Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский химико-технологический университет имени Д.И. Менделеева»

Кафедра информационных компьютерных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5

Выполнил студент группы КС-36 Полковникова Д.Д.

Ссылка на репозиторий: https://github.com/MUCTR-IKT-CPP/Polkovnikova\_CS-36

Приняли: Пысин Максим Дмитриевич

Краснов Дмитрий Олегович

Лобанов Алексей Владимирович

Крашенинников Роман Сергеевич

Дата сдачи: 24.03.2025

Оглавление

[Описание задачи. 2](#_Toc193117184)

[Описание метода/модели. 2](#_Toc193117185)

[Выполнение задачи. 5](#_Toc193117186)

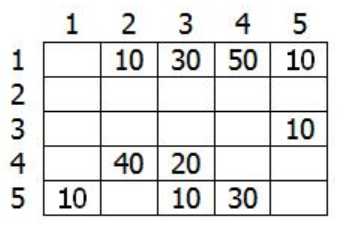
[Заключение. 8](#_Toc193117187)

# Описание задачи.

1. Создайте взвешенный граф, состоящий из [10, 20, 50, 100] вершин.

* Каждая вершина графа связана со случайным количеством вершин, минимум с [3, 4, 10, 20].
* Веса ребер задаются случайным значением от 1 до 20.
* Каждая вершина графа должна быть доступна, т.е. до каждой вершины графа должен обязательно существовать путь до каждой вершины, не обязательно прямой.

1. Выведите получившийся граф в виде матрицы смежности. Пример вывода данных:



1. Для каждого графа требуется провести серию из 5 - 10 тестов, в зависимости от времени затраченного на выполнение одного теста., необходимо:

**Вариант 1**. Найти кратчайшие пути между всеми вершинами графа и их длину с помощью алгоритма Дейкстры.

1. В рамках каждого теста, необходимо замерить потребовавшееся время на выполнение задания из пункта 3 для каждого набора вершин. По окончанию всех тестов необходимо построить график используя полученные замеры времени, где на ось абсцисс (Х) нанести N – количество вершин, а на ось ординат(Y) - значения затраченного времени.

# Описание метода/модели.

**Взвешенным** называют такой граф, каждое ребро которого сопоставимо с каким-либо числовым значением называемым весом графа. Вес ребра в графе может отражать какой-либо параметр в системе, которая этим графом описывается. Например, если есть набор пунктов назначения, соединённых друг с другом дорогами веса на таком графе могут означать длину этих дорог.

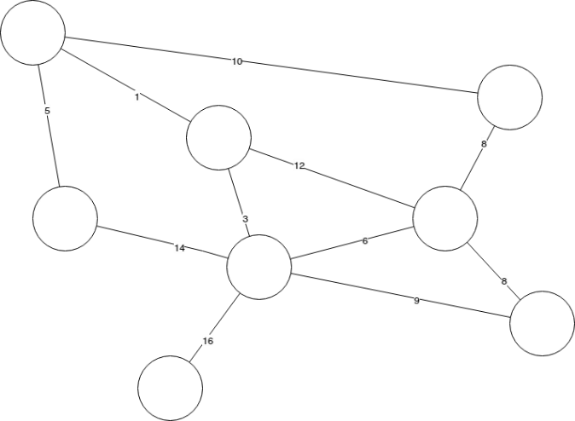


Рисунок 1 – Граф.

**Деревом** называют такой граф в котором не содержатся циклы, так же, для каждой вершины такого графа существует только один маршрут позволяющий добраться до этой вершины.

**Островными деревьями** в графе называются такие подмножества ребер графа, которые создают дерево, содержащее все вершины графа.

Для взвешенного графа существует минимальное островное дерево, это такое дерево пути в котором являются минимальными.

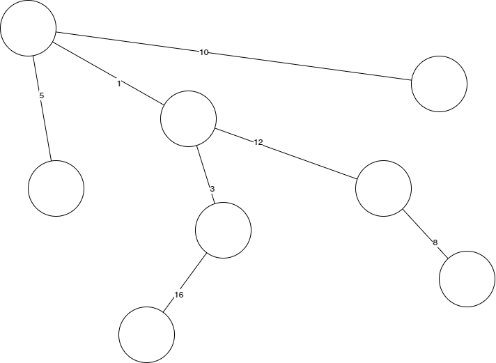
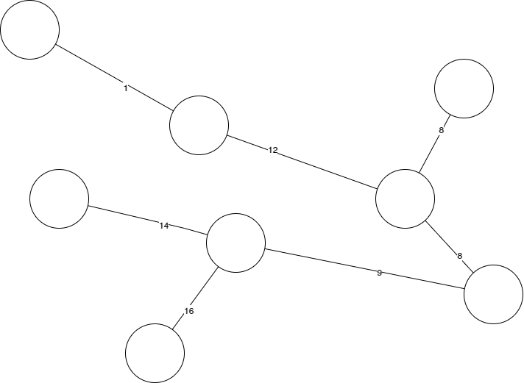
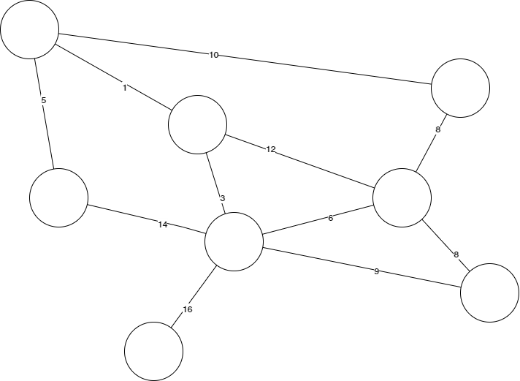


Рисунок 2 – справа налево: Оригинальный граф, Дерево 1, Дерево 2

**Минимальные островные деревья**

Для взвешенного графа существует **минимальное островное дерево,** это такое дерево ребра, в котором суммарно имеют **наименьший** вес из всех существующих деревьев.

Такие деревья позволяют решить задачи соединения множества точек (города, дома, узлы сети) **наименьшим объемом** каких-либо материалов (дорожного полотна, количеством труб, кабеля).

Важно понимать, что для невзвешенного графа любое островное дерево является минимальным, в этом случае его можно найти при помощи алгоритмов обхода в ширину и глубину.

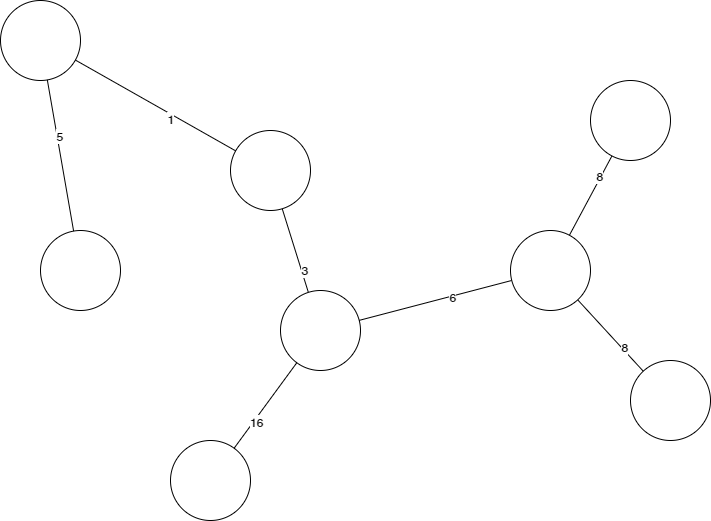
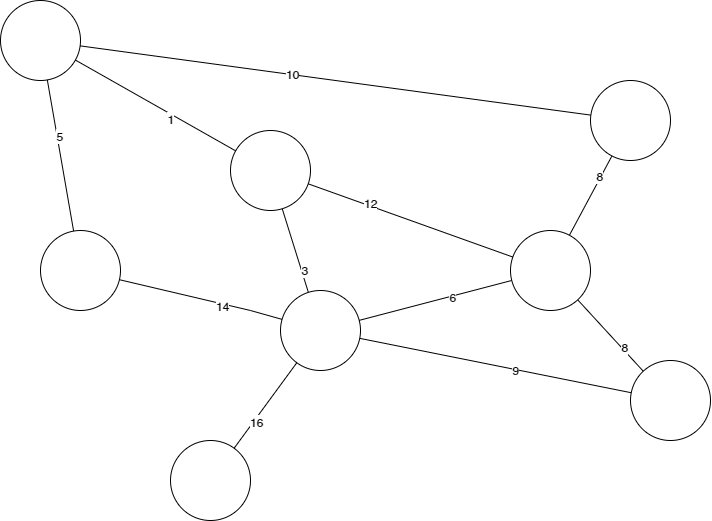


Рисунок 3 – оригинальный граф и минимальное дерево

**Алгоритм Прима:**

Алгоритм Прима для построение минимального островного дерева начинает обход с одной верщины и создает дерево, добавляя по одному ребру, до тех пор пока не будут включены все вершины («жадный алгоритм», выбирает лучшее локальное решение, игнорируя общую структуру дерева, что чревато издержками на длительность обхода этой структуры). Асимптотическая сложность алгоритма O(n^2)

**Алгоритм Крускала**

Альтернатива алгоритму Прима, в свою очередь начинает построение дерева не с одной определенной точки, а со всех точек одновременно, считая их частями дерева связи, между которыми нужно восстановить. Для правильной работы алгоритма требуется обязательно проверять, чтобы вершины, соединенные ребром на каждом шаге, не имели альтернативного пути между друг другом. Асимптотическая сложность алгоритма O(n\*m), но если постараться, можно получить O(m\*lg(m)). Алгоритм показывает лучшую производительность на разряженных графах, чем алгоритм Прима.

**Задача о минимальном пути в графе**

**Путем** называется последовательность рёбер, соединяющих две вершины в графе.

**Алгоритм Дейкстры**

Алгоритм находит кратчайший путь от заданной вершины ко всем другим вершинам графа, включая требуемую конечную вершину t.

*Кратчайший-путь-Дейкстры (Граф, начальная, конечная)*

*Из начальной точки до начальной точки путь занимает 0 единиц.*

*Итерируем от 1 до n, устанавливаем дистанции для каждой вершины в максимум*

*Начиная с начальной точки и до тех пор, пока есть новый узел:*

*Начиная с первого ребра от текущей точки и до тех пор пока есть необследованные ребра*

*Берем вес текущего ребра и номер конечной вершины.*

*Сравниваем уже вычисленное ранее расстояние до конечной вершины ребра и дистанцию до текущего узла, суммированную с весом ребра, выбираем минимальное и записываем как дистанцию до конечной вершины ребра, а так же значение родительского для измененного пути.*

*Выбираем следующей вершиной ту, которая имеет* ***минимальное расстояние от изначальной.***

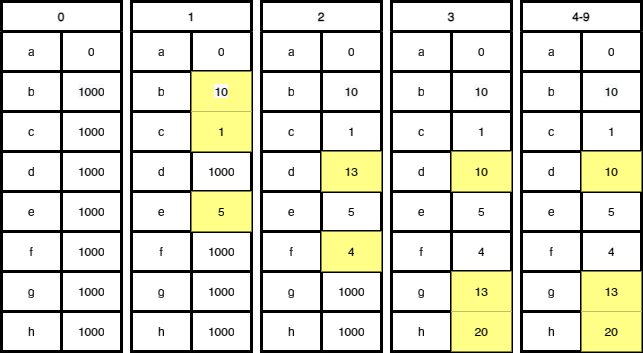
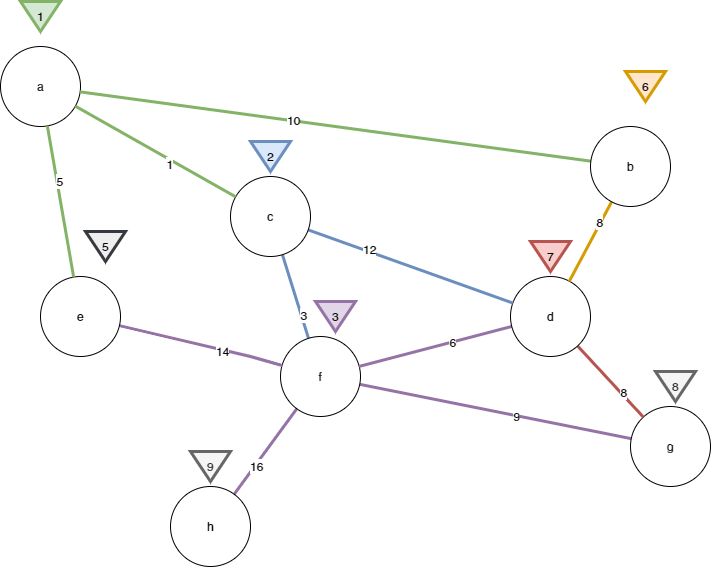
Важно помнить, что алгоритм Дейкстры **не работает** в графах с отрицательным значением. **Сложность алгоритма O(n^2)** Путь из алгоритма получается **обратным раскручиванием массива родителей.**

Алгоритм Дейкстры имеет алгоритмическую сложность **O(n²)** в своей базовой реализации, где **n** — количество вершин в графе. Эта сложность возникает из-за того, что на каждом шаге алгоритма выполняется два основных действия:

1. **Поиск вершины с минимальным расстоянием**: На каждом шаге алгоритм выбирает вершину с наименьшим текущим расстоянием из множества ещё не обработанных вершин. В базовой реализации это делается путём полного перебора всех вершин, что требует **O(n)** операций.
2. **Релаксация рёбер**: После выбора вершины алгоритм обновляет расстояния до всех её соседей. В худшем случае (например, в полном графе) у вершины может быть **O(n)** соседей, и обновление расстояний также требует **O(n)** операций.

Поскольку эти действия выполняются для каждой вершины (всего **n** вершин), общая сложность алгоритма составляет **O(n) × O(n) = O(n²)**.

Предполагается, что алгоритм останавливается только тогда, когда посетит все вершины, даже если таблица маршрутов перестанет меняться



**Алгоритм Флойда-Уоршелла**

Этот алгоритм предназначен для получения кратчайших путей между всеми парами вершин существующих в графе. Результатом работы этого алгоритма является матрица N x N, где N количество вершин. Особенностью, которую нужно учитывать, является то, что в матрицах смежности для взвешенного графа нельзя использовать 0 для указания несуществующего пути, лучше использовать максимальное значение.

Асимптотическая сложность равно O(n^3), что равно n вызову алгоритмов Дейкстры.

Алгоритм можно использовать для определения существования путей между двумя точками на графе.

**Дополнительные алгоритмы поиска**

**Алгоритм А\***

Является модификацией алгоритма Дейкстры с добавлением эвристической оценки приоритетности проверки дальнейшего продолжения маршрута. Сложность алгоритма зависит от качества выбора эвристики, и если она не удовлетворительна то может быть экспоненциальной.

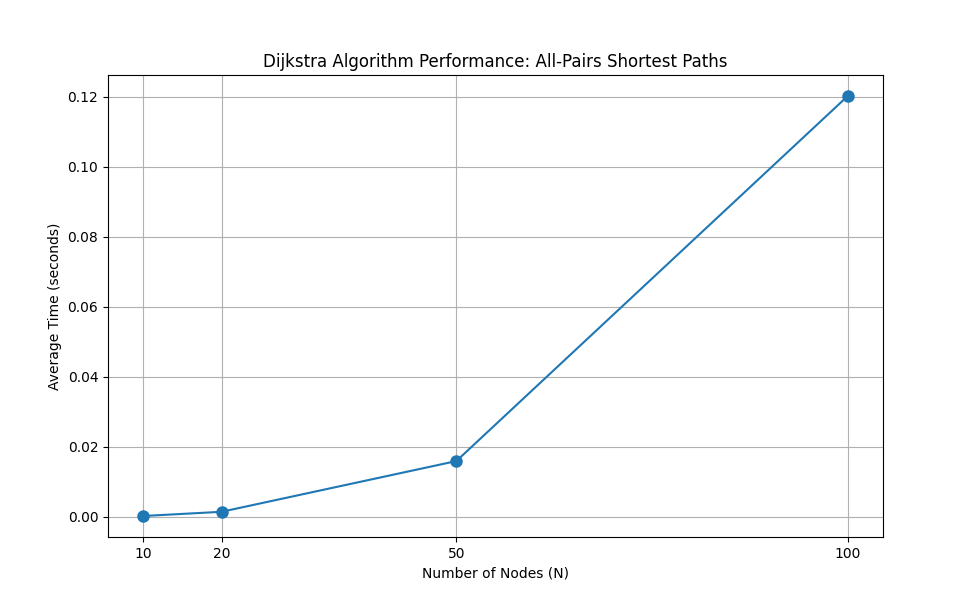
**Алгоритм Беллмана-Форда**

Это алгоритм выполняющий ту же задачу что и алгоритм Дейкстры, с очень близкими к нему идеями, но за счет поиска обратного отрицательного цикла позволяет искать пути на графе с отрицательными весами.

**Алгоритм Джонсона**

Это алгоритм который совмещает алгоритм Беллмана - Форда и алгоритм Дейксты таким образом достигая сложности O(V\*E\*log(V))

# Выполнение задачи.

****

# Заключение.

В ходе лабораторной работы была реализована генерация случайных графов с возможностью задания минимального числа рёбер для каждой вершины.

Был реализован и протестирован алгоритм Дейкстры для поиска кратчайших путей от каждой вершины во всём графе. Проведённые эксперименты показали, как время работы алгоритма изменяется в зависимости от размера графа. Полученные результаты были визуализированы с помощью графика, что позволило проанализировать производительность алгоритма на графах разного размера.

В целом, работа продемонстрировала эффективность алгоритма Дейкстры, но также показала его ограничения при увеличении количества вершин. Для более крупных графов может потребоваться использование более оптимизированных алгоритмов.