Министерство образования и науки Российской федерации

Алтайский государственный технический университет

им. И.И. Ползунова

Отчет

о лабораторной работе №3

по теме: « Модели сетевого планирования и управления (Расчет сетевых графиков)»

предмет: Математические модели и методы поддержки принятия решений

Выполнил Студент группы 8ПИЭ-41

Хартов А.Е.

Проверил Блем А.Г.

Барнаул 2024

Цель работы – развитие навыков самостоятельной разработки программных комплексов (на примере разработке программы расчета параметров сетевого графика)

СПУ основано на моделировании процесса с помощью построения сетевого графика, отображающего планируемый комплекс работ.

Система СПУ позволяет:

- формировать календарный план реализации некоторого комплекса работ;

- выявлять и мобилизовать резервы времени, трудовые, материальные ресурсы и денежные ресурсы;

- осуществлять управление комплексом работ по принципу «ведущего звена» с прогнозированием и предупреждением возможных срывов в ходе работ.

Сетевая модель представляет собой план выполнения некоторого комплекса взаимосвязанных работ (операций), заданную в специфической форме сети, графическое изображение которой называется **сетевым графиком**. Сетевой график – это ориентированный граф без контуров, отражающий логическую взаимосвязь всех операций (работ).

Исходные данные для построения сетевого графика:

№ работы	№ нач. события	№ кон. События	Продолжительность
1	0	1	6
2	0	2	3
3	0	3	7
4	1	4	10
5	2	5	8
6	2	6	5
7	3	6	6
8	4	5	9
9	4	7	4
10	5	8	3
11	5	9	5
12	6	8	7
13	7	9	2
14	8	9	6
15	9	10	2

Код программы:

```
import networkx as nx

# Данные из таблицы
jobs = [
    (0, 1, 6),  # (Начало, Конец, Длительность)
    (0, 2, 3),
    (0, 3, 7),
    (1, 4, 10),
    (2, 5, 8),
    (2, 6, 5),
    (3, 6, 6),
    (4, 5, 9),
    (4, 7, 4),
    (5, 8, 3),
    (5, 9, 5),
    (6, 8, 7),
    (7, 9, 2),
    (8, 9, 6),
    (9, 10, 2),
]

# Создание направленного графа
G = nx.DiGraph()
for start, end, duration in jobs:
    G.add_edge(start, end, weight=duration)
```

```python
# Вычисление ранних сроков
early_times = {node: 0 for node in G.nodes()}
for node in nx.topological_sort(G):
    for pred in G.predecessors(node):
        early_times[node] = max(early_times[node],
                                early_times[pred] + G[pred][node]['weight'])

# Вычисление поздних сроков
late_times = {node: max(early_times.values()) for node in G.nodes()}
for node in reversed(list(nx.topological_sort(G))):
    for succ in G.successors(node):
        late_times[node] = min(late_times[node],
                               late_times[succ] - G[node][succ]['weight'])

# Резервы времени и критический путь
reserves = {}
critical_path = []
for start, end in G.edges():
    duration = G[start][end]['weight']
    reserve = late_times[end] - early_times[start] - duration
    reserves[(start, end)] = reserve
    if reserve == 0:
        critical_path.append((start, end))

# Вывод результатов
print("Ранние сроки: ", early_times)
print("Поздние сроки: ", late_times)
print("Резервы времени: ", reserves)
print("Критический путь: ", critical_path)
```

Результаты работы программы:

```
a1234 лабы ММиППР % /usr/local/bin/python3 "/Users/a1234/Desktop/лабы ММиППР/l3/l3.py"
Ранние сроки:  {0: 0, 1: 6, 2: 3, 3: 7, 4: 16, 5: 25, 6: 13, 7: 20, 8: 28, 9: 34, 10: 36}
Поздние сроки:  {0: 0, 1: 6, 2: 16, 3: 15, 4: 16, 5: 25, 6: 21, 7: 32, 8: 28, 9: 34, 10: 36}
Резервы времени:  {(0, 1): 0, (0, 2): 13, (0, 3): 8, (1, 4): 0, (2, 5): 14, (2, 6): 13, (3, 6):
8, (4, 5): 0, (4, 7): 12, (5, 8): 0, (5, 9): 4, (6, 8): 8, (7, 9): 12, (8, 9): 0, (9, 10): 0}
Критический путь:  [(0, 1), (1, 4), (4, 5), (5, 8), (8, 9), (9, 10)]
```

Ранние сроки наступления событий:

- Событие 0: 0
- Событие 1: 6
- Событие 2: 3
- Событие 3: 7
- Событие 4: 16
- Событие 5: 25

- Событие 6: 13
- Событие 7: 20
- Событие 8: 28
- Событие 9: 34
- Событие 10: 36

Поздние сроки наступления событий:

- Событие 0: 0
- Событие 1: 6
- Событие 2: 16
- Событие 3: 15
- Событие 4: 16
- Событие 5: 25
- Событие 6: 21
- Событие 7: 32
- Событие 8: 28
- Событие 9: 34
- Событие 10: 36

Резервы времени для работ:

- Работа (0 → 1): 0
- Работа (0 → 2): 13
- Работа (0 → 3): 8
- Работа (1 → 4): 0
- Работа (2 → 5): 14
- Работа (2 → 6): 13
- Работа (3 → 6): 8
- Работа (4 → 5): 0
- Работа (4 → 7): 12
- Работа (5 → 8): 0
- Работа (5 → 9): 4
- Работа (6 → 8): 8
- Работа (7 → 9): 12
- Работа (8 → 9): 0
- Работа (9 → 10): 0

Критический путь:

(0 → 1) → (1 → 4) → (4 → 5) → (5 → 8) → (8 → 9) → (9 → 10)

Критический путь определяет минимальное время выполнения проекта: **36**

Итог:

Проект может быть выполнен за 36 единиц времени при условии, что работы на критическом пути будут завершены строго в срок. Работы с резервами времени обеспечивают дополнительную гибкость и возможность для оперативного управления изменениями в графике.