Каркасы минимального веса

Подграфы

- Пусть $G = \langle V, E \rangle$ – граф
- Будем говорить, что граф $G' = \langle V', E' \rangle$ является подграфом графа G, если:
  - $V' \subseteq V$
  - $E' \subseteq E \cap V' \times V'$
- При этом будем говорить, что:
  - G' – подграф, индуцированный множеством вершин V', если $E' = E \cap V' \times V'$
  - G' – частичный граф, если $V' = V$
  - В общем случае $V' \subseteq V$, $E' \subseteq E \cap V' \times V'$ будем говорить, что G' – фрагмент, или подграф в широком смысле

Остовные подграфы

- Частичный граф, т.е. подграф $G' = \langle V, E' \rangle$, где $E' \subseteq E$, также называют остовным подграфом
- Остовное дерево – это остовный подграф, являющийся деревом, остовный лес – остовный подграф, являющийся лесом, и т.д.
- Остовные деревья также называют каркасами

Остовные подграфы

- Ясно, что, если граф G является связным, то у него может существовать более одного каркаса
- Например, униццикличский граф, содержащий цикл длины k, имеет k различных каркасов (впрочем, возможно, с точностью до изоморфизма их меньше)

Каркасы минимального веса

- Рассмотрим граф $G = \langle V, E \rangle$, ребра которого взвешены весовой функцией $w: E \to \mathbb{R}$
- Пусть $G' = \langle V, E' \rangle$ – каркас. Определим его вес естественным образом:

$$w(G') = \sum_{e \in E'} w(e)$$

- Нас будет интересовать задача поиска в данном графе G каркаса, вес которого минимален
- В случае существования нескольких каркасов минимального веса, необходимо найти любой из них

Алгоритмы

- Мы рассмотрим два алгоритма решения задачи
  - Алгоритм Крускала
  - Алгоритм Прима
- Оба алгоритма относятся к классу жадных алгоритмов

Алгоритм Крускала

- Пусть $G = \langle V, E \rangle$ – связный граф, $w: E \to \mathbb{R}$ – весовая функция на ребрах
- Будем строить каркас минимального веса $G' = \langle V, E' \rangle$
- Начнем с $G' = \langle V, E' \rangle$, где $E' = \emptyset$
- Будем повторять следующие шаги
  - Найдем ребро e_i минимального веса, добавление которого в G' не приводит к образованию цикла
  - Положим $G' = \langle V, E' \cup \{e_i\} \rangle$
- Ясно, что алгоритм закончит свою работу и построено будет действительно дерево

Алгоритм Крускала

- Алгоритм Крускала корректен, т.е. он действительно строит каркас минимального веса
- Предположим, что это не так, т.е. существует каркас $G'' = \langle V, E'' \rangle$ такой, что $w(G'') < w(G')$
- Пусть $e = \arg\min_{e' \in E'' \setminus E'} w(e')$ – ребро минимального веса, вошедшее в G'' и не вошедшее в G'
- Пусть также $E''' = \{e' \in E' | w(e') \leq w(e)\}$
- Ясно, что граф $\langle V, E''' \rangle$ не содержит циклов, т.к. это подграф как графа G'', так и графа G'

Алгоритм Крускала

- Добавление в граф $\langle V, E''' \rangle$ ребра e не приводит к образованию цикла, т.к. получающийся при этом граф является подграфом G''
- Значит, ребро e могло быть добавлено в граф $\langle V, E''' \rangle$ без образования цикла и при этом имело вес меньший, чем то ребро, что было добавлено алгоритмом
- Но это противоречит описанию алгоритма: он добавляет в граф ребро минимального веса, добавление которого не приводит к появлению циклов

Алгоритм Крускала

- Алгоритм проще всего реализовать, сначала отсортировав ребра по невозрастанию их весов:
$$w(e_1) \leq w(e_2) \leq \cdots \leq w\left(e_{|E|}\right)$$
- Теперь алгоритм реализуется однократным проходом по отсортированному списку ребер
- Для проверки того, что добавление ребра $\{v, u\}$ не приводит к образованию цикла, можно проверить, принадлежат ли эти вершины различным компонентам связности текущего подграфа; это можно сделать, используя лес непересекающихся множеств

Алгоритм Крускала

- Затраты на сортировку множества ребер составят
$$O(|E|\log|E|) = O(|E|\log|V|^2) = O(|E|\log|V|)$$
- Затраты на осуществление прохода по отсортированному списку ребер при использовании леса непересекающихся множеств составят
$$O(|E|)$$
- Таким образом, временная сложность алгоритма Крускала составляет
$$O(|E|\log|V|)$$

Алгоритм Прима

- Рассмотрим иной способ построения каркаса минимального веса
- Будем строить дерево следующим образом:
  - Изначально $G' = \{\{v\}, \emptyset\}$, где v – произвольная вершина графа G
  - Теперь на каждом шаге находим ребро минимального веса в графе G, соединяющее вершину, входящую в G', с вершиной, в нее не входящей; пусть это ребро $\{v, u\}$
  - Модифицируем граф: $G' = \{V' \cup \{v, u\}, E' \cup \{\{v, u\}\}\}$

Алгоритм Прима

- Предположим, что существует каркас $G'' = \langle V, E'' \rangle$, имеющий вес меньший, чем вес каркаса, найденного алгоритмом Прима
- Рассмотрим первый момент времени, когда в G' было добавлено ребро, не входящее в G''. Пусть это было ребро $e = \{v, u\}$, а S – множество вершин графа G' на тот момент
- В графе G'' вершины v и u соединены некоторым путем; пусть e' - ребро на этом пути, одна из вершин которого принадлежит S, а другая нет. Из алгоритма ясно, что $w(e) \leq w(e')$, т.к. в случае $w(e) > w(e')$ было бы добавлено ребро e', а не e

Алгоритм Прима

- Таким образом, можно рассмотреть новый остовный подграф, содержащий ребро e и не содержащий ребра e': $\langle V, E''\backslash\{e'\} \cup \{e\}\rangle$; ясно, что он также будет обладать минимальным возможным весом
- Повторяя эту операцию необходимое число раз, придем к каркасу G', чем и докажем, что он является каркасом минимального веса

Алгоритм Прима

- Для реализации этого алгоритма необходимо организовать хранение множества ребер с возможностью извлекать из него ребро минимального веса, удалять и добавлять ребра
- Мы будем хранить множество ребер, один конец которых принадлежит текущему множеству вершин, а другой – нет
- После добавления очередного ребра в граф необходимо удалить это ребро из множества хранимых ребер, но в него необходимо добавить все ребра, смежные с добавленной вершиной, второй конец которых находится вне текущего множества вершин

Алгоритм Прима

- Подходящей структурой является бинарная куча: она позволяет проводить добавлять, удалять элементы и извлекать элемент с минимальным значением ключа за логарифмическое время
- Каждое ребро будет добавлено или удалено не более одного раза, поэтому общая временная сложность алгоритма составит

$$O(|E| \cdot \log|E|) = O(|E| \cdot \log|V|)$$