Лекция #9: Потоки

9 ноября

9.1. Основные определения

Дан орграф G, у каждого ребра e есть пропускная способность $c_e \in \mathbb{R}$.

Def 9.1.1. *Поток в орграфе из s в t – сопоставленные рёбрам числа $f_e \in \mathbb{R}$:*

$$(\forall \text{ ребра } e \;\; 0 \leqslant f_e \leqslant c_e) \;\wedge\; (\forall \text{ вершины } v \neq s,t \quad \sum_{e \in in(v)} f_e = \sum_{e \in out(v)} f_e)$$

Вершина s называется *истоком*, вершина t *стоком*.
Говорят, что по ребру e течёт f_e единиц потока.
Определение говорит "поток течёт из истока в сток и ни в какой вершине не задерживается".

Def 9.1.2. *Величина потока* $|f| = \sum_{e \in out(s)} f_e - \sum_{e \in in(s)} f_e$ *(сколько вытекает из истока).*

Утверждение 9.1.3. В сток втекает ровно столько, сколько вытекает из истока.

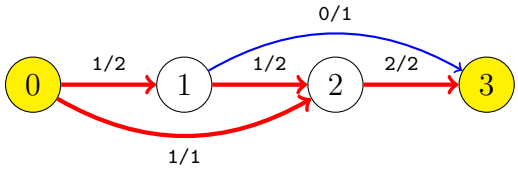
Замечание 9.1.4. $|f|$ может быть отрицательной: пустим по ребру $t \to s$ единицу потока.

Def 9.1.5. *Циркуляцией называется поток величины* 0.

• Примеры потока

Рассмотрим пока граф с единичными пропускными способностями.

1. $\forall$ цикл – циркуляция.
2. $\forall$ путь из s в t – поток величины 1.
3. $\forall$ k не пересекающихся по рёбрам путей из s в t – поток величины k.
4. На картинке поток величины 2, подписи f_e/c_e.



Def 9.1.6. *Остаточная сеть потока f – G_f, граф с пропускными способностями $c_e - f_e$.*

Def 9.1.7. *Дополняющий путь – путь из s в t в остаточной сети G_f.*

Lm 9.1.8. Если по всем рёбрам дополняющего пути p увеличить величину потока на $x = \min_{e \in p}(c_e - f_e)$, получится корректный поток величины $|f| + x$.

9.2. Обратные рёбра

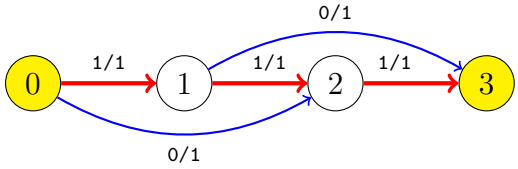
Def 9.2.1. *Для каждого ребра сети G с пропускной способностью c_e создадим обратное ребро e' пропускной способностью* 0. *При этом по определению $f_{e'} = -f_e$.*

Добавим в граф обратные рёбра, упростим определения потока и величины потока:
Теперь $\forall$ потока f должно выполняться $\forall v \neq s,t \sum_{e \in out(v)} f_e = 0$, величина потока $|f| = \sum_{e \in out(s)} f_e$.
Здесь $out(v)$ – множество прямых и обратных рёбер, выходящих из v.

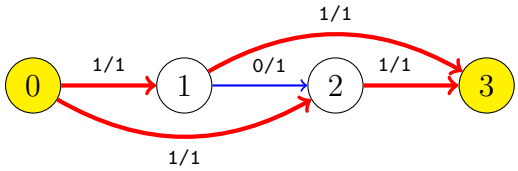
Def 9.2.2. *Ребро называется насыщенным, если $f_e = c_e$, иначе оно ненасыщено.*

Утверждение 9.2.3. Если по прямому ребру течёт поток, обратное ненасыщено.

После добавления обратных рёбер в в G, они появились и в G_f. Поэтому для такого потока из 0 в 3 величины 1 в G_f есть дополняющий путь $0 \to 2 \to 1 \to 3$.



Увеличим поток по пути $0 \to 2 \to 1 \to 3$, получим новый поток. Заметьте, добавляя +1 потока к ребру $2 \to 1$, мы уменьшаем поток по ребру $1 \to 2$.



Замечание 9.2.4. Сейчас мы научимся разбивать поток величины 2 на 2 непересекающихся по рёбрам пути. Если бы мы действовали жадно (*найдём какой-нибудь первый путь, удалим его рёбра из графа, на оставшихся рёбрах найдём второй путь*), нас постигла бы неудача. Потоку же благодаря обратным рёбрам получается даже при неверном первом пути найти дополняющий путь и получить поток размера два.

9.3. Альтернативное определение

Нередко удобнее работать с альтернативными определениями:
Транспортная сеть. У нас есть:

- граф $G(V, E)$ (ориентированный или нет);
- исток $s \in V$ и сток $t \in V$ $(s \neq t)$;
- пропускные способности $c\colon V^2 \to \mathbb{R}_+$;
- поток $f\colon V^2 \to \mathbb{R}$.

И должны быть выполнены следующие свойства:

1. $\forall v, u \in V \;\; f(v,u) = -f(u,v)$;
2. $\forall v, u \in V \;\; f(v,u) \leqslant c(v,u)$;
3. $\forall v \in V \setminus \{s,t\} \;\; \sum_{u \in V} f(v,u) = 0$;
4. если $e = (vu) \in E$, то $c(v,u) = c_e \geqslant 0$; иначе $c(v,u) = 0$.

Величина потока (обозначаем $|f|$) — это число $\sum_{u \in V} f(s,u)$.

9.4. Декомпозиция потока (на практике)

Def 9.4.1. *Элементарный поток – путь из s в t, по которому течёт x единиц потока.*

Def 9.4.2. *Декомпозиция потока f – представление f в виде суммы элементарных потоков (путей) и циркуляции.*

Lm 9.4.3. $|f| > 0 \Rightarrow \exists$ путь из s в t по рёбрам $e\colon f_e > 0$.

• **Алгоритм декомпозиции за $\mathcal{O}(E^2)$**
Пока $|f| > 0$ найдём путь p из s в f по рёбрам $e\colon f_e > 0$,
по всем рёбрам пути p уменьшим поток на $x = \min_{e \in p} f_e$.

Lm 9.4.4. Время работы $\mathcal{O}(E^2)$

Доказательство. По рёбрам $e\colon f_e > 0$ поток только убывает. После отщепления одного пути, как минимум у одного из рёбер f_e обнулится $\Rightarrow$ не более E поисков пути. ■

9.5. Теорема и алгоритм Форда-Фалкерсона

Def 9.5.1. *Для любых множеств $S, T \subseteq V$ определим*

$$F(S,T) = \sum_{a\in S, b\in T} f_{a\to b},\ C(S,T) = \sum_{a\in S, b\in T} c_{a\to b}$$

Сумма включает обратные рёбра $\Rightarrow$ на графе из одного ребра $e\colon t\to s$, $f_e = 1$ $\quad F(\{s\},\{t\}) = -1$.

Lm 9.5.2. $\forall v \in V \begin{cases} F(\{v\},V) = 0 & v \neq s,t \\ F(\{v\},V) = |f| & v = s \end{cases}$

Lm 9.5.3. $\forall S \quad F(S,S) = 0$

Доказательство. В вместе с каждым ребром в сумму войдёт и обратное ему. ■

Lm 9.5.4. $\forall S,T \quad F(S,T) \leqslant C(S,T)$

Доказательство. Сложили неравенства $f_e \leqslant c_e$ по всем рёбрам $e\colon S\to T$. ■

Def 9.5.5. *Разрез – дизъюнктное разбиение вершин $(S,T) : V = S \sqcup T, s\in S, t\in T$.*

Def 9.5.6. *Величина разреза* $(S,T) = C(S,T)$.

Lm 9.5.7. $\forall$ разреза $(S,T) \quad |f| = F(S,T)$

Доказательство. Интуитивно: поток вытекает из s, нигде не задерживается $\Rightarrow$ он весь протечёт через разрез. Строго: $F(S,T) = F(S,T) + F(S,S) = F(S,V) = F(\{s\},V) + 0 + \cdots + 0 = |f|$. ■

Lm 9.5.8. $\forall$ разреза (S,T) и потока $f \quad |f| \leqslant C(S,T)$

Доказательство. $|f| = F(S,T) \leqslant C(S,T)$ (пользуемся леммами 9.5.4 и 9.5.7). ■

Теорема 9.5.9. Форда-Фалекрсона

(1) $|f| = \max \Leftrightarrow \nexists$ дополняющий путь
(2) $\max |f| = \min C(S,T)$ (максимальный поток равен минимальному разрезу)

Доказательство. $\exists$ дополняющий путь $\Rightarrow$ можно увеличить по нему $f \Rightarrow |f| \neq \max$.
Пусть нет дополняющего пути $\Rightarrow$ dfs из s по ненасыщенным рёбрам не посетит t. Множество посещённых вершин обозначим S, обозначим $T = V \setminus S$. Из S в T ведут только $e\colon f_e = c_e$.
Значит, $|f| = F(S,T) = C(S,T)$. Из леммы 9.5.8 следует, что $|f| = \max, C(S,T) = \min$. ■

• Поиск минимального разреза

Из доказательства теоремы 9.5.9 мы заодно получили алгоритм за $\mathcal{O}(E)$ поиска min разреза по max потоку.

• Алгоритм Форда-Фалкерсона

> Из теоремы следует простейший алгоритм поиска максимального потока: пока есть дополняющий путь p, найдём его, толкнём по нему $x = \min_{e\in p}(c_e - f_e)$ единиц потока.

Утверждение 9.5.10. Если все $c_e \in \mathbb{Z}$, алгоритм конечен.

Время работы алгоритма мы умеем оценивать сверху только как $\mathcal{O}(|f| \cdot E)$.
При $c_e \leqslant polynom(|V|, |E|)$ получаем $|f| \leqslant polynom(|V|, |E|) \Rightarrow$ Ф.Ф. работает за полином.
При экспоненциально больших c_e на практике мы построим тест: время работы $\Omega(2^{V/2})$.

9.6. Реализация, хранение графа

Первый способ хранения графа более естественный:

```
struct Edge {
  int a, b, f, c, rev; // a ---> b
};
vector<Edge> c[n]; // c[c[v][i].b][c[v][i].rev] - обратное ребро
for (Edge e : c[v]) // перебор рёбер, смежных с v
  ;
```

Второй часто работает быстрее, и позволяет проще обращаться к обратному ребру.
Поэтому про него поговорим подробнее.

```
struct Edge {
  int a, b, f, c; // собственно ребро
  int next; // интрузивный список, список на массиве
};
vector<Edge> edges;
vector<int> head(n, -1); // для каждой вершин храним начало списка
for (int i = head[v]; i != -1; i = edges[i].next)
  Edge e = edges[i]; // перебор рёбер, смежных с v
```

Добавить орребро можно так:

```
void add(a, b, c):
  edges.push_back({a, b, 0, c}); // прямое
  edges.push_back({b, a, 0, 0}); // обратное
```

Заметим, что взаимообратные рёбра добавляются парами $\Rightarrow$
$\forall i$ к `edges[i]` обратным является `edges[i^1]`.

Теперь реализуем алгоритм Ф.Ф. Также, как и Кун, dfs, ищущий путь, сразу на обратном ходу рекурсии будет изменять поток по пути.

```
bool dfs(int v):
  u[v] = 1;
  for (int i = head[v]; i != -1; i = edges[i].next):
    Edge &e = edges[i];
    if (e.f < e.c && !u[e.b] && (e.b == t || dfs(e.b))):
      e.f++, edges[i ^ 1].f--; // не забудьте пересчитать обратное ребро
      return 1;
  return 0;
```

По сути мы лишь нашли путь из s в t в остаточной сети G_f.

Если мы хотим толкать не единицу потока, а $\min_e(c_e - f_e)$, нужно, чтобы dfs на прямом ходу рекурсии насчитывал минимум и возвращал из рекурсии полученное значение.