Московский авиационный институт
(национальный исследовательский университет)

Факультет компьютерных наук и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №9 по курсу «Дискретный анализ»

Студент: А. А. Каримов
Преподаватель: А. А. Кухтичев
Группа: М8О-306Б-22
Дата:
Оценка:
Подпись:

Москва, 2024

Лабораторная работа №9

Задача: 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.

1 Описание

Алгоритм Дейкстры - это алгоритм-обобщение обхода в ширину для взвешенных графов. На каждой итерации алгоритм берет вершину из очереди и смотрит смежные вершины и пытается до вершины v - вместо пути d[v] - из вершины u построить более оптимальный путь $d[u] + w_{uv}$. Если же такой путь возможен, и он оптимальнее, то обновляется значение d[v] и вершина v вместе с d[v] складывается в приоритетную очередь. Приоритетная очередь выдает нам на каждом шаге вершину с минимальным d[v]. Таким образом, мы всегда будем брать вершину с кратчайшим путем до нее, что представляет собой, своего рода, жадность. Итого, мы перебираем все вершины V, вставляем их в priority_queue - $logV$ и пытаемся релаксировать E ребер: $O(VlogV + E)$ - сложность.

2 Исходный код

Здесь располагается реализация задачи.

```cpp
namespace task {

using ll = int64_t;
struct wedge {
    int u, v;
    ll w;

    wedge (int _u, int _v, ll _w) : u(_u), v(_v), w(_w) {}
};

using wgraph = std::vector<std::vector<wedge>>;

using item = std::pair<ll, int>;


void dijkstra(int u, const wgraph& g, std::vector<ll>& d) {
    int n = g.size();

    d[u] = 0;

    std::priority_queue<item, std::vector<item>, std::greater<item>> pq;
    pq.push(std::make_pair(0, u));

    std::vector<bool> visited(n, false);

    while(!pq.empty()) {
        item cur = pq.top();
        pq.pop();
        u = cur.second;
        if(visited[u]) continue;

        visited[u] = true;

        for(wedge uv : g[u]) {
            int v = uv.v;
            ll w = uv.w;

            if(d[u] + w < d[v]) {
                d[v] = d[u] + w;
                pq.push(std::make_pair(d[v], v));
            }

        }
    }

}
```

3 Консоль

```
karseny99@karseny99:/mnt/study/DA/lab9/src$ cat 1.in
5 6 1 5
1 2 2
1 3 0
3 2 10
4 2 1
3 4 4
4 5 5
karseny99@karseny99:/mnt/study/DA/lab9/src$ ./lab9 <1.in
8
```

4 Тест производительности

Производительность алгоритма на тестах $10^3, 10^4, 10^5$

```
karseny99@karseny99:/mnt/study/DA/lab9$ g++ -std=c++20 main.cpp && ./a.out
<1.in
Dijkstra: 658us
karseny99@karseny99:/mnt/study/DA/lab9$ g++ -std=c++20 main.cpp && ./a.out
<2.in
Dijkstra: 3032us
karseny99@karseny99:/mnt/study/DA/lab9$ g++ -std=c++20 main.cpp && ./a.out
<3.in
Dijkstra: 29601us
```

5 Выводы

В ходе выполнения лабораторной работы я реализовал алгоритм Дейкстры поиска кратчайшего пути в графе. Используя приоритетную очередь, Дейкстра выбирает вершины с минимальным расстоянием, что позволяет быстро находить оптимальные решения.

Список литературы

[1] Кормен, Т. Х. *Алгоритмы: построение и анализ.* — Издательский дом «Вильямс», 2007. Перевод с английского: И. В. Красиков, Н. А. Орехова, В. Н. Романов. — 1296 с. (ISBN 5-8459-0857-4 (рус.))