Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №9 по курсу Дискретный анализ

Студент: А. Н. Марков
Преподаватель: Н. А. Зацепин
Группа: М8О-308Б
Дата:
Оценка:
Подпись:

Москва, 2020

Условие

1. Общая постановка задачи

   Разработать программу на языке C или C++, реализующую указанный алгоритм согласно заданию.

2. Вариант задания

   Вариант 4. Задан взвешенный неориентированный граф, состоящий из **n** вершин и **m** ребер. 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 ребер.

3. Формат входных данных

   В первой строке заданы $1 \leq n \leq 10^5$, $1 \leq m \leq 10^5$, $1 \leq start \leq n$ и $1 \leq finish \leq n$. В следующих m строках записаны ребра. Каждая строка содержит три числа – номера вершин, соединенных ребром, и вес данного ребра. Вес ребра – целое число от 0 до 10^9.

4. Формат результата

   Необходимо вывести одно число – длину кратчайшего пути между указанными вершинами. Если пути между указанными вершинами не существует, следует вывести строку "No solution"(без кавычек).

Метод решения

Алгоритм Дейкстры предназначен для поиска кратчайшего пути из вершины-истока до всех остальных вершин графа, но этот алгоритм легко преобразуется в алгоритм поиска кратчайшего пути из одной вершины в другую. Алгоритм Дейкстры применим только, когда веса ребер графа неотрицательны.

Основная идея данного алгоритма в том, что алгоритме Дейкстры поддерживается множество вершин S, для которых уже вычислены окончательные веса кратчайших путей к ним из истока s. Поочередно выбирается вершина $u \in V - S$, которой на данном этапе соответствует минимальная оценка кратчайшего пути. После добавления этой вершины u в множество S проводится ослабление всех исходящих из нее ребер.

В моей реализации множество вершин S - это массив из n элементов, в котором в i-ой ячейке хранится расстояние для i-ой вершины. Множество $V - S$ в моей реализации представляется очередь с приоритетами.

Анализ сложности алгоритма. Т.к. в моей реализации я использовал очередь приоритетов std::priority_queue из стандартной библиотеки C++, в которой отсутствует функция изменения приоритета одного из элементов очереди, то всего итераций в цикле for O(m), а значит, может быть вставлено в очередь приоритетов O(m) элементов. Тогда

получается, что всего может быть O(m) итераций цикла while. Итераций цикла for будет O(m), поскольку в теле цикла while не будут рассматриваться вершины, в которых уже найден кратчайший путь. Нахождение минимального элемента в очереди приоритетов $O(1)$. Операции вставки и удаления элементов из очереди приоритетов имеют сложность $O(log(m))$. Итоговая временная сложность алгоритма Дейкстры в данной реализации: $O(m * log(m))$. Если бы использовалась реализация очереди приоритетов, в которой была бы функция изменения приоритета элемента очереди, то временная сложность была бы равна $O((n+m) * log(n))$. Сложность по памяти в моей реализации: $O(n)$ памяти для хранения расстояний от истока до другой вершины, $O(m)$ памяти для хранения очереди с приоритетами. Итоговая сложность по памяти $O(n + m)$.

Описание программы

Проект состоит из 3 файлов:

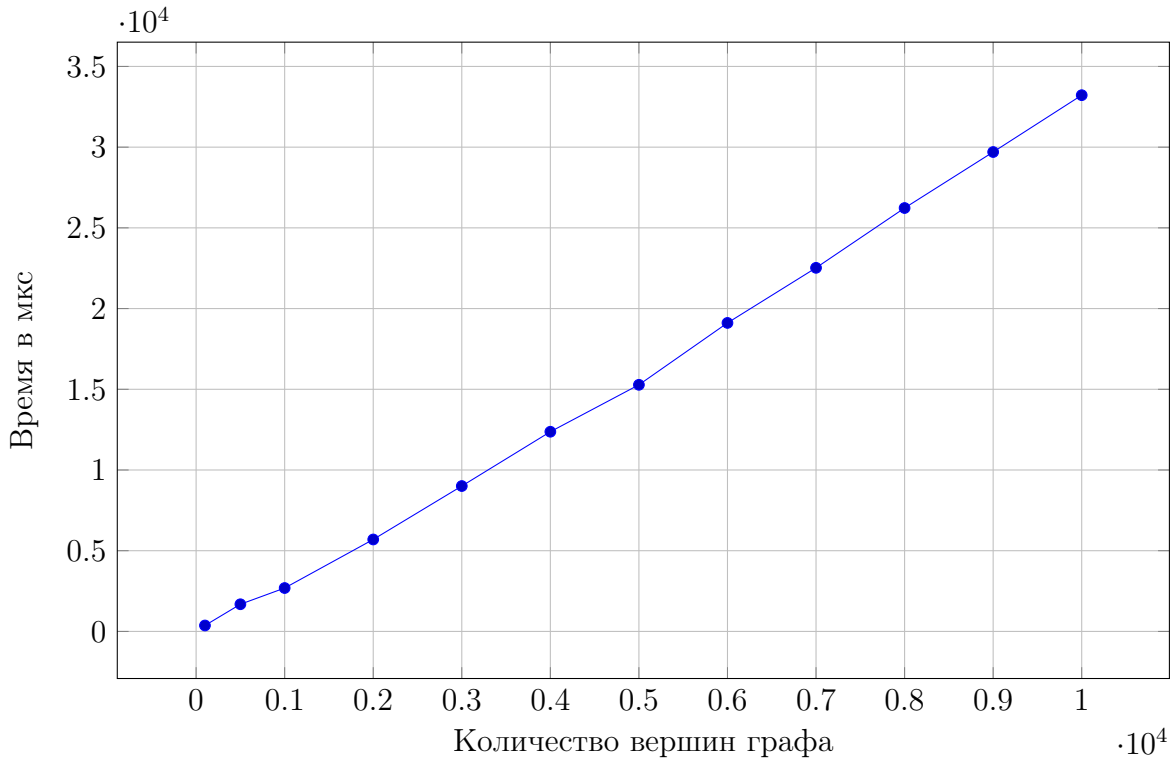
- main.cpp - главный файл, в котором реализована функция main
- graph.hpp - заголовочный файл, в котором находятся объявления классов ребра графа и графа.
- graph.cpp - файл, в котором содержатся реализации методов классов вершины дерева и суффиксного дерева.

Дневник отладки

Программа зашла на чекер с первой попытки.

Тест производительности

Тесты сгенерированы таким обрзом, что количество ребер графа равно количеству вершин графа, умноженному на 4.



Выводы

Графы обширно применяются в современном мире: прокладывание дорог, маршрутов, и, вообще, повсеместно в логистике. Частный случай графа - дерево, имеет широкое применение в информатике.

Алгоритм Дейкстры поиска кратчайшего пути от исходной вершины оказался не очень сложным. Недостатком моей программы является не очень хорошая реализация: я использовал очередь с приоритетами из стандартной библиотеки, но данная реализация не может обновлять приоритет для некоторого нужного элемента, из-за этого окончательная временная сложность получилась $O(m * log(m))$. Чтобы добиться сложности $O((n+m)*log(n))$, необходимо реализовать собственную очередь с приоритетами.