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 6

Выполнил: Калинин Д. В.,
студент группы ИУ8-31

Проверил: Коннова Н.С.,
доцент каф. ИУ8

Москва 2020

1. Цель работы

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

2. Условие 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.

Таблица 1 – Исходные данные

	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 – Задание в соответствии с вариантом

	P_a	P_b	P_c	P_d	P_e	P_f	P_g	P_h	P_i	P_j	P_k
t	$\emptyset$	a	a	a	b	c	e, f	c	d	h, i	j, g

3. Ход работы

Исходный граф приведён на рисунке 1.

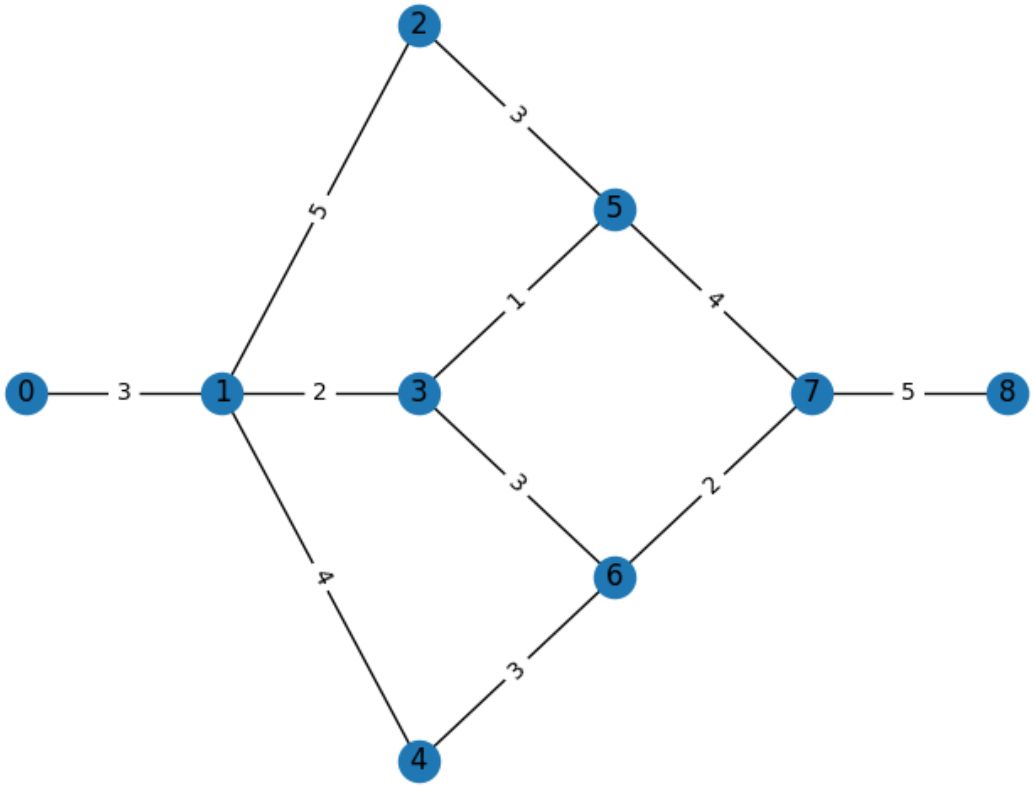

Рисунок 1 – Исходный граф

Найденный с помощью программы критический путь представлен на рисунке 2.

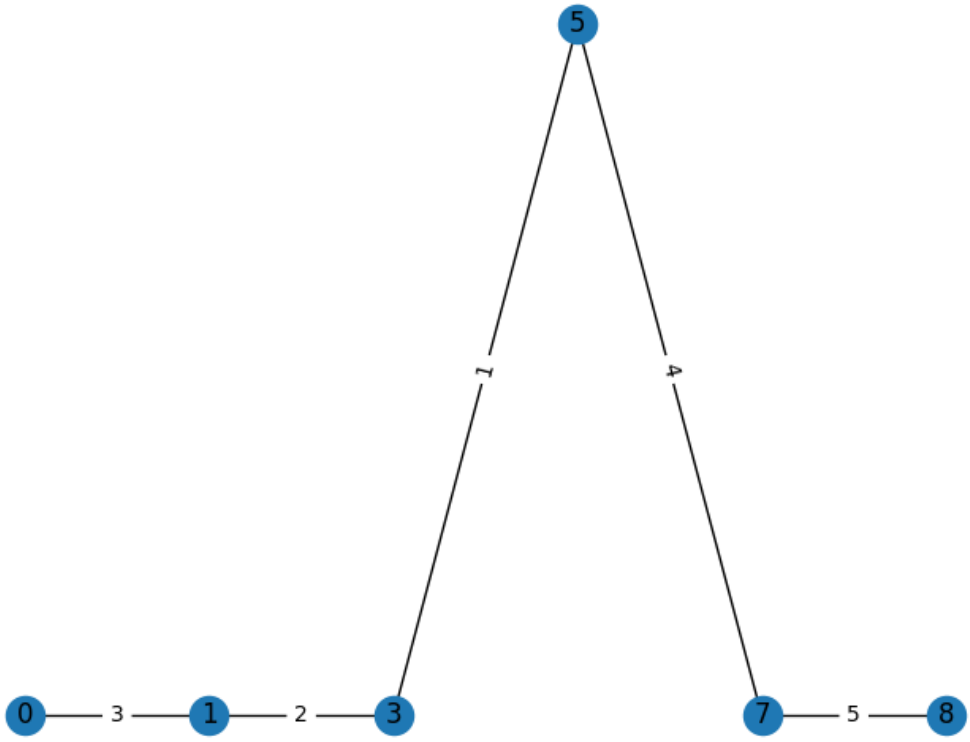

Рисунок 2 – Критический путь

Результат работы программы приведён ниже (рисунок 3).

```
nodes:  [0, 1, 2, 3, 4, 5, 6, 7, 8]
edges:  [(0, 1), (1, 2), (1, 3), (1, 4), (2, 5), (3, 5), (3, 6), (4, 6), (5, 7), (6, 7), (7, 8)]
Number of works in critical path:  5
Total work duration:  15
```

Рисунок 3 – Результат работы программы

Код программы доступен по ссылке: https://github.com/shreddered/lab-06

4. Выводы

С помощью библиотеки *networkx* был реализован поиск критического пути в сетевом графе. Все аналитические вычисления совпали с результатом работы программы.

Приложение 1. Исходный код программы

Файл main.py:

```
#!/usr/bin/env python3

# graphs
import networkx as nx
# plots
import matplotlib.pyplot as plt

def plot(G: nx.Graph):
    G.nodes[0]['pos'] = (0, 1)
    G.nodes[1]['pos'] = (1, 1)
    G.nodes[2]['pos'] = (2, 2)
    G.nodes[3]['pos'] = (2, 1)
    G.nodes[4]['pos'] = (2, 0)
    G.nodes[5]['pos'] = (3, 1.5)
    G.nodes[6]['pos'] = (3, 0.5)
    G.nodes[7]['pos'] = (4, 1)
    G.nodes[8]['pos'] = (5, 1)
    plt.figure()
    positions = nx.get_node_attributes(G, 'pos')
    nx.draw(G, positions, with_labels = True)
    weights = nx.get_edge_attributes(G, 'weight')
    nx.draw_networkx_edge_labels(G, positions, edge_labels = weights)
    plt.show()

# duration table
a = 3
b = 5
c = 2
d = 4
e = 3
f = 1
g = 4
h = 3
i = 3
j = 2
k = 5

G = nx.Graph()

G.add_nodes_from(range(0, 9))

G.add_edge(0, 1, weight = a)
G.add_edge(1, 2, weight = b)
G.add_edge(1, 3, weight = c)
G.add_edge(1, 4, weight = d)
G.add_edge(2, 5, weight = e)
G.add_edge(3, 5, weight = f)
G.add_edge(5, 7, weight = g)
G.add_edge(3, 6, weight = h)
G.add_edge(4, 6, weight = i)
G.add_edge(6, 7, weight = j)
G.add_edge(7, 8, weight = k)

print("nodes: ", G.nodes())
print("edges: ", G.edges())

plot(G)
```

```
# find critical path using dijkstra algorithm
path = G.subgraph(nx.dijkstra_path(G, 0, 8))
plt.figure()
positions = nx.get_node_attributes(path, 'pos')
nx.draw(path, positions, with_labels = True)
weights = nx.get_edge_attributes(path, 'weight')
nx.draw_networkx_edge_labels(path, positions, edge_labels = weights)
plt.show()


edges = path.edges.data('weight')
total_weight = 0
count = 0
for (_, _, weight) in edges:
    count += 1
    total_weight += weight
print("Number of works in critical path: ", count)
print("Total work duration: ", total_weight)
```

Контрольный вопрос

Какие исходные данные необходимы для использования метода критического пути?

Для метода критического пути необходимо определить сетевой граф, т.е. нужны следующие данные:

- Длительность всех работ (или же вес каждого ребра графа);
- Множество всех предшествующих работ (или же каждую вершину графа).