Алгоритмы: дополнительные главы

Константин Макарычев

1. Теория вероятностей

1.1. Напоминания

- Вероятностный алгоритм можно себе представлять так: до начала работы выбирается случайный объект (скажем, длинная битовая строка), и потом он используется в алгоритме.

- Случайный объект задаётся *вероятностным пространством* $(\Omega, \mathcal{F}, P)$. Здесь Ω — пространство исходов, которое мы будем считать конечным, $\mathcal{F}$ — некоторое семейство подмножеств Ω, которые называются *событиями*, а P — функция, которая каждому событию ставит в соответствие число. Для таких пространств есть аксиомы Колмогорова, но в конечном случае события — это любые подмножества Ω, каждому исходу из Ω присвоена некоторая неотрицательная вероятность, а $P(X)$ — это сумма вероятностей всех исходов из X.

  Например, если алгоритм использует N случайных битов, то Ω состоит из всех последовательностей N нулей и единиц, и каждый из 2^N исходов имеет вероятность 2^{-N}.

- Для бесконечных Ω — скажем, отрезка $[0, 1]$ с равномерным распределением — ситуация сложнее, уже не все подмножества будут входить в $\mathcal{F}$, а только измеримые (по Лебегу), а вероятность множества — его мера. Для распределения с некоторой плотностью надо брать не меру, а интеграл по (измеримому) множеству от функции плотности.

- Значение *случайной величины* зависит от того, какой исход выбран, то есть случайная величина есть функция $X : \Omega \to \mathbb{R}$ (для конечных пространств — любая). Например, для пространства битовых строк длины N число единиц в строке является случайной величиной. Если S — некоторое множество, то вероятность попадания X в S определяется как вероятность множества тех исходов, которые отображаются в S:
$$\Pr\{X \in S\} = \Pr\{\omega : X(\omega) \in S\}$$

- Если $X(\omega)$ — число единиц в трёхбитовой строке ω, то X принимает значение 2 с вероятностью 3/8, потому что есть ровно три строки с двумя единицами.

- Union bound: вероятность события "A или B" не больше суммы вероятностей событий A и B:
$$\Pr(A \cup B) \leqslant \Pr(A) + \Pr(B).$$
(правая часть может быть больше за счёт одновременного выполнения A и B).

- *Условная вероятность* события A при условии события B определяется как $\Pr(A \cap B)/\Pr(B)$. Для равновероятных исходов это доля A-исходов среди B-исходов. Это имеет смысл при $\Pr(B) > 0$.
- События A и B независимы, если $\Pr(A|B) = \Pr(A)$, то есть $\Pr(A \cap B) = \Pr(A) \cdot \Pr(B)$.
- События $A_1, \dots, A_n$ *попарно независимы*, если A_i и A_j независимы при любых $i \neq j$, то есть
$$\Pr(A_i \cap A_j) = \Pr(A_i) \cdot \Pr(A_j).$$
- События $A_1, \dots, A_n$ *независимы в совокупности*, если
$$\Pr\left(\bigcap_{i \in I} A_i\right) = \prod_{i \in I} \Pr(A_i)$$
- Случайные величины $X_1, \dots, X_n$ независимы, если для любых множеств $S_1, \dots, S_n$ выполняется равенство
$$\Pr(X_1 \in S_1, \dots, X_n \in S_n) = \Pr(X_1 \in S_1) \cdot \dots \cdot \Pr(X_n \in S_n).$$
- *Индикатором* события A называется функция $\mathbb{1}_A$, равная единице внутри A и нулю вне:
$$\mathbb{1}_A(\omega) = \begin{cases} 1, \omega \in A; \\ 0, \omega \notin A. \end{cases}$$
- События $A_1, \dots, A_n$ независимы тогда и только тогда, когда
$$\Pr(\mathbb{1}_{A_1} = \sigma_1, \dots, \mathbb{1}_{A_n} = \sigma_n) = \Pr(\mathbb{1}_{A_1} = \sigma_1) \cdot \dots \Pr(\mathbb{1}_{A_n} = \sigma_n),$$
для всех $\sigma_1, \dots, \sigma_n \in \{0, 1\}$, что эквивалентно независимости индикаторов этих событий.
- Попарная независимость слабее независимости в совокупности (Пример: два независимых бита и их `xor`).
- *Математическое ожидание* случайной величины X определяется как
$$\mathbb{E}[X] = \sum_{\omega \in \Omega} X(\omega) Pr(\omega) = \sum_x \Pr(X = x) \cdot x.$$
(Среднюю зарплату можно вычислять суммированием по работникам или как сумму произведений каждой зарплаты на долю работников с такой зарплатой.)
- Для бесконечных пространств и величин с бесконечным числом значений для вычисления математического ожидания нужно интегрировать.
- Математическое ожидание *линейно*:
$$\mathbb{E}[\alpha X] = \alpha \, \mathbb{E}[X]$$
для случайной величины X и числа α, а также
$$\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]$$

- Как и для union bound, для линейности не нужна независимость (событий/величин).
- *Дисперсия* случайной величины X определяется как средний квадрат отклонения от среднего:
$$\mathrm{Var}[X] = \mathbb{E}[(X - \mathbb{E}[X])^2]$$
- Другое выражение:
$$\mathrm{Var}[X] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$$
- Для независимых величин X и Y:
$$\mathbb{E}[X \cdot Y] = \mathbb{E}[X] \cdot \mathbb{E}[Y];$$
$$\mathrm{Var}[X + Y] = \mathrm{Var}[X] + \mathrm{Var}[Y].$$
Независимость существенна: у $X + X$ дисперсия вчетверо больше, чем у X.
- *Условное математическое ожидание* величины X при условии A (все в одном пространстве) определяется как
$$\mathbb{E}[X|A] = \sum_{\omega} X(\omega)\Pr(\omega \in A) = \sum_{\omega \in A} X(\omega)\frac{\Pr(\omega)}{\Pr(A)}$$
(ограничиваемся исходами в A и корректируем вероятности делением на $P(A)$)
- Можно написать
$$\mathbb{E}[X|A] = \frac{\mathbb{E}[X \cdot \mathbb{1}_A]}{\Pr(A)}$$
- Можно рассмотреть математическое ожидание величины X, при условии что другая величина Y равна k:
$$\mathbb{E}[X|Y = k]$$
Это математическое ожидание зависит от k, и если в качестве k взять значение Y, то получится новая случайная величина, которая обозначается $\mathbb{E}[X|Y]$. Другими словами, мы усредняем X по множествам уровня величины Y.
- *Неравенство Маркова*: для неотрицательной случайной величины X и любой границы $t > 0$ верна оценка:
$$\Pr\{X \geqslant t\} \leqslant \frac{\mathbb{E}\,X}{t}$$
(доля людей с зарплатой втрое больше средней не превосходит $1/3$ — иначе уже они дали бы бóльшее среднее; всё это при условии, что зарплата неотрицательна).
- *Неравенство Чебышёва*: для любой случайной величины X и границы $t > 0$ выполнена оценка на вероятность отклонения t или больше:
$$\Pr[|X - \mathbb{E}\,X| \geqslant t] \leqslant \frac{\mathrm{Var}[X]}{t^2}$$
(применяем неравенство Маркова к квадрату отклонения от среднего)
- Это неравенство позволяет доказать простой вариант *закона больших чисел*: если $X_1, \ldots, X_n$ — независимые одинаково распределённые случайные величины, с математическим ожиданием E (одинаковым), а $S_n = X_1 + \ldots + X_n$, то
$$\Pr\{|S_n/n - E| \geqslant \varepsilon\} \to 0$$
при $n \to \infty$ для любого фиксированного ε. (Математическое ожидание S_n/n равно E, а дисперсия убывает пропорционально $1/n$.)

1.2. Задачи

1. Есть n независимых случайных величин, принимающих значения 1 и -1 с вероятностью $1/2$. Найдите математическое ожидание $(X_1 + ... + X_n)^k$ при $k = 1, 2, 3, 4$.

2. Последовательность R_n начиается с $R_1 = 1$, $R_2 = 5$, $R_3 = 12$, а затем $R_{n+3} = R_{n+2} + 3R_{n+1} + 5R_n$. Оцените рост R_n (докажите, например, что $R_n < 3^n$.

3. Найдите среднее число неподвижных точек в случайной перестановке (случайной биекции n-элементного множества на себя, все $n!$ вариантов равновероятны).

4. Честную монету бросают n раз (орлы и решки равновероятны, бросания независимы). Оцените сверху вероятность того, что доля орлов будет не менее 60%, показав, что она экспоненциально убывает с ростом n (то есть имеется оценка $O(c^n)$ при $c < 1$). Совет: сравните с другим распределением вероятностей, где вероятность орла 60%.

5. Имеется компрессор, который сжимает n-битовые строки без потерь (преобразует строку в другую, произвольной длины, по которой можно восстановить исходную — так что реально он может некоторые строки и удлинять); для простоты считаем, что на строках другой длины он не определён. «Экономией» для строки k считаем уменьшение длины при сжатии (и нуль, если строка осталась прежней длины или удлинилась). Покажите, что средняя экономия (по всем n-битовым строкам) есть $O(1)$.

2. Хеширование

2.1. Словари

Словарь (dictionary) — структура данных, хранящая пары (ключ, значение), изначально пустая, и поддерживающая операции:

- Add/Insert (key, value): добавить новую пару (ключ, значение) в словарь;
- Find (key): найти значение, соответствующее данному ключу;
- Delete/Remove (key): удалить пару с данным ключом;
- Update (key,value): обновить значение для ключа key, сделав его равным value.

Можно хранить пары в массиве, но поиск ключа долгий (в среднем половина длины массива).

Допустим, мы хотим хранить список пользователей: по идентификатору пользователя надо найти данные о нём. Это типичная задача для баз данных. Обычно для этого используются сбалансированные деревья поиска (скажем, red-black tree, или АВЛ-деревья, на практике при работе с диском удобны B-деревья), можно использовать skip-листы.

Обычно для этих структур данных время логарифмическое по числу ключей.

2.2. Хеш-таблицы

Можно ли придумать что-то быстрее (если структура данных в памяти)? Как, скажем, компилятор Clang (LLVM) хранит таблицу идентификаторов? Он использует хеш-таблицы, которые позволяют выполнять операции описанные выше за время $O(1)$ «в среднем».

Для начала предположим, что нам известно максимальное число m ключей. Создадим массив размера $n \approx m$ из «корзин» (buckets), и будем там хранить пары (key, value). Точнее, мы будем рассматривать вариант, когда в каждой ячейке хранится односвязный список тех пар, которые «положено» хранить в ней.

Пусть есть хеш-функция, то есть какая-то функция h, которая отображает возможные значения ключей в $[0, n)$, и договоримся, что пары с ключом x надо хранить в ячейке номер $h(x)$.

Добавление: узнаём куда и добавляем в соответствующий список (в любое место)

Поиск ключа: знаем, в каком списке искать — и ищем (линейным поиском)

Удаление элемента с ключом: знаем, в каком списке, ищем и удаляем (список односвязный, но всё равно $O(1)$ действий)

Что мы хотим от хеш-функции? Хорошо бы она отображала каждый ключ в свою ячейку («совершенная» хеш-функция). Тогда никаких проблем и $O(1)$ операций. Но это свойство не только функции, но и хранимых ключей — и гарантировать это можно, только если число возможных значений ключей меньше числа ячеек (иногда такое полезно, но редко).

Для любой хеш-функции с большим множеством ключей может оказаться, что все реально встретившиеся ключи попали в одну ячейку, и хеш-таблица превращается в список.

2.3. Случайные хеш-функции

Идея: будем выбирать хеш-функцию случайно в каком-то классе функций. Тогда число действий (при выполнении данной последовательности обращений к структуре) будет случайной величиной, и чтобы оценивать её, нужны предположения о том, как хеш-функция выбирается.

- В модели *случайного оракула* хеш-функция равновероятно выбирается из множества всех функций с n значениями: для каждого ключа x значение $h(x)$ равномерно распределено в $\{0, 1, \dots, n-1\}$, и при разных x эти случайные величины независимы. Функций таких много, и хранить такую функцию сложнее, чем всю структуру данных, так что это скорее модель для анализа, чем реальный алгоритм.

- Можно взять меньшее семейство, сохранив некоторые полезные свойства. Пусть есть семейство функций H, из которого мы выбираем равновероятно некоторую функцию. Требование *универсальности*: для любой пары ключей $x \neq y$ вероятность коллизии для этой пары и случайно взятой функции из H не больше $1/n$:

$$\forall x \forall y \quad (x \neq y) \Rightarrow \Pr_{h \in H}[h(x) = h(y)] \leqslant 1/n.$$

  Случайный оракул этим свойством обладает: $h(x)$ и $h(y)$ независимы и равномерно распределены на множестве из n элементов и потому равны в точности с вероятностью $1/n$.

  Но можно построить гораздо меньшие универсальные семейства.

2.4. Оценка времени поиска

Пусть даны ключи $x_1, \dots, x_m$ (различные) и ключ y (входящий или не входящий в этот список). Возьмём случайную хеш-функцию h из универсального семейства и посмотрим, сколько операций понадобится для поиска y после того, как в таблицу будет помещены ключи $x_1, \dots, x_m$. Это число — случайная величина (на пространстве хеш-функций), для каждого набора ключей своя.

***Теорема* 1.** *Математическое ожидание этой величины есть* $O(\frac{m-1}{n} + 1)$.

Доказательство. Оценим время поиска ключа y. Мы должны вычислить $h(y)$, обычно это несложно, так что считаем это за $O(1)$ операций. После этого надо искать ключ, и это время пропорционально размеру списка (корзины).[1] Поэтому надо понять, каково математическое ожидание размера списка в ячейке $h(y)$ (напомним, что y и x_i фиксированы, а матожидание берётся по h). Это удобно сделать с помощью индикаторов и линейности матожидания: размер списка (число x_i, попавших в ячейку к y) равен

$$\sum_{i=1}^{m} \mathbb{1}(h(x_i) = h(y)),$$

по линейности его математическое ожидание равно

$$\sum_{i=1}^{m} \mathbb{E}(\mathbb{1}(h(x_i) = h(y)),$$

а входящие в сумму математические ожидания индикаторов (то есть вероятности событий) не больше $1/n$, кроме того единственного места, где $x_i = y$, если такое есть — там это математическое ожидание равно 1. Поэтому сумма не больше $O(\frac{m-1}{n} + 1)$ (в константу можно поместить и $O(1)$ действий, о которых мы говорили раньше). В типичном случае $m \leqslant n$, и тогда выражение в скобках не больше 2 — получается $O(1)$ операций (в среднем по h). □

Та же оценка применима и к операциям удаления или добавления.

1. Строго говоря, ещё нужно $O(1)$ операций на обработку конца списка — даже если список пуст, надо в этом убедиться.

2.5. Rehashing

Что делать, если заранее число ключей неизвестно (компилятор не знает, сколько будет идентификаторов)? Когда число ключей достигает размера таблицы, можно завести новую таблицу, скажем, вдвое большего размера, и все элементы по одному перенести в неё (rehashing). Это дорогое действие (ожидаемое время пропорционально числу элементов). Но зато она выполняется редко, и «амортизированное» число операций по-прежнему линейно.

Пусть мы переписываем таблицу, если число добавленных элементов достигнет степени двойки. Тогда размеры таблиц будут $1, 2, 4, 8, \ldots$ (кончается на степени двойки, большей m), и время на rehashing не больше $1 + 2 + 4 + \ldots + m \leqslant 2m$ (последнюю таблицу уже не надо переписывать). Так что суммарное время остаётся линейным по m.

3. Универсальные семейства: примеры

3.1. Афинные отображения по простому модулю

Пусть ключи выбираются из множества $\{0, 1, \ldots, N-1\}$, а таблица имеет размер $n \ll N$ и индексируется числами из $\{0, \ldots, n\}$.

Как построить универсальное семейство? Теорема Чебышёва (постулат Бертрана) позволяет выбрать простое число p от N до $2N$. В качестве хеш-функции возьмём функцию

$$x \mapsto h_{a,b}(x) = [(ax + b) \bmod p] \bmod n.$$

где a и b — случайные остатки по модулю p, причём $a \neq 0$ (всего $(p-1)p$ равновероятных вариантов.

Теорема 2. *Это универсальное семейство.*

Доказательство. Пусть $x, y \in \{0, 1, \ldots, N-1\}$ — два различных ключа. Надо оценить вероятность коллизии, то есть оценить число пар (a, b), при которых $h_{a,b}(x) = h_{a,b}(y)$.

Пусть сначала нет последней операции (деления на n с остатком). Получаются две случайных величины $u = ax + b \bmod p$ и $v = ay + b \bmod p$ — функции от исхода (a, b) (а ключи x и y фиксированы). Поскольку p простое, то тут всё очень регулярно, остатки по модулю p образуют поле $\mathbb{Z}_p$.

Как распределена пара (u, v)? В терминах линейной алгебры можно сказать, что отображение

$$\begin{pmatrix} a \\ b \end{pmatrix} \mapsto \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} ax + b \\ ay + b \end{pmatrix} = \begin{pmatrix} x & 1 \\ y & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}$$

является невырожденным линейным оператором (определитель равен $x - y \neq 0$), и потому является взаимно однозначным отображением $\mathbb{Z}_p \to \mathbb{Z}$ в себя, при котором подпространство $a = 0$ (которое мы выбросили) соответствует подпространству $u = v$. Если не верить в линейную алгебру, можно явно решить систему уравнений, и обратное отображение будет

$$\begin{pmatrix} a \\ b \end{pmatrix} = \frac{1}{x - y} \begin{pmatrix} 1 & -1 \\ -y & x \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

Поэтому нам осталось доказать такое утверждение: *берём два случайных различных элемента $\mathbb{Z}_p$; вероятность того, что они сравнимы по модулю n, не больше* $1/n$.

Это следует из того, что для любого элемента $u \in \mathbb{Z}$ вероятность того, что случайно взятый другой элемент сравним с u по модулю n, не больше $1/n$.

Элементы, сравнимые с u, образуют арифметическую прогрессию с шагом n. Сколько из них попадает в интервал $\{0, 1, \dots, p-1\}$? Пусть это

$$u', u'+n, u'+2n, \dots, \cancel{u}, \dots, u'+kn$$

при каком-то k. Тогда их k (было $k+1$ и один вычеркнут), и $kn \leqslant p-1$, то есть $k \leqslant (p-1)/n$. Всего элементов, отличных от u, имеется $p-1$ штук, то есть доля не больше $1/n$, что и требовалось доказать. □

Зачем нам было нужно, чтобы $N \leqslant p \leqslant 2N$? Первое неравенство нужно, чтобы разные ключи были разными элементами поля — а второе не нужно вовсе (но чем меньше p, тем меньше функций в семействе — нужно «меньше случайности»).

Первый шаг — переход к p — можно объяснить так: универсальное семейство останется универсальным, если некоторые ключи выбросить, так что можно начать с большего и «более регулярного» с алгебраической точки зрения множества.

3.2. Случайные линейные операторы

Пусть ключи представляют собой m-битовые строки, а ячейки индексируются d-битовыми строками (так что размер таблицы $n = 2^d$ является степенью двойки).

В качестве хеш-функции возьмём случайное линейное преобразования m-мерного пространства над $\mathbb{Z}_2$ в d-мерное над тем же полем:

$$h_A(x) = Ax$$

(умножение по модулю 2, то есть можно умножить в целых числах, а потом взять по модулю 2).

***Теорема* 3.** *Семейство $h_A : \mathbb{Z}_2^m \to \mathbb{Z}_2^d$ для случайной матрицы A размером $d \times m$ универсально.*

Доказательство. Какая будет вероятность коллизии двух различных ключей x, y? Другими словами, какова вероятность того, что $Ax = Ay$ для фиксированных различных x, y и случайной матрицы A? По линейности это равносильно тому, что $Az = 0$ для вектора $z = y - x$; заметим, что $z \neq 0$, потому что $x \neq y$.

Случайная матрица A состоит из d независимых случайных битовых строк $a_1, \dots, a_d$. Каждая строка, a_i умножаясь на z, даёт одну из d координат вектора z, и координата эта представляет собой XOR нескольких битов из a_i (каких? тех, где у z стоят единицы). Сумма независимых случайных битов (нули и единицы равновероятны) сама представляет собой случайный бит, и потому все d координат вектора Az обращаются в нуль с вероятностью $1/2$ и независимо, так что одновременно это происходит с вероятностью 2^{-d}, что и требовалось. □

Семейство хорошее и простое, но довольно большое (2^{dm} функций, так что нужно dm случайных битов, чтобы задать хеш-функцию из семейства).

3.3. Задачи

- В криптографии тоже используют хеш-функции, например, хранят не пароли пользователей, а значения какой-то заранее выбранной хеш-функции от этих паролей. Почему в этой схеме не стоит использовать хеш-функции из наших семейств?

- Приведите пример четырёх случайных величин с двумя значениями, в котором любая тройка величин независима, но однозначно определяет значение отсутствующей величины.

- Пусть p — простое число, и $k < p$. Постройте пример p случайных величин, равномерно распределённых в $\mathbb{Z}_p$, в котором любые k величин независимы, но однозначно определяют значения остальных $p - k$ величин.

- Для случайных линейных операторов хеш-функция задаётся случайной матрицей, то есть md битами. Покажите, что можно обойтись меньшим числом битов ($m + d - 1$), сохраняя свойство универсальности. (Совет: рассмотрите матрицы, у которых в любой диагонали, параллельной главной, все элементы одинаковы.)

- Дано двоичное дерево, в вершинах которого стоят целые числа. Требуется найти максимальное расстояние (по дереву) между вершинами с одинаковыми пометками за время $O(n \log n)$, где n — число вершин дерева. (Возможная мотивация: хорошо ли дерево кластеризует пометки.) Немного более простая задача: время работы $O(nh)$, где h — высота дерева.