

Московский физико-технический институт
Физтех-школа прикладной математики и информатики

ОСНОВЫ КОМБИНАТОРИКИ И ТЕОРИИ ЧИСЕЛ
II СЕМЕСТР

Лектор: *Андрей Михайлович Райгородский*



Автор: *Киселев Николай*
Репозиторий на Github

весна 2025

Содержание

1	Минимальные остовные деревья	2
1.1	Алгоритм Прима	2
1.2	Алгоритм Крускала	3
1.2.1	Система непересекающихся множеств (СНМ)	3
1.3	Алгоритм Борувки	4
1.4	Тезис Черча-Тьюринга	5
1.5	Проблема самоприменимости	5
1.6	Проблема остановки	6
1.7	m -сводимость	6
1.7.1	Свойства m -сводимости	6
1.8	Как получать новые неперечислимые множества?	7
1.9	Существует ли универсальная тотально вычислимая функция?	7

«««< HEAD

1 Минимальные остовные деревья

Определение 1.1. Пусть $G = (V, E)$ - неориентированный граф. Тогда остовным подграфом G называется такой граф $H = (V', E')$, что $V' = V, E' \subset E$.

Определение 1.2. Остовным деревом графа G (MST) называется любой его подграф, являющийся деревом.

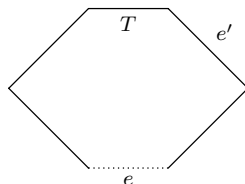
Лемма 1.1. (о безопасном ребре) Пусть S - множество ребер, такие что \exists MST T , содержащее все ребра S . Пусть C - одна из компонент связности относительно S (V, S), e - самое дешевое ребро между C и $V \setminus C$. Тогда $S + e$ - тоже подмножество некоторого MST.

Доказательство. $S + e \subset T$

- а) Если да, то доказывать нечего
- б) Если нет, то рассмотрим путь между u и v в дереве T . Пусть e' - ребро в нем, выходящее из C .

$$\omega(e) \leq \omega(e')$$

Заметим, что если удалить ребро e' и добавить e , то получим остовное дерево с лучшим ребром. Противоречие



□

1.1 Алгоритм Прима

1. Заводим текущее остовное дерево $S = \emptyset$
2. $S_1 = \{e_1\}$
3. $S_2 = \{e_2\} \vee S_1$
4. Пусть S_i - текущее множество ребер. На каждом шаге будем рассматривать самое дешевое ребро между C_i и $V \setminus C_i$ и добавлять его, как в примерах выше.

$$S_{i+1} = S_i + e_{i+1}, C_{i+1} = C_i \cup V_{i+1}$$

Асимптотика: $O(n^2 + m) \sim O(n^2)$

Пусть $d[v]$ - наименьшая стоимость ребра между C_i и v . На каждом шаге в C_i добавляются вершины v с минимальным значением $d[v]$. После выбора v :

```
for (u: g[v]) {
    d[u] = min(d[u], cost(v, u));
}
```

Замечание. Как и в алгоритме Дейкстры, если завести структуру, способную извлекать минимальное значение и делать `decreaseKey`, то можно обойтись асимптотикой $O(m + n \log n)$ с кучей Фибоначчи и $O(m \log n)$ с биномиальной кучей.

1.2 Алгоритм Крускала

1. Сортируем все ребра в порядке возрастания веса.
2. Идем по m ребрам в этом порядке, каждое ребро добавляем в ответ, если оно не образует цикла с другими.

Корректность следует из лемма о безопасном ребре

Как проверить наличие цикла?

1.2.1 Система непересекающихся множеств (СНМ)

⇒ **unite**: объединение двух непересекающихся множеств.

⇒ **same?** : проверить лежат ли два заданных элемента в одном множестве.

Через такую структуру достаточно хранить компоненты связности в качестве рассматриваемых множеств и добавлять ребро, только если оно соединяет вершины разных множеств.

Каждое множество в СНМ храним в виде корневого дерева (дерева с отмеченной вершиной - корнем). Тогда для каждой вершины достаточно дойти до корня дерева и сравнить эти вершины.

```
int get(v) {
    if (p[v] == v) {
        return v;
    }

    return get(p(v));
}
```

Чтобы провести *unite* достаточно подвесить корень одного дерева к корню другого.

Важный нюанс: Подвешиваем всегда меньшее по размеру дерево к большему.

Пусть $s[r]$ - размер поддерева вершины r

```
unite(u, v) {
    u = get(u) \ \ корень 1 дерева
    v = get(v) \ \ корень 2 дерева
    if (s[u] < s[v]) swap(u, v)
    p[v] = u \ \ подвешиваем дерево с меньшим размером
    s[u] += s[v]
}
```

Будем дополнительно использовать эвристику сжатия путей:

Если мы впервые обращаемся к вершине (что означает, что вершина пока не подвешена напрямую к корню), то все промежуточные до корня вершины мы будем обновлять корректно и подвешивать к корню.

Теорема 1.1. (6/8) Пусть функция Акермана $A(m, n)$:

$$A(m, n) =: \begin{cases} n + 1 & \text{если } m = 0 \\ A(m - 1, 1) & \text{если } m > 0, n = 0 \\ A(m - 1, A(m, n - 1)) & \text{если } n, m > 0 \end{cases} \quad (1.1)$$

Определим $\alpha(k) = \min\{n : A(n, n) \geq k\}$ Тогда асимптотика обработки запроса есть $O^*(\alpha(m))$

Замечание. $A(4, 4) = 2^{2^{2^{65536}}} - 3$. Так что $\alpha(n)$ - довольно маленькое число.

Возвращаясь к алгоритму Крускала, получаем асимптотику $O(SORT + m \cdot \alpha(n))$

1.3 Алгоритм Борувки

1. Для каждой вершины определить самое дешевое исходящее ребро.
2. Все выбранные ребра добавить в ответ; сжать компоненты связности и запустить рекурсию.

Замечание. Если из вершины выходит несколько ребер минимального веса, то мы будем брать то из них, которое имеет минимальный номер.

Почему не появляется циклов?

Ориентируем выбранные ребра "от себя". Получим функциональный граф. Понятно, что для каждой вершины будет выбрано ровно 1 исходящее ребро (на этапе рассмотрения этой вершины).

1. Петель не бывает.

Доказательство. Очев. □

2. Цикл длины больше 2 возникнуть не может.

Доказательство. Несложно понять, что пройдя по этому циклу, мы получим, что ребра на нем не возрастают. Но если рассмотреть первую рассмотренную вершину из этого цикла, то мы получим, что все веса должны быть равны. В этом случае произойдет убывание номеров ребер у вершин:

$$number(1) < number(2) < number(3) < \dots < number(n) < number(1)$$

□

3. Цикл длины 2 возникает только при ориентировании одного ребра в 2 стороны

Доказательство. Можно считать, что всегда между 2 вершинами существует не больше 2 ребер (ведь среди кратных можно оставить наименьшее). А тогда цикл на 2 вершинах возможен только при описанном ориентировании. \square

Корректность?

Мы получаем некоторое остовное дерево. Почему же оно является минимальным?

Ответ - в многократном применении леммы о безопасном ребре.

Асимптотика: $O(m \log n)$

Утверждение 1.1. *Глубина рекурсии всегда не превосходит $\log n$*

Доказательство. В худшем случае количество вершин уменьшается вдвое, так как каждая компонента будет содержать хотя бы 2 вершины. Алгоритм остановится, когда вершина будет 1. \square

=====

Теорема 1.2. *Универсальная Машина Тьюринга существует.*

1.4 Тезис Черча-Тьюринга

Подробнее...

1.5 Проблема самоприменимости

Пусть нам дана УМТ U . Рассмотрим $S = \{p | U(p, p) \text{ остановится}\}$

Теорема 1.3 (Тьюринга). S — *перечислимо, неразрешимо*

Доказательство.

Перечислимость: S перечислимо, т.к. $S = \text{Dom } d(p) = U(p, p)$ — область определения вычислимой функции

Нерешимость: от противного. Пусть S разрешимо $\Leftrightarrow \chi_S$ вычислимо. Рассмотрим

$$f(x) = \begin{cases} 0, & \chi_S(x) = 0 \\ \neq U(x, x), & \chi_S(x) = 1 \end{cases}$$

Т.к. $U(p, p)$ — УВФ, то $\exists q \forall x U(q, x) = f(x)$.

(a) $q \in S \Rightarrow U(q, q) = f(q) \neq U(q, q)$

(b) $q \notin S \Rightarrow f(q) = 0$, но $U(q, q)$ не определено

Получили противоречие.

\square

1.6 Проблема остановки

$$H = \{(p, x) | U(p, x) \text{ остановится}\}$$

Теорема 1.4. H — перечислимо, неразрешимо

Доказательство.

Перечислимость: H перечислимо, т.к. $S = \text{Dom } U(p, x)$ — область определения вычислимой функции

Нерешимость: от противного. Пусть H разрешимо, но тогда и $S = H \cap D$ тоже разрешимо, где $D = \{(p, p) | p \text{ произвольное}\}$, противоречие, т.к. S неразрешимо.

□

Рассмотрим множества:

1. $C = \{p | \forall x, y (U(p, x), U(p, y) \text{ определены} \Rightarrow U(p, x) = U(p, y))\}$
2. $T = \{p | \forall x (U(p, x) \text{ определено})\}$
3. $FD = \{p | \{x | U(p, x) \text{ определено}\} \text{ конечно}\}$

Определение 1.3. Множество X называется коперечислимым, если \overline{X} перечислимо

Утверждение 1.2. C — коперечислимо, неразрешимо

Доказательство. $\overline{C} = \{p | \exists (x, y, t, s) \ U(p, x) \text{ ост. за } t \text{ шагов}, U(p, y) \text{ ост. за } s \text{ шагов}, U(p, x) \neq U(p, y)\}$, а оно перечислимо

□

Утверждение 1.3. T, FD — коперечислимо, некоперечислимо

1.7 m -сводимость

Определение 1.4. $A \leq_m B$, если \exists вычислимая всюду определенная функция $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$, такая, что $\forall x (x \in A \Leftrightarrow f(x) \in B)$.

1.7.1 Свойства m -сводимости

Утверждение 1.4. $A \leq_m B, B \text{ разрешимо} \Rightarrow A \text{ — разрешимо}$

Доказательство. $\chi_A(x) = \chi_B(f(x))$

□

Утверждение 1.5. $A \leq_m B \Leftrightarrow \overline{A} \leq_m \overline{B}$

Утверждение 1.6. $A \leq_m B, B \leq_m C \Leftrightarrow A \leq_m C$

Замечание. Т.к. $A \leq_m A$, получили, что \leq_m — предпорядок

Утверждение 1.7. $A \leq_m B, B \text{ перечислимо} \Rightarrow A \text{ — перечислимо}$

Следствие. $A \leq_m B, A \text{ неперечислимо} \Rightarrow B \text{ неперечислимо}$

1.8 Как получать новые неперечислимые множества?

Заметим, что \overline{H} — неперечислимое множество. Если мы покажем, что

$$\begin{cases} \overline{H} \leq_m T \\ \overline{H} \leq_m \overline{T} \end{cases}$$

То мы получим, что T — неперечисливо, неперечисливо. Однако, удобнее доказывать, что

$$\begin{cases} H \leq_m T \\ H \leq_m \overline{T} \end{cases}$$

Для этого

1. (p, x)

Получили T , неперечислимое и неперечислимое.

1.9 Существует ли универсальная тотально вычислимая функция?

Что мы хотим: $U_T : \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}^*$, такую, что

1. $\forall p, x \ U_T(p, x)$ определена
2. U_T вычислима
3. $\forall f : \{0, 1\}^* \rightarrow \{0, 1\}^*$, такая, что f — вычислима, всюду определена и $\exists p \forall x f(x) = U_T(p, x)$

Утверждение 1.8. $\nexists U_T$

Доказательство. Иначе рассмотрим $d'(x) = U_T(x, x) + 1$. Тогда $\exists p \forall x \ d'(x) = U_T(p, x) = d'(p) \Rightarrow U_T(p, p) + 1 = U_T(p, p)$ \square

Рассмотрим множество $NED = \{p | \exists x U(p, x) \text{ определено}\}$. Оно перечисливо (можно добавить квантор, указывающий, за какое число шагов).

Утверждение 1.9. \overline{NED} неперечисливо

Доказательство. Покажем, что $H \leq_m NED$. Сопоставим $(p, x) \mapsto q$, так, что $U(q, y) =$
 $\begin{cases} \text{не определено, } U(p, x) \text{ не остановится} \\ 1, \text{ иначе} \end{cases}$ Пусть p_1, p_2, \dots — перечисление \overline{NED} . \square

»»» > a7b51bb405b4c4ab01f426e170cee462c42220db