



**Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ «Информатика и системы управления» (ИУ)

КАФЕДРА «Информационная безопасность» (ИУ8)

**Отчёт**

по лабораторной работе № 6  
по дисциплине «Теория систем и системный анализ»

**Тема: «Построение сетевого графа работ и его анализ методом  
критического пути (СРМ)»**

Вариант 10

Выполнил:  
Минькова А.А., студент  
группы ИУ8-31

Проверил: Коннова Н.С.,  
доцент каф. ИУ8

г. Москва,  
2020 г.

## 1. Цель работы

Изучить задачи сетевого планирования в управлении проектами и приобрести навыки их решения при помощи метода критического пути.

## 2. Условие задачи

Вариант № 10.

Задан набор работ с множествами непосредственно предшествующих работ (по варианту).

1. Построить сетевой граф, произвести его топологическое упорядочение и нумерацию.
2. Рассчитать и занести в таблицу поздние сроки начала и ранние сроки окончания работ.
3. Рассчитать и занести в таблицу ранние и поздние сроки наступления событий.
4. Рассчитать полный и свободный резервы времени работ.
5. Рассчитать резерв времени событий, определить и выделить на графе критический путь.

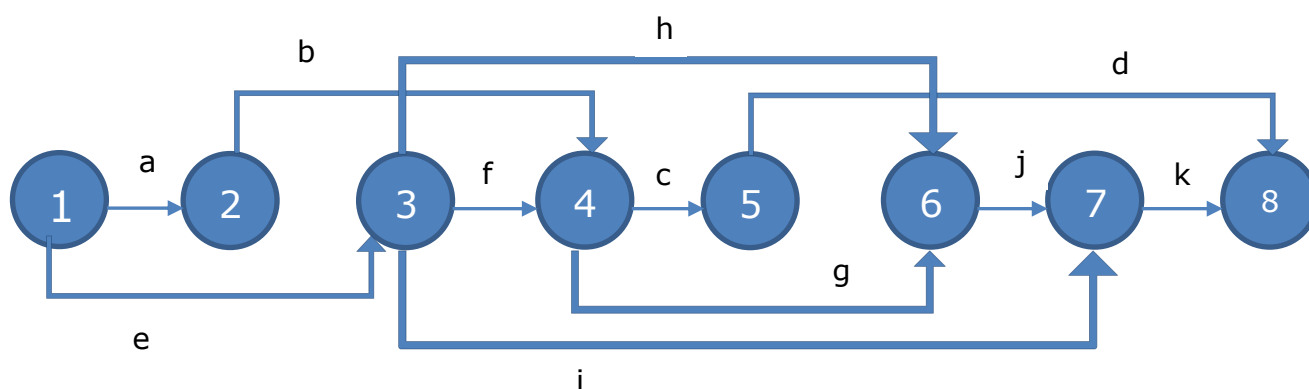


Таблица 1. Длительность работ.

	a	b	c	d	e	f	g	h	i	j	k
t	3	5	2	4	3	1	4	3	3	2	5

Таблица 2. Множества предшествующих работ.

$P_a$	$P_b$	$P_c$	$P_d$	$P_e$	$P_f$	$P_g$	$P_h$	$P_i$	$P_j$	$P_k$
-	a	b, f	c	-	e	b, f	e	e	g, h	i, j

### 3. Таблицы

Таблица 3. Параметры событий.

Number	1	2	3	4	5	6	7	8
$T_j^p$	0	3	3	4	6	6	6	10
$T_j^n$	0	3	3	8	10	12	14	19
$R_j = T_j^p - T_j^n$	0	0	0	4	4	6	8	9

Таблица 4. Параметры работ.

	$t_{ij}$	$t_{ij}^{po}$	$t_{ij}^{nn}$	$r_{ij}^c$	$r_{ij}^n$
<b>1-2</b>	3	3	0	0	0
<b>1-3</b>	3	3	0	0	0
<b>2-4</b>	5	8	3	-4	0
<b>3-4</b>	1	4	7	0	4
<b>3-6</b>	3	6	9	0	6
<b>3-7</b>	3	6	11	0	8
<b>4-5</b>	2	6	8	-4	4
<b>4-6</b>	4	8	8	-6	4
<b>5-8</b>	4	10	15	-4	9
<b>6-7</b>	2	8	12	-8	6
<b>7-8</b>	5	11	14	-9	8

## 4. Результат работы программы

С помощью алгоритма Флойда была найдена длина критического пути:

```
void FloydAlgorithm(std::vector<std::vector<int>> &m){  
    for (int k = 0; k < m.size(); k++) {  
        for (int i = 0; i < m.size(); i++) {  
            for (int j = 0; j < m.size(); j++) {  
                if (i != j && m[i][k] != -1 && m[k][j] != -1) {  
                    if (m[i][j] == -1) {  
                        m[i][j] = m[i][k] + m[k][j];  
                    }  
                    else m[i][j] = std::max(m[i][j], m[i][k] + m[k][j]);  
                }  
            }  
        }  
    }  
}
```

Вывод программы:

```
0  3  3  2 -9 -9 -9 -9 -9  
-9  0 -9  5 -9 -9 -9 -9 -9  
-9 -9  0  1 -9  3  3 -9 -9  
-9 -9 -9  0  2  4 -9 -9 -9  
-9 -9 -9 -9  0 -9 -9  4 -9  
-9 -9 -9 -9 -9  0  2 -9 -9  
-9 -9 -9 -9 -9 -9  0  5 -9  
-9 -9 -9 -9 -9 -9 -9  0 -9
```

```
0 13 13 14 16 12 16 19 -9  
7  0 10 11 13  9 13 16 -9  
3  6  0  7  9  5  9 12 -9  
2  5  5  0  8  4  8 11 -9  
-5 -2 -2 -1  0 -1  1  4 -9  
-2  1  1  2  4  0  4  7 -9  
-4 -1 -1  0  2 -1  0  5 -9  
-9 -6 -6 -5 -3 -1 -3  0 -9
```

## **5. Выводы**

В результате работы были изучены задачи сетевого планирования в управлении проектами и приобретены навыки их решения при помощи метода критического пути.

## **6. Ответ на контрольный вопрос**

1. Какие основные данные необходимы для использования метода критического пути?

Для использования метода критического пути необходимы следующие данные:

список работ проекта; связи между работами; длительность выполнения работ; календарный план рабочего времени; календарный срок начала проекта.

## Приложение 1. Исходный код программы

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <iomanip>

void Print(const std::vector<std::vector<int>>& m){
    for (const auto& k: m){
        for (auto v: k){
            std::cout << std::setw(3) << v;
        }
        std::cout << std::endl;
    }
    std::cout << std::endl;
}

void FloydAlgoritm(std::vector<std::vector<int>> &m){
    for (int k = 0; k < m.size(); k++) {
        for (int i = 0; i < m.size(); i++) {
            for (int j = 0; j < m.size(); j++) {
                if (i != j && m[i][k] != -1 && m[k][j] != -1) {
                    if (m[i][j] == -1) {
                        m[i][j] = m[i][k] + m[k][j];
                    }
                    else m[i][j] = std::max(m[i][j], m[i][k] + m[k][j]);
                }
            }
        }
    }
}

int main() {
    int INF = -9;
    std::vector<std::vector<int>> adjMatrix = {
        {0, 3, 3, 2, INF, INF, INF, INF, INF},
        {INF, 0, INF, 5, INF, INF, INF, INF, INF},
        {INF, INF, 0, 1, INF, 3, 3, INF, INF},
        {INF, INF, INF, 0, 2, 4, INF, INF, INF},
        {INF, INF, INF, INF, 0, INF, INF, 4, INF},
        {INF, INF, INF, INF, INF, 0, 2, INF, INF},
        {INF, INF, INF, INF, INF, INF, 0, 5, INF},
        {INF, INF, INF, INF, INF, INF, INF, 0, INF}
    };
    Print(adjMatrix);
    FloydAlgoritm(adjMatrix);
    Print(adjMatrix);
}
```