МИНОБРНАУКИ РФ

ФГБОУ ВПО Тверской государственный технический университет

Кафедра «Программное обеспечение».

Дисциплина «Алгоритмы и структуры данных».

Тема: «Минимальное остовное дерево».

Выполнил: студент группы

ПИН 17.05

Иванов Р.В

Проверил:

Мальков А.А

Тверь 2019

Оглавление

[Введение. 3](#_Toc10028473)

[Алгоритм Краскала. 3](#_Toc10028474)

[Краскал наглядно. 4](#_Toc10028475)

[Алгоритм Прима. 11](#_Toc10028476)

[Алгоритм Прим наглядно 12](#_Toc10028477)

[Алгоритм Борувки. 12](#_Toc10028478)

[Алгоритм Борувки наглядно. 13](#_Toc10028479)

[Реализация проекта 14](#_Toc10028480)

[Сравнение алгоритмов по времени. 15](#_Toc10028481)

[Тестирование 16](#_Toc10028482)

[Разные построения дерева одним алгоритмом. 17](#_Toc10028483)

[Борувки 18](#_Toc10028484)

[Прима 18](#_Toc10028485)

[Краскала 18](#_Toc10028486)

[Заключение 19](#_Toc10028487)

[Источники 20](#_Toc10028488)

# Введение.

Цель: Ознакомится с алгоритмами минимального остовного дерева и реализовать их.

Задача: Реализовать Windows Form проект на c#.

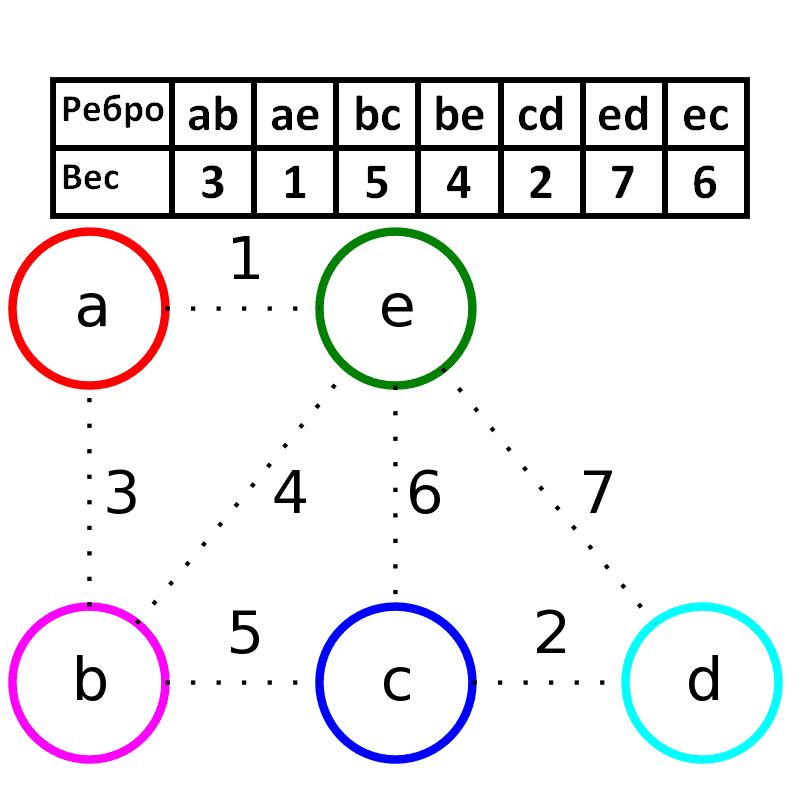
Реализовано: Алгоритмы Прима и Крускала. Написаны тесты под них.

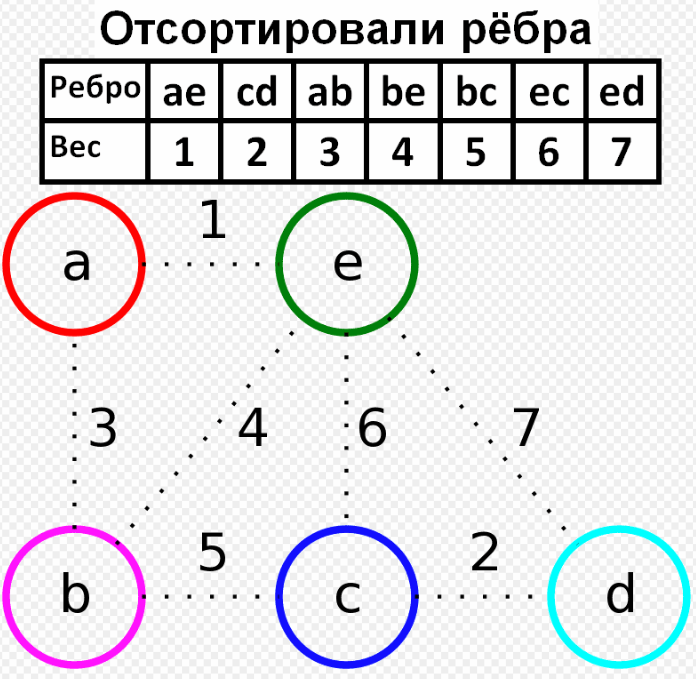
# Алгоритм Краскала.

Вначале 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. Подграф данного графа, содержащий все его вершины и найденное множество рёбер, является его остовным лесом минимального веса.

До начала работы алгоритма необходимо отсортировать рёбра по весу, это требует O(E × log(E)) времени. После чего компоненты связности удобно хранить в виде системы непересекающихся множеств. Все операции в таком случае займут O(E × α(E, V)), где α — функция, обратная к функции Аккермана. Поскольку для любых практических задач α(E, V) < 5, то можно принять её за константу, таким образом общее время работы алгоритма Краскала можно принять за O(E).

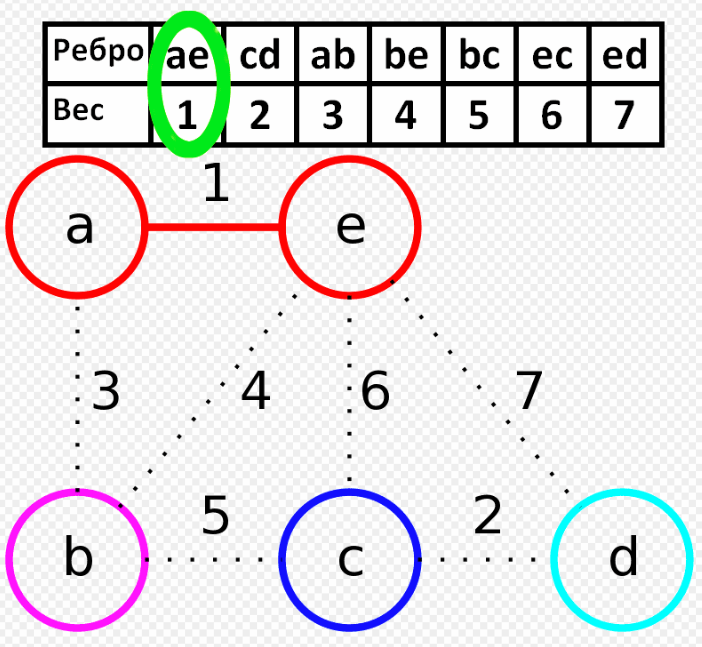
## Краскал наглядно.





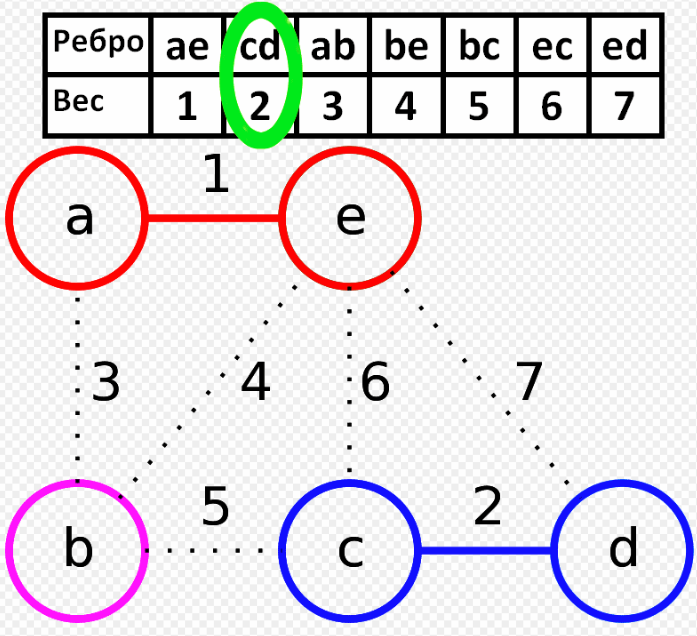
Первое ребро, которое будет рассмотрено — **ae**, так как его вес минимальный.

Добавим его к ответу, так как его концы соединяют вершины из разных множеств (a — красное и e — зелёное). Объединим красное и зелёное множество в одно (красное), так как теперь они соединены ребром.



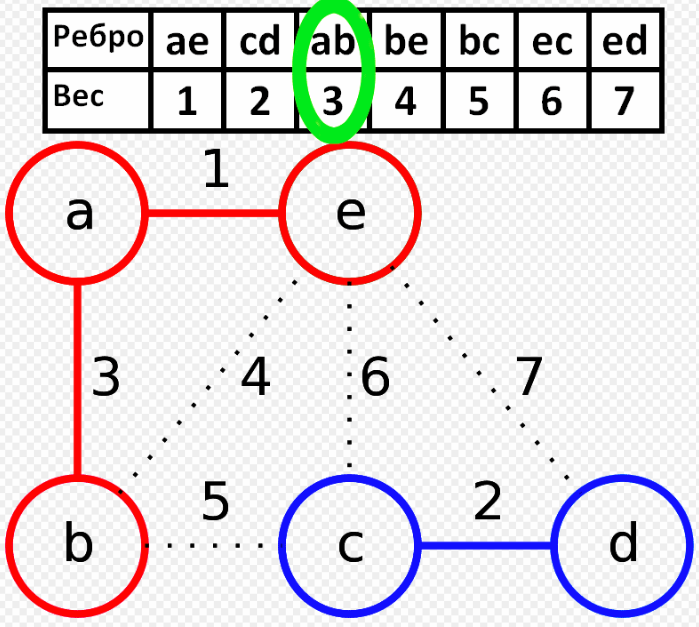
Рассмотрим следующие ребро — cd.

Добавим его к ответу, так как его концы соединяют вершины из разных множеств (c — синее и d — голубое). Объединим синее и голубое множество в одно (синее), так как теперь они соединены ребром.



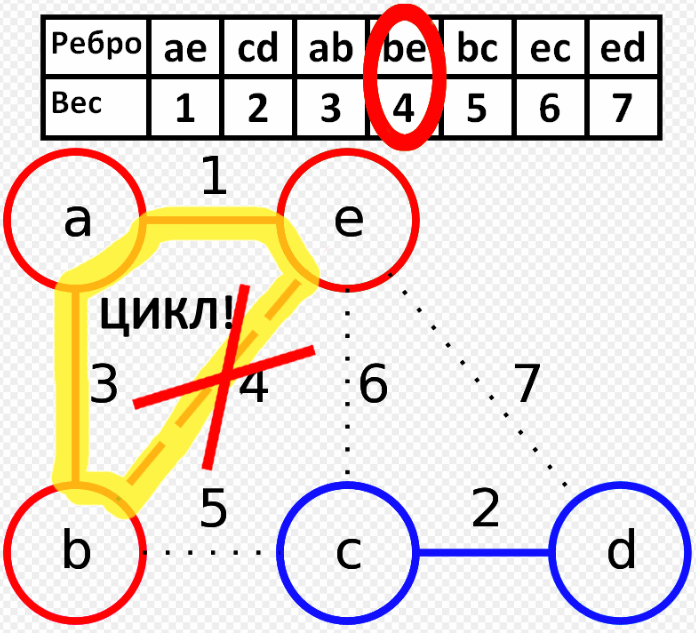
Дальше рассмотрим ребро ab.

Добавим его к ответу, так как его концы соединяют вершины из разных множеств (a — красное и b — розовое). Объединим красное и розовое множество в одно (красное), так как теперь они соединены ребром.

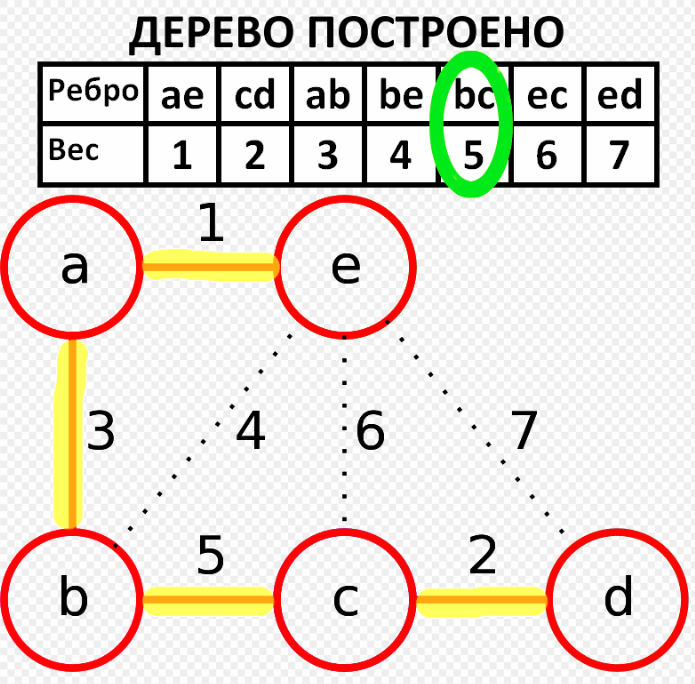


Рассмотрим следующие ребро — be.

Оно соединяет вершины из одного множества, поэтому перейдём к следующему ребру bc Добавим его к ответу, так как его концы соединяют вершины из разных множеств (b — красное и c — синее). Объединим красное и синее множество в одно (красное), так как теперь они соединены ребром.



Рёбра ec и ed соединяют вершины из одного множества, поэтому после их просмотра они не будут добавлены в ответ Всё рёбра были рассмотрены, поэтому алгоритм завершает работу. Полученный граф — минимальное остовное дерево

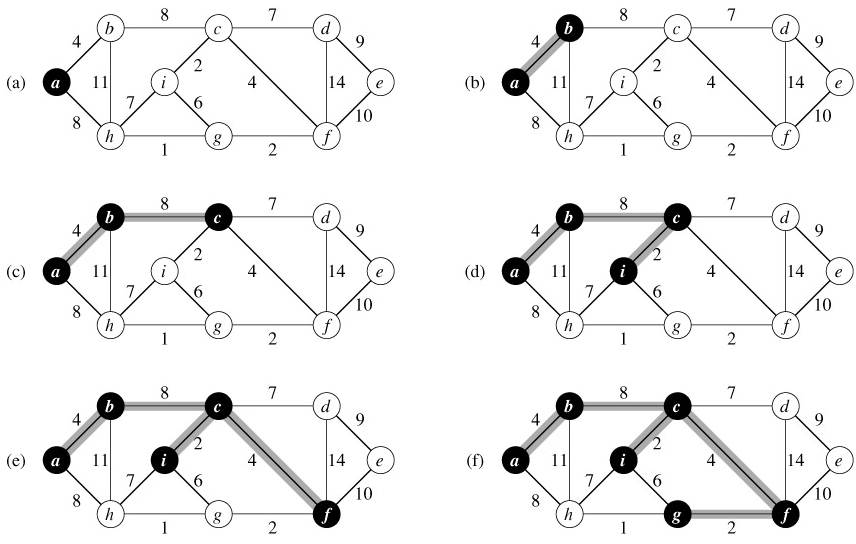


# Алгоритм Прима.

Сам алгоритм имеет очень простой вид. Искомый минимальный остов строится постепенно, добавлением в него рёбер по одному. Изначально остов полагается состоящим из единственной вершины (её можно выбрать произвольно). Затем выбирается ребро минимального веса, исходящее из этой вершины, и добавляется в минимальный остов. После этого остов содержит уже две вершины, и теперь ищется и добавляется ребро минимального веса, имеющее один конец в одной из двух выбранных вершин, а другой — наоборот, во всех остальных, кроме этих двух. И так далее, т.е. всякий раз ищется минимальное по весу ребро, один конец которого — уже взятая в остов вершина, а другой конец — ещё не взятая, и это ребро добавляется в остов (если таких рёбер несколько, можно взять любое). Этот процесс повторяется до тех пор, пока остов не станет содержать все вершины (или, что то же самое, ребро).

В итоге будет построен остов, являющийся минимальным. Если граф был изначально не связен, то остов найден не будет (количество выбранных рёбер останется меньше ).

## Алгоритм Прим наглядно



# Алгоритм Борувки.

Работа алгоритма состоит из нескольких итераций, каждая из которых заключается в последовательном добавлении ребер к основному лесу графа, пока лес не превратится в дерево, то есть лес, состоящий из одного связного компонента.

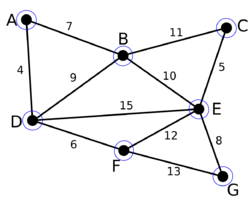
В псевдокоде алгоритм можно описать как:

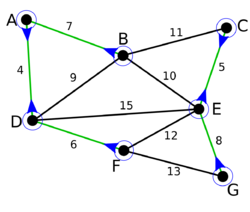
1. Изначально, пусть {\displaystyle T}T — пустое множество рёбер (представляющее собой остовный лес, в который каждая вершина входит в качестве отдельного дерева).
2. Пока {\displaystyle T}T не является деревом (что эквивалентно условию: пока число рёбер в {\displaystyle T}T меньше, чем {\displaystyle V-1}V-1, где {\displaystyle V}V— число вершин в графе):

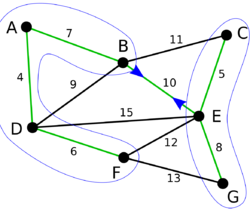
* Для каждой компоненты связности (то есть, дерева в остовном лесе) в подграфе с рёбрами {\displaystyle T}T, найдём самое дешёвое ребро, связывающее эту компоненту с некоторой *другой* компонентой связности. (Предполагается, что веса рёбер различны, или как-то дополнительно упорядочены так, чтобы всегда можно было найти единственное ребро с минимальным весом).
* Добавим все найденные рёбра в множество {\displaystyle T}T.

1. Полученное множество рёбер {\displaystyle T}T является минимальным остовным деревом входного графа.

## Алгоритм Борувки наглядно.

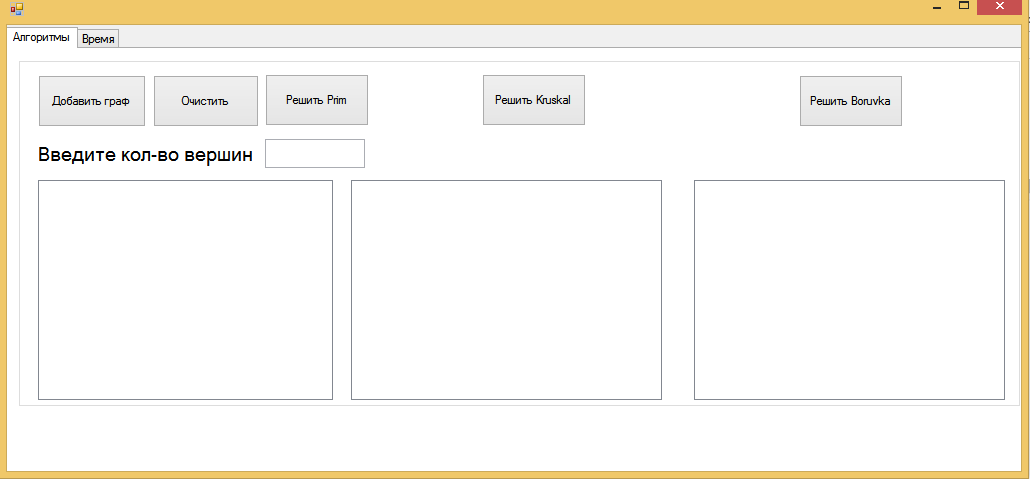
 Начальный граф GG. Каждая вершина является компонентой (синие окружности).

На первой итерации внешнего цикла для каждой компоненты были добавлены минимальные сопряженные ребра. Некоторые ребра добавлены несколько раз (ADAD и CECE). Осталось две компоненты. 

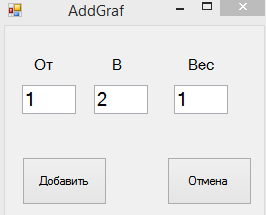


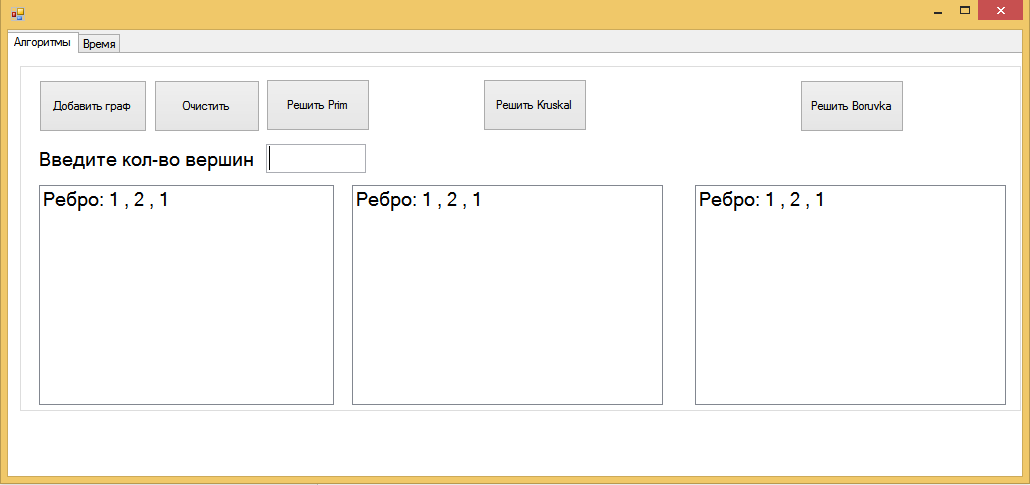
На последней итерации внешнего цикла было добавлено минимальное ребро, соединяющее две оставшиеся компоненты (ребро BEBE). Осталась одна компонента. Минимальное остовное дерево графа GG построено.

# Реализация проекта

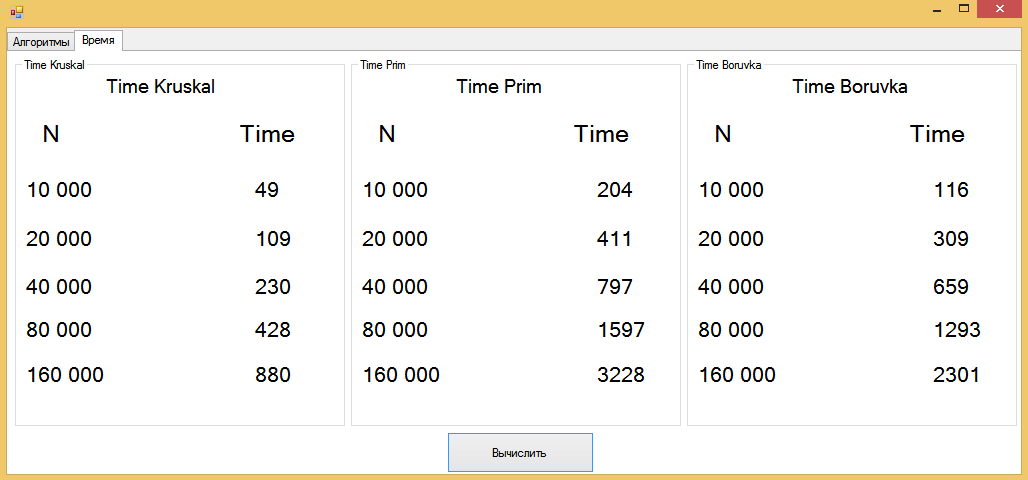


Как происходит добавление графа





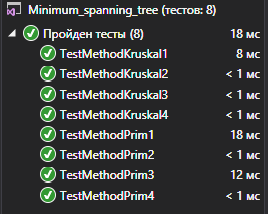
## Сравнение алгоритмов по времени.

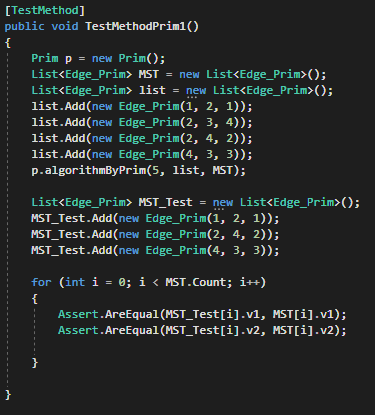


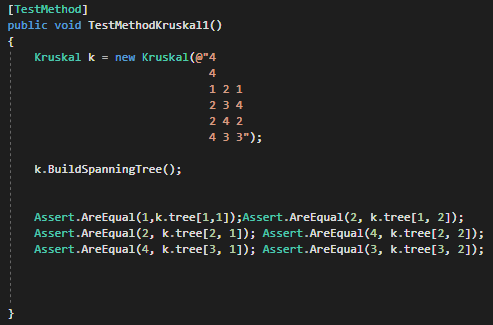
## Тестирование

Для автоматического тестирования добавлены в проект UnitTestProject.

Результаты тестов:



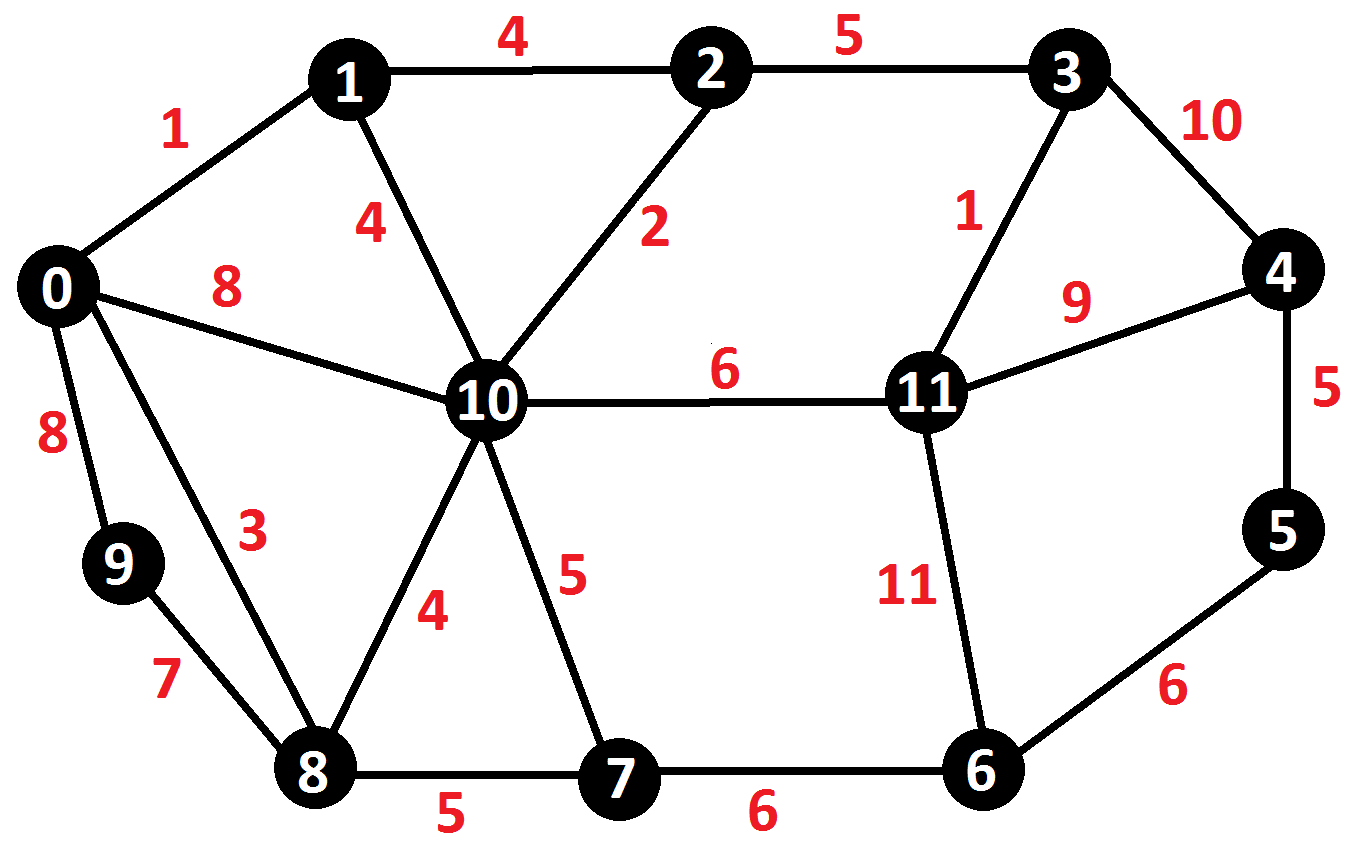




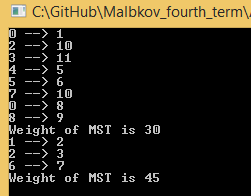
Все тесты успешно пройдены.

# Разные построения дерева одним алгоритмом.

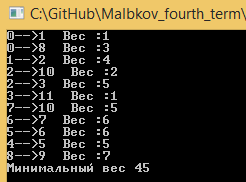
На вход подается такое дерево



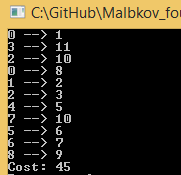
## Борувки



## Прима

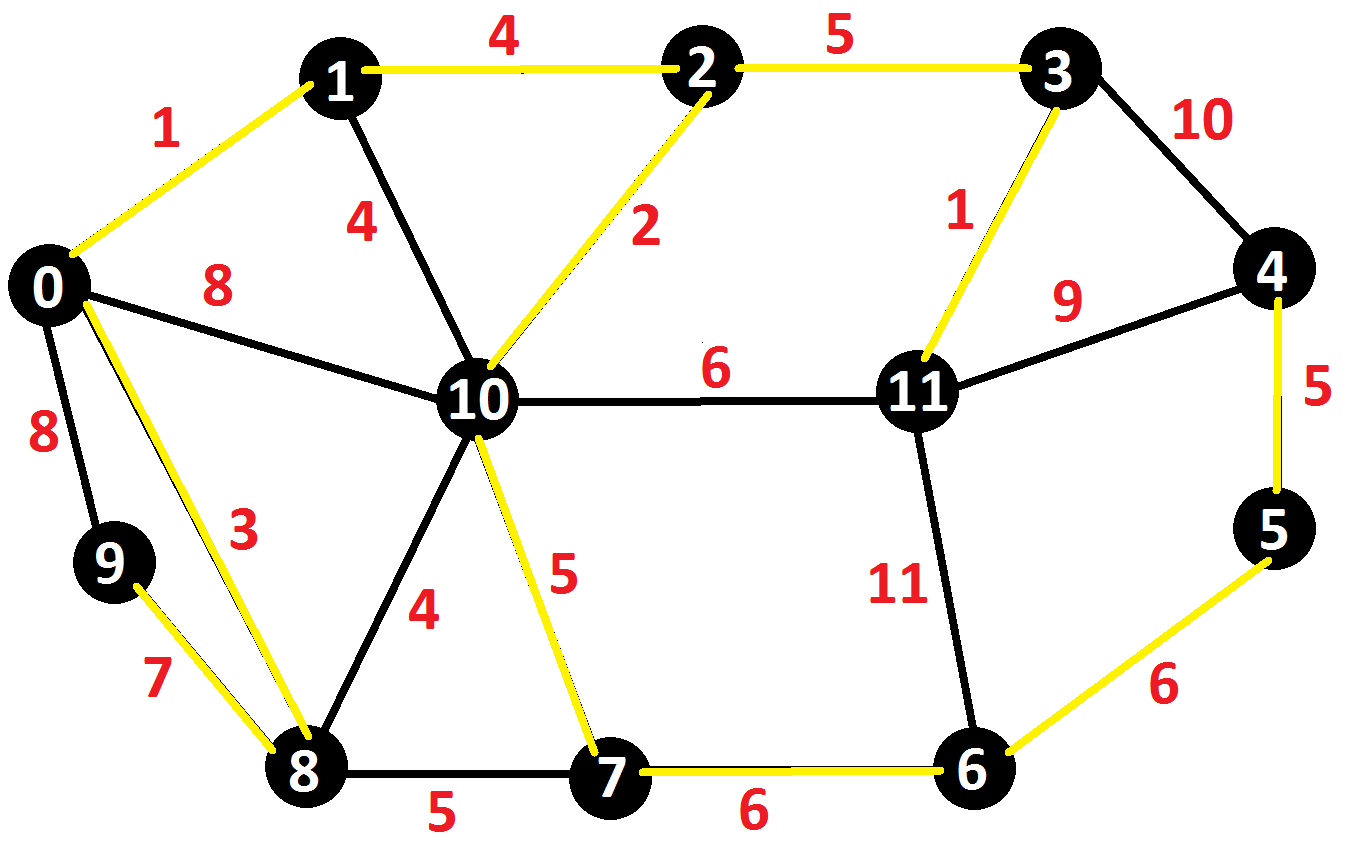


## Краскала



Как мы видим каждый алгоритм строит дерево по своему, но как итог мы все равно получаем минимальное остовное дерево.

Финальный вид дерева



# Заключение

В данной работе я реализовал 3 алгоритма минимального остовного дерева.

Проект был успешно протестирован.

# Источники

<https://ru.wikipedia.org/wiki/Алгоритм_Краскала>

<https://ru.wikipedia.org/wiki/Алгоритм_Прима>

https://neerc.ifmo.ru/wiki/index.php?title=Алгоритм\_Борувки