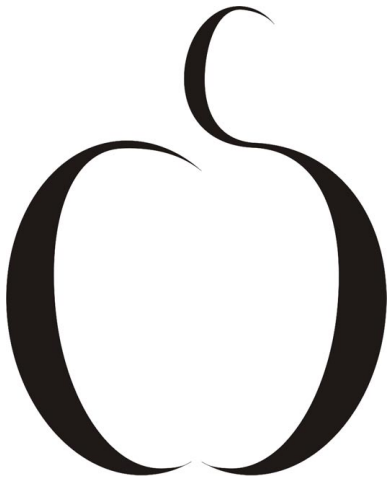
с/к “Эффективные алгоритмы”

Лекция 18: Нахождение максимального потока

А. Куликов

Computer Science клуб при ПОМИ
http://logic.pdmi.ras.ru/~infclub/



План лекции

План лекции

1. Введение

2. Метод Форда-Фалкерсона

План лекции

1. Введение
2. Метод Форда-Фалкерсона
3. Алгоритм проталкивания предпотока

План лекции

1. Введение
2. Метод Форда-Фалкерсона
3. Алгоритм проталкивания предпотока
4. Алгоритм “поднять-и-в-начало”

План лекции

Формулировка задачи

Определение

Формулировка задачи

Определение

- Сетью (flow network) называется ориентированный граф, каждому ребру (u, v) которого сопоставлено неотрицательное число $c(u, v)$ (если $(u, v) \notin E$, полагаем $c(u, v) = 0$), называемое пропускной способностью (capacity), с двумя выделенными вершинами, называемыми исток s (source) и сток t (sink).

Формулировка задачи

Определение

- Сетью (flow network) называется ориентированный граф, каждому ребру (u, v) которого сопоставлено неотрицательное число $c(u, v)$ (если $(u, v) \notin E$, полагаем $c(u, v) = 0$), называемое пропускной способностью (capacity), с двумя выделенными вершинами, называемыми исток s (source) и сток t (sink).
- Потоком (flow) в сети $G = (V, E)$ называется функция $f : V \times V \to \mathbb{R}$, удовлетворяющая следующим свойствам:

Формулировка задачи

Определение

- Сетью (flow network) называется ориентированный граф, каждому ребру (u, v) которого сопоставлено неотрицательное число $c(u, v)$ (если $(u, v) \notin E$, полагаем $c(u, v) = 0$), называемое пропускной способностью (capacity), с двумя выделенными вершинами, называемыми исток s (source) и сток t (sink).
- Потоком (flow) в сети $G = (V, E)$ называется функция $f : V \times V \to \mathbb{R}$, удовлетворяющая следующим свойствам:
  - пропускные способности не нарушены: $\forall u, v \in V, f(u, v) \leq c(u, v)$;

Формулировка задачи

Определение

- Сетью (flow network) называется ориентированный граф, каждому ребру (u, v) которого сопоставлено неотрицательное число $c(u, v)$ (если $(u, v) \notin E$, полагаем $c(u, v) = 0$), называемое пропускной способностью (capacity), с двумя выделенными вершинами, называемыми исток s (source) и сток t (sink).
- Потоком (flow) в сети $G = (V, E)$ называется функция $f : V \times V \to \mathbb{R}$, удовлетворяющая следующим свойствам:
  - пропускные способности не нарушены: $\forall u, v \in V, f(u, v) \leq c(u, v)$;
  - кососимметричность: $\forall u, v \in V, f(u, v) = -f(v, u)$;

Формулировка задачи

Определение

- Сетью (flow network) называется ориентированный граф, каждому ребру (u, v) которого сопоставлено неотрицательное число $c(u, v)$ (если $(u, v) \notin E$, полагаем $c(u, v) = 0$), называемое пропускной способностью (capacity), с двумя выделенными вершинами, называемыми исток s (source) и сток t (sink).
- Потоком (flow) в сети $G = (V, E)$ называется функция $f : V \times V \to \mathbb{R}$, удовлетворяющая следующим свойствам:
  - пропускные способности не нарушены: $\forall u, v \in V, f(u, v) \leq c(u, v)$;
  - кососимметричность: $\forall u, v \in V, f(u, v) = -f(v, u)$;
  - сохранение потока: $\forall u \in V \setminus \{s, t\}, \sum_{v \in V} f(u, v) = 0$.

Формулировка задачи

Определение

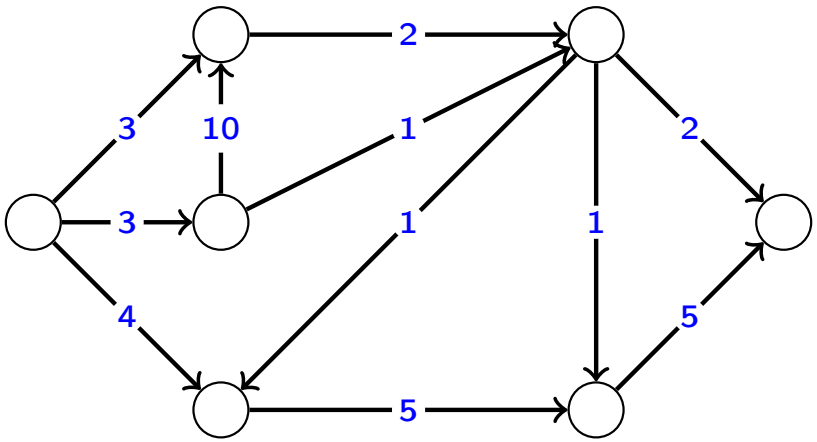
- Сетью (flow network) называется ориентированный граф, каждому ребру (u, v) которого сопоставлено неотрицательное число $c(u, v)$ (если $(u, v) \notin E$, полагаем $c(u, v) = 0$), называемое пропускной способностью (capacity), с двумя выделенными вершинами, называемыми исток s (source) и сток t (sink).
- Потоком (flow) в сети $G = (V, E)$ называется функция $f : V \times V \to \mathbb{R}$, удовлетворяющая следующим свойствам:
  - пропускные способности не нарушены: $\forall u, v \in V, f(u, v) \leq c(u, v)$;
  - кососимметричность: $\forall u, v \in V, f(u, v) = -f(v, u)$;
  - сохранение потока: $\forall u \in V \setminus \{s, t\}, \sum_{v \in V} f(u, v) = 0$.
- Величиной (value) потока называется $\sum_{v \in V} f(s, v)$.

Формулировка задачи

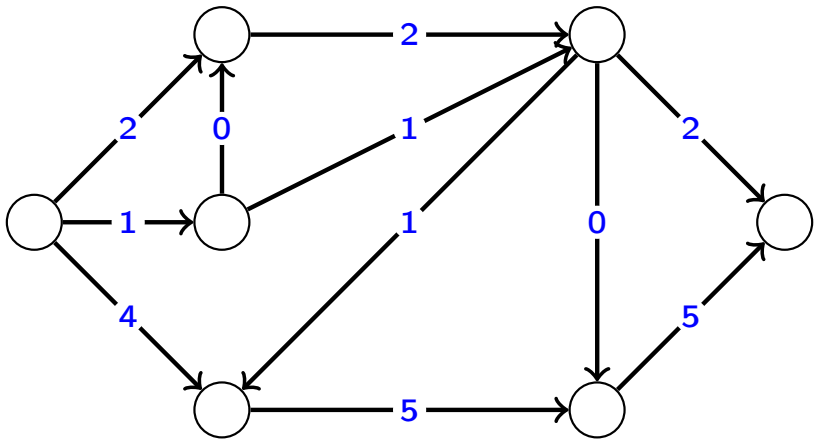
Определение

- Сетью (flow network) называется ориентированный граф, каждому ребру (u, v) которого сопоставлено неотрицательное число $c(u, v)$ (если $(u, v) \notin E$, полагаем $c(u, v) = 0$), называемое пропускной способностью (capacity), с двумя выделенными вершинами, называемыми исток s (source) и сток t (sink).
- Потоком (flow) в сети $G = (V, E)$ называется функция $f : V \times V \to \mathbb{R}$, удовлетворяющая следующим свойствам:
  - пропускные способности не нарушены: $\forall u, v \in V, f(u, v) \leq c(u, v)$;
  - кососимметричность: $\forall u, v \in V, f(u, v) = -f(v, u)$;
  - сохранение потока: $\forall u \in V \setminus \{s, t\}, \sum_{v \in V} f(u, v) = 0$.
- Величиной (value) потока называется $\sum_{v \in V} f(s, v)$.
- Задача о максимальном потоке (maximum flow-problem) заключается в нахождении по данной сети потока максимальной величины.

Пример потока



сеть



поток

Несколько замечаний

Несколько замечаний

Несколько замечаний

Несколько замечаний

- Задача о максимальном потоке является типичной задачей линейного программирования.

Несколько замечаний

Несколько замечаний

- Задача о максимальном потоке является типичной задачей линейного программирования.
- Задача о максимальном потоке в сети с несколькими истоками и стоками легко сводится к задаче с одним истоком и одним стоком (добавим общий исток, из которого направим рёбра бесконечной пропускной способности во все истоки, и общий сток, в который направим рёбра бесконечной пропускной способности из всех стоков).

Несколько замечаний

Несколько замечаний

- Задача о максимальном потоке является типичной задачей линейного программирования.
- Задача о максимальном потоке в сети с несколькими истоками и стоками легко сводится к задаче с одним истоком и одним стоком (добавим общий исток, из которого направим рёбра бесконечной пропускной способности во все истоки, и общий сток, в который направим рёбра бесконечной пропускной способности из всех стоков).
- Такую задачу, однако, не стоит путать с задачей нахождения максимального потока в сети с несколькими веществами (multicommodity flow).

План лекции

Метод Форда-Фолкерсона

FORD-FULKERSON-METHOD(G, s, t)

1 положить поток f равным 0
2 **while** существует дополняющий путь p
3 **do** дополнить f вдоль p
4 **return** f

Остаточная сеть

Определение

Пусть дана сеть G и поток f в ней.

Остаточная сеть

Определение

Пусть дана сеть G и поток f в ней.

- Остаточной пропускной способностью из u в v (residual capacity of (u, v)) называется величина $c_f(u, v) = c(u, v) - f(u, v)$.

Остаточная сеть

Определение

Пусть дана сеть G и поток f в ней.

- Остаточной пропускной способностью из u в v (residual capacity of (u, v)) называется величина $c_f(u, v) = c(u, v) - f(u, v)$.
- Остаточной сетью (residual network) сети G, порожденной потоком f, называется сеть $G_f = (V, E_f)$, где

$$E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\}.$$

Остаточная сеть

Определение

Пусть дана сеть G и поток f в ней.

- Остаточной пропускной способностью из u в v (residual capacity of (u, v)) называется величина $c_f(u, v) = c(u, v) - f(u, v)$.
- Остаточной сетью (residual network) сети G, порожденной потоком f, называется сеть $G_f = (V, E_f)$, где

$$E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\}.$$

Замечания

Остаточная сеть

Определение

Пусть дана сеть G и поток f в ней.

- Остаточной пропускной способностью из u в v (residual capacity of (u, v)) называется величина $c_f(u, v) = c(u, v) - f(u, v)$.
- Остаточной сетью (residual network) сети G, порожденной потоком f, называется сеть $G_f = (V, E_f)$, где

$$E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\}.$$

Замечания

- $c_f(u, v)$ может превосходить $c(u, v)$.

Остаточная сеть

Определение

Пусть дана сеть G и поток f в ней.

- Остаточной пропускной способностью из u в v (residual capacity of (u, v)) называется величина $c_f(u, v) = c(u, v) - f(u, v)$.
- Остаточной сетью (residual network) сети G, порожденной потоком f, называется сеть $G_f = (V, E_f)$, где

$$E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\}.$$

Замечания

- $c_f(u, v)$ может превосходить $c(u, v)$.
- Остаточное ребро не обязано быть ребром исходной сети.

Дополняющие пути

Определение

Дополняющие пути

Определение

- Дополняющим путём (augmenting path) называется простой путь из истока в сток в остаточной сети G_f.

Дополняющие пути

Определение

- Дополняющим путём (augmenting path) называется простой путь из истока в сток в остаточной сети G_f.
- Остаточной пропускной способностью дополняющего пути p (residual capacity of f) называется величина наибольшего потока, который можно пропустить по p: $c_f(p) = \min\{c_f(u, v) : (u, v) \in p\}$.

Дополняющие пути

Определение

- Дополняющим путём (augmenting path) называется простой путь из истока в сток в остаточной сети G_f.
- Остаточной пропускной способностью дополняющего пути p (residual capacity of f) называется величина наибольшего потока, который можно пропустить по p: $c_f(p) = \min\{c_f(u, v) : (u, v) \in p\}$.

Основное свойство дополняющего пути

Если в остаточной сети есть дополняющий путь, то текущий поток не максимален.

Разрезы в сетях

Определение

Разрезы в сетях

Определение

- Разрезом (cut) сети называется разбиение множества вершин на две части S и $T = V \setminus S$, таких что $s \in S$ и $t \in T$.

Разрезы в сетях

Определение

- Разрезом (cut) сети называется разбиение множества вершин на две части S и $T = V \setminus S$, таких что $s \in S$ и $t \in T$.
- Пропускной способностью разреза (cut capacity) называют сумму пропускных способностей рёбер из S в T.

Разрезы в сетях

Определение

- Разрезом (cut) сети называется разбиение множества вершин на две части S и $T = V \setminus S$, таких что $s \in S$ и $t \in T$.
- Пропускной способностью разреза (cut capacity) называют сумму пропускных способностей рёбер из S в T.
- Величиной потока через разрез (cut flow) называется сумма потоков по всем рёбрам, пересекающим разрез.

Разрезы в сетях

Определение

- Разрезом (cut) сети называется разбиение множества вершин на две части S и $T = V \setminus S$, таких что $s \in S$ и $t \in T$.
- Пропускной способностью разреза (cut capacity) называют сумму пропускных способностей рёбер из S в T.
- Величиной потока через разрез (cut flow) называется сумма потоков по всем рёбрам, пересекающим разрез.
- Минимальным разрезом (minimum cut) называется разрез наименьшей пропускной способности.

Разрезы в сетях

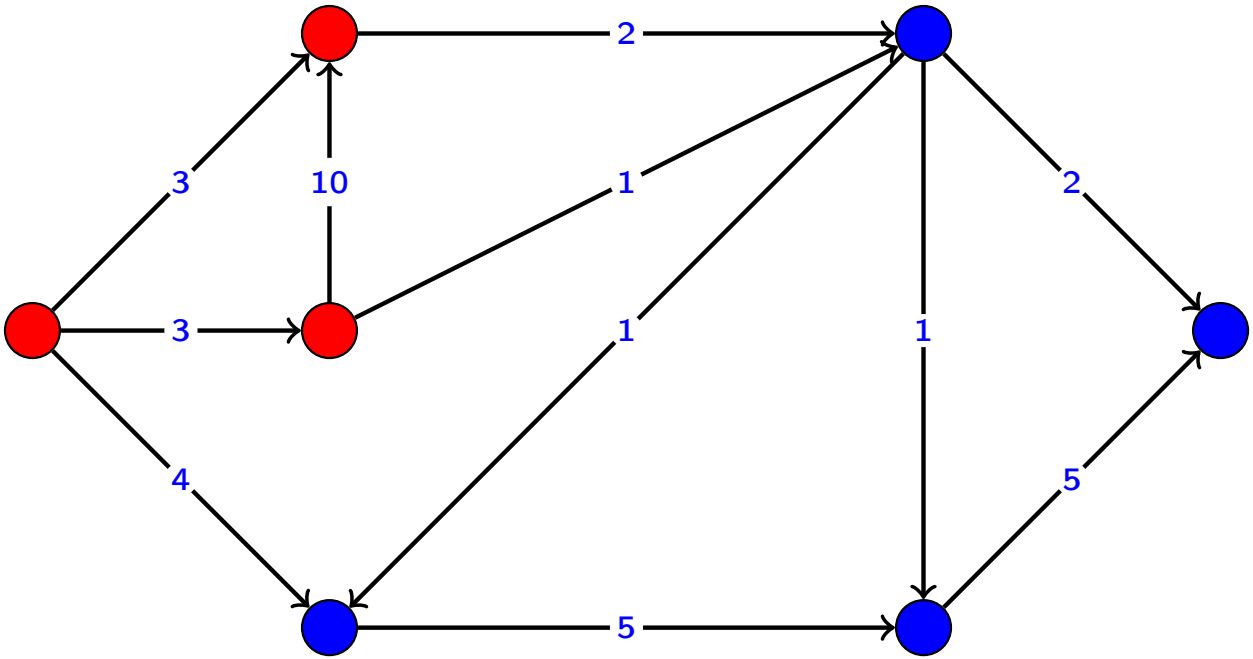
Определение

- Разрезом (cut) сети называется разбиение множества вершин на две части S и $T = V \setminus S$, таких что $s \in S$ и $t \in T$.
- Пропускной способностью разреза (cut capacity) называют сумму пропускных способностей рёбер из S в T.
- Величиной потока через разрез (cut flow) называется сумма потоков по всем рёбрам, пересекающим разрез.
- Минимальным разрезом (minimum cut) называется разрез наименьшей пропускной способности.

Замечание

Легко видеть, что поток через любой разрез совпадает с величиной потока.

Пример минимального разреза



Теорема о максимальном потоке и минимальном разрезе

Теорема

Пусть f — поток сети G. Следующие утверждения равносильны:

Теорема о максимальном потоке и минимальном разрезе

Теорема

Пусть f — поток сети G. Следующие утверждения равносильны:

1. Поток f максимален.

Теорема о максимальном потоке и минимальном разрезе

Теорема

Пусть f — поток сети G. Следующие утверждения равносильны:

1. Поток f максимален.
2. Остаточная сеть G_f не содержит дополняющих путей.

Теорема о максимальном потоке и минимальном разрезе

Теорема

Пусть f — поток сети G. Следующие утверждения равносильны:

1. Поток f максимален.
2. Остаточная сеть G_f не содержит дополняющих путей.
3. Существует разрез (S, T), для которого $|f| = c(S, T)$.

Теорема о максимальном потоке и минимальном разрезе

Теорема

Пусть f — поток сети G. Следующие утверждения равносильны:

1. Поток f максимален.
2. Остаточная сеть G_f не содержит дополняющих путей.
3. Существует разрез (S, T), для которого $|f| = c(S, T)$.

Доказательство

Идеи доказательства:

Теорема о максимальном потоке и минимальном разрезе

Теорема

Пусть f — поток сети G. Следующие утверждения равносильны:

1. Поток f максимален.
2. Остаточная сеть G_f не содержит дополняющих путей.
3. Существует разрез (S, T), для которого $|f| = c(S, T)$.

Доказательство

Идеи доказательства:

$1 \Rightarrow 2$: Если есть дополняющий путь, то поток не максимален.

Теорема о максимальном потоке и минимальном разрезе

Теорема

Пусть f — поток сети G. Следующие утверждения равносильны:

1. Поток f максимален.
2. Остаточная сеть G_f не содержит дополняющих путей.
3. Существует разрез (S, T), для которого $|f| = c(S, T)$.

Доказательство

Идеи доказательства:

$1 \Rightarrow 2$: Если есть дополняющий путь, то поток не максимален.

$2 \Rightarrow 3$: Если дополняющего пути нет, то в качестве S можно взять все вершины, достижимые из s в остаточной сети.

Теорема о максимальном потоке и минимальном разрезе

Теорема

Пусть f — поток сети G. Следующие утверждения равносильны:

1. Поток f максимален.
2. Остаточная сеть G_f не содержит дополняющих путей.
3. Существует разрез (S, T), для которого $|f| = c(S, T)$.

Доказательство

Идеи доказательства:

$1 \Rightarrow 2$: Если есть дополняющий путь, то поток не максимален.

$2 \Rightarrow 3$: Если дополняющего пути нет, то в качестве S можно взять все вершины, достижимые из s в остаточной сети.

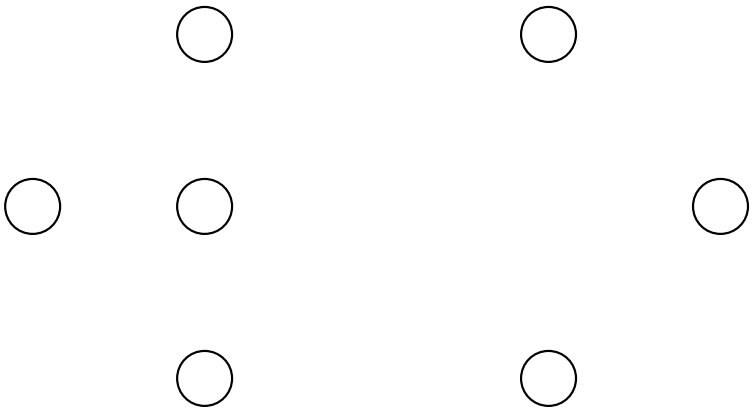
$3 \Rightarrow 1$: Величина любого потока меньше пропускной способности любого разреза. □

Общая схема алгоритма Форда-Фалкерсона

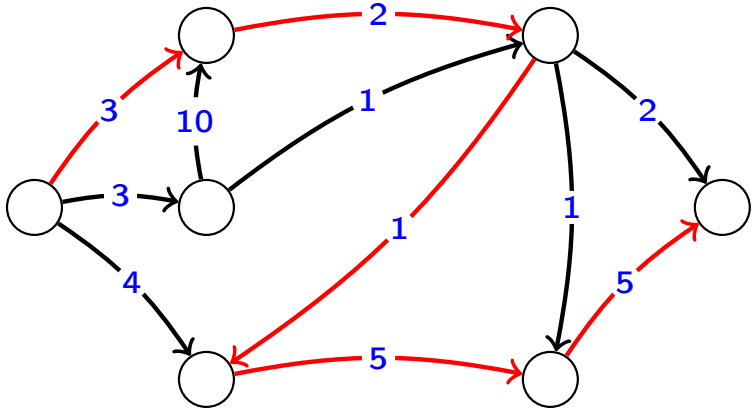
FORD-FULKERSON(G, s, t)

1 **for** каждого ребра $(u, v) \in E$
2 **do** $f[u, v] \leftarrow 0$
3 $f[v, u] \leftarrow 0$
4 **while** в остаточной сети G_f существует путь p из s в t
5 **do** $c_f(p) \leftarrow \min\{c_f(u, v) : (u, v) \in p\}$
6 **for** каждого ребра (u, v) в пути p
7 **do** $f[u, v] \leftarrow f[u, v] + c_f(p)$
8 $f[v, u] \leftarrow -f[u, v]$
9 **return** f

Пример работы алгоритма

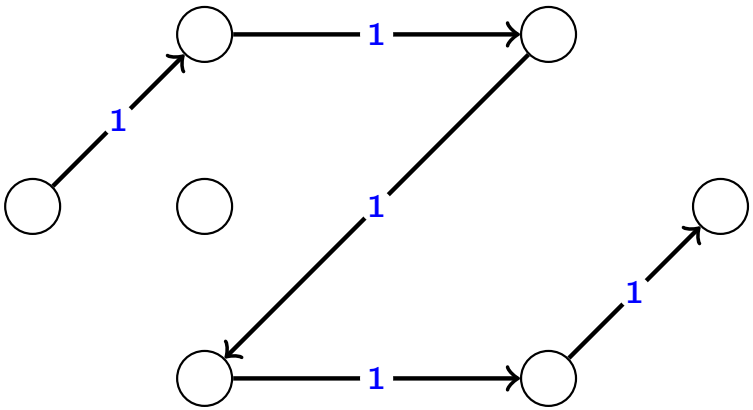


поток

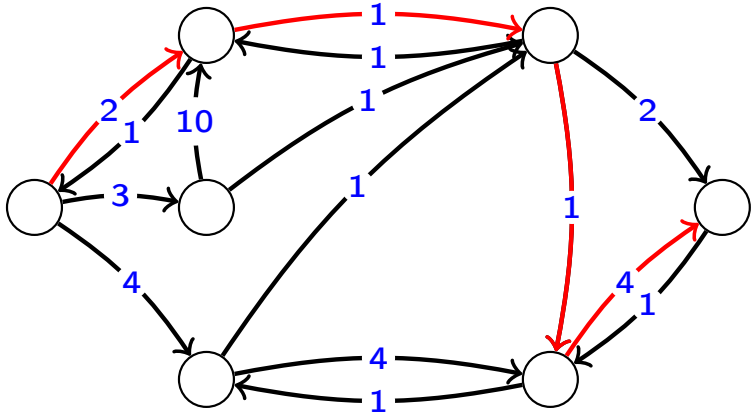


остаточная сеть

Пример работы алгоритма

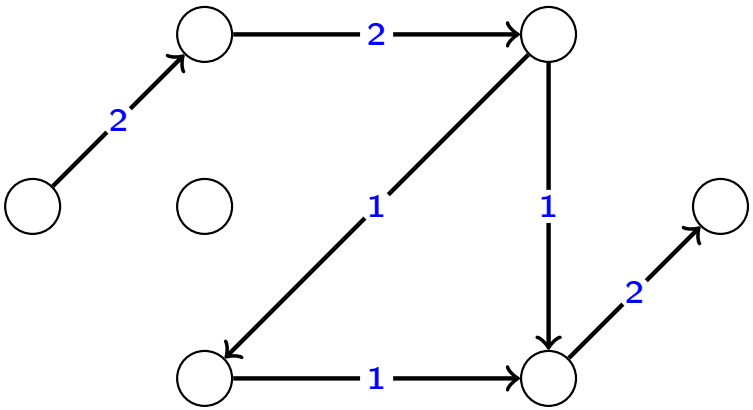


поток

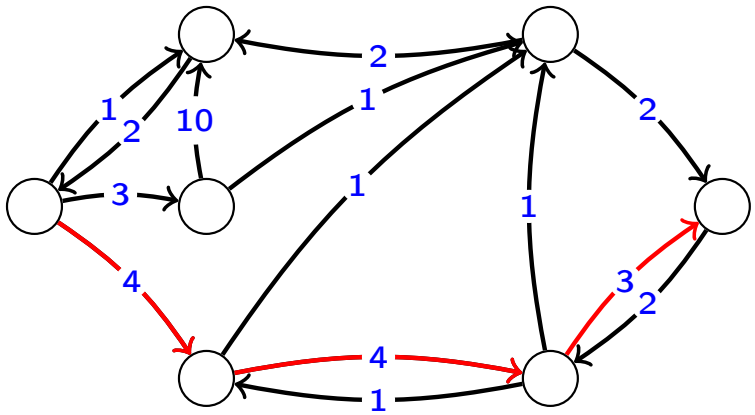


остаточная сеть

Пример работы алгоритма

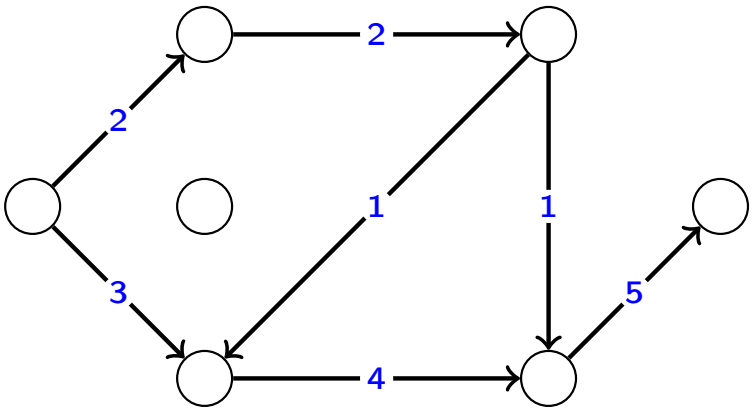


поток

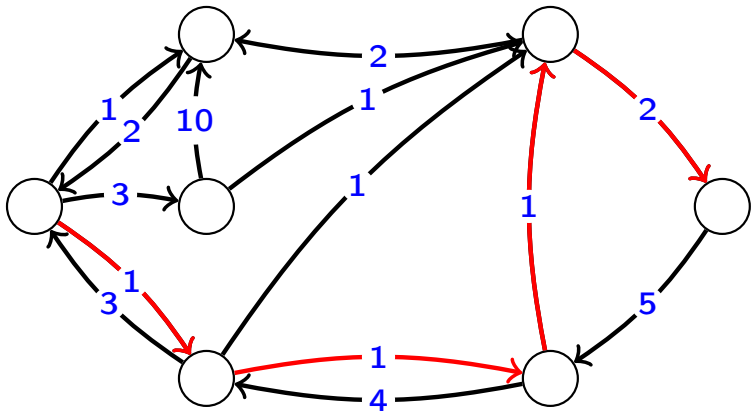


остаточная сеть

Пример работы алгоритма

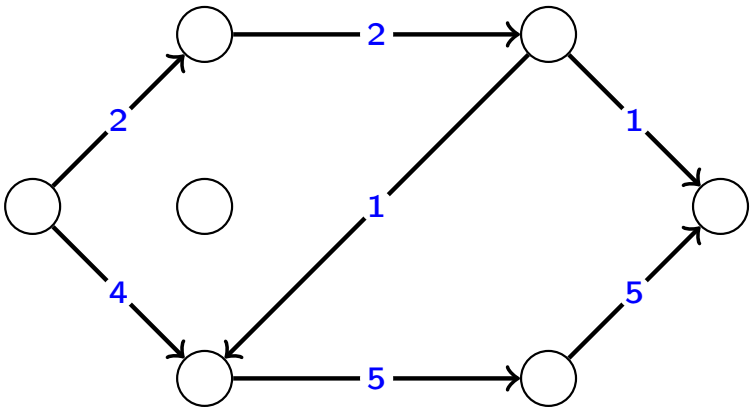


поток

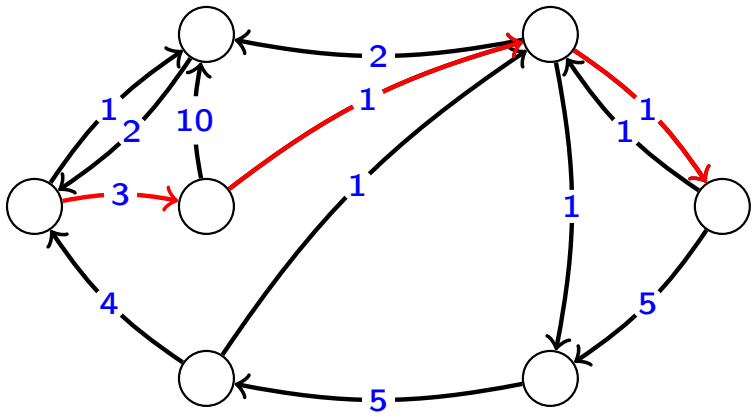


остаточная сеть

Пример работы алгоритма

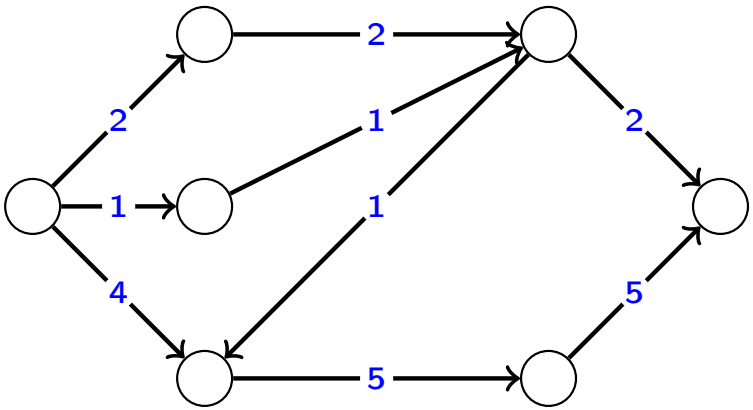


поток

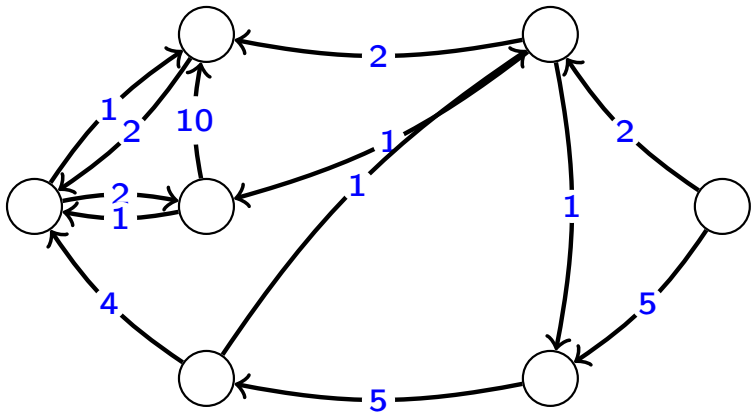


остаточная сеть

Пример работы алгоритма



поток



остаточная сеть

Анализ времени работы

Анализ времени работы

Анализ времени работы

Анализ времени работы

- Допустим, что все пропускные способности являются целыми числами.

Анализ времени работы

Анализ времени работы

- Допустим, что все пропускные способности являются целыми числами.
- Тогда нетрудно видеть, что процедура Ford-Fulkerson имеет время работы $O(|E| \cdot |f^*|)$, где f^* — максимальный поток, поскольку на каждой итерации поток увеличивается хотя бы на 1.

Анализ времени работы

Анализ времени работы

- Допустим, что все пропускные способности являются целыми числами.
- Тогда нетрудно видеть, что процедура Ford-Fulkerson имеет время работы $O(|E| \cdot |f^*|)$, где f^* — максимальный поток, поскольку на каждой итерации поток увеличивается хотя бы на 1.
- Нетрудно, однако, придумать сеть с малым количеством вершин и большими значениями пропускных способностей, для которой такая оценка не будет удовлетворительной.

Анализ времени работы

Анализ времени работы

- Допустим, что все пропускные способности являются целыми числами.
- Тогда нетрудно видеть, что процедура Ford-Fulkerson имеет время работы $O(|E| \cdot |f^*|)$, где f^* — максимальный поток, поскольку на каждой итерации поток увеличивается хотя бы на 1.
- Нетрудно, однако, придумать сеть с малым количеством вершин и большими значениями пропускных способностей, для которой такая оценка не будет удовлетворительной.
- Значит, нужен другой инвариант.

Алгоритм Эдмондса-Карпа

Алгоритм Эдмондса-Карпа

Алгоритм Эдмондса-Карпа является реализацией метода Форда-Фалкерсона, в которой в качестве дополняющего пути выбирается кратчайший путь в остаточной сети (длины всех рёбер равны 1).

Алгоритм Эдмондса-Карпа

Алгоритм Эдмондса-Карпа

Алгоритм Эдмондса-Карпа является реализацией метода Форда-Фалкерсона, в которой в качестве дополняющего пути выбирается кратчайший путь в остаточной сети (длины всех рёбер равны 1).

Теорема

Время работы алгоритма Эдмондса-Карпа есть $O(|V| \cdot |E|^2)$.

Идеи доказательства

Доказательство

Идеи доказательства:

Идеи доказательства

Доказательство

Идеи доказательства:

- Расстояние от истока до любой вершины не убывает в течение работы алгоритма.

Идеи доказательства

Доказательство

Идеи доказательства:

- Расстояние от истока до любой вершины не убывает в течение работы алгоритма.
- Назовём ребро дополняющего пути критическим, если его пропускная способность совпадает с пропускной способностью пути.

Идеи доказательства

Доказательство

Идеи доказательства:

- Расстояние от истока до любой вершины не убывает в течение работы алгоритма.
- Назовём ребро дополняющего пути критическим, если его пропускная способность совпадает с пропускной способностью пути.
- Одно и то же ребро может стать критическим в ходе работы алгоритма не более $O(|V|)$ раз.

Идеи доказательства

Доказательство

Идеи доказательства:

- Расстояние от истока до любой вершины не убывает в течение работы алгоритма.
- Назовём ребро дополняющего пути критическим, если его пропускная способность совпадает с пропускной способностью пути.
- Одно и то же ребро может стать критическим в ходе работы алгоритма не более $O(|V|)$ раз.
- Всего же критическим могут быть не более $O(|E|)$ рёбер.

Идеи доказательства

Доказательство

Идеи доказательства:

- Расстояние от истока до любой вершины не убывает в течение работы алгоритма.
- Назовём ребро дополняющего пути критическим, если его пропускная способность совпадает с пропускной способностью пути.
- Одно и то же ребро может стать критическим в ходе работы алгоритма не более $O(|V|)$ раз.
- Всего же критическим могут быть не более $O(|E|)$ рёбер.
- Поскольку каждая итерация требует времени $O(|E|)$, общее время работы есть $O(|V||E|^2)$. □

План лекции

Предпоток

Определение

Предпоток

Определение

- Предпотоком (preflow) называется функция $f : V \times V \to \mathbb{R}$, которая кососимметрична, не нарушает пропускных способностей, а также удовлетворяет ослабленному закону сохранения: $\forall u \in V \setminus \{s\}, f(V, u) = \sum_{v \in V} f(v, u) \geq 0$

Предпоток

Определение

- Предпотоком (preflow) называется функция $f : V \times V \to \mathbb{R}$, которая кососимметрична, не нарушает пропускных способностей, а также удовлетворяет ослабленному закону сохранения: $\forall u \in V \setminus \{s\}, f(V, u) = \sum_{v \in V} f(v, u) \geq 0$
- Величину $e(u) = f(V, u)$ назовём избытком (excess flow) в вершине u.

Предпоток

Определение

- Предпотоком (preflow) называется функция $f : V \times V \to \mathbb{R}$, которая кососимметрична, не нарушает пропускных способностей, а также удовлетворяет ослабленному закону сохранения: $\forall u \in V \setminus \{s\}, f(V, u) = \sum_{v \in V} f(v, u) \geq 0$
- Величину $e(u) = f(V, u)$ назовём избытком (excess flow) в вершине u.
- Вершину, отличную от истока и стока, назовём переполненной (overflowing), если избыток в ней положителен.

Основные идеи

Основные идеи

Основные идеи

Основные идеи

- Итак, в ходе работы алгоритма в каждой вершине будет накапливаться некоторый избыток.

Основные идеи

Основные идеи

- Итак, в ходе работы алгоритма в каждой вершине будет накапливаться некоторый избыток.
- Все вершины также постепенно поднимаются на некоторую высоту.

Основные идеи

Основные идеи

- Итак, в ходе работы алгоритма в каждой вершине будет накапливаться некоторый избыток.
- Все вершины также постепенно поднимаются на некоторую высоту.
- Избыток из вершины можно направить только вниз.

Основные идеи

Основные идеи

- Итак, в ходе работы алгоритма в каждой вершине будет накапливаться некоторый избыток.
- Все вершины также постепенно поднимаются на некоторую высоту.
- Избыток из вершины можно направить только вниз.
- Если в некоторый момент избыток из вершины никуда не направить, алгоритм поднимает эту вершину.

Высота вершин

Определение

Пусть f — предпоток сети G. Функция $h\colon V \to \mathbb{N}_0$ называется высотной функцией (height function) для предпотока f, если $h(s) = |V|$, $h(t) = 0$ и $h(u) \leq h(v) + 1$ для любого ребра (u, v) остаточной сети E_f.

Высота вершин

Определение

Пусть f — предпоток сети G. Функция $h\colon V \to \mathbb{N}_0$ называется высотной функцией (height function) для предпотока f, если $h(s) = |V|$, $h(t) = 0$ и $h(u) \leq h(v) + 1$ для любого ребра (u, v) остаточной сети E_f.

Замечание

По круто идущим вниз рёбрам, таким образом, идёт максимально возможный поток: если $h(u) > h(v) + 1$, то остаточная сеть E_f не содержит ребра (u, v).

Операция проталкивания

PUSH(u, v)

1 $\triangleright$ дано: вершина u переполнена, $c_f(u, v) > 0$, $h[u] = h[v] + 1$
2 $d_f(u, v) \leftarrow \min\{e[u], c_f(u, v)\}$
3 $f[u, v] \leftarrow f[u, v] + d_f(u, v)$
4 $f[v, u] \leftarrow -f[u, v]$
5 $e[u] \leftarrow e[u] - d_f(u, v)$
6 $e[v] \leftarrow e[v] + d_f(u, v)$

Операция проталкивания

PUSH(u, v)

1 $\triangleright$ дано: вершина u переполнена, $c_f(u, v) > 0$, $h[u] = h[v] + 1$
2 $d_f(u, v) \leftarrow \min\{e[u], c_f(u, v)\}$
3 $f[u, v] \leftarrow f[u, v] + d_f(u, v)$
4 $f[v, u] \leftarrow -f[u, v]$
5 $e[u] \leftarrow e[u] - d_f(u, v)$
6 $e[v] \leftarrow e[v] + d_f(u, v)$

Определение

Операция проталкивания

PUSH(u, v)

1 $\triangleright$ дано: вершина u переполнена, $c_f(u, v) > 0$, $h[u] = h[v] + 1$
2 $d_f(u, v) \leftarrow \min\{e[u], c_f(u, v)\}$
3 $f[u, v] \leftarrow f[u, v] + d_f(u, v)$
4 $f[v, u] \leftarrow -f[u, v]$
5 $e[u] \leftarrow e[u] - d_f(u, v)$
6 $e[v] \leftarrow e[v] + d_f(u, v)$

Определение

- Данная операция называется проталкиванием из u в v.

Операция проталкивания

PUSH(u, v)

1 $\triangleright$ дано: вершина u переполнена, $c_f(u, v) > 0$, $h[u] = h[v] + 1$
2 $d_f(u, v) \leftarrow \min\{e[u], c_f(u, v)\}$
3 $f[u, v] \leftarrow f[u, v] + d_f(u, v)$
4 $f[v, u] \leftarrow -f[u, v]$
5 $e[u] \leftarrow e[u] - d_f(u, v)$
6 $e[v] \leftarrow e[v] + d_f(u, v)$

Определение

- Данная операция называется проталкиванием из u в v.
- Проталкивание называется насыщающим (saturating), если в результате ребро становится насыщенным.

Операция подъёма

LIFT(u)

1 $\triangleright$ дано: вершина u переполнена, $\forall (u, v) \in E_f, h[u] \leq h[v]$

2 $h[u] \leftarrow 1 + \min\{h[v] \mid (u, v) \in E_f\}$

Операция подъёма

LIFT(u)

1 $\triangleright$ дано: вершина u переполнена, $\forall (u, v) \in E_f, h[u] \leq h[v]$
2 $h[u] \leftarrow 1 + \min\{h[v] \mid (u, v) \in E_f\}$

Замечание

Операция подъёма

LIFT(u)

1 $\triangleright$ дано: вершина u переполнена, $\forall (u, v) \in E_f, h[u] \leq h[v]$
2 $h[u] \leftarrow 1 + \min\{h[v] \mid (u, v) \in E_f\}$

Замечание

- $e[u] = f(V, u) > 0$, значит, существует такая вершина v, что $f(v, u) > 0$.

Операция подъёма

LIFT(u)

1 ▷ дано: вершина u переполнена, $\forall (u, v) \in E_f, h[u] \leq h[v]$
2 $h[u] \leftarrow 1 + \min\{h[v] \mid (u, v) \in E_f\}$

Замечание

- $e[u] = f(V, u) > 0$, значит, существует такая вершина v, что $f(v, u) > 0$.
- Тогда $c_f(u, v) = c(u, v) - f(u, v) = c(u, v) + f(v, u) > 0$, а следовательно, $(u, v) \in E_f$.

Операция подъёма

LIFT(u)

1 ▷ дано: вершина u переполнена, $\forall (u, v) \in E_f, h[u] \leq h[v]$
2 $h[u] \leftarrow 1 + \min\{h[v] \mid (u, v) \in E_f\}$

Замечание

- $e[u] = f(V, u) > 0$, значит, существует такая вершина v, что $f(v, u) > 0$.
- Тогда $c_f(u, v) = c(u, v) - f(u, v) = c(u, v) + f(v, u) > 0$, а следовательно, $(u, v) \in E_f$.
- Значит, минимум берётся по непустому множеству.

Общая схема алгоритма

GENERIC-PREFLOW-PUSH(G)

1 положить поток по всем выходящим из истока рёбрам
равным их пропускной способности,
по всем остальным — равным нулю
2 исток поднять на высоту $|V|$, все остальные вершины
оставить на нулевой высоте
3 **while** возможны операции подъёма или проталкивания
4 **do** выполнить одну из этих операций

Корректность

Корректность

Корректность

Корректность

- Если u переполнена, то в ней возможно либо проталкивание, либо подъём.

Корректность

Корректность

- Если u переполнена, то в ней возможно либо проталкивание, либо подъём.
- В течение работы алгоритма высотная функция остаётся таковой и не убывает ни для какой вершины.

Корректность

Корректность

- Если u переполнена, то в ней возможно либо проталкивание, либо подъём.
- В течение работы алгоритма высотная функция остаётся таковой и не убывает ни для какой вершины.
- В течение работы алгоритма в остаточной сети не существует пути из истока в сток: если бы был, то был бы и путь длины менее $|V|$, высота вдоль которого падала бы с $|V|$ до нуля, но тогда бы в этом пути нашлось ребро с перепадом высот хотя бы 2.

Корректность

Корректность

- Если u переполнена, то в ней возможно либо проталкивание, либо подъём.
- В течение работы алгоритма высотная функция остаётся таковой и не убывает ни для какой вершины.
- В течение работы алгоритма в остаточной сети не существует пути из истока в сток: если бы был, то был бы и путь длины менее $|V|$, высота вдоль которого падала бы с $|V|$ до нуля, но тогда бы в этом пути нашлось ребро с перепадом высот хотя бы 2.
- Когда алгоритм закончит работу, текущий предпоток будет потоком. Поскольку в остаточной сети не будет дополняющих путей, он будет максимальным.

Оценка высоты

Лемма

Высота любой вершины в течение работы алгоритма не превосходит $2|V| - 1$.

Оценка высоты

Лемма

Высота любой вершины в течение работы алгоритма не превосходит $2|V| - 1$.

Доказательство

Оценка высоты

Лемма

Высота любой вершины в течение работы алгоритма не превосходит $2|V| - 1$.

Доказательство

- Если вершина u переполнена, то в G_f есть простой пусть из u в s (поскольку в G есть путь из s в u, по которому идёт положительный поток).

Оценка высоты

Лемма

Высота любой вершины в течение работы алгоритма не превосходит $2|V| - 1$.

Доказательство

- Если вершина u переполнена, то в G_f есть простой пусть из u в s (поскольку в G есть путь из s в u, по которому идёт положительный поток).
- Высота не может убывать вдоль этого пути более чем на 1.

Оценка высоты

Лемма

Высота любой вершины в течение работы алгоритма не превосходит $2|V| - 1$.

Доказательство

- Если вершина u переполнена, то в G_f есть простой пусть из u в s (поскольку в G есть путь из s в u, по которому идёт положительный поток).
- Высота не может убывать вдоль этого пути более чем на 1.
- Следовательно, высота u не превосходит $2|V| - 1$. □

Оценка числа подъёмов

Лемма

Общее число операций подъёма не превосходит $2|V|^2$.

Оценка числа подъёмов

Лемма

Общее число операций подъёма не превосходит $2|V|^2$.

Доказательство

Каждую вершину поднять можно не более $2|V| - 1$ раз, всего же вершин, которые можно поднимать, $|V| - 2$. □

Оценка числа насыщающих проталкиваний

Лемма

Общее число насыщающих проталкиваний не превосходит $2|V||E|$.

Оценка числа насыщающих проталкиваний

Лемма

Общее число насыщающих проталкиваний не превосходит $2|V||E|$.

Доказательство

Оценка числа насыщающих проталкиваний

Лемма

Общее число насыщающих проталкиваний не превосходит $2|V||E|$.

Доказательство

- Рассмотрим насыщающее проталкивание между вершинами u и v.

Оценка числа насыщающих проталкиваний

Лемма

Общее число насыщающих проталкиваний не превосходит $2|V||E|$.

Доказательство

- Рассмотрим насыщающее проталкивание между вершинами u и v.
- Если произошло насыщающее проталкивание из u в v, то в этот момент $h(u) = h(v) + 1$.

Оценка числа насыщающих проталкиваний

Лемма

Общее число насыщающих проталкиваний не превосходит $2|V||E|$.

Доказательство

- Рассмотрим насыщающее проталкивание между вершинами u и v.
- Если произошло насыщающее проталкивание из u в v, то в этот момент $h(u) = h(v) + 1$.
- После такого проталкивания ребро (u, v) из остаточной сети пропадёт.

Оценка числа насыщающих проталкиваний

Лемма

Общее число насыщающих проталкиваний не превосходит $2|V||E|$.

Доказательство

- Рассмотрим насыщающее проталкивание между вершинами u и v.
- Если произошло насыщающее проталкивание из u в v, то в этот момент $h(u) = h(v) + 1$.
- После такого проталкивания ребро (u, v) из остаточной сети пропадёт.
- Чтобы такое ребро снова появилось, нужно протолкнуть поток из v в u, но для этого необходимо выполнение условия $h(v) = h(u) + 1$, то есть $h[v]$ нужно увеличить хотя бы на 2.

Оценка числа насыщающих проталкиваний

Лемма

Общее число насыщающих проталкиваний не превосходит $2|V||E|$.

Доказательство

- Рассмотрим насыщающее проталкивание между вершинами u и v.
- Если произошло насыщающее проталкивание из u в v, то в этот момент $h(u) = h(v) + 1$.
- После такого проталкивания ребро (u, v) из остаточной сети пропадёт.
- Чтобы такое ребро снова появилось, нужно протолкнуть поток из v в u, но для этого необходимо выполнение условия $h(v) = h(u) + 1$, то есть $h[v]$ нужно увеличить хотя бы на 2.
- Всего, таким образом, между двумя вершинами может быть не более $2|V|$ проталкиваний. □

Оценка числа ненасыщающих проталкиваний

Лемма

Общее число ненасыщающих проталкиваний не превосходит $4|V|^2(|V| + |E|)$.

Оценка числа ненасыщающих проталкиваний

Лемма

Общее число ненасыщающих проталкиваний не превосходит $4|V|^2(|V| + |E|)$.

Доказательство

Оценка числа ненасыщающих проталкиваний

Лемма

Общее число ненасыщающих проталкиваний не превосходит $4|V|^2(|V| + |E|)$.

Доказательство

- Назовём потенциалом сумму высот переполненных вершин.

Оценка числа ненасыщающих проталкиваний

Лемма

Общее число ненасыщающих проталкиваний не превосходит $4|V|^2(|V| + |E|)$.

Доказательство

- Назовём потенциалом сумму высот переполненных вершин.
- Подъём вершины увеличивает потенциал не более чем на $2|V|$.

Оценка числа ненасыщающих проталкиваний

Лемма

Общее число ненасыщающих проталкиваний не превосходит $4|V|^2(|V| + |E|)$.

Доказательство

- Назовём потенциалом сумму высот переполненных вершин.
- Подъём вершины увеличивает потенциал не более чем на $2|V|$.
- Насыщающее проталкивание также увеличивает не более чем на $2|V|$ (при появлении новой переполненной вершины).

Оценка числа ненасыщающих проталкиваний

Лемма

Общее число ненасыщающих проталкиваний не превосходит $4|V|^2(|V| + |E|)$.

Доказательство

- Назовём потенциалом сумму высот переполненных вершин.
- Подъём вершины увеличивает потенциал не более чем на $2|V|$.
- Насыщающее проталкивание также увеличивает не более чем на $2|V|$ (при появлении новой переполненной вершины).
- Ненасыщающее проталкиваие из u в v уменьшает потенциал хотя бы на 1, поскольку при таком проталкивании u перестаёт быть переполненной, а стать переполненной может только v, высота которой на 1 меньше.

Доказательство (завершение)

Доказательство

Доказательство (завершение)

Доказательство

- Суммарное увеличение потенциала не превосходит $(2|V|)(2|V|^2) + (2|V|)(2|V||E|) = 4|V|^2(|V| + |E|)$.

Доказательство (завершение)

Доказательство

- Суммарное увеличение потенциала не превосходит $(2|V|)(2|V|^2) + (2|V|)(2|V||E|) = 4|V|^2(|V| + |E|)$.
- Поскольку потенциал всегда не меньше нуля, его суммарное уменьшение не превосходит его суммарного увеличения. □

Время работы

Теорема

Алгоритм, основанный на проталкивании предпотока, можно реализовать так, чтобы время его работы было $O(|V|^2|E|)$.

План лекции

Общие идеи

Общие идеи

Общие идеи

Общие идеи

- Алгоритм “поднять-и-в-начало” основан на методе проталкивания предпотока и применяет операции подъёма и проталкивания в разумном порядке.

Общие идеи

Общие идеи

- Алгоритм “поднять-и-в-начало” основан на методе проталкивания предпотока и применяет операции подъёма и проталкивания в разумном порядке.
- Все вершины хранятся в списке.

Общие идеи

Общие идеи

- Алгоритм “поднять-и-в-начало” основан на методе проталкивания предпотока и применяет операции подъёма и проталкивания в разумном порядке.
- Все вершины хранятся в списке.
- Алгоритм просматривает список и находит переполненную вершину, после чего применяет к ней операции подъёма и проталкивания до тех пор, пока избыток в ней не обнулится.

Общие идеи

Общие идеи

- Алгоритм “поднять-и-в-начало” основан на методе проталкивания предпотока и применяет операции подъёма и проталкивания в разумном порядке.
- Все вершины хранятся в списке.
- Алгоритм просматривает список и находит переполненную вершину, после чего применяет к ней операции подъёма и проталкивания до тех пор, пока избыток в ней не обнулится.
- Если при этом вершина была поднята, вершина перемещается в начало списка.

Допустимые рёбра

Определение

Допустимые рёбра

Определение

- Ребро (u, v) называется допустимым (admissible), если оно входит в остаточную сеть и $h(u) = h(v) + 1$.

Допустимые рёбра

Определение

- Ребро (u, v) называется допустимым (admissible), если оно входит в остаточную сеть и $h(u) = h(v) + 1$.
- Оставшиеся рёбра называются недопустимыми (inadmissible).

Допустимые рёбра

Определение

- Ребро (u, v) называется допустимым (admissible), если оно входит в остаточную сеть и $h(u) = h(v) + 1$.
- Оставшиеся рёбра называются недопустимыми (inadmissible).
- Сеть $G_{f,h} = (V, E_{f,h})$, где $E_{f,h}$ — множество допустимых рёбер, назовём сетью допустимых рёбер (admissible network).

Разгрузка переполненной вершины

DISCHARGE(u)

```
1  while e[u] > 0
2        do v ← current[u]
3           if v = NIL
4              then LIFT(u)
5                   current[u] ← head[N[u]]
6           elseif c_f(u, v) > 0 и h[u] = h[v] + 1
7              then PUSH(u, v)
8           else current[u] ← next-neighbor[v]
```

Разгрузка переполненной вершины

DISCHARGE(u)

```
1  while e[u] > 0
2        do v ← current[u]
3           if v = NIL
4              then LIFT(u)
5                   current[u] ← head[N[u]]
6           elseif c_f(u, v) > 0 и h[u] = h[v] + 1
7              then PUSH(u, v)
8           else current[u] ← next-neighbor[v]
```

Замечания

Разгрузка переполненной вершины

DISCHARGE(u)

```
1  while e[u] > 0
2        do v ← current[u]
3           if v = NIL
4              then LIFT(u)
5                   current[u] ← head[N[u]]
6           elseif c_f(u, v) > 0 и h[u] = h[v] + 1
7              then PUSH(u, v)
8           else current[u] ← next-neighbor[v]
```

Замечания

- $N[u]$ — список соседей вершины u (то есть таких вершин v, для которых $(u, v) \in E$ или $(v, u) \in E$).

Разгрузка переполненной вершины

DISCHARGE(u)

```
1  while e[u] > 0
2        do v ← current[u]
3           if v = NIL
4              then LIFT(u)
5                   current[u] ← head[N[u]]
6           elseif c_f(u, v) > 0 и h[u] = h[v] + 1
7              then PUSH(u, v)
8           else current[u] ← next-neighbor[v]
```

Замечания

- $N[u]$ — список соседей вершины u (то есть таких вершин v, для которых $(u, v) \in E$ или $(v, u) \in E$).
- При вызове указатель $current[u]$ находится в позиции, унаследованной от предыдущего вызова.

Корректность процедуры

Корректность процедуры

Корректность процедуры

Корректность процедуры

- Ясно, что проталкивание осуществляется только тогда, когда оно возможно.

Корректность процедуры

Корректность процедуры

- Ясно, что проталкивание осуществляется только тогда, когда оно возможно.
- Покажем, что и подъём осуществляется корректно.

Корректность процедуры

Корректность процедуры

- Ясно, что проталкивание осуществляется только тогда, когда оно возможно.
- Покажем, что и подъём осуществляется корректно.
- Для этого достаточно показать, что в этот момент все выходящие из u остаточные рёбра недопустимы.

Корректность процедуры

Корректность процедуры

- Ясно, что проталкивание осуществляется только тогда, когда оно возможно.
- Покажем, что и подъём осуществляется корректно.
- Для этого достаточно показать, что в этот момент все выходящие из u остаточные рёбра недопустимы.
- Перед подъёмом вершины всё её соседи просматриваются, при этом могут производиться проталкивания.

Корректность процедуры

Корректность процедуры

- Ясно, что проталкивание осуществляется только тогда, когда оно возможно.
- Покажем, что и подъём осуществляется корректно.
- Для этого достаточно показать, что в этот момент все выходящие из u остаточные рёбра недопустимы.
- Перед подъёмом вершины всё её соседи просматриваются, при этом могут производиться проталкивания.
- Нетрудно видеть, что при этом никакое исходящее из u ребро допустимым стать не могло: проталкивания допустимых рёбер не создают, а подъёмы других вершин создают лишь выходящие из них допустимые рёбра.

Алоритм “поднять-и-в-начало”

LIFT-TO-FRONT(G, s, t)

```
 1  задать начальный предпоток
 2  L ← V[G] \ {s, t}
 3  for каждой вершины u ∈ V[G] \ {s, t}
 4        do current[u] ← head[N[u]]
 5  u ← head[L]
 6  while u ≠ NIL
 7        do old-height ← h[u]
 8           DISCHARGE(u)
 9           if h[u] > old-height
10              then переместить u в начало списка L
11           u ← next[u]
```

Корректность алгоритма

Корректность алгоритма

Корректность алгоритма

Корректность алгоритма

- Покажем, что в ходе работы алгоритма список L остаётся линейно упорядоченным относительно текущей допустимой сети.

Корректность алгоритма

Корректность алгоритма

- Покажем, что в ходе работы алгоритма список L остаётся линейно упорядоченным относительно текущей допустимой сети.
- Изначально это верно, поскольку допустимых рёбер нет вообще.

Корректность алгоритма

Корректность алгоритма

- Покажем, что в ходе работы алгоритма список L остаётся линейно упорядоченным относительно текущей допустимой сети.
- Изначально это верно, поскольку допустимых рёбер нет вообще.
- Проталкивания не создают новых допустимых рёбер.

Корректность алгоритма

Корректность алгоритма

- Покажем, что в ходе работы алгоритма список L остаётся линейно упорядоченным относительно текущей допустимой сети.
- Изначально это верно, поскольку допустимых рёбер нет вообще.
- Проталкивания не создают новых допустимых рёбер.
- При подъёме вершины все рёбра, входящие в неё, недопустимы.

Корректность алгоритма

Корректность алгоритма

- Покажем, что в ходе работы алгоритма список L остаётся линейно упорядоченным относительно текущей допустимой сети.
- Изначально это верно, поскольку допустимых рёбер нет вообще.
- Проталкивания не создают новых допустимых рёбер.
- При подъёме вершины все рёбра, входящие в неё, недопустимы.
- Поэтому после перемещения поднятой вершины в начало инвариант сохраняется.

Корректность алгоритма

Корректность алгоритма

- Покажем, что в ходе работы алгоритма список L остаётся линейно упорядоченным относительно текущей допустимой сети.
- Изначально это верно, поскольку допустимых рёбер нет вообще.
- Проталкивания не создают новых допустимых рёбер.
- При подъёме вершины все рёбра, входящие в неё, недопустимы.
- Поэтому после перемещения поднятой вершины в начало инвариант сохраняется.
- Таким образом, алгоритм заканчивает свою работу, разгрузив каждую вершину списка.

Корректность алгоритма

Корректность алгоритма

- Покажем, что в ходе работы алгоритма список L остаётся линейно упорядоченным относительно текущей допустимой сети.
- Изначально это верно, поскольку допустимых рёбер нет вообще.
- Проталкивания не создают новых допустимых рёбер.
- При подъёме вершины все рёбра, входящие в неё, недопустимы.
- Поэтому после перемещения поднятой вершины в начало инвариант сохраняется.
- Таким образом, алгоритм заканчивает свою работу, разгрузив каждую вершину списка.
- Причём при проталкивании потока из u избыток может появиться только в вершинах, идущих в списке после u.

Корректность алгоритма

Корректность алгоритма

- Покажем, что в ходе работы алгоритма список L остаётся линейно упорядоченным относительно текущей допустимой сети.
- Изначально это верно, поскольку допустимых рёбер нет вообще.
- Проталкивания не создают новых допустимых рёбер.
- При подъёме вершины все рёбра, входящие в неё, недопустимы.
- Поэтому после перемещения поднятой вершины в начало инвариант сохраняется.
- Таким образом, алгоритм заканчивает свою работу, разгрузив каждую вершину списка.
- Причём при проталкивании потока из u избыток может появиться только в вершинах, идущих в списке после u.
- В конце работы алгоритма ни проталкивание, ни подъём не применимы.

Анализ времени работы

Теорема

Время работы алгоритма LIFT-TO-FRONT есть $O(|V|^3)$.

Анализ времени работы

Теорема

Время работы алгоритма LIFT-TO-FRONT есть $O(|V|^3)$.

Доказательство

Анализ времени работы

Теорема

Время работы алгоритма LIFT-TO-FRONT есть $O(|V|^3)$.

Доказательство

- Как мы уже знаем, общее количество подъёмов есть $O(|V|^2)$ (поскольку высота каждой вершины ограничена числом $2|V|$).

Анализ времени работы

Теорема

Время работы алгоритма LIFT-TO-FRONT есть $O(|V|^3)$.

Доказательство

- Как мы уже знаем, общее количество подъёмов есть $O(|V|^2)$ (поскольку высота каждой вершины ограничена числом $2|V|$).
- Между двумя последовательными подъёмами происходит не более $|V|$ вызовов процедуры разгрузки.

Анализ времени работы

Теорема

Время работы алгоритма LIFT-TO-FRONT есть $O(|V|^3)$.

Доказательство

- Как мы уже знаем, общее количество подъёмов есть $O(|V|^2)$ (поскольку высота каждой вершины ограничена числом $2|V|$).
- Между двумя последовательными подъёмами происходит не более $|V|$ вызовов процедуры разгрузки.
- При каждом проходе цикла процедуры разгрузки происходит либо подъём, либо проталкивание, либо перемещение указателя.

Анализ времени работы

Теорема

Время работы алгоритма LIFT-TO-FRONT есть $O(|V|^3)$.

Доказательство

- Как мы уже знаем, общее количество подъёмов есть $O(|V|^2)$ (поскольку высота каждой вершины ограничена числом $2|V|$).
- Между двумя последовательными подъёмами происходит не более $|V|$ вызовов процедуры разгрузки.
- При каждом проходе цикла процедуры разгрузки происходит либо подъём, либо проталкивание, либо перемещение указателя.
- Количество подъёмов есть $O(|V|^2)$, на все же подъемы требуется времени $O(|V||E|)$ (подъем одной вершины осуществляется за время, пропорциональное её степени).

Доказательство (завершение)

Доказательство

Доказательство (завершение)

Доказательство

- По тем же причинам общее перемещение указателей есть $O(|V||E|)$.

Доказательство (завершение)

Доказательство

- По тем же причинам общее перемещение указателей есть $O(|V||E|)$.
- Количество насыщающих проталкиваний есть $O(|V||E|)$.

Доказательство (завершение)

Доказательство

- По тем же причинам общее перемещение указателей есть $O(|V||E|)$.
- Количество насыщающих проталкиваний есть $O(|V||E|)$.
- За один вызов процедуры разгрузки производится не более одного ненасыщающего проталкивания, поэтому общее количество ненасыщающих проталкиваний есть $O(|V|^3)$. □

Что мы узнали за сегодня?

Что мы узнали за сегодня?

алгоритм	время работы
Эдмондса-Карпа	$\|V\|\|E\|^2$
проталкивание предпотока	$\|V\|^2\|E\|$
“поднять-и-в-начало”	$\|V\|^3$

Спасибо за внимание!