## 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 (CPM)»**

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.

1. **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.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *Pa* | *Pb* | *Pc* | *Pd* | *Pe* | *Pf* | *Pg* | *Ph* | *Pi* | *Pj* | *Pk* |
|  | - | a | b, f | c | - | e | b, f | е | е | g, h | i, j |

**3.** **Таблицы**

Таблица 3. Параметры событий.

**Number**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| 0 | 3 | 3 | 4 | 6 | 6 | 6 | 10 |
| 0 | 3 | 3 | 8 | 10 | 12 | 14 | 19 |
| 0 | 0 | 0 | 4 | 4 | 6 | 8 | 9 |

*p j*

*T*

*п j*

*T*

*R* =*Tp*−*T*

*j j*

Таблица 4. Параметры работ.

*tij t ро t пн rc rп*

*ij ij ij ij*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **1-2** | 3 | 3 | 0 | 0 | 0 |
| **1-3** | 3 | 3 | 0 | 0 | 0 |
| **2-4** | 5 | 8 | 3 | -4 | 0 |
| **3-4** | 1 | 4 | 7 | 0 | 4 |
| **3-6** | 3 | 6 | 9 | 0 | 6 |
| **3-7** | 3 | 6 | 11 | 0 | 8 |
| **4-5** | 2 | 6 | 8 | -4 | 4 |
| **4-6** | 4 | 8 | 8 | -6 | 4 |
| **5-8** | 4 | 10 | 15 | -4 | 9 |
| **6-7** | 2 | 8 | 12 | -8 | 6 |
| **7-8** | 5 | 11 | 14 | -9 | 8 |
|  |  |  |  |  |  |

# Результат работы программы

С помощью алгоритма Флойда была найдена длина критического пути:

Изображение выглядит как текст

Автоматически созданное описание

Вывод программы:

Изображение выглядит как стол

Автоматически созданное описание

* 1. **Выводы**

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

* 1. **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);  
}