1 Пункт 6

Поиск траектории был реализован с использованием python. Для разных компонент связности для различных вершин были рассмотрены тракетории длинны 10, 50, 100, 1000.

```
import numpy as np

def simulate_markov_chain(transition_matrix, initial_state, steps):
    states = [initial_state]
    current_state = initial_state

    for _ in range(steps):
        current_state = np.random.choice(
            a=len(transition_matrix),
            p=transition_matrix[current_state]
        )
        states.append(current_state)

    return states

transition_matrix = np.array([
    [0, 7/10, 3/10, 0, 0, 0],
    [4/10, 0, 0, 0, 0, 6/10],
    [6/10, 0, 0, 0, 0, 4/10],
    [0, 0, 0, 5/10, 5/10, 0],
    [0, 0, 0, 5/10, 5/10, 0],
    [0, 5/10, 5/10, 0, 0, 0 ]
])

trajectory = simulate_markov_chain(transition_matrix, 0, 10)
print(f'Initial State: 0, Steps: 10, Trajectory: {trajectory}')
trajectory = simulate_markov_chain(transition_matrix, 1, 50)
print(f'Initial State: 1, Steps: 50, Trajectory: {trajectory}')
trajectory = simulate_markov_chain(transition_matrix, 2, 100)
print(f'Initial State: 2, Steps: 100, Trajectory: {trajectory}')
trajectory = simulate_markov_chain(transition_matrix, 5, 1000)
print(f'Initial State: 5, Steps: 1000, Trajectory: {trajectory}')
trajectory = simulate_markov_chain(transition_matrix, 3, 100)
print(f'Initial State: 3, Steps: 100, Trajectory: {trajectory}')
trajectory = simulate_markov_chain(transition_matrix, 4, 1000)
print(f'Initial State: 4, Steps: 1000, Trajectory: {trajectory}')
```

Были получены следующие результаты:

Initial State: 0, Steps: 10, Trajectory: [0, 1, 0, 1, 0, 2, 5, 1, 0, 2, 0]

Initial State: 1, Steps: 50, Trajectory: [1, 5, 2, 5, 2, 0, 2, 5, 2, 0, 1, 0, 1, 5, 1, 0, 2, 0, 1, 5, 2, 0, 2, 0, 1, 5, 1, 0, 1, 5, 1, 5, 1, 5, 1, 0, 1, 0, 1, 5, 2, 0, 1, 5, 1, 0, 2, 0, 1, 5, 1]

Initial State: 2, Steps: 100, Trajectory: [2, 5, 1, 0, 2, 0, 2, 0, 1, 5, 2, 5, 2, 5, 2, 0, 1, 0, 1, 5, 2, 5, 1, 5, 2, 5, 1, 5, 2, 5, 1, 5, 2, 5, 2, 0, 1, 5, 1, 5, 2, 5, 2, 5, 2, 0, 2, 0, 2, 0, 1, 5, 1, 5, 2, 5, 1, 0, 2, 5, 1, 5, 2, 5, 1, 5, 2, 0, 2, 0, 1, 5, 2, 0, 1, 0, 2, 0, 1, 0, 1, 5, 1, 5, 2, 0, 1, 5, 2, 0, 1, 5, 1, 5, 2, 0, 1, 5, 2, 0, 2]

Initial State: 5, Steps: 1000, Trajectory: [5, 1, 0, 1, 0, 1, 0, 1, 5, 2, 0, 1, 5, 2, 0, 1, 5, 2, 5, 2, 5, 1, 5, 2, 0, 1, 5, 2, 0, 1, 5, 1, 5, 2, 0, 1, 0, 1, 5, 1, 0, 1, 5, 2, 0, 2, 5, 1, 5, 2, 0, 1, 0, 1, 5, 2, 0, 1, 5, 2, 0, 2, 5, 2, 0, 1, 0, 2, 0, 2, 5, 1, 0, 1, 0, 1, 5, 1, 5, 1, 0, 1, 5, 2, 5, 1, 5, 1, 0, 2, 5, 1, 0, 1, 0, 1, 0, 1, 5, 2, 0, 2, 0, 1, 5, 2, 5, 2, 5, 1, 0, 1, 0, 1, 5, 1, 0, 1, 5, 1, 0, 2, 0, 1, 0, 1, 0, 2, 0, 1, 5, 1, 5, 1, 5, 1, 5, 1, 5, 1, 0, 1, 5, 2, 0, 2, 0, 1, 5, 2, 0, 2, 5, 1, 5, 2, 5, 2, 0, 1, 0, 1, 0, 1, 5, 1, 0, 2, 0, 2, 0, 2, 0, 1, 0, 1, 0, 1, 0, 1, 5, 2, 0, 2, 0, 1, 0, 1, 5, 2, 0, 1, 5, 2, 0, 1, 0, 1, 0, 1, 5, 1, 5, 2, 0, 1, 5, 2, 0, 1, 5, 2, 0, 2, 5, 2, 5, 2, 5, 1, 0, 2, 5, 1, 0, 1, 5, 1, 5, 1, 5, 1, 0, 1, 0, 2, 5, 2, 5, 1, 0, 2, 5, 2, 5, 1, 5, 1, 0, 1, 0, 1, 0, 1, 5, 1, 5, 2, 0, 1, 5, 1, 5, 1, 0, 1, 5, 2, 0, 1, 0, 1, 5, 1, 5, 2, 5, 2, 0,

1, 5, 2, 0, 1, 0, 2, 5, 1, 5, 2, 5, 1, 5, 1, 0, 1, 5, 2, 0, 2, 5, 1, 0, 2, 5, 1, 5, 1, 0, 1, 5, 2, 0, 1, 0, 1,
0, 2, 5, 1, 5, 2, 5, 1, 0, 2, 5, 2, 5, 2, 0, 2, 0, 1, 5, 1, 5, 1, 5, 2, 0, 1, 0, 1, 5, 1, 0, 1, 0, 1, 5, 2, 5,
1, 5, 2, 5, 2, 5, 1, 5, 1, 0, 1, 5, 1, 0, 1, 5, 2, 5, 2, 5, 2, 0, 2, 5, 1, 5, 2, 5, 1, 5, 1, 0, 2, 0, 1, 5, 1,
5, 2, 0, 2, 0, 1, 5, 2, 0, 1, 0, 1, 5, 1, 0, 1, 5, 1, 0, 1, 5, 2, 5, 2, 0, 2, 5, 2, 5, 1, 0, 1, 5, 1, 5, 1, 0,
1, 5, 2, 5, 2, 5, 2, 0, 2, 0, 1, 5, 2, 0, 1, 5, 1, 5, 2, 5, 2, 0, 1, 0, 1, 5, 2, 5, 1, 5, 2, 0, 2, 0, 1, 0, 1,
5, 1, 5, 2, 5, 2, 5, 1, 0, 1, 5, 1, 5, 2, 5, 2, 0, 1, 5, 2, 0, 2, 0, 1, 5, 2, 0, 1, 0, 2, 0, 1, 5, 2, 0, 2, 5,
2, 0, 1, 5, 2, 5, 1, 0, 1, 5, 2, 0, 2, 0, 1, 5, 2, 0, 1, 5, 1, 0, 1, 0, 1, 5, 2, 5, 1, 5, 1, 0, 2, 0, 1, 0, 1,
5, 1, 5, 2, 5, 1, 5, 1, 5, 2, 5, 2, 0, 2, 0, 1, 0, 1, 5, 1, 0, 1, 5, 2, 0, 1, 5, 1, 5, 1, 5, 2, 0, 1, 0, 1, 5,
1, 5, 1, 5, 1, 5, 2, 0, 1, 5, 1, 5, 1, 5, 1, 0, 1, 0, 1, 5, 1, 5, 1, 0, 1, 0, 1, 0, 1, 5, 1, 5, 1, 5, 1, 5, 1,
0, 2, 0, 2, 5, 1, 5, 1, 0, 1, 5, 2, 0, 2, 0, 1, 5, 2, 0, 1, 0, 2, 0, 2, 0, 2, 5, 1, 0, 1, 5, 1, 5, 1, 5, 1, 0,
1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 2, 0, 1, 0, 1, 0, 1, 5, 2, 5, 1, 5, 2, 0, 2, 0, 1, 5, 2, 0, 2, 5, 2,
0, 1, 5, 2, 0, 1, 5, 1, 5, 2, 0, 1, 5, 1, 5, 1, 0, 1, 5, 1, 5, 1, 5, 1, 5, 2, 0, 1, 5, 2, 0, 1, 5, 1, 5, 2, 0,
1, 5, 2, 0, 2, 5, 1, 0, 1, 5, 1, 0, 1, 5, 1, 5, 2, 0, 1, 5, 1, 5, 2, 0, 1, 5, 1, 5, 2, 5, 1, 0, 2, 0, 1, 0, 1,
5, 2, 0, 2, 0, 1, 5, 2, 0, 1, 0, 1, 0, 2, 5, 1, 5, 1, 0, 2, 5, 2, 0, 1, 5, 2, 0, 2, 0, 1, 0, 1, 0, 1, 0, 2, 5,
1, 5, 1, 0, 1, 5, 2, 0, 1, 0, 2, 0, 1, 5, 1, 5, 2, 0, 2, 5, 1, 0, 1, 0, 2, 5, 2, 0, 1, 0, 2, 0, 1, 0, 1, 5, 2,
0, 1, 5, 1, 0, 2, 5, 2, 0, 1, 5, 2, 0, 1, 5, 1, 0, 1, 0, 2, 5, 2, 5, 1, 5, 2, 0, 2, 0, 1, 5, 1, 5, 1, 0, 1, 5,
2, 0, 1, 5, 2, 5, 1, 5, 1, 5, 1, 0, 2, 5, 1, 5, 1, 5, 1, 0, 1, 5, 2, 0, 2, 5, 1, 5, 2, 0, 1, 5, 1, 0, 1, 0, 1,
0, 1, 5, 2, 0, 1, 5, 1, 5, 2, 5, 2, 5, 1, 5, 2, 0, 1, 5, 1, 5, 1, 5, 1, 5, 1, 5, 1, 5, 2, 0, 1, 5, 1, 0, 2, 0,
1, 0, 1, 5, 1, 5, 1, 0, 1, 5, 2, 5, 1, 0, 1, 0, 2, 5, 1, 5, 2, 0, 1, 0, 2, 5, 2, 5, 1, 5, 1, 0, 1, 5, 2, 0, 2,
0, 1, 5, 2, 5, 2, 5, 2, 5, 1, 5, 2, 0, 1, 5, 2, 5, 1, 5]

Initial State: 3, Steps: 100, Trajectory: [3, 3, 3, 4, 3, 4, 4, 4, 3, 4, 3, 3, 4, 4, 3, 4, 3, 3, 3, 4,
4, 3, 3, 3, 4, 4, 3, 3, 4, 3, 4, 3, 3, 4, 3, 3, 3, 4, 4, 4, 3, 3, 4, 4, 3, 4, 3, 4, 3, 3, 3, 4, 3, 4, 4, 4, 4,
3, 4, 3, 4, 3, 4, 4, 4, 3, 4, 3, 4, 3, 4, 4, 3, 4, 3, 3, 4, 3, 3, 4, 4, 4, 3, 3, 3, 3, 4, 3, 4, 4, 3, 3, 4, 3,
3, 4, 3, 4, 3, 4, 3]

Initial State: 4, Steps: 1000, Trajectory: [4, 4, 4, 4, 3, 4, 3, 4, 3, 4, 4, 4, 3, 4, 4, 3, 4, 4, 3, 4,
3, 3, 3, 3, 3, 4, 3, 3, 4, 3, 4, 3, 4, 4, 3, 4, 3, 4, 4, 4, 4, 4, 4, 3, 3, 4, 3, 4, 3, 3, 3, 3, 4, 3, 3, 3, 4,
4, 4, 4, 4, 3, 3, 3, 3, 3, 4, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 3, 3, 3, 4, 3, 3, 4,
3, 4, 3, 3, 3, 3, 3, 4, 3, 3, 4, 4, 3, 3, 4, 3, 4, 3, 4, 3, 4, 4, 4, 3, 3, 4, 4, 4, 4, 4, 4, 3, 4, 3, 3, 3, 3,
3, 3, 4, 3, 4, 3, 4, 3, 4, 4, 3, 3, 3, 4, 3, 4, 3, 4, 4, 3, 4, 3, 4, 4, 3, 4, 4, 4, 3, 3, 4, 4, 3, 3, 3, 4, 3,
3, 3, 4, 4, 4, 3, 4, 3, 3, 4, 3, 4, 3, 4, 4, 3, 4, 3, 4, 4, 3, 4, 4, 4, 3, 3, 3, 4, 4, 4, 3, 4, 3, 3, 4, 4, 3,
3, 4, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 4, 4, 4, 3, 4, 4, 4, 4, 3, 3, 4, 3, 3, 3, 4, 4, 3, 4, 3,
3, 4, 3, 3, 3, 3, 3, 4, 4, 4, 3, 4, 4, 3, 3, 3, 3, 3, 3, 3, 4, 3, 3, 4, 4, 4, 4, 3, 4, 3, 4, 3, 4, 4, 3, 4, 4,
3, 3, 3, 4, 3, 4, 4, 3, 3, 4, 3, 4, 3, 3, 4, 4, 3, 4, 4, 4, 3, 3, 3, 3, 3, 4, 4, 3, 4, 3, 3, 4, 4, 4, 4, 4, 4,
3, 3, 3, 4, 3, 4, 4, 4, 3, 3, 4, 4, 4, 4, 3, 4, 4, 4, 4, 3, 3, 3, 4, 4, 3, 3, 3, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4,
4, 4, 3, 3, 3, 3, 3, 4, 4, 3, 4, 3, 4, 3, 3, 3, 4, 4, 3, 4, 3, 3, 3, 3, 3, 4, 3, 4, 3, 3, 3, 3, 3, 4, 3, 3, 3,
4, 3, 4, 3, 3, 3, 4, 4, 4, 3, 3, 3, 4, 3, 4, 3, 4, 4, 3, 3, 4, 4, 4, 4, 3, 4, 3, 3, 4, 3, 3, 3, 4, 4, 4, 3, 4,
4, 3, 4, 4, 4, 4, 3, 3, 3, 4, 4, 3, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4, 3, 3, 4, 4, 4, 4, 4, 4, 3, 3, 4, 4, 4, 4, 3,
3, 4, 4, 3, 3, 4, 4, 4, 4, 3, 4, 4, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 3, 3, 4, 4, 4, 3, 4, 3, 3, 4, 3, 4,
3, 4, 4, 3, 3, 3, 3, 3, 4, 4, 3, 3, 3, 3, 3, 3, 4, 3, 3, 4, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 3, 3, 3, 3, 3, 4, 3,
4, 3, 3, 4, 4, 3, 4, 4, 3, 3, 3, 4, 4, 4, 3, 4, 3, 3, 4, 3, 4, 4, 3, 4, 3, 3, 4, 3, 4, 4, 3, 4, 3, 3, 3, 4, 4,
3, 4, 3, 4, 4, 4, 4, 3, 4, 3, 4, 3, 4, 3, 3, 4, 3, 3, 3, 3, 4, 3, 4, 3, 3, 3, 4, 3, 3, 4, 3, 4, 3, 4, 4, 4, 3,
4, 3, 4, 3, 3, 3, 4, 4, 3, 3, 3, 3, 4, 3, 3, 3, 4, 3, 3, 3, 4, 3, 4, 4, 4, 4, 3, 3, 3, 4, 4, 3, 3, 3, 3, 3, 4,
4, 3, 3, 3, 3, 4, 3, 3, 3, 4, 4, 4, 3, 3, 4, 3, 4, 4, 3, 4, 4, 3, 3, 4, 4, 4, 3, 3, 4, 4, 4, 4, 4, 4, 3, 3, 3,
4, 3, 4, 3, 3, 4, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 3, 3, 4, 3, 3, 4, 4, 4, 4, 4, 3, 4, 3, 4,
4, 3, 4, 4, 4, 4, 4, 3, 4, 4, 4, 3, 4, 4, 3, 4, 3, 3, 4, 3, 3, 3, 3, 3, 3, 4, 3, 4, 4, 3, 4, 3, 4, 3, 4, 3, 3,
3, 3, 3, 4, 3, 3, 3, 3, 3, 4, 3, 4, 4, 4, 3, 4, 4, 4, 4, 3, 3, 4, 3, 3, 3, 4, 4, 3, 4, 3, 4, 4, 4, 4, 4, 4, 4,
3, 4, 3, 3, 3, 4, 4, 3, 4, 3, 4, 3, 4, 3, 3, 3, 4, 3, 3, 4, 4, 4, 4, 3, 4, 3, 3, 3, 3, 3, 3, 3, 4, 4, 3, 4, 3,
3, 4, 3, 4, 4, 4, 3, 3, 3, 4, 4, 3, 3, 3, 4, 3, 4, 3, 4, 3, 3, 3, 4, 3, 4, 3, 4, 3, 3, 4, 3, 4, 4, 3, 4, 4, 4,
4, 3, 4, 3, 3, 3, 3, 3, 3, 4, 3, 4, 4, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 4, 4, 4, 3, 3, 4, 4, 3, 3, 4, 3, 4, 3,
4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 4, 3, 4, 4, 3, 4, 4, 3, 3,

3, 4, 3, 4, 4, 4, 3, 4, 3, 4, 4, 3, 4, 3, 4, 4, 4, 3, 3, 3, 3, 3, 4, 3, 3, 3, 3, 3, 3, 3, 4, 3, 3, 4, 3, 4, 3, 3, 3, 3, 3, 4, 4, 3, 3, 3, 3, 4, 3, 4, 3, 3, 3, 3, 3, 4]

По данным траекториям видно, что не происходит выхода из компоненты связности.

2 Пункт 7

Граф марковской цепи был разбит на 3 класса в соответствии с пунктом 5. Для первого класса {v1,v6} и второго класса {v2, v3} были взяты матрицы переходов через 2. Третий класс - {v4, v5} Также программно были пересчитаны значения векторов финальных вероятностей.

```
transition_matrix1 = np.array([
    [ 46/100, 54/100],
    [ 5/10, 5/10]
])
transition_matrix2 = np.array([
    [58/100, 42/100],
    [62/100, 38/100]
])

transition_matrix3 = np.array([
    [0.5, 0.5],
    [0.5, 0.5]])

def simulate_markov_chain(transition_matrix, initial_state, steps):
    current_state = initial_state
    states = [current_state]

    for _ in range(steps):
        current_state = np.random.choice(
            a=len(transition_matrix),
            p=transition_matrix[current_state]
        )
        states.append(current_state)

    return states

def calculate_state_percentages(states, num_states):
    counts = np.bincount(states, minlength=num_states)
    percentages = counts / len(states)
    return percentages

def stationary_distribution(transition_matrix):
    eigenvalues, eigenvectors = np.linalg.eig(transition_matrix.T)
    stationary_vector = eigenvectors[:, np.isclose(eigenvalues, 1)]
    stationary_vector = stationary_vector / np.sum(stationary_vector)
    return stationary_vector.real.flatten()

def simulation(steps, initial_state, transition_matrix):
  print(f"Initial state: {initial_state}, steps: {steps} ")

  trajectory = simulate_markov_chain(transition_matrix, initial_state, steps)

  percentages = calculate_state_percentages(trajectory, len(transition_matrix))
  print(f"Percentages: {percentages}")
  stationary_vector = stationary_distribution(transition_matrix)
```

```python
    print(f"Final distribution: {stationary_vector}")

    difference = np.abs(percentages - stationary_vector)
    print(f"Difference beetween percentages and final distribution: {
        difference}\n")

print("Class 1: \n")
simulation(10, 0, transition_matrix1)
simulation(50, 1, transition_matrix1)
simulation(100, 0, transition_matrix1)
simulation(1000, 1, transition_matrix1)

print("\nClass 2:")
simulation(10, 0, transition_matrix2)
simulation(50, 1, transition_matrix2)
simulation(100, 0, transition_matrix2)
simulation(1000, 1, transition_matrix2)

print("\nClass 3:")
simulation(10, 0, transition_matrix3)
simulation(50, 1, transition_matrix3)
simulation(100, 0, transition_matrix3)
simulation(1000, 1, transition_matrix3)
```

Были получены следующие результаты:

```
Class 1:

Initial state: 0, steps: 10
Percentages: [0.36363636 0.63636364]
Final distribution: [0.48076923 0.51923077]
Difference beetween percentages and final distribution: [0.11713287 0.11713287]

Initial state: 1, steps: 50
Percentages: [0.45098039 0.54901961]
Final distribution: [0.48076923 0.51923077]
Difference beetween percentages and final distribution: [0.02978884 0.02978884]

Initial state: 0, steps: 100
Percentages: [0.5049505 0.4950495]
Final distribution: [0.48076923 0.51923077]
Difference beetween percentages and final distribution: [0.02418126 0.02418126]

Initial state: 1, steps: 1000
Percentages: [0.47452547 0.52547453]
Final distribution: [0.48076923 0.51923077]
Difference beetween percentages and final distribution: [0.00624376 0.00624376]


Class 2:
Initial state: 0, steps: 10
Percentages: [0.45454545 0.54545455]
Final distribution: [0.59615385 0.40384615]
Difference beetween percentages and final distribution: [0.14160839 0.14160839]
```

```
Initial state: 1, steps: 50
Percentages: [0.52941176 0.47058824]
Final distribution: [0.59615385 0.40384615]
Difference beetween percentages and final distribution: [0.06674208 0.06674208]

Initial state: 0, steps: 100
Percentages: [0.56435644 0.43564356]
Final distribution: [0.59615385 0.40384615]
Difference beetween percentages and final distribution: [0.03179741 0.03179741]

Initial state: 1, steps: 1000
Percentages: [0.61238761 0.38761239]
Final distribution: [0.59615385 0.40384615]
Difference beetween percentages and final distribution: [0.01623377 0.01623377]


Class 3:
Initial state: 0, steps: 10
Percentages: [0.63636364 0.36363636]
Final distribution: [0.5 0.5]
Difference beetween percentages and final distribution: [0.13636364 0.13636364]

Initial state: 1, steps: 50
Percentages: [0.54901961 0.45098039]
Final distribution: [0.5 0.5]
Difference beetween percentages and final distribution: [0.04901961 0.04901961]

Initial state: 0, steps: 100
Percentages: [0.46534653 0.53465347]
Final distribution: [0.5 0.5]
Difference beetween percentages and final distribution: [0.03465347 0.03465347]

Initial state: 1, steps: 1000
Percentages: [0.47952048 0.52047952]
Final distribution: [0.5 0.5]
Difference beetween percentages and final distribution: [0.02047952 0.02047952]
```

Можно увидеть, что чем длиннее траектория, тем ближе вектор процентов времени нахождения к вектору финальных распределений. Вектора финальных вероятностей совпали с посчитанными в пункте 5.