МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Санкт-Петербургский национальный исследовательский университет

информационных технологий, механики и оптики»

Факультет информационных технологий и программирования

Кафедра информационных систем

Лабораторная работа №2

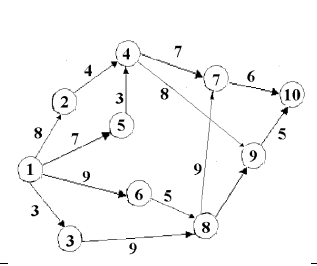
**Динамическое программирование**

Выполнил студент группы № М3307:   
Бойцов Виталий Вячеславич

Санкт-Петербург

2018

## Вариант 3.



## Результирующий массив расстояний

1 2 3 4 5 6 7 8 9 10  
0 8 3 10 7 9 17 12 12 17 - fordbellman  
0 8 3 10 7 9 17 12 12 17 - deijkstra

## Вывод

В результате мы получили одинаковый результат для всех видов подсчёта, т.к. любой правильно написанный метод динамического программирования в конце должен дать эквивалентный результат для всех подзадач, независимо от того, каким способом получался этот результат (если этот алгоритм действительно отвечает всем критериям, например, не допускает множественных решений).

1. Для каких оптимизационных задач применяется метод динамического программирования.   
   Для задач, которые могут быть разбиты на подзадачи и ответ для каждой новой подзадачи будет считаться через предыдущие посчитанные подзадачи.
2. В чём заключается суть метода динамического программирования.   
   Считать ответ через ответы для ранее посчитанных ответов для подзадач.  
   Для реализации задачи на динамическое программирование надо подумать над 5ью вопросами: что храним; формула перехода; порядок обхода; база; где находится ответ.
3. Сформулируйте принцип оптимальности Беллмана.

Оптимальное поведение обладает тем свойством, что каковы бы ни были начальное состояние и начальное решение, последующие решения должны составлять оптимальное поведение относительно состояние, полученного в результате начального решения.

1. Что является целевой функцией в задаче о кратчайшем маршруте?  
   Расстояние до вершины из начальной вершины.
2. Какой параметр определяет состояние системы на каждом шаге?  
   Набор посчитанных расстояний и набор рассмотренных ребёр.

## main.cpp

#include "deijkstra.h"

#include "fordbellman.h"

#include <iostream>

using namespace *std*;

*vector*<Edge> edges = {

{ 1, 2, 8 },

{ 1, 5, 7 },

{ 1, 6, 9 },

{ 1, 3, 3 },

{ 2, 4, 4 },

{ 5, 4, 3 },

{ 6, 8, 5 },

{ 3, 8, 9 },

{ 4, 7, 7 },

{ 4, 9, 8 },

{ 8, 7, 9 },

{ 8, 9, 0 },

{ 9, 10, 5 },

{ 7, 10, 6 }

};

int *main*() {

int n = 0;

for (auto & e : edges) {

n = *max*(n, e.from);

e.from--;

n = *max*(n, e.to);

e.to--;

}

*vector*<int> ans1 = fordbellman(n, edges, 0);

*vector*<*vector*<*pair*<int, int>>> graph(n);

for (auto e : edges)

graph[e.from].*push\_back*({ e.to, e.cost });

*vector*<int> ans2 = deijkstra(graph, 0);

for (int i = 1; i <= n; i++)

*printf*("%3d ", i);

*cout* << '\n';

for (auto d : ans1)

*printf*("%3d ", d);

*cout* << " - fordbellman \n";

for (auto d : ans2)

*printf*("%3d ", d);

*cout* << " - deijkstra \n";

## }

## deijkstra.h

#pragma once

#include <vector>

#include <algorithm>

#include <set>

using namespace *std*;

*vector*<int> deijkstra(*vector*<*vector*<*pair*<int, int>>> const& graph, int start) {

const int INF = 1e6;

*vector*<int> d(graph.*size*(), INF);

d[start] = 0;

*set*<*pair*<int, int>> q;

q.*insert*(*make\_pair*(d[start], start));

while (!q.*empty*()) {

int v = q.*begin*()->*second*;

q.*erase*(q.*begin*());

for (auto now : graph[v]) {

int to = now.*first*;

int cost = now.*second*;

if (d[v] + cost < d[to]) {

q.*erase*(*make\_pair*(d[to], to));

d[to] = d[v] + cost;

q.*insert*(*make\_pair*(d[to], to));

}

}

}

return d;  
}

## fordbellman.h

#pragma once

#include <vector>

#include <algorithm>

using namespace *std*;

struct Edge {

int from, to, cost;

};

*vector*<int> fordbellman(int n, *vector*<Edge> const& edges, int start) {

const int INF = 1e6;

*vector*<int> d(n, INF);

d[start] = 0;

for (int i = 0; i < n - 1; i++)

for (auto e : edges)

if (d[e.from] < INF)

d[e.to] = *min*(d[e.to], d[e.from] + e.cost);

return d;

}