

*Федеральное государственное автономное учреждение
высшего профессионального образования*

**Московский Физико-Технический Институт
КЛУБ ТЕХА ЛЕКЦИЙ**

**А л г о р и т м ы .
И В Т .**

III СЕМЕСТР

Лекторы: А. И. Гришутин. И. Д. Степанов.



Автор: А. Ш. Ильдаров.

Проект на overleaf

Проект на github

Осень 2020 года

Содержание

1 Паросочетания. Минимальное вершинное покрытие.	2
2 Потоки	6

1 Паросочетания. Минимальное вершинное покрытие.

Определение 1. Двудольный граф – граф, множество вершин которого можно разбить на две части таким образом, что каждое ребро графа соединяет какую-то вершину из одной части с какой-то вершиной другой части, то есть не существует рёбер между вершинами одной и той же части.

Определение 2. Хроматическое число – минимальное число цветов, в которые можно раскрасить вершины графа так, чтобы концы любого ребра имели разные цвета.

Определение 3. Паросочетание (англ. matching) в двудольном графе — произвольное множество рёбер двудольного графа, такое что никакие два ребра не имеют общей вершины.

Мощность паросочетания – количество ребер в нем.

Максимальное паросочетание – мощность которого *наибольшая* среди всех возможных паросочетаний в данном графе.

Определение 4. Цепь – некоторый простой путь (т.е. не содержащий повторяющихся вершин или рёбер).

Чредующаяся цепь – цепь, в которой рёбра поочередно принадлежат/не принадлежат паросочетанию.

Увеличивающая цепь – редующаяся цепь, в которой начальное и конечное ребра *НЕ* принадлежат паросочетанию.

Уменьшающая цепь – цепь, в которой начальное и конечное ребра принадлежат паросочетанию.

Определение 5. Вершины двудольного графа, инцидентные рёбрам паросочетания M , называются **насыщенными** или **покрытыми**.

Алгоритм Куна.

Теорема 1.1. Паросочетание M в двудольном графе G – *тах* \iff в G нет увеличивающей цепи относительно M .

Доказательство.

\Rightarrow :

От противного: Пусть в G с максимальным паросочетанием M существует увеличивающая цепь.

Тогда заменив в ней все рёбра, входящие в паросочетание, на невходящие и наоборот, мы получим большее паросочетание.

То есть M не являлось максимальным. Противоречие.

\Leftarrow :

Пусть M – не max, покажем что \exists увеличивающая цепь.

Пусть M' – паросочетание: $|M'| > |M|$

Построим подграф $G' = M \oplus M'$, состоящий из ребер \in только одному из паросочетаний.

M и M' – паросочетания \Rightarrow нет вершин, которые смежны с двумя ребрами из паросочетания. То есть у каждой вершины подграфа есть не более одного ребра из M и не более одного из M' . $\Rightarrow \forall v \in G' \hookrightarrow \deg(v) \leq 2$

Как известно, графы с таким свойством степеней вершин представляют из себя наборы цепей и циклов.

При этом длина цикла должна быть четной, ведь иначе мы будем иметь вершину, у которой два ребра, к ней смежных, принадлежат одному паросочетанию.

В циклах поровну ребер из каждого паросочетания, значит их вклад в отрыв M' от M по числу ребер – нулевой.

Значит обогнать M у M' получится только если в графе имеется цепочка нечетной длины, у которой начальное и конечное ребра лежат в M' . Вот эта вот цепочка и есть увеличивающая. \square

Алгоритм Куна для поиска максимального паросочетания.

1. Фиксируем доли графа: L и R . Изначально считаем паросочетание пустым.
2. В доле L перебираем вершины в порядке увеличения номеров.
3. Если вершина насыщена, то пропускаем ее и идем дальше. В противном случае, пытаемся насытить вершину, запустив поиск увеличивающей цепи из этой вершины следующим образом:
 - 3.1. Стоя в текущей вершине v доли L , посмотрим все ребра из этой вершины.
 - 3.2. Возьмем текущее ребро (v, t_0) : Если t_0 – ненасыщена, то одно ребро (v, t_0) и задает нам увеличивающую цепь, просто увеличим паросочетание с его помощью и прекратим поиск. Если же t_0 насыщена каким-то ребром (t_0, p) , то пойдем вдоль этого ребра, уже в поисках увеличивающейся цепи, проходящей через ребра (v, t_0) и (t_0, p) . Для этого просто перейдем в вершину p и продолжим обход из нее.

4. В конечном итоге, мы либо найдем увеличивающую цепь из вершины v и увеличим паросочетание этой цепью, тем самым насытив вершину, либо же покажем отсутствие увеличивающей цепи и невозможность насыщения вершины.
5. Возьмем следующую по порядку вершину доли L и повторим.
6. После того как мы просмотрим все вершины, попытаемся увеличить паросочетание цепью из них, мы получим максимальное паросочетание.

Минимальное вершинное покрытие.

Определение 6. Вершинное покрытие графа $G=(V,E)$ - такое подмножество S множества вершин графа V , что любое ребро этого графа инцидентно хотя бы одной вершине из множества S .

Определение 7. Минимальное вершинное покрытие графа – вершинное покрытие, состоящие из *наименьшего* числа вершин.

Утверждение 1.2. $|M_{max}| \leq |C_{min}|$. Чтобы покрыть все ребра, нам нужно вершин не меньше, чем мощность наибольшего паросочетания.

Доказательство.

Паросочетание это, как известно, набор непересекающихся рёбер. Мы хотим взять набор вершин, эти ребра покрывающий. Ясно что каждая вершина может покрыть не более одного ребра, ведь они не пересекаются по концам. Оценка снизу доказана. \square

Теорема 1.3. (Кёнига)

В двудольном графе $|M_{max}| = |C_{min}|$.

Доказательство. Явно предъявим покрытие, размер которого равен размеру максимального паросочетания.

1. Разделим граф на доли L и R .
2. Сориентируем ребра: Из $M_{max} \leftarrow$, остальные \rightarrow
3. Обойдем граф из *ненасыщенных* вершин $\in L$.
4. Получим разбиение графа на четыре множества: L^+, R^+ – вершины, лежащие в L или R соответственно, которые доступны из ненасыщенных вершин, лежащих в L . L^-, R^- – аналогично *недоступные* из ненасыщенных вершин, лежащих в L .

5. Поймем какие ребра бывают между каждыми из четырех множеств:
 - 5.1. Покажем, что ребер $L^+ \rightarrow R^-$ не бывает, ведь если бы такое ребро имелось, мы из достижимой вершины, лежащей в L^+ добрались бы по этому ребру до вершины, по определению R^- , недостижимой.
 - 5.2. Из аналогичных рассуждений понятно что не бывает ребер $L^- \rightarrow R^+$. Это не отрицает существование ребер $(R^+ \rightarrow L^-)$
 - 5.3. Покажем отсутствие ребер $R^- \rightarrow L^+$: От противного: Пусть (r^-, l^+) – такое ребро. Это ребро вида \leftarrow , то есть принадлежащее паросочетанию. Вершина l^+ – насыщена, значит чтобы до нее дойти мы должны были начать обход из какой-то другой вершины l' , из которой l^+ достижима. Так как l' и l^+ лежат в одной доле двудольного графа, маршрут из l' в l^+ в какой-то момент должен пойти справа налево, то есть имеется ребро вида (r', l^+) , которое в силу своего направления также лежит в паросочетании. Значит из смежные ребра (r^-, l^+) и (r', l^+) лежат в одном паросочетании. Противоречие.
6. Получаем что любое ребро графа инцидентно или вершине из L^- или вершине из $R^+ \Rightarrow L^- \cup R^+$ – вершинное покрытие.
7. Покажем, что все вершины из $L^- \cup R^+$ насыщены ребрами из паросочетания: Это так, ведь если бы в L^- были ненасыщенные вершины, мы бы запускали из них обход и попадали бы в L^+ , чего быть не может в силу отсутствия ребер $L^- \rightarrow R^+$ и если бы в R^+ была бы ненасыщенная вершина, то существовала бы увеличивающаяся цепь, в этой вершине заканчивающаяся, что противоречит максимальности паросочетания.
8. Как известно, не существует ребер $L^- \rightarrow R^+$, которые бы принадлежали паросочетанию, а значит каждому ребру паросочетания инцидентна ровно одна вершина из покрытия $L^- \cup R^+$, а значит $|M| = |L^- \cup R^+|$, и это вершинное покрытие является наименьшим из оценки.

□

2 Потоки

Определение 8. Сеть – ориентированный граф $G = (V, E)$, с двумя выделенными различными вершинами: s – исток и t – сток, а также с функцией $cap : E \rightarrow \mathbb{Z}_{>0}$.

Определение 9. Поток в сети – функция $f : V \times V \rightarrow \mathbb{Z}$:

1. $\forall u, v \in V \hookrightarrow f(u, v) \leq cap(u, v)$
2. $\forall v \in V \setminus \{s, t\} \hookrightarrow \sum_{\substack{u \\ (u,v) \in E}} f(u, v) = \sum_{\substack{w \\ (v,w) \in E}} f(v, w)$
3. $\forall u, v \in V \hookrightarrow f(u, v) = -f(v, u)$

Замечание. Свойства 2 и 3 можно объединить в следующем виде $\forall v \in V \hookrightarrow \sum_{u \in V} f(v, u) = 0$

Определение 10. Остаточная сеть G_f сети G с потоком f – сеть, у которой модифицируются пропускные способности: $cap_f(u, v) = cap(u, v) - f(u, v)$, и удаляются ребра с нулевой cap_f .

Определение 11. Величина потока $|f|$ – то сколько "вытекает" из s – сумма исходящих из s потоков.

Определение 12. Разрез сети G – пара подсетей (S, T) : $s \in S, t \in T, S \cap T = \emptyset, S \cup T = G$

$$cap(S, T) = \sum_{\substack{u \in S \\ v \in T}} cap(u, v)$$

$$f(S, T) = \sum_{\substack{u \in S \\ v \in T}} f(u, v)$$

Лемма 2.1. (S, T) – разрез $\Rightarrow |f| = f(S, T)$

Доказательство. На семинаре. □

Лемма 2.2. (S, T) – разрез $\Rightarrow f(S, T) \leq cap(S, T)$

Доказательство. $\forall u, v \in V \hookrightarrow f(u, v) \leq cap(u, v) \Rightarrow$ чтд. □

Теорема 2.1. (Форда – Фалкерсона) Следующие утверждения эквивалентны:

1. f – max поток в G
2. в G_f нет пути из s в t
3. $|f| = cap$ некоторого разреза.

Доказательство.

$1 \Rightarrow 2$: От противного:

Пусть f – max, но в G_f есть путь $s \rightarrow t$.

Рассмотрим $x = \min(\text{cap}_f) > 0$ на этом пути.

Значит вдоль пути можно пустить x единиц потока \Rightarrow в исходной сети можно было пустить на x единиц больше Rightarrow f – не max.

$2 \Rightarrow 3$:

Пусть S – множество вершин, доступных из s в G_f , $t \notin S$

$T = V \setminus S$.

По определению (S, T) – разрез.

Покажем что $|f| = \text{cap}(S, T) = \sum_{\substack{u \in S \\ v \in T}} \text{cap}(u, v)$.

Рассмотрим ребро (u, v) : $u \in S, v \in T$

u – доступно в G_f , v – нет Rightarrow оно удаляется в остаточной сети Rightarrow

$f(u, v) = \text{cap}(u, v) \text{Rightarrow} \sum_{\substack{u \in S \\ v \in V}} \text{cap}(u, v) = \sum_{\substack{u \in S \\ v \in V}} f(u, v) \Rightarrow$

$f(S, T) = |f| = \text{cap}(S, T)$.

$3 \Rightarrow 1$:

$\exists(S, T) : |f| = \text{cap}(S, T)$.

При этом $\forall(S, T)$ – разрез $\hookrightarrow f(S, T) \leq \text{cap}(S, T) \Rightarrow f$ – max. □

Алгоритм Форда – Фалкерсона для поиска максимального потока.

1. Изначально поток равен 0.

2. Пока в G_f есть путь из s в t :

2.1. $x = \min(\text{cap}_f \text{ на этом пути }) > 0$

2.2. Увеличим поток f на x и изменим остаточную сеть соответствующим образом.

$|f|$ увеличивается на целое положительное число и ограничен сверху, значит алгоритм закончится. Асимптотика – $O(\text{ans} * |E|)$

Алгоритм Эдмондса – Карпа для поиска максимального потока.

Та же самая идея, только теперь вместо произвольного пути будем выбирать кратчайший путь по ребрам. По сути, DFS меняется на BFS.

1. Изначально поток равен 0.

2. Пока в G_f есть *минимальный* путь из s в t :

2.1. $x = \min(\text{cap}_f \text{ на этом пути }) > 0$.

2.2. увеличим поток f на x и изменим остаточную сеть соответствующим образом.

Алгоритм будет иметь асимптотику $O(|V| \times |E|^2)$

Докажем это.

Определение 13. $\text{dist}(u, v)$ – минимальное число ребер между вершинами.

Лемма 2.3. Пусть поток f' получается из потока f после одной итерации алгоритма Эдмондса – Карна.

Пусть $\text{dist}'(u, v)$ – кратчайшее расстояние между вершинами в $G_{f'}$, тогда $\forall v \in V \setminus \{s, t\} \hookrightarrow \text{dist}'(s, v) \geq \text{dist}(s, v)$.

То есть с каждой итерацией расстояние не убывает.

Доказательство. От противного:

Пусть $\exists v \in V \setminus \{s, t\} : \text{dist}'(s, v) < \text{dist}(s, v)$ и при этом $\text{dist}(s, v)$ – минимальное возможное

Пусть u – предыдущая вершина перед v на кратчайшем пути из s в v в $G_{f'}$

Тогда $\text{dist}'(s, u) = \text{dist}'(s, v) - 1$ (по определению dist).

Также $\text{dist}'(s, u) \geq \text{dist}(s, u)$ ведь мы выбрали минимально удаленное v .

Рассмотрим 2 случая:

$(u, v) \in E_f$, тогда:

$$\text{dist}(s, v) \leq \text{dist}(s, u) + 1 \leq \text{dist}'(s, u) + 1 \leq \text{dist}'(s, v)$$

Но при этом $\text{dist}'(s, v) < \text{dist}(s, v)$

Противоречие.

$(u, v) \notin E_f$, но известно что $(u, v) \in E_{f'}$.

Появление ребра (u, v) после увеличения потока означает увеличение потока по обратному ребру (v, u) . Увеличение потока производится вдоль кратчайшего пути, а значит вершина v – предыдущая перед u на пути из s в u

Это значит что $\text{dist}(s, v) = \text{dist}(s, u) - 1 \leq \text{dist}'(s, u) - 1 = \text{dist}'(s, v) - 2$, но ведь $\text{dist}'(s, v) < \text{dist}(s, v)$. Противоречие.

□

Определение 14. **Насыщенное** – ребро, вдоль которого идет поток, равный его capacity.

Лемма 2.4. Любое ребро сети насыщается не более $O(|V|)$ раз, то есть в алгоритме $O(|V||E|)$ итераций, то есть каждая итерация насыщает хотя бы 1 ребро.

Доказательство.

Рассмотрим ребро (u, v) в момент его насыщения.

Если оно насыщается, то оно лежит на кратчайшем пути от s до v (так работает наш алгоритм).

Значит $dist(s, u) + 1 = dist(s, v)$.

Теперь посмотрим, когда оно могло насытиться еще один раз:

Оно сначала должно перестать быть насыщенным, для этого нужно отменить поток вдоль (u, v) , то есть пустить поток вдоль (v, u)

Пусть $dist'$ – расстояние в момент, когда пускается поток вдоль (v, u) .

$dist'(s, v) + 1 = dist'(s, u)$ (Так как проталкивание происходит вдоль кратчайшего пути).

Со временем расстояния только увеличиваются, значит $dist'(s, v) \geq dist(s, v) + 1 = dist(s, u) + 2$

Таким образом от момента насыщения, до момента повторного насыщения расстояние $dist(s, u)$ должно вырасти по меньшей мере на 2.

Однако все расстояния ограничены $O(|V|)$, значит и насыщений может быть только $O(|V|)$. □