Tutorial 8: Large Graph Processing II

Li Yuan

li.yuan@u.nus.edu

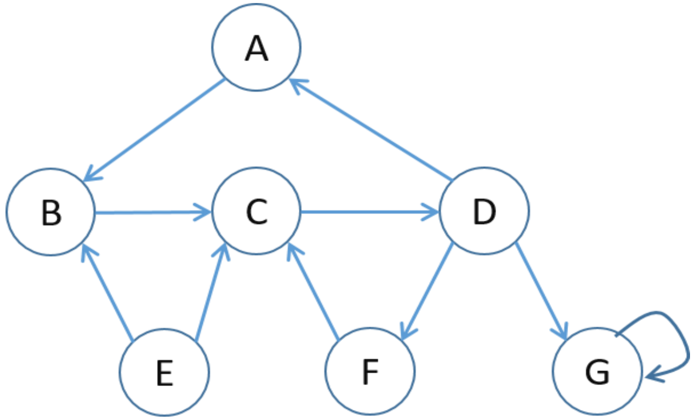
National University of Singapore

Problem 1

1. Given the following graph,

1) how many dead ends are there in the graph? For each dead end (if any), please indicate the set of vertices forming the dead end.

2) how many spider traps are there in the graph? For each spider trap (if any), please indicate the set of vertices forming the spider trap.

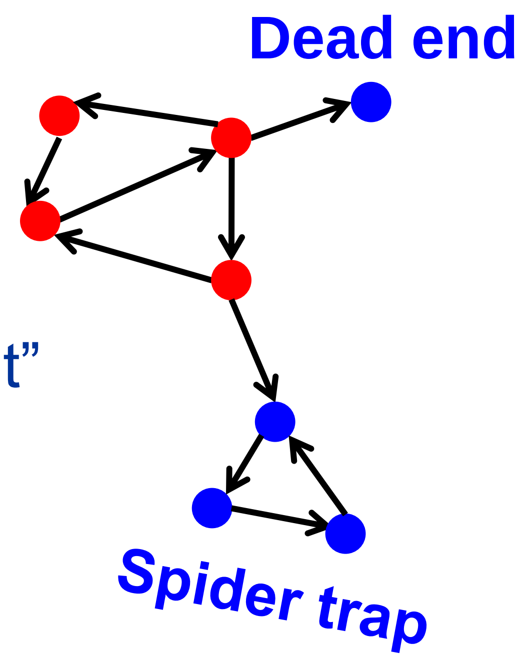


2 problems:

(1) Some pages are
dead ends (have no out-links)

Random walk has "nowhere" to go to

Such pages cause importance to "leak out"



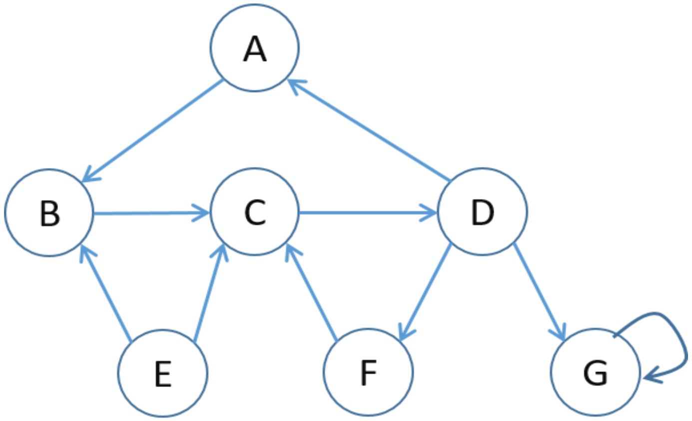
(2) Spider traps:
(all out-links are within the group)

Random walked gets "stuck" in a trap

And eventually spider traps absorb all importance

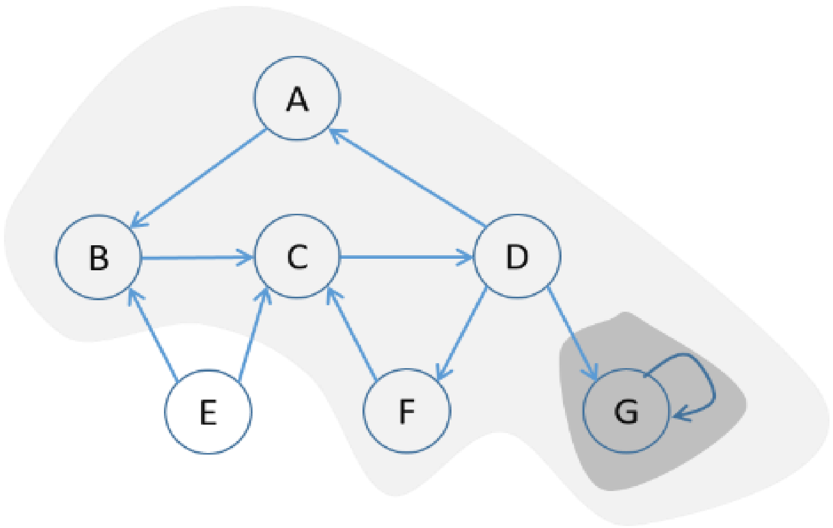
Solution 1

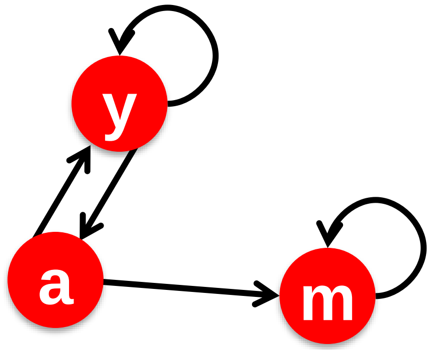
1) No dead ends



Solution 1

1) No dead ends
2) Two spider traps





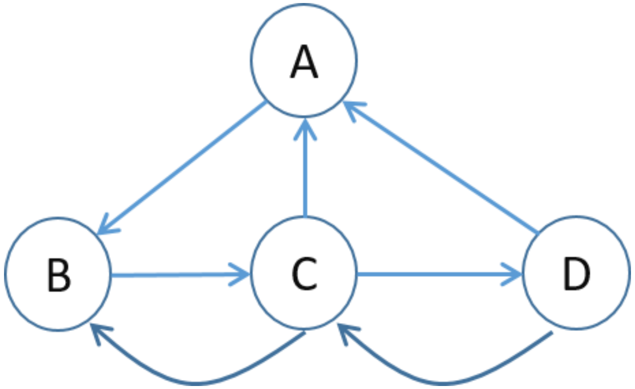
$$A = \beta M + (1-\beta)\left[\frac{1}{N}\right]_{N\times N}$$

$$\beta \times \begin{array}{c|c} & \begin{array}{ccc} \mathrm{y} & \mathrm{a} & \mathrm{m} \end{array} \\ \begin{array}{c} \mathrm{y} \\ \mathrm{a} \\ \mathrm{m} \end{array} & \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 1 \end{bmatrix} \end{array} + (1-\beta) \times \begin{array}{c|c} & \begin{array}{ccc} \mathrm{y} & \mathrm{a} & \mathrm{m} \end{array} \\ \begin{array}{c} \mathrm{y} \\ \mathrm{a} \\ \mathrm{m} \end{array} & \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix} \end{array} = \boldsymbol{A}$$

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \boldsymbol{A} \cdot \begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix}$$

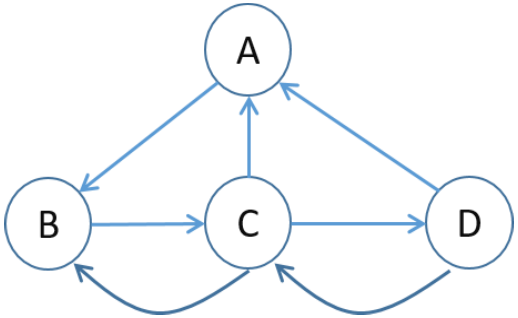
Problem 2

Set up the PageRank equations for the below graph, assuming $\beta = 0.8$ (jump probability = $1-\beta$). Denote the PageRank of node x by $r(x)$.



Solution 2

$$M = \begin{bmatrix} 0 & 0 & \frac{1}{3} & \frac{1}{2} \\ 1 & 0 & \frac{1}{3} & 0 \\ 0 & 1 & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{3} & 0 \end{bmatrix} \qquad \mathrm{N} = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix}$$



Solution 2

$$A = \beta M + (1-\beta)N$$
$$r = Ar = \beta Mr + (1-\beta)Nr$$

$$r(A) = \frac{4}{15}r(C) + \frac{2}{5}r(D) + \frac{1}{20}(r(A) + r(B) + r(C) + r(D))$$
$$r(B) = \frac{4}{5}r(A) + \frac{4}{15}r(C) + \frac{1}{20}(r(A) + r(B) + r(C) + r(D))$$
$$r(C) = \frac{4}{5}r(B) + \frac{2}{5}r(D) + \frac{1}{20}(r(A) + r(B) + r(C) + r(D))$$
$$r(D) = \frac{4}{15}r(C) + \frac{1}{20}(r(A) + r(B) + r(C) + r(D))$$

Problem 3

3. Suppose you have a large graph, and you will implement breadth first traversal on the graph. Each vertex has three attributes: 1) *id*, the vertex ID, 2) *isVisited*, indicating the vertex has been visited or not, and 3) *vList*, the list of neighbour vertices. Show the pseudo code on how you would use Pregel to perform a breadth first traversal on the graph, starting with a pre-defined vertex V_0. You must follow the pseudo code of the below format.

```
compute (vertex v) {
/* your pseudo code*/
/* you can use two APIs given in Pregel:
1) getSuperStep(): Retrieves the current superstep;
2) voteToHalt(): After this is called, the compute() code will no longer be called for this vertex.*/
}
```

Pregel: Computational Model

Based on Bulk Synchronous Parallel (BSP)

- Computational units encoded in a directed graph
- Computation proceeds in a series of supersteps
- Message passing architecture

Each vertex, at each superstep:

- Receives messages directed at it from previous superstep
- Executes a user-defined function (modifying state)
- Emits messages to other vertices (for the next superstep)

Termination:

- A vertex can choose to deactivate itself
- Is "woken up" if new messages received
- Computation halts when all vertices are inactive

```
compute (vertex v) {
    if(getSuperStep() == 0){
        if(v == V0) {
            send messages to v's neighbors.
            isVisited=true;
        }
    }
    else if (isVisited == false) {
        if(there is a message for v){
            send messages to v's neighbors.
            isVisited=true;
        }
    }
    v.voteToHalt();
}
```

Acknowledgement

Thanks to Li Qinbin for making these slides.

liqinbin@u.nus.edu