

СПбПУ Петра Великого

Отчёт по дисциплине «Дискретная математика»

Лабораторная №3

Илья Козлов
23631/2

Contents

Задача.....	3
Формализация.....	3
Решение задачи.....	3
Реализация класса	3
Хранимая информация	3
Построение графа.....	4
Построение таблицы расстояний.....	4
Вывод таблицы	5
Примеры.....	6
Замечания	7
Источники	7

Задача

Имеется схема расположения автостанций, связанных дорогами известной длины. Одна из станций помечена, как «Центральная». Требуется выдать таблицу (кратчайших) расстояний от «Центральной» станции до всех остальных.

Формализация

Мы имеем схему дорог известной длины, что сводится к взвешенному графу, в узлах которого стоят автостанции, ребра которого – те самые дороги, а веса – длины дорог.

Требуется выдать таблицу кратчайших расстояний от «Центральной» станции до всех остальных, то есть необходимо просчитать кратчайшие пути от одного узла до всех остальных, для чего оптимально использовать алгоритм Дейкстры.

Итак, в итоге нам достаточно построить взвешенный граф и найти с помощью алгоритма Дейкстры кратчайшие пути во все узлы. Для этого будем считать, что на вход в программу подается N – число автостанций (узлов в графе) и далее матрица размера $N \times N$ длин дорог между автостанциями (она же – матрица весов).

Решение задачи

Реализация класса

Для решения задачи я реализовал следующий класс

```
class scheme
{
private:
    int **weights;
    int num;
    int *len, *pred;

    bool empty(std::string & line, const char *delim);
    svec split(std::string & line, const char *delim);
    void relax(int s, int v, int u);
public:
    scheme(std::ifstream & file);

    void buildTable(int central);
    void print();

    ~scheme();
}; // end of 'graph' class
```

Хранимая информация

`int **weights` — матрица весов/длин дорог

`int num` — количество узлов/автостанций

`int *len` — массив длин кратчайших путей от одного конкретного узла (Центральная автостанция) до всех остальных

`int *pred` — массив предшествующих узлов, по которому легко восстановить все вышеуказанные кратчайшие пути до узлов.

Построение графа

Построение происходит в конструкторе следующим образом — читаем из файла `N` — количество узлов и далее считываем матрицу весов, это и будет нашим графом. Массив `len` забивается значением `INF = INT_MAX`, что символизирует «бесконечность», а массив `pred` нулями, что означает, что у `i`-того узла нет предшествующего, то есть еще не был построен ни один путь.

Построение таблицы расстояний

С помощью следующего метода заполняются вектора наименьших длин и предшествующих узлов:

```
void scheme::buildTable(int central)
{
    int s = central - 1;
    len[s] = 0;           // source
    pred[s] = -1;
    vector<pair<int, int>> v; // array of pairs [vertex, length to s]

    // init
    for (int i = 0; i < num; i++)
        v.push_back(pair<int, int>(i, len[i]));

    auto cmp = [](const pair<int, int> & a, const pair<int, int> & b){return a.second <
b.second;};
    vector<pair<int, int>>::iterator it;

    while (!v.empty())
    {
        sort(v.begin(), v.end(), cmp); // getting vertex with
        int u = v[0].first;           // least length
        v.erase(v.begin());

        if (len[u] == INF)
            return;
        for (int i = 0; i < num; i++)
            if (weights[i][u] == 0) // i isnt adjacent
                continue;
            else if ((it = find(v.begin(), v.end(), pair<int, int>(i, len[i]))) != v.end()) //
i is inside v
            {
                relax(s, i, u);
                *it = pair<int, int>(i, len[i]);
            }
    }
} // End of 'scheme::buildTable' function
```

Пояснение: в качестве аргумента принимается номер узла, который помечен как «Центральный» (далее - источник).

`s` — индекс центрального узла.

Считаем, что длина до центрального узла = 0, поэтому помещаем в массив len 0 на место источника. Предшествующий узел источника будем считать за -1 (сделано для удобства последующего вывода).

`vector<pair<int, int>> v` – вектор пар чисел: номер вершины-расстояние до источника.

Метод работает следующим образом – заполняем контейнер `v` всеми вершинами нашего графа (добавляем пары: вершина-расстояние до источника). Пока контейнер не пуст, будем брать вершину с минимальным расстоянием до источника, для этого сортируем вектор пар по второй компоненте расстояния и берем первую вершину, после чего удаляем ее из вектора. Если длина от полученной вершины до источника бесконечна, то можно прервать наш цикл в виду того, что и все оставшиеся вершины имеют бесконечную длину, так как мы сортировали и брали по наименьшей. Иначе ищем смежные вершины и, если они все еще находятся в контейнере `v`, вызываем метод релаксации, переназначающий длины путей и предшествующий узел, если через вершину `u` идти быстрее. Не забываем переназначить измененный элемент вектора `v`. Этот метод называется алгоритмом Дейкстры. Таким образом, по окончании цикла, у нас будут заполнены вектора кратчайших длин до нашей центральной станции и для каждого узла будет известен его предшествующий на пути от того же центрального узла, что и было необходимо для составления таблицы расстояний

Вывод таблицы

Вывод производится вызовом метода `print` в следующем формате:

```
D:\SPbPU\discretka\Lab3\Debug\Lab3.exe
nodes: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
length: | 5 | 4 | 7 | 3 | 0 | 1 | 6 | 8 | 12 | 10 |
ways:   | 4 | 6 | 6 | 6 | 0 | 5 | 5 | 6 | 5 | 7 |
```

`nodes` – номера узлов/станций в графе

`length` – длины кратчайший расстояний до центральной станции/узла, где нулем обозначена центральная станция


`ways` – предшествующие узлы для каждого узла на пути к нему от центрального

Примеры

Матрица весов:


$$\begin{pmatrix} 0 & 3 & 4 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 5 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 1 & 6 & 0 & 12 & 0 \\ 0 & 3 & 6 & 2 & 1 & 0 & 8 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6 & 8 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 & 7 & 0 & 0 & 6 & 3 \\ 0 & 0 & 0 & 0 & 12 & 0 & 0 & 6 & 0 & 11 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 11 & 0 \end{pmatrix}$$

Результат для центрального 1 узла:




nodes:		1		2		3		4		5		6		7		8		9		10	
length:		0		3		4		2		5		4		11		11		17		14	
ways:		0		1		1		1		6		4		5		6		5		8	

Результат для центрального 5 узла:




nodes:		1		2		3		4		5		6		7		8		9		10	
length:		5		4		7		3		0		1		6		8		12		10	
ways:		4		6		6		6		0		5		5		6		5		7	

Результат для центрального 7 узла:



nodes:		1		2		3		4		5		6		7		8		9		10	
length:		11		10		13		9		6		7		0		7		13		4	
ways:		4		6		6		6		7		5		0		10		8		7	

Результат для центрального 10 узла:




nodes:		1		2		3		4		5		6		7		8		9		10	
length:		14		13		16		12		10		10		4		3		9		0	
ways:		4		6		6		6		7		8		10		10		8		0	

Матрица весов:

$$\begin{pmatrix} 0 & 7 & 0 & 10 & 1 \\ 7 & 0 & 3 & 0 & 4 \\ 0 & 3 & 0 & 5 & 0 \\ 10 & 0 & 5 & 0 & 15 \\ 1 & 4 & 0 & 15 & 0 \end{pmatrix}$$

Результат для 1 узла:

 D:\SPbPU\discretka\Lab3\x64\Debug\Lab3.exe

nodes:		1		2		3		4		5	

length:		0		5		8		10		1	

ways:		0		5		2		1		1	

Результат для 5 узла:

nodes:		1		2		3		4		5	

length:		1		4		7		11		0	

ways:		5		5		2		1		0	

Замечания

В классе было дополнительно реализовано построение вектора предшествоющих узлов, это не являлось необходимым в задаче, просто для интереса.

Источники

- Новиков Ф.А. «Дискретная математика», 3-е издание
- Кормен Томас, Лейзерсон Чарльз, Ривест Рональд, Штайн Клиффорд «Алгоритмы: построение и анализ», 3-е издание