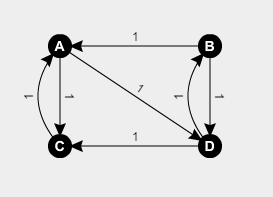
1.1)



1.2) The values per column have to add up to 1, so we fill in the matrix per column. If the node of the column has a link to the node of a row, the value in that cell will be 1/d, with d being the out-degree of the column-node.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A | B | C | D |
| A | 0 | ½ | 1 | 0 |
| B | 0 | 0 | 0 | ½ |
| C | ½ | 0 | 0 | ½ |
| D | ½ | ½ | 0 | 0 |

1.3)

The chance that after one traversal, the surfer is at a node X, is equal to the sum of the chances that the node is randomly initiated at a node which links to node X, and that node chooses the link to get to X. Every link is equally likely to be chosen.

P(A) = (¼ \* ½) + (¼ \* 1) = 3/8

P(B) = (¼ \* ½) = 1/8

P(C) = (¼ \* ½) + (¼ \* ½) = 2/8

P(D) = (¼ \* ½) + (¼ \* ½) = 2/8

The chance sum up to 1, as they should.

Step2

The data is stored as follows :

{A={A=0, B=0, C=1, D=1}, B={A=1, B=0, C=0, D=1}, C={A=1, B=0, C=0, D=0}, D={A=0, B=1, C=1, D=0}}

8.1) What we observe is that C has now become a spider-trap. As the number of iterations increases, the rank of C goes to 1, and the ranks of the other nodes go to 0. The reason is that once the surfer goes to C, he can’t go to any other nodes, so all the “importance” will go to C alone.

9.1) Yes, the results are better. All the importance is no longer concentrated in C, although C is still very “important”.

9.2) If you set beta to 0, then at every iteration the surfer will make a random jump, which will result in an equal rank for all nodes. If you set it to 1, then the surfer will never jump, and the result is the same as with the normal PageRank algorithm. It will still be crippled by spider-traps.