Московский авиационный институт
(национальный исследовательский университет)

Институт информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №9 по курсу дискретного анализа

Студент: А. В. Синявский
Преподаватель: Н. А. Зацепин
Группа: М8О-308Б-18
Дата:
Оценка:
Подпись:

Москва, 2020

Лабораторная работа № 9 по курсу дискретного анализа

Выполнил студент группы М80-308Б-18 МАИ *Синявский Андрей.*

Условие

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

Вариант №7

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.

Формат входных данных

В первой строке заданы $1 \leq n \leq 2000$ и $1 \leq m \leq 10000$. 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 – искомую величину максимального потока. Если пути из истока в сток не существует, данная величина равна нулю.

Метод решения

Будем хранить взвешенный граф с обратными рёбрами. На каждом шаге алгоритма находим кратчайший путь из источника в сток (поиском вширину), пускаем по нему максимально возможный поток, то есть находим эту величину, уменьшаем пропускную способность каждого ребра в пути на эту величину, а у обратных рёбер увеличиваем. Повторяем, пока существует путь (с ненулевым потоком)

Описание программы

Программа реализует класс ориентированного взвешенного графа, одним из методов которого и будет нахождение максимального потока. Храним граф следующим образом:

рёбра и их веса хранятся отдельно. Для рёбер заводим вектор списков, каждый елемент соответсвует начальному узлу (соответствие по индексу), а список, хранящийся в этом элементе - узлам на концах рёбер, вышедших из i-го узла. Веса хранятся в матрице смежности для упрощения доступа (не надо бегать лишний раз по спискам).

Метод	Назначение
ow_graph(std::istream& in)	конструктор, считывает информацию из потока ввода,и создаёт объект
bfs()	поиск вширину пути из источника в сток. Запоминает пути в поле-векторе path
flow_increase()	по пути вычисляет максимальный поток по нему, возвращает его и "пускает по рёбрам корректируя их пропускную способность
max_flow()	вычисляет максимальный поток в графе

Дневник отладки

При создании этой таблицы была использована история посылок.

Время	ведрикт	Описание
2020/10/27 23:52:35 18:50:32	Неправильный ответ	неопределённая ошибка
2020/10/28 13:11:04	Неправильный ответ	неопределённая ошибка
2020/10/28 21:26:38	Неправильный ответ	неопределённая ошибка
2020/10/29 12:04:25	Неправильный ответ	неопределённая ошибка
2020/10/29 12:22:38	Неправильный ответ	неопределённая ошибка
2020/10/29 13:36:34	Неправильный ответ	неопределённая ошибка
2020/10/29 13:54:31	Неправильный ответ	неопределённая ошибка
2020/10/29 14:00:36	Неправильный ответ	неопределённая ошибка
2020/10/29 14:04:04	Неправильный ответ	неопределённая ошибка
2020/10/29 14:22:32	Неправильный ответ	Я определил ошибку. Неверно записывался путь от источника к стоку
2020/10/29 15:19:33	Ожидает подтверждения	ура

листинг

main.cpp

```
#include <iostream>
#include "OW_graph.h"

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);

    ow_graph graph(std::cin);
    std::cout << graph.max_flow() << '\n';
```

```
    return 0;
}
```

OW_graph.h

```
#ifndef DA_LAB9_OW_GRAPH_H
#define DA_LAB9_OW_GRAPH_H

#include <vector>
#include <list>
#include <iostream>
#include <limits>

typedef unsigned long long ull;

class ow_graph {
public:
    explicit ow_graph(std::istream& in);
    ull flow_increase();
    bool bfs();
    ull max_flow();
    std::vector<int> path;
private:
    std::vector<std::list<int>> edges;
    std::vector<std::vector<ull>> g_matrix;
    int sink;
};

#endif //DA_LAB9_OW_GRAPH_H
```

OW_graph.cpp

```
#include "OW_graph.h"
#include <queue>
#include <algorithm>

ow_graph::ow_graph(std::istream& in) {
    int n, m, begin, end;
    ull cap;
    in >> n >> m;
    sink = n;
```

```
    g_matrix.resize(sink+1);
    for (int i = 0; i < n+1; ++i) {
        g_matrix[i].resize(sink+1);
    }
    edges.resize(n+1);
    path.resize(n+1);
    for (int i = 0; i < m; i++) {
        in >> begin >> end;
        in >> cap;
        edges[begin].push_back(end);
        g_matrix[begin][end] = cap;
        edges[end].push_back(begin);
    }
}

bool ow_graph::bfs() {
    bool visited[sink+1];
    std::fill(visited, visited + sink + 1, false);
    std::queue<int> queue;
    visited[1] = true;
    path[1] = -1;
    queue.push(1);
    while (!queue.empty()) {
        int v = queue.front();
        queue.pop();
        for (auto & next_vertice : edges[v]) {
            if (!visited[next_vertice] && g_matrix[v][next_vertice]) {
                visited[next_vertice] = true;
                queue.push(next_vertice);
                path[next_vertice] = v;
            }
        }
        if (visited[sink]) break;
    }
    return visited[sink];
}

ull ow_graph::flow_increase() {
    int u;
    ull res = std::numeric_limits<unsigned long long>::max();
    for (int v = sink; v != 1; v = path[v]) {
        u = path[v];
```

```
        res = std::min(res, g_matrix[u][v]);
    }
    for (int v = sink; v != 1; v = path[v]) {
        u = path[v];
        g_matrix[u][v] -= res;
        g_matrix[v][u] += res;
    }
    return res;
}

ull ow_graph::max_flow() {
    ull res = 0;
    while(bfs()) {
        res += flow_increase();
    }
    return res;
}
```

Недочёты

При вычислении максимального потока программа перезаписывает исходный граф, чего по логике происходить не должно. Так как это требуется сделать лишь один раз, на работу программы это не влияет, но всё же метод стоило сделать константным

Выводы

Проделав данную работу, я изучил алгоритм Форда-Фалкерсона, написал свою первую реализацию ориентированного взвешенного графа, вспомнил теорию графов.