Лабораторная работа № 9 по курсу дискретного анализа: Графы

Выполнил студент группы М8О-310Б-21 МАИ *Катаев Юрий* .

Условие

1. Разработать программу на языке C или C++, реализующую указанный алгоритм. Формат входных и выходных данных описан в варианте задания. Первый тест в проверяющей системе совпадает с примером.

2. **Вариант 4: Поиск кратчайшего пути между парой вершин алгоритмом Дейкстры.** Задан взвешенный неориентированный граф, состоящий из n вершин и m рёбер. Вершины пронумерованы целыми числами от 1 до n. Необходимо найти длину кратчайшего пути из вершины с номером $start$ в вершину с номером $finish$ при помощи алгоритма Дейкстры. Длина пути равна сумме весов ребер на этом пути. Граф не содержит петель и кратных рёбер.

Метод решения

Алгоритм Дейкстры является одним из самых распространённых способов найти кратчайший путь между двумя вершинами в графе. Его недостатком является то, что он не работает в графах с рёбрами с отрицательным весом. Его перимущество заключается в том,что он ищет кратчайшие пути от заданной вершины до вообще всех остальных вершин (хотя в рамках поставленной задачи это будет лишним).

Первым делом сопоставим каждой вершине графа некое число — то самое наименьшее расстояние от неё до начальной вершины, у начальной вершины это будет очевидно 0, у всех остальных бесконечность, то есть достаточно большое число, затем для каждой вершины, связанной с начальной ровно 1 ребром проверим, является ли сопоставленное ей в начале число больше, чем сумма такого же числа той вершины, из которой мы приши и веса ребра, по которому пришли. Если является, заменяем число на эту самую сумму. Как только обошли всех соседей вершины, помечаем вершину как пройденную, это значит, что минимальное расстояние до этой вершины найдено и изменять его больше не надо. То же самое делаем со всеми остальными вершинами, пользуясь поиском в глубину.

Описание программы

К сожалению, тестирующая система не позволила использовать модульный подход к решению поставленной задачи, поэтому весь рабочий код размещён в единственном файле main.cpp

Сначала пользователь вводит количество вершин графа (n), количество рёбер (m), начальную (start) и конечную (finish) вершины. Затем пользователь вводит информацию о каждом ребре: вершины, которые оно соединяет, и его вес.

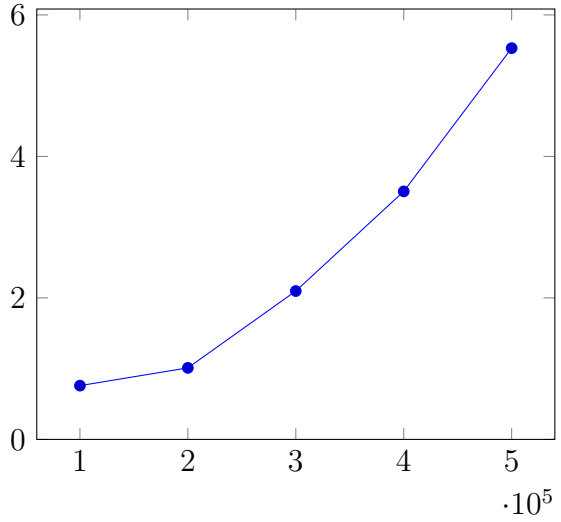
Далее программа находит кратчайший путь между начальной и конечной вершинами с помощью функции Dijkstra, которая принимает на вход граф и начальную и конечную вершины. Функция возвращает длину кратчайшего пути.

Если кратчайший путь существует, программа выводит его длину. Если пути не существует, выводится сообщение "No solution".

Программа использует структуру данных Edge для представления рёбер графа, а также priority queue для эффективной работы с минимальной кучей.

Тест производительности

Ниже приведен тест времени работы алгоритма. По оси X — количество вершин в графе, по оси Y — время выполнения алгоритма в с (меньше — лучше).



Кол-во вершин	Время (в с)
100000	0.7598959806880351
200000	1.0112411892099832
300000	2.0971491552203912
400000	3.5054276940877914
500000	5.5304168745434285

Мы получили ожидаемую сложность алгоритма $O(mlogn)$, так как наш алгоритм работает с двоичной кучей.

Выводы

В ходе выполнения лабораторной работы был успешно реализован Алгоритм Дейкстры для нахождения кратчайших путей во взвешенном графе. Алгоритм позволяет найти

кратчайший путь от одной из вершин графа до всех остальных вершин, учитывая веса ребер. Реализация алгоритма позволила эффективно находить оптимальные пути и использовать их в различных задачах, связанных с поиском кратчайших путей в графах. Работа с алгоритмом Дейкстры позволила лучше понять принципы работы алгоритмов на графах и их применение в реальных задачах.