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

ОСНОВЫ КОМБИНАТОРИКИ И ТЕОРИИ ЧИСЕЛ

II СЕМЕСТР

Лектор: *Андрей Михайлович Райгородский*

$h\nu$

Автор: *Киселев Николай*
Репозиторий на Github

весна 2025

Содержание

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.*

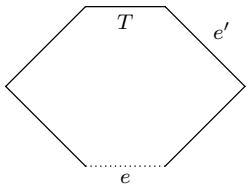
Доказательство. $S + e \subset T$

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.

$$\omega(e) \leqslant \omega(e')$$

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 $S = \varnothing$
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 дешевок ребро между C_i и $V \setminus C_i$ и добавлять его, как в примерах выше.

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

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 добавляется 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)

$\Longrightarrow$ **unite:** объединение двух непересекающихся множеств.

$\Longrightarrow$ **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. *(б/д) Пусть функция Акермана* $A(m,n)$:

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

Определим $\alpha(k) = min\{n : A(n,n) \geqslant k\}$ *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:

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

   □

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. □

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. □