Exercise 4: Graph analysis

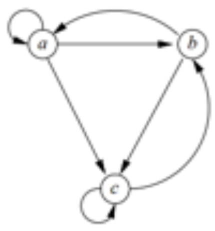
Kyrylo Kolesnichenko

153371848

Task 4.1 (3p)

Compute PageRank for each node in the following graph assuming

- a) no taxation
- b) β = 0.8



So, we are given a graph with three nodes: a with 3 out-links, b with 2, and c with 2 as well. Based on those connections, first we will build the transition matrix with this data.

Then we write a function that will update scores, and it is based on the formula $r_j = \sum_{i \to j} \frac{r_i}{d_i}$. So each node checks rank from its incoming neighbors and divides it by the number of their out-links.

We run it two times with B = 1 (no taxation) and B = 0.8.

In [4]:
```python
import numpy as np

M = np.array([
    [1/3, 1/2,   0  ],
    [1/3,   0 , 1/2 ],
    [1/3, 1/2, 1/2  ]
])

def pagerank(M, beta=1.0, tol=1e-6, max_iter=100):
    N = M.shape[0]
    r = np.ones(N) / N
    teleport = (1 - beta) / N
    for _ in range(max_iter):
        r_new = beta * (M @ r) + teleport * np.ones(N)
        if np.linalg.norm(r_new - r, 1) < tol:
            return r_new / np.sum(r_new)
        r = r_new
    return r / np.sum(r)

pr_no_damping   = pagerank(M, beta=1.0)
pr_with_damping = pagerank(M, beta=0.8)

labels = ['a', 'b', 'c']
print("PageRank β = 0.8:")
for lbl, val in zip(labels, pr_no_damping):
    print(f"  P({lbl}) = {val:.6f}")

print("\nPageRank β = 0.8:")
for lbl, val in zip(labels, pr_with_damping):
    print(f"  P({lbl}) = {val:.6f}")
```

```
PageRank β = 0.8:
  P(a) = 0.230769
  P(b) = 0.307692
  P(c) = 0.461538

PageRank β = 0.8:
  P(a) = 0.259259
  P(b) = 0.308642
  P(c) = 0.432099
```