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

**Вариант 1**

Выполнил студент группы КС-38 (Казанцев Леонард Антонович)

Ссылка на репозиторий: (https://github.com/SimpleMaking/Algorithms-6-sem-/tree/main/Lab\_2(сортировка\_part2))

Приняли: Пысин Максим Дмитриевич

Краснов Дмитрий Олегович

Дата сдачи: (17.03.2023)

Оглавление

[Описание задачи. 2](file:///C:\Users\leo71\source\6_sem\алгоритмы\Lab_4(graphs)\Лаба_4(отчет).docx#_Toc127270410)

[Описание метода/модели. 3](file:///C:\Users\leo71\source\6_sem\алгоритмы\Lab_4(graphs)\Лаба_4(отчет).docx#_Toc127270411)

[Выполнение задачи. 4](file:///C:\Users\leo71\source\6_sem\алгоритмы\Lab_4(graphs)\Лаба_4(отчет).docx#_Toc127270412)

[Заключение. 4](file:///C:\Users\leo71\source\6_sem\алгоритмы\Lab_4(graphs)\Лаба_4(отчет).docx#_Toc127270413)

# 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.

(Можно использовать генератор из предыдущей лабораторной работы.)

1. Выведите получившийся граф в виде матрицы смежности. Пример вывода данных:
2. 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.
   * **Вариант 2**. Построить минимальное остовное дерево взвешенного связного неориентированного графа с помощью алгоритма Прима.
   * **Вариант 3**. Найти кратчайшие пути между всеми вершинами графа и их длину с помощью алгоритма Флойда — Уоршелла.
   * **Вариант 4**. Построить минимальное остовное дерево взвешенного связного неориентированного графа с помощью алгоритма Краскала.
3. 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.

Взвешенный граф — граф, каждому ребру которого поставлено в соответствие некое значение (вес ребра).

Алгоритм Дейкстры

Алгоритм на графах, изобретённый нидерландским учёным Эдсгером Дейкстрой в 1959 году. Находит кратчайшие пути от одной из вершин графа до всех остальных. Алгоритм работает только для графов без рёбер отрицательного веса. Алгоритм широко применяется в программировании, например, его используют протоколы маршрутизации OSPF и IS-IS. Вариант алгоритма для моего задания работает только со связным графом.

Описание алгоритма

Каждой вершине из ***V*** сопоставим метку — минимальное известное расстояние от этой вершины до ***a***.

Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки.

Работа алгоритма завершается, когда все вершины посещены.

**Инициализация**.

Метка самой вершины ***a*** полагается равной 0, метки остальных вершин — бесконечности.

Это отражает то, что расстояния от ***a*** до других вершин пока неизвестны.

Все вершины графа помечаются как непосещённые.

**Шаг алгоритма**.

Если все вершины посещены, алгоритм завершается. В противном случае, из ещё не посещённых вершин выбирается вершина ***u***, имеющая минимальную метку.

Мы рассматриваем всевозможные маршруты, в которых ***u*** является предпоследним пунктом. Вершины, в которые ведут рёбра из ***u***, назовём *соседями* этой вершины. Для каждого соседа вершины ***u***, кроме отмеченных как посещённые, рассмотрим новую длину пути, равную сумме значений текущей метки ***u*** и длины ребра, соединяющего ***u*** с этим соседом.

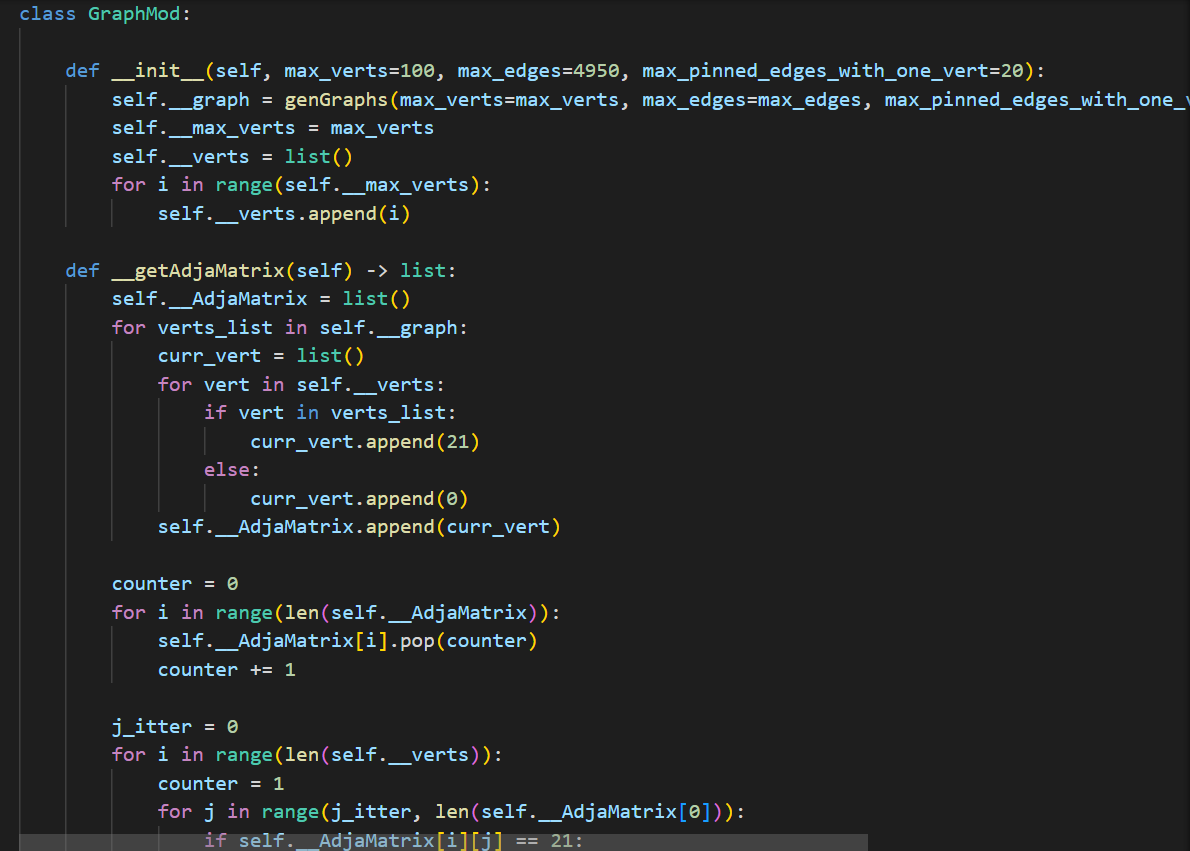
Если полученное значение длины меньше значения метки соседа, заменим значение метки полученным значением длины. Рассмотрев всех соседей, пометим вершину ***u*** как посещённую и повторим шаг алгоритма.

# Выполнение задачи.

Для решения поставленной задачи был выбран язык программирования python. Взвешенный граф реализован в виде класса с методами получения матрицы смежности и списка ребер.

Код класса:

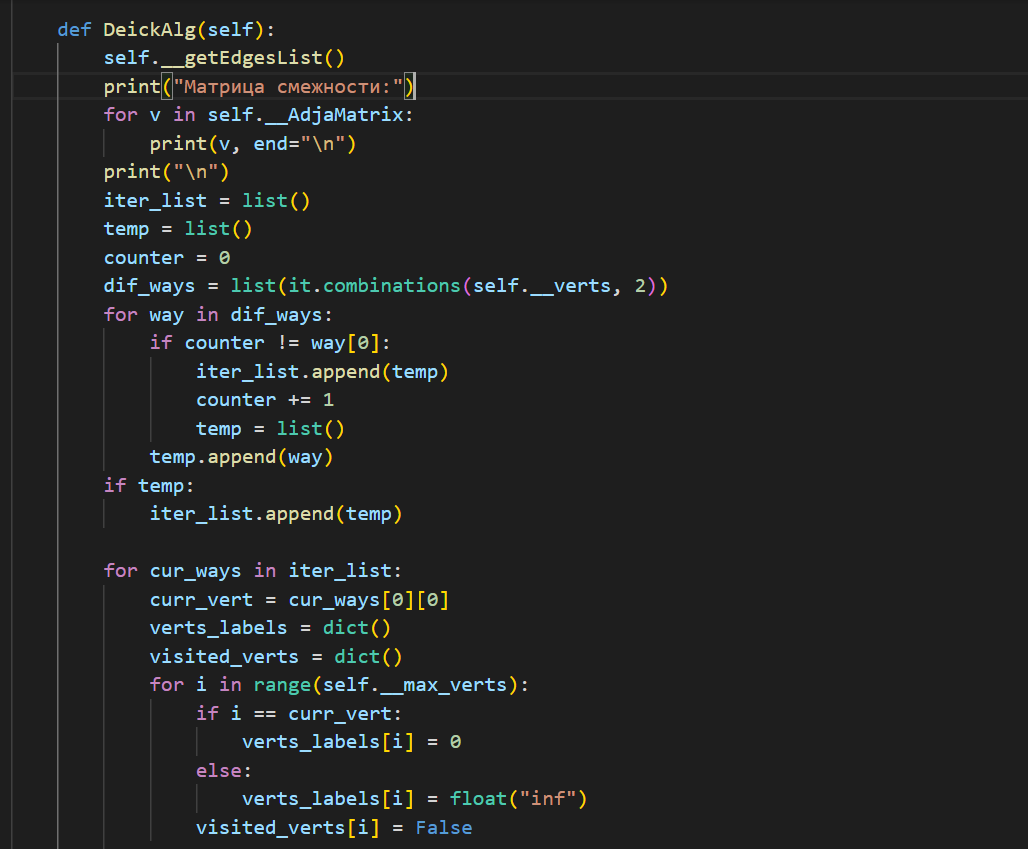
* 1. В конструкторе класса получаем сгенерированный граф, далее он будет взвешенным.
  2. Создаем массив вершин.
  3. Функция получения матрицы смежности.



* 1. Получение списка ребер.

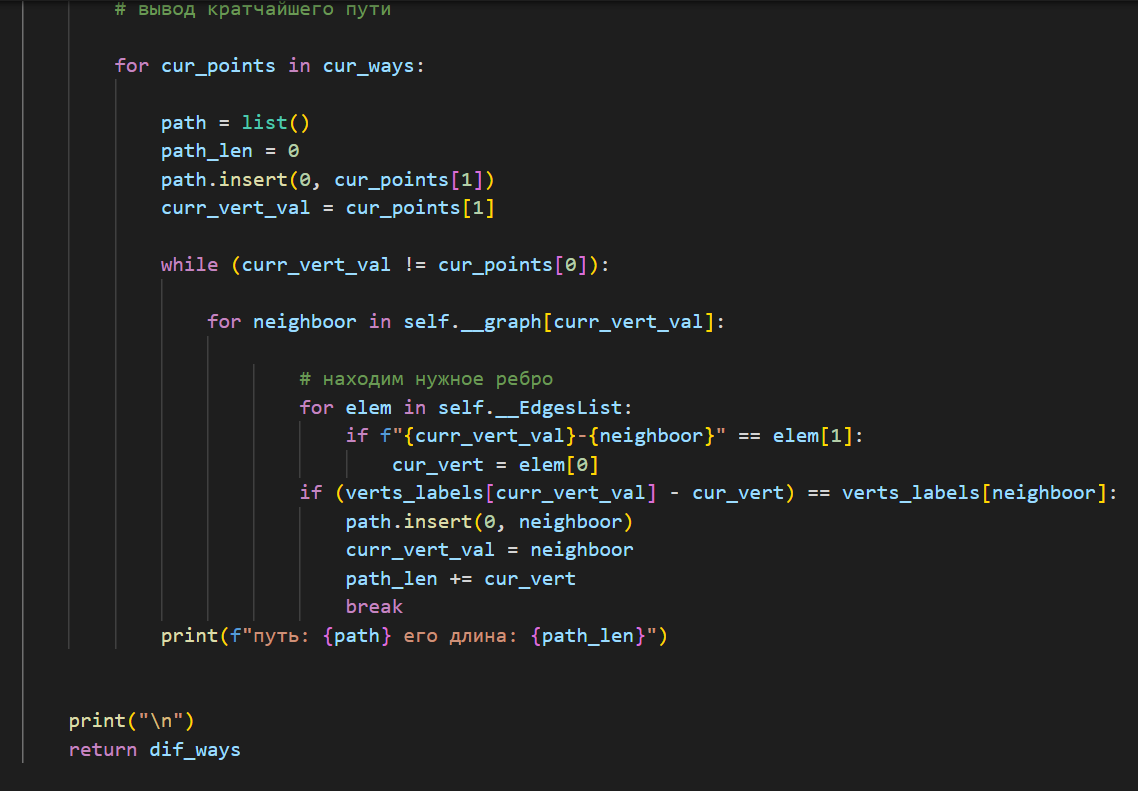


* 1. Алгоритм Дейкстры. Он находит все возможные пути в графе от одной вершины до всех остальных.

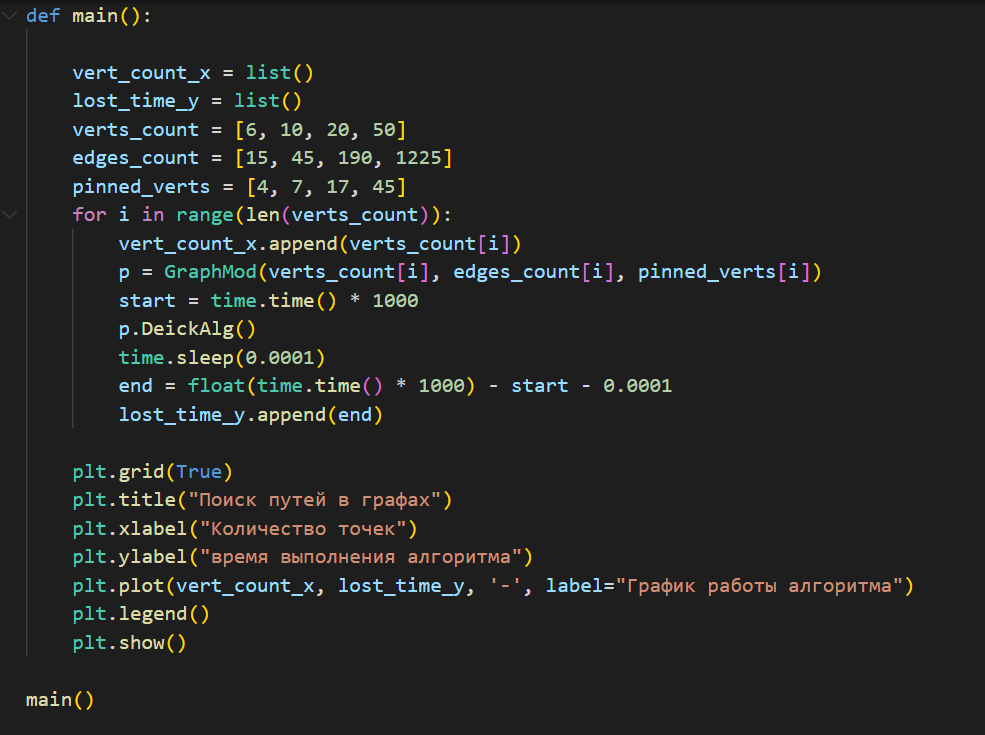


* 1. Получаем все возможные комбинации вершин, чтобы в последствии найти между ними минимальное расстояние.
  2. В цикле для каждой вершины находим минимальное расстояние от нее до всех остальных вершин. Сначала размечаем наш граф, находя для каждой его вершины свой минимальный вес (расстояние до нее), потом восстанавливаем все пути от начальной вершины до всех остальных, идя с конца.



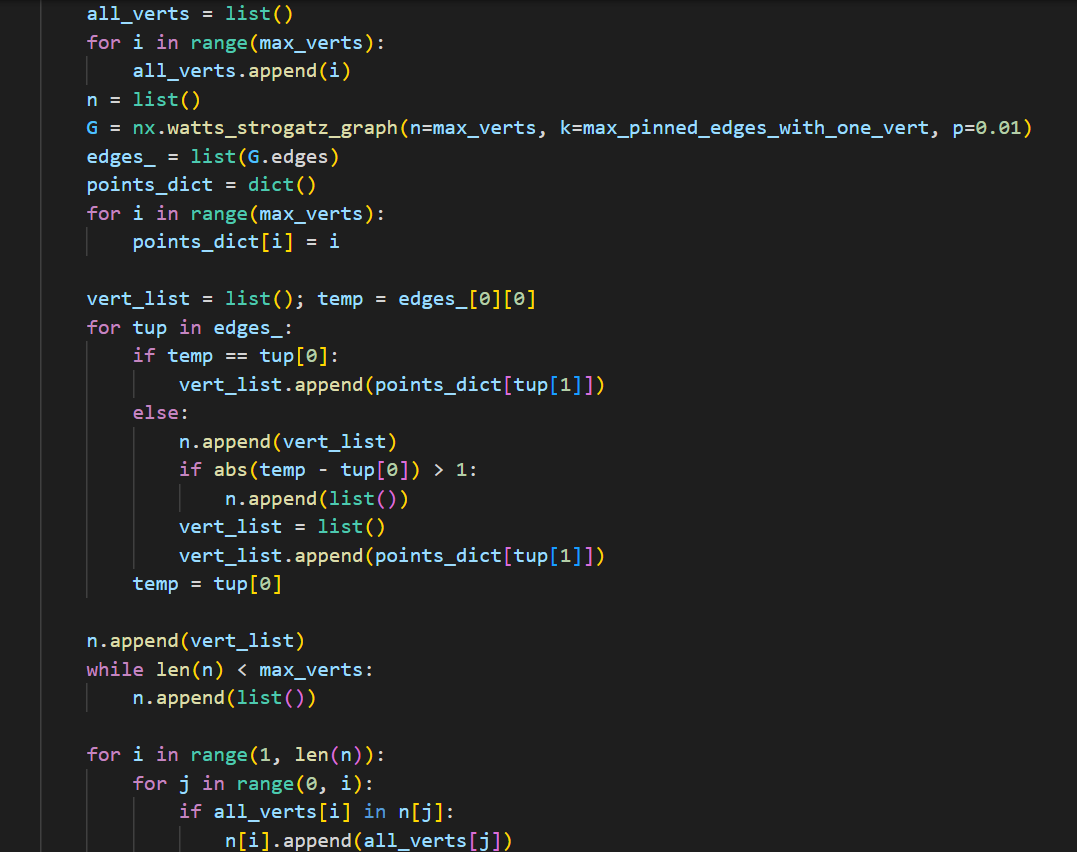


* 1. Это весь код класса, далее в главной функции проводим серию тестов.

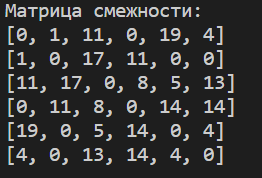


Генерация графов производилась в отдельном файле функцией genGraphs().

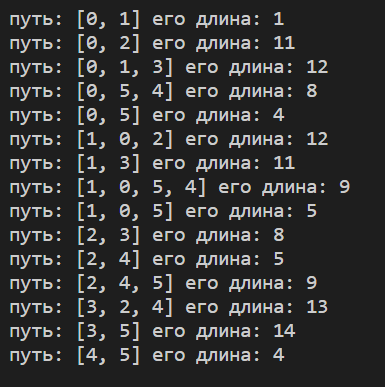
1. В основе графа лежит граф «маленького мира» где из каждой вершины можно попасть в каждую и добавляется связь текущей вершины с предыдущими, при условии, что предыдущие связаны с текущей (только для ненаправленного графа).



Рассмотрим функционал класса и работу функции генерации: для примера взял ненаправленный взвешенный граф из 6 вершин, каждая из них представлена цифрами от нуля и так далее, пока не пометим все вершины. Вывод матрицы смежности данного графа:

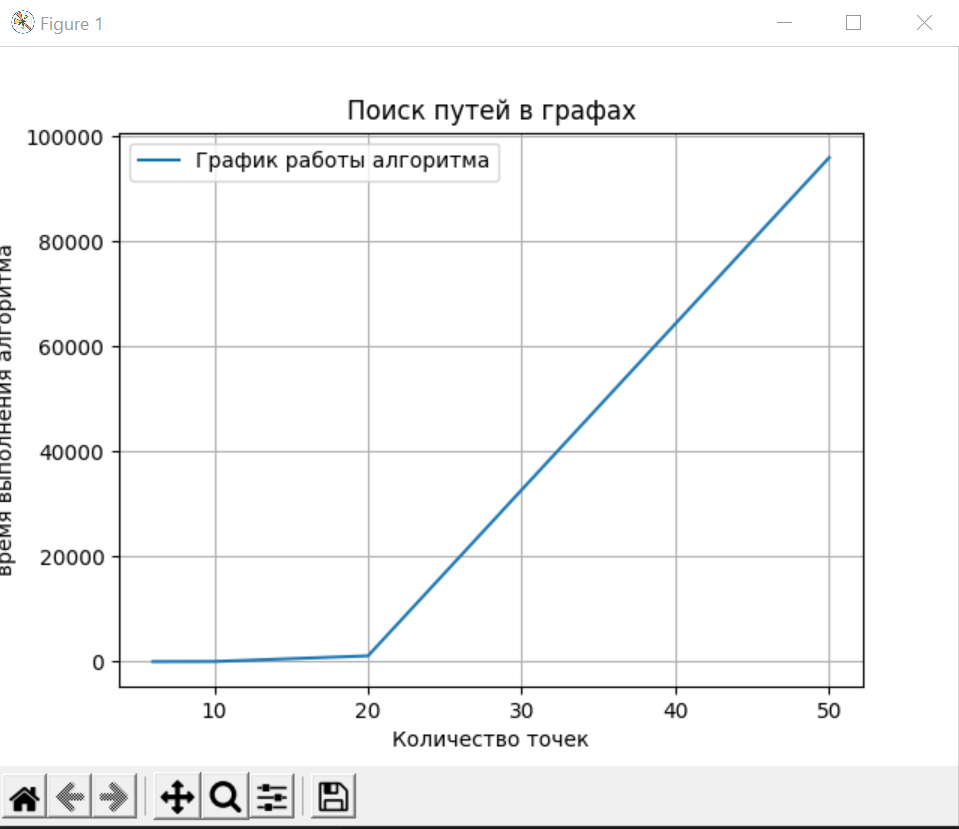


Матрица смежности



Длина пути между всеми вершинами

Далее в качестве теста генерируются 4 графа с разным количеством вершин и ребер: 6, 10, 20, 50 вершин. У каждого из них с помощью алгоритма Дейкстры находится длина кратчайшего пути от начальной точки до конечной, также замеряется время работы алгоритма. График, показывающий зависимость времени выполнения алгоритма в миллисекундах от количества вершин.



Пути в 4 графах

Из графика видно, что чем больше вершин, тем все более дольше выполняется поиск, что довольно логично.

# Заключение.

Реализация графов и их генерация оказалась не очень сложной. Видно, что алгоритм Дейкстры эффективен и может использоваться в реальных задачах поиска наилучших путей. Задание выполнено полностью.