Приложение 1

**Министерство образования и науки Республики Казахстан**

**Astana IT University**

**НАУЧНЫЙ ПРОЕКТ**

на тему:

**«Применение алгоритмов построения 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:???(Под конец)**

* Теоретический анализ литературы по теме минимального остовного дерева.
* Экспериментальное исследование работы алгоритмов на различных графах.
* Сравнительный анализ временной сложности алгоритмов.
* Визуализация результатов с помощью программных инструментов.

## **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.

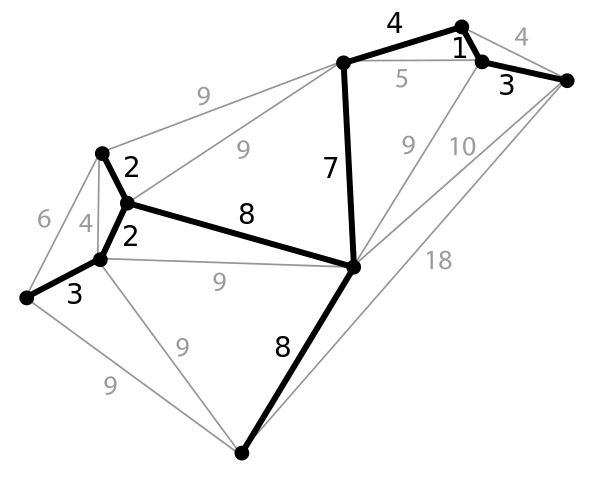
****ГЛАВА1.** ТЕОРЕТИЧЕСКИЕ ОСНОВЫ МИНИМАЛЬНОГО ОСТОВНОГО ДЕРЕВА**

**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.

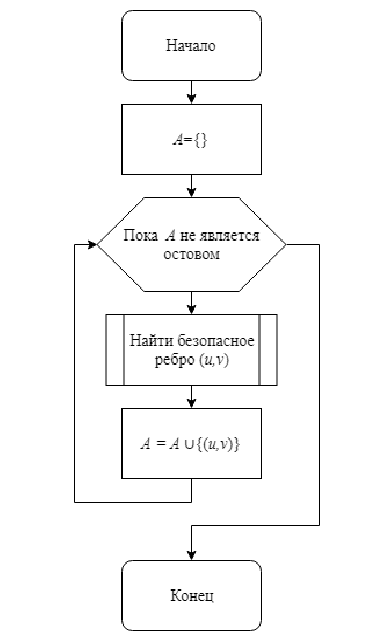
Например представим какой то город где есть различные районы соедененные дорогами. Каждая дорога имеет свою протяженность, и по ней могут ездить машины. Для эффективной работы транспортной системы города требуется чтобы не было замкнутых маршрутов(циклов), которые могут привести к избыточному пробегу 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). Он строит остовное дерево, начиная с произвольной вершины, и на каждом шаге добавляет к дереву ближайшую вершину, соединенную минимальным по весу ребром. 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

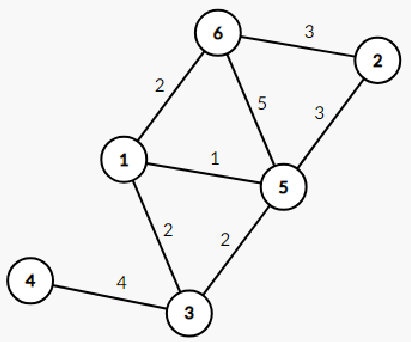


Рисунок 3 – Граф на котором будет запущен алгоритм Прима

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).

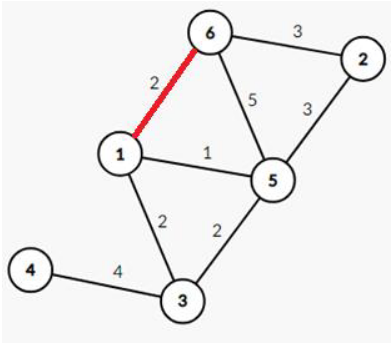


Рисунок 4 – Подграф А после первого шага алгоритма Прима

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)

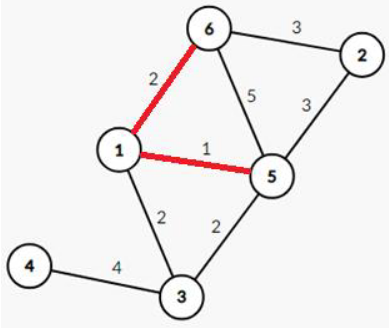


Рисунок 5 – Подграф А после второго шага алгоритма Прима

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

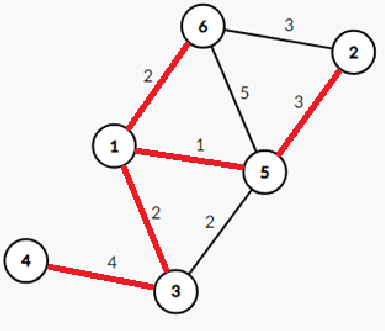


Рисунок 6 – Финальный подграф А(минимальный остов), полученный алгоритма Прима

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

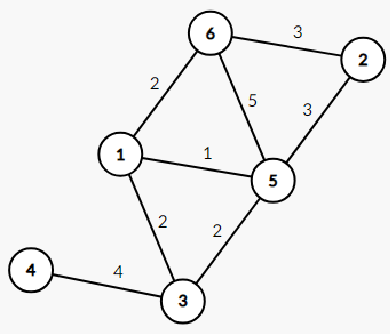
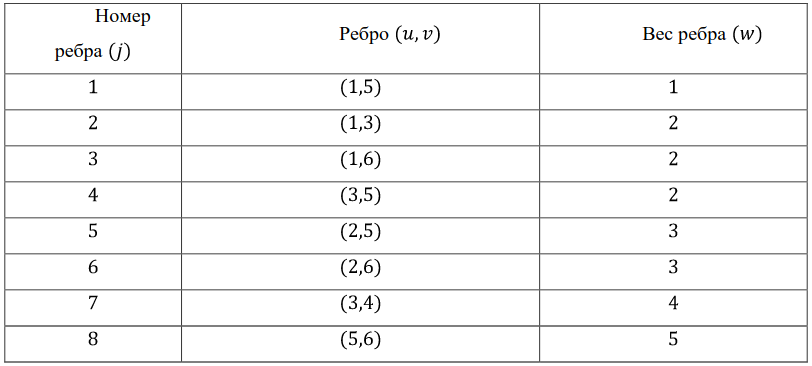


Рисунок 7 – Граф на котором будет запущен алгоритм Крускала

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).

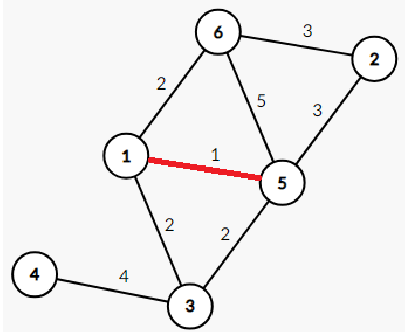


Рисунок 8 – Дерево после первого шага алгоритма Крускала

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

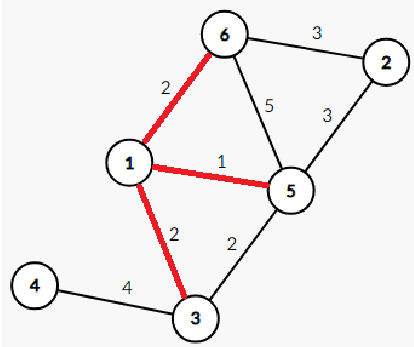


Рисунок 9 – Дерево после третьего шага алгоритма Крускала

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)

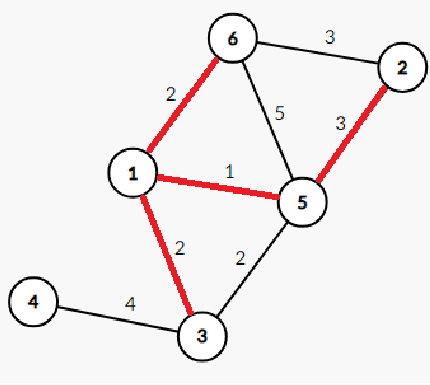


Рисунок 10 – Дерево после четвертого шага алгоритма Крускала

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).

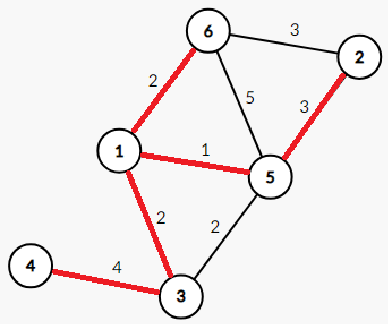


Рисунок 11 – Остов на на пятом шаге алгоритма Крускала

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.

**ГЛАВА2. РЕАЛИЗАЦИЯ И АНАЛИЗ АЛГОРИТМОВ**

#### ****3.1. Инструменты и технологии****

Для реализации алгоритмов были выбраны современные инструменты и технологии, обеспечивающие высокую производительность и удобство разработки. Основным языком программирования стал *Python*, благодаря его простоте, читаемости и наличию множества библиотек для работы с графами и визуализации данных.

Для работы с графами использовалась библиотека *NetworkX*, которая предоставляет широкий набор функций для создания, анализа и визуализации графов. Она поддерживает различные типы графов (ориентированные, неориентированные, взвешенные и т.д.) и включает встроенные алгоритмы для работы с ними.

Для визуализации результатов была задействована библиотека *Matplotlib*, которая позволяет создавать качественные графики и диаграммы. Это особенно полезно для наглядного представления структуры графов и результатов работы алгоритмов.

Среда разработки *VSCode* была выбрана благодаря её удобству, поддержке множества языков программирования, встроенному отладчику и расширениям, которые упрощают процесс написания и тестирования кода.

#### 3.2. Реализация кода

#### 3.2.1. Реализация алгоритмов

Для реализации алгоритмов Прима и Крускала мы использовали встроенные функции библиотеки *NetworkX*, которая предоставляет удобные инструменты для работы с графами. В частности, функции *minimum\_spanning\_tree* с параметрами *algorithm='prim'* и *algorithm='kruskal'* позволяют легко находить минимальное остовное дерево для заданного графа. Однако, чтобы сделать реализацию более гибкой и адаптированной под конкретные задачи, мы расширили функционал, добавив возможность взаимодействия с пользователем и обработки различных сценариев.

#### 3.2.3. Генерация графа

Для удобства тестирования и анализа алгоритмов мы реализовали функцию, которая позволяет пользователю вводить количество вершин n и диапазон [a,b] для случайной генерации весов рёбер. Это позволяет создавать графы с различной структурой и сложностью.

Код включает проверку корректности введённых данных:

* Если пользователь вводит n, a или b меньше нуля, программа выводит сообщение об ошибке и запрашивает повторный ввод.
* Если верхняя граница диапазона b меньше или равна нулю, программа также уведомляет пользователя о некорректном вводе.
* При генерации случайного количества рёбер m программа проверяет, чтобы m не превышало максимально возможное количество рёбер в графе, которое для неориентированного графа без петель и кратных рёбер равно 2n\*(n−1). Если m превышает это значение, программа корректирует его или выводит предупреждение.

#### 3.2.3. Визуализация асимптотики

Для наглядности мы также добавили функцию визуализации графа и найденного минимального остовного дерева с использованием библиотеки *Matplotlib*. Это позволяет пользователю визуально оценить результаты работы алгоритмов.

#### 3.3. Тестирование алгоритмов

Для сравнения алгоритмов поиска минимального оставного дерева

#### 3.4. Сравнительный анализ

Для сравнения производительности алгоритмов Прима и Крускала были проведены тесты на различных типах графов: разреженных (с малым количеством рёбер) и плотных (с большим количеством рёбер).

* *На разреженных графах* алгоритм Крускала показал лучшую производительность. Это связано с тем, что его сложность в основном зависит от сортировки рёбер, которая выполняется за O(ElogE), где E — количество рёбер. В разреженных графах количество рёбер невелико, что делает алгоритм Крускала более эффективным.
* *На плотных графах* алгоритм Прима оказался быстрее. Его сложность составляет O(V2) при реализации с использованием матрицы смежности, что делает его более подходящим для графов с большим количеством рёбер. В таких графах количество рёбер близко к V2, и алгоритм Прима демонстрирует лучшую производительность.

Кроме того, были проведены тесты на графах с различным количеством вершин и рёбер, чтобы оценить масштабируемость алгоритмов. Результаты показали, что оба алгоритма хорошо справляются с задачами среднего и большого размера, но выбор конкретного алгоритма зависит от структуры графа.

**ЗАКЛЮЧЕНИЕ**

В ходе работы были изучены алгоритмы построения минимального остовного дерева, их эффективность и возможности применения в различных областях. Реализация алгоритмов Прима и Крускала в среде Python позволила провести сравнительный анализ их производительности. Полученные результаты подтверждают эффективность алгоритмов при решении задач оптимизации сетей. Дальнейшие исследования могут быть направлены на изучение гибридных алгоритмов и их применение в динамических системах.

**СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

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.(<http://e-maxx.ru/bookz/files/cormen.pdf>)

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

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

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

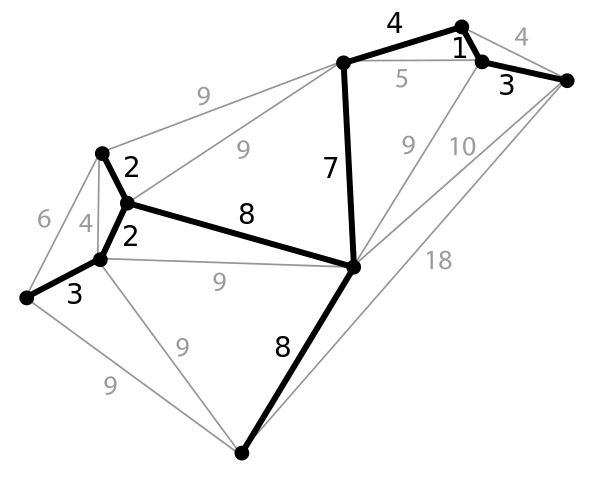
<https://cyberleninka.ru/article/n/nahozhdenie-ostovnogo-dereva-minimalnogo-vesa-s-primeneniem-algoritma-kraskala-i-algoritma-prima/viewer>

[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 Д.Б._ПМИб-1501.pdf)

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

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

**ПРИЛОЖЕНИЯ**

****