PageRank: The Complete Algorithm

- **Input: Graph G and parameter β**
  - Directed graph G with **spider traps** and **dead ends**
  - Parameter β
- **Output: PageRank vector r**
  - **Set:** $r_j^{(0)} = \frac{1}{N}, \quad t = 1$
  - **do:**
    - $\forall j$: $\boldsymbol{r'}_j^{(t)} = \sum_{i \to j} \boldsymbol{\beta} \frac{r_i^{(t-1)}}{d_i}$

      $\boldsymbol{r'}_j^{(t)} = \mathbf{0}$ if in-deg. of $\boldsymbol{j}$ is $\mathbf{0}$
    - **Now re-insert the leaked PageRank:**

      $\forall \boldsymbol{j}$: $\boldsymbol{r}_j^{(t)} = \boldsymbol{r'}_j^{(t)} + \frac{\mathbf{1}-\boldsymbol{S}}{\boldsymbol{N}}$ **where:** $S = \sum_j r'^{(t)}_j$
    - $\boldsymbol{t} = \boldsymbol{t} + \mathbf{1}$
  - **while** $\sum_j \left| r_j^{(t)} - r_j^{(t-1)} \right| > \varepsilon$

Sparse Matrix Encoding

- **Encode sparse matrix using only nonzero entries**
  - Space proportional roughly to number of links
  - Say 10N, or 4*10*1 billion = 40GB
  - **Still won't fit in memory, but will fit on disk**

source node	degree	destination nodes
0	3	1, 5, 7
1	5	17, 64, 113, 117, 245
2	2	13, 23

Basic Algorithm: Update Step

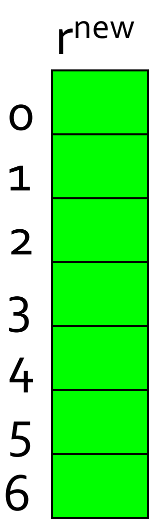
- **Assume enough RAM to fit r^{new} into memory**
  - Store r^{old} and matrix **M** on disk
- **Then 1 step of power-iteration is:**

Initialize all entries of r^{new} to $(1-\beta)/N$

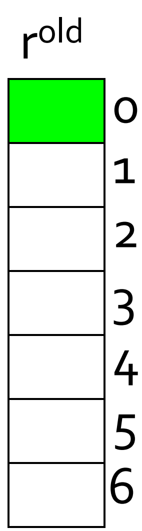
For each page *p* (of out-degree *n*):

Read into memory: $p, n, dest_1, \ldots, dest_n, r^{old}(p)$

for j = 1…n: $r^{new}(dest_j) \mathrel{+}= \beta\, r^{old}(p) / n$



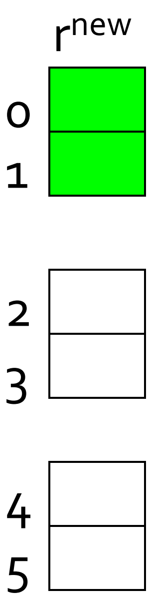
src	degree	destination
0	3	1, 5, 6
1	4	17, 64, 113, 117
2	2	13, 23



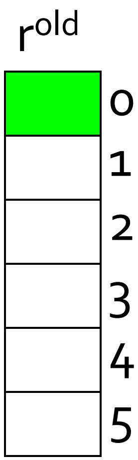
Analysis

- **Assume enough RAM to fit r^{new} into memory**
  - Store $\boldsymbol{r}^{old}$ and matrix **M** on disk
- **In each iteration, we have to:**
  - Read $\boldsymbol{r}^{old}$ and **M**
  - Write $\boldsymbol{r}^{new}$ back to disk
  - Input/Output cost = $2|\boldsymbol{r}| + |\boldsymbol{M}|$
- **Question:**
  - What if we could not even fit $\mathbf{r}^{\text{new}}$ in memory?

Block-based Update Algorithm



src	degree	destination
0	4	0, 1, 3, 5
1	2	0, 5
2	2	3, 4



Analysis of Block Update

- **Similar to nested-loop join in databases**
  - Break $\boldsymbol{r}^{\text{new}}$ into k blocks that fit in memory
  - Scan $\boldsymbol{M}$ and $\boldsymbol{r}^{\text{old}}$ once for each block
- $\boldsymbol{k}$ scans of $\boldsymbol{M}$ and $\boldsymbol{r}^{\text{old}}$
  - $k(|\boldsymbol{M}| + |\boldsymbol{r}|) + |\boldsymbol{r}| = k|\boldsymbol{M}| + (k+1)|\boldsymbol{r}|$
- **Can we do better?**
  - Hint: $\boldsymbol{M}$ is much bigger than $\boldsymbol{r}$ (approx 10-20x), so we must avoid reading it $\boldsymbol{k}$ times per iteration

Block-Stripe Update Algorithm

r^{new}

0
1

src	degree	destination
0	4	0, 1
1	3	0
2	2	1

2
3

0	4	3
2	2	3

4
5

0	4	5
1	3	5
2	2	4

r^{old}

0
1
2
3
4
5

Block-Stripe Analysis

- **Break M into stripes**
  - Each stripe contains only destination nodes in the corresponding block of $\boldsymbol{r}^{\text{new}}$
- Some additional overhead per stripe
  - But it is usually worth it
- Input/Output cost per iteration
  - $|\boldsymbol{M}|(1+\varepsilon) + (k+1)|\boldsymbol{r}|$

Some Problems with Page Rank

- **Measures generic popularity of a page**
  - Biased against topic-specific authorities
  - **Solution:** Topic-Specific PageRank
- **Uses a single measure of importance**
  - Other models e.g., hubs-and-authorities
  - **Solution:** Hubs-and-Authorities
- **Susceptible to Link spam**
  - Artificial link topographies created in order to boost page rank
  - **Solution:** TrustRank