Остовные деревья

Denis Bakin

Задача о минимальном остове

Дан: связный неориентированный взвешенный граф $G = (V, E)$

Найти: остов минимального суммарного веса

Задача о минимальном остове

Дан: связный неориентированный взвешенный граф $G = (V, E)$

Найти: остов минимального суммарного веса

Остов графа:

- содержит все вершины
- связен
- не содержит циклов

Почему ответ обязательно дерево?

- Если в выбранном подграфе есть цикл, то из него можно удалить любое ребро и не нарушить связность
- Значит, в оптимальном ответе циклов быть не может
- Если подграф не связен, то он просто не удовлетворяет условию задачи

Почему ответ обязательно дерево?

- Если в выбранном подграфе есть цикл, то из него можно удалить любое ребро и не нарушить связность
- Значит, в оптимальном ответе циклов быть не может
- Если подграф не связен, то он просто не удовлетворяет условию задачи

Следовательно, оптимальный ответ — это дерево на всех вершинах, то есть **минимальный остов** (*minimum spanning tree*, MST)

Жадная стратегия

Пусть T — уже выбранные рёбра

Жадная стратегия

Пусть T — уже выбранные рёбра

- Подграф T назовем **безопасным**, если он содержится в некотором минимальном остове
- Хотим на каждом шаге добавлять к T ребро так, чтобы это свойство сохранялось

Жадная стратегия

Пусть T — уже выбранные рёбра

- Подграф T назовем **безопасным**, если он содержится в некотором минимальном остове
- Хотим на каждом шаге добавлять к T ребро так, чтобы это свойство сохранялось

Ключевой вопрос: **какие рёбра можно добавлять жадно?**

Разрез графа

- **Разрез** — разбиение множества вершин на две непустые части
- Ребро **пересекает** разрез, если его концы лежат в разных частях
- Разрез **согласован с** T, если ни одно ребро из T его не пересекает

Разрез графа

- **Разрез** — разбиение множества вершин на две непустые части
- Ребро **пересекает** разрез, если его концы лежат в разных частях
- Разрез **согласован с** T, если ни одно ребро из T его не пересекает

Если выбрать правильный разрез, то минимальное ребро через него можно безопасно добавить в ответ

Лемма о безопасном ребре

Лемма. Пусть T — безопасный подграф, а разрез согласован с T. Тогда любое ребро минимального веса среди рёбер, пересекающих этот разрез, безопасно для T

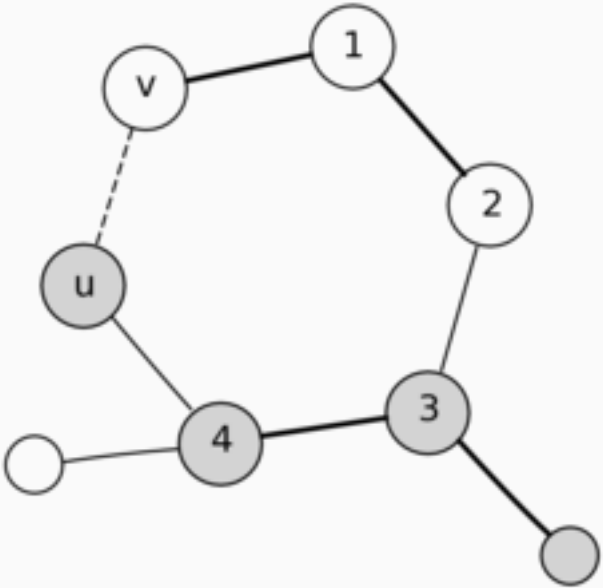
Лемма о безопасном ребре

Лемма. Пусть T — безопасный подграф, а разрез согласован с T. Тогда любое ребро минимального веса среди рёбер, пересекающих этот разрез, безопасно для T

Следствие: минимальный остов можно строить **жадно**:

- на каждом шаге находим некоторый согласованный разрез
- добавляем минимальное ребро, пересекающее этот разрез

Идея доказательства



Идея доказательства

- Возьмём минимальный остов M, содержащий T
- Если выбранного ребра e нет в M, добавим его: появится цикл
- На этом цикле найдётся ребро f, пересекающее тот же разрез
- Так как e минимально на разрезе, то $w(e) \leq w(f)$
- Заменим f на e и получим другой минимальный остов, уже содержащий $T \cup \{e\}$

Полезные следствия

- Если все веса рёбер различны, то минимальный остов **единственен**
- Любой минимальный остов минимизирует вес **самого тяжёлого ребра**
- Если все веса положительны, то минимальный остов также минимизирует **произведение** весов рёбер

Алгоритм Прима

Идея: строим остов постепенно, поддерживая множество уже выбранных вершин

Алгоритм Прима

Идея: строим остов постепенно, поддерживая множество уже выбранных вершин

1. Выбираем произвольную стартовую вершину
2. Находим ребро минимального веса, которое выходит из текущего остова наружу
3. Добавляем это ребро и новую вершину
4. Повторяем, пока не добавим все вершины

Алгоритм Прима

Идея: строим остов постепенно, поддерживая множество уже выбранных вершин

1. Выбираем произвольную стартовую вершину
2. Находим ребро минимального веса, которое выходит из текущего остова наружу
3. Добавляем это ребро и новую вершину
4. Повторяем, пока не добавим все вершины

Корректность следует из леммы: в каждый момент мы берём минимум на разрезе «уже выбранные вершины / остальные»

Три реализации Прима

Идея	Сложность	Когда удобна
Каждый раз просматривать все рёбра	$O(nm)$	самая простая
Хранить `min_edge[v]`, искать минимум линейно	$O(n^2)$	плотные графы
Использовать `set` / кучу для выбора минимума	$O(m \log n)$	разреженные графы

Алгоритм Прима за $O(n^2)$

```
std::vector<std::vector<std::pair<int, int>>> g(n);
std::vector<bool> used(n, false);
std::vector<int> min_edge(n, INF), parent(n, -1);
min_edge[0] = 0;

for (int i = 0; i < n; ++i) {
    int v = -1;
    for (int u = 0; u < n; ++u) {
        if (!used[u] && (v == -1 || min_edge[u] < min_edge[v])) {
            v = u;
        }
    }
    used[v] = true;
    for (auto [u, w] : g[v]) {
        if (!used[u] && w < min_edge[u]) {
            min_edge[u] = w;
            parent[u] = v;
        }
```

Алгоритм Прима за $O(m \log n)$

Используем те же массивы, что и в реализации за $O(n^2)$:

- `min_edge[v]` — минимальный вес ребра, которым можно присоединить вершину v
- `parent[v]` — из какой вершины это ребро приходит
- `used[v]` — уже добавлена ли вершина в остов

Алгоритм Прима за $O(m \log n)$

Используем те же массивы, что и в реализации за $O(n^2)$:

- `min_edge[v]` — минимальный вес ребра, которым можно присоединить вершину v
- `parent[v]` — из какой вершины это ребро приходит
- `used[v]` — уже добавлена ли вершина в остов

Вместо линейного поиска минимума поддерживаем `set`:

- в структуре лежат пары `(min_edge[v], v)`
- извлечение вершины с минимальным `min_edge[v]` занимает $O(\log n)$

Алгоритм Прима за $O(m \log n)$

Используем те же массивы, что и в реализации за $O(n^2)$:

- `min_edge[v]` — минимальный вес ребра, которым можно присоединить вершину v
- `parent[v]` — из какой вершины это ребро приходит
- `used[v]` — уже добавлена ли вершина в остов

Вместо линейного поиска минимума поддерживаем `set`:

- в структуре лежат пары `(min_edge[v], v)`
- извлечение вершины с минимальным `min_edge[v]` занимает $O(\log n)$

После выбора вершины v:

- помечаем её как использованную
- просматриваем все рёбра (v, u)
- если ребро улучшает текущее значение `min_edge[u]`, обновляем `min_edge[u]`, `parent[u]` и значение вершины в очереди

Какую реализацию выбирать?

Какую реализацию выбирать?

Граф	$O(n^2)$	$O(m \log n)$
Плотный ($m \approx n^2$)	$O(n^2)$	$O(n^2 \log n)$ — хуже
Разреженный ($m \approx n$)	$O(n^2)$	$O(n \log n)$ — лучше

Итоги

- Минимальный остов — это остов минимального суммарного веса
- Лемма о безопасном ребре объясняет, почему жадный подход работает
- Алгоритм Прима растит один остов, каждый раз добавляя минимальное «выходящее» ребро
- Реализации: $O(n^2)$ и $O(m \log n)$