А **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 1

Выполнил:
Александров А А.,
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.

3. Ход работы

Таблица варианта 1

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

№пп	P_a	P_b	P_c	P_d	P_e	P_f	P_g	P_h	P_i	P_j	P_k
1	∅	∅	∅	a	b	b	d	e	f,c	g	h,i

Реализация алгоритма нахождения критического пути сетевого графа производилось с помощью библиотеки NetworkX.

Результат работы программы представлен на следующих фотографиях.

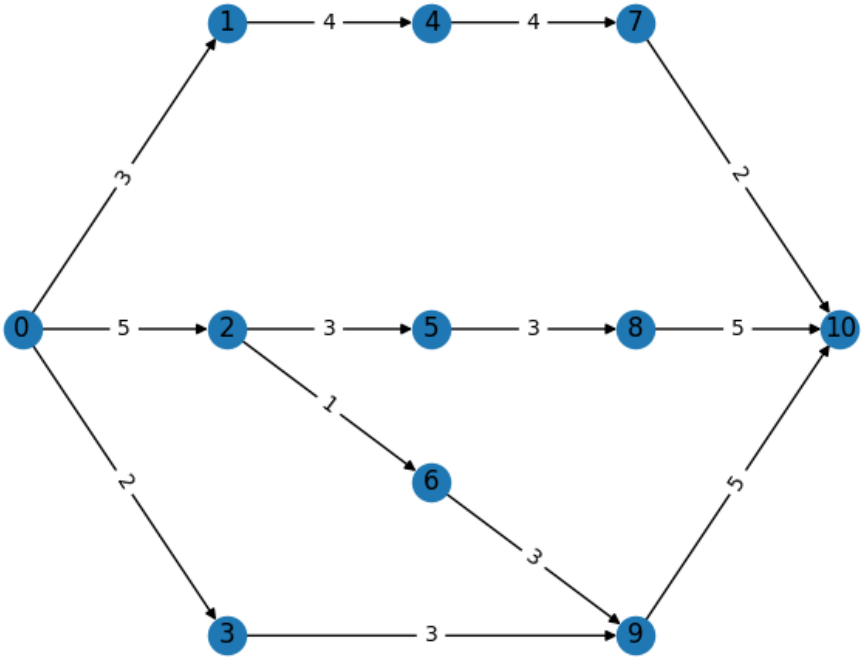

Рисунок исходного графа

```
Saving source graph...
Saving critical path...

Critical path:
(0) ---2---> (3) ---3---> (9) ---5---> (10)

Amount of works in critical path: 3
Total duration of works in critical path: 10
```

Результат работы программы

Ссылка на git-репозиторий: https://github.com/Vumba798/tsisa_lab06

4. Выводы

С помощью библиотеки NetworkX была реализована модель сетевого графа. Критический путь был найден с помощью алгоритма Дейкстры

С помощью библиотеки NetworkX реализовал решение поставленной задачи. Все аналитические вычисления совпали с результатом работы программы.

Приложение 1. Исходный код программы

Файл graph_analyser.py

```
import networkx as nx # library for graphs
import matplotlib.pyplot as plt # library for plots

# Time duration of works:
a = 3
b = 5
c = 2
d = 4
e = 3
f = 1
g = 4
h = 3
i = 3
j = 2
k = 5

# Initializes directed graph
graph = nx.DiGraph()

# Adds nodes with grid layout
graph.add_node(0, pos = (0, 1))
graph.add_node(1, pos = (1, 2))
graph.add_node(2, pos = (1, 1))
graph.add_node(3, pos = (1, 0))
graph.add_node(4, pos = (2, 2))
graph.add_node(5, pos = (2, 1))
graph.add_node(6, pos = (2, 0.5))
graph.add_node(7, pos = (3, 2))
graph.add_node(8, pos = (3, 1))
graph.add_node(9, pos = (3, 0))
graph.add_node(10, pos = (4, 1))

# Adds directed edges with weight
graph.add_edge(0, 1, weight = a)
graph.add_edge(0, 2, weight = b)
graph.add_edge(0, 3, weight = c)
graph.add_edge(1, 4, weight = d)
```

```
graph.add_edge(2, 5, weight = e)
graph.add_edge(2, 6, weight = f)
graph.add_edge(4, 7, weight = g)
graph.add_edge(5, 8, weight = h)
graph.add_edge(6, 9, weight = i)
graph.add_edge(3, 9, weight = i)
graph.add_edge(7, 10, weight = j)
graph.add_edge(8, 10, weight = k)
graph.add_edge(9, 10, weight = k)


# Draws source graph of works
plt.figure()
pos = nx.get_node_attributes(graph, 'pos')
nx.draw(graph, pos, with_labels = True)
labels = nx.get_edge_attributes(graph, 'weight')
nx.draw_networkx_edge_labels(graph, pos, edge_labels=labels)
# Uncomment string below if you want to look at the picture
# plt.show()
print("Saving source graph...")
plt.savefig('screenshots/graph.png')

# Finds and draws graph of critical path
plt.figure()
subgraph = graph.subgraph(nx.dijkstra_path(graph, 0, 10))
pos = nx.get_node_attributes(subgraph, 'pos')
nx.draw(subgraph, pos, with_labels = True)
labels = nx.get_edge_attributes(subgraph, 'weight')
nx.draw_networkx_edge_labels(subgraph, pos, edge_labels=labels)
# Uncomment string below if you want to look at the picture
# plt.show()
print("Saving critical path...")
plt.savefig('screenshots/critical_path.png')

# Prints critical path
print("\nCritical path:")
edges = subgraph.edges.data('weight')
edges = sorted(edges, key = lambda x: (x[0], x[1], x[2]))
counter = 0
total_weight = 0
```

```
for (source, destination, weight) in edges:
    if counter % 2 == 0:
        print('(', source, ') ---',weight, '---> (', destination, ')', sep='', end='')
    else:
        print(' ---', weight, '---> ', sep='', end='')
    total_weight += weight
    counter += 1
print("\n\nAmount of works in critical path:", counter)
print("Total duration of works in critical path:", total_weight)
```

Контрольный вопрос

Какие исходные данные необходимы для использования метода критического пути?

Метод критического пути требует знание длительности всех работ и множество предшествующих работ для каждой отдельной работы. Такую модель позволяет построить сетевой граф работ с заданными на нём весами (длительностью работ) рёбер.