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 «Методы оптимизации»

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

Вариант 15

Выполнил: Петросян А.Р.,
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.

3. Ход работы

Для выполнения задачи был реализован двухпроходный алгоритм построения сетевого графика на языке программирования Python 3. Код программы вынесен в Приложение 1. Алгоритм последовательно вычисляет характеристики (раннее/позднее время начала/конца работы, запас по времени) для каждой из работ и находит критический путь.

Чтобы не проводить дополнительный предварительный анализ графа на число завершающих вершин, введем дополнительную фиктивную работу длительностью 0 времени, требующую завершения всех предыдущих работ.

```
a, 3,
b, 5, a,
c, 2, b,
d, 4, c;g,
e, 3,
f, 1, e,
g, 4, f;j,
h, 3, b,
i, 3, f;j,
j, 2,
k, 5, i;h,
final, 0, a;b;c;d;e;f;g;h;i;j;k
```

Рис. 1. Входные данные файла “input.txt”

	Длительность задачи	Раннее начало	Ранний конец	Позднее начало	Поздний конец	Запас	Крит. путь
a	3	0	3	0	3	0	True
b	5	3	8	3	8	0	True
c	2	8	10	10	12	2	False
d	4	10	14	12	16	2	False
e	3	0	3	4	7	4	False
f	1	3	4	7	8	4	False
g	4	4	8	8	12	4	False
h	3	8	11	8	11	0	True
i	3	4	7	8	11	4	False
j	2	0	2	6	8	6	False
k	5	11	16	11	16	0	True
final	0	16	16	16	16	0	True

Рис. 2. Результаты расчетов программы.

По итоговой таблице видно, что работы a-b-h-k входят в критический путь, а значит задержка их выполнения повлияет на время завершения всего проекта.

4. Выводы

В данной лабораторной работе был изучен метод анализа сетевого графика критическим путем. Данный метод позволяет определить те части проекта (задачи в графике), замедление в выполнении которых влечет за собой замедление выполнения всего графика, как и минимально необходимое время для выполнения всех задач. Поскольку в самой лабораторной работе не было ограничивающих условий (директивно установленное время завершения работ и

прочее), в ходе лабораторной работы критический путь был найдем из соображения, что все задачи в совокупности должны быть выполнены как можно быстрее. По этим условиям был найден список задач, составляющих собой критический путь.

Приложение 1. Исходный код программы

```
import pandas as pd
from math import inf

def min_col(frame, col):
    min_d = inf
    for e in frame:
        min_d = min(e[col], min_d)
    return min_d

line = list()
singleElement = list()
tasks = dict()
number = -1
input_file = open('input.txt')

for line in input_file:
    singleElement = (line.split(','))
    number += 1
    for i in range(len(singleElement)):
        tasks['task' + str(singleElement[0])] = dict()
        tasks['task' + str(singleElement[0])]['id'] = singleElement[0]
        tasks['task' + str(singleElement[0])]['name'] = singleElement[0]
        tasks['task' + str(singleElement[0])]['duration'] = singleElement[1]
        if singleElement[2] != "\n":
            tasks['task' + str(singleElement[0])]['dependencies'] =
singleElement[2].strip().split(';')
        else:
            tasks['task' + str(singleElement[0])]['dependencies'] = ['-1']
        tasks['task' + str(singleElement[0])]['ES'] = 0
        tasks['task' + str(singleElement[0])]['EF'] = 0
        tasks['task' + str(singleElement[0])]['LS'] = 0
        tasks['task' + str(singleElement[0])]['LF'] = 0
        tasks['task' + str(singleElement[0])]['float'] = 0
        tasks['task' + str(singleElement[0])]['isCritical'] = False

# ========================================================================
# Прямой ход
# ========================================================================
for taskFW in tasks:
    if '-1' in tasks[taskFW]['dependencies']:  # проверка если это 1я задача
        tasks[taskFW]['ES'] = 0
        tasks[taskFW]['EF'] = (tasks[taskFW]['duration'])
    else:
        for k in tasks.keys():
            for dep in tasks[k]['dependencies']:
                if dep != '-1' and len(tasks[k]['dependencies']) == 1:  # у
задачи 1 зависимость
                    tasks[k]['ES'] = int(tasks['task' + dep]['EF'])
                    tasks[k]['EF'] = int(tasks[k]['ES']) +
int(tasks[k]['duration'])
                elif dep != '-1':  # у задачи больше 1 зависимости
                    if int(tasks['task' + dep]['EF']) > int(tasks[k]['ES']):
                        tasks[k]['ES'] = int(tasks['task' + dep]['EF'])
                        tasks[k]['EF'] = int(tasks[k]['ES']) +
int(tasks[k]['duration'])
```

```
aList = list()
for element in tasks.keys():
    aList.append(element)

bList = aList[:]
bList.reverse()

# ================================================================================
# Обратный ход
# ================================================================================
for taskBW in bList:
    if bList.index(taskBW) == 0:  # Если это последняя задача
        tasks[taskBW]['LF'] = tasks[taskBW]['EF']
        tasks[taskBW]['LS'] = tasks[taskBW]['ES']

    for dep in tasks[taskBW]['dependencies']:
        if dep != '-1':  # Если не последняя задача
            if tasks['task' + dep]['LF'] == 0:
                tasks['task' + dep]['LF'] = int(tasks[taskBW]['LS'])
                tasks['task' + dep]['LS'] = int(tasks['task' + dep]['LF']) -
int(tasks['task' + dep]['duration'])
                tasks['task' + dep]['float'] = int(tasks['task' + dep]['LF']) -
int(tasks['task' + dep]['EF'])
            if int(tasks['task' + dep]['LF']) > int(tasks[taskBW]['LS']):
                tasks['task' + dep]['LF'] = int(tasks[taskBW]['LS'])
                tasks['task' + dep]['LS'] = int(tasks['task' + dep]['LF']) -
int(tasks['task' + dep]['duration'])
                tasks['task' + dep]['float'] = int(tasks['task' + dep]['LF']) -
int(tasks['task' + dep]['EF'])
# ================================================================================
# Печать
# ================================================================================
id = []
data = {"Длительность задачи": [], "Раннее начало": [], "Ранний конец": []
    ,"Позднее начало":[],"Поздний конец":[], "Запас":[], "Крит. путь":[]}
for task in tasks:
    id.append(tasks[task]['id'])
    data["Длительность задачи"].append(tasks[task]['duration'])
    data["Раннее начало"].append(tasks[task]['ES'])
    data["Ранний конец"].append(tasks[task]['EF'])
    data["Позднее начало"].append(tasks[task]['LS'])
    data["Поздний конец"].append(tasks[task]['LF'])
    data["Запас"].append(tasks[task]['float'])
    data["Крит. путь"].append(tasks[task]['float'] == 0)

with pd.option_context('display.max_rows', None, 'display.max_columns', None,
'display.width', None):
    print(pd.DataFrame(data, index=id))
```