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](#_Toc9287049)

[Алгоритм Краскала. 3](#_Toc9287050)

[Краскал наглядно. 4](#_Toc9287051)

[Алгоритм Прима. 10](#_Toc9287052)

[Прим наглядно 11](#_Toc9287053)

[Реализация проекта 11](#_Toc9287054)

[Тестирование 12](#_Toc9287055)

[Заключение 14](#_Toc9287056)

[Источники 14](#_Toc9287057)

# 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.

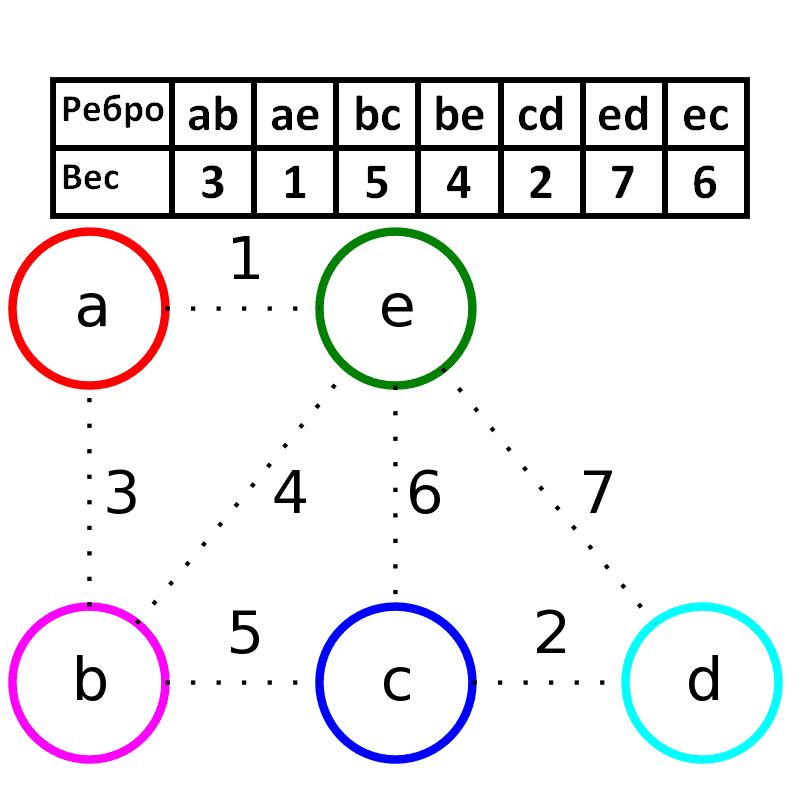
Алгоритм Краскала в псевдокоде:

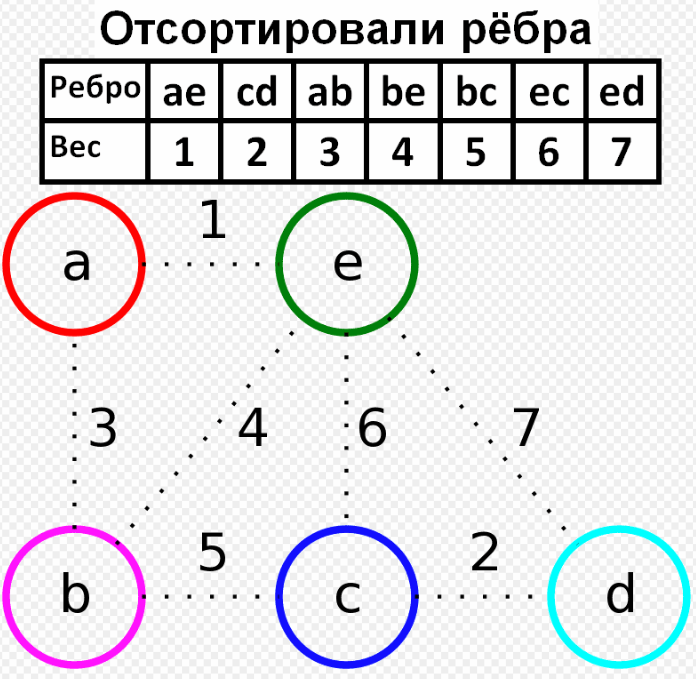
|  |  |
| --- | --- |
| 1  2  3  4  5  6 | Kruskal's algorithm:      sort the edges of G **in** increasing order by length      keep a subgraph S of G, initially empty  **for** each edge e **in** sorted order  **if** the endpoints of e are disconnected **in** S          add e to S  **return** S |

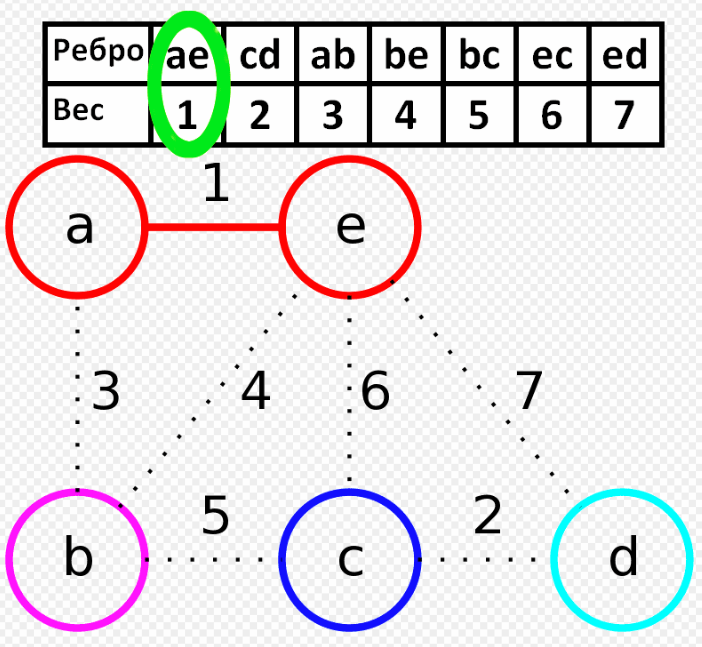
Вначале текущее множество рёбер устанавливается пустым. Затем, пока это возможно, проводится следующая операция: из всех рёбер, добавление которых к уже имеющемуся множеству не вызовет появление в нём цикла, выбирается ребро минимального веса и добавляется к уже имеющемуся множеству. Когда таких рёбер больше нет, алгоритм завершён. Подграф данного графа, содержащий все его вершины и найденное множество рёбер, является его остовным лесом минимального веса.

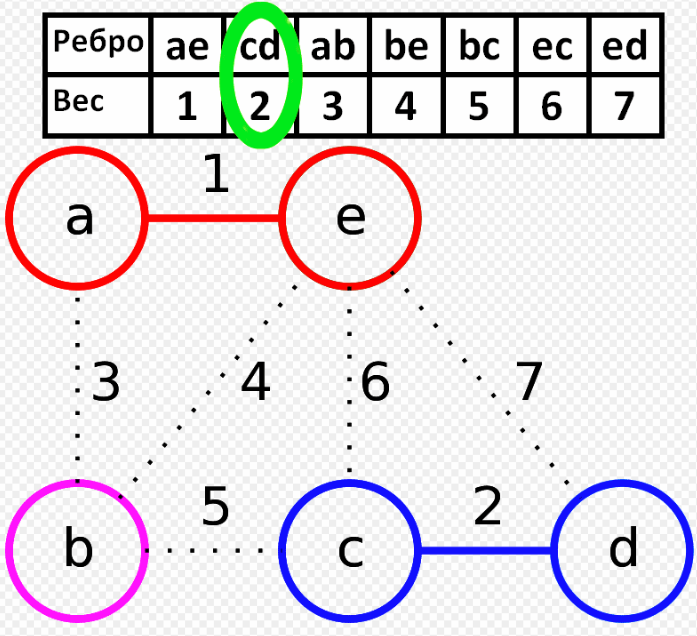
До начала работы алгоритма необходимо отсортировать рёбра по весу, это требует O(E × log(E)) времени. После чего компоненты связности удобно хранить в виде системы непересекающихся множеств. Все операции в таком случае займут O(E × α(E, V)), где α — функция, обратная к функции Аккермана. Поскольку для 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).

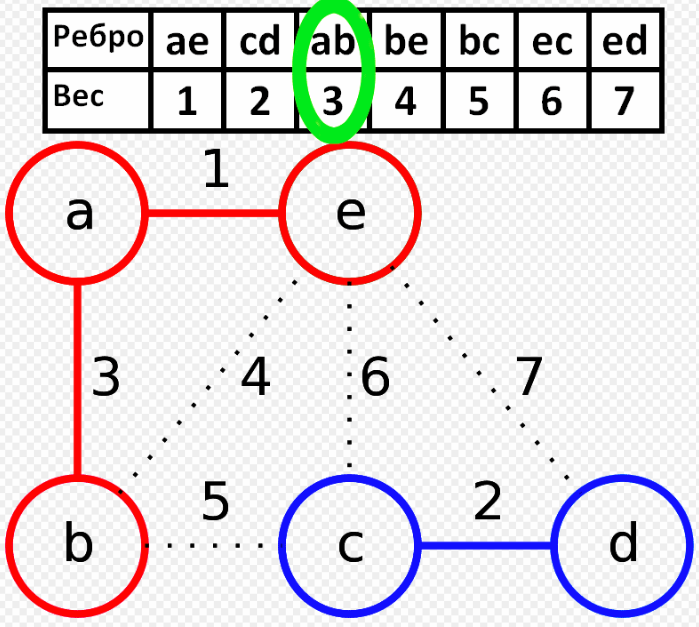
## Краскал наглядно.

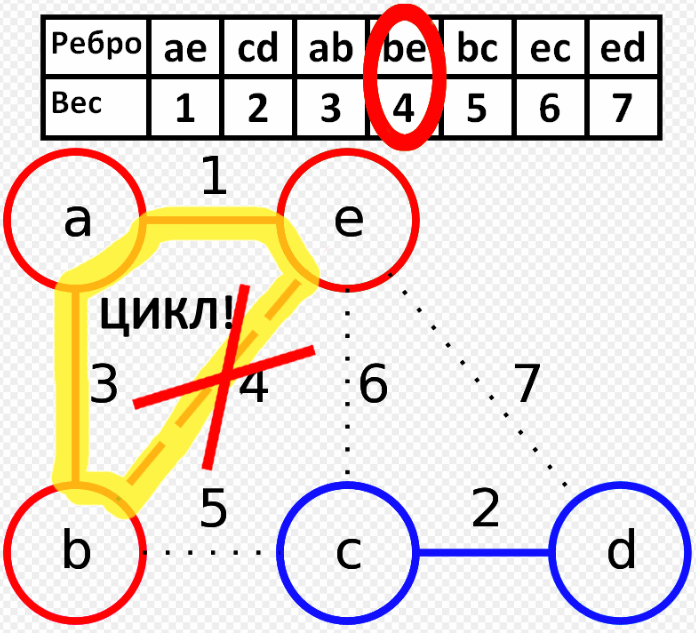


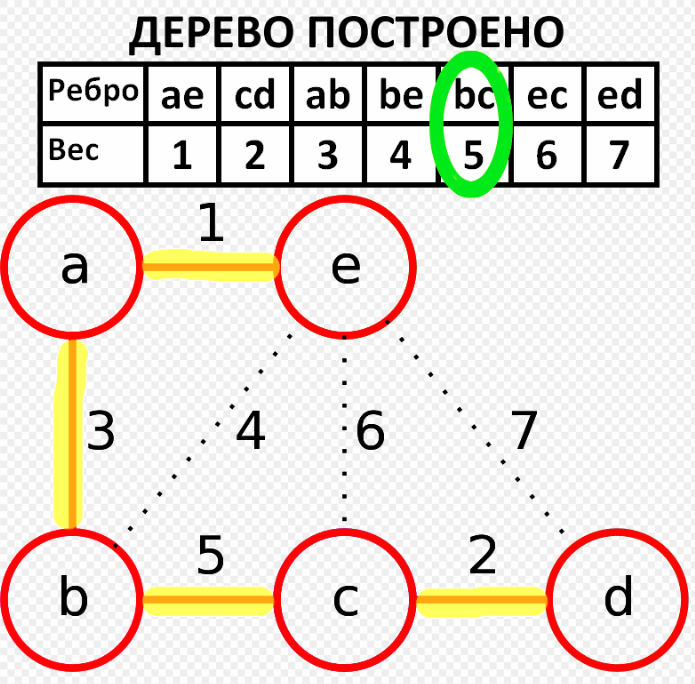










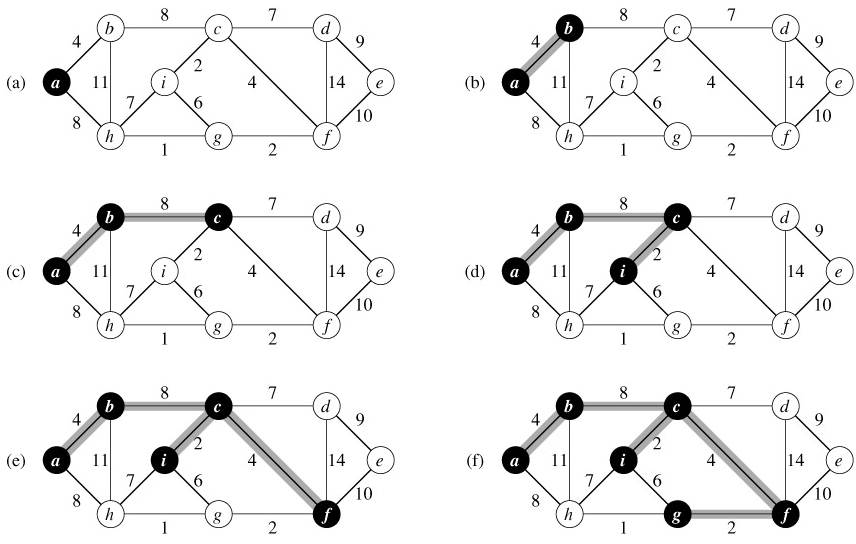


# Алгоритм Прима.

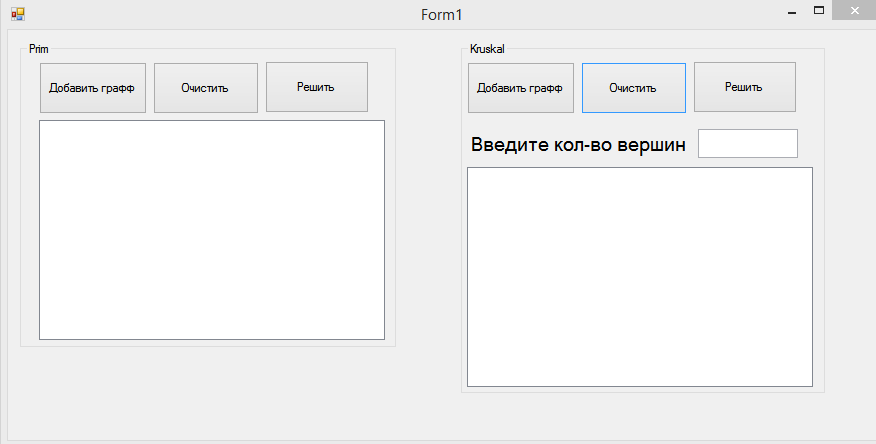
Сам алгоритм имеет очень простой вид. Искомый минимальный остов строится постепенно, добавлением в него рёбер по одному. Изначально остов полагается состоящим из единственной вершины (её можно выбрать произвольно). Затем выбирается ребро минимального веса, исходящее из этой вершины, и добавляется в минимальный 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 ).

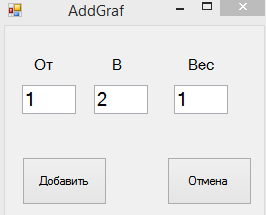
## Прим наглядно

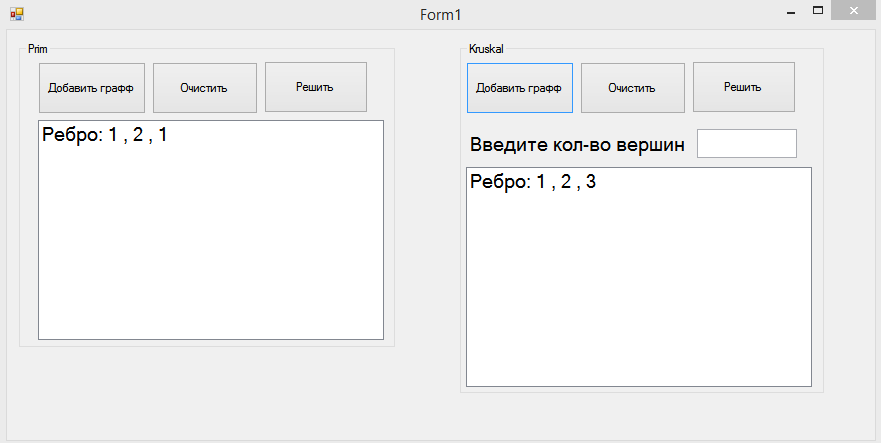


# Реализация проекта



Как происходит добавление графа

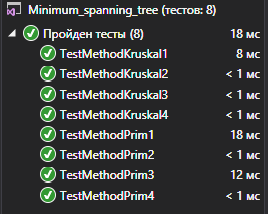


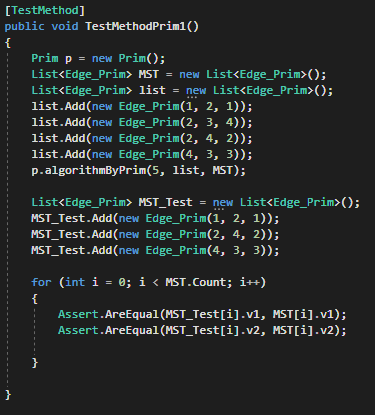


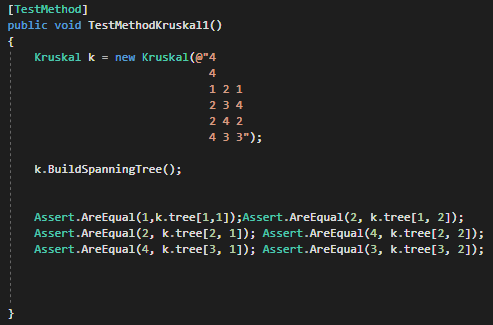
## Тестирование

Для автоматического тестирования добавлены в проект UnitTestProject.

Результаты тестов:







Все тесты успешно пройдены.

# Заключение

В данной работе я реализовал 2 алгоритма минимального остовного дерева.

Проект был успешно протестирован.

# Источники

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

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