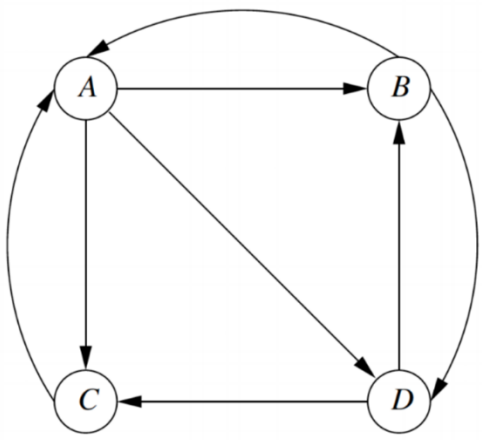
HOMEWORK 3

Jie Tang

OCTOBER 22, 2020

1 [20 marks]

Given the following graph.



a) Write down the transition matrix for the graph [10 marks]

b) Compute the PageRank of each page (describe the process and the results) [10 marks]

a) Our flow equation is as follows:

$$r_A = \frac{1}{2} r_B + r_C$$
$$r_B = \frac{1}{3} r_A + \frac{1}{2} r_D$$
$$r_C = \frac{1}{3} r_A + \frac{1}{2} r_D$$
$$r_D = \frac{1}{3} r_A + \frac{1}{2} r_B$$

So the transition matrix is:

$$M = \begin{bmatrix} 0 & 1/2 & 1 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

b) Here are two methods we can use to solve this equation:

i) since there are four equations with four variables, so the solution will not be unique, so we can add a constrian to the above flow equation: $r_A + r_B + r_C + r_D = 1$

ii) we can use power iteration to solve the equation. The flow equation can be written as: **M * r = r**

Power iteration: a simple iterative scheme

- Suppose there are N web pages
- Initialize: r^(0) = [1/N,….,1/N]T
- Iterate: r^(t+1) = M · r^(t)
- Stop when $|r^{(t+1)} - r^{(t)}|_1 < \varepsilon$

Since there are only 4 variables, I chose the first method:

$$r_A = \frac{1}{2}r_B + r_C \qquad 2r_A = r_B + 2r_C$$
$$r_B = \frac{1}{3}r_A + \frac{1}{2}r_D \qquad 6r_B = 2r_A + 3r_D$$
$$r_C = \frac{1}{3}r_A + \frac{1}{2}r_D \quad \text{------>} \quad 6r_C = 2r_A + 3r_D$$
$$r_D = \frac{1}{3}r_A + \frac{1}{2}r_B \qquad 6r_D = 2r_A + 3r_B$$
$$r_A + r_B + r_C + r_D = 1 \qquad r_A + r_B + r_C + r_D = 1$$

Then I got:

$$\frac{2}{3}r_A = r_B = r_C = r_D$$
$$r_A + r_B + r_C + r_D = 1$$

Finally: $\boldsymbol{r_B = r_C = r_D = \frac{2}{9},\ r_A = \frac{3}{9}}$

2 [30 marks]

Write a hadoop program to complete the PageRank calculation on Berkeley-Stanford web graph. Set the damping factor to 0.85 and run the algorithm for a total of 10 iterations. Report the NodeId of Top 100 pages and the corresponding PageRank scores.

Here is the link for Berkeley-Stanford web graph. The dataset is stored in a TXT file, where each line denotes an edge. The following screenshot shows the first several lines of this file.

```
# Directed graph (each unordered pair of nodes is saved once): web-BerkStan.txt
# Berkely-Stanford web graph from 2002
# Nodes: 685230 Edges: 7600595
# FromNodeId    ToNodeId
1       2
1       5
1       7
1       8
1       9
1       11
1       17
1       254913
1       438238
254913  255378
254913  255379
254913  255383
254913  255384
254913  255392
```

The algorithm is like the following:

- **Input:** Graph G and parameter β
  - Directed graph G (can have **spider traps** and **dead ends**)
  - Parameter β
- **Output:** PageRank vector r^{new}
  - **Set:** $r_j^{old} = \frac{1}{N}$
  - **repeat until convergence:** $\sum_j |r_j^{new} - r_j^{old}| > \varepsilon$
    - $\forall j$: $r'^{new}_j = \sum_{i \to j} \beta \frac{r_i^{old}}{d_i}$

      $r'^{new}_j = 0$ if in-degree of j is 0
    - **Now re-insert the leaked PageRank:**

      $\forall j$: $r_j^{new} = r'^{new}_j + \frac{1-S}{N}$ where: $S = \sum_j r'^{new}_j$
    - $r^{old} = r^{new}$

If the graph has no dead-ends then the amount of leaked PageRank is 1-β. But since we have dead-ends the amount of leaked PageRank may be larger. We have to explicitly account for it by computing S.

Instead of using Hadoop, I used PySpark to do this exercise. And I found it much easier to use PySpark since there are already maps and reduces function built inside. It took me **373** seconds to run the codes. Here is what I got from my codes.

```
top 100 links are as follows:
272919 has rank: 5135.337685312974.
438238 has rank: 3491.0960312742745.
571448 has rank: 1864.6376899641693.
601656 has rank: 1602.690114457544.
319209 has rank: 1399.2280593912456.
316792 has rank: 1332.0106563322552.
184094 has rank: 1282.3453401223012.
401873 has rank: 1180.345170594501.
571447 has rank: 1148.8928422268498.
284306 has rank: 1138.1227323344524.
768 has rank: 1082.2358974713154.
927 has rank: 1010.7247942388504.
66244 has rank: 973.5471932820052.
68949 has rank: 972.971066110794.
68948 has rank: 965.9204723030336.
95552 has rank: 954.2712527121563.
77284 has rank: 954.2712527121563.
86237 has rank: 954.2712527121563.
95551 has rank: 954.2712527121563.
68947 has rank: 954.2712527121563.
96070 has rank: 954.2712527121563.
86238 has rank: 954.2712527121563.
68946 has rank: 954.2712527121563.
86239 has rank: 954.2712527121563.
66909 has rank: 954.2712527121563.
184142 has rank: 781.19488314776.
299039 has rank: 770.664978342663.
571451 has rank: 749.2130806752585.
570985 has rank: 744.4415127078548.
299040 has rank: 721.5266967317406.
319412 has rank: 717.3411614327924.
```

Notes: codes are in another file.