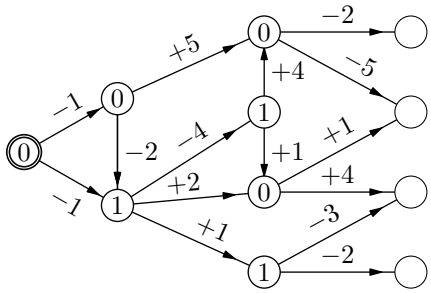


Рис. 1.9: Аддитивные платежи, указаны на рёбрах

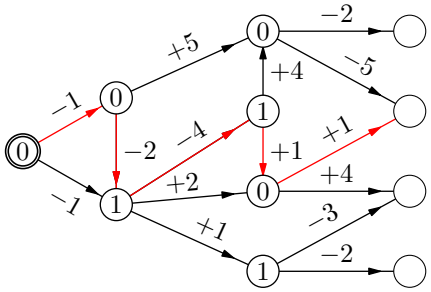


Рис. 1.10: Партия игры с результатом -5 (проигрыш игрока 1)

Только теперь цена в позиции v_t вычисляется по формулам

$$c(v_t) = 0, \qquad \text{если } v_t \text{ терминальная;} \tag{1.1}$$

$$c(v_t) = \min_{v_j:(v_t,v_j)\in E} \big(w(v_t,v_j) + c(v_j)\big), \qquad \text{если } v_t \in V_0; \tag{1.2}$$

$$c(v_t) = \max_{v_j:(v_t,v_j)\in E} \big(w(v_t,v_j) + c(v_j)\big), \qquad \text{если } v_t \in V_1. \tag{1.3}$$

Идея, стоящая за этими формулами, прозрачна: если делать ход (v_t, v_j) из позиции v_t, а дальше играть оптимально (то есть придерживаясь стратегии, гарантирующей цену игры), то в такой партии цена будет не меньше $w(v_t, v_j) + c(v_j)$ для игрока 1 и не больше $w(v_t, v_j) + c(v_j)$ для игрока 0. Игрок 1 стремится увеличить результат, поэтому берёт максимум по таким величинам. Цель игрока 0 противоположная, поэтому он берёт минимум. □

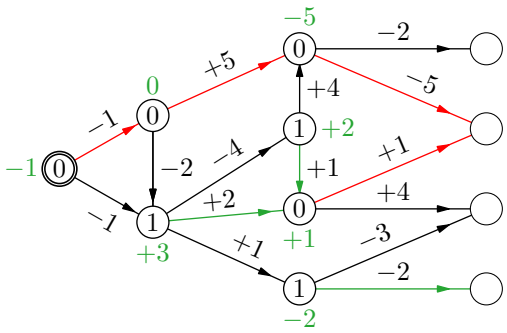


Рис. 1.11: Анализ игры с аддитивными платежами, цены указаны зелёным

На рисунке 1.11 указаны цены во всех позициях и отмечены гарантирующие их стратегии: зелёным — стратегия игрока 1, красным — стратегия игрока 0. В частности, из этого рисунка видно, что цена всей игры (с отмеченной начальной позицией) равна -1, то есть игра выгодна для игрока 0.

Как и в случае игр на выигрыш, доказательства теорем 1.2 и 1.3 превращаются в эффективные алгоритмы вычисления цены игры и позиционных стратегий, гарантирующих эту цену для обоих игроков.

Тема 2

Конечные игры и машины Тьюринга

В обычных играх — наподобие шахмат или го — правила игры не задаются списком всех возможных позиций и возможных ходов. Причина уважительная: возможных позиций очень много. Правила таких игр задают множество возможных позиций неявно, описание правил получается намного короче, чем полный список всех возможных позиций.

Если использовать алгоритмы, основанные на разборе позиций с конца, как в предыдущей главе, то решить шахматы окажется практически невозможно даже на современных компьютерах.

Однако в некоторых случаях решение игр, заданных короткими правилами, оказывается несложным.

Пример 1. Игроки по очереди присваивают значения булевым переменным x_0, ..., x_{100}: игрок 0 присваивает значения переменным с чётными номерами, а игрок 1 — переменным с нечётными номерами. Первым ходит игрок 0. Игра заканчивается, когда всем переменным присвоены значения. Если больше половины переменных имеют значение 1, то выиграл игрок 1; в противном случае выиграл его противник.

В такой игре огромное количество позиций (учтите, что игроки могут присваивать значения не обязательно в порядке возрастания). Тем не менее, решение этой игры очень простое: выигрышная стратегия есть у игрока 0. Она состоит в том, чтобы присваивать каждой переменной с чётным номером значение 0. Таких переменных 51, так что больше половины переменных получат значение 0.

Задача 2.1. Решите аналогичную игру. Теперь правило выигрыша другое: если найдутся три переменные подряд с одинаковым значением, $x_i = x_{i+1} = x_{i+2}$, то выиграл 0, иначе выиграл 1.

Игры, которые задаются не списком позиций, а достаточно коротким описанием правил игры, оказываются тесно связаны со структурной теорией сложности вычислений. В терминах этой теории удаётся формально описать «самые трудные игры» и привести косвенные аргументы против возможности эффективного решения таких игр. (В частности, обобщение го на доску $N \times N$ оказывается «трудным» в этом смысле.)

2.1. Модели вычислений и сложностные классы

Начнём с того, что вспомним базовые понятия из теории сложности вычислений.

2.1.1 Машины Тьюринга

Стандартная модель вычисления — машина Тьюринга. Конкретная машина Тьюринга M задаётся набором $(A, \Lambda, Q, q_0, Q_f, \delta)$, где

- A конечное множество, называемое *алфавитом*;
- $\Lambda \in A$ *пробельный* (или пустой) символ;
- Q конечное множество, элементы которого называются *состояниями*;
- $q_0 \in Q$ *начальное* состояние, $Q_f \subseteq Q$ *финальные* состояния;
- $\delta\colon Q \times A \to Q \times A \times \{-1, 0, +1\}$ *функция переходов.*

Машина Тьюринга имеет доступ к *рабочей ленте*, которая является по сути бесконечным в обе стороны словом в алфавите A. Лишь конечное число символов на ленте отлично от пробельного. В каждый момент времени машине доступна ровно одна ячейка ленты (то есть, один символ бесконечного слова), с которой машина может выполнять действия (чтение, запись). Обычно говорят, что над доступной в данный момент времени ячейкой находится *головка МТ*.

За *такт работы* МТ выполняет действия, схематически изображённые на рисунке 2.1: в соответствии с функцией переходов изменяет состояние, символ в ячейке и сдвигает головку на указанную величину.

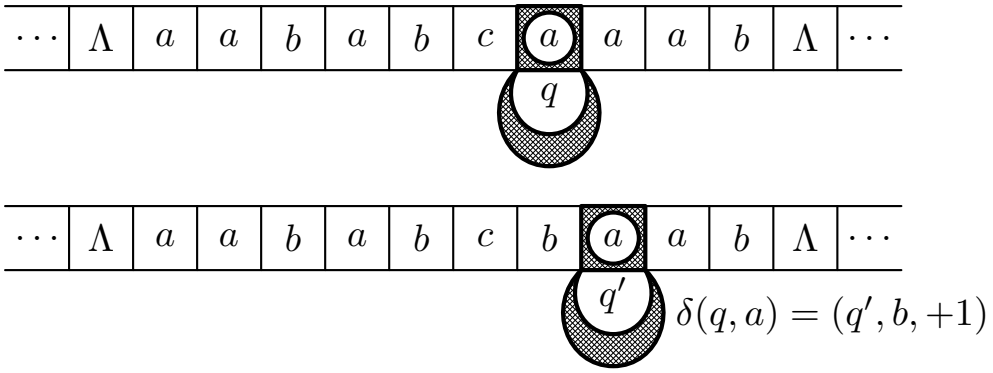


Рис. 2.1: Такт работы машины Тьюринга

Машина начинает работать в *начальной конфигурации*, изображённой на рисунке 2.2: машина находится в начальном состоянии q_0; головка расположена над самой левой ячейкой, содержащей непробельный символ (если все символы ленты пробельные, то положение головки неважно); непробельные символы идут подряд и образуют *входное слово* $w \in (A \setminus \{\Lambda\})^*$.

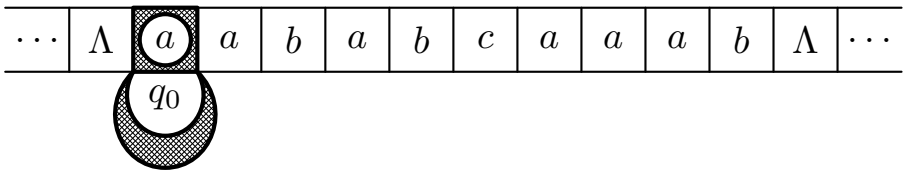


Рис. 2.2: Начальная конфигурация машины Тьюринга

Машина заканчивает работу, когда оказывается в финальном состоянии. *Результат* работы можно определять по-разному, но все разумные способы оказываются по существу эквивалентными.

Будем считать, что результат работы МТ — это слово в алфавите $(A \setminus \{\Lambda\})^*$ от текущего положения головки до ближайшего справа пробельного символа.

Удобно также рассматривать другой вариант МТ — *решающие* МТ. В этом случае множество финальных состояний разбито на два подмножества: $Q_f = Q_y \sqcup Q_n$ (соответственно, *принимающие* и *отвергающие* состояния). Говорят, что решающая МТ *принимает входное слово*, если она заканчивает работу в принимающем состоянии. Мы предполагаем, что алфавит машины содержит все символы, входящие в слово.

Две основных характеристики работы МТ — время (количество тактов) и память (максимальный размер непустой части ленты за всё время работы).

Замечание 2.1. Эффективные алгоритмы неудобно описывать в такой модели, для этого обычно используется модель RAM с размером слова, ограниченным логарифмом от длины входа.

2.1.2 Сложностные классы

Среди всего разнообразия алгоритмических задач мы выделим *задачи разрешения*. Пусть задано множество L слов в алфавите $\{0, 1\}$ (как обычно говорят, *язык*). Задача разрешения языка L состоит в том, чтобы по входному слову w проверить, принадлежит ли оно языку L.

Говорят, что решающая МТ *распознает язык* L, если она принимает в точности слова из языка L.

Сложностные классы состоят из языков, задачи разрешения которых требуют некоторого заданного количества ресурсов. Мы предполагаем, что используются решающие машины Тьюринга.

Приведём основные примеры сложностных классов.

Класс DTIME($T(n)$). Язык L принадлежит этому классу, если существует решающая МТ M_L, которая принимает в точности слова из языка L и работает за время $O(T(n))$, где n — длина входа.

Класс DSPACE($S(n)$). Язык L принадлежит этому классу, если существует решающая МТ M_L, которая принимает в точности слова из языка L и работает на памяти $O(S(n))$, где n — длина входа.

Класс P (полиномиальное время):

$$\mathrm{P} = \bigcup_{k=1}^{\infty} \mathrm{DTIME}(n^k).$$

Этот класс рассматривается в теории сложности вычислений как основная формализация понятия эффективного вычисления.

Класс PSPACE (полиномиальная память):

$$\mathrm{PSPACE} = \bigcup_{k=1}^{\infty} \mathrm{DSPACE}(n^k).$$

Этот класс интересен для теории сложности. У него существует несколько характеризаций. В частности, одна из них основана на играх. Мы её рассмотрим ниже.

Упражнение 3. Докажите, что $\mathrm{P} \subseteq \mathrm{PSPACE}$.

До сих пор неизвестно, различаются ли класс P и PSPACE. «Стандартная» гипотеза состоит в том, что они различаются.

2.1.3 Сводимости

Ещё нам потребуется понятие сводимости. Напомним определение *полиномиальной сводимости* между языками (то есть подмножествами множества двоичных слов). Язык L_1 сводится к языку L_2 (обозначение $L_1 \leq_p L_2$), если существует такая всюду определённая и вычислимая за полиномиальное время функция $f\colon \{0,1\}^* \to \{0,1\}^*$ на множестве двоичных слов (*сводящая функция*), что $x \in L_1$ равносильно $f(x) \in L_2$.

В силу этого определения композиция сводящей функции и алгоритма разрешения для языка L_2 даёт алгоритм разрешения для языка L_1, время работы которого увеличивается не слишком сильно: оно ограничено полиномом от времени работы алгоритма разрешения для языка L_2.

Это оправдывает использование обозначения $L_1 \leq_p L_2$: язык L_1 в таком случае «проще» языка L_2.

Если для любого языка X из некоторого сложностного класса $\mathcal{C}$ (семейства языков) выполняется $X \leq_p L$, то язык L называется *$\mathcal{C}$-трудным*. Если дополнительно $L \in \mathcal{C}$, то язык L называется *$\mathcal{C}$-полным*.

Полные языки являются в указанном выше смысле самыми трудными в классе. Из-за известных проблем с доказательством нижних оценок алгоритмической сложности полнота в достаточно широком классе является сейчас в теории сложности обычным способом указать на трудность задачи.

Заметим также, что композиция сводимостей является сводимостью. Поэтому если свести язык L_1, для которого уже доказана трудность в классе, к языку L_2, то язык L_2 также трудный в том же классе.

2.2. Альтернирующие машины Тьюринга

АМТ определяется почти так же, как и обычная МТ. Два важных отличия:

1. Все состояния АМТ разбиты на два множества: $Q_\vee$ (экзистенциальные состояния) и $Q_\wedge$ (универсальные состояния).

2. Вместо функции переходов для АМТ определено *отношение переходов*
$$\delta \subseteq Q \times A \times Q \times A \times \{-1, 0, +1\}.$$

Вычисление такой машины не определено однозначно. Теперь это любая последовательность конфигураций (то есть троек (состояние машины, положение головки, содержимое ленты)), в которой каждые две последовательные конфигурации согласованы с отношением переходов, как показано на рисунке 2.3. Вычисление либо заканчивается в финальной конфигурации (то есть конфигурации с финальным состоянием машины); либо продолжается вечно.

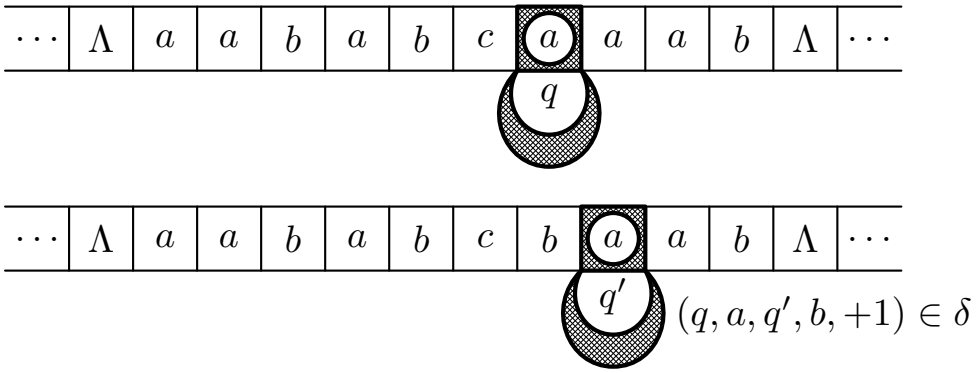


Рис. 2.3: Такт работы АМТ

Как же теперь определить, что АМТ принимает слово? Для этого мы рассмотрим игру на *графе конфигураций АМТ*. Две конфигурации соединены ребром, если из первой можно перейти во вторую согласованным с отношением переходов способом. Договоримся, что финальные конфигурации — тупиковые вершины в этом графе. Считаем, что АМТ решающая, то есть финальные состояния разбиты на принимающие и отвергающие.

В состояниях из $Q_\vee$ (то есть в любой конфигурации, состояние которой принадлежит $Q_\vee$) ходит игрок 1 (Оптимист). В состояниях из $Q_\wedge$ ходит игрок 0 (Пессимист). Игра начинается из начальной конфигурации и заканчивается в конфигурации, состояние которой финальное.

Партия такой игры и есть вычисление АМТ. Игрок 1 выигрывает, если партия завершается в принимающем состоянии.

По определению АМТ принимает входное слово w, если у игрока 1 есть выигрывающая стратегия в описанной выше игре. То есть, чтобы понять, принимается ли данное слово, нужно решить игру, заданную АМТ.

Зафиксировав такую выигрывающую стратегию, можно определить *время работы* АМТ, принимающей входное слово w. Это максимальная длина партии в игре на графе конфигураций, в которой игрок 1 придерживается выбранной стратегии.

Заметим, что граф конфигураций не обязательно ациклический. Поэтому игра не обязательно конечная. На словах, отвергаемых АМТ, определить время работы при таком подходе невозможно.

Эта трудность не возникает, если граф конфигураций ациклический. В таком случае наибольшую длину вычисления можно считать временем работы АМТ и на тех входах, которые отвергаются АМТ.

Возникают новые сложностные классы.

Класс AltTIME($T(n)$). Язык L принадлежит этому классу, если существует АМТ M_L, которая принимает в точности слова из языка L за время $O(T(n))$, где n — длина входа.

Класс AltP (полиномиальное время):

$$\mathrm{AltP} = \bigcup_{k=1}^{\infty} \mathrm{AltTIME}(n^k).$$

В теории сложности вычислений интересен даже случай, когда все состояния контролируются одним из игроков.

Недетерминированная МТ (НМТ) — это АМТ, у которой $Q = Q_\vee$. Сложностные классы $\mathrm{NTIME}(T(n))$ и NP определяются как и раньше, но с заменой АМТ на НМТ.

Как известно, одним из самых интригующих вопросов теории сложности является соотношение между классами P и NP. «Стандартная» гипотеза состоит в том, что эти классы различаются. В этом случае даже решение игр с одним игроком невозможно сделать эффективным в общем случае.

2.3. Альтернация превращает время в память

Альтернирующие классы сложности можно соотнести с обычными сложностными классами. Мы сейчас убедимся, что альтернирующее время не сильнее обычной памяти.

При этом мы ограничимся «разумными» классами $\mathrm{AltTIME}(T(n))$. Функций от натуральных аргументов очень много, среди них есть невычислимые. А нас, в сущности, интересуют асимптотические оценки времени работы достаточно простого вида: полиномиальный рост, экспоненциальный рост и т.п.

Все нужные нам функции с избытком покрываются следующим определением.

Определение 2.1. Функция $f\colon \mathbb{N} \to \mathbb{N}$ называется *конструируемой по памяти*, если существует МТ (обычная, детерминированная), которая по входу 1^n вычисляет $f(n)$ на памяти $O(f(n))$.

Задача 2.2. Докажите, что функции n^k, где k — константа; 2^n; $\lfloor 2^{\sqrt{n}} \rfloor$ конструируемые по памяти.

Замечание 2.2. В некоторых случаях интересна память, много меньшая n, скажем, $\log n$. В сделанных нами определениях вычисления на такой памяти не слишком осмысленны: они не позволяют даже прочитать вход. Когда говорят о сублинейной памяти, имеют в виду другие машины Тьюринга: с лентой входа, которую можно только читать, и рабочей лентой, на которой разрешены и чтение, и запись. Память вычисления в такой модели измеряется только количеством непустых ячеек на рабочей ленте. Поэтому вычисления на сублинейной памяти становятся осмысленными. Нам такая модель не понадобится.

Теорема 2.1. *Пусть $T(n)$ — такая конструируемая по памяти функция, что $T(n) \geqslant n$. Тогда* $\mathrm{AltTIME}(T(n)) \subseteq \mathrm{DSPACE}(T(n))$.

Доказательство. Вспомним, как доказывалось существование выигрывающих стратегий в игре на ациклическом графе. Оно основано на рекурсивной процедуре оценки каждой позиции игры.

Основная идея доказательства состоит в том, что эту рекурсивную процедуру можно реализовать на ограниченной памяти.

Для того, чтобы оценить начальную позицию, нужно помнить в каждый момент времени текущий путь по дереву рекурсии и результаты вычисления по остальным веткам ветвления. Это проиллюстрировано на рисунке 2.4.

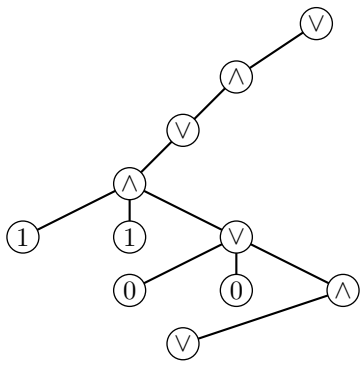


Рис. 2.4: Работа рекурсивной процедуры оценки позиции

Если АМТ M принимает в точности слова из языка L за время $O(T(n))$, то требуемая память ограничена $O(T(n))$, так как возможных ветвлений $O(1)$ (возможных изменений конфигурации не больше, чем $3|Q| \cdot |A|$; здесь Q — множество состояний M, а A — алфавит). Запоминать все конфигурации на каждом шаге рекурсии не требуется: их можно восстановить, зная ветвь рекурсии, выбор ветвлений определяет однозначно вычисление.

Поскольку $T(n)$ конструируемая по памяти, значение $T(n)$ можно вычислить перед началом оценки начальной конфигурации. Далее нужно поддерживать счётчик глубины рекурсии и обрывать ветвление, если глубина рекурсии превышает $CT(n)$. Здесь C — та константа, которая скрыта в обозначении $O(T(n))$. Её вычислять не нужно, она зависит только от машины M и не зависит от входа. □

В обратную сторону можно лишь доказать более слабый результат, но зато для произвольных функций.

Теорема 2.2. $\mathrm{DSPACE}(S(n)) \subseteq \mathrm{AltTIME}(S(n)^2)$.

Доказательство. Зафиксируем МТ M, распознающую язык $L \in \mathrm{DSPACE}(S(n))$ на памяти $C \cdot S(n)$. Множество состояний M обозначим Q, алфавит — A.

АМТ M', которая принимает в точности слова из языка L, реализует рекурсивную процедуру вычисления функции $R(\alpha, \beta, t)$, которая равна 1, если из конфигурации α машина M переходит в конфигурацию β за t тактов работы, и равна 0 в противном случае.

Если $t = 0$, то $R(\alpha, \beta, 0) = 1$ тогда и только тогда, когда $\alpha = \beta$.

Если $t = 1$, то $R(\alpha, \beta, 0) = 1$ тогда и только тогда, когда β получается из α за один такт работы МТ.

Теперь заметим, что $R(\alpha, \beta, t) = 1$ тогда и только тогда, когда существует такая конфигурация γ, что $R(\alpha, \gamma, \lfloor t/2 \rfloor) = 1$ и $R(\gamma, \beta, t - \lfloor t/2 \rfloor) = 1$.

Это логическое условие естественно реализуется игрой двух игроков: Оптимист (игрок 1) указывает γ, Пессимист (игрок 0) выбирает один из двух интервалов, на которых нужно дальше проверять условие. В терминальных позициях $t = 0$ или $t = 1$, выигрыш Оптимиста определяется значением функции $R(\alpha, \beta, t)$. Если есть выигрывающая стратегия у Оптимиста, то по ней восстанавливается путь из α в β. И наоборот, если такой путь есть, то выигрывающая стратегия Оптимиста состоит в том, чтобы выбирать γ, следуя этому пути.

Эта игра и реализуется АМТ M'.

На начальном этапе в состояниях $Q_\vee$ (то есть ходы делает Оптимист) записывается начальная конфигурация α, целевая β с принимающим финальным состоянием (Оптимист утверждает, что МТ M заканчивает работу в конфигурации β) и время работы T (в двоичной записи). Затем АМТ M' моделирует описанную выше игру, которая рекурсивно вычисляет предикат $R(\alpha, \beta, T)$, как объяснялось выше. В каждом раунде Оптимист указывает γ, отвечающий середине текущего интервала, а Пессимист выбирает, какой из подинтервалов рассматривать далее.

Сколько времени работает такая АМТ? Количество раундов игры ограничено логарифмом времени работы T.

Поскольку память ограничена $O(S(n))$ на входах длины n, то возможных конфигураций МТ M не больше $|Q| \cdot O(S(n)) \cdot |A|^{O(S(n))}$. Время работы МТ M не больше количества конфигураций: если какая-то конфигурация повторилась, машина «зацикливается» и никогда не останавливается.

Поэтому Оптимист заведомо не выигрывает, если выбирает на начальном этапе слишком большое время T, длина двоичной записи которого больше $C' \cdot S(n)$ (здесь константа C' зависит и от размера алфавита машины M и от константы C). Поэтому для дальнейшего анализа интересен только случай $T = O(S(n))$.

Аналогично, Оптимист не выигрывает, если на каком-то ходе выбирает слишком длинную конфигурацию, длина которой больше $C \cdot S(n)$: по предположению M работает на меньшей памяти.

Если $w \in L$, у Оптимиста есть выигрывающая стратегия, которая заключается в том, чтобы записывать конфигурации, возникающие на принимающем вычислении машины M. На каждом раунде Оптимисту нужно не больше $C \cdot S(n)$ времени, чтобы записать очередную конфигурацию γ, а Пессимисту нужно $O(1)$ времени, чтобы выбрать один из двух интервалов. Всего получается альтернирующее время $O(S(n)^2)$.

Если $w \notin L$, то у Оптимиста нет выигрывающей стратегии, так как вычисление M заканчивается в отвергающей конфигурации. Поэтому Пессимист всегда может выбрать один из двух интервалов, на котором предикат $R(\alpha, \beta, t)$ равен 0. □

Комбинацией этих двух теорем получается характеризация класса PSPACE, о которой говорилось выше.

Теорема 2.3. PSPACE = AltP.

2.4. Класс PSPACE и PSPACE-полные задачи

В классе PSPACE много полных задач и среди них значительную долю составляют задачи решения игр.

Простейший и не очень интересный пример получается пересказом предыдущих общих теорем.

Пример 2 (Решение AMT-игры, ATMgame)**.** Даны AMT M, граница времени T в унарном представлении (то есть слово из T единичек), слово w. Нужно узнать, принимает ли M за время T входное слово w.

Здесь предполагается, что если какое-то вычисление длится дольше T тактов, оно обрывается в отвергающем состоянии.

Утверждение 2.4. *Решение AMT-игры является* PSPACE*-полной задачей.*

Доказательство. Пусть $L \in$ PSPACE и машина M распознаёт язык L на памяти $S(n) = \mathrm{poly}(n)$. Теорема 2.2 говорит, что существует AMT M', которая распознаёт L за альтернирующее время $T = O(S(n)^2) = \mathrm{poly}(n)$. Поэтому сводимость языка L к ATMgame устроена очень просто: слову w сопоставляется тройка (M', T, w).

Нужно ещё проверить, что ATMgame $\in$ PSPACE. Для этого построим AMT, которая принимает язык ATMgame. Эта AMT на входе (M, T, w) моделирует работу M на входе w за время T обычным образом (тут нужно вспомнить конструкцию универсальной машины Тьюринга, добавить к ней счётчик ходов и расклассифицировать состояния так же, как в M). □

Дальнейшие примеры полных в PSPACE задач можно получать сводимостями.

Пример 3 (Значение квантифицированной булевой формулы, TQBF)**.** Нужно вычислить значение формулы

$$\mathsf{Q}_1 \mathsf{Q}_2 \ldots \mathsf{Q}_n \varphi(x_1, \ldots, x_n), \tag{2.1}$$

где φ — булева формула от переменных $x_1, \ldots, x_n$, а $\mathsf{Q}_1\mathsf{Q}_2\ldots\mathsf{Q}_n$ — последовательность кванторов, $\mathsf{Q}_i \in \{\forall, \exists\}$.

Можно описать эту задачу в терминах игр, обобщая пример 1. Игроки присваивают значения булевым переменным $x_1, \ldots, x_n$: игрок 0 присваивает значения переменным x_i, для которых $\mathsf{Q}_i = \forall$; игрок 1 — переменным x_i, для которых $\mathsf{Q}_i = \exists$.

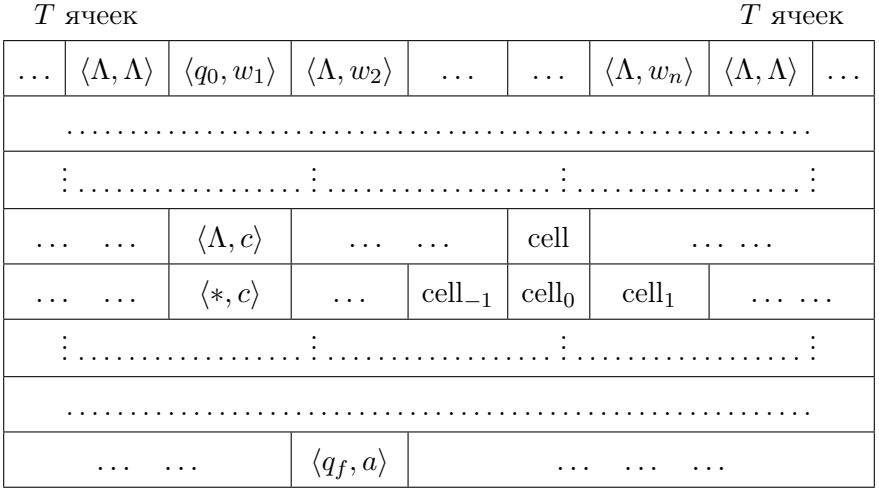
Игра заканчивается, когда всем переменным присвоены значения. Критерий выигрыша игрока 1: $\varphi(x_1, \ldots, x_n) = 1$.

Теорема 2.5. TQBF PSPACE-*полна.*

Доказательство. Принадлежность TQBF классу AltP очевидна: требуемая AMT присваивает значения переменным по очереди в состояниях, соответствующих кванторам; затем происходит (детерминированное) вычисление значения формулы.

Докажем PSPACE-трудность построением сводимости ATMgame $\leq_p$ TQBF. Эта сводимость устроена аналогично сводимости в теореме Кука–Левина об NP-полноте задачи выполнимости. Пусть $(M', 1^T, w)$ — вход задачи ATMgame. Для простоты дальнейших рассуждений преобразуем машину M' к такому виду M, что экзистенциальные и универсальные состояния обязательно чередуются на каждом такте работы. Это очень легко сделать, добавив для каждого состояния q новое «ленивое» состояние $\tilde{q}$, которое имеет другой класс, чем q, и в котором разрешён лишь тривиальный переход (ничего не изменяется, кроме состояния). Каждому переходу $(q_1, a_1, q_2, a_2, d) \in \delta'$ машины M', в котором классы q_1 и q_2 одинаковые, сопоставим два перехода $(q_1, a_1, \tilde{q}_2, a_2, d)$ и $(\tilde{q}_2, a_2, q_2, a_2, 0)$. Остальные переходы добавляем в отношение переходов δ машины M без изменений. Легко видеть, что после такого преобразования результат применения машины к любому слову не меняется, но классы состояний теперь чередуются на каждом такте.

Рассмотрим пространственно-временну́ю диаграмму вычисления машины M за время T на входе w:



В i-й строке диаграммы закодирована конфигурация машины после i тактов работы. Каждая клетка таблицы — двоичное слово, которое кодирует ячейку МТ

в некоторый момент времени. Столбцы отвечают одной и той же ячейке ленты. Помимо ячеек входного слова используются T ячеек слева и T ячеек справа (дальше головка машины за время T не доберётся). Код ячейки описывает пару q, a, если над данной ячейкой находится головка и пару Λ, a, если головка в другом месте. Длина кода $N = O(\log|Q| + \log|A|)$.

Договоримся также, что если состояние машины финальное, то соответствующая строка таблицы при необходимости повторяется (машина может закончить работу и быстрее, чем за время T).

Всего в диаграмме записано $(T+1)(n+2T)N$ битов. Из них коды первой строки фиксированы (булевы константы), а остальные $m = T(n+2T)N$ битов будут переменными x_i для задачи TQBF. Кванторы расставлены в чередующемся порядке по строкам, для переменных первой строки кванторы отвечают классу начального состояния: если $q_0 \in Q_\vee$, то все кванторы $\exists$; иначе все кванторы $\forall$.

Формула $\mathrm{Accept}(x_1, \ldots, x_m)$ равна 1 тогда и только тогда, когда диаграмма описывает корректное принимающее вычисление: все переходы согласованы с отношением переходов и последнее состояние принимающее. Можно записать такую формулу полиномиального размера (от длины входа задачи ATMgame): корректность переходов задаётся конъюнкцией T формул, которые проверяют

– согласованность четвёрок cell_{-1}, cell_0, cell_1, cell с отношением переходов, если над верхней ячейкой стоит головка (код вида $\langle q, a\rangle$, $q \neq \Lambda$);

– совпадение символов на ленте в тех парах ячеек, которые соседствуют по вертикали и над верхней не стоит головка (код вида $\langle \Lambda, c\rangle$);

и формулы, проверяющей принадлежность состояния q_f из последней строки множеству принимающих состояний.

В первом случае необходимо проверить, что среди трёх кодов $\mathrm{cell}_{-1} = (s_{-1}, a_{-1})$, $\mathrm{cell}_0 = (s_0, a')$, $\mathrm{cell}_1 = (s_1, a_1)$ ровно один содержит непустую первую компоненту (головка машины перемещается не более, чем на одну ячейку), а также проверить, что пятёрка (a, q, a', s_d, d) принадлежит отношению переходов машины.

Проследив за построением, нетрудно убедиться, что полученная в результате TQBF будет истинна тогда и только тогда, машина M принимает слово w за время T. □

Упражнение 4. Восстановите детали построения формулы Accept указанного в доказательстве размера.

Задача TQBF удобна для построения сводимостей, она для PSPACE-полных задач играет ту же роль, что задача выполнимости SAT для NP-полных задач.

Пример 4 (TQCNF)**.** Это частный случай TQBF, когда формула является КНФ (конъюнкцией дизъюнкций литералов, то есть переменных или отрицаний переменных).

Упражнение 5. Докажите, что TQCNF PSPACE-полна.

Указание. Используйте тот же способ, что и при доказательстве NP-полноты выполнимости КНФ. А именно, для построения сводимости произвольных квантифицированных формул к КНФ, выразите значение формулы φ в виде

$$\varphi(x) = \exists y C(x, y),$$

где $C(x, y)$ — КНФ, а количество переменных y равно размеру формулы φ плюс 1 (размер формулы — количество пропозициональных связок в ней). □

Пример 5 («География»)**.** Игра задана конечным ориентированным графом G и начальной вершиной v_0. В начале игры в v_0 помещается фишка. Игроки по очереди двигают фишку в соседнюю по графу вершину. Вершины не должны повторяться. Кто не может сделать ход, тот проиграл.

PSPACE-полнота географии описана во многих учебниках по теории сложности. Мы рассмотрим другой пример.

Пример 6 («Вершинные кегли», NODE KAYLES)**.** Игра задана конечным неориентированным графом G. Игроки по очереди закрашивают вершины. На каждом ходу игрок должен выбрать незакрашенную вершину, у которой все соседи также не закрашены, и покрасить. Кто не может сделать ход, тот проиграл.

Другими словами, игроки строят независимое множество в графе, добавляя на каждом ходу по одной вершине. Тот, кому не удаётся расширить независимое множество, проиграл.

Ещё одна интерпретация игры оправдывает название. Пусть в вершинах графа стоят кегли. Игрок может метнуть биту в одну из вершин и выбить все кегли в этой вершине и её соседях. Требуется на каждом ходу выбивать хотя бы одну кеглю.

Теорема 2.6 ([16])**.** *Решение игры «вершинные кегли»* PSPACE*-полно.*

Доказательство. Проверка принадлежности классу AltP делается легко и оставляется читателю в качестве самостоятельного упражнения.

Доказательство PSPACE-трудности состоит в построении сводимости TQCNF к вершинным кеглям.

Будем считать без ограничения общности, что квантифицированная КНФ от переменных $x_1, \ldots, x_n$ имеет вид

$$\Phi = \exists x_n \forall x_{n-1} \ldots \exists x_1 \bigwedge_{j=1}^{m} D_j,$$

где D_j — дизъюнкты (дизъюнкции литералов), причём $D_1 = x_1 \vee \neg x_1$. Последнего легко добиться, поскольку добавление тавтологичного дизъюнкта ничего не меняет в оценке формулы. Аналогичным приёмом и добавлением при необходимости новой переменной можно добиться, чтобы первым стоял квантор существования.

Теперь построим по формуле Φ граф G_Φ. Вершины этого графа состоят из блоков X_i, $0 \leqslant i \leqslant n$. Вершины из блока X_0 (блок дизъюнктов) обозначим $d_1, \ldots, d_m$, их столько же, сколько в формуле Φ дизъюнктов. Вершины блока X_i, $i > 0$, содержат пару вершин с именами x_i, $\neg x_i$ (литералы с переменной x_i), а также группу вспомогательных вершин $y_{i,0}, \ldots, y_{i,i-1}$. Всего в блок X_i входит $2 + i$ вершин.

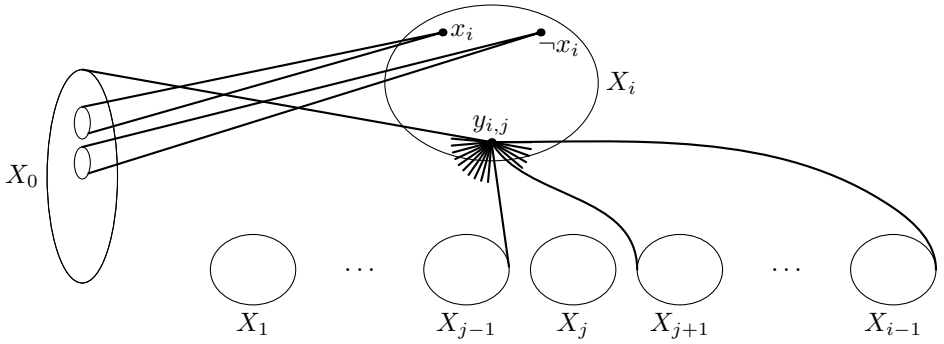


Рис. 2.5: Граф G_Φ

Теперь опишем рёбра графа G_Φ (см. рисунок 2.5). Блоки являются кликами, то есть каждая пара вершин в одном блоке соединена ребром. Вершина блока дизъюнктов соединена в точности с вершинами, помеченными литералами, входящими в этот дизъюнкт:

$$(d_j, \ell_i) \in E(G_\Phi) \;\Leftrightarrow\; \text{дизъюнкт } D_j \text{ содержит литерал } \ell_i.$$

Вспомогательная вершина $y_{i,j}$ соединена со всеми вершинами из блоков X_k при $0 \leqslant k < i$, $k \neq j$, и только с ними.

Нетрудно видеть, что граф G_Φ может быть построен по формуле Φ за полиномиальное время. Поэтому осталось доказать, что $\Phi \mapsto G_\Phi$ задаёт сводимость TQCNF к вершинным кеглям.

Первое наблюдение состоит в том, что в любой партии вершинных кеглей в каждый блок возможно не более одного хода, поскольку все блоки — клики.

Второе наблюдение состоит в том, что порядок ходов в партии на графе G_Φ в значительной степени предопределён.

Назовём партию *правильной*, если игроки закрашивают вершины из блоков X_n, X_{n-1} и так далее в нисходящем порядке; причём эти вершины помечены литералами (вспомогательные вершины не закрашиваются).

Отклонение от правильной партии наказуемо проигрышем. Действительно, пусть начало партии правильное, выбраны вершины-литералы из блоков $X_n, \ldots, X_{t+1}$, а на следующем ходу игрок отклонился от правильной партии. Рассмотрим возможные варианты.

Игрок A закрасил вершину в блоке X_i, $i < t$. Тогда вспомогательная вершина $y_{t,i}$ из блока X_t ещё не соединена ни с одной из закрашенных вершин. Противник игрока A закрашивает эту вершину. После этого хода выбиты все вершины: в блоках $X_n, \ldots, X_{t+1}$ на правильном начале партии, в блоке X_i отклоняющимся ходом, а во всех остальных блоках — ответом противника. Игра заканчивается, игрок A проигрывает.

Игрок A закрасил вспомогательную вершину $y_{t,i}$, $i < t$, в блоке X_t. После этого хода вершины не выбиты только вершины из блока X_i, причём в этом блоке есть невыбитые вершины. При $i > 0$ все вершины этого блока не выбиты. Противник

ходит в одну из них и выбивает все вершины. Игра заканчивается, игрок A проигрывает.

Если же $i = 0$, то в блоке дизъюнктов благодаря сделанному предположению $D_1 = x_1 \vee \neg x_1$ о формуле Φ есть невыбитая вершина d_1. Противник закрашивает её и выбивает все вершины. Поэтому игрок A проигрывает и в этом случае.

Теперь докажем, что если у игрока 1 в игре на формуле Φ есть выигрывающая стратегия, то она есть и у игрока F, который ходит первым в вершинных кеглях. Этот игрок следует стратегии игрока 1, разыгрывая правильную партию (здесь существенно, что формула начинается с квантора существования, то есть игра на формуле начинается ходом игрока 1). Если в игре на формуле переменной x_t присваивается значение 1, то игрок F выбирает литерал x_t; в противном случае он выбирает литерал $\neg x_t$.

Как только противник отклоняется от правильной партии, игрок F его наказывает, как объяснялось выше, и выигрывает. Поэтому игра продолжается до тех пор, пока не закрашены вершины во всех блоках $X_n, \ldots, X_1$. По сделанному о формуле Φ предположению, последним ходит игрок F.

Стратегия игрока 1 в игре на формуле выигрывающая. Поэтому при выбранных значениях переменных каждый дизъюнкт в КНФ обращается в 1. Но это означает, что в вершинных кеглях в блоке дизъюнктов выбиты все вершины. Противнику некуда ходить, игрок F выиграл.

Аналогично доказываем, что если у игрока 0 в игре на формуле Φ есть выигрывающая стратегия, то она есть и у игрока S, который ходит вторым в вершинных кеглях. Игрок S следует стратегии игрока 0 в игре на формуле, разыгрывая правильную партию. Если F отклоняется от правильной партии, то S его наказывает и выигрывает.

Посмотрим на тот момент, когда выбраны вершины во всех блоках $X_n, \ldots, X_1$. Последним, как мы помним, ходил игрок F. Поскольку игрок S следовал выигрывающей стратегии игрока 0 в игре на формуле, при выбранных значениях переменных хотя бы один дизъюнкт D_j обращается в 0. То есть, вершина d_j в блоке X_0 ещё не выбита. Игрок S закрашивает её и выигрывает. □

Тема 3

Беспристрастные игры

Результаты о трудности решения игр, которые приведены выше, относятся к довольно сложным играм. Скажем, NODE KAYLES трудна для очень больших графов, да ещё и очень сложно устроенных.

Есть много приёмов решения игр. Например, симметричные стратегии. Проиллюстрируем их на примере исходной игры KAYLES. В этой игре есть ряд кеглей. Игроки ходят по очереди. Игрок может выбить либо одну кеглю, либо две, но стоящие рядом (см. рис. 3.1). Если игрок не может сделать ход, он проиграл.

В игре KAYLES с любым положительным количеством кеглей выигрывающая стратегия есть у того игрока, который делает первый ход. Своим первым ходом этот игрок разделяет кегли на две равные группы, в зависимости от чётности числа кеглей выбивая либо одну центральную кеглю, либо две.

После этого стратегия этого игрока состоит в том, чтобы повторять ход противника в другой группе кеглей. Это всегда возможно, так как положения кеглей в обеих группах будут одинаковыми при такой стратегии.

Поскольку первый игрок всегда может сделать ход, придерживаясь этой стратегии, он выигрывает.

Симметричные стратегии не всегда применимы. Рассмотрим близкий пример: игру NODE KAYLES на графе, который является путём. Если говорить о кеглях, теперь мы требуем, чтобы каждым ударом игрок выбивал кеглю и её соседей. Таким образом, если игрок бьёт по крайней кегле, то он выбивает две кегли, а если по любой другой — три (см. рис. 3.2).

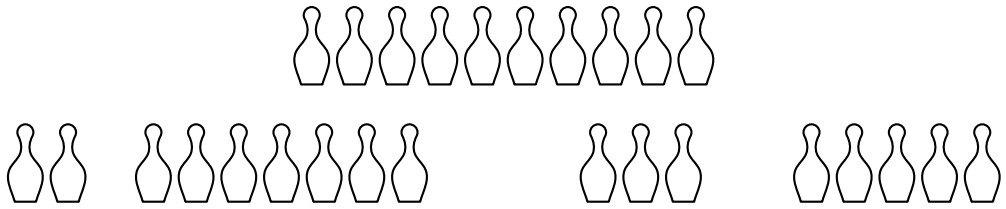


Рис. 3.1: Возможные ходы в игре KAYLES

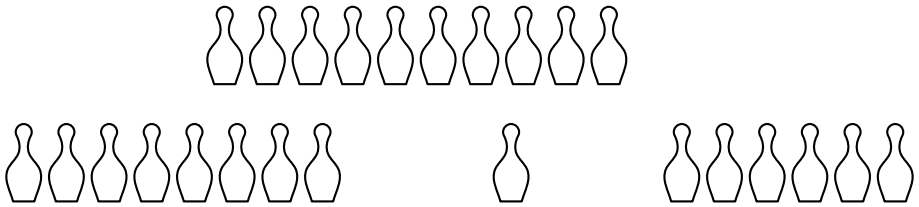


Рис. 3.2: Возможные ходы в игре NODE KAYLES на графе-пути

Если количество кеглей нечётно, первый игрок опять имеет выигрывающую стратегию того же вида: первым ходом выбивает три центральные кегли и далее отвечает симметрично.

Но при чётном количестве кеглей эта стратегия неприменима.

Выписывая все позиции, можно решить игру NODE KAYLES на путях небольшого размера.

Упражнение 6. Проверьте, что для 6 кеглей выигрывающая стратегия есть у того игрока, который ходит первым; для 8 кеглей — у того, кто ходит вторым.

Однако для 2016 кеглей решить эту игру «на бумажке» уже невозможно: позиций слишком много.

Количество позиций для игры NODE KAYLES с $6n$ кеглями не меньше, чем количество разбиений числа n в сумму положительных слагаемых.

Упражнение 7. Проверьте это утверждение.

Асимптотика чисел разбиений $p(n)$ известна: $p(n) = e^{\Omega(\sqrt{n})}$. Таков же порядок роста количества позиций в игре NODE KAYLES с $6n$ кеглями.

Для 2016 кеглей это количество ещё доступно для прямого компьютерного вычисления на графе всех возможных позиций. Проделав такое вычисление, можно убедиться, что выигрывающая стратегия есть у того, кто ходит первым.

Но, скажем, для 34402 кеглей прямое компьютерное вычисление представляется невозможным. Тем не менее, можно решить такую игру (не на бумажке, а с помощью компьютерных вычислений). В ней выигрывающая стратегия есть у того, кто ходит вторым.

Сокращение перебора, которое позволяет решать такие игры, основано на так называемых *ним-функциях* или функциях Шпрага–Гранди.

3.1. Беспристрастные игры: оценка позиций и ним-функция

Игры NODE KAYLES и KAYLES являются примерами так называемых *беспристрастных игр*. В таких играх ходы игроков чередуются и в каждой позиции множество возможных ходов одинаково для обоих игроков. Это позволяет уменьшить количество анализируемых позиций вдвое.

Для беспристрастных игр на выигрыш оценка позиций несколько упрощается. Теперь позиции разделяются на N-позиции, в которых есть выигрывающая стратегия для игрока, делающего ход из данной позиции, и P-позиции, в которых есть выигрывающая стратегия для другого игрока. Мы будем также обозначать эти позиции булевыми значениями: 1 отвечает N-позициям, 0 — P-позициям.

Индуктивное правило определения оценки игры упрощается для беспристрастных игр. Поскольку игроки ходят по очереди, то делающий ход игрок имеет выигрывающую стратегию, если у него есть ход в позицию с оценкой 0 (то есть, в P-позицию). Действительно, после сделанного хода ходить будет другой. В противном случае выигрывающая стратегия есть у другого игрока.

В булевых значениях это правило записывается простой формулой.

Утверждение 3.1. *Если из позиции v есть ходы в позиции $u_1, \ldots, u_k$ с оценками $p_1, \ldots, p_k$, то оценка $p(v)$ позиции v вычисляется по формуле*

$$p(v) = \neg \bigwedge_{i=1}^{k} p(v_i) = \bigvee_{i=1}^{k} \neg p(v_i). \tag{3.1}$$

3.1.1 Ним

Классическим примером беспристрастной игры является *ним*. В этой игре есть три кучки камней. Игроки ходят по очереди. Игрок на любом ходе может взять любое количество камней из одной кучки, причём хотя бы один камень должен быть взят. Кто не может сделать ход, проиграл.

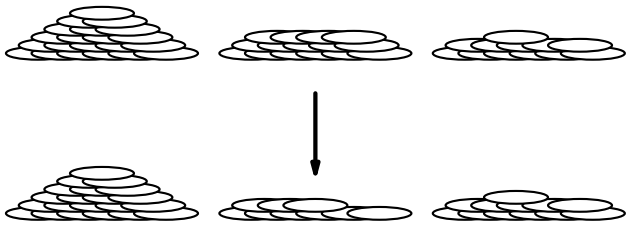


Рис. 3.3: Возможный ход в игре ним

Позиция в игре ним задаётся количеством камней в каждой из кучек, то есть тремя неотрицательными целыми числами.

Однако удобнее обобщить игру и считать, что количество кучек произвольное. Теперь позицию можно задавать набором положительных целых чисел. Есть также единственная терминальная позиция 0.

Самый простой вариант нима — игра с одной кучкой из n камней. Очевидно, что при $n > 0$ выигрывает делающий ход игрок (он забирает все камни из кучки), а при $n = 0$ — другой (терминальные позиции беспристрастных игр всегда P-позиции, так как в них нет возможности сделать ход).

3.1.2 Ним-функция

Оказывается, что по существу любая беспристрастная игра «совпадает» с нимом на одной кучке камней.

В ниме с одной кучкой из позиции с n камнями возможны ходы в позиции с $n-1, n-2, \ldots, 0$ камнями. Вот это свойство простейшего нима мы и обобщим на все беспристрастные игры.

Рассмотрим беспристрастную игру на ациклическом графе G, в терминальных позициях которой проигрывает тот, кто должен сделать ход. Построим *ним-функцию* $S(v)$ (или функцию Шпрага–Гранди) на множестве V позиций этой игры. В терминальных позициях $S(v) = 0$.

В остальных позициях определение рекурсивное:

$$S(v) = \operatorname{mex}(\{S(u) : (vu) \in E\}),$$

где функция mex от конечного множества X натуральных чисел равна наименьшему натуральному числу, не входящему в X.

У функции mex есть два свойства, непосредственно вытекающих из определения:

(1) если среди аргументов есть 0, то $\operatorname{mex}(\cdot) > 0$;

(2) если все аргументы положительные, то $\operatorname{mex}(\cdot) = 0$.

Это напоминает правило оценки позиции в беспристрастной игре, и аналогия неслучайна.

Лемма 3.2. *Пусть $S\colon V \mapsto \mathbb{N}$ — ним-функция. Тогда P-позиции — это в точности те позиции, для которых $S(v) = 0$. Остальные позиции являются N-позициями.*

Доказательство. Используем обратную индукцию.

Для терминальных позиций утверждение очевидно. Индуктивный переход состоит в проверке равенства (3.1).

Пусть есть ход (v, u) из позиции v в P-позицию u. По индуктивному предположению $S(u) = 0$. Поэтому $S(v) > 0$, а позиция v является N-позицией.

Пусть все ходы из позиции v ведут в N-позиции. По индуктивному предположению, во всех этих позициях ним-функция положительная. Значит, значение функции mex равно 0. При этом позиция v является P-позицией. □

Если оценка позиции v равна $s > 0$, из неё возможен ход в позиции с оценками $s-1, \ldots, 0$ как и в ниме в одной кучкой камней. Это и есть та аналогия, о которой мы говорили выше.

3.1.3 Сумма игр

Польза от ним-функции состоит в том, что её в некоторых случаях легко вычислить, не решая игру прямым разбором всех возможных позиций.

Пусть есть две беспристрастные игры G_1, G_2 с множествами позиций V_1 и V_2. *Суммой игр* $G_1 \oplus G_2$ называется игра, в которой игроки параллельно играют в игры G_1 и G_2, на каждом ходу игрок должен сделать ход в одной из игр.

Формально, множество позиций в сумме игр — это пары позиций в первой и второй игре, а ход состоит в переходе от позиции (v_1, v_2) либо к позиции (v_1', v_2), либо к позиции (v_1, v_2'), где в первой игре возможен ход из позиции v_1 в позицию v_1', а во второй — из позиции v_2 в позицию v_2'.

Легко видеть, что ним с k кучками камней является суммой k экземпляров нима с одной кучкой камней.

Ним-функция для суммы игр легко вычисляется по ним-функциям слагаемых, если проигрывает тот, кто не может сделать ход.

Теорема 3.3. *Пусть $S_1 \colon V_1 \to \mathbb{N}$; $S_2 \colon V_2 \to \mathbb{N}$ — ним-функции игр G_1, G_2, в которых все терминальные позиции является P-позициями. Тогда ним-функция игры $G_1 \oplus G_2$ равна*

$$S(v_1, v_2) = S(v_1) \oplus S(v_2),$$

где $x \oplus y$ обозначает поразрядную сумму по модулю 2: число, двоичная запись которого является поразрядной суммой по модулю 2 двоичных записей чисел x и y.

В доказательстве теоремы 3.3 нам потребуются свойства поразрядного сложения по модулю 2.

Лемма 3.4. *Для поразрядного сложения по модулю 2 выполняются следующие свойства.*

1. *$x \oplus y = y \oplus x$;*
2. *если $a \neq a'$, то $a \oplus x \neq a' \oplus x$ для любого x;*
3. *если $x < a \oplus b$, то либо $x = a' \oplus b$, $a' < a$; либо $x = a \oplus b'$, $b' < b$.*

Доказательство. Первое свойство очевидно из определения.

Если поразрядно прибавить к двум разным двоичным строкам одну и ту же строку, полученные суммы будут различаться в тех же самых позициях, что и исходные строки. Отсюда следует второе свойство.

Теперь докажем третье свойство. В самом старшем разряде, в котором x отличается от $a \oplus b$, в двоичной записи числа x стоит 0, а в двоичной записи $a \oplus b$ стоит 1. Поэтому в двоичных записях чисел a, b в этом разряде ровно одна единица. Считаем без ограничения общности, что в этом разряде запись a содержит 1, а запись b содержит 0.

Построим число a', поместив в этот разряд 0, а в меньших разрядах поставим такие значения, чтобы выполнялось равенство $x = a' \oplus b$. Число a' однозначно задаётся этими условиями и числами x, b, как показано на диаграмме ниже:

a:	a_1	1	a_2
b:	b_1	0	b_2
x:	x_1	0	x_2
a':	a_1	0	a_2'

По правилу сравнения чисел $a' < a$, что и требовалось. □

Доказательство теоремы 3.3. Опять обратная индукция. Для терминальных позиций утверждение теоремы очевидно, так как $0 \oplus 0 = 0$.

Теперь проверим, что число $S(v_1) \oplus S(v_2)$ отличается от всех чисел $S(v_1') \oplus S(v_2)$; $S(v_1) \oplus S(v_2')$, где из v_1 есть ход в v_1' в игре G_1, а из v_2 есть ход в v_2' в игре G_2.

По определению ним-функции $S(v_1') \neq S(v_1)$; $S(v_2') \neq S(v_2)$. Поэтому достаточно применить второе свойство из леммы 3.4.

Осталось доказать, что $S(v_1) \oplus S(v_2)$ — наименьшее из чисел, которые отличаются от всех чисел $S(v_1') \oplus S(v_2)$; $S(v_1) \oplus S(v_2')$. Для этого используем третье свойство из леммы 3.4.

Пусть $k < S(v_1) \oplus S(v_2)$. Тогда по указанному свойству либо $k = s_1 \oplus S(v_2)$, $s_1 < S(v_1)$; либо $k = S(v_1) \oplus s_2$, $s_2 < S(v_2)$. В обоих случаях из свойств ним-функции одной из двух игр следует, что в первом случае $s_1 = S(v_1')$, а во втором $s_2 = S(v_2')$. Поэтому $k = S(v_1') \oplus S(v_2)$ или $k = S(v_1) \oplus S(v_2')$, причём в первом случае в первой игре есть ход из v_1 в v_1', а во втором случае во второй игре есть ход из v_2 в v_2'. □

Уже для классического нима — с тремя кучками — множество P-позиций устроено крайне неочевидно. Не очень понятно, почему тройка $(53, 46, 27)$ — это P-позиция, а $(60, 46, 26)$ — это N-позиция.

Однако теорема 3.3 даёт полиномиальный алгоритм решения нима, даже если количества камней в кучках заданы в двоичной записи.

Эту же теорему можно применить для анализа игры NODE KAYLES на графе-пути. Позиция такой игры задаётся количеством кеглей в группах стоящих подряд кеглей. Нетрудно видеть, что игра на двух группах кеглей из n и m штук есть сумма игр на отрезке из n кеглей и отрезке из m кеглей. Отсюда получаем рекуррентное соотношение для ним-функции $K(n)$ игры NODE KAYLES на графе-пути из n вершин:

$$\begin{aligned} &K(n) = \operatorname{mex}\big(K(x) \oplus K(y) : x = n-2, y = 0 \ \vee \ x + y = n - 3\big), \\ &K(0) = 0, \ K(1) = 1, \ K(2) = 1. \end{aligned} \tag{3.2}$$

Решение такой рекурренты выглядит трудным делом. Однако, если выполнить вычисления для достаточно большого диапазона значений n, можно заметить, что ним-функция $K(n)$ становится периодической. В таблице приведены значения $K(n)$ до $n = 137$.

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$K(n)$	0	1	1	2	0	3	1	1	0	3	3	2	2	4	0	5	2
$K(17+n)$	2	3	3	0	1	1	3	0	2	1	1	0	4	5	2	7	4
$K(34+n)$	0	1	1	2	0	3	1	1	0	3	3	2	2	4	4	5	5
$K(51+n)$	2	3	3	0	1	1	3	0	2	1	1	0	4	5	3	7	4
$K(68+n)$	8	1	1	2	0	3	1	1	0	3	3	2	2	4	4	5	5
$K(85+n)$	9	3	3	0	1	1	3	0	2	1	1	0	4	5	3	7	4
$K(102+n)$	8	1	1	2	0	3	1	1	0	3	3	2	2	4	4	5	5
$K(119+n)$	9	3	3	0	1	1	3	0	2	1	1	0	4	5	3	7	4
$K(136+n)$	8	1															

Этих вычислений уже достаточно, чтобы утверждать периодичность последовательности $K(n)$.

Задача 3.1. В предположении, что таблица значений $K(n)$, $0 \leqslant n \leqslant 137$, составлена правильно, докажите периодичность $K(n)$ с предпериодом 68 и периодом 34.

Периодичность последовательности $K(n)$ позволяет решить игру NODE KAYLES на графе-пути из n вершин за полиномиальное время, даже если n задано двоичной записью.

3.2. Игры вычитания

Ещё один пример игр, которые решаются столь же эффективно, как ним и NODE KAYLES на графе-пути из n вершин, дают игры вычитания с конечным множеством разрешённых разностей.

Игра вычитания задаётся множеством разрешённых разностей $\mathcal{D} \subseteq \mathbb{N} \setminus \{0\}$ и количеством камней n. Игроки ходят по очереди. Игрок обязан взять такое количество камней, которое принадлежит множеству разрешённых разностей. Кто не может сделать ход, проиграл.

Игры вычитания также являются примером беспристрастных игр. В случае $\mathcal{D} = \mathbb{N} \setminus \{0\}$ игра вычитания совпадает с нимом на одной кучке камней. Особенно проста игра с $\mathcal{D} = \{1\}$.

Упражнение 8. Вычислите ним-функцию для игры вычитания с $\mathcal{D} = \{1\}$ и найдите множество P-позиций.

Но и игра с $\mathcal{D} = \{1, 2\}$ не слишком трудна. Вычислим несколько первых значений оценки позиции, пользуясь рекурсивным правилом (3.1).

n	0	1	2	3	4	5
$\mathrm{val}(\{1,2\}; n)$	0	1	1	0	1	1

Ясно, что далее последовательность оценок позиции будет периодически повторяться: длина периода 3 больше величины разрешённых разностей.

Для игры вычитания с $\mathcal{D} = \{2, 5\}$ получается период 7:

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$\mathrm{val}(\{2,5\}, n)$	0	0	1	1	0	1	1	0	0	1	1	0	1	1

Разумеется, не всегда последовательность оценок будет периодической.[1)]

Задача 3.2. Постройте пример такого множества $\mathcal{D}$, что оценки позиций в игре вычитания с множеством разрешённых разностей $\mathcal{D}$ не образуют периодическую последовательность (даже с предпериодом).

Однако для конечных множеств периодичность всегда имеет место.

[1)] Для краткости мы всюду называем периодическими те последовательности, которые становятся периодическими, начиная с некоторого места.

Теорема 3.5. *Если множество $\mathcal{D}$ конечно, то последовательность* $\mathrm{val}(\mathcal{D}, n)$ *оценок в игре вычитания будет периодической. Длины периода и предпериода ограничены* 2^A*, где* $A = \max(i : i \in \mathcal{D})$.

Доказательство. Поскольку на каждом ходе можно взять не более A камней, оценка позиции $\mathrm{val}(\mathcal{D}, n)$ зависит только от оценок позиций $\mathrm{val}(\mathcal{D}, n-1), \ldots, \mathrm{val}(\mathcal{D}, n-A)$.

Определим последовательность A-мерных булевых векторов v_i, где

$$v_{i,j} = \mathrm{val}(\mathcal{D}, Ai + j), \quad 0 \leqslant j < A.$$

Тогда $v_{i+1} = F(v_i)$, где $F\colon \mathbb{B}^A \to \mathbb{B}^A$ — отображение множества A-мерных булевых векторов в себя.

Как только $v_k = v_{k+T}$, эта последовательность векторов становится периодической. Но рано или поздно векторы в последовательности обязаны повториться: всего таких векторов не больше, чем 2^A. □

Отсюда получаем простой полиномиальный алгоритм решения игры вычитания с фиксированным множеством разрешённых разностей: нужно найти период, что требует $O(1)$ операций; построить таблицу периода и предпериода; и далее для любого n делить $n-\ell$ на p с остатком (здесь ℓ длина предпериода, а p — длина периода.

Однако неизвестно общего полиномиального алгоритма решения игр вычитания (входом является и множество разрешённых разностей, и число камней). Вполне возможно, что такая задача алгоритмически трудна.

3.3. Многомерные игры вычитания

Мы покажем трудность многомерного обобщения игр вычитания (несколько кучек камней). В этом варианте множество позиций — d-мерные векторы с неотрицательными целыми координатами, а $\mathcal{D} \subseteq \mathbb{N}^d$. Ход возможен из позиции x в позицию $x - a$, $a \in \mathcal{D}$.

Потребуем, чтобы для каждого $d \in \mathcal{D}$ сумма координат была положительной (при этом некоторые координаты могут быть отрицательными). В терминах игры с камнями в кучках это означает, что можно не только забирать камни, но и перекладывать их из кучки в кучку (одним из разрешённых способов).

Положительность суммы координат гарантирует конечность любой партии. Для оценки позиции $(x_1, \ldots, x_d)$ разбором с конца нужно построить граф всех позиций с $\leqslant N$ камнями в кучке, где

$$N = \sum_{i=1}^{d} x_i.$$

Всего таких позиций

$$\binom{N + d - 1}{d - 1}.$$

Если количества камней заданы в двоичной системе, то эта величина $\Omega(2^n)$, где n — длина входа. (Мы предполагаем, что $\mathcal{D}$ конечно. В случае бесконечного $\mathcal{D}$ неочевид-

ным становится уже вопрос о сложности порождения всего множества возможных ходов.)

Итак, если количества камней заданы двоичными записями, то алгоритм решения многомерной игры разбором с конца работает за экспоненциальное время от длины входа. Можно ли его ускорить? Как мы выясним далее, не всегда.

Но начнём с простого замечания: если размерность не ограничена, то решение многомерной игры вычитания PSPACE-трудно. Чтобы это понять, нужно превратить игру NODE KAYLES в частный случай игры вычитания. Размерность такой игры равна количеству рёбер в графе G игры NODE KAYLES. Векторов в множестве разрешённых разностей столько, сколько вершин в графе G. Их координаты равны 0 или 1. Эти векторы являются характеристическими функциями *звёзд* в графе G:

$$a_e^{(v)} = \begin{cases} 1, & \text{если вершина } v \text{ является концом ребра } e, \\ 0, & \text{в противном случае.} \end{cases}$$

Начальная позиция игры: вектор $\mathbb{1}$, в котором все координаты равны 1.

После вычитания вектора $a^{(v)}$ координаты, помеченные инцидентными v рёбрами, становятся нулевыми. Поэтому вычитание вектора $a^{(u)}$, $(u, v) \in E(G)$, становится в дальнейшем невозможно.

С другой стороны, если текущая позиция

$$\mathbb{1} - \sum_{v \in X} a^{(v)}$$

и вершина u не связана ребром ни с одной вершиной из множества X, то вычитание вектора $a^{(u)}$ возможно.

Поэтому графы позиций и ходов для игры NODE KAYLES и построенной выше игры вычитания совпадают.

Тема 4

Игры на вычитания и клеточные автоматы

Как мы уже видели, решение многомерных игр вычитания — PSPACE-трудная задача.

Однако неограниченное количество кучек представляется слишком большим для любителей решать игры. Поэтому мы продолжим анализ многомерных игр вычитания, считая далее, что и размерность, и абсолютная величина координат векторов из множества разрешённых разностей $O(1)$. В одномерном случае при таком ограничении получается алгоритм, полиномиальный по длине двоичной записи числа камней в кучке.

Мы увидим, что в достаточно большой (но фиксированной) размерности это уже не так. В частности, множество P-позиций для такой игры не будет иметь периодической структуры.

Для построения таких игр мы воспользуемся описанной в [10] связью между играми вычитания и двоичными клеточными автоматами.

4.1. Клеточные автоматы

Хорошо известным примером двумерного клеточного автомата является знаменитая игра «Жизнь», изобретённая Дж. Конвеем. Но у нас будут одномерные клеточные автоматы.

Такие автоматы напоминают машины Тьюринга. Только в данном случае головки нет, а вычисление новых состояний происходит одновременно во всех ячейках.

Более формально, клеточный автомат (КА) C задаётся алфавитом A, размером окрестности r и функцией переходов $\delta\colon A^{2r+1} \to A$. Автомат работает на бесконечной в обе стороны ленте, в ячейках которой записаны символы из алфавита A. Таким образом, конфигурация автомата — это бесконечная в обе стороны последовательность символов алфавита или, что то же самое, функция $c\colon \mathbb{Z} \to A$.

На каждом такте работы КА изменяет состояние каждой ячейки в соответствии с

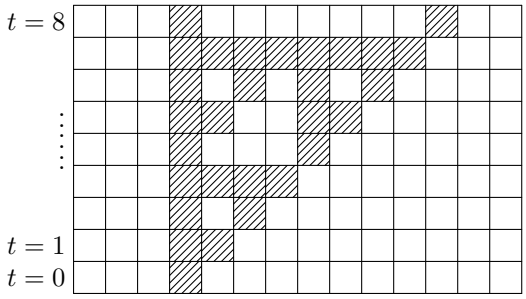


Рис. 4.1: Клеточный автомат Паскаля, **нули** заштрихованы

таблицей переходов. Новая конфигурация c' выражается через старую c по формуле

$$c'(x) = \delta\big(c(x-r), c(x-r+1), \ldots, c(x), \ldots, c(x+r-1), c(x+r)\big).$$

То есть, изменение символов локально: значение нового символа в ячейке зависит не более, чем от $2r+1$ символа в ближайших ячейках.

Как и в случае машин Тьюринга, мы требуем существования пустого символа Λ. Будем всегда предполагать, что $\delta(\Lambda, \ldots, \Lambda) = \Lambda$ («из ничего ничего не возникает»). Такое соглашение гарантирует, что если в начальной конфигурации все символы за исключением конечного числа были пустыми, то и в любой последующий момент времени лишь конечное количество символов может отличаться от пустого.

На рисунке 4.1 изображено несколько тактов работы КА с алфавитом $\{0, 1\}$, пустым символом $\Lambda = 1$ (нам будет удобнее определять пустой символ именно таким образом), размером окрестности 1 и функцией переходов $\delta(x, y, z) = 1 \oplus x \oplus y$, где $\oplus$ — сложение по модулю 2. Другими словами, новое состояние ячейки равно 1, если предыдущие состояния этой ячейки и ближайшей слева были одинаковы.

Последовательность конфигураций этого автомата по сути задаёт инвертированную таблицу чётности биномиальных коэффициентов. Поэтому он называется *автоматом Паскаля*.

Чтобы увидеть эту таблицу, нужно нарисовать конфигурации более симметрично, сдвигая ячейки. В этом случае состояние ячейки на картинке будет выражаться через состояния ближайших соседей снизу (слева и справа), как показано на рисунке 4.2. Вспоминая основное рекуррентное соотношение для биномиальных коэффициентов, получаем искомое.

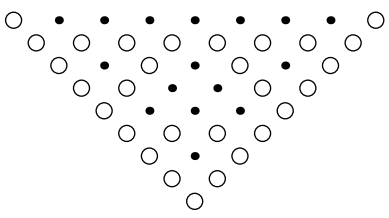


Рис. 4.2: Повёрнутая диаграмма, **нули** изображены кружочками

Задача 4.1. Докажите, что вдоль любой прямой

$$\{(x,t) : x = a_1 + b_1\lambda, t = a_2 + b_2\lambda,\ a_i, b_j > 0,\ b_1 < b_2,\ \lambda \in \mathbb{N}\}$$

последовательность конфигураций автомата Паскаля непериодична.

(Начальная конфигурация содержит ровно один 0.)

Задача 4.2. Докажите, что существует полиномиальный алгоритм, который по двоичным записям x, t находит $c(x,t)$: состояние ячейки x в момент времени t работы автомата Паскаля.

(Начальная конфигурация содержит ровно один 0.)

Параллельное вычисление «сильнее» последовательного. Поэтому нет ничего удивительного, что КА могут моделировать (детерминированные) МТ. Если посмотреть на диаграмму работы МТ (см. доказательство теоремы 2.5 на с. 22), то легко увидеть, что состояние ячейки диаграммы в момент $T+1$ является функцией состояний соседних ячеек диаграммы в момент времени T.

Поэтому работа МТ $M = (A, \Lambda, Q, q_0, Q_f, \delta)$ моделируется клеточным автоматом с алфавитом $(Q \cup \{\Lambda\}) \times A$, размером окрестности 1 и модифицированной функцией переходов Δ. Заметим, что этот модифицированный автомат обладает свойством стабильности $\Delta((\Lambda,\Lambda),(\Lambda,\Lambda),(\Lambda,\Lambda)) = (\Lambda,\Lambda)$: пустое значение первой компоненты в паре означает отсутствие головки в данной ячейке ленты МТ, поэтому символ в этой ячейке не изменяется.

Нам также важно, что работа любого КА $C = (A, r, \delta)$ моделируется работой двоичного КА $C' = (\{0,1\}, r', \delta')$ с увеличенным размером окрестности.

Идея такого моделирования очевидна — нужно кодировать символы алфавита A двоичными словами. Однако кодировку нужно подбирать аккуратно: ведь функция переходов клеточного автомата одна и та же для всех ячеек, независимо от их места в кодировке.

Считаем без ограничения общности, что алфавит $A = \{0, 1, \ldots, \ell - 1\}$. Символу a алфавита сопоставим двоичное слово

$$\varphi(a) = 1^{1+\ell-a}0^a1$$

длины $L = \ell + 2 = |A| + 2$. Договоримся, что пустой символ в алфавите A это 0. Тогда пустой символ в алфавите $\{0,1\}$ обязан быть 1, так как коды пустого символа алфавита A обязаны заполнять почти всю ленту.

Выберем размер окрестности двоичного КА равным $r' = (r+1)L$. Сформулируем основное свойство выбранной кодировки.

Лемма 4.1. *Пусть $\varphi(c)$ — посимвольная кодировка конфигурации c КА C. Тогда по $(r+1)L$-окрестности любого бита в $\varphi(c)$ однозначно восстанавливается r-окрестность того символа c, коду которого принадлежит данный бит.*

Доказательство. Если в r-окрестности символа из конфигурации c есть хотя бы один ненулевой символ, то в $(r+1)L$-окрестности любого бита из кодировки этого символа есть нулевой, то есть непустой, символ (максимально возможное расстояния между битами кодов символов из r-окрестности $L-1+(r-1)L+L-1 = (r+1)L-2$).

Процедура восстановления r-окрестности символа конфигурации c, коду которого принадлежит данный бит устроена следующим образом.

Ищем 0 в $(r+1)L$-окрестности данного бита b в конфигурации $\varphi(c)$.

Если нулей в этой окрестности нет, то по сделанному выше замечанию вся r-окрестность c состоит из одних нулей.

Возьмём ближайший к исходному биту ноль и найдём максимальную серию 0^a нулей, которая его содержит. Устройство кодирующего отображения φ гарантирует, что этим нулям предшествует $k \geqslant 1+\ell-a$ единиц. Возьмём самые правые $1+\ell-a$ из них. Получаем слово $1^{1+\ell-a}0^a1$, которое является кодом символа a конфигурации c. Поскольку длины всех кодовых слов одинаковые, позиции кодов остальных символов восстанавливаются однозначно.

Коды тех символов, которые лежат в искомой r-окрестности, попадают в $(r+1)L$-окрестность исходного бита b. □

Из этой леммы становится ясно, как нужно строить функцию переходов δ' КА C'. Она является композицией восстанавливающего отображения из $\{0,1\}^{r'}$ в A^{2r+1}, функции переходов δ, кодирующего отображения φ и проекции на тот бит, которому отвечает центральный бит аргументов функции δ'.

Замечание 4.1. В дальнейшем мы будем использовать моделирование МТ двоичным КА, которое состоит в композиции двух описанных выше переходов: от МТ к КА в произвольном алфавите и от произвольного КА к двоичному.

Существенно, что при этом комбинированном переходе можно добиться выполнения следующего свойства: начальной конфигурации МТ при работе на пустом входе отвечает начальная конфигурация $\ldots 11011 \ldots$ двоичного КА. Для этого достаточно при переходе от МТ к КА закодировать символ (Λ, Λ) диаграммы МТ символом 0 в алфавите КА, а символ (q_0, Λ) — символом 1.

При переходе к двоичному КА $0 \mapsto 1^{|A|+2}$; $1 \mapsto 1^{|A|}01$. Поэтому выполняется требуемое свойство согласования начальных конфигураций.

Здесь A — алфавит МТ, q_0 — начальное состояние МТ, а Λ — пустой символ.

4.2. Двоичные клеточные автоматы и модулярные игры

Мы хотим представить эволюцию двоичного КА $C = (\{0,1\}, r, \delta)$, начинающуюся в конфигурации $\ldots 11011 \ldots$, с помощью функции оценки позиций в некоторой многомерной вычитания.

Для этого в игре вычитания выделим две координаты (кучки камней), обозначим их x_1, x_2. Время клеточного автомата будет задаваться направлением $(1,1)$, а пространственная координата — направлением $(1,-1)$.

Количество камней в кучке неотрицательно. Поэтому таким способом мы можем задать лишь конечный кусок конфигурации КА. Это не испортит симуляцию, так как мы начинаем с конфигурации КА, в которой почти все символы пустые (то есть равны 1 по нашему соглашению).

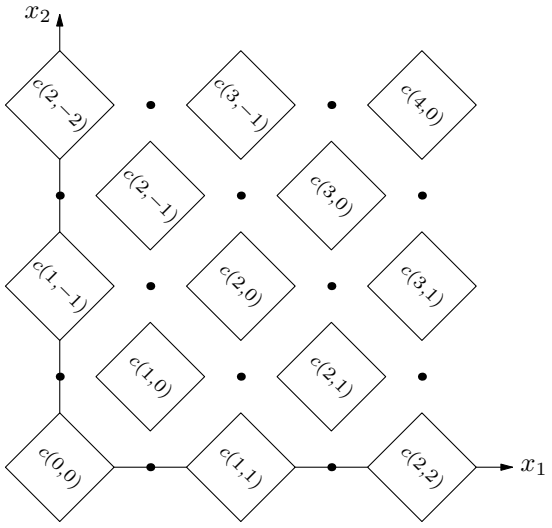


Рис. 4.3: Кодировка конфигурации КА камнями в кучках

Опишем соответствие более точно. Моменту времени t будут отвечать значения координат на прямой $x_1 + x_2 = 2Nt$, где число N будет подбираться по функции переходов автомата, причём всегда будет выполняться условие $N \geqslant r$. Номер ячейки КА будет равен $(x_1 - x_2)/2$. На рисунке 4.3 изображён фрагмент такого соответствия при $N = 1$. Там $c(t, i)$ обозначает состояние ячейки i КА в момент времени t.

Условие соответствия состоит в том, что

$$c(t, i) = \mathrm{val}(Nt + i, Nt - i) \quad \text{при } |i| \leqslant Nt,$$

при остальных значениях t, i должно выполняться $c_{t,i} = 1$.

Пусть $r = 1$ и $\delta(x, y, z) = \neg x \vee \neg y \vee \neg z$. Проверим, что такому клеточному автомату соответствует игра вычитания с разрешёнными разностями

$$\mathcal{D} = \{(2, 0),\ (1, 1),\ (0, 2)\}.$$

Действительно, вычисление значения $v(i, j)$ позиции (i, j) по правилу (3.1) даёт

$$v(i, j) = \neg v(i - 2, j) \vee \neg v(i - 1, j - 1) \vee \neg v(i, j - 2).$$

Поэтому если на прямой $x_1 + x_2 = 2t$ условие соответствия выполнено, то оно выполнено и на прямой $x_1 + x_2 = 2(t+1)$. Условие соответствия выполнено по определению на прямой $x_1 + x_2 = 0$. Значит, оно выполнено и для всех последующих моментов времени.

Мы пропустили в этом анализе краевые эффекты. Здесь можно заметить, что функция

$$[a_1, \ldots, a_k] = \bigvee_{i=1}^{k} \neg a_i$$

обладает таким свойством: $[a_1, \ldots, a_k, 1, \ldots, 1] = [a_1, \ldots, a_k]$ (от добавления в список аргументов единиц значение не изменяется).

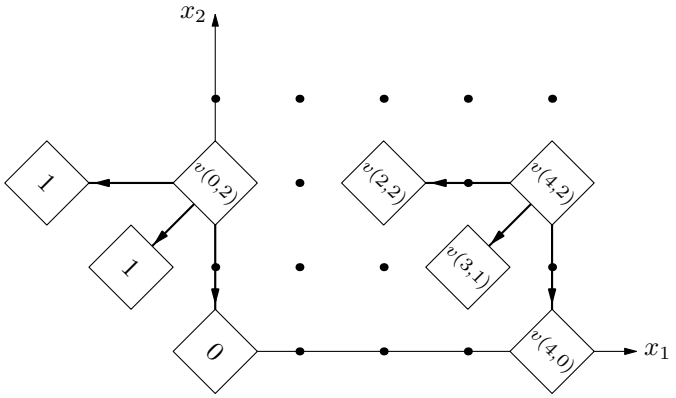


Рис. 4.4: Пример соответствия КА и игры вычитания

Мы предполагаем, что КА начинает работу в конфигурации $\dots 11011 \dots$ Поэтому к «настоящим» аргументам в формуле для оценки позиции добавляются единицы, что не влияет на результат вычисления. (См. рисунок 4.4.)

Однако такой приём работает лишь для тех автоматов, у которых функция переходов задаётся функцией $[\dots]$ от некоторого подмножества аргументов и r-окрестности.

Для других автоматов придётся применить более сложную конструкцию. Начнём с того, что расширим класс игр.

Модулярная игра вычитания отличается от обыкновенной тем, что теперь задано не одно множество разрешённых разностей, а много: $\mathcal{D}_0, \mathcal{D}_1, \dots, \mathcal{D}_{k-1}$. Ход в позиции $x \in \mathbb{N}^d$ возможен в позиции $x - a^{(j)}$, где $a^{(j)} \in \mathcal{D}_j$, а j — остаток от деления суммы координат вектора x на k.

Построим соответствие указанного выше типа между КА и двумерными модулярными играми.

Заметим, что любая булева функция выражается схемой, включающей только функции $[x_1, \dots, x_k]$ (эти функции образуют *полный базис*). Например, выразим функции стандартного полного базиса через $[\dots]$:

$$\neg x = [x],$$
$$x \vee y = [[x], [y]],$$
$$x \wedge y = [[x, y]].$$

Выпишем такую схему для функции переходов КА $\delta(x_{-r}, \dots, x_{-1}, x_0, x_1, \dots, x_r)$:

$$s_1 := [\text{аргументы}_1]$$
$$s_2 := [\text{аргументы}_2]$$
$$\dots \quad \dots$$
$$s_N := [\text{аргументы}_N]$$

Здесь аргументы в каждом присваивании s_j содержат какие-то входные переменные и какие-то s_i, $i < j$. Значение последнего присваивания s_N совпадает со значением

функции $\delta(x_{-r},\dots,x_{-1},x_0,x_1,\dots,x_r)$.

Без ограничения общности считаем, что $N \geqslant r$ (всегда можно добавить фиктивные присваивания). Тогда размер схемы N и будет параметром, который использовался выше при описании соответствия.

Мы собираемся построить такую модулярную игру вычитания, в которой модуль равен $2N$ и выполняются соотношения

$$v(Nt+i, Nt-i) = c(t,i),$$
$$v(Nt+i+j, Nt-i+j) = s_j, \quad 1 \leqslant j < N,$$

где s_j — значение j-го присваивания при вычислении значения функции

$$\delta(c(t,i-r),\dots,c(t,i),\dots,c(t,i+r))$$

по заданной схеме.

Идея такого построения понятна. Чтобы вычислить s_j, нужны какие-то входные переменные и какие-то предыдущие значения. Их положения на диаграмме кучек камней связаны конкретными векторами $a^{(j,1)},\dots,a^{(j,m)}$. Эти векторы и образуют разностное множество $\mathcal{D}_{2j}$. Как устроены разностные множества $\mathcal{D}_{2j+1}$ при нечётных остатках, неважно: они не повлияют на оценки позиций с чётной суммой координат.

В качестве примера рассмотрим автомат Паскаля, $\delta(x_{-1},x_0,x_1) = 1 \oplus x_{-1} \oplus x_0$ (см. рисунок 4.1). Выпишем схему вычисления такой функции в базисе $[\dots]$:

$$\begin{aligned}
s_1 &:= [x_0] && (s_2 = \neg x_0),\\
s_2 &:= [x_{-1}] && (s_1 = \neg x_{-1}),\\
s_3 &:= [s_1, s_2] && (s_3 = x_{-1} \vee x_0)\\
s_4 &:= [x_{-1}, x_0] && (s_4 = \neg x_{-1} \vee \neg x_0),\\
s_5 &:= [s_3, s_4] && \big(s_5 = (\neg x_{-1} \wedge \neg x_0) \vee (x_{-1} \wedge x_0) = 1 \oplus x_{-1} \oplus x_0\big).
\end{aligned}$$

Поэтому $N = 5$ для такого КА. Разностные множества легко выписываются по схеме:

$$\mathcal{D}_0 = \{(1,1),(2,2)\},\ \mathcal{D}_8 = \{(4,4),(5,3)\},\ \mathcal{D}_6 = \{(1,1),(2,2)\},$$
$$\mathcal{D}_4 = \{(3,1)\},\ \mathcal{D}_2 = \{(1,1)\}.$$

(См. рисунок 4.5.)

Заметим, что соответствие продолжается на область $|i| > Nt$ по той же причине, что и раньше: начальная конфигурация КА состоит из единиц и одного нуля, а функция переходов сохраняет единицы, как было оговорено.

4.3. Избавление от остатков

Чтобы вернуться от модулярных игр к обычным играм вычитания, нам потребуются дополнительные кучки камней в количестве $2N$ штук. Камни в этих кучках будут контролировать возможность вычитания векторов из разностных множеств $\mathcal{D}_j$.

Идея очень проста: будем рассматривать только такие позиции, в которых в каждый момент ровно в одной из этих дополнительных кучек камней лежит ровно

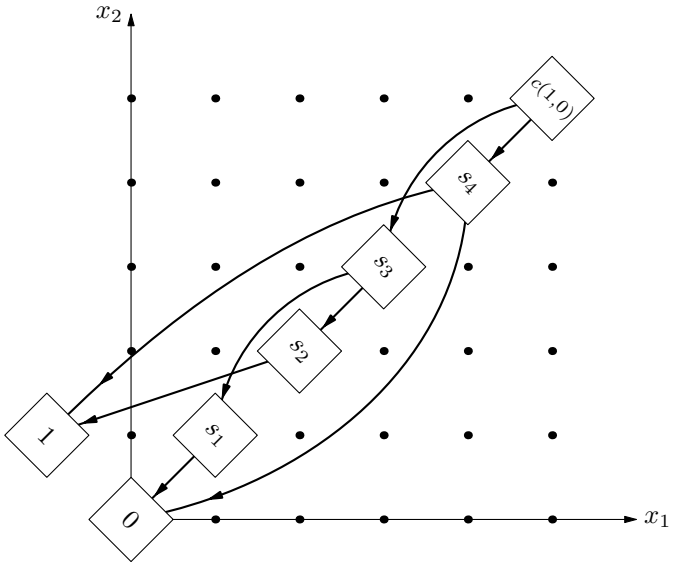


Рис. 4.5: От схемы к разностным множествам (автомат Паскаля)

один камень (условие контроля). Добиться выполнения это свойства можно, если добавлять к векторам разностного множества координаты

$$0, \dots, 0, 1, 0, \dots, -1, 0, \dots, 0.$$

Если условие контроля выполнено, то возможны лишь те ходы, которые вынимают камень из кучки, отвечающей координате 1 (и добавляют камень в кучку, отвечающую координате -1). Это позволяет имитировать модулярные разрешённые разности обычными.

Более точно, пусть двумерная игра по модулю $2N$ с множествами разрешённых разностей $\mathcal{D}_j$, $0 \leqslant j < 2N$, соответствует клеточному автомату C. Построим по ней $(2 + 2N)$-мерную игру с множеством разностей

$$\mathcal{D} = \big\{(a, 0^{2N}) + (0, 0, e_j - e_k) : a \in \mathcal{D}_j\big\},$$

где e_j — j-й координатный вектор, а $k = j - a_1 - a_2 \pmod{2N}$.

Теорема 4.2. *Пусть по двоичному КА $C = (\{0,1\}, r, \delta)$ построена игра на вычитание с множеством разрешённых разностей $\mathcal{D}$ как описано выше.*

Тогда при $|i| \leqslant Nt$ выполняется условие соответствия

$$c(t,i) = \mathrm{val}(Nt + i, Nt - i, \underbrace{0, 0, \dots, 0, 1}_{2N \text{ координат}}), \tag{4.1}$$

где $c_{t,i}$ — символ в i-й ячейке КА в момент времени t при начальной конфигурации $c(0,0) = 0$, $c(0,i) = 1$ при $i \neq 0$; $\mathrm{val}(x)$ — оценка позиции x в игре.

Доказательство. Соотношения соответствия

$$\mathrm{val}((Nt + i, Nt - i, 0^{2N}) + (0, 0, e_{2N})) = c(t,i),$$
$$\mathrm{val}((Nt + i + j, Nt - i + j, 0^{2N}) + (0, 0, e_{2j}) = s_j, \quad 1 \leqslant j < N$$

доказываются индукцией по времени (сумме первых двух координат).

База индукции очевидна, так как в начальной конфигурации есть ровно один 0 и он соответствует оценке позиции

$$(0, 0, \underbrace{0, 0, \ldots, 0, 1}_{2N \text{ координат}}).$$

Для индуктивного перехода заметим, что ходы из позиции

$$(Nt + i + j, Nt - i + j, 0^{2N}) + (0, 0, e_{2j})$$

ведут лишь в позиции такого же вида, но с меньшим значением времени. □

4.4. Трудность многомерных игр вычитания

После всей этой длительной подготовки можно, наконец, сформулировать результат о трудности игр вычитания.

Теорема 4.3. *Для некоторой константы d существует такая игра вычитания с конечным множеством разрешённых разностей $\mathcal{D} \subset \mathbb{N}^d$, что любой алгоритм решения этой игры, принимающий на вход координаты позиции в двоичной записи, работает дольше $2^{n/11}$, где n — длина входа.*

Обратите внимание, что эта теорема не зависит ни от каких гипотез теории сложности.

Заметим также, что тривиальный алгоритм решения (построить граф позиций и применить рекурсивную оценку) для такой игры работает за время $2^{O(dn)}$. Это не слишком хуже нижней оценки из теоремы 4.3.

4.4.1 Теорема об иерархии

Откуда вообще берутся абсолютные результаты о трудности задач? Основным приёмом доказательств таких результатов является *диагональный метод*, который можно считать далёким обобщением парадокса лжеца.

С помощью диагонального метода получаются доказательства алгоритмической неразрешимости, например, неразрешимость проблемы остановки машины Тьюринга. После некоторой технической работы это рассуждение можно перенести и на случай вычислений, ограниченных по времени. Сформулируем частный случай такого переноса.

Теорема 4.4 (теорема об иерархии, частный случай)**.** $\mathrm{DTIME}(2^{n/2}) \subset \mathrm{DTIME}(2^n)$.

Будем применять теорему об иерархии в доказательстве теоремы 4.3 следующим образом. Выберем какой-нибудь язык $L \in \mathrm{DTIME}(2^n) \setminus \mathrm{DTIME}(2^{n/2})$. Тогда существуют константа C и решающая машина Тьюринга M, которая на любом входе длины n работает за время $T(n) = C2^n$ и для которой $L(M) = L$. По этой машине мы построим другую машину U, которая выполняет вычисления машины M

параллельно на всех входах без существенной потери во времени работы. По машине U построим КА и соответствующую игру вычитания. Эта игра будет иметь размерность $O(1)$, так как мы строим её по конкретной машине Тьюринга U.

Чтобы узнать результат работы M на входе w, нужно посмотреть на состояние ленты машины M в достаточно поздний момент времени. Это состояние закодировано в конфигурациях двоичного КА, которым соответствуют оценки позиций в игре. Решив полученную игру, получаем ответ на вопрос $w \in L$, то есть другой алгоритм разрешения L. По выбору языка L этот алгоритм не может работать слишком быстро.

4.4.2 МТ, которая работает на всех входах сразу

Итак, пусть дана МТ M, которая работает за время $T(n)$ на любом входе длины n. Построим МТ U и опишем её работу на пустом входе (другие не понадобятся).

Работа машины U разбивается на этапы, а состояние её ленты — на зоны. Перед началом этапа каждая зона имеет вид, показанный на рисунке 4.6.

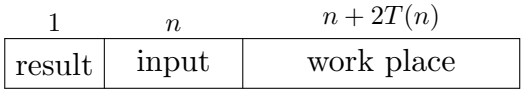


Рис. 4.6: Зона на ленте машины U

Первый блок в зоне имеет размер 1 и в нём записана 1, если работа в зоне закончена и машина M принимает вход w. Сам вход записан во втором блоке зоны. Наконец, в остальной части зоны записана конфигурация машины M. Этот блок имеет размер $n + 2T(n)$ (конфигурация заведомо не длиннее, поскольку M останавливается за время $T(n)$).

В записи конфигурации мы используем то же соглашение, что и при записи диаграммы работы MT. То есть, алфавит U должен содержать $(Q \cup \{\Lambda\}) \times A$, где Q — множество состояний M, A — алфавит M.

Перед началом этапа k на ленте машины U записаны $k-1$ зон, отвечающих первым входам w_1, w_2, …, w_{k-1}. Входные слова мы считаем двоичными, упорядочены они по длинам, а в пределах одной длины — лексикографически. На зоне i записана конфигурация работы M на входе w_i после $k-1-i$ тактов (то есть, в последней зоне записана начальная конфигурация M).

Этап k состоит в том, что U перемещается по ленте от зоны к зоне и в каждой зоне выполняет один такт работы M. После этого записывается новая зона, в которую заносится вход w_k и начальная конфигурация M при работе на этом входе. В этой конфигурации слева и справа от слова w_k оставлено свободное место размера $T(n)$, см. рисунок 4.7.

Если работа M в какой-то зоне закончена, обновляется при необходимости блок результата и далее содержимое этой зоны не изменяется.

Таким образом, машина U работает без остановки и постепенно порождает на ленте результаты работы машины M на всех входах.

1	n	$n + 2T(n)$
0	w_k	$(\Lambda, \Lambda) \dots (\Lambda, \Lambda)(q_0, w_{k,1})(\Lambda, w_{k,2}) \dots (\Lambda, w_{k,n})(\Lambda, \Lambda) \dots (\Lambda, \Lambda)$

Рис. 4.7: Свежая зона на этапе k

Нам требуется оценка времени порождения результата. Для этого, нужны оценки выполнения действий на каждом этапе, что требует уточнения действий U.

Утверждение 4.5. *Обновление конфигурации M занимает время $O(L)$, где L — размер зоны.*

Это утверждение очевидно: U движется по конфигурации пока не обнаруживает место, в котором находится головка M. После этого U обновляет окрестность этого места в соответствии с таблицей переходов M. Иногда требуется изменить блок результата. Это выполняется за пару лишних проходов по зоне (вернутся влево, переписать блок результата, пройти к следующей справа зоне).

Утверждение 4.6. *Добавление свежей зоны на этапе k занимает время $O(nT(n))$, где n — длина входа w_k.*

Доказательство. Запись блока результата требует времени $O(1)$.

Вычисление следующего в лексикографическом порядке слова занимает $O(nT(n))$ тактов работы. Для этого нужно скопировать предыдущее слово (n символов перенести на расстояние $2n + 2T(n) = O(T(n))$) и прибавить к текущему слову 1 по модулю 2^n (напомним, слова двоичные; в случае увеличения длины нужно ещё заменить дописать один 0, это требует времени $O(n)$).

Далее нужно разметить пространство $n+2T(n)$ под третий блок зоны. Напомним, что $T(n) = C2^n$, где C — некоторая константа. Чтобы выделить место $T(n)$, нужно завести счётчик, в который поместить значение $n+2T(n)$, а потом обновлять его на каждом шаге. Длина записи счётчика $\leqslant 1 + \log(n + T(n)) = O(n)$.

Для требуемой оценки времени счётчик нужно поддерживать плавающим, он должен перемещаться вслед за текущей границей отмеченного пространства. Биты записи счётчика мы представляем записанными на отдельную «дорожку». Для реализации такой записи нужно увеличить алфавит МТ и представлять себе, что в каждой ячейке записан столбец из двух символов: один символ исходного алфавита МТ, а второй — бит счётчика.

Вычисление начальной оценки требует времени $O(n \log n)$ (нужно вычислить двоичную запись длины n и скопировать её в счётчик).

Сдвиг и обновление счётчика требуют времени, пропорционального его длине, то есть $O(n)$.

Всего этап разметки потребует

$$O(n \log n + n(n + 2T(n))) = O(nT(n))$$

тактов работы.

Далее нужно поместить в центр размеченного третьего блока код начальной конфигурации машины M на входе w_k. Это также требует времени $O(nT(n))$ (копирование из блока входа в блок рабочего места с модификацией символов). □

Лемма 4.7. *При достаточно больших n результат работы $MT\,M$ на входе w длины n появляется на ленте машины U не позднее, чем после 2^{4n} тактов работы.*

Доказательство. Если считать время в этапах, то зона, отвечающая входу w, появится на первых 2^{n+1} этапах. Результат работы M на входе w появится после дополнительного количества этапов $T(n)$.

Теперь нужно оценить время выполнения этапа k. Обозначим $s = |w_k|$. Выполнение этапа потребует $O(k(s + T(s)))$ тактов для обновления существующих зон (утверждение 4.5) и $O(sT(s))$ для записи свежей зоны (утверждение 4.6). Количество зон оценивается как 2^{s+1}. Итого, получаем оценку времени выполнения этапа

$$O\big(2^{s+1}(s + T(s)) + sT(s)\big) = O(sT(s)^2).$$

Отсюда получаем итоговую оценку момента появления результата работы M на входе w:

$$O\big((2^{n+1} + T(n))nT(n)^2\big) = O(n2^{3n}) < 2^{4n}$$

при достаточно больших n. □

4.4.3 Доказательство теоремы о трудности игр вычитания

Построим по машине U соответствующий двоичный КА C_U. У него будет размер окрестности $O(1)$ (зависит от числа состояний и алфавита исходной машины M). К описанной выше конструкции нужно ещё добавить два соглашения:

1. Код пары (q_0, Λ) должен быть двоичным словом $1^\ell 01$, код пары (Λ, Λ) должен быть двоичным словом $1^{\ell+2}$. Тогда начальной конфигурации работы U на пустом входе будет соответствовать начальная конфигурация $\ldots 11011 \ldots$ автомата C_U.

2. Код пары $(\Lambda, 1)$ должен быть двоичным словом $110^{\ell-1}1$. Это гарантирует, что в коде блоков результата третий бит равен 0 лишь тогда, когда над этим битом нет головки U и машина M принимает вход из этой зоны.

По автомату C_U построим d-мерную игру вычитания с множеством разрешённых разностей $\mathcal{D}_U$. Здесь $d = O(1)$, поскольку параметр N в конструкции зависит от функции переходов δ КА C_U, а эта функция имеет $O(1)$ аргументов.

Условие согласования (4.1) позволяет по (t, i) выразить символ $c(t, i)$ на ленте C_U через оценку позиции

$$\mathrm{val}(Nt + i, Nt - i, \underbrace{0, 0, \ldots, 0, 1}_{2N \text{ координат}}).$$

Получаем алгоритм проверки принадлежности слова w (длины n) языку L, использующий процедуру решения игры C_U:

1. Определить номер зоны k, в которой машина U записывает результаты работы машины M на слове w. При выбранном порядке зон это
$$k = 2^n + \mathrm{bin}(w), \quad \mathrm{bin}(w) \text{ — число с двоичной записью } w.$$

2. Определить номер значащего бита результата в блоке результата зоны k. Это
$$i = \sum_{\ell=0}^{n-1} 2^\ell(1 + 2\ell + 2^{\ell+1}) + \mathrm{bin}(w)(1 + 2n + 2^{n+1}) + 3.$$

3. Выбрать три момента времени $t_1 = 2^{4n}$, $t_2 = 2^{4n} + 1$, $t_3 = 2^{4n} + 2$.

4. Применить процедуру решения игры $\mathcal{D}_U$ для оценки позиций
$$\mathrm{val}(Nt_0 + i, Nt - i, \underbrace{0, 0, \ldots, 0, 1}_{2N \text{ координат}}),$$
$$\mathrm{val}(Nt_1 + i, Nt - i, \underbrace{0, 0, \ldots, 0, 1}_{2N \text{ координат}}),$$
$$\mathrm{val}(Nt_2 + i, Nt - i, \underbrace{0, 0, \ldots, 0, 1}_{2N \text{ координат}}).$$

5. Если хотя бы две из этих трёх оценок дают 0, принять слово w.

Корректность этого алгоритма не вызывает сомнений. Комментариев требуют последние действия. Они связаны с тем, что мы не хотим настолько конкретизировать работу машины U, чтобы можно было точно определить, над какой позицией находится головка в момент времени t.

Однако без ограничения общности можно считать, что машина U возвращается в блок результата не раньше, чем через три такта работы (это её замедлит разве что втрое). Поэтому из выбранных трёх последовательных моментов времени голова над блоком результата будет находится не более одного. В остальные два момента в i-м бите записан результат работы M на входе w (лемма 4.7).

Размер описания позиций игры $\mathcal{D}_U$ на шаге 4 оценивается как $4n+o(n)$, то есть не больше $5n$. Если время работы алгоритма решения игры $S(n)$, где n — длина входа, то время работы описанного выше алгоритма разрешения языка L оценивается как $\mathrm{poly}(n) + 3S(5n)$. Если $S(n) < 2^{n/11}$, то получаем алгоритм разрешения языка L, работающий за время
$$< \mathrm{poly}(n) + 2^{5n/11} < 2^{n/2},$$
что противоречит выбору языка L.

Тема 5

Бесконечные игры

5.1. О детерминированности бесконечных игр

Теперь мы переходим к играм, которые длятся бесконечное количество ходов. В таких играх уже неочевидно даже существование выигрышной стратегии у одного из игроков (такие игры называются *детерминированными*).

Разберём важный пример. Игроки 0 и 1 по очереди красят числа натурального ряда (целые неотрицательные числа) в два цвета (скажем, игрок 0 красит в красный, а игрок 1 — в синий). Если уже закрашены числа от 0 до N, то на следующем ходу игрок может закрасить в свой цвет любой *конечный* отрезок из оставшихся чисел: от $N+1$ до некоторого числа N'. Хотя бы одно число закрасить нужно. На рисунке 5.1 изображено начало партии в такой игре.

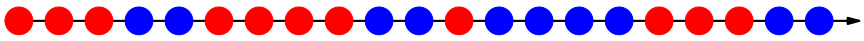


Рис. 5.1: Начало партии игры в закрашивание натурального ряда

Поскольку натуральных чисел бесконечно много, то любая партия продолжается бесконечное количество ходов. Итогом партии является разбиение множества натуральных чисел на красные (обозначим его S_0) и синие (S_1). Игрок 0 выигрывает в партии, если множество S_0 принадлежит некоторому, заранее заданному, семейству Win_0 подмножеств натурального ряда. Иначе выигрывает игрок 1.

Мы описали целый класс игр в закрашивание натурального ряда, конкретная игра задаётся семейством Win_0. Трудность игры сильно зависит от выбора семейства. В некоторых случаях решить игру несложно.

Пример 7 (Игра на сходимость синего ряда)**.** Пусть Win_0 состоит в точности из тех подмножеств X натуральных чисел, для которых ряд

$$\sum_{i \in \mathbb{N} \setminus X} \frac{1}{i+1}$$

сходится. Другими словами, сумма обратных к синим числам должна быть конечной. В некотором смысле это означает, что Красный (игрок 0) закрасил «почти все»

числа в свой цвет.

Легко видеть, что у игрока 1 (Синего) есть выигрышная стратегия. Вспомним, что гармонический ряд

$$\sum_{i \geqslant 0} \frac{1}{i+1}$$

расходится. Это означает, что для каждого N найдётся такое N', что

$$\sum_{i=N+1}^{N'} \frac{1}{i} > 1.$$

Поэтому на каждом ходе Синий может увеличить частичную сумму синих чисел по крайней мере на 1. В итоге получится синее множество натуральных чисел, сумма обратных к которым бесконечна.

Задача 5.1. [Игра на сходимость красного ряда] Решите игру на закрашивание натурального ряда, если Win_0 состоит в точности из тех подмножеств X натуральных чисел, для которых ряд

$$\sum_{i \in X} \frac{1}{i+1}$$

сходится («почти все» числа покрашены синим).

Если семейство Win_0 удовлетворяет некоторым естественным требованиям, то в игре в закрашивание натурального ряда выигрышных стратегий может не быть ни у одного из игроков.

Разделим все подмножества натурального ряда на «большие» (принадлежащие Win_0) и «маленькие» (не принадлежащие Win_0) так, чтобы выполнялись следующие (вполне естественные) свойства:

1: любое конечное множество маленькое;

2: X маленькое тогда и только тогда, когда $\mathbb{N} \setminus X$ большое;

3: если $X \subset Y$ и X большое, то и Y большое;

4: пересечение больших множеств большое.

Проверим, что если правило выигрыша в игре в закрашивание натурального ряда состоит в том, что множество красных чисел большое, то ни у одного из игроков нет выигрышной стратегии.

Прежде всего заметим, что множество красных чисел большое тогда и только тогда, когда множество синих чисел маленькое (свойство 2). Кроме того, если $X \oplus Y$ конечно (знаком $\oplus$ мы обозначаем симметрическую разность множеств), то X и Y одной величины (либо оба большие, либо оба маленькие). Более точно, если $X \cap Y$ большое, то и X, и Y также большие (свойство 3). Если $X \cap Y$ маленькое, то и $X = (X \cap Y) \cup (X \setminus Y)$, и $Y = (X \cap Y) \cup (Y \setminus X)$ маленькие, так как из свойства 1

следует, что $X \setminus Y$, $Y \setminus X$ маленькие, а из свойств 2 и 4 следует, что объединение маленьких множеств маленькое.

Поэтому каждый игроков может имитировать стратегию второго с переменой размеров красного и синего множеств.

Предположим, что выигрышная стратегия s существует для Синего. Рассмотрим такую стратегию Красного: посмотреть, какой ход предписывает Синему стратегия s в ответ на первый ход Красного, закрашивающий только 0. Пусть Синий должен закрасить N_0 чисел. Сделать ход, который закрашивает числа от 0 до N_0. Далее следовать стратегии s Синего. Пример такой перемены стратегии показан на рисунке 5.2.

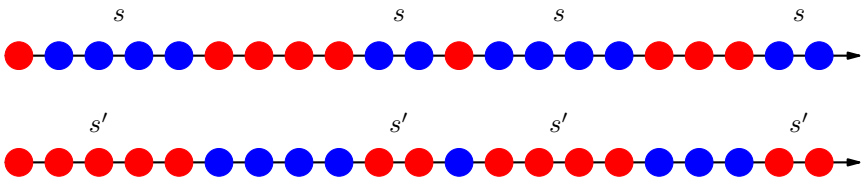


Рис. 5.2: Красный перехватывает стратегию Синего

Аналогично и Синий может перехватить стратегию Красного. Пусть стратегия s Красного предписывает первым ходом закрасить числа от 0 до N_0, а в ответ на ход Синего, закрашивающий отрезок $[N_0 + 1, N_1]$, закрасить числа от $N_1 + 1$ до $N_2(N_1)$. Тогда перехватывающая стратегия s' Синего состоит в том, чтобы в ответ на ход $[0, X]$ Красного закрасить отрезок $[X + 1, N_2(X - N_0)]$ отрезок, если $X > N_0$, и отрезок $[X + 1, N_2(N_0 + 1)]$, если $X \leqslant N_0$. Далее следовать стратегии Красного. Пример такой перемены стратегии показан на рисунке 5.3.

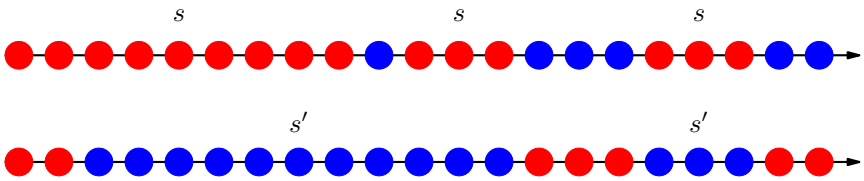


Рис. 5.3: Синий перехватывает стратегию Красного

В обоих случаях перехватывающая стратегия гарантирует, что красное множество, возникающее в партии, в которой один игрок придерживается стратегии s, лишь на конечное множество отличается от синего множества, возникающего в партии, в которой другой игрок придерживается перехватывающей стратегии s'. Поэтому если s — выигрывающая стратегия для одного игрока, то перехватывающая стратегия s' будет выигрывающей для другого.

Ясно, что у обоих игроков выигрывающей стратегии быть не может: кто выигрывает в партии, в которой оба придерживаются таких стратегий?

Поэтому заключаем, что при таком правиле выигрыша в игре в закрашивание натурального ряда выигрышной стратегии нет ни у одного из игроков.

Свойства 1–4 разбиения на большие и маленькие множества кажутся очень простыми и естественными. Но если попробовать сформулировать более явно правило разделения на большие и маленькие множества, удовлетворяющее этим свойствам, ничего не выйдет. Это не случайно: доказать существование такого разбиения (научное название «неглавный ультрафильтр») можно лишь, используя аксиому выбора или трансфинитную индукцию. Подробнее см., например, вводную книгу Верещагина и Шеня по теории множеств [1].

Задача 5.2. [Задача 123 из [1]] Используя трансфинитную индукцию, докажите существование неглавного ультрафильтра.

Однако и утверждение о существовании выигрывающей стратегии у одного из игроков кажется вполне очевидным. Можно включить его в аксиомы теории множеств (*аксиома детерминированности*. При этом придётся отказаться от аксиомы выбора и существования неглавных ультрафильтров, чтобы не прийти к противоречию. Возникающая теория множеств обладает многими интересными свойствами (в частности, всякое подмножество действительных чисел оказывается в ней измеримым). Подробнее об аксиоме детерминированности можно прочитать в книге Кановея [3].

В рамках стандартной теории множеств, основанной на аксиоме выбора, можно доказать детерминированность многих игр. Наиболее сильным результатом в этом направлении является теорема Мартина о детерминированности борелевских игр [12]. Нам теорема Мартина не понадобится, поэтому точную её формулировку пропустим.

Мы далее будем рассматривать только игры, в которых существуют выигрышные стратегии особого вида: позиционные (memoryless) или стратегии с конечной памятью (выбор хода зависит лишь от конечной информации и начале партии, размер этой информации не зависит от длины начального отрезка партии).

Как мы увидим дальше, существование таких стратегий приводит к существенному ограничению алгоритмической сложности решения игр.

5.2. Игры достижимости

Начнём с очень простого примера, который понадобится нам в дальнейшем. Рассмотрим игру на ориентированном графе, как и выше, но теперь не будем предполагать, что граф ациклический. Тогда становятся возможными бесконечные партии и нужно уточнить правило выигрыша.

Дадим точные определения.

Игровое поле: ориентированный граф $G = (V, E)$ на n вершинах, которые разбиты на два множества $V = V_0 \sqcup V_1$, выделена стартовая вершина v_0 и целевые множества двух игроков $\mathrm{Win}_0 \subseteq V$, $\mathrm{Win}_1 \subseteq V$, $\mathrm{Win}_0 \cap \mathrm{Win}_1 = \varnothing$. Считаем, что множество терминальных вершин (выходной степени 0) содержится в $\mathrm{Win}_0 \cup \mathrm{Win}_1$.

Правила игры. Игроки (называем их 0 и 1) двигают по вершинам графа фишку. В начале игры фишка находится в стартовой вершине. В вершинах множества V_0

ходит игрок 0, в вершинах множества V_1 — игрок 1. Ход состоит в перемещении фишки в одну из соседних вершин. Более точно, из вершины u ход возможен в те и только те вершины v, для которых $(u, v) \in E$.

Игрок 1 выигрывает (и партия заканчивается с результатом $+1$), если в какой-то момент фишка оказалась в вершине из множества Win_1. Игрок 0 выигрывает (и партия заканчивается с результатом -1), если в какой-то момент фишка оказалась в вершине из множества Win_0.

Если же партия длится бесконечно, то фиксируется ничья (результат 0).

Хотя в играх достижимости возможны бесконечные партии, их легко анализировать разбором игры с конца.

Аналогично доказательству теоремы 1.1, построим разбиение множества вершин графа игры на три множества (отвечающие возможной цене игры: ± 1 или 0) и пару позиционных стратегий игроков.

В начале полагаем $S_{-1} = \mathrm{Win}_0$, $S_{+1} = \mathrm{Win}_1$, $S_0 = V \setminus (S_{-1} \cup S_{+1})$.

Просматриваем все позиции из S_0 одна за другой. Если из позиции $v \in V_0$ есть ход в позицию u множества S_{-1}, то включаем эту позицию в S_{-1}, убираем её из S_0 и полагаем $s_0(v) = u$. Если все ходы из позиции $v \in V_0$ ведут в множество S_{+1}, включаем эту позицию в S_{+1}, убираем её из S_0 и выбираем значение $s_0(v)$ произвольным образом. Аналогично для позиции из множества V_1: если существует ход в позицию u из множества S_{+1}, включаем эту позицию в S_{+1} и полагаем $s_1(v) = u$; если все ходы ведут в S_{-1}, включаем эту позицию в S_{+1} и выбираем значение $s_0(v)$ произвольным образом.

Повторяем эту процедуру до тех пор, пока хотя бы одно из множеств S_{-1} или S_{+1} увеличивается.

После этого доопределяем позиционные стратегии в вершинах полученного множества $S_0 = V \setminus (S_{-1} \cup S_{+1})$. По построению, в каждой такой позиции $v \in V_i$ есть ход, ведущий в позицию $u \in S_0$ (иначе одно из множеств S_{-1}, S_{+1} можно расширить). Полагаем $s_i(v) = u$.

Пример исполнения такой процедуры показан на рисунке 5.4.

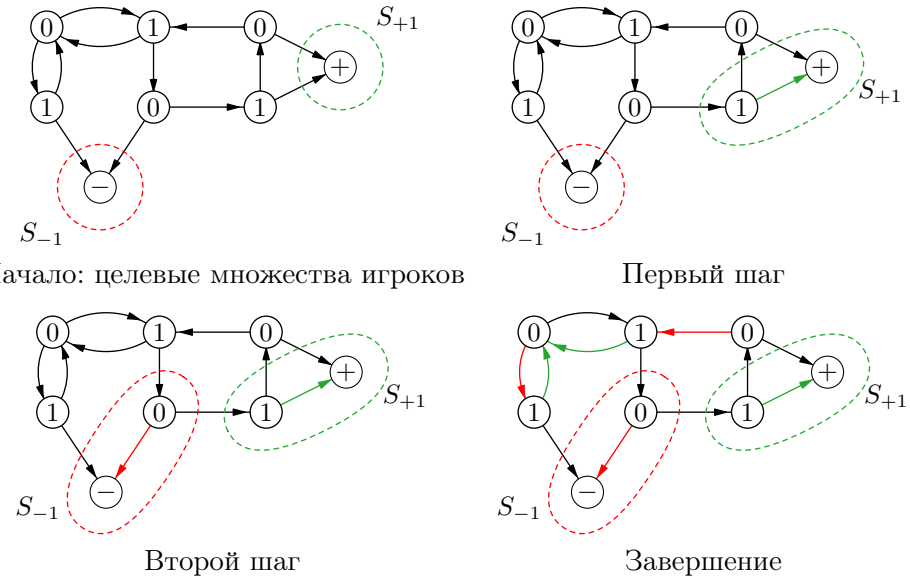
Легко видеть, что описанная процедура реализуется алгоритмом, который работает за полиномиальное от количества позиций время. Количество шагов $O(n^2)$ (множества S_{-1}, S_{+1} увеличиваются не более n раз, и каждый шаг увеличения требует просмотра не более n вершин).

Теорема 5.1. *Цена игры в позиции $v \in S_i$ равна i, и эту цену гарантируют стратегии s_0, s_1.*

Доказательство. Индукцией по количеству шагов в описанной выше процедуре проверим, что утверждение теоремы справедливо на каждом шаге для множеств S_{-1}, S_{+1}.

Перед начальным шагом утверждение теоремы выполняется тривиальным образом. Предположим, что оно выполнено перед шагом, на котором в S_{-1} добавляется позиция v.

Если $v \in V_0$, то стратегия s_0 предписывает ход $(v, s_0(v))$ в позицию, для которой утверждение теоремы выполняется по индуктивному предположению. Если $v \in V_1$,



Начало: целевые множества игроков | Первый шаг

Второй шаг | Завершение

Рис. 5.4: Построение оптимальных стратегий в игре достижимости

то любой ход из v ведёт в позицию из S_{-1}, для которой утверждение теоремы выполняется по индуктивному предположению.

Аналогично рассуждаем в случае, когда позиция v добавляется в S_{+1}.

Теперь докажем, что если игра начинается в позиции $v \in S_0$, стратегия s_i гарантирует, что партия не закончится в позиции из множества Win_{1-i}. Действительно, стратегия s_i ведёт в позицию из S_0 в этом случае. Если же игрок $1-i$ делает ход $(u'u'')$ в позиции $u' \in S_0$, то этот ход не ведёт в множество $S_{1-2i} \supseteq \mathrm{Win}_{1-i}$ (иначе множество S_{1-2i} можно расширить). □

У игр достижимости есть вариант игры на выигрыш. В этом случае есть только одно целевое множество, скажем, Win_1. Цель игрока 1 состоит в том, чтобы партия попала в позицию из этого множества, цель игрока 0 — воспрепятствовать этому. Ясно, что анализ игры в этом случае точно такой же, только множество S_{-1} пустое. В позициях их S_{+1} выигрывает игрок 1, в остальных позициях выигрывает игрок 0.

5.3. Игры чётности

Настала пора ввести главного героя нашего рассказа.

Игровое поле: ориентированный граф $G = (V, E)$ на n вершинах, которые разбиты на два множества $V = V_0 \sqcup V_1$ и выделена стартовая вершина v_0. Рёбрам[1)] графа

[1)] Иногда приоритеты приписываются не рёбрам, а вершинам графа. Мы рассматриваем для удобства более общий случай. Если приоритеты приписаны вершинам, то для перехода к нашему определению, нужно приписать каждому ребру приоритет той вершины, из которой оно исходит.

приписаны числа в диапазоне от 0 до $d-1$ (*приоритеты*). Другими словами, задана функция $p\colon E \to \{0, \ldots, d-1\}$.

Предполагаем, что тупиковых вершин в графе нет: из каждой вершины исходит хотя бы одно ребро.

Правила игры. Игроки (называем их Чёт, он же 0, и Нечет, он же 1) двигают по вершинам графа фишку. В начале игры фишка находится в стартовой вершине. В вершинах множества V_0 ходит Чёт, в вершинах множества V_1 — Нечет. Ход, как и раньше, состоит в перемещении фишки в одну из соседних вершин.

Партия игры. Поскольку тупиковых вершин в графе нет, игра продолжается бесконечно долго. Партия игры чётности — это бесконечный путь по графу игры G

$$v_0,\ v_1,\ \ldots,\ v_t,\ \ldots; \quad (v_i, v_{i+1}) \in E.$$

Результат игры. Определяется значением $\limsup_t p(v_t, v_{t+1})$. Это целое число. Если оно чётно, то выиграл Чёт. В противном случае выигрывает Нечет.

Рассмотрим пример игры чётности, изображённый на рисунке 5.5. Возможная последовательность приоритетов в партии этой игры может быть такой (проверьте!):

$$6,\ 7,\ 4,\ 4,\ 5,\ 6,\ (5,\ 6),$$

в скобках указана периодически повторяющаяся часть последовательности. В такой партии выигрывает Чёт, так как максимальный приоритет, который встречается бесконечное количество раз, равен 6 (чётное).

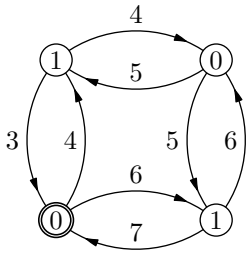


Рис. 5.5: Пример игры чётности

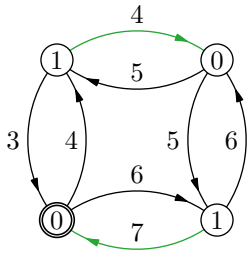


Рис. 5.6: Выигрывающая позиционная стратегия Нечета

Но Нечет может сыграть лучше. Вместо хода из правой нижней вершины в правую верхнюю (приоритет 6), он может сыграть в левую нижнюю (приоритет 7). На рисунке 5.6 показана выигрывающая позиционная стратегия Нечета. Как проверить, что это и впрямь выигрывающая стратегия?

Для этого нужно посмотреть на *стратегический граф*: граф, полученный удалением всех ходов Нечета, не входящих в стратегию. Для данного примера стратегический граф изображён на рисунке 5.7. Аналогично определяется стратегический граф для стратегии Чёта.

Будем называть цикл $(u_0, u_1, \ldots, u_{\ell-1})$ в графе игры *нечётным* (или циклом доминирующего Нечета), если максимальное значение приоритета на рёбрах этого цикла нечётно. Аналогично определим чётные циклы (циклы доминирующего Нечета).

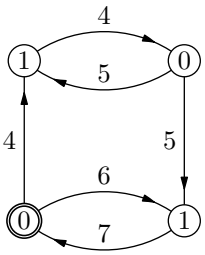


Рис. 5.7: Стратегический граф G_s, все достижимые циклы нечётные

В графе на рисунке 5.7 все циклы, которые достижимы из начальной позиции, нечётные. Проверить это можно разбором возможных случаев. Пусть есть цикл, на котором максимальный приоритет 6. Тогда этот цикл должен быть и в графе $G_s(\leqslant 6)$, который получается удалением всех рёбер приоритета больше 6. Этот граф изображён на рисунке 5.8. Из рисунка видно, что единственное ребро приоритета 6 не лежит в компоненте сильной связности графа $G_s(\leqslant 6)$, поэтому нет цикла, содержащего это ребро.

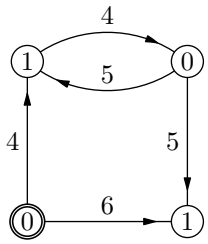


Рис. 5.8: Граф $G_s(\leqslant 6)$

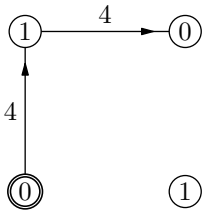


Рис. 5.9: Граф $G_s(\leqslant 4)$

Аналогично рассуждаем для графа $G_s(\leqslant 4)$, который получается удалением всех рёбер приоритета больше 4. Этот граф изображён на рисунке 5.9.

Теперь докажем общую лемму.

Лемма 5.2. *Если граф игры чётности не содержит чётных циклов, то в любой партии выигрывает Нечет. Аналогично, если граф игры чётности не содержит нечётных циклов, то в любой партии выигрывает Чёт.*

Доказательство. Достаточно доказать первое утверждение, второе получается аналогично.

Рассмотрим последовательность приоритетов

$$p_1,\ p_2,\ \ldots, p_t,\ \ldots,$$

возникающую в партии игры чётности на графе, в котором нет чётных циклов. Пусть p^* — максимальный среди приоритетов, которые встречаются бесконечно часто в этой партии. Выберем такой момент времени, после которого уже не встречаются приоритеты больше p^*. После этого момента приоритет p^* встретится бесконечно много раз. Возьмём первые $n+1$ раз (n — количество вершин в графе).

Среди позиций, из которых ведёт ход с приоритетом p^* из этого списка, есть хотя бы две одинаковые. Но тогда отрезок партии между этими двумя позициями образует цикл, на котором максимальный приоритет p^*. Поскольку чётных циклов нет, то приоритет p^* нечётен. □

Задача 5.3. Постройте алгоритм, который за время, полиномиально ограниченное количеством вершин графа игры чётности, проверяет, что в графе нет достижимых из начальной позиции чётных циклов.

В общем случае доказать детерминированность игр чётности (существование выигрышной стратегии у одного из игроков) не так уж просто. Для этого можно применить теорему Мартина. Но мы далее докажем более сильный факт: существование позиционной выигрывающей стратегии у одного из игроков. Это можно сделать и напрямую, комбинаторными рассуждениями (см., например, [7]). Однако мы пойдём другим путём и сведём игры чётности к играм более общего вида, так называемым циклическим играм.

5.4. Циклические игры

Циклические игры (mean payoff games, MPG) — бесконечные игры на конечных графах.

Игровое поле: ориентированный граф $G = (V, E)$ на n вершинах, которые разбиты на два множества $V = V_0 \sqcup V_1$ и выделена начальная вершина v_0. Рёбрам графа приписаны действительные числа (*веса* в дальнейшем)[2)]. Функцию веса обычно будем обозначать w.

Предполагаем, что тупиковых вершин в графе нет: из каждой вершины исходит хотя бы одно ребро.

Правила игры. Игроки (называем их Мин, он же 0, и Макс, он же 1) двигают по вершинам графа фишку. В начале игры фишка находится в стартовой вершине. В вершинах множества V_0 ходит Мин, в вершинах множества V_1 — Макс. Ход, как и раньше, состоит в перемещении фишки в одну из соседних вершин.

Партия игры. Поскольку тупиковых вершин в графе нет, игра продолжается бесконечно долго. Партия циклической игры, как и в игре чётности, — это бесконечный путь по графу игры G

$$v_0,\ v_1,\ \ldots,\ v_t,\ \ldots; \quad (v_i, v_{i+1}) \in E.$$

Результат игры. Теперь это число

$$\limsup_n \frac{1}{n} \sum_{i=0}^{n-1} w(v_i, v_{i+1}),$$

то есть верхний предел суммарных весов начальных отрезков партии. Это число является выигрышем Макса и проигрышем Мина.

[2)] Как обычно, при обсуждении алгоритмов мы будем предполагать, что веса рациональные и заданы двоичной записью числителя и знаменателя.

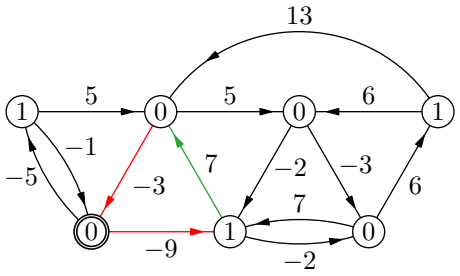


Рис. 5.10: Пример применения жадной стратегии обоими игроками

Не очень понятно, как играть в циклическую игру. Рассмотрим, например, такую *жадную* стратегию: Макс каждый раз выбирает ход наибольшего веса, а Мин — наименьшего веса. Обе стратегии позиционные, поэтому партия сводится через несколько ходов к повторению цикла (см. рисунок 5.10). Результат в такой партии равен среднему весу цикла, как нетрудно посчитать.

В примере на рисунке 5.10 применение жадной стратегии обоими игроками даёт цикл среднего веса $(-9+7-3)/3 = -5/3$. Как выяснится позже, Макс может добиться лучшего результата из начальной вершины, а именно гарантировать результат не меньше -1. Так что его жадный ход неудачен. Более того, первый жадный ход Мина также неудачен: после этого хода Макс сможет гарантировать себе результат не меньше $+2$.

Оказывается, у каждой циклической игры есть цена p (у Макса есть стратегия, гарантирующая результат не меньше p, у Мина есть стратегия, гарантирующая результат не выше p). Но мы будем доказывать более сильное утверждение: оптимальные стратегии игроков могут быть выбраны равномерно позиционными (ход зависит только от текущей позиции и не зависит ни от предыстории партии, ни от начальной позиции).

5.4.1 Каноническая форма циклической игры

Для некоторых циклических игр сформулированное выше сильное свойство почти очевидно.

Циклическая игра в *канонической форме* имеет следующий вид. Существуют две функции на множестве вершин V, $p\colon V \to \mathbb{R}$ (функция цены) и $s\colon V \to V$ (позиционные стратегии обоих игроков), для которых выполняются следующие свойства:

1. Корректность s: $(i, s(i)) \in E(G)$ для всех $i \in V$.
2. Стратегия s сохраняет цену: $p(s(i)) = p(i)$ для всех $i \in V$.
3. Вес стратегического ребра совпадает с ценой концов ребра: $w(i, s(i)) = p(i)$ для всех $i \in V$.

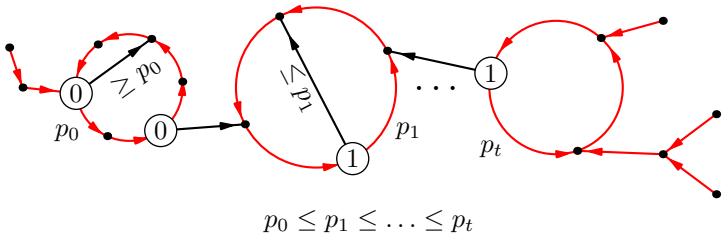


Рис. 5.11: Каноническая форма циклической игры

4. Стратегическое ребро локально лучшее: $p(i) = \min\limits_{(i,j)\in E} w(i,j)$ для всех $i \in V_0$; $p(i) = \max\limits_{(i,j)\in E} w(i,j)$ для всех $i \in V_1$.

5. У Макса нет ходов, увеличивающих цену; у Мина нет ходов, уменьшающих цену: если $(ij) \in E$, то $p(j) \geqslant p(i)$ для всех $i \in V_0$ и $p(j) \leqslant p(i)$ для всех $i \in V_1$.

Схематически каноническая форма представлена на рисунке 5.11.

Свойства канонической формы таковы, что любой игрок, отклоняющийся от стратегии, задаваемой функцией s, не улучшает свой результат. Поэтому для игры в канонической форме сформулированные выше свойства выполнены. Оптимальные стратегии задаются функцией s, а функция $p(v)$ задаёт цену игры, начинающейся в вершине v.

5.4.2 Эквивалентные игры и потенциальные преобразования

Пусть на вершинах графа задана функция $\varphi\colon V \to \mathbb{R}$ (потенциал). Она определяет потенциальную функцию на рёбрах графа по правилу

$$d\varphi\colon (ij) \mapsto \varphi(j) - \varphi(i).$$

Лемма 5.3. *Прибавление потенциальной функции к весам графа не меняет результата игры на любой последовательности ходов.*

Доказательство. Сумма значений потенциальной функции вдоль любого маршрута равна разности потенциалов в конечной и начальной вершинах маршрута. Поэтому средняя длина бесконечного маршрута относительно потенциальной функции равна 0. □

Таким образом, различающиеся на потенциальную функцию игры эквивалентны в сильном смысле: результат в любой партии один и тот же.

Приведём пример потенциального преобразования к канонической форме для игры, изображённой на рисунке 5.10. Рассматривая игру в канонической форме, нетрудно убедиться в справедливости замечаний, сделанных при анализе жадных стратегий.

Теорема 5.4 ([2])**.** *Для каждой циклической игры есть эквивалентная игра в канонической форме.*

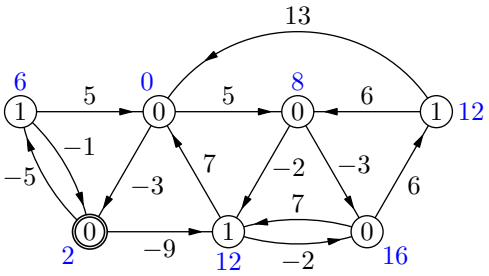


Рис. 5.12: Потенциалы

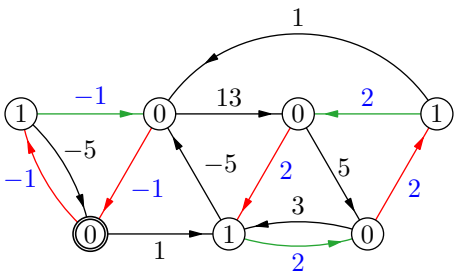


Рис. 5.13: Каноническая форма после потенциального преобразования

Из этой теоремы сразу же следуют сформулированные выше свойства циклических игр.

Следствие 5.5. *У каждой циклической игры есть цена. Эта цена гарантируется равномерными позиционными стратегиями игроков.*

В [2] теорема о канонической форме доказана алгоритмически: приведён (псевдополиномиальный) алгоритм, который строит каноническую форму. Однако есть и другие доказательства этого факта. В частности, эта теорема легко вытекает из некоторого общего факта о свойствах нерастягивающих кусочно-линейных отображений. Именно такое доказательство мы изложим ниже.

5.5. Связь между играми чётности и циклическими играми

Из существования позиционных оптимальных стратегий в циклических играх (следствие 5.5) следует сводимость игр чётности к циклическим играм. Пусть $\mathfrak{G}$ — игра чётности $G = (V, E)$, $V = V_0 \sqcup V_1$, v_0, $p\colon E \to \{0, \dots, d-1\}$. Построим по ней циклическую игру $\mathfrak{G}'$ с тем же графом, той же начальной позицией и тем же разбиением позиций между игроками. Веса рёбер определим так:

$$w(u, v) = (-1)^{p(u,v)} M^{p(u,v)}, \quad \text{где } M = n + 1, \tag{5.1}$$

как обычно, n обозначает количество вершин в графе игры.

Лемма 5.6. *У Чёта существует выигрывающая позиционная стратегия в игре $\mathfrak{G}$ тогда и только тогда, когда цена игры $\mathfrak{G}'$ положительная. В противном случае выигрывающая позиционная стратегия существует у Нечета.*

Для циклических игр нам потребуется аналог леммы 5.2 для игр чётности.

Лемма 5.7. *Если граф циклической игры не содержит циклов неположительного веса, то результат любой партии положительный. Аналогично, если граф циклической игры не содержит циклов положительного веса, то результат любой партии неположительный.*

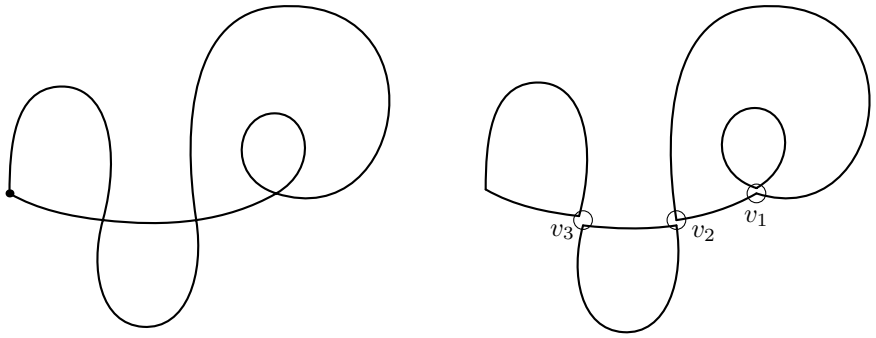


Рис. 5.14: Разложение цикла в простые циклы

Доказательство. Достаточно доказать первое утверждение, второе получается аналогично.

Рассмотрим последовательность весов

$$w_1,\ w_2,\ \ldots, w_t,\ \ldots,$$

возникающую в партии циклической игры без неположительных циклов. Хотя бы одна вершина v^* встречается в этой партии бесконечно часто. Суммы весов между последовательными посещениями v^* — это длины циклов, они положительны. Отсюда следует лемма.

Есть, однако, одна тонкость. Из сказанного легко следует неотрицательность результата партии. Но почему он (то есть верхний предел частичных сумм) положительный?

Вес любого цикла равен сумме весов простых циклов (в которых позиции не повторяются), см. рисунок 5.14. Простых циклов в графе конечное количество, длина каждого из них не больше количества вершин n графа игры. Если обозначить Δ наименьший вес простого цикла, а T — время между двумя посещениями v^*, то сумма весов между этими посещениями не меньше $\Delta\lfloor T/n\rfloor$. Поэтому верхний предел частичных не меньше $\Delta/n > 0$. □

Доказательство леммы 5.6. Выбор весов по формуле (5.1) гарантирует, что простой цикл чётный тогда и только тогда, когда его вес положительный: длина простого цикла не больше n, поэтому знак суммарного веса цикла определяется наибольшим по модулю слагаемым.

По следствию 5.5 у игры $\mathfrak{G}'$ существует цена $\mathrm{val}(\mathfrak{G}')$.

Пусть эта цена положительная. Тогда граф стратегии Макса, гарантирующей эту цену, не содержит циклов неположительного веса (иначе Мин использовал бы такой цикл, чтобы достичь неположительного результата). Поэтому граф той же стратегии, но в игре чётности, содержит только чётные циклы. По лемме 5.2 Чёт выигрывает, придерживаясь этой стратегии.

Аналогично разбирается случай, когда цена неположительная. Теперь у Мина есть стратегия, гарантирующая эту цену. Граф этой стратегии не содержит положительных циклов и потому все циклы в игре чётности нечётные. □

Тема 6

Теорема Колберга и позиционные стратегии в бесконечных играх

6.1. Игры с затухающим платежом

Рассмотрим ещё один класс игр на графах, для которого доказать существование позиционных оптимальных стратегий проще, чем для циклических игр. Это игры с затухающим платежом (discounted payoff games, DPG).

Игровое поле: ориентированный граф $G = (V, E)$ на n вершинах, которые разбиты на два множества $V = V_0 \sqcup V_1$ и выделена стартовая вершина v_0. Рёбрам графа приписаны действительные числа (*веса* в дальнейшем). Функцию веса обозначаем по-прежнему w.

Предполагаем, что тупиковых вершин в графе нет: из каждой вершины исходит хотя бы одно ребро.

Правила игры. Игроки (называем их Мин, он же 0, и Макс, он же 1) двигают по вершинам графа фишку. В начале игры фишка находится в стартовой вершине. В вершинах множества V_0 ходит Мин, в вершинах множества V_1 — Макс. Ход, как и раньше, состоит в перемещении фишки в одну из соседних вершин.

Партия игры. Это бесконечный путь по графу игры G

$$v_0,\ v_1,\ \ldots,\ v_t,\ \ldots; \quad (v_i, v_{i+1}) \in E.$$

Результат игры. У игры с затухающим платежом есть ещё один параметр λ, предполагаем, что $0 < \lambda < 1$. Он используется в формуле для результата игры:

$$(1 - \lambda) \sum_{i=0}^{\infty} \lambda^i w(v_i, v_{i+1}).$$

Легко проверить, что такой ряд сходится. Это число является выигрышем Макса и проигрышем Мина. Множитель $1 - \lambda$ выбран для удобства в сравнении игр с затухающим платежом и циклических игр.

Влияние ходов на результат падает по мере партии. Это подсказывает, что цена игры с затуханием должна получаться предельным переходом от игр с затуханием

за конечное время. В конечном варианте игры с затуханием выбирается параметр T, игра длится T ходов и её результат равен

$$(1-\lambda)\sum_{i=0}^{T-1}\lambda^i w(v_i, v_{i+1}).$$

Это фактически игра на конечном графе с аддитивными платежами, мы уже рассматривали такие игры в разделе 1.2. Вершины графа — это пары (v,t), где $v \in V$, $0 \leqslant t \leqslant T$. Рёбра имеют вид

$$\big((u,t),(v,t+1)\big), \qquad (u,v)\in E,\ 0 \leqslant t < T.$$

Вес такого ребра равен $(1-\lambda)\lambda^t w(u,v)$.

Терминальные позиции — это пары (v,T). Начальная позиция — $(v_0,0)$.

По теореме 1.3 у такой конечной игры с затуханием есть цена $\mathrm{val}(v,T)$. Более того, из доказательства теоремы легко следуют соотношениями между ценами игр, начинающихся в других вершинах графа:

$$\begin{aligned}
&\mathrm{val}(v,T) = 0;\\
&\mathrm{val}(u,t) = \min_{v:(u,v)\in E}\big((1-\lambda)\lambda^t w(u,v) + \mathrm{val}(v,t+1)\big), \quad \text{если } u\in V_0,\ 0\leqslant t<T;\\
&\mathrm{val}(u,t) = \max_{v:(u,v)\in E}\big((1-\lambda)\lambda^t w(u,v) + \mathrm{val}(v,t+1)\big), \quad \text{если } u\in V_1,\ 0\leqslant t<T.
\end{aligned} \tag{6.1}$$

Цену игры за время T можно вычислить немного иным способом. Определим на пространстве $\mathbb{R}^n$, базис которого индексирован вершинами графа, отображение

$$f(x)_i = \begin{cases} \max\limits_{(ij)\in E}(\lambda x_j + (1-\lambda)w(i,j)), & \text{если } i\in V_1;\\ \min\limits_{(ij)\in E}(\lambda x_j + (1-\lambda)w(i,j)), & \text{если } i\in V_0. \end{cases} \tag{6.2}$$

Лемма 6.1. $\mathrm{val}_T(v) = f^{(T)}(0^n)_v$, *здесь* $\mathrm{val}_T(v)$ *— цена игры с затуханием длительности* T*, начинающейся в вершине* v*.*

Здесь показатель в круглых скобках обозначает последовательные итерации отображения:

$$f^{(n)}(x) = \underbrace{f(f(\dots f(x)\dots))}_{n \text{ раз}}.$$

Доказательство. Индукция по T. База — $T=0$ — очевидна.

Пусть лемма верна для T. По формулам (6.1), применённым к $t=0$, получаем

$$\begin{aligned}
\mathrm{val}_{T+1}(u) &= \min_{v:(u,v)\in E}\big((1-\lambda)w(u,v) + \lambda\,\mathrm{val}_T(v)\big), \quad \text{если } u\in V_0;\\
\mathrm{val}_{T+1}(u) &= \max_{v:(u,v)\in E}\big((1-\lambda)w(u,v) + \lambda\,\mathrm{val}_T(v)\big), \quad \text{если } u\in V_1.
\end{aligned}$$

Множитель λ во вторых слагаемых возникает из-за того, что в формулах (6.1) мы должны считать цену игры от 1 до $T+1$, а не от 0 до T.

Поэтому $\mathrm{val}_{T+1} = f(\mathrm{val}_T) = f^{(T+1)}(0^n)$, что и требовалось. □

У отображения f есть неподвижная точка. Это следует из хорошо известной теоремы о сжимающих отображениях $\mathbb{R}^n$.

Напомним, что *нормой* на пространстве $\mathbb{R}^n$ называется такая функция $\|\cdot\|\colon \mathbb{R}^n \to \mathbb{R}$, что

1. $\|x\| \geqslant 0$ для всех $x \in \mathbb{R}^n$; если $\|x\| = 0$, то $x = 0^n$;
2. $\|\lambda x\| = |\lambda| \cdot \|x\|$ для всех $x \in \mathbb{R}^n$, $\lambda \in \mathbb{R}$;
3. $\|x + y\| \leqslant \|x\| + \|y\|$ для всех $x, y \in \mathbb{R}^n$.

Пример 8. Функция

$$\|x\|_\infty = \max_i |x_i|$$

удовлетворяет свойствам нормы (проверьте!). Такая норма называется ℓ_∞.

Теорема 6.2. *Пусть $f\colon \mathbb{R}^n \to \mathbb{R}^n$ — сжимающее отображение относительно некоторой нормы $\|\cdot\|$, то есть для некоторого $0 < k < 1$ и для любых $x, y \in \mathbb{R}^n$ выполнено неравенство*

$$\|f(x) - f(y)\| \leqslant k\|x - y\|.$$

Тогда у отображения f есть единственная неподвижная точка x^* и для любого $x \in \mathbb{R}^n$ последовательность итераций $f^{(n)}(x)$ сходится к x^*.

Для полноты изложения приведём доказательство этой теоремы. Оно проиллюстрировано на рисунке 6.1.

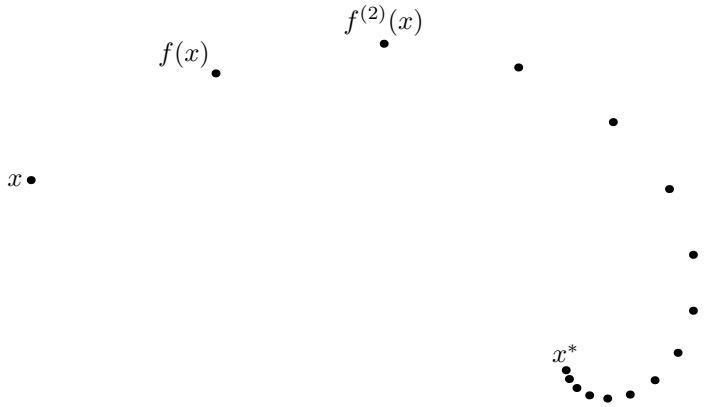


Рис. 6.1: Итерации сжимающего отображения

Доказательство. Единственность неподвижной точки мгновенно следует из сжимаемости: если $f(x) = x$ и $f(y) = y$, то

$$\|x - y\| = \|f(x) - f(y)\| \leqslant k\|x - y\| < \|x - y\|$$

при $x \neq y$.

Теперь рассмотрим последовательность итераций $x_n = f^{(n)}(x)$. Из условия сжимаемости следует $\|x_{n+1} - x_n\| \leqslant k\|x_n - x_{n-1}\|$, то есть $\|x_{n+1} - x_n\| \leqslant k^n\|x_1 - x_0\|$. Применив неравенство треугольника, видим, что последовательность x_n фундаментальная и потому сходится. Её предел x^* обязан быть неподвижной точкой, поскольку $f(x^*)$ также будет пределом этой последовательности. □

Лемма 6.3. *Отображение f, задаваемое формулами* (6.2)*, является сжимающим относительно нормы ℓ_∞.*

Доказательство. Пусть $i \in V_1$, $x, y \in \mathbb{R}^n$. Обозначим через ℓ того соседа вершины i, на котором достигается максимум $\lambda x_j + (1-\lambda)w(i,j)$ в формуле (6.2); а через k — того соседа вершины i, на котором достигается максимум $\lambda y_j + (1-\lambda)w(i,j)$.

Так как по определению

$$f(x)_i - f(y)_i = \max_{(ij)\in E}(\lambda x_j + (1-\lambda)w(ij)) - \max_{(ij)\in E}(\lambda y_j + (1-\lambda)w(ij)),$$

то получаем неравенства

$$f(x)_i - f(y)_i \leqslant (\lambda x_\ell + (1-\lambda)w(i,\ell)) - (\lambda y_\ell + (1-\lambda)w(i,\ell)) = \lambda(x_\ell - y_\ell).$$
$$f(x)_i - f(y)_i \geqslant (\lambda x_k + (1-\lambda)w(i,k)) - (\lambda y_k + (1-\lambda)w(i,k)) = \lambda(x_k - y_k).$$

Если $b \leqslant x \leqslant a$, то $|x| \leqslant \max(|a|,|b|)$. Из полученных выше неравенств заключаем, что значение модуля i-й координаты $f(x) - f(y)$ хотя бы в λ раз меньше значения модуля какой-то координаты $x - y$, то есть $|f(x)_i - f(y)_i| \leqslant \lambda\|x - y\|_\infty$.

Для $i \in V_0$ рассуждаем аналогично. Так что $\|f(x) - f(y)\|_\infty \leqslant \lambda\|x - y\|_\infty$. □

Итак, для любой игры с затухающими платежами у отображения (6.2) есть неподвижная точка. Обозначим её val. Из формул (6.2) следует, что выполняются соотношения

$$\mathrm{val}_i = \begin{cases} \max\limits_{(ij)\in E}(\lambda\,\mathrm{val}_j + (1-\lambda)w(i,j)), & \text{если } i \in V_1; \\ \min\limits_{(ij)\in E}(\lambda\,\mathrm{val}_j + (1-\lambda)w(i,j)), & \text{если } i \in V_0. \end{cases} \tag{6.3}$$

Наша цель — доказать, что val_i является ценой позиции i и эта цена гарантируется позиционными стратегиями Макса и Мина.

Рассмотрим позицию i. Пусть максимум (минимум) для val_i достигается в позиции $k = s(i)$. Функция $s\colon V \to V$ задаёт пару стратегий s_1, s_0 Макса и Мина соответственно.

Утверждение 6.4. *Если Макс придерживается стратегии s_1, то результат в партии, начинающейся из позиции i, не меньше* val_i*. Аналогично для Мина, придерживающегося стратегии s_0: результат не выше* val_i*.*

Доказательство. Пусть Макс придерживается стратегии s_1, а игра начинается из позиции v_0. Из (6.3) следует

$$\mathrm{val}_i \leqslant \lambda\,\mathrm{val}_j + (1-\lambda)w(i,j).$$

Неравенство возникает из-за того, что Мин может делать ходы, не соответствующие стратегии s_0 и на таких ходах равенство в (6.3) заменяется на неравенство.

Поэтому

$$(1-\lambda)\sum_{i=0}^{T-1}\lambda^i w(v_i,v_{i+1}) \geqslant \sum_{i=0}^{T-1}\lambda^i\big(\operatorname{val}_{v_i}-\lambda\operatorname{val}_{v_{i+1}}\big) \geqslant \operatorname{val}_{v_0}-\lambda^T\operatorname{val}_{v_T}.$$

Переходя к пределу при $T\to\infty$, получаем нужное неравенство: результат в партии не меньше val_{v_0}.

Аналогично доказывается утверждение для Мина, придерживающегося стратегии s_0. □

Итак, для игр с затухающими платежами есть цена и она гарантируется позиционными стратегиями игроков. Посмотрим, что изменяется, если перейти к циклическим играм.

6.2. От сжимающих отображений к нерастягивающим

Как и для игр с затухающими платежами, для циклических игр можно определить конечный вариант, в котором игра продолжается T ходов. Результат такой игры с точностью до множителя $1/T$ равен сумме весов ходов, сделанных в партии. Поэтому цены конечной игры, умноженные на T, будут задаваться соотношениями, аналогичными (6.1):

$$\begin{aligned}
&\operatorname{val}(v,T)=0;\\
&\operatorname{val}(u,t)=\min_{v:(u,v)\in E}\big(w(u,v)+\operatorname{val}(v,t+1)\big),\quad \text{если } u\in V_0,\ 0\leqslant t<T;\\
&\operatorname{val}(u,t)=\max_{v:(u,v)\in E}\big(w(u,v)+\operatorname{val}(v,t+1)\big),\quad \text{если } u\in V_1,\ 0\leqslant t<T.
\end{aligned}\tag{6.4}$$

Эти цены, как и для игр с затухающими платежами, можно выразить через итерации подходящего отображения. Теперь оно имеет вид

$$f(x)_i=\begin{cases}\max\limits_{(ij)\in E}(x_j+w(i,j)), & \text{если } i\in V_1;\\ \min\limits_{(ij)\in E}(x_j+w(i,j)), & \text{если } i\in V_0.\end{cases}\tag{6.5}$$

Упражнение 9. Проверьте, что $\operatorname{val}_T(v)=f^{(T)}(0^n)_v$, здесь $\operatorname{val}_T(v)$ — цена циклический игры длительности T, начинающейся в вершине v.

Задаваемое формулами (6.5) отображение уже не обязательно сжимающее.

Задача 6.1. Приведите пример циклической игры, для которой отображение (6.5) не является сжимающим в любой норме (и докажите, что оно несжимающее).

Однако про отображение (6.5) всё-таки можно утверждать нечто позитивное.

Лемма 6.5. *Отображение* (6.5) *нерастягивающее в ℓ_∞-норме, то есть*

$$\|f(x)-f(y)\|_\infty\leqslant\|x-y\|_\infty$$

для любых $x,y\in\mathbb{R}^n$.

Доказательство. Пусть $i \in V_1$, $x, y \in \mathbb{R}^n$. Обозначим через ℓ того соседа вершины i, на котором достигается максимум $x_j + w(i,j)$; а через k — того соседа вершины i, на котором достигается максимум $y_j + w(i,j)$.

Так как по определению

$$f(x)_i - f(y)_i = \max_{(ij)\in E}(x_j + w(i,j)) - \max_{(ij)\in E}(y_j + w(i,j)),$$

то получаем неравенства

$$f(x)_i - f(y)_i \leqslant (x_\ell + w(i,\ell)) - (y_\ell + w(i,\ell)) = x_\ell - y_\ell,$$
$$f(x)_i - f(y)_i \geqslant (x_k + w(i,k)) - (y_k + w(i,k)) = x_k - y_k,$$

из которых следует $\|f(x) - f(y)\|_\infty \leqslant \|x - y\|_\infty$ (значение каждой координаты $f(x) - f(y)$ зажато между значениями каких-то координат $x - y$).

Для $i \in V_0$ рассуждаем аналогично. □

У нерастягивающих отображений может не быть неподвижной точки. Например, у параллельного переноса её нет.

Колберг обобщил теорему о сжимающих отображения на случай нерастягивающих кусочно-линейных отображений.

Теорема 6.6 (теорема Колберга, [9])**.** *Пусть отображение $f\colon \mathbb{R}^n \to \mathbb{R}^n$ нерастягивающее относительно некоторой нормы $\|\cdot\|$, то есть для любых $x, y \in \mathbb{R}^n$ выполнено неравенство*

$$\|f(x) - f(y)\| \leqslant \|x - y\|.$$

Пусть также f — кусочно-линейное, то есть для некоторого разбиения пространства на конечное число полиэдров $\{x : C_j x \geqslant d_j\}$ отображение f аффинное на каждом полиэдре разбиения: $f(x) = A_j x + b_j$, если $C_j x \geqslant d_j$.

Тогда существуют единственный вектор $a \in \mathbb{R}^n$ и вектор $b \in \mathbb{R}^n$ такие, что $f(b + at) = b + a(t+1)$ для всех $t \geqslant 0$.

Другими словами, у нерастягивающего кусочно-линейного отображения есть либо неподвижная точка, либо инвариантный луч, ограничение на который является параллельным переносом. Инвариантных лучей может быть несколько (как в случае параллельного переноса), однако направление у них одно и то же.

У теоремы Колберга есть два важных следствия.

Следствие 6.7. *Пусть отображение $f\colon \mathbb{R}^n \to \mathbb{R}^n$ нерастягивающее относительно некоторой нормы $\|\cdot\|$. Тогда существует единственный a такой, что для любого $x \in \mathbb{R}^n$ последовательность $f^{(n)}(x) - na$ ограничена.*

Следствие 6.8. *Пусть отображение $f\colon \mathbb{R}^n \to \mathbb{R}^n$ нерастягивающее относительно некоторой нормы $\|\cdot\|$. Тогда для любого $x \in \mathbb{R}^n$ последовательность $f^{(n)}(x)/n$ сходится к одному и тому же вектору a.*

Более того, f имеет неподвижную точку тогда и только тогда, когда $a = 0$.

6.3. Теорема о канонической игре из теоремы Колберга

Прежде чем доказывать теорему Колберга, выведем из неё теорему 5.4 о существовании игры в канонической форме, эквивалентной данной циклической игре. Это делается очень просто.

Очевидно, что отображение (6.5) кусочно-линейное (оно аффинно на полиэдрах, которые задают упорядочения $x_j + w(i,j)$ в соседях каждой вершины).

Применим к этому отображению теорему Колберга. Обозначим через $b + pt$ инвариантный луч: $f(b + pt) = b + p(t + 1)$.

Заметим, что начало луча можно сдвигать вдоль луча с сохранением свойства инвариантности.

Условие инвариантности луча даёт равенства

$$(b + p(t+1))_i = \begin{cases} \max\limits_{(ij)\in E}((b+pt)_j + w(i,j)), & \text{если } i \in V_1; \\ \min\limits_{(ij)\in E}((b+pt)_j + w(i,j)), & \text{если } i \in V_0 \end{cases}$$

для любого $t \geqslant 0$.

Под максимумом (минимумом) стоят линейные функции от t. Поэтому можно выбрать такое T, что при всех $t \geqslant T$ для каждого i максимум (минимум) достигается на одном и том же соседе $s(i)$.

Далее считаем без ограничения общности, что $T = 0$ (то есть при необходимости переходим к сдвинутому на T лучу).

Проверим, что в этом случае игра с весовой функцией $w' = w + db$ является канонической с функцией цены p.

Выберем вершину $i \in V$ и обозначаем $k = s(i)$. Для всех $t \geqslant 0$ выполняется равенство

$$b_k + p_k t + w(i,k) = b_i + p_i(t+1) = b_i + p_i t + p_i,$$

то есть $p_k = p_i = w(i,k) + b_k - b_i = w'(i,k)$.

Если $i \in V_1$, то для любого другого соседа j вершины i при всех $t \geqslant 0$ выполняется неравенство

$$b_k + p_k t + w(i,k) \geqslant b_j + p_j t + w(i,j),$$

из которого следуют неравенства $p_j \leqslant p_k$ (ход Макса не увеличивает цену игры) и

$$w'(i,j) = w(i,j) + b_j - b_i \leqslant w(i,k) + b_k - b_i = w'(i,k)$$

(стратегический ход $(i, s(i))$ локально лучший).

Это и есть свойства канонической игры. Для $i \in V_0$ всё аналогично, с заменой неравенств.

6.4. Доказательство теоремы Колберга и следствий

Для любого $t > 0$ дисконтированное отображение

$$F_t(x) = \frac{1}{1 + 1/t} f(x)$$

является сжимающим с коэффициентом $t/(t+1) < 1$. Поэтому для него существует неподвижная точка x_t. Для этой точки выполняется соотношение

$$f(x_t) = \Big(1 + \frac{1}{t}\Big)x_t.$$

Задача 6.2. Докажите, что для любого нерастягивающего отображения f из ограниченности множества x_t следует существование неподвижной точки x^* отображения f (для которой $x^* = f(x^*)$).

Пусть множество x_t неограничено. Рассмотрим центральные проекции u_t точек x_t на единичную сферу $S_1 = \{x : \|x\| = 1\}$. Это множество ограничено и потому у него есть предельная точка u^*. Естественно предположить, что это и есть направление инвариантного луча для отображения f.

Однако извлечь отсюда доказательство теоремы Колберга невозможно.

Задача 6.3. Постройте пример нерастягивающего отображения, для которого теорема Колберга не выполняется.

Для доказательства теоремы Колберга потребуются более тонкие средства.

Расширим поле действительных чисел $\mathbb{R}$ до поля рациональных функций с действительными коэффициентами $\mathbb{R}(t)$. Рациональные функции строятся из многочленов с действительными коэффициентами точно так же, как рациональные числа строятся из целых. Элемент поля $\mathbb{R}(t)$ представляется отношением двух многочленов $p(t)/q(t)$, $q(t) \neq 0(t)$ (не равен тождественно нулю).

Поле $\mathbb{R}(t)$ упорядоченное. Заметим, что отношение двух многочленов $p(t)/q(t)$ как функция действительного аргумента определено почти всюду, за исключением конечного множества нулей многочлена $q(t)$. Кроме того, знак $p(t)/q(t)$ одинаковый для достаточно больших t (достаточно проверить это для многочленов). Определим теперь линейный порядок на поле $\mathbb{R}(t)$. По определению, $p(t)/q(t) > 0$ как элемент поля $\mathbb{R}(t)$, если соответствующая функция действительного аргумента $p(t)/q(t)$ положительна для всех достаточно больших действительных t. Скажем, что $f(t) > g(t)$, если $f(t) - g(t) > 0$.

Упражнение 10. Проверьте, что данное выше определение задаёт линейный порядок на поле $\mathbb{R}(t)$.

Упражнение 11. Проверьте, что этот порядок согласован с операциями в поле в обычном смысле: (1) если $f \geqslant g$, то $f + h \geqslant g + h$ для любого h; (2) если $f \geqslant 0$ и $g \geqslant 0$, то $fg \geqslant 0$.

Над упорядоченным полем определены линейные неравенства. Кусочно-линейное отображение f из теоремы Колберга продолжается до кусочно-линейного отображения $\tilde{f}\colon \mathbb{R}(t)^n \to \mathbb{R}(t)^n$ векторного пространства над полем рациональных функций. (Тут существенно, что знак рациональной функции совпадает со знаком её значений при всех достаточно больших аргументах t.)

Утверждение 6.9. *Дисконтированное уравнение*

$$\tilde{f}(x) = \Big(1 + \frac{1}{t}\Big)x$$

имеет решение в пространстве $\mathbb{R}(t)^n$ над полем рациональных функций от одной переменной.

Другими словами, у дисконтированного отображения $\tilde{F}$, продолженного на пространство $\mathbb{R}(t)^n$, есть неподвижная точка.

И вот этого уже оказывается достаточным. Выведем теорему Колберга из утверждения 6.9. Пусть $x(t) = p(t)/q(t)$ — решение дисконтированного уравнения в $\mathbb{R}(t)^n$. Разложим координаты $x(t)$ в ряд Лорана в окрестности ∞, получаем

$$x(t) = \sum_{-\infty}^{N} a_i t^i, \qquad a_i \in \mathbb{R}^n, \quad N = \deg p - \deg q.$$

На $x(t)$ можно смотреть как на векторнозначную функцию действительного аргумента t, определённую для почти всех значений t. В силу единственности неподвижной точки для сжимающего отображения $F_t\colon \mathbb{R}^n \to \mathbb{R}^n$ заключаем, что $x(t) = x_t$ для всех достаточно больших t.

Выведем отсюда, что $N \leqslant 1$. Действительно,

$$\Big(1 + \frac{1}{t}\Big)\|x_t\| - \|f(0)\| = \|f(x_t)\| - \|f(0)\| \leqslant \|f(x_t) - f(0)\| \leqslant \|x_t - 0\| = \|x_t\|,$$

то есть $\|x_t\| \leqslant \|f(0)\| \cdot t$.

Теперь докажем, что $a_0 + a_1 t$ — инвариантный луч при достаточно больших t. Из разложения в ряд Лорана видим, что $\|x_t - (a_0 + a_1 t)\| = o(1)$. Поскольку f нерастягивающее, то же самое верно и для $\|f(x_t) - f(a_0 + a_1 t)\|$. Но

$$f(x_t) = \Big(1 + \frac{1}{t}\Big)x_t = \Big(1 + \frac{1}{t}\Big)(a_0 + a_1 t + o(1)) = a_0 + a_1(t+1) + o(1).$$

Значит,

$$\|f(a_0 + a_1 t) - (a_0 + a_1(t+1))\| = o(1).$$

Теперь заметим, что при достаточно больших t точки луча $a_0 + a_1 t$ лежат в одном из полиэдров разбиения, и при таких t отображение f аффинное. Отсюда следует, что

$$f(a_0 + a_1 t) = a_0 + a_1(t+1),$$

что и требовалось.

Проверим единственность a_1. Для этого воспользуемся тем, что композиция нерастягивающих отображений — нерастягивающее. В частности,

$$\|f^{(n)}(x) - f^{(n)}(a_0)\| = \|f^{(n)}(x) - (a_0 + n a_1)\| \leqslant \|x - a_0\|.$$

Значит, для любого x последовательность $f^{(n)}(x)/n$ сходится к a_1. Заодно получили первое утверждение следствия 6.8 и второе утверждение этого следствия в одну сторону (если есть неподвижная точка, то $a = 0$). В другую сторону столь же просто: мы уже проверили, что $a = a_1$, а $a_1 = 0$ в точности означает, что инвариантный луч вырождается в неподвижную точку.

Следствие 6.7 получается так. Во-первых, заметим, что для $a = a_1$ и любого $x \in \mathbb{R}^n$ последовательность $f^{(n)}(x) - na = f^{(n)}(x) - na_1$ ограничена $\|a_0\| + \|x - a_0\|$ (здесь мы выбираем какой-нибудь инвариантный луч). Если же $a \neq a_1$, то

$$\|f^{(n)}(x) - na\| \geqslant \|na_1 - na\| - \|f^{(n)}(x) - na\| \to \infty \quad \text{при } n \to \infty.$$

Осталось доказать утверждение 6.9. Для этого нам потребуется лемма Фаркаша для произвольного упорядоченного поля $\Bbbk$.

Лемма 6.10 (лемма Фаркаша)**.** *Пусть $Ax \geqslant b$ — система линейных неравенств в упорядоченном поле $\Bbbk$. Эта система несовместна тогда и только тогда, когда существует такой $y \geqslant 0$, что $yA = 0$ и $yb > 0$.*

Эту лемму можно доказать методом исключения переменных Фурье–Моцкина, который работает для любого упорядоченного поля.

Доказательство утверждения 6.9. Предположим, что дисконтированное уравнение не имеет решений в $\mathbb{R}(t)^n$. Это означает, что для любого j (индекс полиэдра разбиения) несовместна система

$$\begin{aligned} &A_j x - \Big(1 + \frac{1}{t}\Big)x = -b_j, \\ &C_j x \geqslant d_j. \end{aligned}$$

По лемме Фаркаша существуют такие y_j, $v_j \geqslant 0$ что

$$\begin{aligned} &y_j A_j - \Big(1 + \frac{1}{t}\Big)y_j = 0, \\ &v_j C_j = 0, \qquad\qquad - y_j b_j + v_j d_j > 0. \end{aligned}$$

Значит, при достаточно больших t выполняются соотношения

$$\begin{aligned} &y_j(t) A_j - \Big(1 + \frac{1}{t}\Big)y_j(t) = 0, \\ &v_j(t) C_j = 0, \qquad\qquad - y_j(t) b_j + v_j(t) d_j > 0. \end{aligned}$$

Теперь применяем лемму Фаркаша для обычного пространства $\mathbb{R}^n$ и видим, что при этих достаточно больших t для любого j нет такого x, для которого выполнялись бы соотношения

$$\begin{aligned} &A_j x + b_j = \Big(1 + \frac{1}{t}\Big)x, \\ &C_j x \geqslant d_j. \end{aligned}$$

Но это противоречит существованию неподвижной точки x_t для дисконтированного отображения. □

Тема 7

Алгоритмические следствия из теоремы Колберга

7.1. Языки, связанные с решением игр

Игры с затухающим платежом и циклические игры — это игры на результат. Под решением такой игры естественно понимать определение цены игры и нахождение оптимальных стратегий. Такая алгоритмическая задача не является задачей разрешения языка, что не позволяет прямо поместить её в иерархию сложностных классов.

Однако с этими играми естественно связаны языки, алгоритмы разрешения которых позволяют и находить цену, и находить стратегии. Сейчас мы эти языки определим.

Для удобства мы считаем далее, что веса рёбер целочисленные. Параметр затухания λ для DPG считаем рациональным числом. Под описанием $\langle G \rangle$ игры G будем понимать двоичное слово, которое кодирует каким-нибудь естественным образом граф игры, разбиение вершин на множества, контролируемые игроками, начальную вершину, веса рёбер и (для DPG) параметр затухания.

Определим два языка:

$$\mathsf{DPG} = \{\langle G \rangle : \text{цена DPG } G \text{ положительная,}\} \tag{7.1}$$

$$\mathsf{MPG} = \{\langle G \rangle : \text{цена MPG } G \text{ положительная.}\} \tag{7.2}$$

Утверждение 7.1. *Существует алгоритм решения игр с затухающим платежом, который имеет возможность обращаться к оракулу* DPG *и работает за полиномиальное время.*

Под обращением к оракулу L мы понимаем очень простую вещь: в алгоритме предусмотрена процедура вызова алгоритма разрешения языка L и при подсчёте времени его работы считается, что исполнение этой процедуры занимает 1 такт работы.

Таким образом, из этого утверждения следует, что если $\mathsf{DPG} \in \mathrm{P}$, игры с затухающим платежом решаются за полиномиальное время.

Доказательство утверждения 7.1. Опишем сначала, как находить цену игры.

Если прибавить ко всем весам Δ, то результат любой партии игры изменится на

$$(1-\lambda)\sum_{i=0}^{\infty}\lambda^i\Delta = \Delta$$

(значит, и цена изменится на Δ).

Аналогичный подсчёт показывает, что результат любой партии DPG лежит в интервале $[-W;W]$, где W обозначает максимальную абсолютную величину весов рёбер графа игры.

Поэтому, используя проверку цены DPG со сдвинутыми весами на положительность, методом деления пополам можно приблизить цену циклической игры с точностью ε за $\mathrm{poly}(n, \log W/\varepsilon)$ проверок цены игры на положительность.

Мы ранее доказали, что для DPG цена гарантируется позиционными стратегиями. Поэтому она рациональная:

$$\begin{aligned}\mathrm{val}(G) = (1-\lambda)\sum_{i=0}^{k-1}\lambda^i w(v_i, v_{i+1}) + (1-\lambda)\lambda^k\sum_{i=0}^{\infty}\lambda^{i\ell}w(C) = \\ = (1-\lambda)\sum_{i=0}^{k}\lambda^i w(v_i, v_{i+1}) + \frac{1-\lambda}{1-\lambda^\ell}\lambda^k w(C),\end{aligned}$$

где ℓ — длина предельного цикла, порождаемого оптимальными стратегиями (см. рис. 7.1), k — количество рёбер, которые нужно пройти перед выходом на предельный цикл, а $w(C)$ — вес предельного цикла, вычисленный с учётом затухания:

$$w(C) = \sum_{i=0}^{\ell-1}\lambda^i w(v_{k+i}, v_{k+i+1}). \tag{7.3}$$

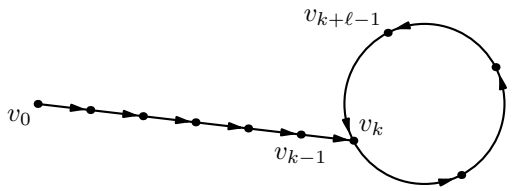


Рис. 7.1: Партия, порождаемая оптимальными стратегиями

Пусть $\lambda = p/q$ — представление параметра затухания в виде несократимой дроби. Тогда знаменатель Q дроби, представляющей $\mathrm{val}(G)$, не превосходит $q^{k+2\ell}p^\ell$.

Как хорошо известно, с помощью разложения в цепную дробь можно точно восстановить рациональное число P/Q, если оно задано двоичной дробью с аддитивной точностью $< 1/(2Q^2)$. Алгоритм такого восстановления работает за полиномиальное от длины входных данных время.

Для точного восстановления цены DPG нужно приблизить её с точностью $\varepsilon \approx 1/2(q^{2(k+2\ell)}p^{2\ell}) > (pq)^{-6n}/2$ (последнее неравенство следует из того, что $k+\ell$ не

превосходит количества вершин n в графе игры). Это потребует

$$\mathrm{poly}(n, \log W/\varepsilon) = \mathrm{poly}(n, \log W + (2k + 6\ell)\max(p, q)) = \mathrm{poly}(\mathrm{size}(\langle G\rangle))$$

вызовов процедуры проверки положительности цены DPG.

Теперь покажем, как найти оптимальные стратегии, используя $\mathrm{poly}(\mathrm{size}(\langle G\rangle))$ вызовов процедуры вычисления цены DPG. Для ребра $(uv) \in E(G)$ построим вспомогательную игру G' на графе, в котором из рёбер графа G, исходящих из вершины u оставлено только ребро (uv), а веса остальных рёбер, параметр затухания и разбиение вершин на V_0, V_1 оставлены теми же самыми. Легко понять, что если $\mathrm{val}(G') = \mathrm{val}(G)$, то ребро (uv) можно включить в оптимальную стратегию.

Продолжая такие проверки, после не более чем $|E(G)|$ итераций полностью определим оптимальные стратегии для игры G. □

Аналогичное утверждение справедливо и для циклических игр.

Утверждение 7.2. *Существует алгоритм решения циклических игр, который имеет возможность обращаться к оракулу* MPG *и работает за полиномиальное время.*

Доказательство. Аналогично предыдущему. Теперь знаменатель дроби, представляющей цену игры, будет не больше длины ℓ предельного цикла. □

7.2. Сводимости

Обозначим через PG язык, состоящий из описаний тех игр чётности, в которых выигрывающая стратегия есть у Чёта.

Лемма 7.3. $\mathsf{PG} \leq_p \mathsf{MPG}$.

Доказательство. Сводимость уже фактически приведене в лемме 5.6). Есть лишь один нюанс. Если приоритеты в игре чётности записаны в двоичной системе, эта сводимость не обязательно полиномиальна: длина записи числа $M^{p(u,v)}$ может оказаться слишком велика, если $p(u, v)$ сверхполиномиально по величине.

Решается это затруднение очень просто: любая игра чётности сводится к игре чётности с $d = O(n^2)$, где d — максимальный приоритет, а n — количество вершин в графе игры. Действительно, результат партии игры чётности зависит только от порядка и чётности приоритетов, а не от точных их значений. Поэтому любой набор приоритетов можно «сжать» до набора из $\leqslant 2|E(G)|$ приоритетов. □

Таким образом, игры чётности «не сложнее» циклических игр. Сейчас проверим, что циклические игры «не сложнее» игр с затухающим платежом.

Лемма 7.4. $\mathsf{MPG} \leq_p \mathsf{DPG}$.

Доказательство. Пусть G — циклическая игра. Её цена, как уже сказано выше, — это рациональное число со знаменателем, не превосходящим n — количества вершин в графе игры G.

Преобразуем веса рёбер графа игры G по правилу

$$w'(u,v) = 2nw(u,v) - 1 \tag{7.4}$$

и получим новую игру G'. Цена новой игры выражается через цену старой как $\operatorname{val}(G') = 2n \operatorname{val}(G) - 1$. Значит, если $\operatorname{val}(G) > 0$, то $\operatorname{val}(G') \geqslant 2n/n - 1 = 1$. А если $\operatorname{val} G \leqslant 0$, то $\operatorname{val}(G') \leqslant -1$.

Теперь построим игру G'' с затухающим платежом, в которой все данные точно такие же, как в игре G'. Докажем, что если параметр затухания λ достаточно близок к 1, то цена игры G положительная тогда и только тогда, когда цена игры G'' положительная.

Пусть цена игры G' положительная, Макс придерживается в игре G'' оптимальной позиционной стратегии для игры G, гарантирующей цену игры G; а Мин придерживается оптимальной позиционной стратегии в игре G''. Партия такой игры выходит на предельный цикл, как на рис. 7.1. Обозначим сумму весов k рёбер до вхождения в цикл через A, сумму положительных рёбер на цикле через P, а сумму неположительных рёбер через $-N$. Длину цикла, как и выше, обозначаем через ℓ.

Если $\varepsilon = 1 - \lambda > 0$ достаточно мало, то вклад каждого прохождения предельного цикла в результат игры G'' с затухающим платежом положительный. Действительно, вклад каждого положительного ребра (u_+, v_+) не меньше $\lambda^\ell w'(u_+, v_+)$, а вклад каждого неположительного ребра (u_-, v_-) не меньше $w'(u_-, v_-)$. Поэтому вес предельного цикла, вычисленный с учётом затухания по формуле (7.3), не меньше $\lambda^\ell P - N = (P-N) - (1-\lambda^\ell)P$, то есть, не меньше обычного веса цикла, уменьшенного на $(1-\lambda^\ell)P$.

Как и выше, обозначаем максимум модуля весов рёбер в игре G через W. В этих обозначениях $P \leqslant 2n\ell W$ (см. (7.4)). Значит, если

$$(1-\lambda^\ell)2n\ell W < \frac{1}{2}\,, \tag{7.5}$$

то вес предельного цикла не меньше $1/2$, а вклад $(i+1)$-го прохождения цикла в результат партии игры G'' не меньше $\lambda^{i\ell}/2$.

Оценим вклад k рёбер, не входящих в цикл: $|A| \leqslant nk(2W+1)$. Это означает, что если

$$\frac{1}{2} \cdot \lambda^{4nk(2W+1)\ell} > \frac{1}{4}, \tag{7.6}$$

то результат рассматриваемой партии игры G'' будет положительным: прохождения первых $4nk(2W+1)$ циклов дадут положительный вклад больше $nk(2W+1)$, что компенсирует возможный отрицательный вклад A.

Но в этой партии Мин следовал оптимальной для игры G'' стратегии. Поэтому цена игры G'' при выполнении условий (7.5) и (7.6) будет положительной.

Для оценки требуемого ε применим неравенство Бернулли $(1+x)^n \geqslant 1 + nx$. Если

$$\varepsilon < \frac{1}{4n\ell^2 W}\,,$$
$$\varepsilon < \frac{1}{8nk\ell(2W+1)}\,,$$

то оба неравенства (7.5) и (7.6) выполняются. Поскольку $k \leqslant n$, $\ell \leqslant n$, при $\varepsilon < \bigl(8n^3(2W+1)\bigr)^{-1}$ эти неравенства выполняются.

Аналогичные вычисления справедливы и для случая, когда цена игры G' отрицательная. Нужно рассмотреть партию, в которой Макс придерживается оптимальной для игры G'' позиционной стратегии, а Мин — оптимальной позиционной стратегии для игры G.

Поэтому сводимость $G \mapsto G''$, где в G'' параметр затухания выбран как $\lambda = 1 - \bigl(8n^3(2W+1)\bigr)^{-1}$, является полиномиальной сводимостью MPG к DPG: длина записи λ равна $O(\log \varepsilon^{-1}) = O(\log n + \log W) = \mathrm{poly}(\mathrm{size}(\langle G \rangle))$. □

7.3. Принадлежность классу NP ∩ co-NP

Теорема 7.5. $\mathsf{DPG} \in \mathrm{NP} \cap \mathrm{co\text{-}NP}$.

В силу сводимостей

$$\mathsf{PG} \leq_p \mathsf{MPG} \leq_p \mathsf{DPG}$$

получаем следствия:

Следствие 7.6. $\mathsf{MPG} \in \mathrm{NP} \cap \mathrm{co\text{-}NP}$.

Следствие 7.7. $\mathsf{PG} \in \mathrm{NP} \cap \mathrm{co\text{-}NP}$.

Доказательство теоремы 7.5. Как мы проверили в разделе 6.1, цены игр с затухающим платежом на одном и том же графе с разными начальными позициями i удовлетворяют системе уравнений (6.3):

$$\mathrm{val}_i = \begin{cases} \max\limits_{(ij)\in E} (\lambda\, \mathrm{val}_j + (1-\lambda) w(i,j)), & \text{если } i \in V_1; \\ \min\limits_{(ij)\in E} (\lambda\, \mathrm{val}_j + (1-\lambda) w(i,j)), & \text{если } i \in V_0. \end{cases}$$

Более того, аргмаксимумы и аргминимумы в этих равенствах задают оптимальные стратегии игроков (утверждение 6.4).

Если заданы значения val_i, за полиномиальное время можно проверить выполнение равенств (6.3). Поскольку неподвижная точка у отображения (6.2) пересчёта цен конечных игр единственная, значения val_i будут совпадать с ценами игр, начинающихся в разных начальных позициях.

Теперь уже теорема становится очевидной: сертификатом и для DPG и для co-DPG будет набор val_i, удовлетворяющий (6.3). Напомним, что цены игр рациональные. Выше мы уже оценили их знаменатели, длины их записи полиномиально ограничены длиной описания игры. Числители также полиномиально ограничены длиной описания игры, так как цена игры не превосходит максимума модуля весов рёбер. Поэтому длина сертификата полиномиальна. В каждом случае нужно для указанной начальной вершины i сравнить val_i с 0, что требует полиномиально ограниченного времени. □

Упражнение 12. Докажите, что если язык $L \in \mathrm{NP} \cap \text{co-NP}$ является NP-полным, то $\mathrm{NP} = \text{co-NP}$.

Одной из гипотез теории сложности является $\mathrm{NP} \neq \text{co-NP}$. В таком случае решение игр чётности не является NP-полной задачей. Однако до сих пор неизвестны полиномиальные алгоритмы решения игр чётности. Это объясняет интерес к данному виду игр. (Отметим также, что у них есть приложения к верификации логических теорий и программ.)

Недавно появились алгоритмы решения игр чётности, которые работают за квазиполиномиальное время $N^{O(\log N)}$, где N — длина входа, то есть описания игры. Это ещё один аргумент против NP-полноты решения игр чётности, так как популярная гипотеза ETH противоречит существованию квазиполиномиальных алгоритмов для NP-полных задач.

Мы обсудим эти алгоритмы ниже. А пока проанализируем те алгоритмы, которые получаются непосредственно из отображений пересчёта цен.

7.4. Псевдополиномиальные алгоритмы

Напомним, что при рассмотрении алгоритмов решения циклических игр мы предполагаем, что веса рёбер целочисленные, и обозначаем максимум модуля весов рёбер через W. Параметр затухания λ предполагается рациональным. Считаем, что он задан обыкновенной дробью p/q. Количество вершин в графе игры обозначаем через n.

Анализ доказательства теоремы Колберга и отображений пересчёта цен позволяет построить алгоритмы вычисления цены игры, которые работают за время $\mathrm{poly}(n, W, p, q)$. Такие алгоритмы называются *псевдополиномиальными*.

Для циклических игр анализ проще, поэтому начнём с него.

Утверждение 7.8. *Существует псевдополиномиальный алгоритм решения циклических игр.*

Доказательство. Вычисления по формулам (6.4)

$$\begin{aligned}
\mathrm{val}(v, T) &= 0;\\
\mathrm{val}(u, t) &= \min_{v:(u,v)\in E} \big(w(u,v) + \mathrm{val}(v, t+1)\big), \quad \text{если } u \in V_0,\ 0 \leqslant t < T;\\
\mathrm{val}(u, t) &= \max_{v:(u,v)\in E} \big(w(u,v) + \mathrm{val}(v, t+1)\big), \quad \text{если } u \in V_1,\ 0 \leqslant t < T
\end{aligned}$$

возможны за время, полиномиально ограниченное n и T, что позволяет достаточно быстро вычислять цены игр на T ходов. Считаем для простоты платёж в конечной игре аддитивным, то есть не делим результат на T.

Докажем неравенства

$$T\,\mathrm{val} - 2nW \leqslant \mathrm{val}_T \leqslant T\,\mathrm{val} + 2nW.$$

Действительно, пусть в игре на T ходов Мин следует оптимальной позиционной стратегии в бесконечной циклической игре. Рассмотрим некоторую партию из T

ходов, она задаёт маршрут по графу. Этот маршрут разбивается на простые циклы и не более чем n рёбер, не входящих в простые циклы. Вклад каждого простого цикла не больше val, а вклад каждого из остальных рёбер не больше W. Отсюда получаем оценку

$$\mathrm{val}_T \leqslant (T-n)\,\mathrm{val} + nW \leqslant T\,\mathrm{val} + 2nW,$$

так как цена игры по модулю не превосходит W.

Аналогично доказываем второе неравенство, рассматривая оптимальную стратегию Макса.

Итак, цену в циклической игре можно ограничить с двух сторон как

$$\frac{\mathrm{val}_T}{T} - \frac{2nW}{T} \leqslant \mathrm{val} \leqslant \frac{\mathrm{val}_T}{T} + \frac{2nW}{T}$$

Цена игры — это средний вес цикла, то есть рациональное число со знаменателем не больше n. Поэтому из решения циклической игры за T ходов, где $T > 4n^3W$, цена игры восстанавливается по написанному выше приближению. Это и даёт псевдополиномиальный алгоритм решения циклических игр. □

Сводимость игр чётности к циклическим играм и описанный выше псевдополиномиальный алгоритм дают вместе алгоритм решения игр чётности, который работает за время $n^{O(d)}$. Эта оценка экспоненциальная в общем случае. Но она становится полиномиальной, если $d = O(1)$. Впрочем, те алгоритмы решения игр чётности, который мы рассмотрим далее, будут более эффективными и в случае $d = O(1)$.

Аналогичный анализ для игр с затухающим платежом усложняется арифметическими трудностями, которые возникают из-за параметра затухания.

Утверждение 7.9. *Существует псевдополиномиальный алгоритм решения игр с затуханием.*

Доказательство. В доказательстве леммы 6.3 мы не только доказали, что отображение f пересчёта цен

$$f(x)_i = \begin{cases} \max\limits_{(ij)\in E} (\lambda x_j + (1-\lambda) w(i,j)), & \text{если } i \in V_1; \\ \min\limits_{(ij)\in E} (\lambda x_j + (1-\lambda) w(i,j)), & \text{если } i \in V_0 \end{cases}$$

является сжимающим относительно нормы ℓ_∞, но и оценили коэффициент сжатия: он не больше λ.

Обозначим неподвижную точку отображения f через val (это вектор цен игр, начинающихся в разных позициях графа игры).

Рассмотрим итерации этого отображения, начинающиеся с нуля: $x^{(t)} = f^{(t)}(0)$. Оценим отклонение итераций от неподвижной точки:

$$\|\,\mathrm{val} - x^{(t)}\|_\infty = \|f^{(t)}(\mathrm{val}) - f^{(t)}(0)\|_\infty < \lambda^t \|\,\mathrm{val} - 0\| \leqslant \lambda^t W.$$

В последнем неравенстве мы использовали тот факт, что цена DPG по абсолютной величине не превосходит W (аналогично вычислению в доказательстве утверждения 7.1). В том же доказательстве мы оценили точность, с которой требуется

приблизить цену игры, чтобы затем восстановить её точно методом цепных дробей: $\varepsilon \approx (pq)^{-6n}/2$.

Сравнивая эти оценки, заключаем, что требуемая точность достигается после $t = O(p(n \log(pq) + \log W))$ итераций. Это количество псевдополиномиально зависит от размера входных данных.

Однако, чтобы такой алгоритм вычисления цены игры с помощью последовательных итераций отображения пересчёта цен был псевдополиномиальным, нужно ещё убедиться, что координаты векторов в последовательности итераций задаются не слишком длинными числами (чтобы арифметические вычисления при вычислении итерации занимали не слишком много времени).

Поскольку формулы линейные, проблем с длинной арифметикой не возникает. Проверим это.

Обозначим через L_t максимальную длину записи одной из координат вектора $x^{(t)}$. Из формул пересчёта цен получаем верхнюю оценку на длину записи знаменателя q^t. Все цены в играх за конечное время по абсолютной величине не превосходят W (формулы пересчёта цен показывают, что следующая цена является выпуклой комбинацией какой-то из предыдущих цен и веса ребра). Значит, числитель по абсолютной величине не больше Wq^t. Поэтому $L_t \leqslant 2t \log q + \log W$.

Итак, при вычислениях итераций до достижения требуемой точности, нужно выполнять арифметические действия с числами, двоичные записи которых не превосходят $O(p \log q(n \log(pq) + \log W))$. Эта величина псевдополиномиально зависит от размера входных данных. □

Следствие 7.10. *DPG на n-вершинном графе с параметром затухания $1 - 1/n$ может быть решена за полиномиальное время.*

Задача 7.1. На самом деле, решать игры с затухающим платежом можно и без арифметических вычислений с очень длинными числами.

Постройте псевдополиномиальный алгоритм решения игр с затухающими платежами, в котором все арифметические операции выполняются с числами, длина записи которых не превосходит $\mathrm{poly}(n, \log(pqW))$ (количество таких арифметических операций может быть псевдополиномиально велико).

Тема 8

Игры достижимости, игры чётности, игры с судьёй

Начиная с 2017 года, появилось несколько квазиполиномиальных алгоритмов решения игр чётности. Большинство из них используют, явно или неявно, сводимость игр чётности к играм достижимости. Отметим, однако, что в апреле 2019 P. Parys предложил ещё один квазиполиномиальный алгоритм решения игр чётности [14]. Этот алгоритм основан на другой идее.

8.1. Игры с судьёй

Чтобы описать сводимость игр чётности к играм достижимости, нам нужна игра по другим правилам. Представим такую ситуацию. Чёт и Нечет играют себе на графе игры чётности, а за их игрой следит Судья. Судья в некоторый момент может присудить победу Нечету. Если он этого не сделал в течение всей бесконечной партии, то Нечет проиграл.

Про Судью мы сделаем два предположения: его действия детерминированные и он имеет конечную память. Попросту говоря, Судья — это конечный детерминированный автомат $\mathcal{A}$ с алфавитом $\Sigma = E \times \{0, 1, \ldots, d-1\}$ (мы здесь и всюду далее обозначаем через $d-1$ максимальный приоритет, минимальный полагаем равным 0;

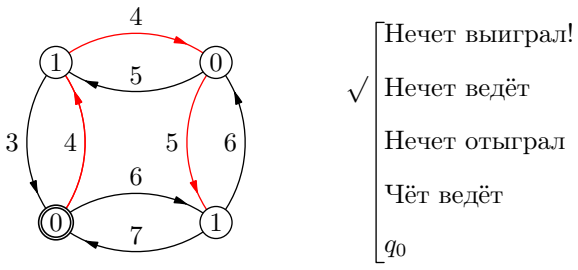


Рис. 8.1: Игра с Судьёй

считаем без ограничения общности, что d чётно). Конечность означает, что различных состояний Судьи конечное число (скажем, на рисунке 8.1 их 5), множество состояний мы обозначим $Q(\mathcal{A})$. Детерминированность означает, что после каждого хода (по ребру e с приоритетом $p(e)$) Судья изменяет своё состояние в соответствии с функцией переходов $\delta_{\mathcal{A}}\colon Q \times \Sigma \to Q$. То есть, если состояние Судьи было q, то новое состояние будет $\delta_{\mathcal{A}}(q, (e, p(e))$ (алфавит автомата состоит из пар «ребро, приоритет»). В некотором состоянии Судья присуждает победу Нечету (на рисунке 8.1 самое верхнее). Мы обозначим такое состояние q_{odd} (принимающее состояние автомата).

Кроме того, нужно указать начальное состояние q_0, в котором Судья начинает следить за партией (на рисунке 8.1 самое нижнее).

Получаем новый вид игр: игры с судьёй-автоматом. На самом деле, это те же самые игры достижимости, которые уже были раньше. Действительно, по игре чётности G и автомату $\mathcal{A}$ построим *расширенный граф* $G \times \mathcal{A}$. У этого графа множество вершин $V \times Q$, то есть пары «вершина графа, состояние автомата». Рёбра расширенного графа — это упорядоченные пары пар вида $\big((u, q'), (v, q'')\big)$, для которых $(u, v) \in E$, $q'' = \delta_{\mathcal{A}}(q', (e, p(u, v)))$.

Определим целевое множество Нечета как $\mathrm{Win} = V(G) \times Q_a$.

Задача 8.1. Проверьте, что существование выигрывающей стратегии для одного из игроков в игре достижимости на расширенном графе со стартовой вершиной (v_0, q_0) равносильно существованию выигрывающей стратегии для того же игрока в игре с Судьёй со стартовой вершиной v_0.

Замечание 8.1. Мы сформулировали не самый общий вариант игр с Судьёй. Судья ограничен в своих возможностях, а игроки нет. Поэтому в более общем варианте игры игроки могут комментировать свои ходы и ходы противника («но тут же стояла ладья!»). Если множество таких комментариев конечно, то такой игре также будет соответствовать автомат и игра достижимости на расширенном графе. Теперь в расширенный граф нужно добавить вспомогательные позиции, в которых игроки выбирают свои комментарии. Чтобы комментарии были доступны Судье, нужно добавить их в алфавит автомата.

Однако извлечь пользу из этих расширенных возможностей пока никому не удалось. Более того, во всех дальнейших играх Судья будет смотреть только на приоритеты, а не на рёбра.

Пока неясно, какое отношение имеют игры с судьями к играм чётности. Для произвольного автомата, конечно же, никакой связи и нет. Однако, если сконструировать автомат подходящим образом, то можно достичь выполнения следующих условий:

корректность: если у Нечета есть выигрывающая стратегия в игре чётности, то у него есть выигрывающая стратегия в игре с Судьёй;

полнота: если у Чёта есть выигрывающая стратегия в игре чётности, то у него есть выигрывающая стратегия в игре с Судьёй.

Обратите внимание, что корректность и полнота автомата зависят, вообще говоря, от игры чётности.

Таким образом, если судья-автомат корректный и полный, то решение игры чётности сводится к решению игры достижимости на расширенном графе. Решение игры достижимости возможно за время, полиномиально ограниченное размером графа и размером автомата. То есть, для хорошего алгоритма решения игры чётности нужны корректные и полные автоматы со сравнительно небольшим количеством состояний.

Существование каких-нибудь корректных и полных автоматов для любой игры чётности легко вывести из существования позиционных стратегий в игре чётности.

Пример 9. Представим, что Судья следит за тем, чтобы игроки придерживались позиционных стратегий. То есть, если из одной и той же позиции игрок сделал два разных хода, Судья тут же объявляет этого игрока проигравшим. В случае Нечета это формально будет означать, что автомат переходит в поглощающее состояние q_{ev} (чтение любого символа не меняет такое состояние), не принадлежащее множеству Q_a.

Кроме того Судья отсчитывает n ходов (здесь n количество вершин графа игры), после чего определяет максимальный приоритет за следующие n ходов. Если этот максимальный приоритет p^* чётный, Судья переходит в состояние q_{ev} и никогда не присуждает победу Нечету. А если p^* нечётный, Судья переходит в состояние q_{odd} и присуждает победу Нечету.

Корректность и полнота такого Судьи вполне очевидны. Отклонение от позиционной стратегии карается немедленным проигрышем. Поэтому каждый из игроков вынужден придерживаться позиционной стратегии. После $\leqslant n$ ходов партия выйдет на цикл. Если Чёт придерживается выигрывающей стратегии, максимальный приоритет за следующие n ходов (такое количество гарантирует, что все рёбра цикла будут пройдены) будет чётным и Чёт выиграет (полнота). Аналогично, если Нечет придерживается выигрывающей позиционной стратегии, максимальный приоритет будет нечётным и Нечет выиграет (корректность).

Какое количество состояний у такого автомата? Нужен счётчик до n, нужно помнить максимальный текущий приоритет. Кроме того, нужно следить за тем, что игроки следуют позиционной стратегии. И вот для этого требуется большая память: нужно помнить для каждой вершины, какой ход в ней уже сделал игрок. Таким образом, количество состояний у такого автомата при прямолинейной реализации будет $O(dn^{n+1})$. Такой автомат совершенно бесполезен: решение соответствующей игры достижимости не лучше прямого перебора всех возможных позиционных стратегий.

8.2. Автомат с несколькими счётчиками

Есть несложный способ улучшить автомат из примера 9.

Пусть Судья использует несколько счётчиков, по одному для каждого нечётного

приоритета $p < d$ (напомним, что d чётно, а приоритеты принимают значения из $[0, d-1]$).

Формально нужный нам автомат состоит из произведения однотипных автоматов $\mathcal{A}_p$, где p нечётное.

Каждый автомат $\mathcal{A}_p$ подсчитывает, сколько раз встретился приоритет p с тех пор, как не встречалось приоритетов выше p. То есть, при чтении приоритета p счётчик увеличивается на 1. При чтении приоритета больше p счётчик сбрасывается в 0.

Если значение счётчика становится равным $n = |V(G)|$, наступает переполнение счётчика и автомат $\mathcal{A}_p$ переходит в принимающее состояние q_{odd} (Судья присуждает выигрыш Нечету).

Докажем, что если в игре достижимости на графе $G \times \prod_p \mathcal{A}_p$ игрок i придерживается позиционной выигрывающей стратегии в игре чётности, то этот игрок выигрывает и в игре достижимости.

Рассмотрим какую-то партию игры достижимости, в которой Нечет придерживается выигрывающей стратегии в игре чётности. Тогда $p = \limsup p(v_t, v_{t+1})$ — нечётное число. Поэтому счётчик автомата $\mathcal{A}_p$ рано или поздно переполнится.

Рассмотрим какую-то партию игры достижимости, в которой Чёт придерживается позиционной выигрывающей стратегии в игре чётности. Докажем от противного, что ни один счётчик не переполняется. Действительно, в противном случае партия прошла n раз через рёбра с приоритетом p, остальные приоритеты на этом отрезке меньше p, а какие-то из концов этих рёбер (их не меньше $n+1$) совпадают. Получили цикл, на которым наивысший приоритет нечётный. Тогда Нечет может форсировать выигрыш в игре чётности, продолжая двигаться по этому циклу (рисунок 8.2). Противоречие.

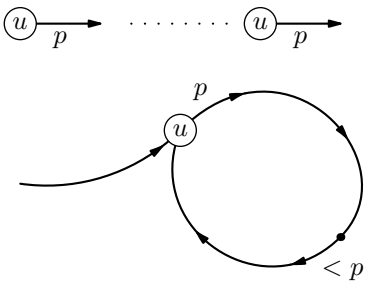


Рис. 8.2: Переполнение счётчика влечёт выигрыш Нечета

Количество состояний у такого автомата $\theta(n^{1+d/2})$. Так что такой автомат уже даёт алгоритм решения игр чётности, который лучше алгоритма, получающегося из псевдополиномиального алгоритма решения циклических игр.

8.3. Абстрактные счётчики на упорядоченных деревьях

Для квазиполиномиального алгоритма нужно обобщение предыдущей конструкции. Будем рассматривать автоматы с одним счётчиком, но правила изменения этого счётчика будут намного сложнее.

По-прежнему автомат обращает внимание только на приоритеты рёбер, возникающих в партии, но не на сами рёбра. Состояния автомата будут линейно упорядочены, причём максимальное состояние будет как раз целевым для Нечета состоянием q_{odd}.

Изменения состояний управляются корневым упорядоченным однородным деревом T глубины $d/2$. Напомним, что в корневом дереве выделена специальная вершина: корень; в упорядоченном корневом дереве дочки каждой вершины линейно упорядочены; в однородном дереве длина пути из корня в лист одна и та же для всех листьев (и она называется *глубиной* дерева).

В корневом дереве выделяем *слои* (одно поколение потомков корня) — вершины на одинаковом расстоянии от корня. В однородном дереве листья образуют слой.

В нужных нам деревьях будем нумеровать слои, отличные от корня, нечётными числами от $d-1$ до 1, начиная от корня (чем ближе к корню, тем выше приоритет). На рисунке 8.3 изображён пример такого дерева для $d = 6$.

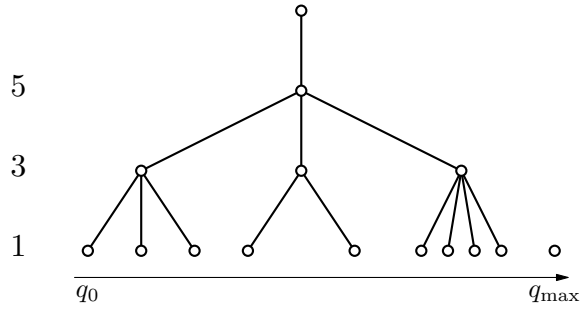


Рис. 8.3: Листья упорядоченного дерева — состояния автомата

Листья $L(T)$ этого дерева образуют состояния автомата $\mathcal{A}_T$. Помимо них есть ещё особое состояние $q_{\max} = q_{\mathrm{odd}}$, которое больше всех листьев. Оно и является принимающим (фиксирует победу Нечета). Начальное состояние q_0 соответствует минимальному листу в дереве T.

Определим *обрезку* $v|_p$ листа v как вершину на пути от корня к листу v на уровне p, если p нечётно, на уровне $p+1$, если p чётно. См. рисунок 8.4.

Обрезка нам нужна для определения функции переходов $\delta(q,p)$ автомата $\mathcal{A}_T$:

$$\delta(q,p) = \begin{cases} \min(q' : q'|_p = q|_p), & (p \text{ чётно}), \\ \min(q' : q'|_p > q|_p), & (p \text{ нечётно и } q|_p \text{ не максимально на слое } p), \\ q_{\mathrm{odd}}, & (p \text{ нечётно и } q|_p \text{ максимально на слое } p). \end{cases} \quad (8.1)$$

Эти правила проиллюстрированы на рисунке 8.5.

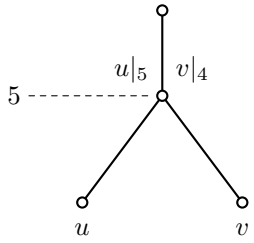


Рис. 8.4: Примеры обрезки листьев

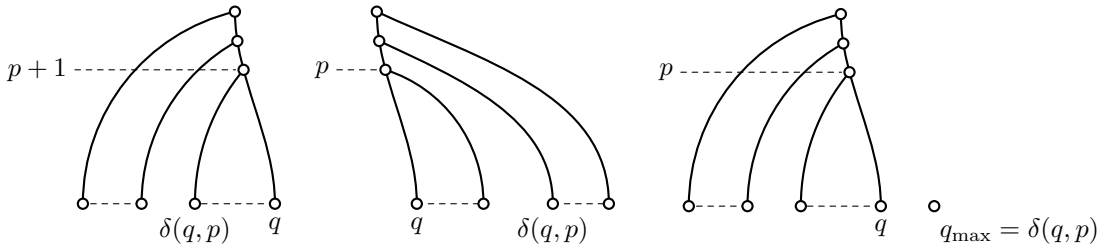


Рис. 8.5: Переходы автомата $\mathcal{A}_T$

Утверждение 8.1. *Функция переходов автомата $\mathcal{A}_T$ монотонна по состояниям: для любого p и любых двух состояний $q_1 \leqslant q_2$ выполняется неравенство $\delta(q_1, p) \leqslant \delta(q_2, p)$.*

Доказательство. Заметим, что обрезки монотонны: если $q_1 \leqslant q_2$, то $q_1|_p \leqslant q_2|_p$. В обратную сторону слабее: если $q_1|_p < q_2|_p$, то $q_1 < q_2$.

Проверим для чётного p и $q_1 \leqslant q_2$ неравенство

$$\min(q' : q'|_p = q_1|_p) \leqslant \min(q' : q'|_p = q_2|_p).$$

Если $q_1|_p < q_2|_p$, то каждый элемент множества, по которому берётся минимум в левой части, меньше любого элемента множества, по которому берётся минимум в правой части.

Если же $q_1|_p = q_2|_p$, то множества совпадают, так что минимумы равны.

Проверим для нечётного p и $q_1 \leqslant q_2$ неравенство

$$\min(q' : q'|_p > q_1|_p) \leqslant \min(q' : q'|_p > q_2|_p).$$

Если $q_1|_p \leqslant q_2|_p$, множество из левой части неравенства содержит множество из правой части неравенства. Минимум не увеличивается при расширении множества. □

Правила построения автомата по дереву гарантируют как минимум корректность такого автомата.

Лемма 8.2 (корректность любого автомата из дерева)**.** *Для любого графа игры G и любого упорядоченного дерева T верно, что если Нечет имеет выигрывающую стратегию в игре чётности на графе G со стартовой вершиной v_0, то он имеет*

выигрывающую стратегию в игре с судьёй $\mathcal{A}_T$ на графе G и стартовой вершиной v_0.

Доказательство. Проверим, что если Нечет придерживается своей выигрывающей стратегии в игре чётности, когда играет в игру с судьёй-автоматом $\mathcal{A}_T$, то он побеждает в любой партии. В такой партии верхний предел приоритетов нечётен. Обозначим его p и рассмотрим партию с того момента, когда приоритеты выше p не встречаются.

Каждый раз, когда автомат делает переход по ребру с приоритетом p, обрезка его состояния на уровне p увеличивается. Когда автомат делает переход с чётным приоритетом (который при сделанных предположениях меньше p), обрезка его состояния на уровне p не изменяется. Когда автомат делает переход с нечётным приоритетом, меньшим p, обрезка его состояния на уровне p не уменьшается.

Таким образом, рано или поздно автомат окажется в состоянии q_{odd}. □

Обратное утверждение для произвольного дерева неверно. Возьмём, например, дерево из одного пути. Тогда любой ход по ребру с нечётным приоритетом переведёт состояние автомата в q_{odd}. То есть, Нечет выигрывает в игре с судьёй, если только он в состоянии добиться хода по ребру с нечётным приоритетом. Нетрудно построить примеры игр чётности, в которых выигрывающая стратегия у Чёта, но Нечет всегда может добиться прохождения по ребру с нечётным приоритетом.

Упражнение 13. Приведите пример такой игры.

8.4. Полнота автомата на дереве сильно связных компонент стратегического графа

Мы сейчас построим для игры чётности G, в которой есть выигрывающая стратегия $x\colon V_0 \to V$ у Чёта, такое дерево небольшого размера, что автомат, построенный по этому дереву, будет полным (и корректным, как всякий автомат, см. лемму 8.2).

Напомним, что стратегии x соответствует стратегический граф G_x, который получается удалением всех ходов Чёта, не входящих в стратегию (см. раздел 5.3). Сейчас нам удобнее несколько изменить определение этого графа. Будем считать, что при построении G_x также удаляются все позиции и рёбра, недостижимые из начальной.

Основное свойство этого графа, которое мы уже фактически использовали выше: в стратегическом графе Чёта нет нечётных циклов. Действительно, ходы Чёта полностью определены стратегией. Если в стратегическом графе есть нечётный цикл, то Нечет может выйти на этот цикл и двигаться по нему неограниченно долго. В такой партии Нечет выигрывает, что противоречит выбору стратегии Чёта.

Граф $G_x(p)$ получается из графа G_x удалением всех рёбер с приоритетами выше p.

Напомним, что отношение *сильной связанности* в ориентированном графе означает, что из вершины u достижима вершина v и наоборот. Такое отношение рефлексивно, симметрично (по определению) и транзитивно (как легко видеть). То есть, это отношение эквивалентности. Классы эквивалентности этого отношения называются *компонентами сильной связности*.

Утверждение 8.3. *Рёбра с нечётным приоритетом p не лежат в компонентах сильной связности графа $G_x(p)$.*

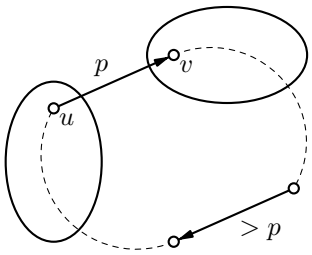


Рис. 8.6: Иллюстрация к утверждению 8.3

Доказательство. Пусть (u, v) — ребро с нечётным приоритетом p. Позиция v очевидно достижима из u. Но u недостижима из v: в противном случае в стратегическом графе G_x есть нечётный цикл. □

Теперь перейдём к самой конструкции дерева, которое мы обозначим T_x. Вершинами дерева удобно считать множества позиций графа игры. Корнем дерева T_x является всё множество вершин $V(G_x)$ стратегического графа. Дочками корня будут компоненты сильной связности графа $G_x(d-1) = G_x$ (напомним, что $d-1$ это максимальный приоритет).

Упорядочим дочки корня следующим образом. *Конденсацией* графа называется граф сильно связных компонент (ребро соединяют пару компонент, для которой существует ребро исходного графа, выходящее из первой компоненты и заходящее во вторую). Легко проверить, что конденсация — это ациклический граф. Его рёбра задают частичный порядок на множестве связных компонент. Считаем, что рёбра конденсации ведут от меньших компонент к бóльшим. Возьмём конденсацию графа $G_x(d-1)$ и упорядочим сильно связные компоненты каким-нибудь линейным продолжением описанного выше частичного порядка (это ещё называется топологической сортировкой или топологическим упорядочением вершин ациклического графа).

Что будет далее важно, из утверждения 8.3 следует, что любое ребро приоритета $d-1$ (нечётного как мы предполагаем во всём этом анализе) ведёт из меньшей компоненты в бóльшую.

Далее продолжаем точно так же. Если U — компонента сильной связности графа $G_x(p+2)$, то дочками U являются компоненты сильной связности графа $G_x(p)$,

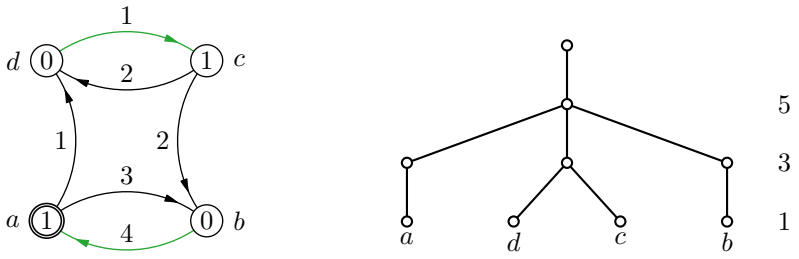


Рис. 8.7: От графа стратегии к дереву связных компонент

которые лежат в U. Дочки упорядочены каким-нибудь линейным продолжением частичного порядка на компонентах связности графа $G_x(p)$.

В построенном дереве T_x слои естественным образом занумерованы нечётными числами. Слой с номером p образован компонентами сильной связности графа $G_x(p)$.

Пример построения дерева по графу изображён на рисунке 8.7 ($d = 6$ в этом примере).

Утверждение 8.4. *В дереве T_x не более $1+dn/2$ вершин, здесь n — количество вершин в графе игры чётности, d — количество приоритетов.*

Доказательство. Каждый слой p, всего $d/2$ штук, соответствует разбиению позиций графа G_x (не более n позиций) на компоненты сильной связности, их тоже не больше n. И ещё есть корень. □

Каждой вершине v стратегического графа сопоставим лист $q(v)$ дерева T_x — ту компоненту сильной связности графа $G_x(1)$, которой эта вершина принадлежит. Лист $q(v)$ задаёт единственный путь из $q(v)$ в корень дерева T_x. Вершину дерева на этом пути на слое p будем обозначать $q(v)|_p$ и называть *обрезкой* на слое p. Кроме того, будем определять обрезку $q(v)|_p$ для чётного p как $q(v)_{p+1}$.

Утверждение 8.5. *Для любого ребра $(u, v) \in E(G_x)$ с приоритетом $p = p(u, v)$ выполняются неравенства*

$$\begin{aligned} &q(v)|_p \geqslant q(u)|_p \\ &q(v)|_p > q(u)|_p\,, \quad \textit{если } p \textit{ нечётно.} \end{aligned} \tag{8.2}$$

Доказательство. При построении дерева T_x мы выбирали такие упорядочения компонент сильной связности графов $G_x(p)$, чтобы рёбра конденсаций шли от меньших компонент к бóльшим.

Отсюда и из утверждения 8.3 следует второе неравенство (ребро с приоритетом p идёт из одной компоненты сильной связности в другую). Первое неравенство может обращаться в равенство для чётного p, если вершины u, v лежат в одной компоненте сильной связности графа $G_x(p+1)$. □

Отсюда легко следует полнота дерева T_x для игры G.

Лемма 8.6. *Если x — выигрывающая позиционная стратегия Чёта в игре чётности, то автомат $\mathcal{A}_x$, построенный по дереву T_x, обладает свойством полноты: у Чёта есть выигрывающая стратегия в игре достижимости на расширенном графе $G \times \mathcal{A}_x$.*

Доказательство. Пусть Чёт придерживается стратегии x в партии

$$v_0, v_1, \ldots, v_t, \ldots .$$

Последовательность состояний автомата $\mathcal{A}_x$, которая возникает в такой партии, обозначим

$$q_0, q_1, \ldots, q_t, \ldots .$$

Докажем индукцией по моментам времени, что в такой партии всегда выполняется неравенство $q_t \leqslant q(v_t) < q_{\text{odd}}$.

База индукции очевидна, так как q_0 наименьшее состояние (лист дерева T_x).

Индуктивный переход получается из неравенств (8.2) утверждения 8.5. Пусть $q_t \leqslant q(v_t)$ и приоритет ребра (v_t, v_{t+1}) равен p.

Если p чётно, то $q(v_{t+1})_p \geqslant q(v_t)|_p$ по (8.2), а $q_{t+1}|_p$ — минимальное состояние, для которого обрезка на уровне p совпадает с $q(v_t)|_p$ (так определена функция переходов $\mathcal{A}_x$, см. (8.1)). Значит, $q_{t+1} \leqslant q(v_{t+1})$.

Если p нечётно, то $q(v_{t+1})_p > q(v_t)|_p$, а $q_{t+1}|_p$ — минимальное состояние, для которого обрезка на уровне p строго больше $q(v_t)|_p$. Значит, $q_{t+1} \leqslant q(v_{t+1})$. □

Итак, судья из автомата T_x получается отменный: корректный, полный и его размер не превосходит $1 + dn/2$, то есть он полиномиально ограниченный.

С этим автоматом одна трудность: чтобы его построить, нужно знать выигрывающую стратегию Чёта. А если знать выигрывающую стратегию, то и автомат, казалось бы, не нужен.

8.5. Сохранение полноты при расширении дерева

Чтобы преодолеть эту трудность, сделаем ещё одно наблюдение о свойствах автоматов, построенных на играх. А именно, проверим, что полнота сохраняется при расширении дерева.

Будем говорить, что корневое упорядоченное дерево T' *вкладывается* в корневое упорядоченное дерево T'', если существует инъективное отображение $V(T') \hookrightarrow V(T'')$ вершин дерева T' в вершины дерева T'', которое корень переводит в корень, сохраняет отношение «быть дочкой» и линейный порядок на дочках.

Лемма 8.7. *Если дерево T' вкладывается в дерево T'' и для игры с судьёй-автоматом $\mathcal{A}_{T'}$ существует выигрывающая стратегия для Чёта, то выигрывающая стратегия для Чёта существует и для игры с судьёй-автоматом $\mathcal{A}_{T''}$.*

Доказательство. Проверим, что выигрывающая стратегия Чёта в игре с автоматом $\mathcal{A}_{T'}$ является выигрывающей и в игре с автоматом $\mathcal{A}_{T''}$.

Рассмотрим партию, которая проходит через позиции

$$v_0, v_1, \ldots, v_t, \ldots .$$

Последовательность состояний автомата $\mathcal{A}_{T'}$, которая возникает в такой партии, обозначим

$$q'_0, q'_1, \ldots, q'_t, \ldots ,$$

а последовательность состояний автомата $\mathcal{A}_{T''}$, возникающую в той же партии, обозначим

$$q''_0, q''_1, \ldots, q''_t, \ldots .$$

Докажем индукцией по числу ходов неравенство $q'_t \geqslant q''_t$. Тогда, поскольку среди q'_t нет максимального состояния q_{odd} (Чёт следует выигрывающей стратегии в игре с автоматом $\mathcal{A}_{T'}$), то этого состояния не будет и среди q''_t. Значит, Чёт выигрывает в такой партии, что и требовалось доказать.

Неравенство $q'_0 \geqslant q''_0$ выполнено, потому что это минимальные листья поддерева и всего дерева соответственно.

Пусть $q'_t \geqslant q''_t$.

Если приоритет p ребра (v_t, v_{t+1}) чётный, то

$$\min_{q \in L(T')} (q : q|_p = q'_t|_p) \geqslant \min_{q \in L(T'')} (q : q|_p = q'_t|_p) \geqslant \min_{q \in L(T'')} (q : q|_p = q''_t|_p),$$

поскольку листья T' образуют подмножество листьев T''. Значит, из определения функции переходов автомата (8.1) следует $q'_{t+1} \geqslant q''_{t+1}$.

Если приоритет p ребра (v_t, v_{t+1}) нечётный, получаем аналогичные неравенства

$$\min_{q \in L(T')} (q : q|_p < q'_t|_p) \geqslant \min_{q \in L(T'')} (q : q|_p < q'_t|_p) \geqslant \min_{q \in L(T'')} (q : q|_p < q''_t|_p).$$

Опять из (8.1) следует $q'_{t+1} \geqslant q''_{t+1}$. □

Лемма 8.7 показывает, что для построения полного автомата необязательно искать именно дерево сильно связных компонент стратегического графа. Достаточно найти накрывающее дерево. А это уже проще.

Тема 9

Универсальные деревья и квазиполиномиальные алгоритмы

Итак, лемма 8.6 говорит, что если у Чёта есть выигрывающая стратегия, то существует такое дерево размера не больше $1+dn/2$, что автомат из этого дерева полный. Лемма о расширении 8.7 показывает, что это свойство сохраняется при вложении деревьев.

Поэтому появляется естественный способ подобрать полное дерево, не зная стратегии: нужно взять такое дерево, в которое вкладываются все деревья с заданными границами на размер и глубину.

Далее окажется, что в качестве характеристики размера дерева удобнее использовать количество листьев, а не общее количество вершин.

Определение 9.1. Корневое упорядоченное дерево U называется (n,h)-*универсальным*, если для любого дерева T глубины h, в котором не больше n листьев, существует вложение в U.

В приложениях универсальных деревьев к решению игр чётности нужны оценки количества вершин универсальных деревьев, а не листьев. Но трудности здесь не возникает: в дереве глубины h количество вершин превосходит количество листьев не более, чем в h раз. Этот дополнительный множитель h ничего не портит.

Получаем лёгкое следствие из предыдущих результатов и определения универсального дерева.

Следствие 9.1. *Если U — (n,h)-универсальное дерево, то автомат, построенный по этому дереву является корректным и полным для любой игры чётности с $2h$ приоритетами.*

Пример (n,h)-универсального дерева легко построить. Рассмотрим полное n-арное дерево глубины h. Ясно, что любое дерево глубины h с $\leqslant n$ листьями в нём содержится. Следствие 9.1 показывает, что автомат, построенный из этого дерева, корректный и полный. Количество листьев в таком дереве n^h. Отсюда получаем ещё одним способом алгоритм решения игр чётности за время $O(n^{d/2+\mathrm{const}})$.

Однако это не единственный пример. Универсальные деревья могут быть намного меньше.

9.1. Почти оптимальная конструкция

Построим семейство деревьев $T(n,h)$ глубины h рекурсивно.

Дерево $T(n,1)$ — это корень с n дочками. Дерево $T(1,h)$ — это путь длины h.

Дерево $T(n,h)$ получается так: из корня выпустим «центральное» ребро, к концу которого приклеим дерево $T(n,h-1)$. Слева от «центрального» ребра приклеим прямо к корню $T(n,h)$ дерево $T(\lceil n/2\rceil,h)$. Такое же дерево приклеим к корню $T(n,h)$ справа от «центрального» ребра. См. рисунок 9.1.

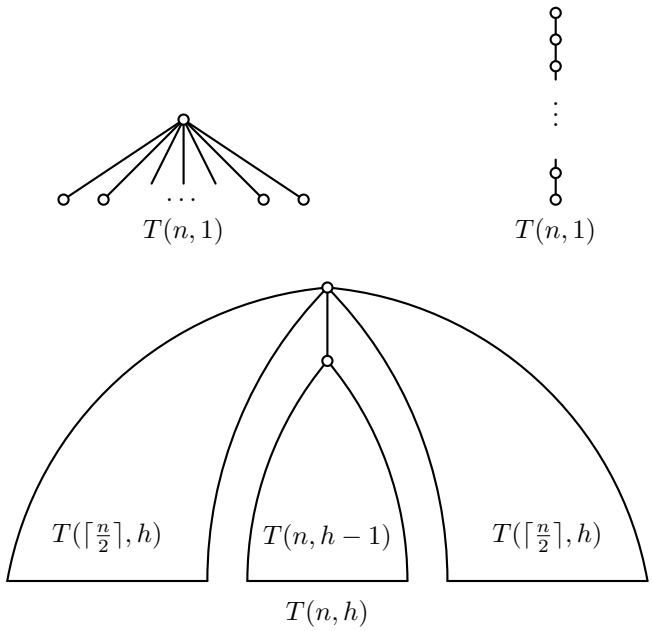


Рис. 9.1: Иллюстрация рекурсивного определения

Лемма 9.2. *Дерево $T(n,h)$ является универсальным для любых n, h.*

Доказательство. Доказательство индукцией по числу листьев и по глубине.

База состоит из двух случаев: $T(n,1)$ и $T(1,h)$. В первом случае листьев в $T_{n,1}$ ровно n, так что $(n,1)$-универсальность очевидна. Во втором случае есть вообще лишь одно дерево глубины h с одним листом (путь), так что универсальность ещё очевиднее.

Предположим, утверждение верно для всех $(n',h') < (n,h)$ (порядок в сравнении покоординатный).

Индуктивный переход. Возьмём дерево T глубины h, в котором не больше n листьев. Обозначим n_i количество листьев в поддереве с корнем в i-й дочке корня

дерева T. Так как

$$n_1 + n_2 + \cdots + n_t \leqslant n,$$

то существует такой индекс i^*, что

$$\sum_{i=1}^{i^*-1} n_i \leqslant \frac{n}{2}\,, \qquad \sum_{i=i^*+1}^{t} n_i \leqslant \frac{n}{2}\,.$$

Разобьём дерево T на три части: центральная — корень, дочка i^* и её потомки (в этой части не больше n листьев); левая — корень, дочки $i < i^*$ и их потомки (в этой части не больше $n/2$ листьев); правая — корень, дочки $i > i^*$ и их потомки (в этой части также не больше $n/2$ листьев). См. рисунок 9.2.

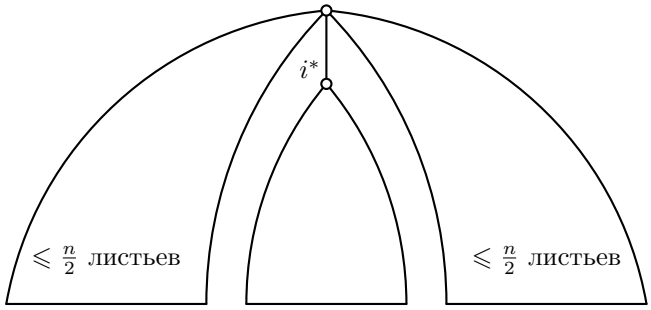


Рис. 9.2: Разделения дерева на три части

По предположению индукции, левая часть T вкладывается в $T(\lceil n/2 \rceil, h)$; поддерево центральной части, начиная от дочки i^* — в $T(n, h-1)$ (отбросили корень дерева T, поэтому высота уменьшилась на 1); правая часть — в $T(\lceil n/2 \rceil, h)$. Вместе это даёт вложение T в дерево $T_{n,h}$. □

9.2. Оценка размера

Теорема 9.3 ([8]). *Количество листьев в дереве $T(n,h)$ не превосходит*

$$2n\binom{\lceil \log_2 n \rceil + h - 1}{h-1}.$$

Будем использовать функцию $\ell(q,h)$, заданную рекуррентными соотношениями

$$\begin{aligned} \ell(0,h) &= 1, \\ \ell(q,1) &= 2^q, \\ \ell(q,h) &= 2\ell(q-1,h) + \ell(q,h-1). \end{aligned} \tag{9.1}$$

Утверждение 9.4. $\ell(q,h) = 2^q\binom{q+h-1}{h-1}$.

Доказательство. Индукция по q и h.

Проверим базу:

$$\ell(0,h) = 1 = 2^0\binom{0+h-1}{h-1}; \quad \ell(q,1) = 2^q = 2^q\binom{q+1-1}{1-1}.$$

Индуктивный переход:

$$\begin{aligned}\ell(q,h) &= 2\ell(q-1,h) + \ell(q,h-1) = \\ &= 2\cdot 2^{q-1}\binom{q-1+h-1}{h-1} + 2^q\binom{q+h-1-1}{h-1-1} = \\ &= 2^q\binom{q+h-1}{h-1}.\end{aligned}$$

□

Доказательство теоремы 9.3. Всюду далее в этом доказательстве $q = \lceil \log_2 n \rceil$.

Докажем индукцией по n, h, что количество листьев в дереве $T(n,h)$ не превосходит $\ell(q,h)$, теорема тогда следует из утверждения 9.4.

База индукции:

$$1 \leqslant \ell(0,h) = 1, \qquad n = 2^{\log_2 n} \leqslant \ell(q,1) = 2^q = 2^{\lceil \log_2 n \rceil}.$$

Индуктивный переход. По предположению индукции и построению дерева $T(n,h)$ количество листьев в дереве $T(n,h)$ не превосходит $2\ell(q',h) + \ell(q,h-1)$, где $q' = \lceil \log_2 \lceil n/2 \rceil \rceil$. Учитывая рекуррентное определение $\ell(q,h)$, осталось проверить, что $q' = q-1$, то есть

$$\left\lceil \log_2 \left\lceil \frac{n}{2} \right\rceil \right\rceil = \lceil \log_2 n \rceil - 1. \tag{9.2}$$

Равенство (9.2) очевидно выполняется при чётном n.

Пусть $n = 2s-1$, где $s = 2^r - \Delta$, $0 \leqslant \Delta < 2^{r-1}$. Тогда левая часть (9.2) равна r. Вычислим правую часть: $n = 2s-1 = 2^{r+1} - 2\Delta - 1$, при этом $1 + 2\Delta < 1 + 2^r$. Но тогда $1 + 2\Delta < 2^r$ (нечётное число заведомо не равно чётному).

Таким образом, правая часть равна $(r+1) - 1 = r$, что и требовалось. □

9.3. Собирая всё вместе

Теорема 9.5. *Существует алгоритм решения игр чётности, который работает за квазиполиномиальное время от длины входа.*

Доказательство. Заметим, что достаточно решать игры чётности с $d = 2n^2$). Действительно, важны только чётности приоритетов и их порядок на разных рёбрах. Сохраняя чётности приоритетов и порядок, можно уменьшить общее количество приоритетов до удвоенного количества рёбер.

Искомый алгоритм строит автомат на универсальном дереве $T(n, d/2)$, где n — количество вершин графа игры чётности, а d — количество приоритетов, и решает получающую игру достижимости.

Из следствия 9.1 заключаем, что ответ в решении такой игры достижимости совпадает с ответом в решении исходной игры чётности.

В силу оценки на размер универсального дерева, размер графа игры достижимости не больше

$$n \cdot 2n \binom{\lceil \log_2 n \rceil + n^2 - 1}{n^2 - 1} = n^{O(\log n)} = 2^{O(\log^2 n)}.$$

Алгоритм решения игры такого размера также работает за время $2^{O(\log^2 n)}$. □

9.4. Нижние оценки на размер универсальных деревьев

Оказывается, построенные в разделе 9.1 универсальные деревья по размерам близки к оптимальным. Это, в частности означает, что получить полиномиальные алгоритмы, совершенствуя игры с автоматами-судьями, весьма проблематично.

Теорема 9.6. *У всякого (n, h)-универсального дерева не меньше* $\binom{\lfloor \log_2 n \rfloor + h - 1}{h - 1}$ *листьев.*

В доказательстве этой теоремы потребуется другая числовая функция, заданная рекуррентно:

$$\begin{aligned} g(1, h) &= 1, \\ g(n, 1) &= n, \\ g(q, h) &= \sum_{d=1}^{n} g(\lfloor n/d \rfloor, h - 1). \end{aligned} \tag{9.3}$$

Утверждение 9.7. $g(n, h) \geqslant \binom{\lfloor \log_2 n \rfloor + h - 1}{h - 1}$.

Доказательство. Из определения нетрудно вывести индукцией по n и h, что функция $g(n, h)$ монотонно возрастает по n и по h.

Поэтому достаточно доказать неравенство только для степеней двойки, когда $n = 2^q$. Оставляя первые два слагаемых в сумме (9.3), получаем

$$g(2^p, h) \geqslant \sum_{k=0}^{p} g(2^{p-k}, h - 1). \tag{9.4}$$

Теперь искомое неравенство легко проверяется индукцией по p и h.

База индукции очевидна:

$$g(1, h) = 1 \geqslant 1; \qquad g(2^p, 1) = 2^p \geqslant \binom{p + 1 - 1}{1 - 1} = 1.$$

Индуктивный переход основан на неравенстве (9.4):

$$\begin{aligned} g(2^p, h) &\geqslant \sum_{k=0}^{p} g(2^{p-k}, h - 1) \geqslant \\ &\geqslant \sum_{k=0}^{p} \binom{p - k + h - 2}{h - 2} = \binom{p + h - 1}{h - 1}. \end{aligned}$$

Последнее равенство в этой выкладке — хорошо известное тождество с биномиальными коэффициентами. □

Для завершения доказательства теоремы 9.6 осталось доказать следующую лемму.

Лемма 9.8. *Во всяком (n, h)-универсальном дереве не меньше $g(n, h)$ листьев.*

Доказательство. Как и прежде, индукция по n и h. База индукции очевидна из определения универсального дерева.

Индуктивный переход. Пусть T — (n, h)-универсальное дерево. Обозначим $f_{\geqslant d}$ количество вершин в дереве T на слое $h-1$, у которых не меньше d потомков.

Докажем, что $f_{\geqslant d} \geqslant g(\lfloor n/d \rfloor, h-1)$. Для этого возьмём произвольное дерево T' глубины $h-1$ с $\lfloor n/d \rfloor$ листьями. Добавим к каждому листу ровно d потомков. Получаем дерево T'' глубины h, в котором не больше n листьев. Значит, это дерево вкладывается в T.

Из построения ясно, что на слое $h-1$ вложение T'' в T использует только вершины с $\geqslant d$ потомками на слое d. Если ограничить это вложение на первые $h-1$ слоёв, получаем вложение T' в усечённое дерево T_{-d}, которое получается из T отбрасыванием всех листьев и всех вершин слоя $h-1$ степени меньше d. В усечённом дереве вершины дерева T, у которых степени $\geqslant d$, становятся листьями.

Из построенного вложения видно, что дерево T_{-d} является $(\lfloor n/d \rfloor, h-1)$-универсальным. По индуктивному предположению, количество листьев в этом дереве не меньше $g(\lfloor n/d \rfloor, h-1)$.

Обозначим f_d количество вершин в дереве T на слое $h-1$, у которых ровно d потомков. Мы доказали, что

$$\sum_{i=d}^{n} f_i \geqslant g(\lfloor n/d \rfloor, h-1).$$

Отсюда получаем оценку на общее количество листьев в дереве T:

$$\sum_{d=1}^{n} f_d \cdot d = \sum_{d=1}^{n} \sum_{i=d}^{n} f_i \geqslant \sum_{d=1}^{n} g(\lfloor n/d \rfloor, h-1) = g(n, h),$$

что и требовалось. (Последнее равенство — это определение $g(n, h)$.) □

Тема 10

Субэкспоненциальные алгоритмы для циклических игр

Пока никому не удалось придумать квазиполиномиальные алгоритмы решения циклических игр или игр с затуханием. Наилучшие известные на данный момент алгоритмы работают за субэкспоненциальное время $2^{O(\sqrt{n})}$, где n — размер входа. Причём эти алгоритмы вероятностные: правильный ответ получается в них с некоторой вероятностью ошибки.

Субэкспоненциальные алгоритмы решения циклических игр, игр чётности и более общих *простых стохастических игр* [11, 15, 5, 6] получаются как частный случай алгоритмов оптимизации на булевом кубе. Такие алгоритмы возникли как обобщение алгоритмов для линейного программирования [13].

10.1. Решение циклических игр: меры успешности стратегии

Рассмотрим циклическую игру на графе G с n вершинами и весами рёбер $w\colon E \to G$. Решаем задачу определения знака цены этой игры. Без ограничения общности считаем, что цена не равна 0 ни для одной начальной позиции. Если это не так, то преобразование (7.4)

$$w'(u,v) = 2nw(u,v) - 1,$$

как показано в доказательстве леммы 7.4, сохраняет положительные знаки цены игры и делает все остальные цены отрицательными.

Будем определять знак цены сразу для всех стратегий. Из основной теоремы о циклических играх следует, что существует равномерная оптимальная позиционная стратегия для Макса, то есть такая позиционная стратегия, которая является оптимальной для любой начальной позиции в графе игры.

Можно искать эту стратегию перебором всех возможных стратегий Макса. Нужно научиться решать две задачи: (1) проверять, что данная стратегия оптимальная; (2) организовать перебор так, чтобы не рассматривать заведомо плохие стратегии.

В работе [6] предложен способ решения обеих задач, основанный на локаль-

ном улучшении стратегий. Идея такая: давайте введём некоторую меру успешности стратегии Макса. Мы хотим, чтобы эта мера достигала максимального значения на оптимальной стратегии, причём проверка оптимальности сводилась к проверке возможности *локального улучшения*: увеличения меры успешности изменением стратегии в одной вершине. После этого остаётся, конечно, проблема: как находить такую локально оптимальную стратегию. Для этой последней задачи есть общий метод решения, приводящий к субэкспоненциальным алгоритмам.

Начнём с того, что определим нужную меру успешности. Для этого мы модифицируем игру. Добавим в каждой вершине Макса ход веса 0 в дополнительную терминальную позицию $\triangle$, в которой есть петля веса 0. Неформально это означает, что Макс отказывается от дальнейшей борьбы за достижение положительного результата партии: если игра перешла в позицию $\triangle$, то результат в партии будет 0. Стратегия Макса теперь может включать ход в терминальную позицию $\triangle$.

В такой модифицированной игре цена позиции отрицательная только тогда, когда эта позиция Мина и на подграфе игры, порождённом ходами Мина из этой позиции, можно выйти на цикл отрицательного веса. (В противном случае Макс может уйти в терминал из любой своей позиции и гарантировать неотрицательность результата). Этот особый случай нужно выделить по графу игры с самого начала и удалить такие позиции (ниже обсудим, как это делается).

В этом разделе далее считаем, что цена любой позиции модифицированной игры неотрицательная. Положительные цены позиций не изменяются (Макс попросту не использует ходы в терминал). Таким образом, для модифицированной игры с добавленной терминальной позицией задача состоит в том, чтобы различать нулевую и положительную цены игры.

Пусть задана стратегия Макса $x\colon V_1 \to V \cup \{\triangle\}$ в модифицированной игре. Предположим, что Мин играет против такой стратегии и стремится минимизировать сумму весов *всех ходов в партии*. Эта сумма может оказаться и $+\infty$, и $-\infty$.

Первый случай возможен, если x гарантирует положительный результат для игры, начинающейся в некоторой вершине v. Как бы ни играл Мин, общая сумма весов будет стремиться к $+\infty$.

Второй случай возможен, если в стратегическом графе G_x есть циклы отрицательного веса, достижимые из начальной позиции. Мин доберётся до такого цикла и будет дальше ходить по этому циклу. Общая сумма весов будет стремиться к $-\infty$.

Однако наличие терминальных ходов Макса может привести к тому, что наилучший результат Мина, начинающего из некоторой вершины v, окажется конечным.

Мы будем называть этот результат *расстоянием от v до $\triangle$* и обозначать его ρ_v. Это неточное название: веса не обязаны удовлетворять неравенству треугольника и могут быть отрицательными. Однако такое название более наглядное и короткое, чем более точное название «длина кратчайшего пути из v в $\triangle$». Заметим, что и это последнее название не слишком точное. Предположим, что Мин вышел на цикл отрицательного веса и ходит по нему. Общая сумма весов стремится к $-\infty$, но терминал остаётся недостижимым на таком пути. Для единообразия будем в таком случае кратчайшим путём из v в $\triangle$ считать любой бесконечный путь, который из верши-

ны v достигает цикла отрицательного веса и далее продолжается бесконечно вдоль этого цикла. Аналогично для $\rho_v = +\infty$: кратчайшим путём будем считать любой бесконечный путь, который выходит на положительный цикл и далее продолжается по этому циклу.

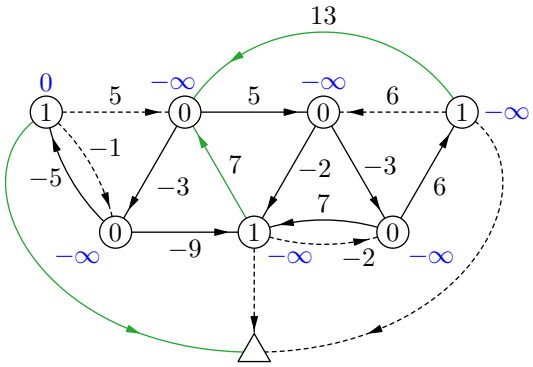


Рис. 10.1: Кратчайшие расстояния до терминала в стратегическом графе

Пример вычисления расстояний до терминала в стратегическом графе показан на рисунке 10.1. Штриховыми линиями обозначены ходы Макса, не входящие в стратегию.

Набор расстояний до терминала и будет нашей мерой успешности стратегии. Формально, это n-мерный вектор $\rho = (\rho_1, \ldots, \rho_n)$, где n — количество позиций в графе игры, а $\rho_i \in \mathbb{R} \cup \{\pm\infty\}$. Сравнивать такие векторы мы будем покоординатно: $\rho' \leqslant \rho''$ означает, что $\rho'_i \leqslant \rho''_i$ для всех i. Чем больше вектор, тем лучше стратегия Макса (в идеале Максу хочется иметь все координаты вектора расстояний $+\infty$; конечно, такой идеальный для него вариант может оказаться нереализуемым).

Стратегию, изображённую на рисунке 10.1 легко локально улучшить. Макс в любой своей позиции v может выбрать ход в терминал, и это гарантирует ему $\rho_v = 0 > -\infty$.

А можно ли локально улучшить эту *осторожную* стратегию, которая состоит только из ходов в терминал? Ответ положительный, как показывает рисунок 10.2.

В двух позициях расстояния до терминала увеличились, а в остальных не уменьшились. Однако и эту стратегию можно улучшить, как показывает рисунок 10.3.

Из четырёх позиций, нарисованных справа, невозможно попасть в терминал. Поэтому расстояния до терминала в этих позициях равны $+\infty$. В остальных позициях расстояния не уменьшаются (в некоторых увеличиваются).

Можно проверить, что полученную стратегию уже невозможно локально улучшить, изменяя ход только в одной позиции. Более того, она оказывается оптимальной в смысле циклической игры.

Перейдём к общему случаю. Для начала докажем, что существует глобальный максимум расстояний до терминала в модифицированной циклической игре.

Тут оказывается полезной каноническая форма циклической игры. Для модифицированной игры будем обозначать канонические веса $w'(i,j) = w(i,j) + \varphi_i - \varphi_j$,

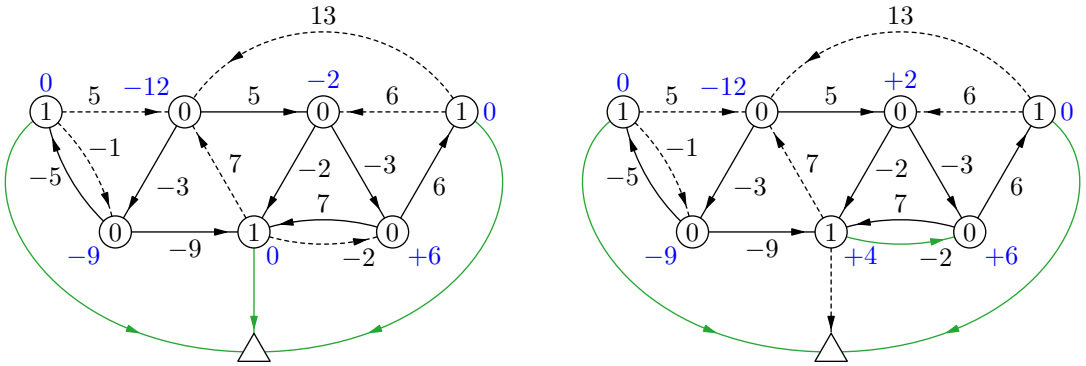


Рис. 10.2: Улучшение осторожной стратегии

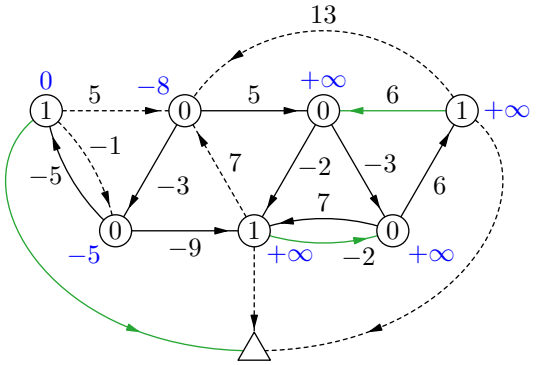


Рис. 10.3: Оптимальная стратегия Макса

здесь φ_i — потенциалы, приводящие игру к канонической.

Веса всех путей от данной вершины v до терминала $\triangle$ изменяются при потенциальной преобразовании на одну и ту же величину $\varphi_\triangle - \varphi_v$. А поскольку для любого вектора $a \in \mathbb{R}^m$ равносильны неравенства $x \leqslant y$ и $x + a \leqslant y + a$, при сравнении стратегий достаточно рассматривать только канонические игры.

Утверждение 10.1. *В канонической модифицированной игре глобальный максимум $\rho^{\max}$ вектора расстояний до терминала задаётся соотношениями $\rho_v^{\max} = 0$, если $p(v) = 0$; $\rho_v^{\max} = +\infty$, если $p(v) > 0$. Здесь $p(v)$ — цена позиции в модифицированной циклической игре.*

Доказательство. В позициях с $p(v) > 0$ у Макса есть ход веса $p(v)$ и он не ведёт в терминал (где цена позиции 0). В соответствующей стратегии терминал недостижим. С другой стороны, в позициях с $p(v) = 0$ у Мина есть ход веса 0, а у Макса ход веса 0 максимально возможный. Поэтому расстояние от такой позиции до терминала неположительно. Оно достигается, если Макс выбирает каноническую стратегию. □

Теперь убедимся, что локальных максимумов, отличных от глобального, для

вектора расстояний до терминала нет.

Будем говорить, что стратегия x' получается *изменением стратегии x в позиции v*, если $x'(u) = x(u)$ при $u \neq v$. Если при этом $\rho'_v > \rho_v$, то будем говорить, что стратегия x' *даёт локальное улучшение.*

Утверждение 10.2. *Пусть x' получается изменением стратегии x в позиции v и даёт локальное улучшение. Тогда для векторов расстояний в стратегических графах выполняется неравенство $\rho(G_{x'}) > \rho(G_x)$.*

Доказательство. Из неравенства локального улучшения следует, что $\rho(x')_v > -\infty$. Поэтому из вершины v недостижимы циклы отрицательного веса в графе $G_{x'}$.

Разобьём все пути по стратегическому графу $G_{x'}$ из какой-то вершины u в терминал на две группы.

Первая состоит из путей, которые вообще не проходят через v. Такие пути присутствуют в обоих стратегических графах G_x, $G_{x'}$ и их длины одинаковые.

Вторая группа путей состоит из путей, проходящих через вершину v. Длина такого пути в $G_{x'}$ равна $W_1 + W_2$, где W_1 — длина какого-то пути τ из u в v (присутствует в обоих стратегических графах в силу сделанного в начале замечания), а W_2 — длина пути из v в терминал по стратегическому графу $G_{x'}$. Она не меньше $\rho(G_{x'})_v > \rho(G_x)_v$ (по предположению о локальном улучшении). Присоединив к τ кратчайший путь из v в $\triangle$, получаем путь по стратегическому графу G_x, длина которого меньше $W_1 + W_2$.

Итак, для каждого пути из произвольной вершины u в терминал $\triangle$ по стратегическому графу $G_{x'}$ мы указали путь из v в $\triangle$ по стратегическому графу G_x, вес которого не больше. Значит, $\rho(G_{x'})_u \geqslant \rho(G_x)_u$. В вершине v неравенство по условию строгое. Поэтому $\rho(G_{x'}) > \rho(G_x)$. □

Теперь нужно понять, на каких стратегиях невозможны локальные улучшения. Оказывается, что локальные улучшения невозможны только при оптимальных стратегиях, если выполняется сделанное выше предположение, что в расширенной циклической игре нет позиций с отрицательной ценой.

Из этого предположения следует простое наблюдение: для неулучшаемой стратегии все расстояния до терминала больше $-\infty$. Действительно, в любой вершине Макса можно переключиться на ход в терминал, после которого расстояние от этой вершины до терминала становится нулевым.

Утверждение 10.3. *Пусть стратегия x не допускает локальных улучшений. Тогда на этой стратегии достигается глобальный максимум вектора расстояний.*

Доказательство. Как и выше, достаточно рассмотреть только каноническую форму модифицированной игры. Расстояние от вершины v до терминала относительно канонических весов в стратегическом графе G_x обозначаем $\rho(x)_v$. Цену игры в позиции v обозначаем $p(v)$.

Если $\rho(x)_v = +\infty$, то из утверждения 10.1 следует, что $\rho(x)_v = \rho_v^{\max}$ (и $p(v) > 0$).

Те позиции v, для которых $-\infty < \rho(x)_v < +\infty$, образуют множество V_f. Позиций с $\rho'(x) = -\infty$ нет в силу сделанного выше предположения.

Обозначим через V_f^+ множество тех позиций $v \in V_f$, для которых $p(v) > 0$. Докажем, что $V_f^+ = \varnothing$, если стратегия x неулучшаемая.

Пусть $V_f^+ \neq \varnothing$. Выберем позицию $v \in V_f^+$ с минимальным значением $\rho(x)_v$. Проследим за кратчайшим путём из v в $\triangle$. На этом пути обязательно встретится позиция Макса с положительной ценой. Действительно, в канонической игре нет ходов из позиций Мина, понижающих цену игры, все такие позиции принадлежат Максу. А на кратчайшем пути из v в $\triangle$ какой-то ход должен понижать цену игры.

Пусть v' — ближайшая к v позиция Макса на кратчайшем пути в $\triangle$ (возможно $v' = v$). Тогда все ходы до v' делает Мин в позициях с положительной ценой. Значит, веса этих ходов также положительные. Поэтому $\rho(x)_v \geqslant \rho(x)_{v'}$ (равенство возможно при $v = v'$).

Обозначим $s\colon V_1 \to V \cup \{\triangle\}$ каноническую стратегию Макса в модифицированной игре. Обозначим $v^* = s(v')$.

Заметим, что если $x(v') = s(v')$, то

$$\rho(x)_v \geqslant \rho(x)_{v'} = p(v') + \rho(x)_{v^*} > \rho(x)_{v^*}.$$

Поэтому по выбору v получаем неравенство $x(v') \neq s(v')$.

Если $\rho(x)_{v'} > \rho(x)_{v^*}$, то

$$\rho(x)_v \geqslant \rho(x)_{v'} > \rho(x)_{v^*}.$$

По выбору v получаем неравенство $\rho(x)_{v'} \leqslant \rho(x)_{v^*}$.

Из двух полученных неравенств следует возможность локального улучшения в позиции v'. Новая стратегия задаётся соотношением $x'(v') = v^*$, в остальных позициях она совпадает с x.

Кратчайший путь из v^* в терминал по графу $G_{x'}$ не короче кратчайшего пути из v^* в терминал по графу G_x (первый путь не проходит через v' и потому лежит также и в графе G_x). Поэтому

$$\rho(x')_{v'} = p(v') + \rho(x')_{v^*} > \rho(x)_{v^*} \geqslant \rho(x)_{v'},$$

что и означает локальное улучшение.

Таким образом, для неулучшаемой стратегии $V_f^+ = \varnothing$, то есть для всех позиций из V_f цена нулевая.

Если в позиции Макса расстояние до терминала отрицательное, то возможно локальное улучшение (в этой позиции выбираем ход в терминал). Из утверждения 10.1 следует, что оно нулевое.

Ходы из позиций Мина в данном случае имеют неотрицательный вес (минимум весов в каждой позиции равен 0). Поэтому и расстояние от позиции Мина до терминала неотрицательное, а значит, нулевое в силу утверждения 10.1. □

10.2. Вероятностный алгоритм поиска оптимальной стратегии

Изложенные выше идеи приводят к алгоритму разбиения множеств позиций графа на позиции с положительной ценой и позиции с неположительной ценой. Этот алгоритм основан на рекурсивной процедуре BV поиска стратегии с максимальной

мерой успешности. У каждого вызова такой процедуры два аргумента: граф игры G и стратегия Макса x. Все графы, на которых работает процедура, являются подграфами графа исходной расширенной циклической игры. Весовая функция для каждого такого подграфа получается ограничением весовой функции исходной игры. Результатом работы процедуры BV является стратегия x^*, которая совпадает с оптимальной стратегией в тех позициях, для которых цена игры положительная.

Алгоритм определения знака цены циклической игры (G, w, v_0):

1. Преобразовать веса по формуле (7.4). Новые веса обозначаем w'.

2. Найти в ограничении графа G на ходы Мина компоненты, которые содержат циклы отрицательного веса. Множество вершин этих компонент обозначаем M. Решить игру достижимости на графе G с целевым множеством Мина M. Если M достижимо из v_0, то ответ «отрицательная цена»; Иначе исключить из графа G те позиции, из которых Мин достигает M. Полученный в результате сокращения граф обозначаем G'.

3. Расширить граф G' добавлением терминальной позиции и ходов из позиций Макса в эту позицию. Получаем расширенный граф $G_\triangle$ с весовой функцией $w_\triangle$.

4. В качестве начальной стратегии Макса взять осторожную стратегию $x_0(v) = \triangle$. Применить процедуру $BV(G_\triangle, x_0)$, используя весовую функцию $w_\triangle$. Результат обозначаем x^*.

5. Вычислить вектор расстояний $\rho(x^*)$. Если $\rho(x^*)_{v_0} = +\infty$, то цена позиции v_0 объявляется положительной. В противном случае, цена неположительная.

Из обсуждения предыдущего раздела ясна корректность этого алгоритма.

Что касается времени работы, то все шаги исполняются за полиномиальное время за исключением шага 4, на котором вызывается процедура BV.

Что касается шага 2, на котором требуется проверять, что в некотором графе есть циклы отрицательного веса, то эта проверка может быть исполнена с помощью алгоритма Беллмана–Форда за полиномиальное время.

На шаге 5 и в процедуре BV требуется также находить кратчайшие пути в графах, в которых есть рёбра отрицательного веса, но все циклы имеют неотрицательный вес. Эта задача также решается за полиномиальное время с помощью алгоритма Беллмана–Форда.

Таким образом оценка времени работы алгоритма сводится к оценке времени работы процедуры BV с точностью до слагаемых полиномиального роста.

Теперь опишем работу процедуры $\mathrm{BV}(G, x)$:

1. Если в каждой позиции Макса возможен только один ход, то $\mathrm{BV}(G, x) \leftarrow x$.

2. Выбрать случайный ход (uv) Макса, который не входит в стратегию x (выбор равновероятен).

3. $x^* \leftarrow \mathrm{BV}(G \setminus \{(uv)\}, x)$. Здесь $G \setminus \{(uv)\}$ — граф, который получается из G удалением хода (uv). Построить стратегию $\tilde{x}$ из x^* изменением стратегии в вершине u: $\tilde{x}(u) = v$.

4. Если $\rho(\tilde{x})_u > \rho(x^*)_u$, то

   (a) $\mathrm{BV}(G, x) \leftarrow \mathrm{BV}(G_{(uv)}, \tilde{x})$, здесь $G_{(uv)}$ — граф, который получается из G удалением всех ходов из позиции u за исключением хода (uv);

   (b) иначе $\mathrm{BV}(G, x) \leftarrow x^*$.

Корректность алгоритма основана на утверждениях 10.2 и 10.3. Если работа процедуры заканчивается на шаге 1, корректность очевидна.

Если исполняется шаг 4b, то стратегия x^* неулучшаемая. Действительно, изменения стратегии, которые согласованы с графом $G \setminus \{(uv)\}$, не дают улучшения, так как x^* предоставляет максимум меры успешности на этом подграфе. В графе G есть ещё одна возможность локального улучшения стратегии x^*: стратегия $\tilde{x}$. Но эта возможность исключается проверкой на шаге 4. Из утверждения 10.3 следует, что на стратегии x^* достигается максимум меры успешности.

Если исполняется шаг 4a, то максимальная стратегия в графе G обязана использовать ход (uv): все стратегии, не использующие этот ход, хуже $\tilde{x}$, как следует из утверждения 10.2. Поэтому процедура BV возвращает в этом случае правильный результат.

Оценка времени работы процедуры оказывается непростым делом. Она сводится к оценке количества рекурсивных вызовов при исполнении процедуры, так как остальные действия выполняются за полиномиальное время.

Процедура BV является частным случаем более общей процедуры «абстрактной комбинаторной оптимизации». Оценивать количество вызовов удобно в этой более общей постановке.

10.3. Оптимизация функций на многомерных боксах

В этом разделе мы рассмотрим общую задачу оптимизации функций на конечных множествах, которые логично было бы называть «комбинаторными параллелепипедами». Но такое название длинное, поэтому будем использовать более краткое — *бокс*. По определению, бокс — это декартово произведение нескольких конечных множеств,

$$\mathcal{P} = \mathcal{P}_1 \times \cdots \times \mathcal{P}_n, \quad |\mathcal{P}_i| < \infty.$$

Другими словами, бокс состоит из упорядоченных наборов вида $(p_1, p_2 \ldots, p_n)$, в которых первая координата принимает значения в множестве $\mathcal{P}_1$, вторая — в $\mathcal{P}_2$, …, последняя — в $\mathcal{P}_n$.

Примером бокса является булев куб $\mathcal{B}_n$ размерности n, то есть множество $\{0,1\}^n$ двоичных слов длины n. В этом случае $\mathcal{P}_i = \{0,1\}$.

Другим примером бокса является множеством стратегий Макса в циклической игре. В этом случае $\mathcal{P}_v$ — множество ходов из позиции v, а декартово произведение берётся по всем вершинам Макса.

Количество d тех координат i, для которых $|\mathcal{P}_i| > 1$ называется *размерностью* бокса и обозначается $\dim \mathcal{P}$. Остальные координаты одинаковы для всех точек бокса и потому они несущественны. 0-мерный бокс состоит из одной точки.

Подбокс $\mathcal{P}'$ бокса $\mathcal{P}$ получается, если в каждом множестве $\mathcal{P}_i$ выбрать подмножество $\mathcal{P}'_i$. Тогда

$$\mathcal{P}' = \mathcal{P}'_1 \times \cdots \times \mathcal{P}'_n.$$

В описании процедуры BV используются два подбокса бокса стратегий Макса. Первый отвечает исключению из $\mathcal{P}_u$ одного хода. Второй — исключению всех ходов кроме одного. Этот последний случай в общей ситуации называется *фасетой*. Любой подбокс бокса стратегий является боксом стратегий для графа игры, получающегося удалением части рёбер.

Дадим общие определения. Если зафиксировать значения части координат d-мерного бокса, получим *грань* бокса $\mathcal{F}$. Грань сама по себе является боксом размерности $d - k$, где k — количество координат, значения которых зафиксированы.

Пример 10. В кубе $\mathcal{B}_5$ зафиксируем значения координат следующим образом: $x_1 = x_4 = 0$, $x_5 = 1$. Получаем грань, состоящую из четырёх вершин, эта грань является двумерным кубом:

$$0\mathbf{01}01 \qquad 0\mathbf{11}01$$

$$0\mathbf{00}01 \qquad 0\mathbf{10}01$$

Фасетой будем называть грань d-мерного бокса размерности $d-1$ (коразмерности 1). Дополнение к фасете не всегда является фасетой, но всегда является подбоксом исходного бокса (из множества $\mathcal{P}_i$ вычеркнут один из элементов).

Легко видеть, что в n-мерном кубе есть $2n$ фасет, поскольку значение каждой координаты можно зафиксировать ровно двумя способами.

В общем случае количество фасет равно

$$m = \sum_{i:|\mathcal{P}_i|>1} |\mathcal{P}_i|.$$

Действительно, значение i-й координаты можно зафиксировать $|\mathcal{P}_i|$ способами. Однако при $|\mathcal{P}_i| = 1$ ничего не меняется: получается не фасета, а исходный бокс.

Напомним что *соседями* в булевом кубе считаются слова, которые различаются ровно в одной координате. Другими словами это означает, что соседние слова образуют 1-мерную грань. Легко понять, что у каждой точки булева куба есть ровно n соседей. Отношение соседства задаёт *граф булева куба*.

В общем случае отношение соседства точек в боксе определяется аналогично: две точки соседние, если они различаются ровно в одной координате. Теперь уже

неверно, что пара соседних точек образует 1-мерную грань. Количество соседей в произвольном боксе нетрудно сосчитать.

Из общего количества m фасет, ровно d содержат данную точку $x = (x_1, \ldots, x_d)$. Проверим, что в каждой из остальных $m - d$ фасет есть ровно один сосед точки x. Пусть фасета задаётся уравнением $y_i = a$, то есть это множество $\mathcal{F} = \{y \in \mathcal{P} : y_i = a\}$. Если $x \notin \mathcal{F}$, то $x_i \neq a$ и точка $\Pi_{\mathcal{F}} x = (x_1, \ldots, x_{i-1}, a, x_{i+1}, \ldots, x_d)$ является соседом x. Других соседей у точки x в фасете $\mathcal{F}$ нет: любая другая точка фасеты различается с x по крайней мере в двух координатах.

Итак, у каждой точки в боксе есть ровно $m - d$ соседей.

В боксе стратегий соседними оказываются стратегии, которые получаются изменением хода в одной позиции.

Мы будем рассматривать функции из бокса со значениями в некотором частично упорядоченном множестве $\mathcal{D}$. Частичная упорядоченность означает, что для двух разных элементов $d' \neq d''$ этого множества возможны три результата сравнения: $d' < d''$; $d' > d''$; элементы d' и d'' несравнимы.

Примером частично упорядоченного множества является множество n-мерных векторов с отношением покоординатного порядка. Процедура BV из предыдущего раздела оптимизирует функцию $\rho\colon \mathcal{P} \to \mathbb{R}^n$, где n — количество вершин в графе игры.

Локальным максимумом функции $f\colon \mathcal{P} \to \mathcal{D}$ будем называть такую точку x бокса, что для любого соседа x' точки x выполняется неравенство $f(x) \geqslant f(x')$ (в частности, значения функции во всех соседях точки x сравнимы со значением $f(x)$).

Утверждение 10.2 говорит, что если достигнуто улучшение в одной позиции, то стратегия не является локальным максимумом.

Глобальным максимумом функции $f\colon \mathcal{P} \to \mathcal{D}$ будем называть такую точку x булева куба, что для любой другой точки x' выполняется неравенство $f(x) \geqslant f(x')$.

Утверждение 10.3 говорит, что неулучшаемая стратегия является глобальным оптимумом для любого графа циклической игры.

На 1-мерном боксе понятия глобального и локального максимумов совпадают (все пары точек соседние). Уже для 2-мерного булева куба и функций в линейно упорядоченное множество (действительные числа) это не так, см. рисунок 10.4.

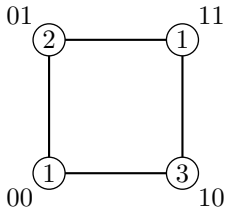


Рис. 10.4: 01 — локальный максимум, но не глобальный

Если у функции f из бокса в линейно упорядоченное множество всякий локальный максимум является глобальным, найти глобальный максимум можно *локаль-*

ным поиском. Процедура локального поиска начинается из произвольной вершины. На каждой итерации она ищет *улучшающего соседа*: такого соседа x' текущей точки x_t, что $f(x') > f(x_t)$. Если улучшающий сосед найден, то процедура переходит в него: $x_{t+1} = x_t$. Если не найден, процедура локального поиска останавливается.

Поскольку бокс конечен, а значения функции на каждой итерации увеличиваются, процедура обязательно останавливается. Конечная точка является локальным максимумом.

Одна из проблем с локальным поиском в том, что он может работать очень долго. Мы не указали точное правило, по которому выбирается улучшающий сосед, если их много. Если это правило выбрано неудачно, то пример долгой работы указать легко.

Кодом Грея называется цикл длины 2^n в графе n-мерного булева куба. Другими словами, код Грея — это такая последовательность $X_0, X_1, \ldots, X_{N-1}$ двоичных слов длины n, что для любого i слова X_i и X_{i+1} различаются ровно в одной позиции (сложение в индексе происходит по модулю $N = 2^n$).

Кодов Грея много, приведём один пример. Он строится индуктивно. Для 1-мерного куба это последовательность $X^{(1)} = \big(0, 1\big)$. Далее последовательности определяются так:

$$X^{(n)} = \big(X_0^{(n-1)}0, X_1^{(n-1)}0, \ldots X_{2^{n-1}-1}^{(n-1)}0, X_{2^{n-1}-1}^{(n-1)}1, X_{2^{n-1}-2}^{(n-1)}1, \ldots X_1^{(n-1)}1, X_0^{(n-1)}1\big).$$

На рисунке 10.5 изображён такой код Грея для $n = 3$.

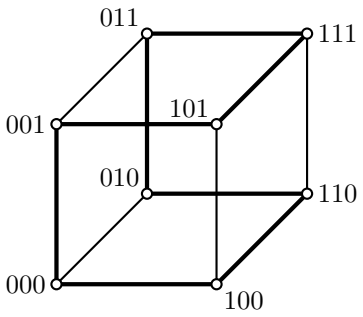


Рис. 10.5: 3-мерный код Грея

Используя код Грея, легко придумать функцию $f\colon \mathcal{B}_n \to \mathbb{R}$, у которой один локальный максимум (он же глобальный) и при этом есть путь длины $2^n - 1$ по рёбрам графа булева куба, на котором значения функции возрастают.

Пример 11. Полагаем $G(x)$ равным номеру x в порядке построенного выше кода Грея. Тогда на всех рёбрах, которые соединяют соседние слова в коде Грея, функция G возрастает.

При неудачном выборе начальной точки и неудачном правиле выбора улучшающего соседа, поиск глобального максимума функции из примера 11 занимает много времени: нужно узнать значения функции во всех точках.

Однако заметим, что в построенном примере из точки 0^n можно попасть в точку глобального максимума $0^{n-1}1$ за один шаг. Если выбирать самого лучшего улучшающего соседа, то глобальный максимум отыщется за $n+2$ вычисления значения функции.

Пример 11 можно усилить и добиться того, что никакое правило выбора улучшающего соседа не позволит избежать длинного пути улучшения.

Пример 12 (Двойной код Грея)**.** Рассмотрим последовательность точек $2n$-мерного куба

$$(X_0, X_0), (X_1, X_0), (X_1, X_1), \ldots, (X_{N-1}, X_{N-1}).$$

В этой последовательности через одну стоят точки, которые являются удвоениями соответствующих точек n-мерного кода Грея. Они связаны дополнительными точками: между (X_k, X_k) и (X_{k+1}, X_{k+1}) вставляем точку (X_{k+1}, X_k), чтобы соседние в последовательности точки были соседями в булевом кубе.

Проверим, что несоседние точки в этой последовательности различаются по крайней мере в двух позициях. Это очевидно для дублей (X_k, X_k) и (X_ℓ, X_ℓ): при $k \neq \ell$ эти точки различаются по крайней мере в двух позициях, так как $X_k \neq X_\ell$.

Аналогично для пар промежуточных точек: при $k \neq \ell$ не только $X_k \neq X_\ell$, но и $X_{k+1} \neq X_{\ell+1}$, поэтому (X_{k+1}, X_k) и $(X_{\ell+1}, X_\ell)$ различаются по крайней мере в двух позициях.

Осталось рассмотреть смешанный случай (X_k, X_k) и $(X_{\ell+1}, X_\ell)$, $X_k \neq X_\ell$, $X_k \neq X_{\ell+1}$ (последнее потому, что иначе эти пары точек соседние в последовательности). Но это значит, что точки (X_k, X_k) и $(X_{\ell+1}, X_\ell)$ различаются по крайней мере в двух позициях: одной в первой половине и одной во второй половине.

Теперь определим функцию $F\colon \mathcal{B}_{2n} \to \mathbb{R}$ следующим образом. В точках построенной последовательности значение функции равно номеру точки в этой последовательности. В остальных точках значение отрицательное, зададим его так:

$$F(x_1, \ldots, x_n, y_1, \ldots, y_n) = -\sum_{i=1}^{n}(x_i + y_i).$$

Такой выбор обеспечивает единственность глобального максимума. В любой точке вне последовательности есть хотя бы одна координата, равная 1. Заменяя её на 0, получаем в любом случае улучшающего соседа (либо модуль значения функции уменьшился, либо значение функции стало неотрицательным).

Если начинать локальный поиск из точки $(X_0, X_0) = 0^{2n}$, то на каждом шаге будет лишь один улучшающий сосед: следующая точка последовательности. Поэтому длина улучшающего пути окажется $2^n - 1$.

Мы справились с произволом в выборе правила улучшающего соседа. Однако остаётся иное возражение: мы нашли длинную последовательность улучшений только при выборе в качестве начальной точки (X_0, X_0).

Естественный способ избежать «плохих» начальных условий — выбирать их случайно. Для построенной в примере 12 функции это помогает «в оптимистическом

смысле». А именно, с вероятностью, близкой к 1/4 будет выбрана точка вне последовательности со значениями $x_n = 1$, $y_n = 1$. Из этой точки, как нетрудно видеть, существует короткий улучшающий путь в глобальный оптимум: на каждом шаге нужно обращать одну из остальных координат в 0. Если поверить, что у нас есть правильный способ выбора улучшающего соседа, после не очень большого количества попыток выбора начальной точки глобальный максимум будет найден.

Пример 13. Построим из той же последовательности, что и в примере 12, другую функцию $B(x, y)$. Она отличается от функции $F(x, y)$ из примера 12 в точках вида $(u0^{n/2-1}1, v0^{n/2-1}1)$, не принадлежащих последовательности (для простоты считаем n чётным). В этих точках $B(x, y)$ принимает очень большие отрицательные значения (по модулю больше максимума модуля $F(x, y)$). Для определённости считаем, что в этих точках

$$B(x, y) = F(x, y) \cdot \max_{(u,v)} |F(u, v)|.$$

Локальных максимумов от такого переопределения не добавится.

У точек, в которых $x_n + y_n \geqslant 1$ есть улучшающий сосед, в котором одна из координат x_n, y_n заменяется на 0.

Пусть $x_n = y_n = 0$ для точки (x, y). Тогда замена любой другой единицы в (x, y) на нуль даёт точку, в которой значение функции B больше (так как такой сосед не лежит в множестве, на котором функции B и F различаются).

А нулевая точка, как уже сказано, лежит на последовательности, так что у неё есть улучшающий сосед: следующий член последовательности.

Последние $2^{n/2}$ членов последовательности имеют вид $(u0^{n/2-1}1, v0^{n/2-1}1)$, так как первые $2^{n/2}$ членов соответствующего кода Грея имеют вид $u0^{n/2}$. Поэтому с вероятностью, очень близкой к 1, длина улучшающего пути из случайной точки будет не меньше $2^{n/2}$: все соседи точек, которые ближе на последовательности к глобальному максимуму, имеют очень маленькие значения и никакой путь улучшения через них не проходит.

Вторая проблема с локальным поиском возникает для функций со значениями в частично упорядоченном множестве. Вполне может оказаться, что глобальный максимум есть, но улучшающих соседей нет ни одной точки, кроме глобального максимума (все значения в этих точках между собой несравнимы).

Однако для некоторых классов функций локальный поиск работает намного лучше. Определим нужный нам класс (в [6] примерно этот класс функций назывался RLG-функции).

Определение 10.1. Функцию $f\colon \mathcal{P} \to \mathcal{D}$, где $\mathcal{P}$ — бокс, а $\mathcal{D}$ — частично упорядоченное множество, будем называть *BV-функцией*, если для любого подбокса $\mathcal{P}' \subseteq \mathcal{P}$ выполняется свойство: если у точки $x \in \mathcal{P}'$ нет *улучшающего* соседа в подбоксе $\mathcal{P}'$, то точка x является глобальным максимумом ограничения функции f на подбокс $\mathcal{P}'$.

Определение выглядит громоздко. Оно по сути означает, что у всех ограничений функции на подбоксы есть только один локальный максимум. Более сложная

формулировка удобнее для функций со значениями в частично упорядоченном множестве.

Заметим, что утверждение 10.3 можно переформулировать так: функция расстояний до терминала является BV-функцией.

Чтобы освоиться с определением, применим его к построенным в примерах функциям.

Утверждение 10.4. *Функция $G(x)$ из примера 11 является BV-функцией.*

Доказательство. Удобно доказывать индукцией по n более сильное свойство: у ограничения G на любую грань булева куба есть ровно один локальный максимум и ровно один локальный минимум (они же глобальные).

База $n = 1$ очевидна.

Индуктивный переход. Предполагаем, что для размерностей $k < n$ утверждение доказано. Рассмотрим в n-мерном кубе некоторую грань $\mathcal{F}$. Если в этой грани значение последней координаты фиксировано, то непосредственно применяем предположение индукции (заметьте, что при $x_n = 1$ локальные максимумы и минимумы меняются местами по сравнению с гранью, лежащей в $x_n = 0$, так как порядок обхода этих фасет противоположный).

Если же в грани $\mathcal{F}$ значение x_n не фиксировано, то у любой точки в $F_0 = \mathcal{F} \cap \{x_n = 0\}$ есть сосед в $F_1 = \mathcal{F} \cap \{x_n = 1\}$, на котором значение функции G больше, и наоборот. Поэтому единственный локальный максимум G на грани F совпадает с единственным локальным максимумом на грани F_1, а единственный локальный минимум G на грани F совпадает с единственным локальным минимумом на грани F_0 (предположение индукции). □

Утверждение 10.5. *Функция $B(x, y)$ из примера 13 не является BV-функцией при $n \geqslant 4$.*

Доказательство. Рассмотрим подкуб $\mathcal{F}$, состоящий из точек вида $1u01^{n-1}0$, длина слова u равна $n - 2$.

В этой грани есть по крайней мере два локальных максимума: глобальный максимум, который лежит на последовательности (так как $1^{n-1}01^{n-1}0 \in F$), и точка $a = 10^{n-1}1^{n-1}0$. Она и её соседи в грани $\mathcal{F}$ не лежат на последовательности и потому значение функции F в любом соседе точки a меньше, чем значение в точке a. □

10.4. Общий алгоритм оптимизации

Для BV-функций известны более быстрые алгоритмы поиска глобального максимума, чем простой локальный поиск, который мы обсуждали раньше. Эти алгоритмы работают за субэкспоненциальное по размерности бокса количество вычислений функции.

Мы рассмотрим один из таких алгоритмов, подробно описанный в [5] и восходящий к [13].

Алгоритм MSW рекурсивный. Он получает на вход бокс $\mathcal{P}$ и начальную точку x_0. Результатом $\mathrm{MSW}(\mathcal{P}, v_0)$ работы алгоритма является точка глобального максимума функции $f\colon \mathcal{P} \to \mathcal{D}$. Алгоритм вызывает себя на гранях исходного бокса, функция всё время одна и та же.

Работа MSW на входе $(\mathcal{P}, v_0)$:

1. Если $\dim \mathcal{P} = 0$, то $\mathrm{MSW}(\mathcal{P}, v_0) \leftarrow v_0$.
2. Выбрать случайную фасету $\mathcal{F}$ бокса $\mathcal{P}$, не содержащую точку v_0 (выбор равновероятен).
3. $v^* \leftarrow \mathrm{MSW}(\mathcal{P} \setminus \mathcal{F}, v_0)$, u — сосед v^* в фасете $\mathcal{F}$.
4. Если $f(u) > f(v^*)$, то
   (a) $\mathrm{MSW}(\mathcal{P}, v_0) \leftarrow \mathrm{MSW}(\mathcal{F}, u)$;
   (b) иначе $\mathrm{MSW}(\mathcal{P}, v_0) \leftarrow v^*$.

Сравнивая это описание с описанием процедуры BV, видим, что процедура BV — это алгоритм MSW, применённый к боксу стратегий (с осторожной стратегией в качестве начальной точки).

Как видно из описания алгоритма MSW, в нём возможны два рекурсивных вызова. Второй происходит на шаге 4a и исполняется не всегда. Каждый из этих вызовов обращается к боксу с меньшим количеством фасет, что гарантирует сходимость (нулевое количество фасет приводит к завершению алгоритма на шаге 1).

Утверждение 10.6 (корректность MSW алгоритма)**.** *Если $f\colon \mathcal{P} \to \mathcal{D}$ — BV-функция, то* $\mathrm{MSW}(\mathcal{P}, v_0)$ *является глобальным максимумом функции f для любой начальной точки.*

Доказательство. Индукция по количеству фасет в боксе.

База: если нет ни одной фасеты, то $\mathcal{P}$ состоит из одной точки v_0, утверждение тривиально выполняется.

Индуктивный переход.

Пусть значение $\mathrm{MSW}(\mathcal{P}, v_0)$ определяется на шаге 4a. По индуктивному предположению значение $f(v^*)$ глобального максимума на $\mathcal{P} \setminus \mathcal{F}$ меньше, чем значение $f(u^*)$ глобального максимума на фасете $\mathcal{F}$. Значит, значение в любой точке бокса $\mathcal{P}$ меньше $f(u^*)$, где $u^* = \mathrm{MSW}(\mathcal{P}, v_0)$ — результат работы алгоритма.

Пусть значение $\mathrm{MSW}(\mathcal{P}, v_0)$ определяется на шаге 4b. По индуктивному предположению v^* — глобальный максимум на $\mathcal{P} \setminus \mathcal{F}$. И у v^* нет улучшающего соседа (u — единственный сосед вне $\mathcal{P} \setminus \mathcal{F}$ — не улучшающий, поскольку исполняется шаг 4b). Так как f — BV-функция, то v^* — глобальный максимум на $\mathcal{P}$. □

Оценить скорость работы алгоритма MSW непросто. Мы будем оценивать количество листьев L в дереве рекурсивных вызовов, см. рисунок 10.6.

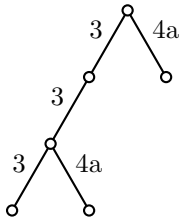


Рис. 10.6: Дерево рекурсии

Как нетрудно видеть, это количество на единицу больше количества исполняемых алгоритмом шагов 4a. С другой стороны, глубина дерева рекурсии не превосходит общего количества фасет m в боксе $\mathcal{P}$. Поэтому количество вершин в дереве рекурсии не больше mL. При каждом вызове (за исключением вызова на 0-мерном боксе) алгоритма MSW выполняется вычисление двух значений функции f. Таким образом, время работы такого алгоритма пропорционально mL, где коэффициент пропорциональности зависит от времени вычисления функции f и времени на выполнение вспомогательных действий (выбор случайной фасеты и т.п.).

При оценке количества листьев (точнее, математического ожидания количества листьев — ведь алгоритм вероятностный) мы будем использовать понятие *скрытой размерности* пары $(\mathcal{P}, v)$.

Всюду далее мы предполагаем, что решается задача оптимизации BV-функции $f\colon \mathcal{P} \to \mathcal{D}$.

Определение 10.2. *Экстремальная* фасета (относительно функции $f\colon \mathcal{P} \to \mathcal{D}$) — это такая фасета, которая содержит все локальные максимумы функции f в боксе $\mathcal{P}$.

Поскольку экстремальные фасеты должны иметь хотя бы одну общую точку, их количество не превосходит размерности $\dim \mathcal{P}$.

Определение 10.3. *Скрытой размерностью* $\operatorname{hdim}(\mathcal{P}, v)$ пары из бокса и точки этого бокса называется разность $\dim \mathcal{P}$ и количества тех фасет, которые содержат v и все те u, для которых $f(u) > f(v)$.

(Скрытая размерность также определяется относительно функции $f\colon \mathcal{P} \to \mathcal{D}$.)

Из определения ясно, что скрытая размерность не превосходит размерности бокса. Столь же ясно, что в точке глобального максимума скрытая размерность равна 0. Неформально эта величина указывает, насколько «близко» расположена текущая точка от глобального максимума.

Если скрытая размерность пары $(\mathcal{P}, v)$ равна k, то точка v и все точки, в которых значение функции строго больше, чем в v, лежат в k-мерной грани $\mathcal{P}'$ бокса $\mathcal{P}$. Алгоритм MSW, который начинает работу на паре $(\mathcal{P}, v)$, не выходит из этой грани.

Обозначим через $L(k, m, d)$ максимальное матожидание количества листьев в дереве рекурсии алгоритма MSW на паре $(\mathcal{P}, v)$, для которой $\dim \mathcal{P} \leqslant d$, $\operatorname{hdim}(\mathcal{P}, v) \leqslant k$, количество фасет в $\mathcal{P}$ не превосходит m.

Зададим функцию $g(k,n)$ соотношениями

$$\begin{aligned} &g(k,0) = 1,\\ &g(0,n) = 1,\\ &g(k,n) = g(k,n-1) + \frac{1}{n}\sum_{j=1}^{\min(k,n)} g(k-j,n). \end{aligned} \tag{10.1}$$

Лемма 10.7. *$L(k,m,d) \leqslant g(k,m-d)$.*

Доказательство. Обозначим $d = \dim \mathcal{P}$, $k = \operatorname{hdim}(\mathcal{P}, v)$, m — количество фасет $\mathcal{P}$. Будем доказывать индукцией по k, $m-d$, что математическое ожидание количества листьев в дереве рекурсии алгоритма MSW, работающего на входе $(\mathcal{P}, v)$, не превосходит $g(k,m-d)$.

База индукции. Пусть $m-d=0$. Так как $m \geqslant 2d$ (напомним, что в определении размерности учитываются только те координаты, которые могут принимать хотя бы два разных значения), отсюда следует, что $m=d=0$. То есть алгоритм исполняет лишь шаг 1. Количество листьев в дереве рекурсии $1 \leqslant 1 = g(k,0)$, неравенство леммы выполняется.

Пусть $k=0$, то есть точка v — глобальный максимум функции f. Тогда при исполнении шага 3 получим $v^* = v$. То же самое справедливо на всех рекурсивных вызовах на подбоксах. Дерево рекурсии — путь, то есть у него один лист. Неравенство леммы выполняется: $1 \leqslant 1 = g(0,m-d)$.

Индуктивный переход. Предположим, что неравенство леммы выполняется при всех значениях параметров, в которых хотя бы один меньше k, m, d.

При исполнении шага 3 количество фасет в $\mathcal{P} \setminus \mathcal{F}_j$ меньше количества фасет $\mathcal{P}$. Размерность очевидно не увеличивается (может уменьшится, если выбрана фасета, фиксирующая значение координаты, которая может принимать всего два различных значения). Скрытая размерность также не увеличивается: все фасеты, которые содержали v и все точки u, для которых $f(u) > f(v)$, в пересечении с $\mathcal{P} \setminus \mathcal{F}$ будут содержать те же точки в $\mathcal{P} \setminus \mathcal{F}$ (при ограничении могут разве что появиться дополнительные фасеты с таким свойством). По индуктивному предположению матожидание количества листьев при вызове MSW на шаге 3 не превосходит $g(k,m-1-d)$.

Предположим, что исполняется шаг 4а. Это возможно лишь если на шаге 2 выбрана экстремальная фасета (в противном случае на шаге 3 найден глобальный максимум и шаг 4а не исполняется).

Из определения скрытой размерности заключаем, что $d-k$ фасет содержат v и все точки, в которых значение функции больше, чем в v. Все эти фасеты экстремальные, так как должны содержать все глобальные максимумы f.

Заметим, что ни одна из этих $d-k$ экстремальных фасет не может быть выбрана на шаге 2, так как все они содержат точку v.

Помимо этих фасет, есть ещё $k' \leqslant k$ экстремальных фасет. Занумеруем их в порядке возрастания максимальных значений f на дополнениях к этим фасетам. Поскольку мы рассматриваем, вообще говоря, частичный порядок $\mathcal{D}$, нужно сформулировать это условие следующим образом: упорядочим дополнительные экстре-

мальные фасеты $\mathcal{F}_1, \ldots, \mathcal{F}_{k'}$ так, чтобы для максимальных значений функции на дополнениях к этим фасетам

$$M_i = \max_{x \in \mathcal{P} \setminus \mathcal{F}_i} f(x)$$

выполнялось свойство линейного продолжения: если $M_i \leqslant M_j$ в порядке $\mathcal{D}$, то $i \leqslant j$. (То, что такое упорядочение существует, мы уже не раз использовали.)

Для линейного порядка это означает, что последовательность M_i (нестрого) монотонно возрастающая.

Пусть на шаге 2 выбрана экстремальная фасета $\mathcal{F}_j$. Тогда, в обозначениях алгоритма, $f(v^*) = M_j < f(u)$. Напомним, что u — это сосед v^* в фасете $\mathcal{F}_j$.

Докажем, что для любого $i \leqslant j$ выполняется включение $u \in \mathcal{F}_i$. Действительно, если $u \notin F_i$, то $u \in \mathcal{P} \setminus \mathcal{F}_i$ и потому $M_j = f(v^*) < f(u) \leqslant M_i$, противоречие со свойством линейного продолжения.

Это означает, что скрытая размерность пары $(\mathcal{F}_j, u)$ по крайней мере на j меньше, чем скрытая размерность пары $(\mathcal{P}, v)$.

Количество k' экстремальных фасет, которые можно выбрать на шаге 2, не превосходит k, как было сказано выше, и, разумеется, не превосходит $m - d$. При выборе фасеты $\mathcal{F}_j$ в выбранном выше порядке размерность уменьшается на 1, скрытая размерность — хотя бы на j, а количество фасет — хотя бы на 1. По индуктивному предположению, матожидание количества листьев при вызове алгоритма на шаге 4а не превосходит $g(k - j, m - d)$.

Вероятность выбрать какую-то определённую фасету равна $1/(m - d)$ (общее количество фасет, которые можно выбрать). Из линейности матожидания получаем, что матожидание общего количества листьев не превосходит

$$g(k, m - 1 - d) + \frac{1}{m - d} \sum_{j=1}^{\min(k, m-d)} g(k - j, m - d) = g(k, m - d),$$

что и требовалось доказать. □

Функция $g(k, n)$ растёт по k субэкспоненциально.

Теорема 10.8. $g(k, n) = \exp\left(\sqrt{k} \ln n + O(\sqrt{k} + \ln n)\right)$ *при* $k, n > 0$.

Доказательство теоремы 10.8 не очень простое, мы приводим его ниже в разделе 10.5. Пока получим из неё интересующую нас оценку работы алгоритма решения циклических игр.

Следствие 10.9. *Математическое ожидание времени работы алгоритма* BV *решения циклических игр не превосходит* $\exp\left(O(\sqrt{n} \ln n)\right)$, *где* n — *количество вершин в графе игры.*

Доказательство. Как легко видеть, алгоритм BV является частным случаем алгоритма MSW. В этом частном случае количество фасет m не превосходит количества рёбер графа игры, то есть $m \leqslant n^2$. Размерность не превосходит количества вершин. Скрытая размерность не превосходит обычной размерности. Вычисление значения функции в точке выполняется за полиномиальное время, как проверено выше.

Подставляя эти оценки в лемму 10.7 и применяя оценку из теоремы 10.8, получаем оценку матожидания времени T работы алгоритма BV:

$$\begin{aligned}\mathbf{E}[T] &\leqslant \operatorname{poly}(n)\cdot m\cdot g(n,n^2)\leqslant \exp(O(\ln n)\cdot\exp(2\sqrt{n}\ln n+O(\sqrt{n}+\ln n))=\\ &=\exp\bigl(O(\sqrt{n}\ln n)\bigr).\end{aligned}$$

□

Нетрудно преобразовать алгоритм BV так, чтобы преобразованный алгоритм BV′ за время $\varepsilon^{-1}\exp\bigl(O(\sqrt{n}\ln n)\bigr)$ определял знак цены циклической игры с вероятностью ошибки меньше ε.

Для этого стандартным образом используется неравенство Маркова

$$\mathbf{Pr}[T>a]\leqslant\frac{\mathbf{E}[T]}{a}\,.$$

Нужно ограничить количество рекурсивных вызовов в алгоритме BV величиной $\varepsilon^{-1}\exp\bigl(O(\sqrt{n}\ln n)\bigr)$ и, если результат за это время не получен, выдать произвольный результат.

10.5. Решение рекурренты

В этом разделе мы доказываем теорему 10.8, следуя работе [13]. В этой работе даны более точные оценки $g(k,n)$, как верхние, так и нижние. Мы ограничиваемся более простой оценкой, приведённой в теореме 10.8.

Рассмотрим производящую функцию

$$G_n(z)=\sum_{k=0}^{\infty}g(k,n)z^k.$$

Из граничного условия получаем

$$G_0(z)=\sum_{k=0}^{\infty}z^k=\frac{1}{1-z}\,.$$

Рекуррентное соотношение в (10.1) даёт рекуррентное соотношение между $G_{n-1}(z)$ и $G_n(z)$:

$$G_n(z)=G_{n-1}(z)+\frac{z+z^2+\cdots+z^n}{n}\cdot G_n(z),$$

то есть

$$G_n(z)=G_{n-1}(z)\cdot\frac{1}{1-\frac{z+z^2+\cdots+z^n}{n}}\,.$$

Отсюда получаем выражение для $G_n(z)$ в «замкнутой форме»:

$$G_n(z)=\frac{1}{1-z}\cdot\prod_{j=1}^{n}\frac{1}{1-\frac{z+z^2+\cdots+z^j}{j}}\,. \tag{10.2}$$

Нас интересует оценка коэффициентов в разложении функции $G_n(z)$ в ряд. Посмотрим на $G_n(z)$ как на функцию комплексного переменного. Поскольку $g(0,n)=$ $=1$, эта функция равна 1 в точке 0. Из формулы Коши получаем выражение для

коэффициента $[z^k]G_n(z) = g(k,n)$:

$$g(k,n) = \frac{1}{2\pi i}\int_\gamma \frac{G_n(z)}{z^{k+1}}\,dz.$$

Здесь контур интегрирования должен один раз обходить 0 и не содержать других точек, в которых знаменатели в (10.2) обращаются в 0.

Мы выберем таким контуром окружность $|z| = t < 1$. При $|z| \leqslant t$ выполняется неравенство

$$\Big|\frac{z+z^2+\cdots+z^j}{j}\Big| \leqslant \frac{t+t^2+\cdots+t^j}{j} \leqslant t,$$

так что каждый знаменатель в (10.2) по модулю не меньше $1-t$.

Из того же неравенства следует, что на окружности $|z| = t$ выполняется оценка

$$|G_n(z)| \leqslant \frac{1}{1-t}\prod_{j=1}^{n}\frac{1}{1-\frac{t+t^2+\cdots+t^j}{j}}\ ,$$

из которой получаем оценку интеграла и $g(k,n)$:

$$g(k,n) \leqslant \frac{1}{t^{k+1}}\cdot\frac{1}{1-t}\prod_{j=1}^{n}\frac{1}{1-\frac{t+t^2+\cdots+t^j}{j}}\ . \tag{10.3}$$

Будем оценивать правую часть этого неравенства при $t = 1-1/q$, где $q \geqslant 2$ целое. Оценим первый множитель, используя неравенства

$$e^{-1} \leqslant \left(1+\frac{1}{n}\right)^{-n} \leqslant \left(1-\frac{1}{n}\right)^{n-1}.$$

Получаем

$$\frac{1}{t^{k+1}} = \left(1-\frac{1}{q}\right)^{-k-1} \leqslant e^{(k+1)/(q-1)}.$$

Второй множитель равен q. Произведение в (10.3) разобьём на две части и оценим их по отдельности. При $j \geqslant q$ получаем

$$1-\frac{t+t^2+\cdots+t^j}{j} = 1-\frac{t}{j}\cdot\frac{1-t^j}{1-t} = 1-\frac{1-1/q}{j/q}\cdot\left(1-\left(1-\frac{1}{q}\right)^j\right) \geqslant 1-\frac{q-1}{j}\ ,$$

откуда

$$\prod_{j=q}^{n}\frac{1}{1-\frac{t+t^2+\cdots+t^j}{j}} \leqslant \prod_{j=q}^{n}\frac{j}{j-q+1} = \frac{q\cdot(q+1)\cdot\ldots\cdot n}{1\cdot 2\cdot\ldots\cdot(n-q+1)} = \binom{n}{q-1}.$$

При $j < q$ оценим знаменатель в произведении иначе:

$$1-\frac{t+t^2+\cdots+t^j}{j} = 1-\frac{q}{j}\cdot(t-t^{j+1}) = \frac{j/q-1+1/q+(1-1/q)^{j+1}}{j/q} =$$
$$= \frac{q}{j}\cdot\left(\binom{j+1}{2}\cdot\frac{1}{q^2}-\binom{j+1}{3}\cdot\frac{1}{q^3}+\ldots\right)$$

В скобках стоит знакочередующаяся сумма, причём модули слагаемых убывают.

Поэтому первые два слагаемых дают нижнюю оценку всей суммы. Получаем

$$1 - \frac{t + t^2 + \cdots + t^j}{j} \geqslant \frac{1}{j} \cdot \left(\frac{(j+1)j}{2} \cdot \frac{1}{q} - \frac{(j+1)j(j-1)}{6} \cdot \frac{1}{q^2} \right) \geqslant \frac{j}{3q} \,.$$

Поэтому, используя простую оценку факториала $n! \geqslant (n/3)^n$, получаем

$$\prod_{j=1}^{q-1} \frac{1}{1 - \frac{t+t^2+\cdots+t^j}{j}} \leqslant \prod_{j=1}^{q-1} \frac{3q}{j} \leqslant \frac{(3q)^q}{q!} \leqslant 9^q.$$

Соберём эти оценки вместе:

$$g(k,n) \leqslant e^{(k+1)/(q-1)} \cdot q \cdot \binom{n}{q-1} \cdot 9^q.$$

Выберем $q = \lceil \sqrt{k+1} \rceil \geqslant 2$ при $k > 0$. Тогда

$$\left| \frac{k+1}{q-1} - \sqrt{k} \right| \leqslant 1, \quad \left| q - \sqrt{k} \right| \leqslant 1$$

и потому

$$g(k,n) \leqslant \exp\left(\sqrt{k} + \ln\sqrt{k} + (\sqrt{k}-1)\ln n + \sqrt{k}\ln 9 + O(\ln n), \right)$$

откуда следует оценка теоремы 10.8.

Литература

[1] Верещагин Н.К., Шень А. *Лекции по математической логике и теории алгоритмов. Часть 1. Начала теории множеств.* М.: МЦНМО, 2012.

[2] Гурвич В.А., Карзанов А.В., Хачиян Л. Г. Циклические игры и нахождение минимаксных средних циклов в ориентированных графах. *ЖВМиМФ*, 28(9):1407–1417, 1988.

[3] Кановей В.Г. *Аксиома выбора и аксиома детерминированности.* М.: Наука, 1984.

[4] S. Arora and B. Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[5] H. Björklund, S. Sandberg, and V. Sergei. Randomized subexponential algorithms for infinite games. Technical Report 2004–09, DIMACS, 2004.

[6] H. Björklund and S. G. Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. *Discrete Applied Mathematics*, 155:210–229, 2007.

[7] M. Bojańczyk and W. Czerwiński. An automata toolbox. A book of lecture notes, available at `https://www.mimuw.edu.pl/ bojan/upload/reduced-may-25.pdf`, 2018.

[8] M. Jurdzinski and R. Lazic. Succinct progress measures for solving parity games. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE, 2017.

[9] E. Kohlberg. Invariant half-lines of nonexpansive piecewise-linear transformations. *Math. of OR*, 5(3):366–372, 1980.

[10] U. Larsson and W. J. Wätlund. From heaps of mathces to the limits of computability. *The Electronic Journal of Combinatorics*, 20(3):#P41, 2013.

[11] W. Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Information and Computation*, 117(1):151–155, 1995.

[12] D. A. Martin. A purely inductive proof of borel determinacy. In *Recursion Theory, Proceedings of Symposia in Pure Mathematics*, volume 42, pages 303–308. American Mathematical Society, 1985.

[13] J. Matousek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. *Algorithmica*, 16(4–5):498–516, 1996.

[14] P. Parys. Parity games: Zielonka's algorithm in quasi-po;ynomial time. *arXiv preprint:1904.12446*, 2019.

[15] V. Petersson and S. G. Vorobyov. A randomized subexponential algorithm for parity games. *Nordic Journal of Computing*, 8(3):324–345, 2001.

[16] T. J. Schaefer. On the complexity of some two-person perfect-information games. *Journal of Computer and System Sciences*, 16:185–225, 1978.

[17] M. Sipser. *Introduction to the Theory of Computation*. Course Technology, second edition, 2006.

Список версий текста с краткими указаниями изменений

Версия 17.01.2020. Исправлено много мелких опечаток и неточностей. Исправлено ошибочное доказательство полноты дерева сильно связных компонент: лемма 8.5 в предыдущей версии, теперь лемма 8.6 — нумерация изменилась так как вставлено ключевое пропущенное утверждение о связи вершин графа и листьев дерева ССК (спасибо Петру Смирнову за внимательность).
Версия 24.11.2019. Исходная версия этого черновика.