Ranqueamento de Páginas na web

Application: Google's PageRank Algorithm

Goal. Determine which web pages on Internet are important.
Solution. Ignore keywords and content, focus on hyperlink structure.

Random surfer model.

- Start at random page.
- With probability 0.85, randomly selects a link on page to visit next.
  With probability 0.15, randomly select a page.
- Never hit "Back" button.
- PageRank = proportion of time random surfer spends on each page.

Intuition.

- Each page evenly distributes its rank to all pages that it points to.
- Each page receives rank from all pages that point to it.
- Hard to cheat.

Ranqueamento de Páginas na web

When Google was just a research project, they wrote a paper detailing a formula that gives the PageRank for a page. Whilst they may not still be using this exact formula, it seems pretty accurate for today's purposes. Here it is …[1]

$$PR(A) = (1-d) + d\,(PR(T1)/C(T1) + \ldots + PR(Tn)/C(Tn))$$

Where PR(A) is the PageRank of Page A (the one we want to work out).

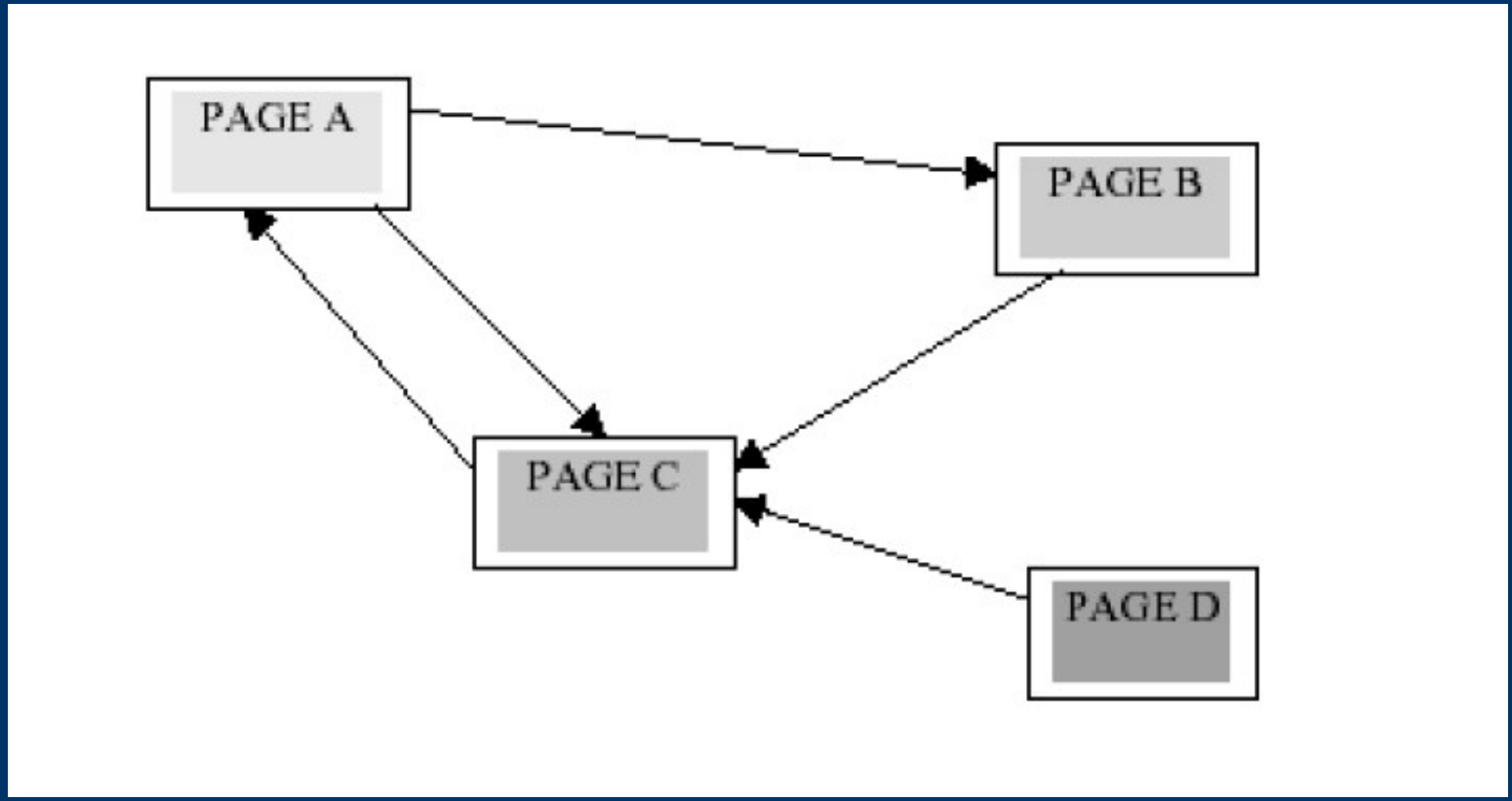
D is a dampening factor. Nominally this is set to 0.85

PR(T1) is the PageRank of a site pointing to Page A

C(T1) is the number of links off that page

PR(Tn)/C(Tn) means we do that for each page pointing to Page A

Exemplo: Ranqueamento



Associar valor 1 para todos inicialmente

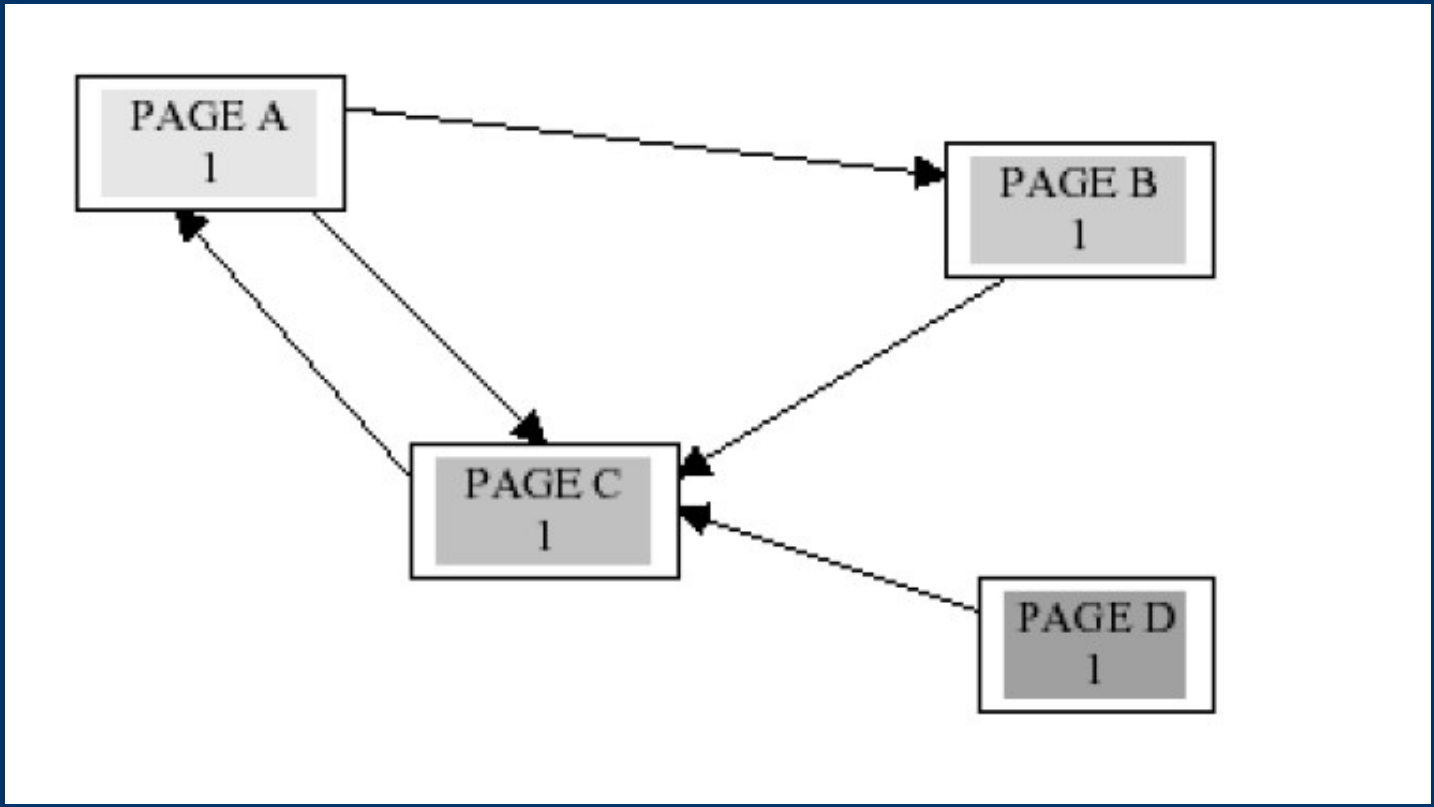


Tabela de iterações seguindo a fórmula

iteração	PR(A)	PR(B)	PR(C)	PR(D)
0	1.0	1.0	1.0	1.0
1	1.0	0.51125	1.8595625	0.15
2	1.730628125	0.885516953125	1.76570636328	0.15
...				

PR(A) = (1-0.85) + 0.85(PR(C)
PR(B) = (1-0.85) + 0.85(PR(A)/2)
PR(C) = (1-0.85) + 0.85(PR(A)/2 + PR(B) + PR(D))
PR(D) = (1-0.85)

Ranqueamento de Páginas na web

Application: Google's PageRank Algorithm

Solution 1: Simulate random surfer for a long time.

Solution 2: Compute ranks directly.

```
for (i = 0; i < PHASES; i++) {
   for (v = 0; v < G->V; v++) oldrank[v] = rank[v];
   for (v = 0; v < G->V; v++) rank[v] = 0;

   for (v = 0; v < G->V; v++) {
      for (t = G->adj[v]; t != NULL; t = t->next) {
         w = t->w;
         rank[w] += 1.0 * oldrank[v] / outdegree[v];
      }
   }
}
```

Solution 3: Compute eigenvalues of adjacency matrix!

Ranqueamento de Páginas na web

Application: Google's PageRank Algorithm

Solution 1: Simulate random surfer for a long time.

Solution 2: Compute ranks directly.

```
for (i = 0; i < PHASES; i++) {
   for (v = 0; v < G->V; v++) oldrank[v] = rank[v];
   for (v = 0; v < G->V; v++) rank[v] = 0;

   for (v = 0; v < G->V; v++) {
      for (t = G->adj[v]; t != NULL; t = t->next) {
         w = t->w;
         rank[w] += 1.0 * oldrank[v] / outdegree[v];
      }
   }
}
```

Solution 3: Compute eigenvalues of adjacency matrix!

Referências

- Cormen, T.; Leiserson, C. & Rivest, R. *Algoritmos: teoria e prática*, Campus Editora, RJ, 2002.
- Sedgewick, R. *Algorithms in C,* 3rd edition, Addison Wesley, EUA, 2002.
- Tenenbaum, A.; Langsam, Y. & Augenstein, M. *Estruturas de Dados usando C*, Makron Books, RJ, 1995.
- Ziviani, N. *Projetos de Algoritmos com Implementações em Pascal e C,* Cengage Learning, SP, 2004.