

Министерство образования Российской Федерации

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ
им. Н.Э. БАУМАНА

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

Методы оптимизации

Домашнее задание №3 на тему:
«Построение сетевого графа работ и его анализ методом
критического пути»

Вариант 5

Преподаватель:
Коннова Н.С.

Студент:
Девяткин Е.Д.

Группа:
ИУ8-34

Репозиторий работы: <https://github.com/ledibonibell/МО-hw03>

Москва 2023

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.

Ход работы

Рассмотрим начальную таблицу данных (таблица 1.)

Работа (путь)	Длительность работы	Предшествующие работы
a	3	-
b	5	-
c	2	b
d	4	b
e	3	-
f	1	a
g	4	e, d
h	3	f, c, g
i	3	f, c, g
j	2	h
k	5	i

Таблица 1.

Теперь построим сетевой граф, основываясь на начальные условие (рис 2-3)

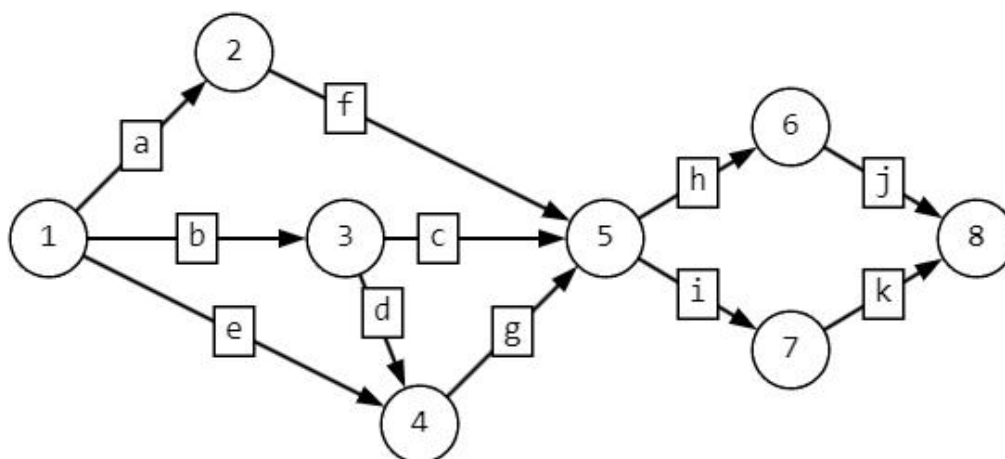


Рис. 1.

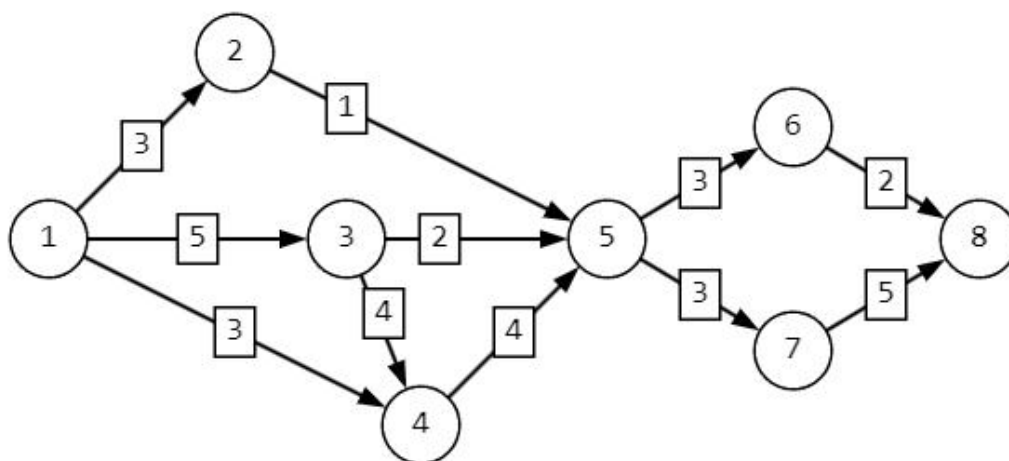


Рис. 2.

Для удобства работы, также приведем его частично упорядоченный вид на рисунке 3.

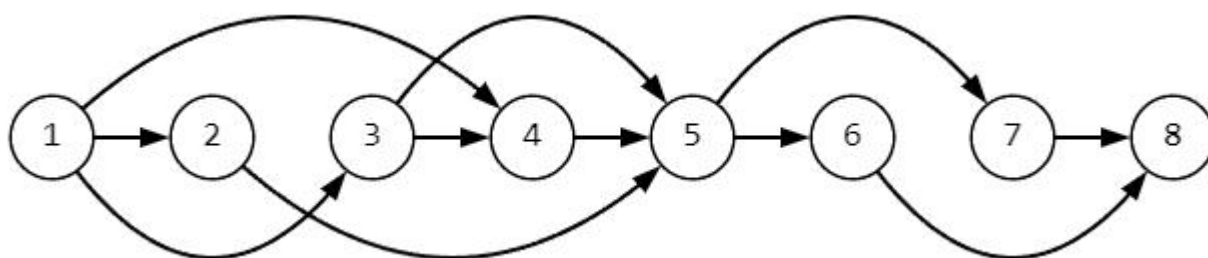


Рис. 3.

Соответственно рассчитав ранний срок окончания и поздний срок начала, получим следующее значение (таблица 2). Два последних столбца рассчитаем из таблицы ранних и поздних сроков (таблица 3).

Рассчитаем ранний срок начала: $t_{i-j}^{po} = T_i^p$

Рассчитаем поздний срок окончания: $t_{i-j}^{nh} = T_j^n$

Рассчитаем поздний срок начала: $t_{i-j}^{nh} = T_i^n + t_{i-j}$

Рассчитаем ранний срок окончания: $t_{i-j}^{nh} = T_j^p - t_{i-j}$

Аналогично рассчитываем ранние и поздние сроки и резерв:

$$\begin{cases} R_i = T_i^n - T_i^p \\ r_{i-j}^n = T_j^n - T_i^p - t_{i-j} \\ r_{i-j}^c = T_j^p - T_i^p - t_{i-j} \end{cases}$$

Работа	Вершины	Длина	t_{i-j}^{PH}	$t_{i-j}^{ПН}$	t_{i-j}^{PO}	$t_{i-j}^{ΠO}$	$r_{i-j}^{Π}$	r_{i-j}^C
a	1-2	3	0	9	3	12	9	0
b	1-3	5	0	0	5	5	0	0
c	3-5	3	5	10	8	13	5	5
d	3-4	4	5	5	9	9	0	0
e	1-4	3	0	6	3	9	6	6
f	2-5	1	3	12	4	13	9	9
g	4-5	4	9	9	13	13	0	0
h	5-6	3	13	16	16	19	3	0
i	5-7	3	13	13	16	16	0	0
j	6-8	2	16	19	18	21	3	3
k	7-8	5	16	16	21	21	0	0

Таблица 2.

Вершина	T_i^P	$T_i^{Π}$	Резерв
1	0	0	0
2	3	3	0
3	5	5	0
4	3	9	6
5	4	13	9
6	6	16	10
7	6	16	10
8	8	21	13

Таблица 3.

Рассмотрим переработанную схему с временной шкалой, с учетом найденного нами времени (рис. 4)

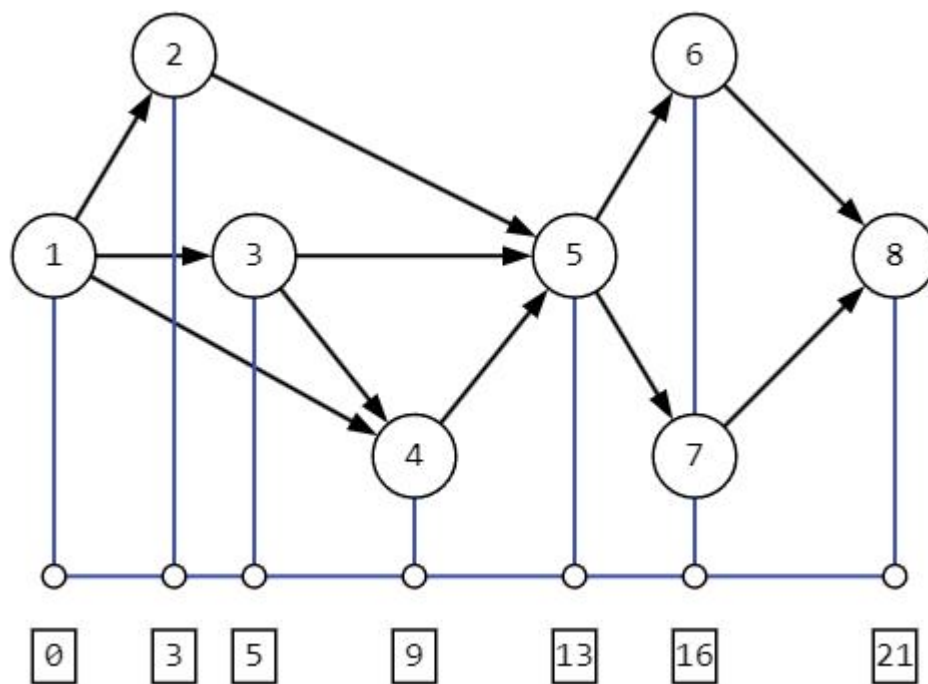


Рис. 4.

Для поиска критического пути, возьмем нулевые значения полного резерва из таблицы 2:

Критический путь: $b \rightarrow d \rightarrow g \rightarrow i \rightarrow k$ (рис.5)

Также, для проверки, перебором попытаемся найти критический путь, то есть путь, занимающий наибольшее время:

Он также равен: $b \rightarrow d \rightarrow g \rightarrow i \rightarrow k$

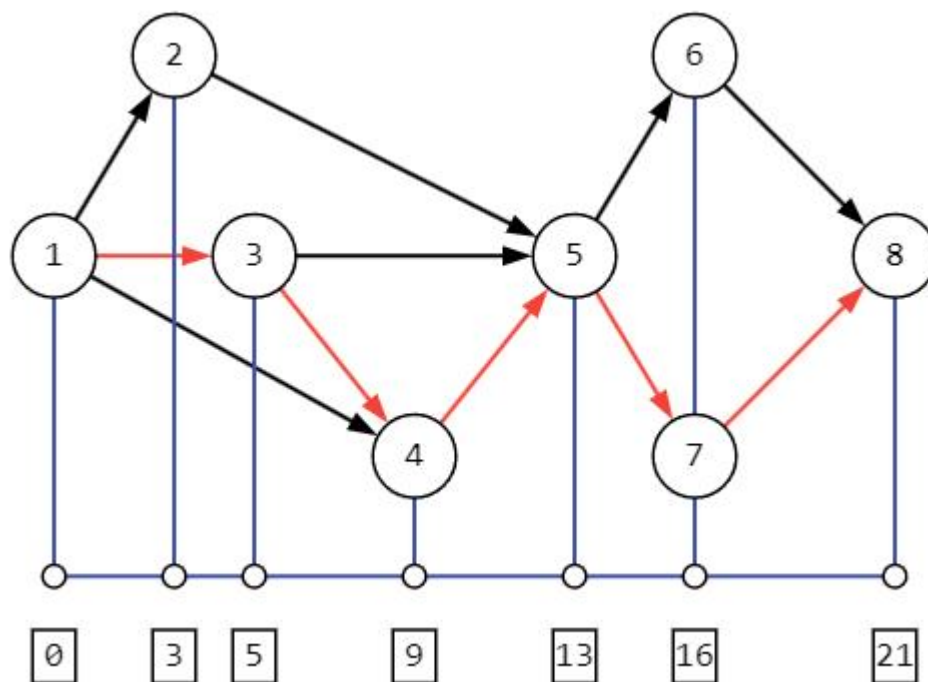


Рис. 5.

Вывод

В ходе выполнения работы был изучен сетевой граф и построена схема, показывающая ранние и поздние сроки окончания работ

Также по средствам анализа полного резерва был найден критический путь и проверен методом перебора. Тем самым мы доказали, что критический путь есть путь с нулевым запасом полного резерва, то есть максимальный по времени путь

Приложение А

Файл 'Main.py':

```
def critical_path(amount, start_params, end_params, duration):
    earliest_start_times = [0] * amount
    latest_start_times = [float('inf')] * amount

    for k in range(amount):
        max_start = earliest_start_times[start_params[k]] + duration[k]
        if earliest_start_times[end_params[k]] < max_start:
            earliest_start_times[end_params[k]] = max_start

    latest_start_times[end_params[amount - 1]] =
    earliest_start_times[end_params[amount - 1]]

    for k in range(amount - 1, -1, -1):
        min_finish = latest_start_times[end_params[k]] - duration[k]
        if latest_start_times[start_params[k]] > min_finish:
            latest_start_times[start_params[k]] = min_finish

    earliest_start = [earliest_start_times[start_params[k]] for k in range(amount)]
    earliest_finish = [earliest_start[k] + duration[k] for k in range(amount)]
    latest_finish = [latest_start_times[end_params[k]] for k in range(amount)]
    latest_start = [latest_finish[k] - duration[k] for k in range(amount)]
    total_float = [latest_finish[k] - earliest_finish[k] for k in range(amount)]
    free_float = [earliest_start_times[end_params[k]] - earliest_finish[k] for k in
    range(amount)]
    critical_path_tasks = [1] + [end_params[k] for k in range(amount) if total_float[k]
    == 0]

    return earliest_start, latest_start, earliest_finish, latest_finish, total_float, free_float,
    critical_path_tasks

# def input_tasks_data(amount):
#     works = []
#     start_work = []
#     end_work = []
#     durations = []
#
#     for i in range(amount):
#         work_name = input(f'Enter name for task {i + 1}: ')
#         start_time = int(input(f'Enter start time for task {work_name}: '))
#         end_time = int(input(f'Enter end time for task {work_name}: '))
#         duration_time = int(input(f'Enter duration for task {work_name}: '))
#
#         works.append(work_name)
```



```

#     start_work.append(start_time)
#     end_work.append(end_time)
#     durations.append(duration_time)
#
#     return works, start_work, end_work, durations
#
# n = int(input("Enter the number of tasks: "))
# result = critical_path(*input_tasks_data(n))

n = 11

works = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k"]

start_work = [1, 1, 3, 3, 1, 2, 4, 5, 5, 6, 7]
end_work = [2, 3, 5, 4, 4, 5, 5, 6, 7, 8, 8]
durations = [3, 5, 3, 4, 3, 1, 4, 3, 3, 2, 5]

# start_work = [1, 2, 7, 1, 3, 4, 5, 8, 1, 6, 9]
# end_work = [2, 7, 8, 3, 4, 5, 8, 9, 6, 9, 10]
# durations = [1, 14, 1, 2, 3, 1, 8, 2, 10, 1, 2]

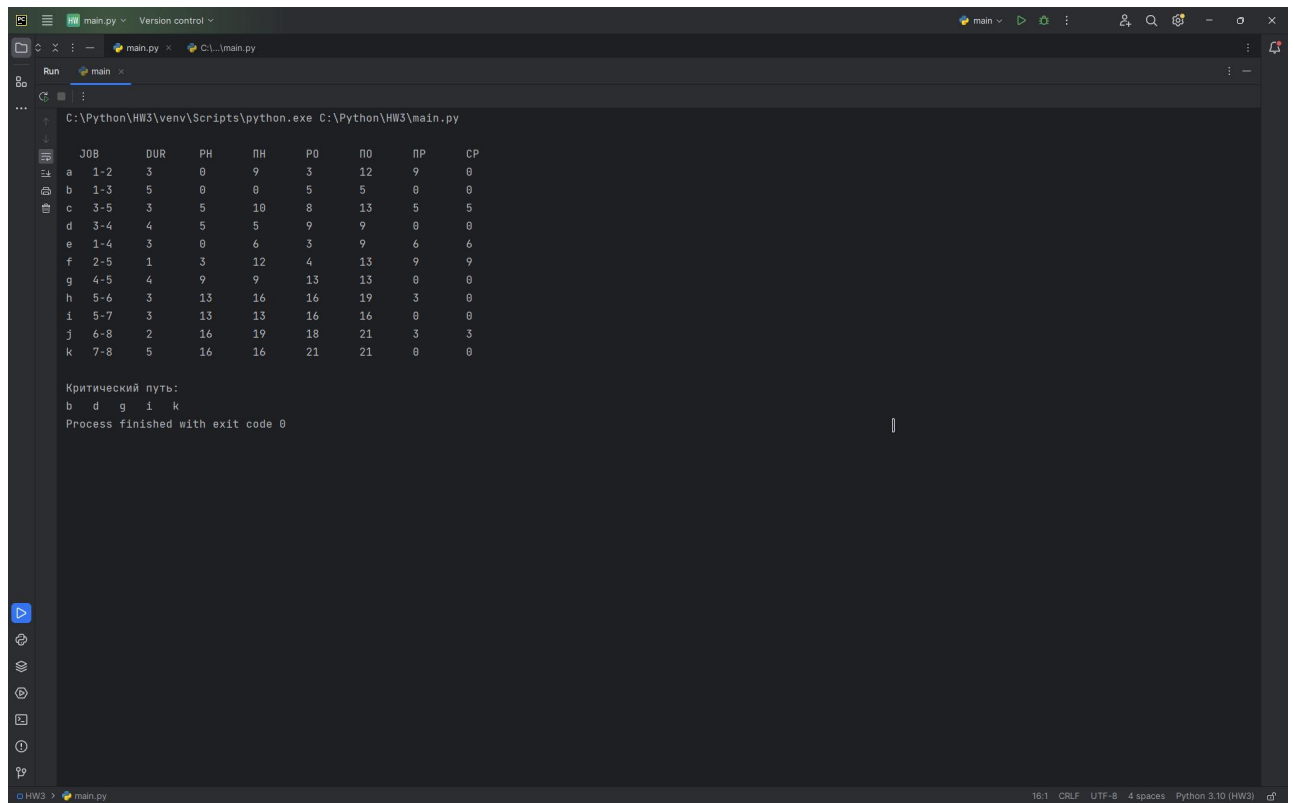
result = critical_path(n, start_work, end_work, durations)

print("\n JOB\t\tDUR\t\tPH\t\tPIH\t\tPO\t\tPIO\t\tPIP\t\tCP")
for i in range(n):
    print(f'{works[i]}\t{start_work[i]}\t{end_work[i]}\t{durations[i]}\t{result[0][i]}\t{result[1][i]}\t{result[2][i]}\t{result[3][i]}\t{result[4][i]}\t{result[5][i]}')

print("\nКритический путь:")
for j in range(len(result[6])-1): # Iterate up to the second-to-last element
    for i in range(n):
        if result[6][j] == start_work[i] and result[6][j + 1] == end_work[i]:
            print(f'{works[i]}\t', end="")

```

Приложение Б



The screenshot shows a Python IDE window titled 'main.py' with a 'Run' button and a 'main' tab. The main area displays the output of a program, which is a table of jobs and their dependencies, followed by the critical path and the exit code.

	JOB	DUR	РН	ПН	Р0	П0	ПР	СР
a	1-2	3	0	9	3	12	9	0
b	1-3	5	0	0	5	5	0	0
c	3-5	3	5	10	8	13	5	5
d	3-4	4	5	5	9	9	0	0
e	1-4	3	0	6	3	9	6	6
f	2-5	1	3	12	4	13	9	9
g	4-5	4	9	9	13	13	0	0
h	5-6	3	13	16	16	19	3	0
i	5-7	3	13	13	16	16	0	0
j	6-8	2	16	19	18	21	3	3
k	7-8	5	16	16	21	21	0	0

Критический путь:
b d g i k
Process finished with exit code 0