

Каркасы графа

Лекция 9

Определения

$G(V,E)$ - связный неориентированный граф с заданной функцией стоимости, отображающей ребра в вещественные числа.

Остовное дерево или **каркас (скелет)** графа – это подграф, который :

- 1) содержит все вершины графа,
- 2) является деревом.

Нас интересуют алгоритмы построения **МИНИМАЛЬНОГО** каркаса.

Минимальным каркасом является такой каркас, сумма весов ребер которого минимальна.

Алгоритм Краскала (Джозеф Крускал, 1956 год)

1. Сортируем ребра графа по возрастанию весов.
2. Полагаем, что каждая вершина относится к своей компоненте связности.
3. Проходим ребра в "отсортированном" порядке. Для каждого ребра выполняем:
 - a) если вершины, соединяемые данным ребром, лежат в разных компонентах связности, то объединяем эти компоненты в одну, а рассматриваемое ребро добавляем к минимальному остовному дереву;
 - b) если вершины, соединяемые данным ребром лежат в одной компоненте связности, то исключаем ребро из рассмотрения.
4. Если есть еще нерассмотренные ребра и не все компоненты связности объединены в одну, то переходим к шагу 3, иначе **ВЫХОД**.

Время работы:

Сортировка рёбер - $O(|E| \times \log |E|)$

Компоненты связности удобно хранить в виде системы **непересекающихся** множеств.

Все операции в таком случае займут $O(E)$

Алгоритм Краскала

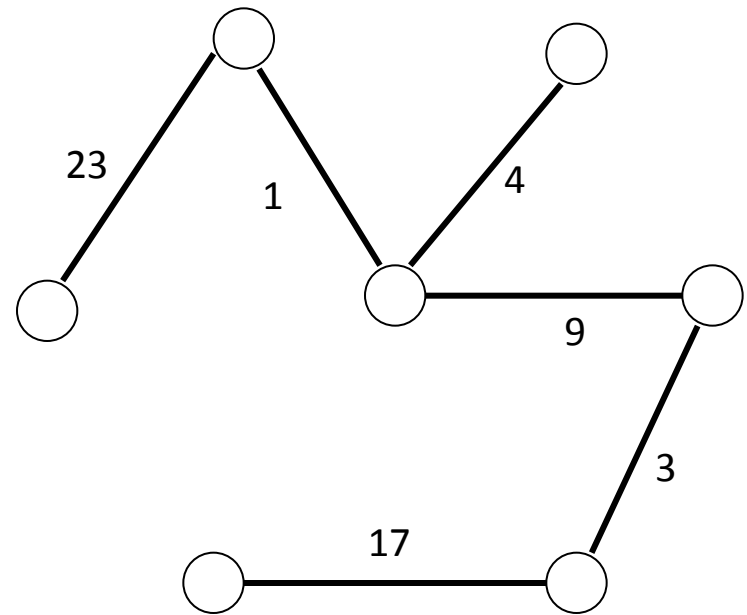
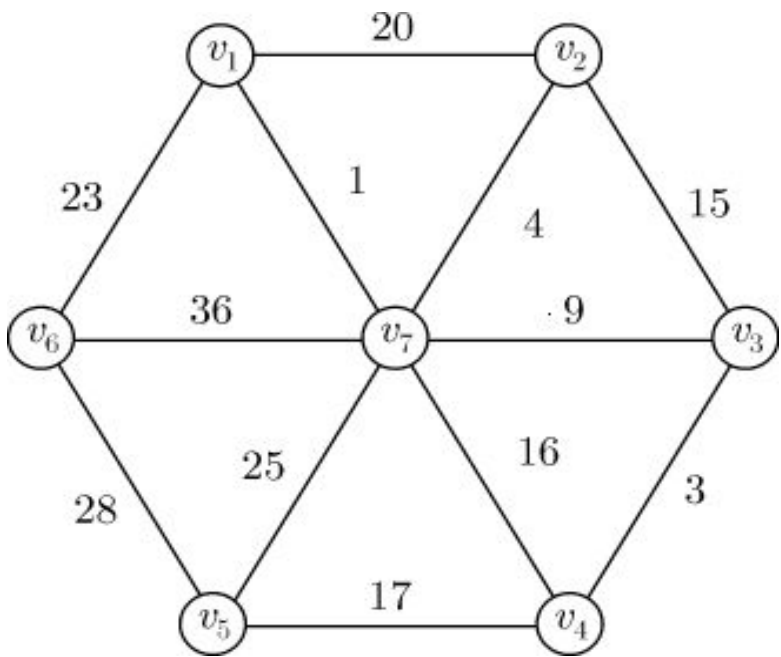
Вход. Неориентированный граф $G = (V, E)$ с функцией стоимости c , заданной на его ребрах.

Выход. Остовное дерево $S = (V, T)$ наименьшей стоимости для G .

Метод:

1. $T \leftarrow \emptyset$; // остовное дерево (каркас)
2. $VS \leftarrow \emptyset$; // набор непересекающихся множеств вершин
3. построить очередь с приоритетами Q , содержащую все ребра из E ;
4. Для $\forall v \in V$ выполнить: добавить $\{v\}$ к VS ;
Пока $|VS| > 1$ выполнить:
 6. { выбрать в Q ребро (v, w) наименьшей стоимости;
 7. удалить (v, w) из Q ;
 8. если v и w принадлежат различным множествам $W1$ и $W2$ из VS то
 9. { заменить $W1$ и $W2$ на $W1 \cup W2$ в VS ;
 10. добавить (v, w) к T ;
 - }
- }

Пример



Получившееся дерево является каркасом минимального веса.

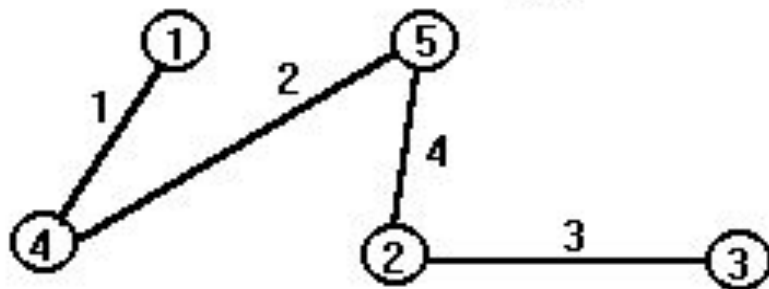
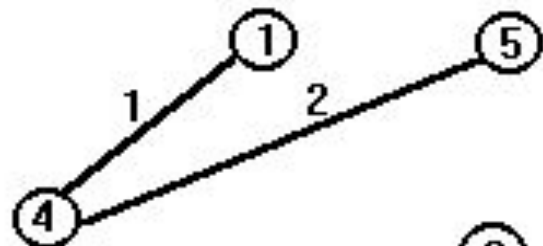
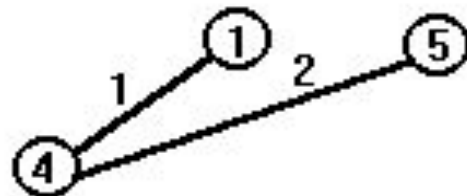
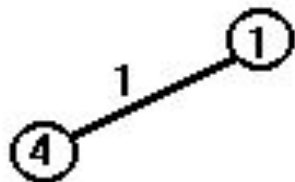
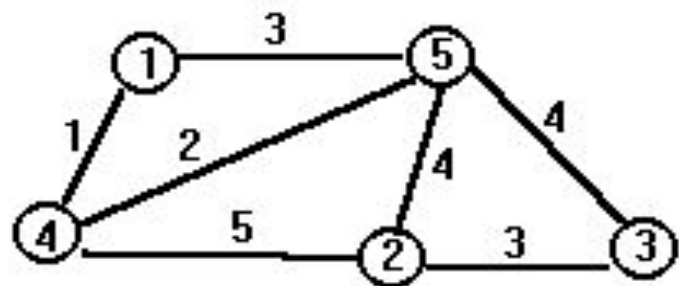
Введем массив меток вершин графа *Mark*.

Начальные значения элементов массива равны номерам

соответствующих вершин ($Mark[i] = i; i \in 1..N$).

Ребро выбирается в каркас в том случае, если вершины, соединяемые им, имеют разные значения меток.

Пример приведен на следующем слайде, изменения *Mark* показаны в таблице.



Номер итерации	Ребро	Значения элементов Mark
начальное значение	-	[1,2,3,4,5]
1	<1,4>	[1,2,3,1,5]
2	<4,5>	[1,2,3,1,1]
3	<2,3>	[1,2,2,1,1]
4	<2,5>	[1,1,1,1,1]

Алгоритм Прима

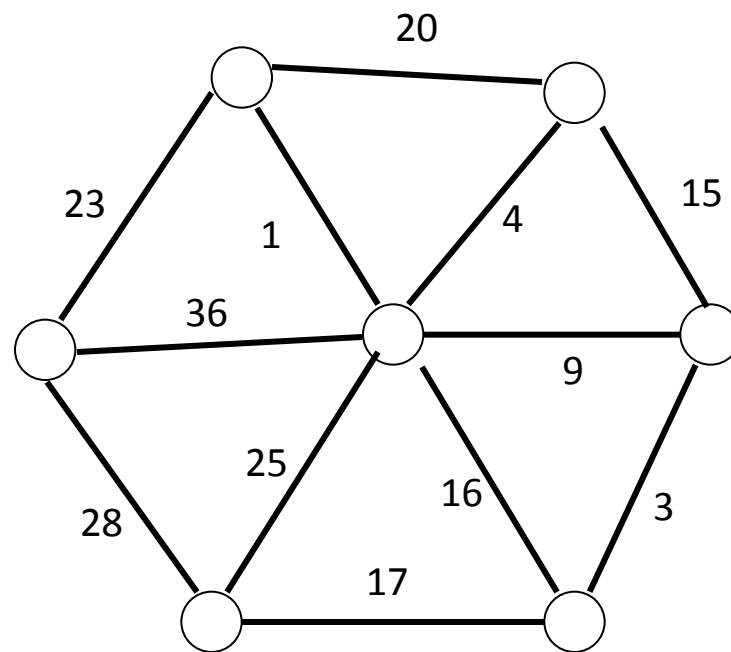
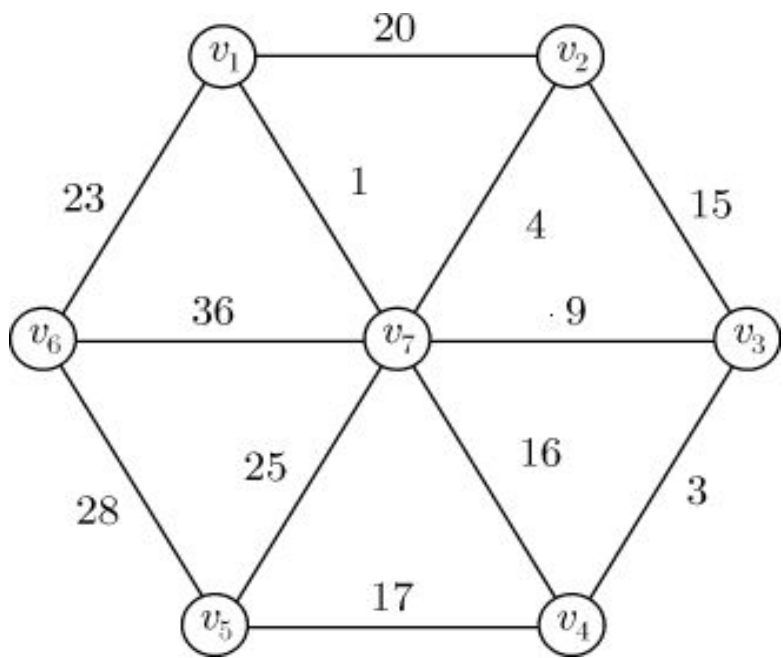
На каждом шаге вычеркиваем из графа дугу максимальной стоимости с тем условием, что она не разрывает граф на две или более компоненты связности, т.е. после удаления дуги граф должен оставаться связным.

Для того, чтобы определить, остался ли граф связным, достаточно запустить поиск в глубину от одной из вершин, связанных с удаленной дугой.

Условие окончания алгоритма?

Например, пока количество ребер больше либо равно количеству вершин, нужно продолжать, иначе – остановиться.

Пример



Алгоритм Прима (другой)

Алгоритм впервые был открыт в 1930 году чешским математиком Войцехом Ярником, позже переоткрыт Робертом Примом в 1957 году, и, независимо от них, Э. Дейкстрой в 1959 году.

Время работы алгоритма – $O(V * E)$

Можно улучшить – $O(E \log V + V^2)$

При использовании двоичной кучи – $O(E \log V)$

При использовании фибоначчиевой кучи – $O(E + V \log V)$

1) Выбирается произвольная вершина - она будет корнем остовного дерева;

2) Измеряется расстояние от нее до всех других вершин, т.е. находится

минимальное расстояние s от дерева до вершин, которые не включены в

дерево;

3) До тех пор, пока в дерево не добавлены все вершины делать:

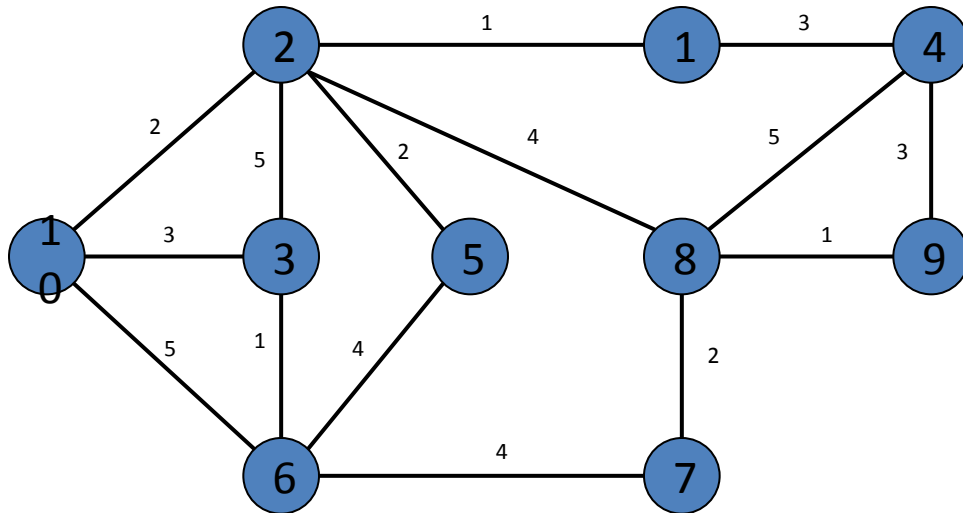
- найти вершину u , расстояние от дерева до которой минимально;

- добавить ее к дереву;

- пересчитать расстояния от невключенных вершин до дерева следующим образом:

если расстояние до какой-либо вершины от u меньше текущего расстояния s от дерева, то в s записывается новое расстояние.

Запускаем алгоритм обхода графа, начиная с произвольной вершины.
 В качестве контейнера выбираем очередь с приоритетами. Приоритет – текущая величина найденного расстояния до уже построенной части остовного дерева.
 Релаксации подвергаются прямые и обратные ребра.



n	1	2	3	4	5	6	7	8	9	10
π		1	1	1	2	3	8	9	4	2
d	0	1	3	3	2	1	2	1	3	2

В результате работы получаем список ребер остовного дерева вместе с весами

Реализация за $O(M \log N + N^2)$

Отсортируем все рёбра в списках смежности каждой вершины по увеличению

веса – $O(M \log N)$.

Для каждой вершины заведем указатель, указывающий на первое доступное

ребро в её списке смежности. Изначально все указатели указывают на начала

списков.

На i -ой итерации перебираем все вершины, и выбираем наименьшее по весу

ребро среди доступных. Поскольку все рёбра уже отсортированы по весу, а

указатели указывают на первые доступные рёбра, то выбор наименьшего

ребра осуществится за $O(N)$.

После этого обновляем указатели (сдвигаем вправо), т.к. некоторые из них

указывают на ставшие недоступными рёбра (оба конца которых оказались

внутри дерева).