

План лекции

- 1 Введение
- 2 Алгоритм
- 3 Время работы в худшем случае
- 4 Время работы в среднем

с/к “Эффективные алгоритмы”

Лекция 16: Линейный вероятностный алгоритм построения минимального покрывающего дерева

А. Куликов

Computer Science клуб при ПОМИ
<http://logic.pdmi.ras.ru/~infclub/>

План лекции

- 1 Введение
- 2 Алгоритм
- 3 Время работы в худшем случае
- 4 Время работы в среднем

Минимальное покрывающее дерево

Определение

- **Покрывающим деревом (связного) графа** (spanning tree) называется его поддерево на всех вершинах графа.
- **Задача о минимальном покрывающем дереве** (minimum spanning tree problem, MST) заключается в нахождении по данному взвешенному (с вещественными весами на рёбрах) графу покрывающего дерева минимального веса.

Замечания

- Будем считать, что веса всех рёбер различны. При таком предположении у графа есть ровно одно минимальное покрывающее дерево.
- Для несвязного графа имеет смысл искать минимальный покрывающий лес.

Общая схема детерминированных алгоритмов

Общая схема детерминированных алгоритмов

- На каждом шаге поддерживаем ациклический подграф F графа G , который будем называть **промежуточным остовным лесом** (intermediate spanning forest).
- F является подграфом минимального покрывающего дерева, а каждая компонента F является минимальным покрывающим деревом подграфа на вершинах этой компоненты.
- На каждом шаге алгоритм соединяет различные компоненты, добавляя рёбра графа G в F .
- Изначально, F содержит $|V|$ деревьев, состоящих из одной вершины.
- В конце работы алгоритма F состоит из одного дерева на $|V|$ вершинах, которое и является минимальным покрывающим деревом.

Безопасные и бесполезные рёбра

Определение

Рассмотрим граф G и его промежуточный подлес F .

- Ребро G называется **бесполезным** (useless), если оно не является ребром F , в то время как оба его конца принадлежат одной компоненте F .
- Ребро G называется **безопасным** (safe), если для некоторой компоненты F это ребро является самым лёгким из всех рёбер, соединяющих вершины этой компоненты с оставшимися вершинами.

Два важных свойства минимального покрывающего дерева

Свойство цикла

Для любого цикла C самое тяжёлое ребро C не содержится в минимальном покрывающем дереве.

Свойство разреза

Для любого подмножества вершин $X \subset V$ самое лёгкое ребро между X и $V \setminus X$ содержится в минимальном покрывающем дереве.

Следствие

Минимальное покрывающее дерево содержит все безопасные рёбра и не содержит бесполезных рёбер.

Известные алгоритмы

Известные алгоритмы

Алгоритм Борувки Добавляем все безопасные рёбра и производим рекурсивный вызов.

Алгоритм Прима В каждый момент есть ровно одна нетривиальная компонента связности. На каждом шаге добавляем в неё безопасное ребро.

Алгоритм Крускала Перебираем рёбра в порядке возрастания их веса. Если ребро является безопасным, добавляем его.

Общая схема алгоритма Борувки

Общая схема

- Для каждой вершины выберем ребро минимального веса, инцидентное ей.
- Каждую компоненту связности, образованную выбранными рёбрами, стянем в вершину, после чего удалим висячие вершины, петли, а при появлении кратных рёбер оставим только самое лёгкое.
- Произвести рекурсивный вызов для полученного графа.

Время работы

На каждом шаге количество вершин уменьшается хотя бы вдвое, а количество рёбер не увеличивается. Значит, время работы есть $O(|E| \log |V|)$.

F -тяжёлые рёбра

Определение

Пусть лес F является подграфом графа G .

- Через $F(x, y)$ обозначим путь (если такой есть вообще) из вершины x в вершину y в лесе F .
- Через $w_F(x, y)$ обозначим вес тяжелейшего ребра пути $F(x, y)$ (если такого пути нет, положим $w_F(x, y)$ равным ∞).
- Ребро (x, y) называется **F -тяжёлым** (F -heavy), если $w(x, y) > w_F(x, y)$, и **F -лёгким** (F -light) в противном случае.

Замечания

- Все рёбра леса F являются F -лёгкими.
- Для любого леса F ни одно F -тяжёлое ребро не входит в минимальный остовный лес.

Лёгкие и тяжёлые рёбра

Лёгкие и тяжёлые рёбра

- Ребро улучшает лес, если либо добавление этого ребра уменьшает количество деревьев в лесу, либо добавление этого ребра с последующим удалением самого тяжёлого ребра единственного образованного цикла ведёт к лесу меньшего веса.
- Лёгкие рёбра могут улучшить лес, в то время как тяжёлые — нет.

Важный факт

Факт

Существует алгоритм, проверяющий, является ли данный лес минимальным остовным лесом, за линейное время. Модификация данного алгоритма также по лесу F находит за линейное время все F -тяжёлые рёбра.

Важная лемма

Лемма

Пусть H – подграф G , полученный случайным выбором каждого ребра G с вероятностью p , а F – минимальный остовный лес H . Тогда мат. ожидание количества F -лёгких рёбер в G не превосходит n/p (где $n = |V(G)|$).

Доказательство

- Будем строить граф H и его минимальный остовный лес F одновременно.
- Изначально H и F пусты.
- Обрабатываем рёбра в порядке увеличения веса.
- Для ребра e сперва проверим, не лежат ли оба конца e в одной компоненте F .

Доказательство (продолжение)

Доказательство

- Если лежат, то e является F -тяжёлым, поскольку все текущие рёбра F имеют меньший вес.
- С вероятностью p добавим e в H . Если e добавлено в H и является F -лёгким, добавим его в F .
- Построенный таким образом лес F является минимальным остовным лесом H (в точности такой лес строит алгоритм Крускала).
- Ясно, что любое ребро e , которое было F -тяжёлым на момент добавления, будет F -тяжёлым и дальше, поскольку из F рёбра не удаляются.
- Аналогично каждое F -лёгкое ребро остаётся таковым, поскольку после добавления e добавляются только F -тяжёлые рёбра.
- При обработке ребра мы заранее знаем, является ли оно F -тяжёлым.

Доказательство (продолжение)

Доказательство

- Рассмотрим подбрасывания монетки, соответствующие F -лёгким рёбрам.
- Каждый такой эксперимент с вероятностью p (при выпадении решки) добавляет ребро в F .
- Размер F в конце не превосходит $n - 1$, поэтому подбрасывания заканчиваются после выпадения не более чем $n - 1$ решек.
- Представим, что мы подкидываем монетку до выпадения ровно n решек. Пусть Y есть общее число подбрасываний.
- Ясно, что количество F -лёгких рёбер не превосходит Y .
- Распределение случайной величины Y – отрицательное биномиальное распределение с параметрами n и p (количество независимых подбрасываний монетки до n выпадений решки, где решка выпадает с вероятностью p).
- $E[Y] = n/p$. □

План лекции

- 1 Введение
- 2 Алгоритм
- 3 Время работы в худшем случае
- 4 Время работы в среднем

Алгоритм

RANDOMIZED-MST(G)

- 1 ▷ Выход: минимальный остовный лес для G
- 2 применить три шага алгоритма Борувки к G
(количество вершин в полученном графе будет хотя бы в восемь раз меньше);
полученный граф назовем G^*
- 3 в G^* выбрать подграф H ,
включая в H каждое ребро с вероятностью $1/2$
- 4 $F = \text{RANDOMIZED-MST}(H)$
- 5 найти все F -тяжёлые рёбра графа G^* и удалить их из G^* ;
полученный граф назовём G'
- 6 $F' = \text{RANDOMIZED-MST}(G')$
- 7 **return** рёбра F' и рёбра, стянутые на первом шаге

Схема работы



Схема работы

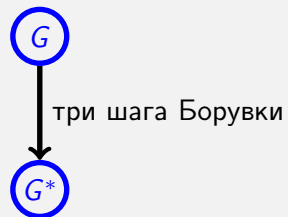


Схема работы

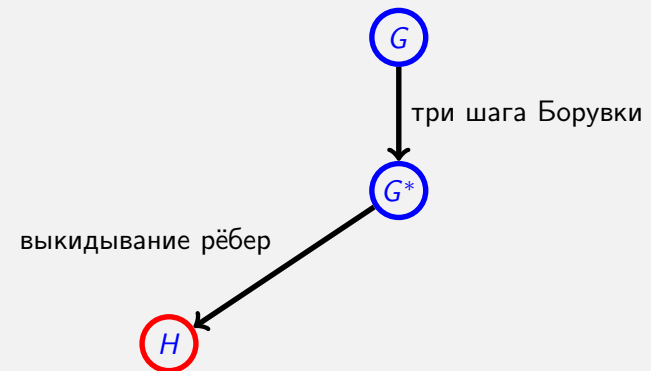


Схема работы

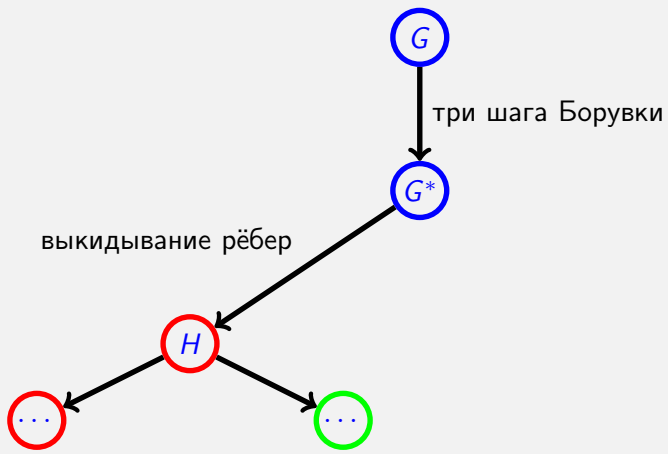


Схема работы

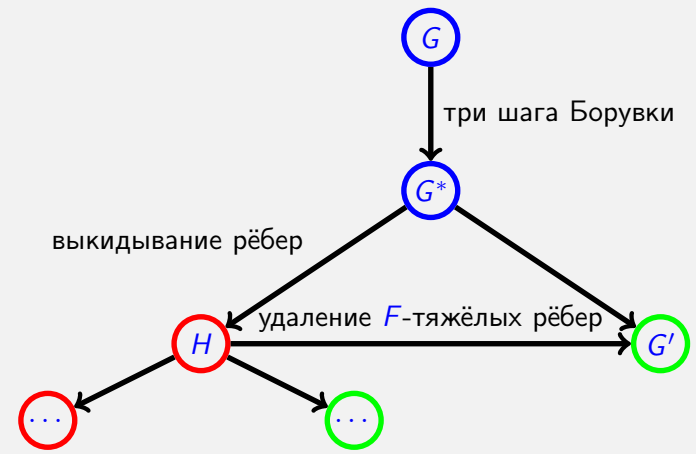
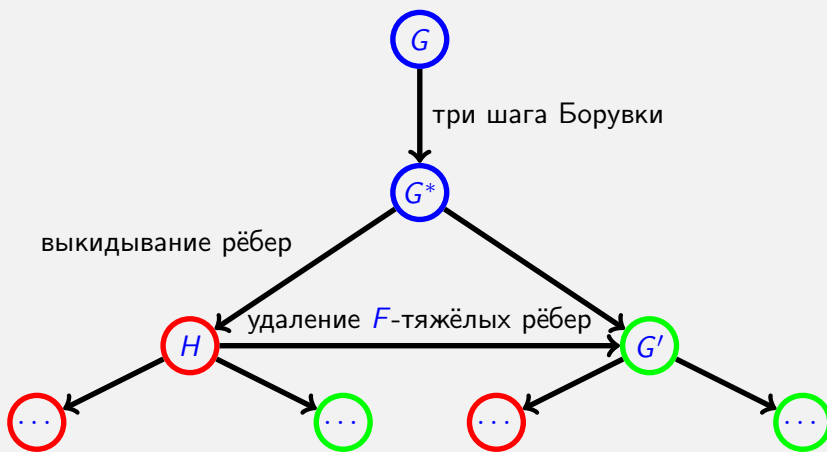


Схема работы



Анализ алгоритма

Анализ алгоритма

- Применение двух шагов Борувки занимает линейное от количества рёбер время.
- Нахождение всех F -тяжёлых рёбер также требует линейного от количества рёбер времени.
- Для оценки времени работы, таким образом, достаточно оценить общее количество рёбер в исходной задаче и всех рекурсивных вызовах.
- Допустим, что у исходного графа n вершин и m рёбер. НУО, $m \geq n/2$.
- На уровне d дерева работы алгоритма находится не более 2^d графов, в каждом из которых не более $n/8^d$ вершин.
- Значит, общее количество вершин не превосходит $\sum_{d=0}^{\infty} 2^d n/8^d = \sum_{d=0}^{\infty} n/4^d \leq 2n$.

План лекции

- 1 Введение
- 2 Алгоритм
- 3 **Время работы в худшем случае**
- 4 Время работы в среднем

Время работы в худшем случае

Теорема

Время работы алгоритма **RANDOMIZED-MST** в худшем случае есть

$$O(\min\{n^2, m \log n\}).$$

Доказательство

Оценим количество рёбер двумя способами.

- 1 Поскольку кратных рёбер нет, любой граф на уровне d содержит не более $(n/8^d)^2/2$ рёбер. Просуммировав по всем уровням, получим верхнюю оценку $O(n^2)$.
- 2 Покажем, что общее количество рёбер двух графов, на которых производятся рекурсивные вызовы, не превосходит количества рёбер исходного графа. Из этого будет следовать, что на каждом уровне не более m рёбер.

Доказательство (продолжение)

Доказательство

- Количество рёбер в левой ветке есть $|E(H)|$.
- Количество рёбер в правой ветке есть
$$\begin{aligned} & |E(G^*)| - |F\text{-тяжёлые в } G^*| \\ & \leq |E(G^*)| - |E(H)| + |E(F)| \\ & \leq (|E(G)| - |V(G)|/2) - |E(H)| + |V(G)|/2 \\ & \leq |E(G)| - |E(H)|. \end{aligned}$$

□

План лекции

- 1 Введение
- 2 Алгоритм
- 3 **Время работы в худшем случае**
- 4 **Время работы в среднем**

Время работы в среднем

Теорема

Мат. ожидание времени работы алгоритма **RANDOMIZED-MST** есть $O(m)$.

Доказательство

- Обозначим через $T(n, m)$ мат. ожидание времени работы алгоритма на графе из n вершин и m рёбер.
- Применение трёх шагов алгоритма Борувки требует $O(n + m)$ шагов и производит граф G^* с не более чем $n/8$ вершинами.
- На следующем шаге за время $O(n + m)$ производится граф H с мат. ожиданием количества рёбер не более $m/2$.

Доказательство (продолжение)

Доказательство

- Нахождение минимального остовного леса F для H требует $T(n/8, m/2)$ шагов в среднем.
- Нахождение всех F -тяжёлых рёбер делается за время $O(n + m)$ и производит граф G' с не более чем $n/8$ вершинами и мат. ожиданием количества рёбер не более $n/4$ (по доказанной лемме).
- Нахождение минимального остовного леса для G' требует времени $T(n/8, n/4)$.
- Итого,

$$T(n, m) \leq T(n/8, m/2) + T(n/8, n/4) + c(n + m),$$

откуда следует, что $T(n, m) \leq 2c(n + m)$. \square

Что мы узнали за сегодня?

Что мы узнали за сегодня?

- Существует детерминированный линейный алгоритм проверки, является ли данный лес минимальным остовным лесом данного графа.
- Используя этот алгоритм, можно построить вероятностный алгоритм для нахождения минимального остовного леса, который в худшем случае имеет время работы $O(\min\{n^2, m \log n\})$, а в среднем — $O(n + m)$.