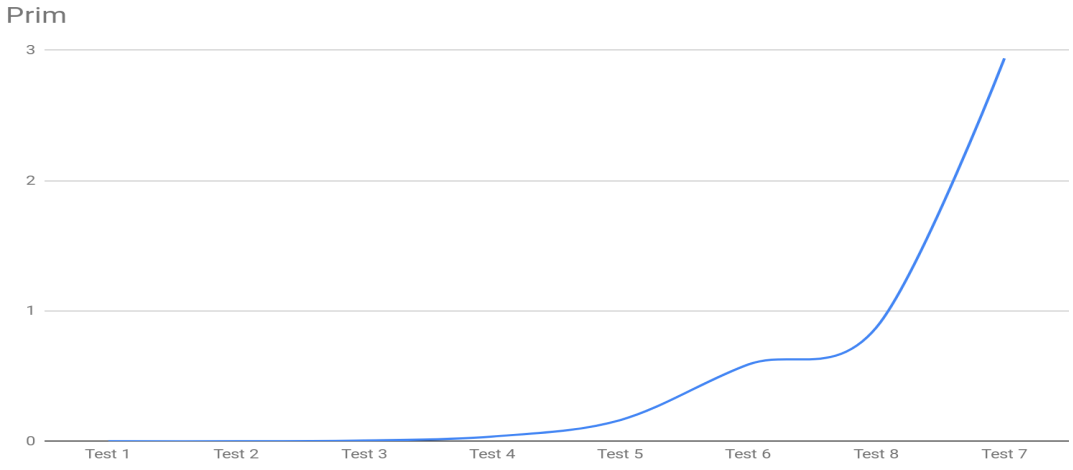
Исследование алгоритмов поиска минимального отстовного дерева

Замечания

В ходе данного исследования были реализованы и протестированы алгоритмы нахождения минимального остовного дерева (MST).
Здесь и далее приняты следующие обозначения:
V – множество вершин
E – множество рёбер
G = (E, V) - граф
n - количество вершин в графе
m – количество рёбер в графе
Для тестирования алгоритмов были использованы случайные графы, в последних тестах количество вершин превышал 100 000. Подробнее ознакомиться с реализацией и тестами можно на https://github.com/uslon/minimum_spanning_trees.

1. Алгоритм Прима

Описание:
Сначала берётся произвольная вершина и находится ребро, инцидентное данной вершине и обладающее наименьшей стоимостью. Найденное ребро и соединяемые им две вершины образуют дерево. Затем, рассматриваются рёбра графа, один конец которых — уже принадлежащая вершине дерева, а другой — нет; из этих рёбер выбирается ребро наименьшей стоимости. Выбираемое на каждом шаге ребро присоединяется к дереву. Рост дерева происходит до тех пор, пока не будут исчерпаны все вершины исходного графа.
Сложность алгоритма:
В ходе алгоритма нужно n раз извлечь минимум из кучи и m раз её обновить. Что дает $O(mlogn)$, так как $m \geq n$.
Время работы на тестах:



2. Алгоритм Краскала

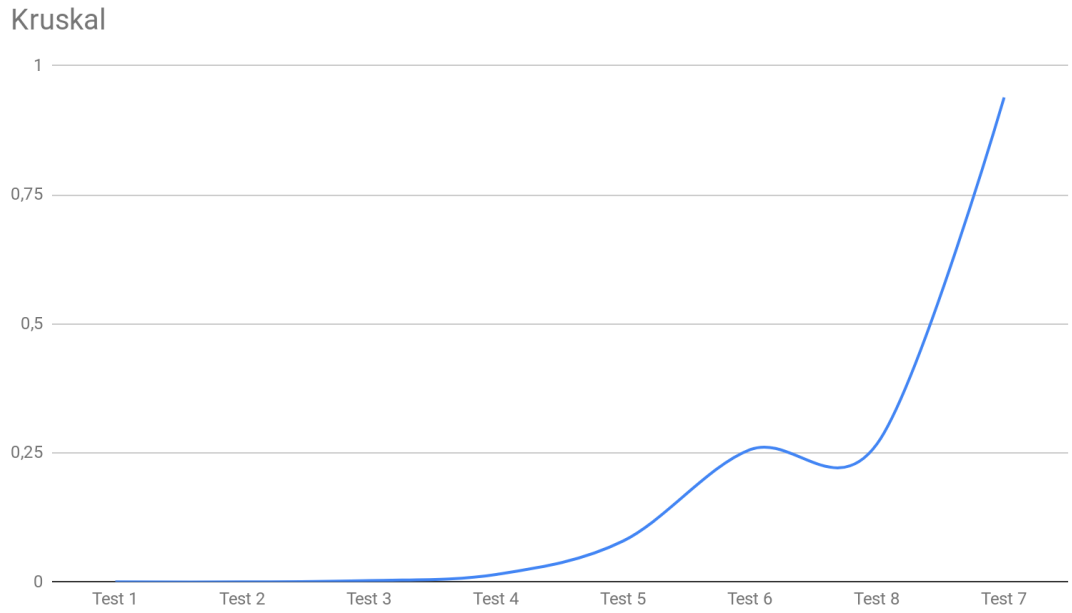
Описание:

В начале текущее множество рёбер устанавливается пустым. Затем, пока это возможно, проводится следующая операция: из всех рёбер, добавление которых к уже имеющемуся множеству не вызовет появление в нём цикла, выбирается ребро минимального веса и добавляется к уже имеющемуся множеству. Когда таких рёбер больше нет, алгоритм завершён.

Сложность алгоритма:

Основное время тратится на сортировку рёбер по весу, а также на поддержание системы непересекающихся множеств(СНМ). Поэтому ассимптотика - *O(mlogn)*

Время работы на тестах:



3. Алгоритм Борувки

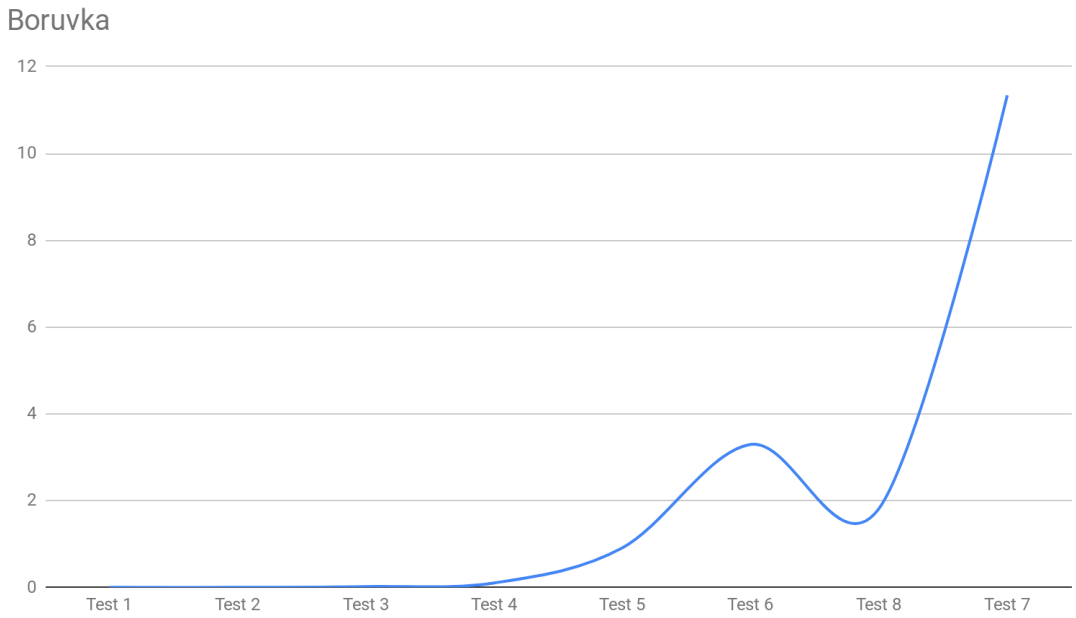
Описание:

Работа алгоритма состоит из нескольких итераций, каждая из которых состоит в последовательном добавлении рёбер к остовному лесу графа, до тех пор, пока лес не превратится в дерево, то есть, лес, состоящий из одной компоненты связности.

Сложность алгоритма:

Алгоритм состоит из *log n* фаз, так как после каждой фазы количество компонент уменьшается как минимум вдвое. На каждой фазе мы рассматриваем m рёбер, отсюда *O(mlogn)*.

Время работы на тестах:



4. Алгоритм из лекции

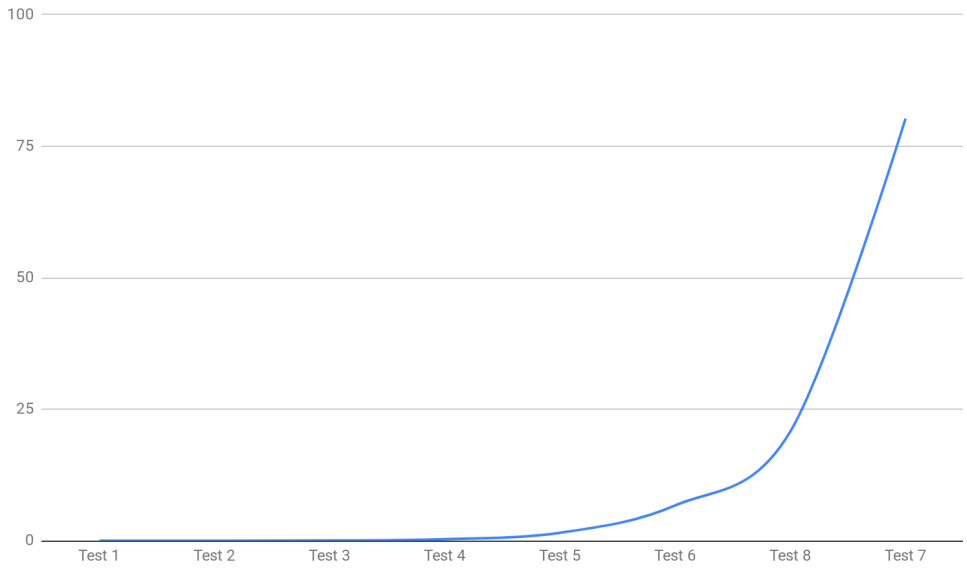
Описание:

Работа алгоритма похожа на сочетание идей Борувки и Прима. Он состоит из нескольких фаз. В начале каждой фазы алгоритм пытается расширить из компонент, полученных на прошлом шаге, остовные деревья, используя алгоритм Прима, но как только на *i-й* фазе количество элементов в куче превосходит k_i, расширение остовного дерева из данной компоненты прекращается, все посещенные компоненты объявляются обработанными. Также расширение прекращается после присоединения обработанной компоненты, потому что в таком случае, количество элементов в куче превзойдёт k_i. С использованием фибоначчиевой кучи и константы $k_i = 2^i$, удается получить хорошую ассимптотику.

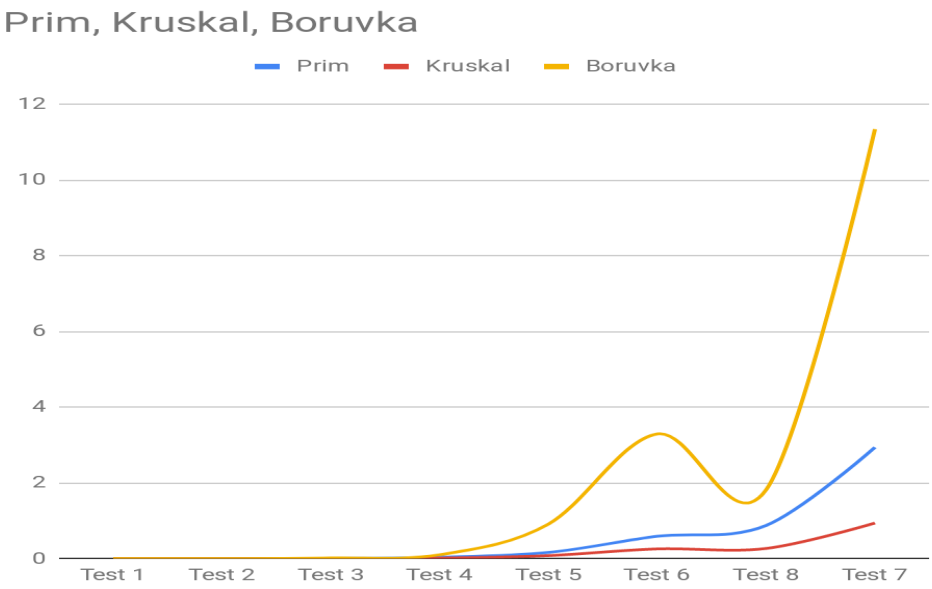
Сложность алгоритма:

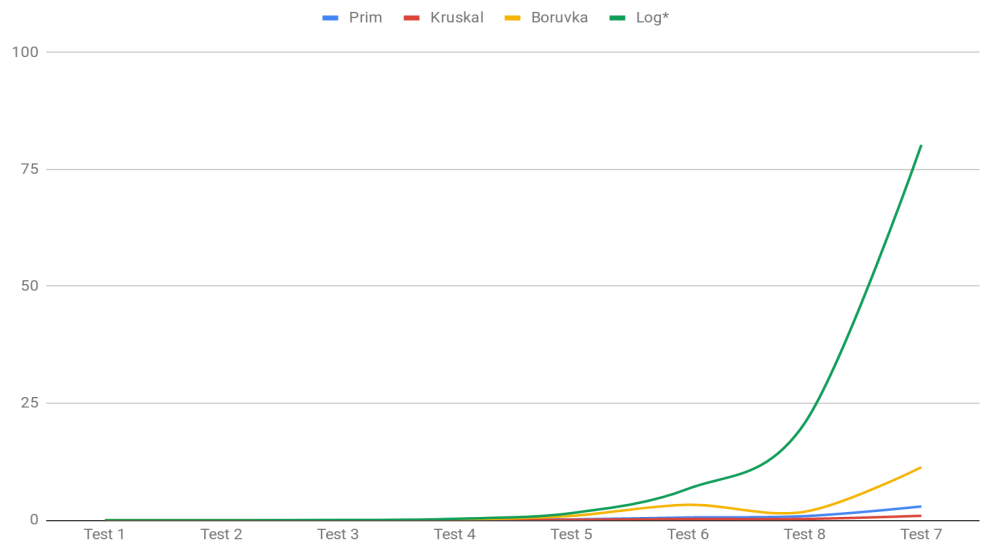
Фаза алгоритма работает за $O(m + t \log k_i)$, где t – количество компонент. После каждой фазы в графе остается $\leq 2m / k_i$ независимых компонент, мы хотим чтобы каждая фаза выполнялась за $O(m)$, отсюда получаем уравнение: $m * \log k_i / k_{i-1} = m$. Отсюда $k_{i+1} = 2^{k\_i}$. Поэтому ассимптотика - $O(m \log^* n)$

Время работы на тестах:



Сводные графики





Заключение

На данный момент самым быстрым является алгоритм Крускала, потому что он использует только сортировку и СНМ. Остальные же производят значительные, но не влияющие на ассимптотику вычисления, из-за чего их практическая ценность не высока.