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: П. А. Мохляков
Преподаватель: Н. С. Капралов
Группа: М8О-308Б

Дата:
Оценка:
Подпись:

Москва, 2021

Лабораторная работа №9

Задача: Разработать программу на языке C или C++.

Вариант: 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.

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.

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).

1 Описание

Алгоритм Дейкстры — это алгоритм поиска оптимального пути на взвешенном графе. При этом стоит заметить, что данный алгоритм будет работать некоректно при отрицателных весах.

Рассмотрим принцип работы данного алгоритма. Для начала мы помечам длину пусти до стартовой вершины равной нулю, а для всех остальных равной бесконечности. После чего мы рассматриваем не рассмотренную ранее вершину с минимальным значением пути,в начале это будет стартовая вершина. Если сумма длины до рассматриваемой вершины и пути до ее ребенка меньше, чем записанное значение длины до ребенка, то обновляем его. Поторяем данную процедуру пока не закончатся вершины, если нужно найти кратчайшие пути для всех вершин, или пока целевая вершина не станет рассматриваемой.

В моем случае, чтобы каждый раз не искать нерассматриваемую вершину с минимальной длиной пути, я использую очередь с преоритетом. Изначально в нее помещается только стартовая точка. В дальнейшем, на каждой итерации в нее будут помещаться все непосещенные вершины или вершины для которых найден более оптимальный путь.

2 Исходный код

```
#include <iostream>
#include <vector>
#include <map>
#include <unordered_set>


uint64_t Deikstra(int s, int f, std::vector<std::map<int,int>> &graph){
    std::map<uint64_t,int> priority_queue;
    priority_queue.emplace(std::make_pair(0,s));
    std::unordered_set<int> visited;
    std::map<int,uint64_t> visited_cost;
    visited.insert(s);
    visited_cost[s] = 0;
    while(!priority_queue.empty()){
        int node = priority_queue.begin()->second;
        priority_queue.erase(priority_queue.begin());
        if(node == f){
            break;
        }
        for(auto &i:graph[node]){
            if(visited.count(i.first) == 0 || visited_cost[node] + i.second <
                visited_cost[i.first]){
                priority_queue.emplace(std::make_pair(visited_cost[node] + i.second,
                    i.first));
                visited_cost[i.first] = visited_cost[node] + i.second;
                visited.insert(i.first);
            }
        }
    }
    if(visited.count(f) == 0)
        return -1;
    else
        return visited_cost[f];
}

int main(){
    int n,m,s,f;
    std::cin >> n >> m >> s >> f;
    --s;
    --f;
    std::vector<std::map<int,int>> graph(n);
    for(int i = 0; i < m; ++i){
        int a, b, cost;
        std::cin >> a >> b >> cost;
        graph[a-1][b-1] = cost;
        graph[b-1][a-1] = cost;
    }
```

```
46          uint64_t ans = Deikstra(s,f,graph);
47          if(ans == -1){
48              std::cout << "No solution" << std::endl;
49          } else {
50              std::cout << ans << std::endl;
51          }
52      }
```

3 Консоль

```
pavel@DESKTOP-SVKRTNN ~/work/MAI/2_course/DA/LB9 cat test
5 6 1 5
1 2 2
1 3 0
3 2 10
4 2 1
3 4 4
4 5 5
pavel@DESKTOP-SVKRTNN ~/work/MAI/2_course/DA/LB9 g++ main.cpp -o main
pavel@DESKTOP-SVKRTNN ~/work/MAI/2_course/DA/LB9 cat test | ./main
8
```

4 Тест производительности

Алгоритм Дейкстры останавливается, когда находит минимальную длину пути между искомыми вершинами.

```
pavel@DESKTOP-SVKRTNN ~/work/MAI/2_course/DA/LB9 cat tests/test1.txt| ./banch
Graph size: 5
Deikstra time: 1.3144e-05
FloydWarshall time: 2.555e-06
pavel@DESKTOP-SVKRTNN ~/work/MAI/2_course/DA/LB9 cat tests/test2.txt| ./banch
Graph size: 4
Deikstra time: 1.5398e-05
FloydWarshall time: 1.814e-06
pavel@DESKTOP-SVKRTNN ~/work/MAI/2_course/DA/LB9 cat tests/test3.txt| ./banch
Graph size: 10
Deikstra time: 2.5067e-05
FloydWarshall time: 1.8946e-05
pavel@DESKTOP-SVKRTNN ~/work/MAI/2_course/DA/LB9 cat tests/test4.txt| ./banch
Graph size: 50
Deikstra time: 4.8671e-05
FloydWarshall time: 0.00241784
pavel@DESKTOP-SVKRTNN ~/work/MAI/2_course/DA/LB9 cat tests/test5.txt| ./banch
Graph size: 100
Deikstra time: 0.00123547
FloydWarshall time: 0.0188629
```

Алгоритм Дейкстры делает полный обход.

```
pavel@DESKTOP-SVKRTNN ~/work/MAI/2_course/DA/LB9 cat tests/test1.txt| ./banch
Graph size: 5
Deikstra time: 1.5369e-05
FloydWarshall time: 2.885e-06
pavel@DESKTOP-SVKRTNN ~/work/MAI/2_course/DA/LB9 cat tests/test2.txt| ./banch
Graph size: 4
Deikstra time: 1.7022e-05
FloydWarshall time: 1.844e-06
pavel@DESKTOP-SVKRTNN ~/work/MAI/2_course/DA/LB9 cat tests/test3.txt| ./banch
Graph size: 10
Deikstra time: 4.9633e-05
FloydWarshall time: 2.126e-05
pavel@DESKTOP-SVKRTNN ~/work/MAI/2_course/DA/LB9 cat tests/test4.txt| ./banch
```

```
Graph size: 50
Deikstra time: 0.00043767
FloydWarshall time: 0.00214278
pavel@DESKTOP-SVKRTNN ~/work/MAI/2_course/DA/LB9 cat tests/test5.txt| ./banch
Graph size: 100
Deikstra time: 0.0101413
FloydWarshall time: 0.0160607
```

Так как алгоритм Дейкстры работает примерно за $O(n*log(m))$, а алгоритм Флойда-Уоршелла работает за $O(n^3)$, то с увеличением размера теста алгоритм Дейкстры работает все быстрее в сравнении с алгоритмом Флойда-Уоршелла. Также в алгоритме Дейкстры мы можем предусмотреть остановку обхода при нахождении минимального пути между нужными точками, в алгоритме Флойда-Уоршелла нам в любом случае нужно сделать полных обход.

5 Выводы

Выполнив данную лабораторную работу по курсу «Дискретный анализ» я узнал об алгоритме поиска оптимального пути на взвещенном графе, алгоритме Дейкстры. Данный алгоритм находит «вес» путей до всех вершин от стартовой вершины за $O(n^2)$. Минусом данного алгоритма является то, что работает некоректно при отрицательном «весе» вершин.

Список литературы

[1] Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. *Алгоритмы: построение и анализ, 2-е издание.* — Издательский дом «Вильямс», 2007. Перевод с английского: И. В. Красиков, Н. А. Орехова, В. Н. Романов. — 1296 с. (ISBN 5-8459-0857-4 (рус.))