**Министерство образования Российской Федерации**

**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**им. Н.Э. БАУМАНА**

Факультет: Информатика и системы управления Кафедра: Информационная безопасность (ИУ8)

МЕТОДЫ ОПТИМИЗАЦИИ

Лабораторная работа №6 на тему:

«**Построение сетевого графа работ и его анализ методом критического пути (CPM)**»

Вариант 5

**Преподаватель:**

Коннова Н. С.

**Студент**:

Григорьев А. С.

**Группа:**

ИУ8-32

Москва 2020

# Цель работы

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

# Постановка задачи

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

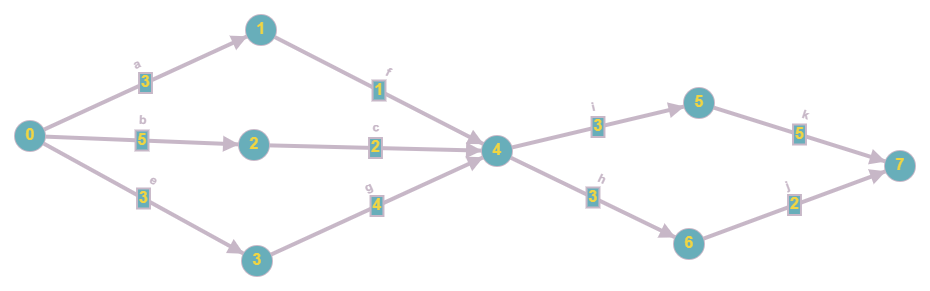
1. Построить сетевой граф, произвести его топологическое упорядочение и нумерацию.
2. Рассчитать и занести в таблицу поздние сроки начала и ранние сроки окончания работ.
3. Рассчитать и занести в таблицу ранние и поздние сроки наступления событий.
4. Рассчитать полный и свободный резервы времени работ.
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 |  |  |  |  |  |  |  |  |  |  |  |
| 5 | - | - | b | b | - | a | e, d | f, c, g | f, c, g | h | i |

# Ход работы

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



**Табл. 1** Параметры работ.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № |  |  |  |  |  |
| 0-1 | 3 | 3 | 3 | 0 | 3 |
| 0-2 | 5 | 5 | 0 | 0 | 0 |
| 0-3 | 3 | 3 | 0 | 0 | 0 |
| 1-4 | 1 | 4 | 6 | 0 | 3 |
| 2-4 | 2 | 7 | 5 | 0 | 0 |
| 3-4 | 4 | 7 | 3 | 0 | 0 |
| 4-5 | 3 | 10 | 7 | 0 | 0 |
| 4-6 | 3 | 10 | 10 | 0 | 3 |
| 5-7 | 5 | 15 | 10 | 0 | 0 |
| 6-7 | 2 | 12 | 13 | 0 | 3 |

# Табл. 2 Параметры событий.

|  |  |  |  |
| --- | --- | --- | --- |
| № |  |  |  |
| 0 | 0 | 0 | 0 |
| 1 | 3 | 6 | 3 |
| 2 | 5 | 5 | 0 |
| 3 | 3 | 3 | 0 |
| 4 | 7 | 7 | 0 |
| 5 | 10 | 10 | 0 |
| 6 | 10 | 13 | 3 |
| 7 | 15 | 15 | 0 |

Найдем критический путь с помощью алгоритма Флойда.

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

Матрица после работы алгоритма:

Видно, что длина критического пути – 15. Критическими являются следующие пути: 0 – 2 – 4 – 5 – 7, 0 – 3 – 4 – 5 – 7

# Выводы

В ходе проделанной работы были изучены задачи сетевого планирования в управлении проектами и приобретены навыки их решения при помощи метода критического пути. Результат работы алгоритма Флойда совпал c аналитическими результатами.

# Вопросы

1. Опишите основную идею и практическую пользу метода критического пути.

[Метод критического пути](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%BA%D1%80%D0%B8%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B3%D0%BE_%D0%BF%D1%83%D1%82%D0%B8) (МКП) — это метод планирования операций, в основе которого лежит математический алгоритм. Использование такой методики подразумевает создание модели [проекта](http://www.wrike.com/ru/project-management), включающей следующие элементы:

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

Зная эти значения, с помощью метода критического пути можно определить наиболее длительную последовательность операций, необходимую для завершения проекта, а также самые ранние и самые поздние моменты начала и окончания каждой операции, которые не приведут к задержке выполнения проекта. В процессе определяются так называемые «критические» операции (то есть лежащие на самом длинном пути), а также операции с общим временным резервом (их сроки можно передвинуть, но продолжительность проекта от этого не увеличится).

# Приложение А.

*Файл ‘lab06.cpp’.*

#include <iostream>

#include <vector>

#include <algorithm>

#include <iomanip>

int main() {

int INF = -1;

std::vector<std::vector<int>> matrix = {

{0, 3, 5, 3, INF, INF, INF, INF},

{INF, 0, INF, INF, 1, INF, INF, INF},

{INF, INF, 0, INF, 2, INF, INF, INF},

{INF, INF, INF, 0, 4, INF, INF, INF},

{INF, INF, INF, INF, 0, 3, 3, INF},

{INF, INF, INF, INF, INF, 0, INF, 5},

{INF, INF, INF, INF, INF, INF, 0, 2},

{INF, INF, INF, INF, INF, INF, INF, 0}

};

for (int i = 0; i < matrix.size(); i++) {

for (int j = 0; j < matrix.size(); j++) {

std::cout << std::setw(5) << matrix[i][j] << " ";

}

std::cout << std::endl;

}

std::cout << std::endl;

for (int k = 0; k < matrix.size(); k++) {

for (int i = 0; i < matrix.size(); i++) {

for (int j = 0; j < matrix.size(); j++) {

if (i != j && matrix[i][k] != -1 && matrix[k][j] != -1) {

if (matrix[i][j] == -1) {

matrix[i][j] = matrix[i][k] + matrix[k][j];

}

else

{

matrix[i][j] = std::max(matrix[i][j], matrix[i][k] + matrix[k][j]);

}

}

}

}

}

for (int i = 0; i < matrix.size(); i++) {

for (int j = 0; j < matrix.size(); j++) {

std::cout << std::setw(5) << matrix[i][j] << " ";

}

std::cout << std::endl;

}

std::cout << std::endl;

}