

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

Выполнили студенты:  
Ивницкий Алексей М3305  
Шеремет Сергей М3305  
Шипкова Мария М3303

Проверил:  
Москаленко Мария Александровна

САНКТ-ПЕТЕРБУРГ  
2018

# Динамическое программирование

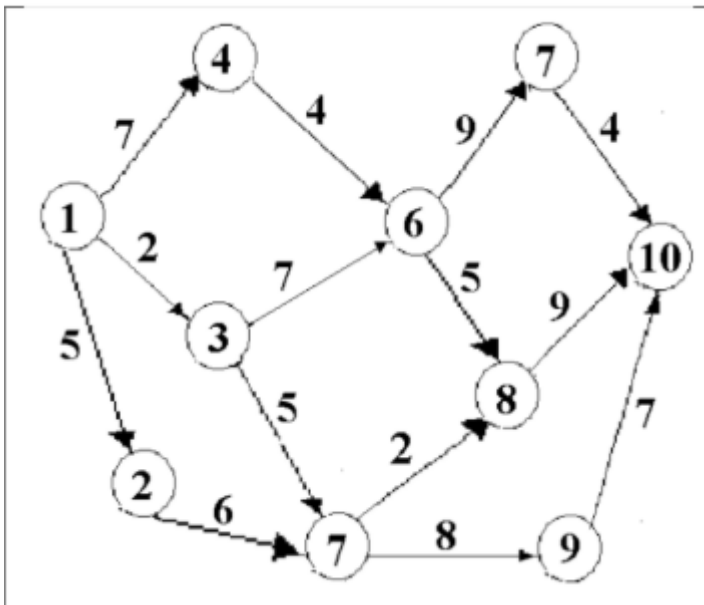
---

## Цель работы

Изучить теорию и методы решения задач динамического программирования; приобрести навыки решения задач динамического программирования на ЭВМ.

## Задача

Рассмотрим тривиальную задачу динамического программирования. Пусть требуется перевезти груз из города в город. Сеть дорог, связывающая эти города, изображена в виде графа:



города – вершины графа, а ребра – дороги между ними.

## Алгоритм решения

Проанализируем входной граф. Зная, что у нас есть строго одна начальная и конечная точка, мы можем реализовать решение используя модифицированный BreadthFirstSearch. Начиная с первой точке, делаем рекурсивно проверку: находить все точки в которые мы можем попасть с текущей ( $E[i][j] \neq 0$ , где  $i$  - точка, из которой мы идем,  $j$  - точка, которую мы проверяем) и которые мы еще не начали проверять. Если таких точек нет – значит мы находимся в конечной точке, а значит расстояние от нее к конечной точке  $dist[i] = 0$ , где  $dist[i]$  - расстояние от  $i$  до конечной точки. Для всех других точек начиная с последней итерации рекурсии будем находить для точки  $i$ :  $min(E[i][j] + dist[j])$ .

## Результат

Результат выполнения кода для данного графа:

```
10 - destination
Optimal way: 8 => 10
```

Optimal way: 9 => 10  
Optimal way: 7 => 10  
Optimal way: 5 => 8  
Optimal way: 6 => 7  
Optimal way: 2 => 5  
Optimal way: 3 => 5  
Optimal way: 4 => 6  
Optimal way: 1 => 3  
 $d(1) = 18$   
 $d(2) = 17$   
 $d(3) = 16$   
 $d(4) = 17$   
 $d(5) = 11$   
 $d(6) = 13$   
 $d(7) = 4$   
 $d(8) = 9$   
 $d(9) = 7$   
 $d(10) = 0$

Ручной расчет:

10 - конечная точка

$\text{dist}[9] = E[9, 10] = 7$

$\text{dist}[8] = E[8, 10] = 9$

$\text{dist}[7] = E[7, 10] = 4$

$\text{dist}[6] = \min(E[6, 7] + \text{dist}[7], E[6, 8] + \text{dist}[8]) = \min(9 + 4, 5 + 9) = 13$

$\text{dist}[5] = \min(E[5, 8] + \text{dist}[8], E[5, 9] + \text{dist}[9]) = \min(2 + 9, 8 + 7) = 11$

$\text{dist}[4] = E[4, 6] + \text{dist}[6] = 17$

$\text{dist}[3] = \min(E[3, 6] + \text{dist}[6], E[3, 5] + \text{dist}[5]) = \min(7 + 13, 5 + 11) = 16$

$\text{dist}[2] = E[2, 5] + \text{dist}[5] = 6 + 11 = 17$

$\text{dist}[1] = \min(E[1, 2] + \text{dist}[2], E[1, 3] + \text{dist}[3], E[1, 4] + \text{dist}[4]) = \min(17 + 5, 16 + 2, 17 + 7) = 18$

Вывод: Данный алгоритм позволяет оптимально находить расстояние между вершинами в графах заданного типа. Ручной расчет совпал с выводом алгоритма, что подтверждает его работоспособность.