Приложение 1

**Министерство образования и науки Республики Казахстан**

**Astana IT University**

**НАУЧНЫЙ ПРОЕКТ**

на тему:

**«Применение алгоритмов поиска минимального остовного дерева (МОД)»**

Выполнили: Серикбай А. А, Оралбай М.

Группа: MCS-2301

Проверила: ассоц. проф. Молдахметова З. Н.

Астана

2025

Приложение 2

**СОДЕРЖАНИЕ**

**ВВЕДЕНИЕ**….……………………………………………………………………

**ГЛАВА1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ МИНИМАЛЬНОГО ОСТОВНОГО ДЕРЕВА…………………………………………………………**

[1.1. Основные понятия теории графов…………………………………](#_Toc483955736)...............

[1.2. Остовное дерево. Задача построения остовного дерева минимального](#_Toc483955737) веса……..…………………………………………………………………………..

[1.3. А](#_Toc483955736)лгоритм Прима……………………………………………………………...

[1.4. А](#_Toc483955737)лгоритм Крускала…………………………………………………………..

**ГЛАВА2. РЕАЛИЗАЦИЯ И АНАЛИЗ АЛГОРИТМОВ**…………………....

2.1. **Инструменты и технологии…………………………………………………**

2.2. **Реализация кода………………………….…………………………...………**

2.3. Тестирование алгоритмов**.**……………………………………………..

2.4. **Сравнительный анализ.**………………………………………………………

**ЗАКЛЮЧЕНИЕ**….………………………………………………………..……...

**СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**…..……………...…….…......

**ПРИЛОЖЕНИЕ**…………………………………………………………...……..

**Введение**

## **Актуальность темы исследования:**

Актуальность и практическая значимость предлагаемой работы заключается в том, что в большинстве областей возникает необходимость поисков эффективных способов планирования сложных процессов и проектов, которые в итоге можно реализовать с помощью теорий графов. С помощью теории графов можно минимизировать затраты в ходе производства, а далее – принимать оптимальные управленческие решения. И в теории графов часто возникает задача поиска минимального остовного дерева (МОД). И сложно определить по получившемуся графу какой алгоритм решит эту задачу оптимально. Примеры таких задач:

- **Инфраструктура:** Оптимизация прокладки дорог, трубопроводов, кабелей и других сетей.

- **Компьютерные сети:** Минимизация затрат на подключение серверов и маршрутизаторов.

- **Энергетика:** Построение минимальных сетей энергоснабжения.

- **Генетика и биоинформатика:** Анализ филогенетических деревьев.

## **Объект исследования:**

Выбор оптимального алгоритма для поиска минимального остовного дерева для определенного графа.

## **Предмет исследования:**

Алгоритмы Прима и Крускала для поиска минимального остовного дерева.

**Цель:**

Изучить алгоритмы Прима и Крускала для поиска минимального остовного дерева и исследовать их эффективность на различных графах.

**Задачи работы:**

1. Проанализировать основные теоретические аспекты задачи минимального остовного дерева;

2. Реализовать алгоритмы Крускала и Прима на языке Python;

3. Разработать алгоритм для генерации тестовых графов и их визуализации;

4. Сделать сбор данных;

5. Из полученной информации сравнить эффективность алгоритмов;

## **Методы исследования:**

* Метод сравнения;
* **Экспериментальный метод;**
* **Математическое моделирование;**

## **Теоретические основы работы:**

Работа базируется на теории графов, алгоритмах прима и крускала для поиска минимального остовного дерева

## **Практическая значимость:**

Результаты работы могут быть использованы в прикладных задачах для минимизации затрат. Примером такой задачи может быть транспортное планирование где необходимо найти такой путь чтобы общая длина пути и стоимость была минимальной. Такая же задача возникает в проектирование компьютерных сетей, а именно в протоколе STP, который устраняет петли в сети выбирая при этом лучшие петли по скорости соеденения.

## **Структура работы:**

Работа состоит из введения, двух глав, заключения, списка литературы и приложения.

В введении мы написали характеристики работы которые определяют тему нашей работы и ее актуальность, объект и предмет исследования, цели и задачи, которые необходимо рассмотреть, методы исследования, теоретические основы и практическую значимость нашей работы.

В первой главе рассмотрена теоритическая часть поиска минимального остовного дерева

Во второй главе произведен анализ алгоритмов поиска минимального остовного дерева и их реализация.

В заключении сделаны выводы на основе полученной информации.

В списке литературы приведены источники информации и ссылки на них.

В приложении представлены графики, диаграмы и прочие рисунки использованные в исследовании.

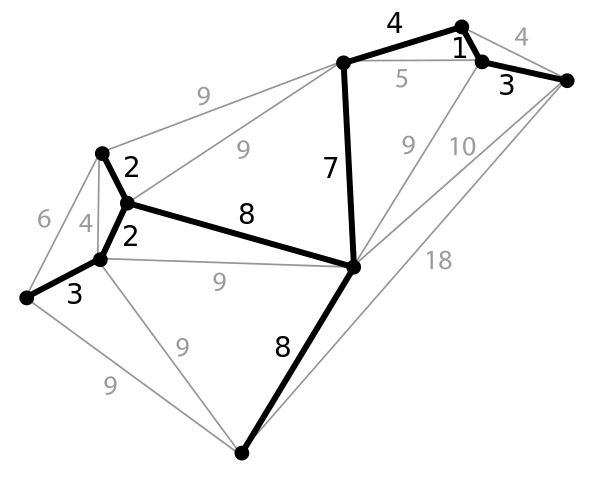
****ГЛАВА 1.** ТЕОРЕТИЧЕСКИЕ ОСНОВЫ МИНИМАЛЬНОГО ОСТОВНОГО ДЕРЕВА**

**1.1. Основные понятия теории графов**  
Теория графов — это раздел дискретной математики, изучающий структуры, состоящие из вершин и рёбер, соединяющих их. Граф определяется как G = (V, E), где V — множество вершин, а E — множество рёбер, соединяющих вершины. Графы бывают ориентированными и неориентированными, взвешенными и невзвешенными, связными и несвязными.

**1.2. Задача построения минимального остовного дерева**  
Остовное дерево (spanning tree) — это подграф, содержащий все вершины исходного графа, но не содержащий циклов и имеющий минимальное возможное количество рёбер. Минимальное остовное дерево (МОД) — это остовное дерево с минимальной суммой весов рёбер. Построение МОД — это важная задача в теории графов, находящая применение в сетевом планировании, маршрутизации и инфраструктурных проектах.

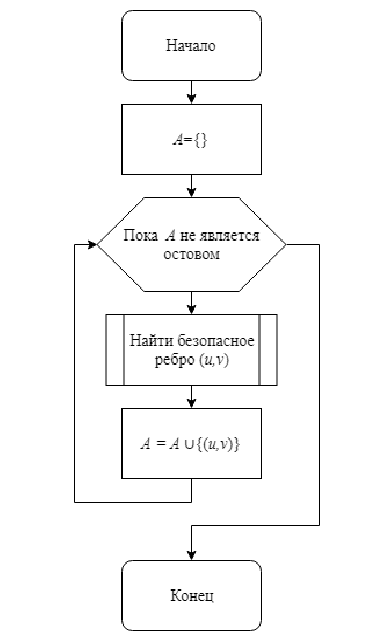
Например представим какой то город где есть различные районы соедененные дорогами. Каждая дорога имеет свою протяженность, и по ней могут ездить машины. Для эффективной работы транспортной системы города требуется чтобы не было замкнутых маршрутов(циклов), которые могут привести к избыточному пробегу машин. Наша задача состоит в том чтобы каждый район был доступен для машин, и суммарная протяженность дорог по которым ездят машины была минимальной. Такую задачу можно описать как задачу поиска минимального остовного дерева, где узлы — это районы города, а ребра — это дороги между районами с весом, равным протяженности дороги.

Задача в общем виде выглядит так. Дан связанный, взвешенный,  
неориентированный граф G = (V, E), где V — множество вершин, а E — множество рёбер, соединяющих вершины. Для каждого ребра задан определенный вес задающий стоимость соедениние двух вершин. Задача состоит в нахождении такого подграфа который соединяет все вершины и общий вес которой минимален(см Рисунок 1)



**Рисунок 1 – Минимальное остовное дерево связанного графа**

**Искомый остов получается из подграфа А. Граф А строится постепенно и  
изначально является пустым. На каждом шаге алгоритм добавляет одно новое ребро и так пока А не станет остовом. Подграф А графа G называется *безопасным* еслионо является подграфом какого-то минимального остого.**



**Рисунок 2 – Блок-схема алгоритма построения минимального остовного  
дерева в общем виде**

**1.3. Алгоритм Прима**  
Алгоритм Прима является жадным методом поиска минимального остовного дерева(поскольку на каждом шаге он добавляет ребро минимального веса). Кормен и соавторы определяют алгоритм Прима следующим образом: "Алгоритм Прима строит минимальное остовное дерево, начиная с произвольной вершины и добавляя на каждом шаге минимальное по весу ребро" (Кормен, 2006, с. 870). Основные шаги алгоритма:

1. Выбираем произвольную начальную вершину.
2. Инициализируем множество А остовного дерева (изначально пустое).
3. Добавляем минимальное ребро, соединяющее вершину с остальной частью графа к множеству.
4. Повторяем процесс, пока все V-1 ребер не будут включены в множество.

Приведем пример работы алгоритма. На рисунке 3 изображен граф G

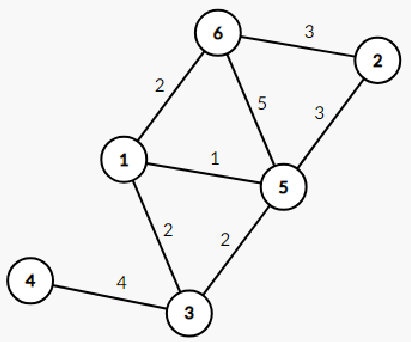


Рисунок 3 – Граф на котором будет запущен алгоритм Прима

Выберем произвольную вершину r = 6. Просмотрим все ребра исходящие из r  
среди таких минимальный вес имеет ребро (6, 1). Добавим его в остов  
(рисунок 4).

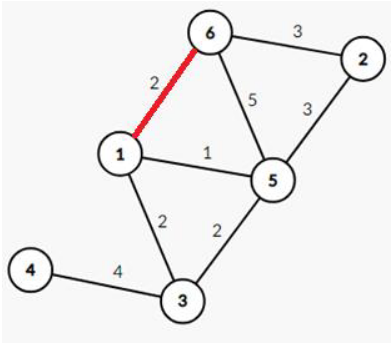


Рисунок 4 – Подграф А после первого шага алгоритма Прима

Теперь у нас есть две вершины 1 и 6. Рассмотрим все инцидентный им  
ребра. Среди них наименьший вес имеет ребро (1, 5). Добавим его в остов  
(рисунок 5)

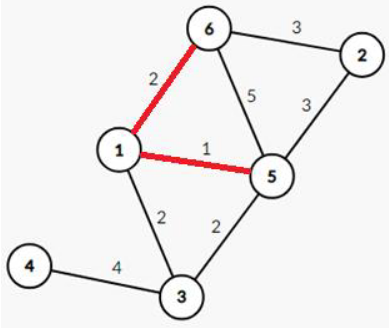


Рисунок 5 – Подграф А после второго шага алгоритма Прима

Аналогично продолжим на каждой итерации выбирать безопасное ребро  
минимального веса. В итоге получим остов, изображенный на рисунке 6.

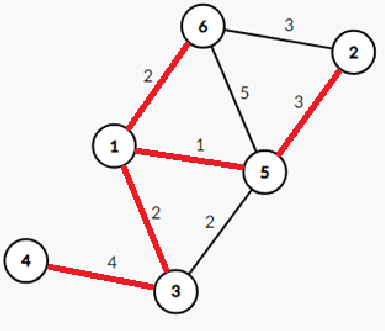


Рисунок 6 – Финальный подграф А(минимальный остов), полученный алгоритма Прима

Если это будет разреженный граф асимптотика поиска минимального ребра займет O(E\*log2(V) операции. Иначе если это будет полный граф это займет O(V^2) операции.

**1.4. Алгоритм Крускала**  
Алгоритм Крускала — еще один жадный алгоритм(поскольку на каждом шаге он добавляет ребро минимального веса), который сортирует все рёбра по весу и последовательно добавляет их в остовное дерево, если они не создают цикла. Основные шаги алгоритма:

1. Сортируем рёбра графа по возрастанию веса.
2. Инициализируем множество А остовного дерева (изначально пустое).
3. Последовательно добавляем рёбра, если они не образуют цикла.
4. Повторяем процесс, пока все V-1 ребер не будут включены в множество.

Приведем пример работы алгоритма. На рисунке 7 изображен граф G

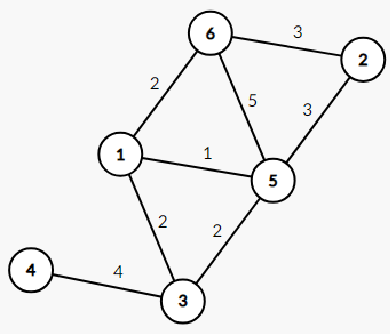
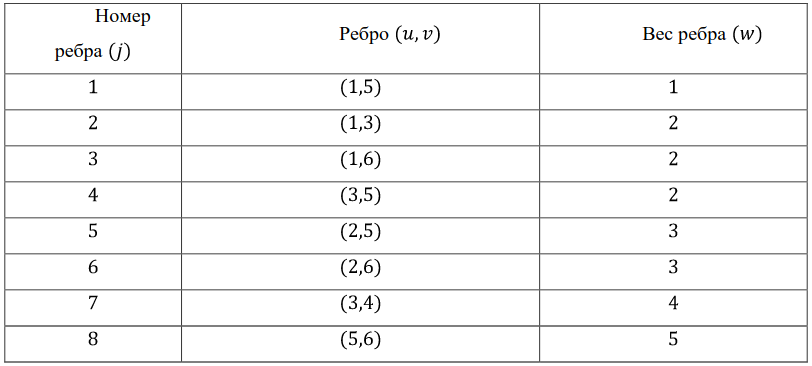


Рисунок 7 – Граф на котором будет запущен алгоритм Крускала

Выпишем все ребра в порядке возрастания их весов. Результат приведен в  
таблице 1.

Таблица 1 – Ребра графа в порядке возрастания весов

Для начала берем ребро с минимальным весом(1, 5) и добавляем в наш остов (рисунок 8).

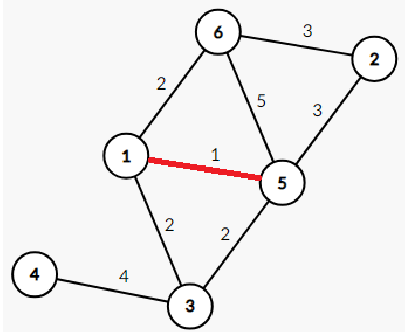


Рисунок 8 – Дерево после первого шага алгоритма Крускала

Такой же логикой берем ребра (1, 3),(1, 6) и добавляем в наш остов (рисунок 9).

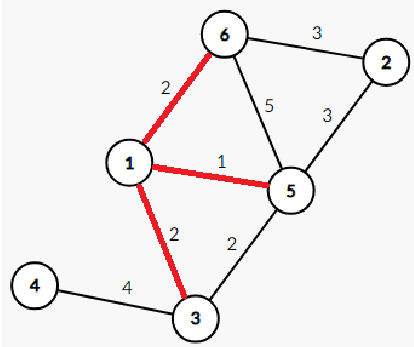


Рисунок 9 – Дерево после третьего шага алгоритма Крускала

Пропускаем ребро (3, 5) (так как тогда образуется цикл из ребер 1, 2, 4) и проверяем ребро (2, 5). Текущее ребро не создает противоречий, значит мы включаем его в остов (рисунок 10)

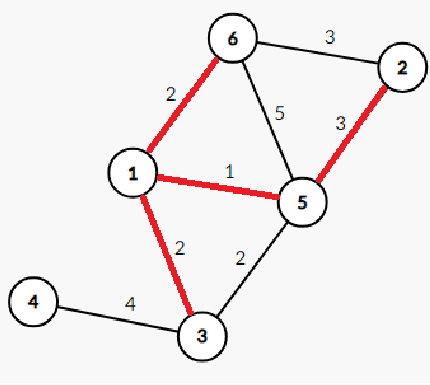


Рисунок 10 – Дерево после четвертого шага алгоритма Крускала

Ребро (2, 6) нам не подходит так как будет образован цикл из ребер 1, 3, 5, 6. Ребро (3, 4) мы включаем в минимальный остов (рисунок 11).

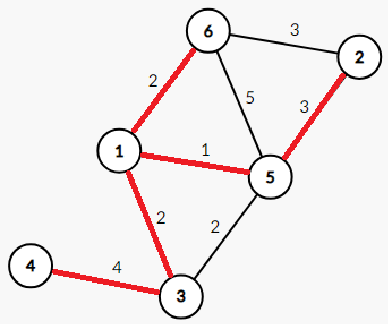


Рисунок 11 – Остов на на пятом шаге алгоритма Крускала

Так как мы включили n-1 ребро, алгоритм можно завершить. Минимальный остов найден.

Асимптотика алгоритма Крускала сильно зависит от используемой  
сортировки и при использовании быстрой сортировки она равна O(E\*log2(E)) операциям.

**ГЛАВА 2. РЕАЛИЗАЦИЯ И АНАЛИЗ АЛГОРИТМОВ  **цитирование:** Алгоритм Крускала показывает эффективность в разреженных графах (Кормен, 2006) **2.1. Инструменты и технологии****

**Для реализации алгоритмов минимального остовного дерева использовался язык программирования Python и библиотека NetworkX, которая предоставляет удобные инструменты для работы с графами. Также использовались matplotlib для визуализации графов и numpy для статистической обработки результатов.**

**Используемые библиотеки:**

**- **time** – для измерения времени выполнения алгоритмов.**

**- **random** – для генерации случайных рёбер графа.**

**- **networkx** – для работы с графами и реализации алгоритмов Прима и Краскала.**

**- **matplotlib.pyplot** – для построения графов и визуализации результатов.**

**- **numpy** – для вычисления среднего времени выполнения алгоритмов.**

****2.2. Реализация кода****

**Аргументы:**

**-** *num\_vertices* **– количество вершин в графе.**

**-** *edge\_range* **– диапазон возможных весов рёбер.**

**-** *edge\_probability***– вероятность создания рёбер между вершинами.**

**Программа состоит из следующих функций:**

*generate\_graph(num\_vertices, edge\_range, edge\_probability)* **- генерирует случайный неориентированный граф, соединяя вершины рёбрами с весами из заданного диапазона. Пользователь задаёт вероятность появления рёбер** *edge\_probability***, что влияет на плотность графа.**

*prim\_mst(edges, num\_vertices)***- реализует алгоритм Прима для нахождения минимального остовного дерева.**

*kruskal\_mst(edges, num\_vertices)* ***-* реализует алгоритм Краскала для нахождения минимального остовного дерева.**

*save\_results(num\_vertices, edge\_range, runs, edge\_probability, prim\_time, kruskal\_time)***- сохраняет результаты работы программы в файл** mst\_results.txt**.**

*draw\_graph(graph, title)* ***-* визуализирует граф, отображая вершины и веса рёбер. Расположение вершин остаётся неизменным для удобства сравнения.**

****2.3. Тестирование алгоритмов****

**Тестирование проводилось на графах с различным количеством вершин, диапазонами весов рёбер и вероятностями появления рёбер. Оценивалось среднее время выполнения алгоритмов.**

****2.4. Сравнительный анализ****

**Для оценки эффективности алгоритмов выполнялось сравнение их времени выполнения на одинаковых входных данных. Результаты визуализировались в виде графиков. В итоге мы пришли к выводу что в большинстве случаев алгоритм Прима быстрее алгоритма Краскала. Алгоритм Краскала может приближаться по скорости к Приму, но только в условиях очень разреженных графов. Среднее время работы алгоритмов сохраняется в файле** mst\_results.txt **для дальнейшего анализа.**

**Таблица тестов:**

Таблица 2 – Таблица тестов

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения научного проекта была достигнута основная цель работы - изучение алгоритмов Прима и Крускала для поиска минимального остовного дерева и исследована их эффективность на различных графах. Проведен анализ теоретических аспектов задачи минимального остовного дерева, включая изучение основных понятий теории графов и принципов работы алгоритмов Прима и Крускала. Осуществлена практическая реализация алгоритмов на языке Python с использованием библиотек NetworkX для работы с графами, matplotlib для визуализации и numpy для статистической обработки результатов. Разработан инструментарий для генерации тестовых графов с различными параметрами (количество вершин, диапазон весов рёбер, вероятность появления рёбер), что позволило провести комплексное тестирование алгоритмов. Произведен сбор и анализ данных о производительности алгоритмов при различных входных параметрах. Выполнено сравнение эффективности алгоритмов, которое показало, что: aлгоритм Прима демонстрирует более высокую производительность в большинстве случаев, алгоритм Крускала приближается по эффективности к алгоритму Прима только на разреженных графах, результаты тестирования подтверждают теоретические оценки сложности алгоритмов. Практическая значимость работы подтверждается возможностью применения реализованных алгоритмов для решения реальных задач оптимизации в различных областях, таких как проектирование компьютерных сетей, транспортное планирование и построение инфраструктуры. Разработанный программный код может быть использован как основа для дальнейших исследований и практического применения в задачах построения минимальных остовных деревьев. Результаты исследования могут служить отправной точкой для дальнейшего изучения оптимизации алгоритмов поиска минимального остовного дерева и их адаптации под конкретные прикладные задачи.

**СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

1. Кормен Т. Х. и др. Часть VI. Алгоритмы для работы с графами // Алгоритмы: построение и анализ = Introduction to Algorithms. — 2-е изд. — М.: Вильямс, 2006. – 1296 c.

(<https://evil-teacher.on.fleek.co/books/tp/Cormen_Algorithms.pdf>)

1. Лутц М. Программирование на Python, том I, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с.

(<https://disnetern.ru/wp-content/uploads/2016/11/Mark_Lutts_-_Programmirovanie_na_Python_4-e_izd_1_TOM.pdf>)

1. Лутц М. Программирование на Python, том II, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с.

(<https://disnetern.ru/wp-content/uploads/2016/11/Mark_Lutts_-_Programmirovanie_na_Python_4-e_izd_2_TOM.pdf>)

1. Зыков А.А. Основы теории графов. – М.: Наука, 2007. – 384 c.\

(<https://vk.com/doc409016625_554042700?hash=eHqSYZiudMh9sMvJTW4VBaQzG8V0jvq1LARLWjNT3qc&api=1&no_preview=1>)

1. Оре О. Теория графов. — 2-е изд.— М.: Наука, Главная редакция физико-математической литературы, 2010. – 336 c.

(<https://vk.com/doc409016625_554042835?hash=7X3rDMqPUVtzyzPf60cdiwK0vuM5PleGJjvDlsL1ywT&api=1&no_preview=1>)

1. Уилсон Р. Введение в теорию графов. Пер. с анг. 2007. – 208 c.

(<https://vk.com/doc409016625_554042882?hash=vA5ZtuImJZp7zKv7BJgAwrEvRXr6dKYFUT5Htdy1D5k&api=1&no_preview=1>)

1. Харари Ф. Теория графов / Пер.с англ. и предисл. В. П. Козырева. Под ред. Г. П. Гаврилова. Изд. 2-е. - М.: Едиториал УРСС, 2003. – 296 c.

(<https://vk.com/doc409016625_554043363?hash=QJvcIN4LUqb1q9exfPAt5WTp91ElHUR3lsndG1Nfczs&api=1&no_preview=1>)

<https://cyberleninka.ru/article/n/postroenie-minimalnogo-ostovnogo-dereva-algoritmom-boruvki-programmnaya-realizatsiya/viewer>

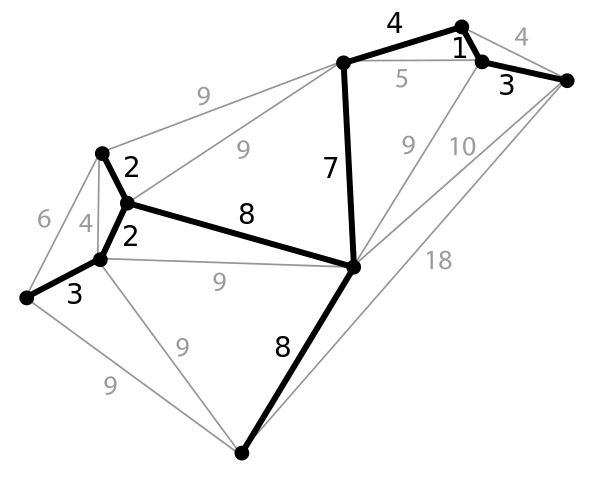
<https://cyberleninka.ru/article/n/nahozhdenie-ostovnogo-dereva-minimalnogo-vesa-s-primeneniem-algoritma-kraskala-i-algoritma-prima/viewer>

[https://dspace.tltsu.ru/bitstream/123456789/9734/1/%D0%A1%D0%B0%D0%B1%D0%B8%D1%80%D0%BE%D0%B2%20%D0%94.%D0%91.\_%D0%9F%D0%9C%D0%98%D0%B1-1501.pdf](https://dspace.tltsu.ru/bitstream/123456789/9734/1/Сабиров%20Д.Б._ПМИб-1501.pdf)

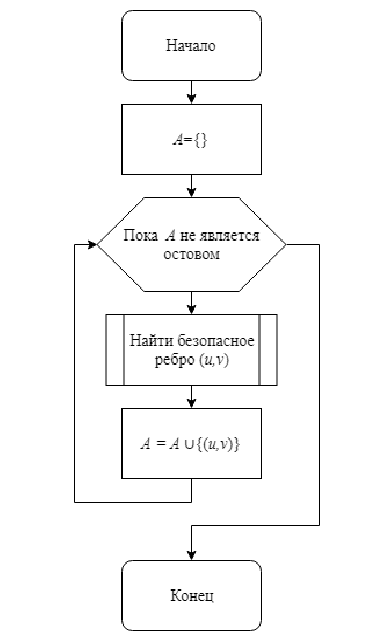
<https://brestprog.by/topics/mst/>

<https://algorithmica.org/ru/mst>

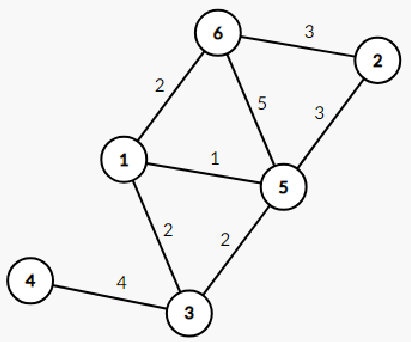
**ПРИЛОЖЕНИЯ**



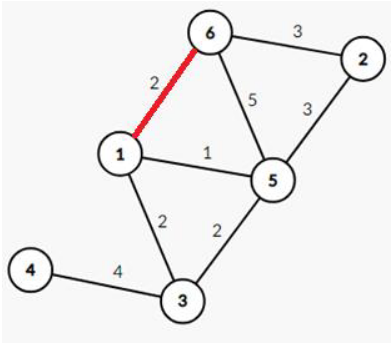
**Рисунок 1 – Минимальное остовное дерево связанного графа**

****

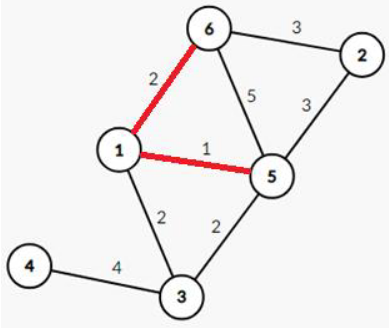
**Рисунок 2 – Блок-схема алгоритма построения минимального остовного  
дерева в общем виде**



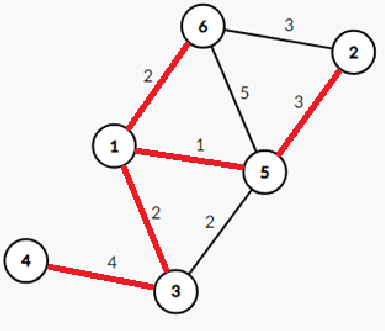
**Рисунок 3 – Граф на котором будет запущен алгоритм Прима**



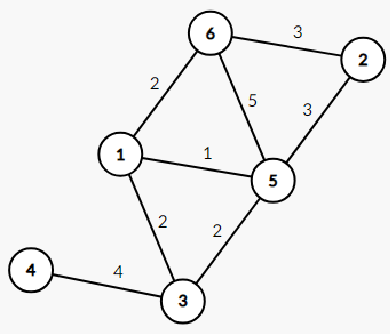
**Рисунок 4 – Подграф А после первого шага алгоритма Прима**



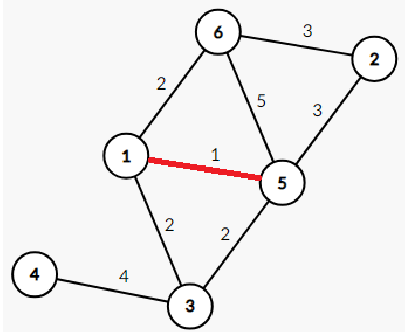
**Рисунок 5 – Подграф А после второго шага алгоритма Прима**



**Рисунок 6 – Финальный подграф А(минимальный остов), полученный алгоритма Прима**



**Рисунок 7 – Граф на котором будет запущен алгоритм Крускала**



**Рисунок 8 – Дерево после первого шага алгоритма Крускала**

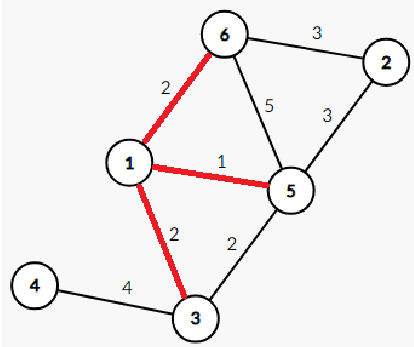


Рисунок 9 – Дерево после третьего шага алгоритма Крускала

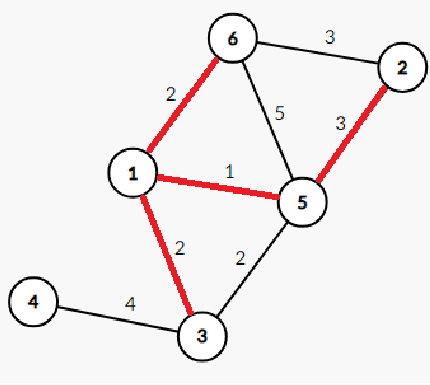


Рисунок 10 – Дерево после четвертого шага алгоритма Крускала

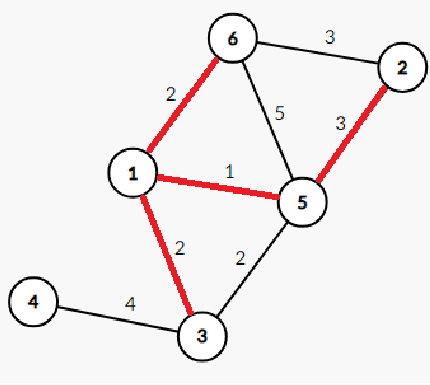


Рисунок 11 – Остов на на пятом шаге алгоритма Крускала

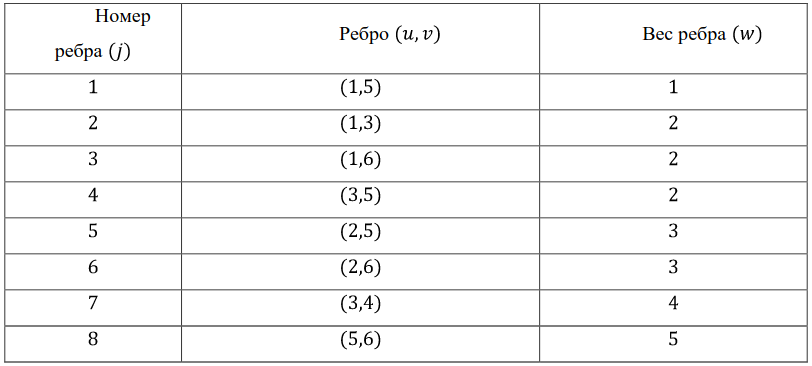
Таблица 1 – Ребра графа в порядке возрастания весов

Таблица 2 – Таблица тестов