РГР №2

Бондарева Елена М8О-109СВ-25

```
# Часть 1: Основные функции и матрицы предпочтений
import numpy as np
from fractions import Fraction
import networkx as nx
import matplotlib.pyplot as plt

def get_column(matrix, col_index):
    return [row[col_index] for row in matrix]

def prefeMax(date):
    p = np.zeros((4, 4))
    for i in range(4):
        for j in range(4):
            p[i][j] = (date[j]/(date[i]+date[j]))
    return p

def prefeMin(date):
    p = np.zeros((4, 4))
    for i in range(4):
        for j in range(4):
            p[i][j] = (date[i]/(date[i]+date[j]))
    return p

def fracprint(mat):
    fraction_matrix = []
    for row in mat:
        fraction_row = [Fraction(num).limit_denominator() for num in row]
        fraction_matrix.append(fraction_row)
    for row in fraction_matrix:
        print([str(frac) for frac in row])

# Матрица из вашего условия
cin = np.array([
    [1, 4, 2, 4],
    [2, 2, 1, 3],
    [1, 2, 2, 2],
    [3, 3, 1, 4]
])

print("Исходная матрица:")
print(cin)
print()

# Матрицы предпочтений
print("="*55)
K1 = get_column(cin, 0)
```

```
print("Матрица предпочтений p1 (K1 max)")
matrix_data1 = prefeMax(K1)
fracprint(matrix_data1)
print()

print("="*55)
K2 = get_column(cin, 1)
print("Матрица предпочтений p2 (K2 min)")
matrix_data2 = prefeMin(K2)
fracprint(matrix_data2)
print()

print("="*55)
K3 = get_column(cin, 2)
print("Матрица предпочтений p3 (K3 max)")
matrix_data3 = prefeMax(K3)
fracprint(matrix_data3)
print()

print("="*55)
K4 = get_column(cin, 3)
print("Матрица предпочтений p4 (K4 min)")
matrix_data4 = prefeMin(K4)
fracprint(matrix_data4)
print()

# Сумма матриц предпочтений
print("="*55)
print("Матрица P (суммарная)")
P = matrix_data1 + matrix_data2 + matrix_data3 + matrix_data4
fracprint(P)
print()

print("="*55)
print("Матрица P12 (критерии 1,2)")
P12 = matrix_data1 + matrix_data2
fracprint(P12)
print()

print("="*55)
print("Матрица P34 (критерии 3,4)")
P34 = matrix_data3 + matrix_data4
fracprint(P34)
print()


Исходная матрица:
[[1 4 2 4]
 [2 2 1 3]
 [1 2 2 2]
 [3 3 1 4]]
```

```
==========================================================
Матрица предпочтений p1 (K1 max)
['1/2', '2/3', '1/2', '3/4']
['1/3', '1/2', '1/3', '3/5']
['1/2', '2/3', '1/2', '3/4']
['1/4', '2/5', '1/4', '1/2']

==========================================================
Матрица предпочтений p2 (K2 min)
['1/2', '2/3', '2/3', '4/7']
['1/3', '1/2', '1/2', '2/5']
['1/3', '1/2', '1/2', '2/5']
['3/7', '3/5', '3/5', '1/2']

==========================================================
Матрица предпочтений p3 (K3 max)
['1/2', '1/3', '1/2', '1/3']
['2/3', '1/2', '2/3', '1/2']
['1/2', '1/3', '1/2', '1/3']
['2/3', '1/2', '2/3', '1/2']

==========================================================
Матрица предпочтений p4 (K4 min)
['1/2', '4/7', '2/3', '1/2']
['3/7', '1/2', '3/5', '3/7']
['1/3', '2/5', '1/2', '1/3']
['1/2', '4/7', '2/3', '1/2']

==========================================================
Матрица P (суммарная)
['2', '47/21', '7/3', '181/84']
['37/21', '2', '21/10', '27/14']
['5/3', '19/10', '2', '109/60']
['155/84', '29/14', '131/60', '2']

==========================================================
Матрица P12 (критерии 1,2)
['1', '4/3', '7/6', '37/28']
['2/3', '1', '5/6', '1']
['5/6', '7/6', '1', '23/20']
['19/28', '1', '17/20', '1']

==========================================================
Матрица P34 (критерии 3,4)
['1', '19/21', '7/6', '5/6']
['23/21', '1', '19/15', '13/14']
['5/6', '11/15', '1', '2/3']
['7/6', '15/14', '4/3', '1']
```

```python
# Часть 2: Преобразование в матрицы C и R
def process_with_visualization(matrix):
    result = np.array(matrix, dtype=float)
    n = len(result)
    result[0][0], result[1][1], result[2][2], result[3][3] = 0, 0, 0, 0

    for i in range(n):
        for j in range(i + 1, n):
```

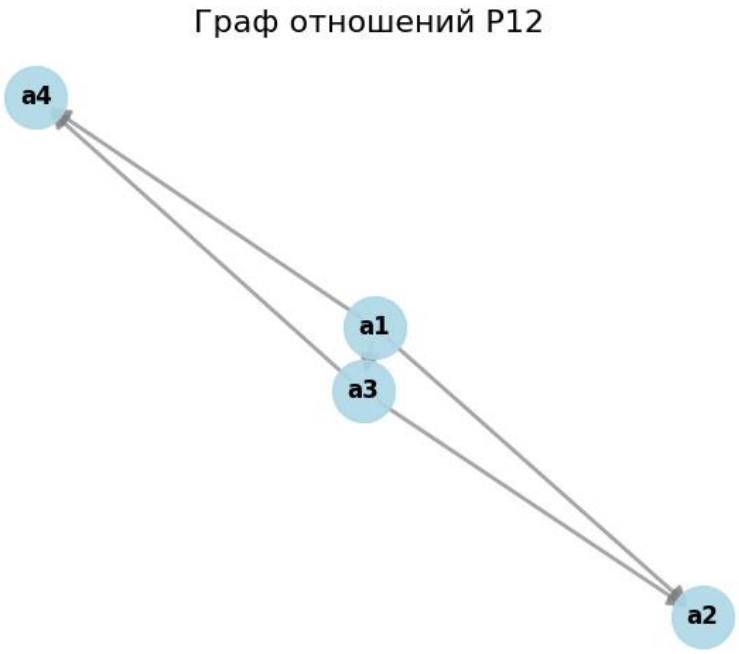
```
            if result[i,j] > result[j,i]:
                diff = result[i,j] - result[j,i]
                result[i,j] = diff
                result[j,i] = 0
            elif result[j,i] > result[i,j]:
                diff = result[j,i] - result[i,j]
                result[j,i] = diff
                result[i,j] = 0
            else:
                result[i,j] = 0
                result[j,i] = 0
    return result

print("="*55)
C = process_with_visualization(P)
print("Матрица C:")
print(np.round(C, 4))
print()

print("="*55)
C12 = process_with_visualization(P12)
print("Матрица C12:")
print(np.round(C12, 4))
print()

print("="*55)
C34 = process_with_visualization(P34)
print("Матрица C34:")
print(np.round(C34, 4))
print()

# Матрицы R
def create_binary_matrix_with_unit_diagonal(matrix):
    result = np.array(matrix, dtype=float)
    result[result > 0] = 1
    np.fill_diagonal(result, 1)
    return result

print("="*25)
R = create_binary_matrix_with_unit_diagonal(C)
print("Матрица R:")
print(R)
print()

print("="*25)
R12 = create_binary_matrix_with_unit_diagonal(C12)
print("Матрица R12:")
print(R12)
print()

print("="*25)
```

```
R34 = create_binary_matrix_with_unit_diagonal(C34)
print("Матрица R34:")
print(R34)
print()
```

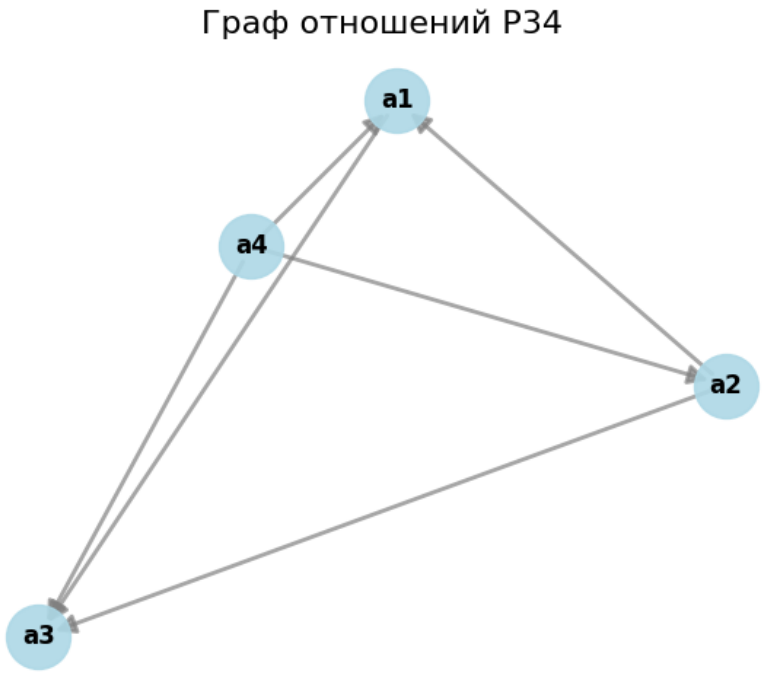
```
==========================================================
Матрица C:
[[0.     0.4762 0.6667 0.3095]
 [0.     0.     0.2    0.    ]
 [0.     0.     0.     0.    ]
 [0.     0.1429 0.3667 0.    ]]

==========================================================
Матрица C12:
[[0.     0.6667 0.3333 0.6429]
 [0.     0.     0.     0.    ]
 [0.     0.3333 0.     0.3   ]
 [0.     0.     0.     0.    ]]

==========================================================
Матрица C34:
[[0.     0.     0.3333 0.    ]
 [0.1905 0.     0.5333 0.    ]
 [0.     0.     0.     0.    ]
 [0.3333 0.1429 0.6667 0.    ]]

==========================
Матрица R:
[[1. 1. 1. 1.]
 [0. 1. 1. 0.]
 [0. 0. 1. 0.]
 [0. 1. 1. 1.]]

==========================
Матрица R12:
[[1. 1. 1. 1.]
 [0. 1. 0. 0.]
 [0. 1. 1. 1.]
 [0. 0. 0. 1.]]

==========================
Матрица R34:
[[1. 0. 1. 0.]
 [1. 1. 1. 0.]
 [0. 0. 1. 0.]
 [1. 1. 1. 1.]]
```

```
# Часть 3: Визуализация графов
def create_binary_matrix(matrix):
    result = np.array(matrix, dtype=float)
    result[result > 0] = 1
    return result

def build_graph_from_relation_matrix(matrix, node_names=None):
    if node_names is None:
        node_names = [f'a{i+1}' for i in range(len(matrix))]
```

```
    G = nx.DiGraph()
    G.add_nodes_from(node_names)

    for i in range(len(matrix)):
        for j in range(len(matrix)):
            if matrix[i][j] != 0 and i != j:
                G.add_edge(node_names[i], node_names[j])
    return G

def visualize_graph(G, title="Граф отношений"):
    plt.figure(figsize=(6, 5))
    pos = nx.spring_layout(G, seed=42)

    nx.draw_networkx_nodes(G, pos, node_size=1000, node_color='lightblue',
alpha=0.9)
    nx.draw_networkx_edges(G, pos, edge_color='gray', width=2, alpha=0.7,
arrows=True, arrowsize=20)
    nx.draw_networkx_labels(G, pos, font_size=12, font_weight='bold')

    plt.title(title, fontsize=16)
    plt.axis('off')
    plt.tight_layout()
    plt.show()

# Граф для матрицы P
print("Построение графа для матрицы P...")
relation_matrix = create_binary_matrix(C)
G = build_graph_from_relation_matrix(relation_matrix, ['a1', 'a2', 'a3',
'a4'])
visualize_graph(G, "Граф отношений P")

# Граф для матрицы P12
print("Построение графа для матрицы P12...")
relation_matrix = create_binary_matrix(C12)
G = build_graph_from_relation_matrix(relation_matrix, ['a1', 'a2', 'a3',
'a4'])
visualize_graph(G, "Граф отношений P12")

# Граф для матрицы P34
print("Построение графа для матрицы P34...")
relation_matrix = create_binary_matrix(C34)
G = build_graph_from_relation_matrix(relation_matrix, ['a1', 'a2', 'a3',
'a4'])
visualize_graph(G, "Граф отношений P34")
```



Построение графа для матрицы P34

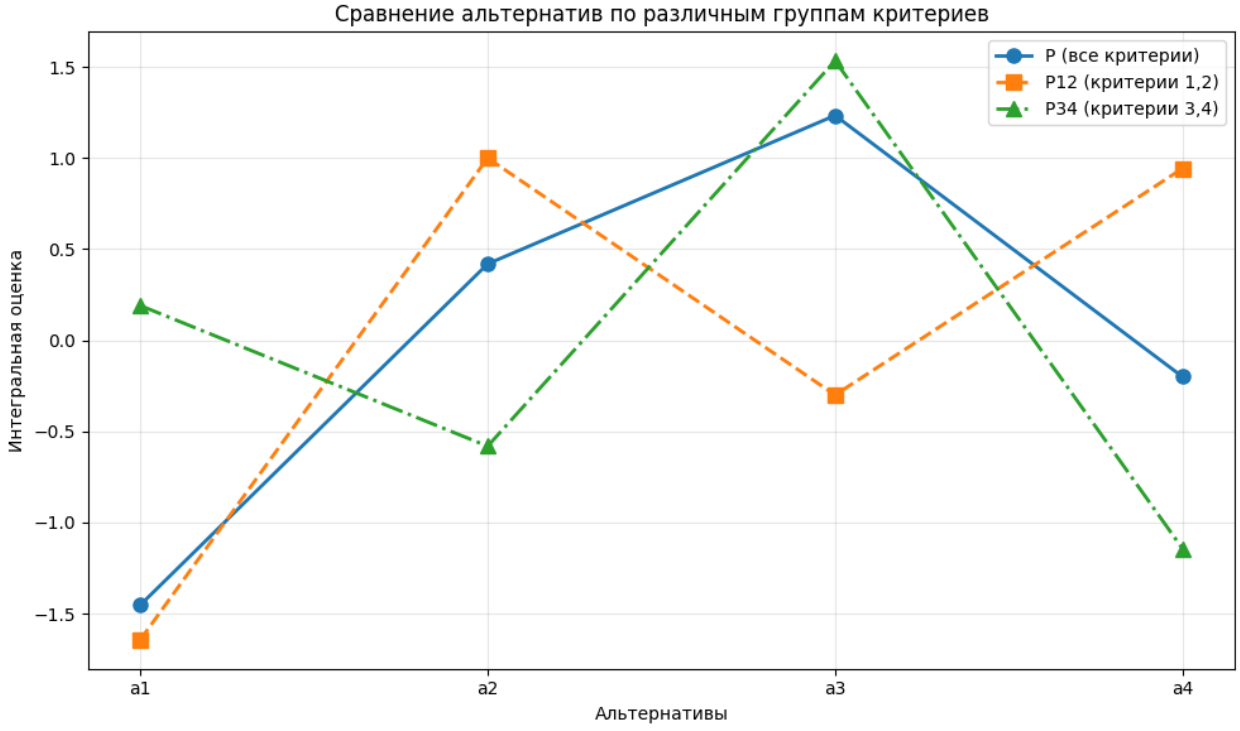



```
# Часть 4: Расчет индексов и построение графика
def FindA(matrix):
    row_sums = []
    for i in range(4):
        row_sum = sum(matrix[i])
        row_sums.append(row_sum)

    col_sums = []
    for j in range(4):
        col_sum = 0
```

```
        for i in range(4):
            col_sum += matrix[i][j]
        col_sums.append(col_sum)

    differences = []
    for i in range(4):
        difference = col_sums[i] - row_sums[i]
        differences.append(difference)
        print(f"a{i+1}: = {difference:.4f}")

    return differences

print("Индексы альтернатив для матрицы P:")
diff_P = FindA(P)
print()

print("Индексы альтернатив для матрицы P12:")
diff_P12 = FindA(P12)
print()

print("Индексы альтернатив для матрицы P34:")
diff_P34 = FindA(P34)
print()

# Ранжирование
def rank_alternatives(differences):
    indexed = list(enumerate(differences, 1))
    ranked = sorted(indexed, key=lambda x: x[1], reverse=True)
    return [f"a{i}" for i, diff in ranked]

print("Ранжирование альтернатив:")
print("По матрице P:", rank_alternatives(diff_P))
print("По матрице P12:", rank_alternatives(diff_P12))
print("По матрице P34:", rank_alternatives(diff_P34))
print()

# Построение графика
criteria = ['a1', 'a2', 'a3', 'a4']

plt.figure(figsize=(10, 6))
plt.plot(criteria, diff_P, 'o-', linewidth=2, markersize=8, label='P (все
критерии)')
plt.plot(criteria, diff_P12, 's--', linewidth=2, markersize=8, label='P12
(критерии 1,2)')
plt.plot(criteria, diff_P34, '^-.', linewidth=2, markersize=8, label='P34
(критерии 3,4)')

plt.xlabel('Альтернативы')
plt.ylabel('Интегральная оценка')
plt.title('Сравнение альтернатив по различным группам критериев')
plt.legend()
```

```
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()

# Вывод результатов
print("="*60)
print("РЕЗУЛЬТАТЫ:")
print("="*60)
print("Лучшая альтернатива по всем критериям:", rank_alternatives(diff_P)[0])
print("Лучшая альтернатива по критериям 1,2:",
rank_alternatives(diff_P12)[0])
print("Лучшая альтернатива по критериям 3,4:",
rank_alternatives(diff_P34)[0])
print()
print("Полное ранжирование:")
print("P (все критерии):", " > ".join(rank_alternatives(diff_P)))
print("P12 (критерии 1,2):", " > ".join(rank_alternatives(diff_P12)))
print("P34 (критерии 3,4):", " > ".join(rank_alternatives(diff_P34)))
```

```
Индексы альтернатив для матрицы P34:
a1: = 0.1905
a2: = -0.5810
a3: = 1.5333
a4: = -1.1429

Ранжирование альтернатив:
По матрице P: ['a3', 'a2', 'a4', 'a1']
По матрице P12: ['a2', 'a4', 'a3', 'a1']
По матрице P34: ['a3', 'a1', 'a2', 'a4']
```



```
============================================================
Вывод:
============================================================
Лучшая альтернатива по всем критериям: a3
Лучшая альтернатива по критериям 1,2: a2
Лучшая альтернатива по критериям 3,4: a3

Полное ранжирование:
P (все критерии): a3 > a2 > a4 > a1
P12 (критерии 1,2): a2 > a4 > a3 > a1
P34 (критерии 3,4): a3 > a1 > a2 > a4
```