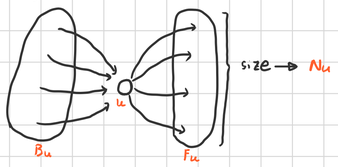
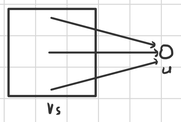
- PageRank Algorithm's basic theorems

[Poy highlight 01]





Transfer contributions

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v}$$

$c < 1$: there are a number of pages with on forward links, and their weight is lost from the system

[Poy highlight 02] Use matrix for computing & Simple Model

Initial Status:

$A =$ [ ]

$A_{u,v} = 1/N_u$ iff $u \to v$

$A_{u,v} = 0$ iff $u \not\to v$

Aim: find a vector $R = (r_1\ r_2\ \dots\ r_n)^T$, $R[i]$ = Page i's important value

then

$$cA^TR = c \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix} \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{pmatrix} = \begin{pmatrix} c\sum_{j \in B_1} \frac{R_j}{N_j} \\ \vdots \\ \vdots \end{pmatrix} \rightsquigarrow A^TR = \frac{1}{c}R$$

c | A^T | R

R: eigenvector; $\frac{1}{c}$: eigenvalue

↳ want dominant eigenvector

$A^T_{u,v} = 1/N_u$ iff $v \to u$

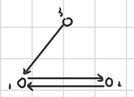
$A^T_{u,v} = 0$ iff $v \not\to u$

[Poy highlight 02 below] Two problems & Completed / Random Surfur Model

"rank sink": there are some nodes with 0-indegree

① The node with 0 indegree's contribution will be zero;

② The node with 0 in-degree finally has no influence to the others;



$$A = \begin{bmatrix} 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 0 & 0 \end{bmatrix}$$

for each $A_{u,v}$: $u \to v$

row vector = out-edges

$$A^T = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

for each $A^T_{u,v}$: $v \to u$

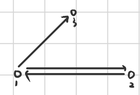
row vector = in-edges

① so if there is a node with 0 in-degree, surely, the eigenvector's corresponding value will be zero.

② because the 0 in-degree's contribution is zero, so its out-edge will have no influence to other node

"rank leak": there are some nodes with 0-outdegree

① Finally, all node which link to the node with 0-outdegree will be zero contribution



$$A = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

for each A_{ij}: $i \to j$

row vector = out-edges

$$A^T = \begin{bmatrix} 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 0 & 0 \end{bmatrix}$$

always < 1

for each A^T_{ij}: $j \to i$

row vector = in-edges

For each time, the node with 0-ourdegree have on contribution to the others, but the others which link to it save < 1 ratio than last time, so finally, their contribution → 0

Solution:

Let the "pure matrix A" becomes a "stable probability stochastic matrix A"

$$R'(u) = c \sum_{v \in B_u} \frac{R'(v)}{N_v} + cE(u)$$

$$R' = c(A^T + E \times 1)R'$$

$$R(u) = \frac{1-c}{N} + c \sum_{v \in B_u} \frac{R(v)}{N_v}$$

N: All pages Number

c: Damping Factor (usually is 0.85)

Random Surfer Model

Proof of convergence: Markov chain convergence theorem

- Implement PageRank Algorithm by Hadoop MapReduce

Implement Idea:

3 parts:

1. GraphBuilder
2. PageRankIter
3. RankViewer

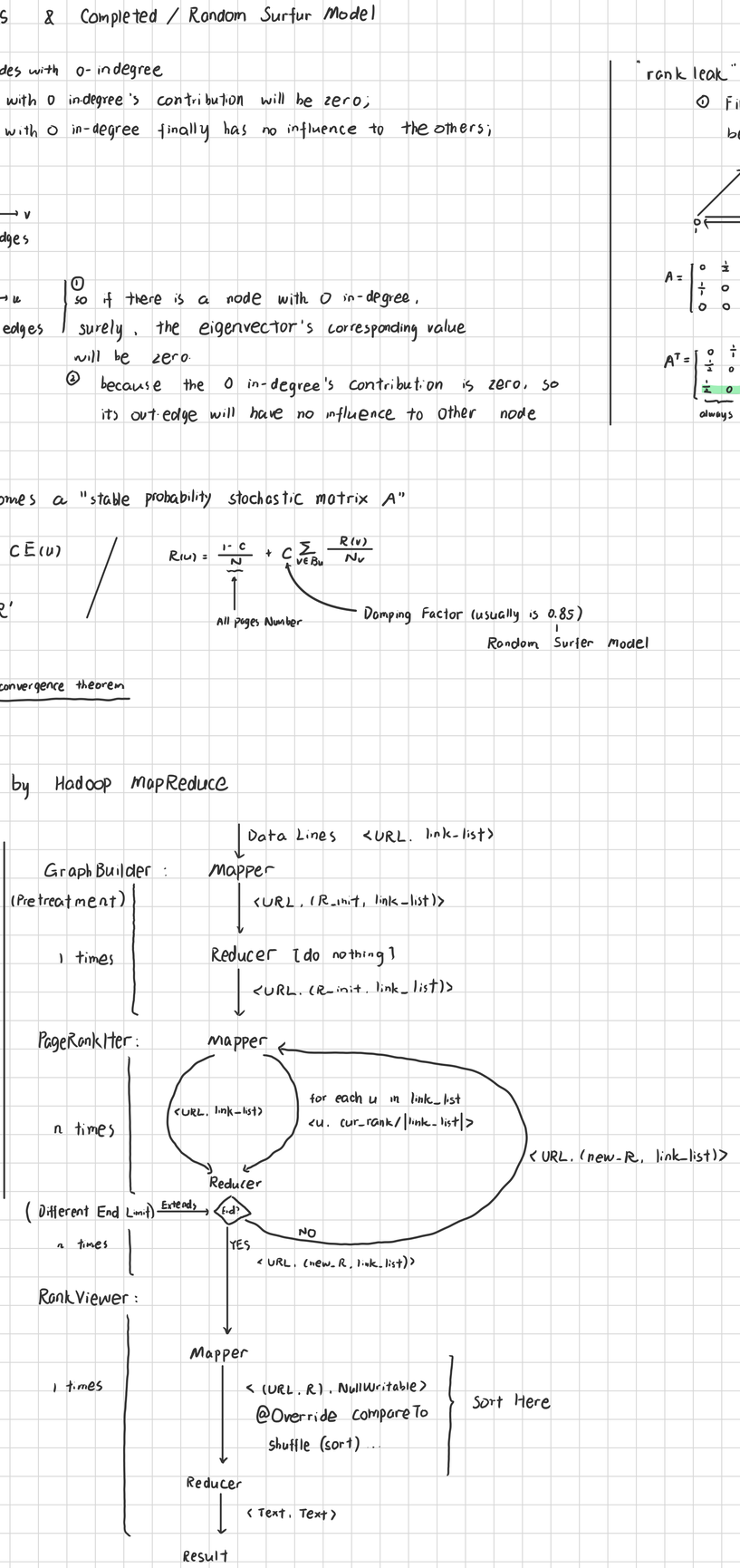
2 parallel parts:

$$R(u) = \frac{1-d}{N} + d \sum_{v \in B_u} \frac{R(v)}{N_v}$$

MapReduce Job Iteration

especially

1. Mapper parallel op
2. Reducer parallel op



Hard Points

① How to use Hadoop MapReduce iteration workflow? ☑

② How to implement and apply a extendable end limit class & methods? ☐

③ How to deal with input/output files store, especially for "Pretreatment" → "Real PageRank" ☑

More problem: How to set initial PageRank value for each page? ☐

How to get all pages number whink is the parameter N in

$$R(u) = \frac{1-c}{N} + c \sum_{v \in B_u} \frac{R(v)}{N_v}$$ ☑

(use a single process for scanning, but how to optimize it?) ☐