MST

Гусев Илья, Булгаков Илья

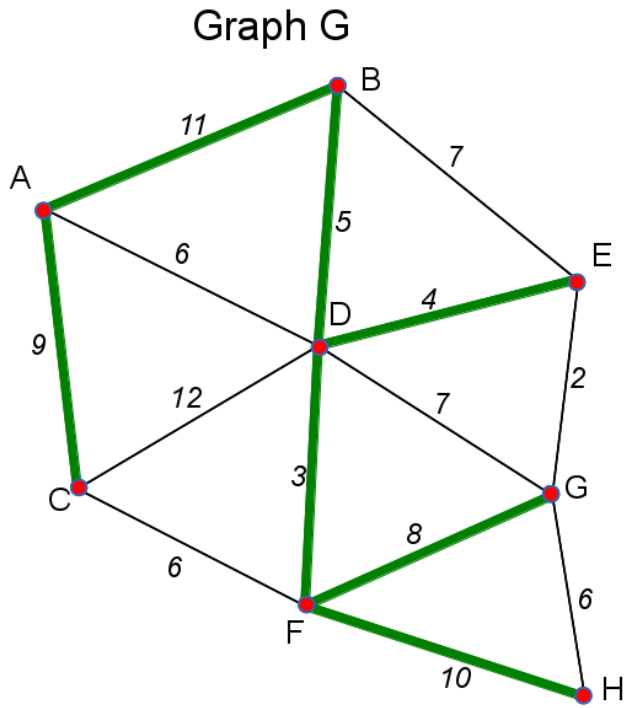
Московский физико-технический институт

Москва, 2019

Содержание

Минимальное остовное дерево

- Остовное дерево - ациклический связный подграф данного связного неориентированного графа, в который входят все его вершины.
- Минимальное остовное дерево - остовное дерево минимального веса



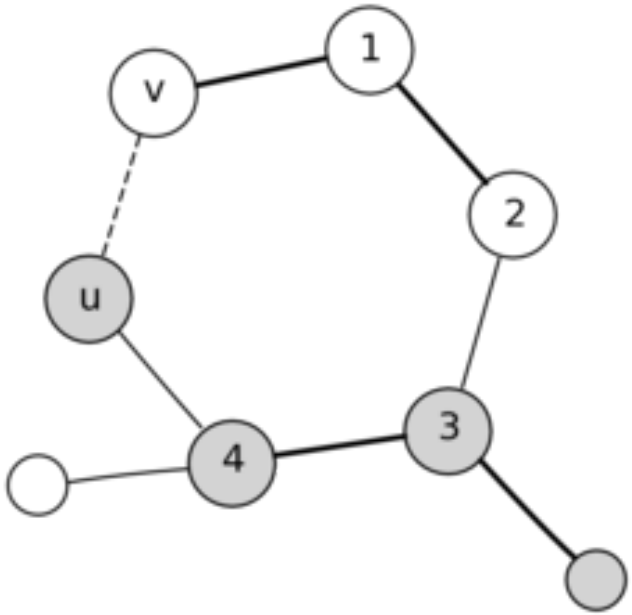
Минимальное остовное дерево

Алгоритмы для построения

- Алгоритм Прима (повторение алгоритма Дейкстры)
- Алгоритм Краскала
- Алгоритм Борувки

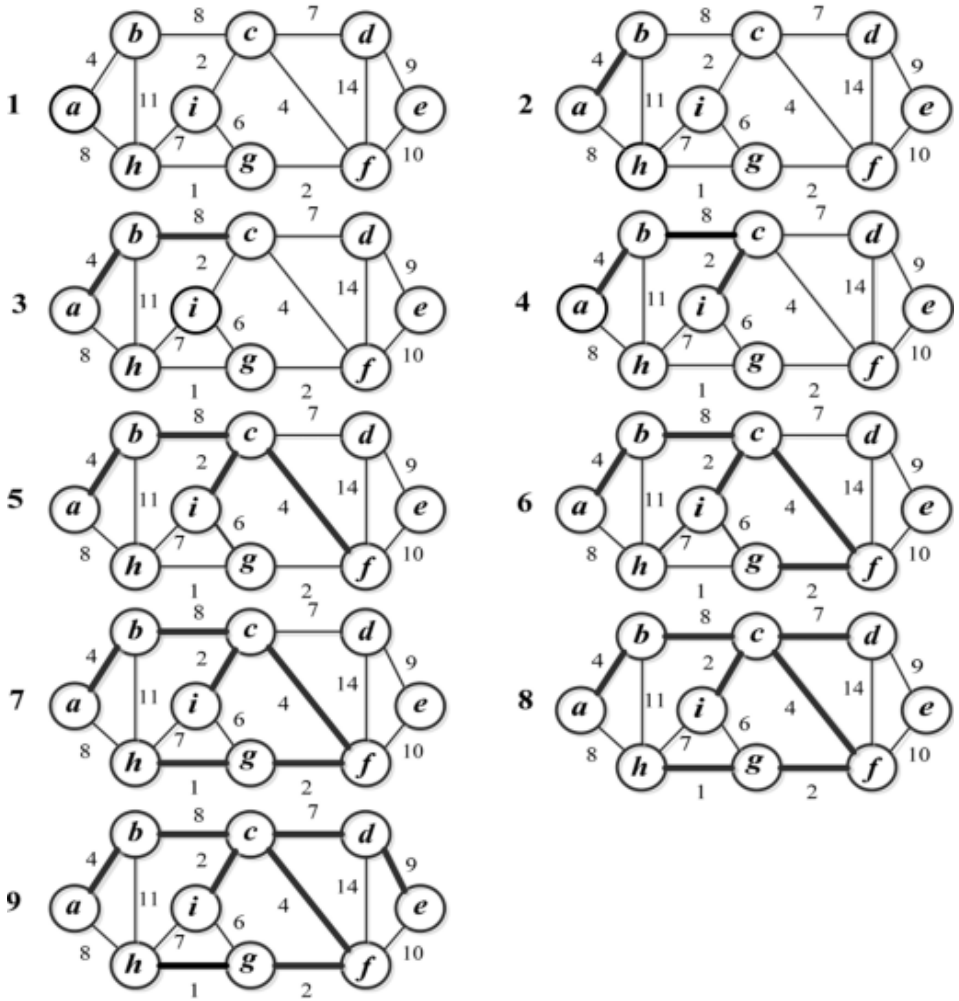
Лемма о безопасном ребре

Рассмотрим связный неориентированный взвешенный граф $G = (V, E)$. Пусть $G^1 = (V, E^1)$ — подграф некоторого минимального остовного дерева G, $< S, T >$ — разрез G, такой, что ни одно ребро из E^1 не пересекает разрез, а (u, v) — ребро минимального веса среди всех ребер, пересекающих разрез $< S, T >$. Тогда ребро $e = (u, v)$ является безопасным для G^1.



Алгоритм Прима

Выбираем всегда кратчайшее ребро из соседних к уже построенному поддереву

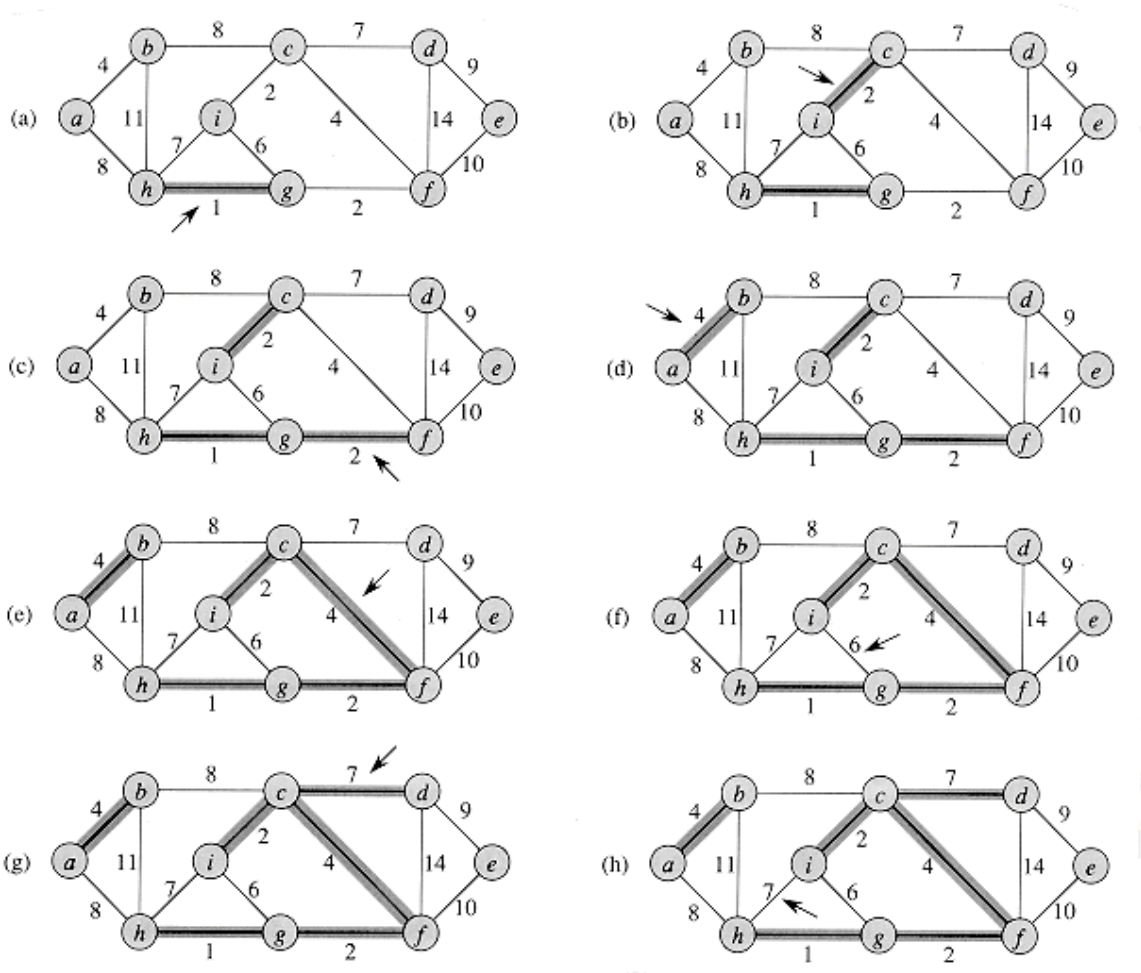


Алгоритм Прима

Ничего не напоминает?
Какая асимптотика?

Алгоритм Крускала

- Сортировка всех рёбер по весу
- На каждом шаге берём минимальное, которое не образует цикл



Алгоритм Крускала

- Наивная реализация: $O(E \cdot log(E) + E + V^2)$
- CHM: $O(E \cdot log(E) + E + V)$

Система непесекающихся множеств

Структура, поддерживающая операции над множествами: объединить два множества; проверить, в каком множестве элемент. Английское название — disjoint set union (DSU)

- make_set(x) — добавляет новый элемент x, помещая его в новое множество, состоящее из одного него.
- union_sets(x,y) — объединяет два указанных множества (множество, в котором находится элемент x, и множество, в котором находится элемент y).
- find_set(x) — возвращает, в каком множестве находится указанный элемент x. На самом деле при этом возвращается один из элементов множества (называемый представителем или лидером (в англоязычной литературе "leader")). Этот представитель выбирается в каждом множестве самой структурой данных (и может меняться с течением времени, а именно, после вызовов union_sets())

Система непесекающихся множеств

Наивная реализация. Каждое множество храним в виде дерева. Используем массив parent - для каждого элемента храним родителя.

```
vector<int> parent(elementsCount);

void make_set (int v) {
        parent[v] = v;
}

int find_set (int v) {
        if (v == parent[v])
                return v;
        return find_set (parent[v]);
}

void union_sets (int a, int b) {
        a = find_set (a);
        b = find_set (b);
        if (a != b)
                parent[b] = a;
}
```

Система непесекающихся множеств. Оптимизации

Эвристика сжатия пути

```
int find_set (int v) {
        if (v == parent[v])
                return v;
        return parent[v] = find_set (parent[v]);
}
```

Система непесекающихся множеств. Оптимизации

Эвристика объединения по рангу или размеру дерева. Заводим дополнительный массив size, который хранит число вершин в поддереве (есть модификация с глубиной дерева)

```
vector<int> size(elementsCount);

void union_sets (int a, int b) {
        a = find_set (a);
        b = find_set (b);
        if (a != b) {
                if (size[a] < size[b])
                        swap (a, b);
                parent[b] = a;
                size[a] += size[b];
        }
}
```

Система непесекающихся множеств. Сложность

Время работы

- Наивный - может вырожиться в цепочку вплоть до O(n)
- Использование одной из эвристик дает среднее время O(log n)
- Использование обоих эвристик дает среднее время O(1)

Задача коммивояжёра

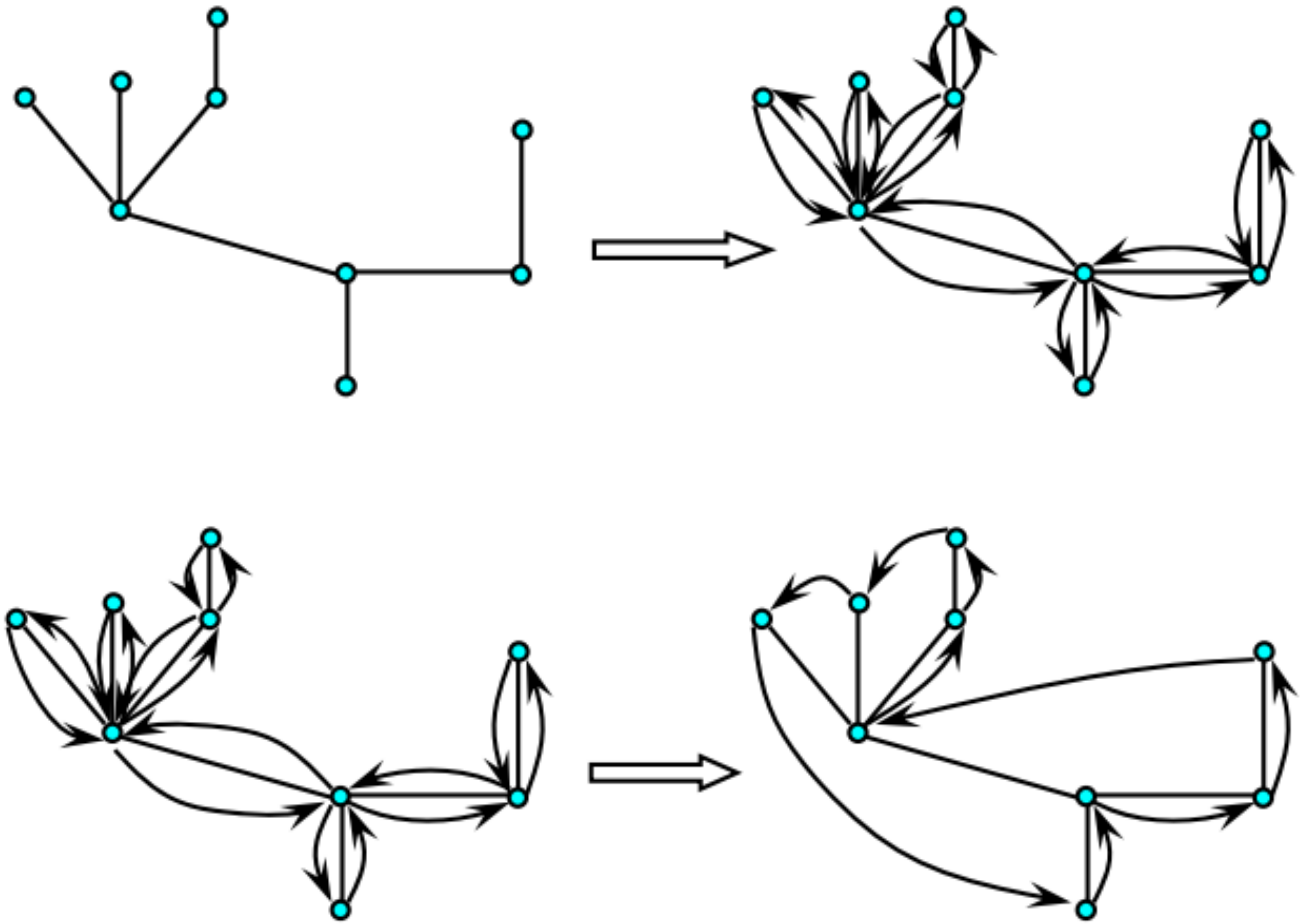
- Задача коммивояжёра — нахождение мнимального пути между городами с посещением каждого города ровно один раз.
- Решение задачи коммивояжёра — это нахождение гамильтонова цикла минимального веса.

Задача коммивояжёра

Ограничения:

- Полностью связный граф
- Используем метрическую постанвку: выполняется неравенство треугольника

Задача коммивояжёра и MST



Полезные ссылки I

- Задача коммивояжера - Институт математики СО РАН
  http://www.math.nsc.ru/LBRT/k5/OR-MMF/TSPr.pdf
- Викиконспекты: лемма о безопасном пути
  https://bit.ly/2YIlQvk
- Викиконспекты: лемма о безопасном пути
  https://bit.ly/2YIlQvk