|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 4

Выполнил: Велинецкий А. В.,

студент группы ИУ8-32

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. Постановка задачи**

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

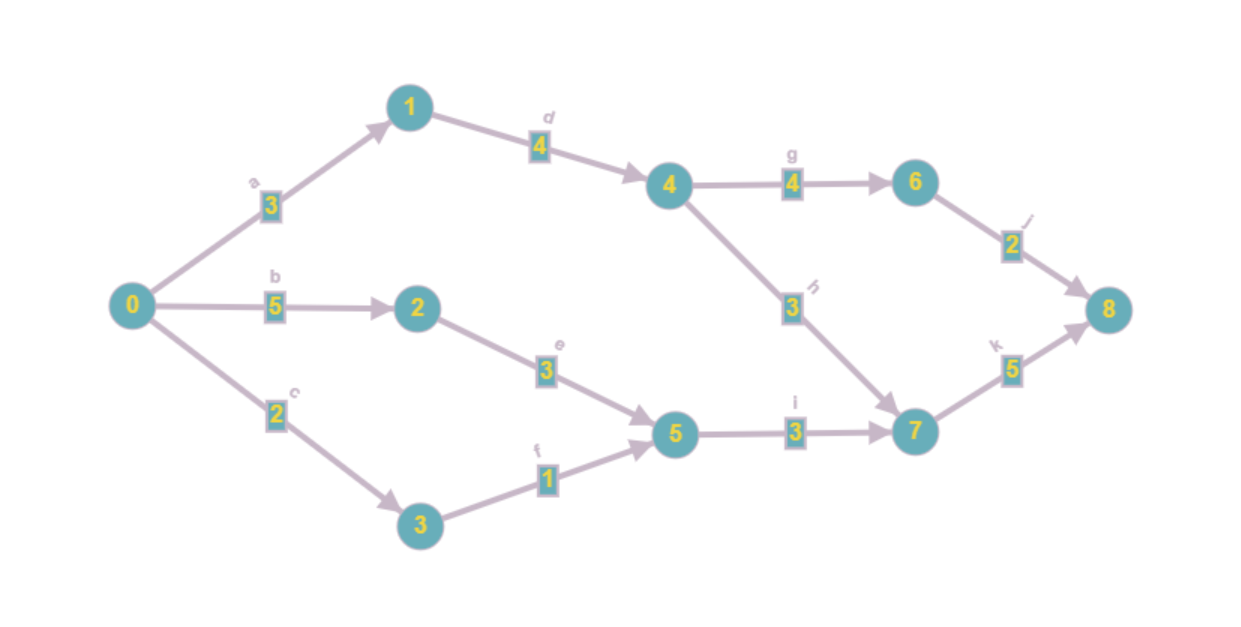
1. Построить сетевой граф, произвести его топологическое упорядочение и нумерацию.
2. Рассчитать и занести в таблицу поздние сроки начала и ранние сроки окончания работ.
3. Рассчитать и занести в таблицу ранние и поздние сроки наступления событий.
4. Рассчитать полный и свободный резервы времени работ.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

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

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № пп |  |  |  |  |  |  |  |  |  |  |  |
| 4 | - | - | - | a | b | c | d | d | e, f | g | h, i |

**3. Ход работы**

**Рис. 1** Топологически упорядоченный и пронумерованный граф



**Табл. 1** Параметры работ.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № |  |  |  |  |  |
| 0-1 | 3 | 3 | 1 | 0 | 1 |
| 0-2 | 5 | 5 | 0 | 0 | 0 |
| 0-3 | 2 | 2 | 7 | 0 | 5 |
| 1-4 | 4 | 7 | 4 | 0 | 1 |
| 2-5 | 3 | 8 | 5 | 0 | 0 |
| 3-5 | 1 | 3 | 7 | 0 | 5 |
| 4-6 | 4 | 11 | 10 | 0 | 3 |
| 4-7 | 3 | 10 | 8 | 0 | 1 |
| 5-7 | 3 | 11 | 8 | 0 | 0 |
| 6-8 | 2 | 13 | 14 | 0 | 3 |
| 7-8 | 5 | 16 | 11 | 0 | 0 |

# Табл. 2 Параметры событий.

|  |  |  |  |
| --- | --- | --- | --- |
| № |  |  |  |
| 0 | 0 | 0 | 0 |
| 1 | 3 | 4 | 1 |
| 2 | 5 | 5 | 0 |
| 3 | 2 | 7 | 5 |
| 4 | 7 | 8 | 1 |
| 5 | 8 | 8 | 0 |
| 6 | 11 | 14 | 3 |
| 7 | 11 | 11 | 0 |
| 8 | 16 | 16 | 0 |

Найдем критический путь с помощью алгоритма Флойда.

Матрица смежности до начала работы алгоритма (в ней -1 означает отсутствие ребра между вершинами):

0 3 5 2 -1 -1 -1 -1 -1

-1 0 -1 -1 4 -1 -1 -1 -1

-1 -1 0 -1 -1 3 -1 -1 -1

-1 -1 -1 0 -1 1 -1 -1 -1

-1 -1 -1 -1 0 -1 4 3 -1

-1 -1 -1 -1 -1 0 -1 3 -1

-1 -1 -1 -1 -1 -1 0 -1 2

-1 -1 -1 -1 -1 -1 -1 0 5

-1 -1 -1 -1 -1 -1 -1 -1 0

Матрица после работы алгоритма:0 3 5 2 7 8 11 11 16

-1 0 -1 -1 4 -1 8 7 12

-1 -1 0 -1 -1 3 -1 6 11

-1 -1 -1 0 -1 1 -1 4 9

-1 -1 -1 -1 0 -1 4 3 8

-1 -1 -1 -1 -1 0 -1 3 8

-1 -1 -1 -1 -1 -1 0 -1 2

-1 -1 -1 -1 -1 -1 -1 0 5

-1 -1 -1 -1 -1 -1 -1 -1 0

После проведения работы 2-х алгоритмов видно, что длина критического пути – 16. Критическим является следующий путь: 0 – 2 – 5 – 7 – 8

**4. Выводы**

В ходе проделанной работы были изучены задачи сетевого планирования в управлении проектами и приобретены навыки их решения при помощи метода критического пути. Также был изучен алгоритм Флойда, результаты которого совпали с аналитическими данными.

**5. Контрольные вопросы**

2. Какие исходные данные необходимы для использования метода критического пути?

* список всех операций, необходимых для выполнения проекта;
* зависимости между этими операциями;
* период времени, необходимый для выполнения каждой операции (длительность, вес).

**Приложение 1. Исходный код программы. Файл main.cpp**

#include <iostream>  
#include <vector>  
#include <algorithm>  
#include <iomanip>  
  
using namespace std;  
  
void print(const vector<vector<int>>& m){  
 for (const auto& i: m){  
 for (auto f: i){  
 cout << setw(3) << f;  
 }  
 cout << endl;  
 }  
 cout << endl;  
}  
  
void FloydAlg(vector<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] = max(m[i][j], m[i][k] + m[k][j]);  
 }  
 }  
 }  
 }  
}  
  
int main() {  
 int INF = -1;  
 vector<vector<int>> adjMatrix = {  
 { 0, 3, 5, 2, INF, INF, INF, INF, INF},  
 {INF, 0, INF, INF, 4, INF, INF, INF, INF},  
 {INF, INF, 0, INF, INF, 3, INF, INF, INF},  
 {INF, INF, INF, 0, INF, 1, INF, INF, INF},  
 {INF, INF, INF, INF, 0, INF, 4, 3, INF},  
 {INF, INF, INF, INF, INF, 0, INF, 3, INF},  
 {INF, INF, INF, INF, INF, INF, 0, INF, 2},  
 {INF, INF, INF, INF, INF, INF, INF, 0, 5},  
 {INF, INF, INF, INF, INF, INF, INF, INF, 0}  
 };  
 print(adjMatrix);  
 FloydAlg(adjMatrix);  
 print(adjMatrix);  
}

**Приложение 2. Ход работы программы.**

