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: М. М. Парфенов
Преподаватель: С. А. Михайлова
Группа: М8О-301Б-22

Дата:
Оценка:
Подпись:

Москва, 2025

Лабораторная работа №9

Задача: Задан взвешенный ориентированный граф, состоящий из n вершин и m ребер. Вершины пронумерованы целыми числами от 1 до n. Необходимо найти величину максимального потока в графе при помощи алгоритма Форда-Фалкерсона. Для достижения приемлемой производительности в алгоритме рекомендуется использовать поиск в ширину, а не в глубину. Истоком является вершина с номером 1, стоком – вершина с номером n. Вес ребра равен его пропускной способности. Граф не содержит петель и кратных ребер.

1 Описание

Д

Алгоритм

Алгоритм Форда-Фалкерсона использует поиск в ширину (BFS) для нахождения пути с остаточным потоком от истока к стоку. Этот путь выбирается таким образом, чтобы по нему можно было провести дополнительный поток, равный минимальной пропускной способности на пути. После нахождения такого пути, пропускные способности всех ребер на этом пути уменьшаются на величину потока, а пропускные способности обратных ребер увеличиваются на эту же величину. Алгоритм повторяется до тех пор, пока не удастся найти путь с остаточным потоком.

Для повышения производительности в реализации применяется алгоритм Эдмондса-Карпа, который является улучшенной версией алгоритма Форда-Фалкерсона и использует BFS для поиска пути с остаточным потоком.

Описание алгоритма

1. Входные данные содержат:

   - Количество вершин n и количество ребер m.
   - Для каждого ребра: две вершины u и v, а также пропускную способность c ребра от u к v.

2. Используем структуру данных, представляющую граф в виде матрицы пропускных способностей $capacity[u][v]$, которая хранит информацию о пропускной способности ребра от вершины u к вершине v.

3. Метод Эдмондса-Карпа:

   1. Для каждого пути с остаточным потоком от истока к стоку, находим минимальную пропускную способность на этом пути.
   2. Уменьшаем пропускные способности вдоль пути и увеличиваем обратные пропускные способности.
   3. Добавляем найденный поток в общий максимальный поток.

4. Процесс повторяется до тех пор, пока можно найти путь от истока к стоку с положительным остаточным потоком.

5. Результатом работы алгоритма является величина максимального потока от истока к стоку.

Описание кода

В коде создается класс Graph, который управляет матрицей пропускных способностей и реализует метод BFS для поиска пути с остаточным потоком. Метод `edmondsKarp` выполняет основной цикл поиска путей и обновления потока. Входные данные читаются с использованием стандартного ввода, затем для каждого ребра добавляются пропускные способности в граф. Алгоритм выводит максимальный поток, который можно провести от истока до стока.

Сложность

- Время работы алгоритма Эдмондса-Карпа: $O(V \cdot E^2)$, где V — количество вершин, E — количество ребер в графе. - Пространственная сложность: $O(V^2)$ для хранения матрицы пропускных способностей.

2 Исходный код

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;

const ll INF = 1e18;

class Graph {
public:
    int V;
    vector<vector<ll>> capacity;

    Graph(int V) {
        this->V = V;
        capacity.resize(V, vector<ll>(V, 0));
    }

    void addEdge(int u, int v, ll cap) {
        capacity[u][v] += cap;
    }

    bool bfs(int s, int t, vector<int>& parent) {
        vector<bool> visited(V, false);
        queue<int> q;
        q.push(s);
        visited[s] = true;
        parent[s] = -1;

        while (!q.empty()) {
            int u = q.front();
            q.pop();

            for (int v = 0; v < V; v++) {
                if (!visited[v] && capacity[u][v] > 0) {
                    q.push(v);
                    parent[v] = u;
                    visited[v] = true;
                    if (v == t) return true;
                }
            }
        }

        return false;
    }

    ll edmondsKarp(int source, int sink) {
        ll maxFlow = 0;
        vector<int> parent(V);
```

```
48
49          while (bfs(source, sink, parent)) {
50              ll pathFlow = INF;
51
52              for (int v = sink; v != source; v = parent[v]) {
53                  int u = parent[v];
54                  pathFlow = min(pathFlow, capacity[u][v]);
55              }
56
57              for (int v = sink; v != source; v = parent[v]) {
58                  int u = parent[v];
59                  capacity[u][v] -= pathFlow;
60                  capacity[v][u] += pathFlow;
61              }
62
63              maxFlow += pathFlow;
64          }
65
66          return maxFlow;
67      }
68  };
69
70  int main() {
71      ios::sync_with_stdio(false);
72      cin.tie(0);
73
74      int n, m;
75      cin >> n >> m;
76      Graph g(n);
77
78      for (int i = 0; i < m; i++) {
79          int u, v;
80          ll c;
81          cin >> u >> v >> c;
82          --u, --v;
83          g.addEdge(u, v, c);
84      }
85
86      cout << g.edmondsKarp(0, n - 1) << endl;
87
88      return 0;
89  }
```

3 Консоль

```
 [±main]-> cat input.txt                                                5 6
1 2 4
```

```
1 3 3
1 4 1
2 5 3
3 5 3
4 5 10


 [±main]-> g++ main.cpp
 [±main]-> ./a.out < input.txt
 [±main]->
7
```

4 Выводы

Выполнив лабораторную работу по поиску максимального потока с помощью алгоритма Форда-Фалкерсона, я научился эффективно решать задачи, связанные с потоками в графах. Использование поиска в ширину вместо глубины позволяет существенно улучшить производительность, что важно при работе с большими графами. Мне также понравился алгоритм за его простоту и эффективность, несмотря на свою базовость.

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.))

[2] Список использованных источников оформлять нужно по ГОСТ Р 7.05-2008