Лабораторная работа №6, Вариант 76

1. Построение матрицы смежности

```
import numpy as np

# Вершины
vertices = 20
# Рёбра графа (неориентированный граф)
edges = [
   (0, 2), (0, 10), (0, 3), (3, 19), (1, 6), (1, 7), (7, 13), (2, 11),
   (2, 3), (2, 6), (2, 19), (3, 10), (4, 1), (4, 5), (5, 16), (5, 6),
   (6, 12), (6, 14), (6, 2), (7, 0), (7, 10)
]

# Создаём пустую матрицу смежности
adj_matrix = np.zeros((vertices, vertices), dtype=int)

# Заполняем матрицу смежности
for u, v in edges:
   adj_matrix[u][v] = 1
   adj_matrix[v][u] = 1  # так как граф неориентированный

print("Матрица смежности:")
print(adj_matrix)
```

2. Преобразование в матрицу инцидентности

```
# Количество рёбер
num_edges = len(edges)

# Матрица инцидентности размером (вершины x рёбра)
inc_matrix = np.zeros((vertices, num_edges), dtype=int)

for idx, (u, v) in enumerate(edges):
   inc_matrix[u][idx] = 1  # начальная вершина
   inc_matrix[v][idx] = -1  # конечная вершина

print("Матрица инцидентности:")
```

```
print(inc_matrix)
```

3. Проверка на эйлеров граф

```
# Проверка, является ли граф Эйлеровым (если степени всех вершин чётные)
degrees = np.sum(adj_matrix, axis=1)
is_eulerian = np.all(degrees % 2 == 0)

if is_eulerian:
    print("Граф является Эйлеровым.")
else:
    print("Граф не является Эйлеровым.")
```

4. Поиск кратчайших путей (алгоритм Флойда-Уоршелла)

```
INF = float('inf')

# Инициализация матрицы расстояний
dist = np.full((vertices, vertices), INF)
np.fill_diagonal(dist, 0)

for u, v in edges:
    dist[u][v] = 1
    dist[v][u] = 1  # для неориентированного графа

# Алгоритм Флойда-Уоршелла
for k in range(vertices):
    for i in range(vertices):
        for j in range(vertices):
            dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j])

print("Матрица кратчайших путей:")
print(dist)
```

Лабораторная работа №8, Вариант 68

1. Поиск максимального потока (Алгоритм Эдмондса-Карпа)

```
from collections import deque

# Граф с пропускными способностями
capacity = {
   'S': {'A': 14, 'C': 16},
   'A': {'B': 31, 'C': 37},
   'B': {'D': 27, 'E': 29},
   'C': {'E': 22, 'G': 26},
   'D': {'T': 20},
   'E': {'T': 25},
   'G': {'T': 14},
   'T': {}
}

# Алгоритм Эдмондса-Карпа (на основе BFS)
def bfs_flow(source, sink, parent):
   visited = set()
   queue = deque([source])
   visited.add(source)

   while queue:
      u = queue.popleft()

      for v, cap in capacity[u].items():
         if v not in visited and cap > 0:  # если вершина не посещена и есть пропускная
способность
            queue.append(v)
            visited.add(v)
            parent[v] = u
            if v == sink:
               return True
   return False

def edmonds_karp(source, sink):
   max_flow = 0
   parent = {}

   while bfs_flow(source, sink, parent):
      path_flow = float('Inf')
      s = sink
```

```
        while s != source:
            path_flow = min(path_flow, capacity[parent[s]][s])
            s = parent[s]

        max_flow += path_flow
        v = sink

        while v != source:
            u = parent[v]
            capacity[u][v] -= path_flow
            capacity[v][u] = capacity.get(v, {}).get(u, 0) + path_flow
            v = parent[v]

    return max_flow

max_flow = edmonds_karp('S', 'T')
print(f"Максимальный поток: {max_flow}")
```

2. Генерация случайных пропускных способностей

```
import random

# Изменяем пропускные способности дуг случайным образом в диапазоне [100, 1000]
for u in capacity:
    for v in capacity[u]:
        capacity[u][v] = random.randint(100, 1000)

max_flow_random = edmonds_karp('S', 'T')
print(f"Максимальный поток с случайными пропускными способностями:
{max_flow_random}")
```