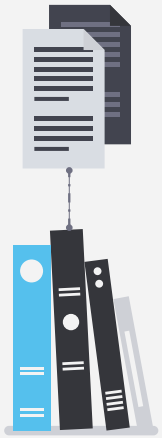
Page Rank

Created by Logan Roche

Table of contents



Introduction

Development and Inspiration

The Page Rank algorithm was developed by Larry Page and Sergey Brin in the late 1990s at Stanford University and became one of the key technologies behind Google's early success. The algorithm was inspired by the academic citation system, where a paper's importance is inferred from the number and quality of citations it receives.

Applications

Beyond web search, Page Rank has been applied to social network analysis, ranking academic papers, identifying influential players in sports, and prioritizing significant genes or proteins in biological networks. Its versatility makes it a valuable tool across many domains.

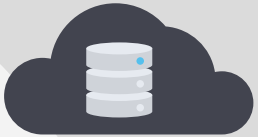
Importance and Legacy

Page Rank significance lies in its historical role in shaping early search engines and its ongoing relevance. The principles behind the algorithm continue to underpin modern information retrieval and network analysis, influencing how we navigate and interpret interconnected data.

Intuition

PageRank assigns a numerical score to each web page in a network, representing the likelihood of a "random surfer" landing on that page. This score is influenced by the structure of incoming links, with links from highly ranked pages contributing more weight. Iterative calculations refine these scores to highlight the most relevant and authoritative pages.

Pseudocode



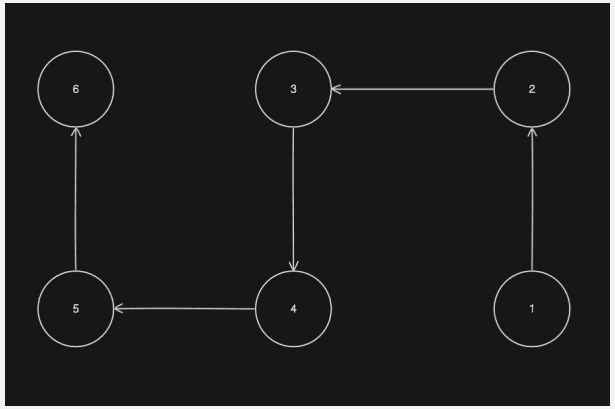
```
Input:
  G = (V, E)  // Graph with vertices V and edges E
  d = 0.85    // Damping factor (commonly set to 0.85)
  ε = 1e-6    // Convergence threshold (small positive value)
  N = |V|     // Number of nodes in the graph
Repeat:
  new_PR[v] = 0 for all v in V   // Temporary storage for updated scores

  for each node v in V:
    Incoming = {u ∈ V | (u, v) ∈ E} // Nodes linking to v
    pagerank_sum = 0

    for each u in Incoming:
      pagerank_sum += PR[u] / OutDegree[u] // Contribution from incoming nodes

    new_PR[v] = (1 - d) / N + d * pagerank_sum // Update PageRank for v

  diff = 0  // Compute total change in PageRank scores
  for each v in V:
    diff += |new_PR[v] - PR[v]| // Measure difference between old and new scores
    PR[v] = new_PR[v]         // Update PageRank values

Until diff < ε  // Stop when PageRank scores converge (total change < threshold)

Return: PR[v] for all v in V
```

Run Time

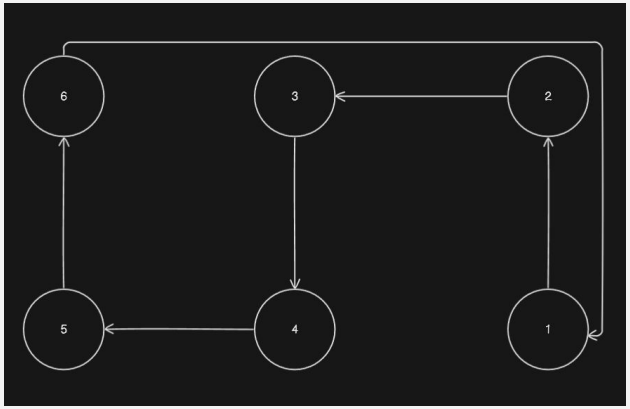
Step-by-Step Analysis

1. **Initialization**:
   - Initializing the PageRank scores to 1/N for all N nodes takes O(N).
2. **Main Loop**:
   - For each iteration:
     - For each node v, compute its new PageRank based on contributions from its incoming neighbors.
     - Across all nodes, this is proportional to the total number of edges E (since the sum of in-degrees equals the total number of edges in the graph).
     - Updating and normalizing PageRank values across N nodes adds an O(N) overhead per iteration.
   - Total cost per iteration is therefore O(E+N)
3. **Convergence**:
   - The algorithm runs for *I* iterations until convergence. *I* depends on the graph's structure and the convergence threshold ϵ, but it is generally considered logarithmic relative to the graph size for most practical cases.
4. **Overall Runtime**:
   - $O(I \cdot (E + N))$
   - *I*: Number of iterations to convergence.
   - *E*: Total number of edges in the graph.
   - *N*: Total number of nodes in the graph.

Implementation

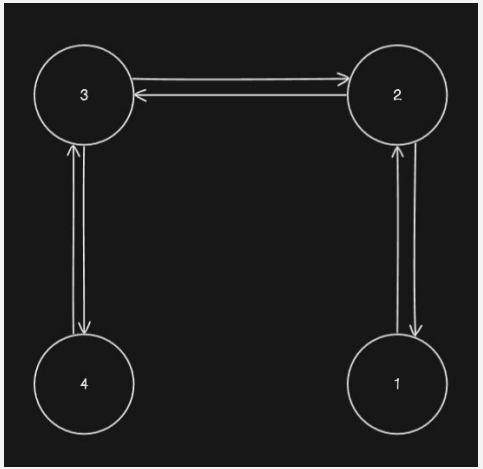



```
Page Rank:
[0.061, 0.112, 0.156, 0.193, 0.225, 0.252]
```

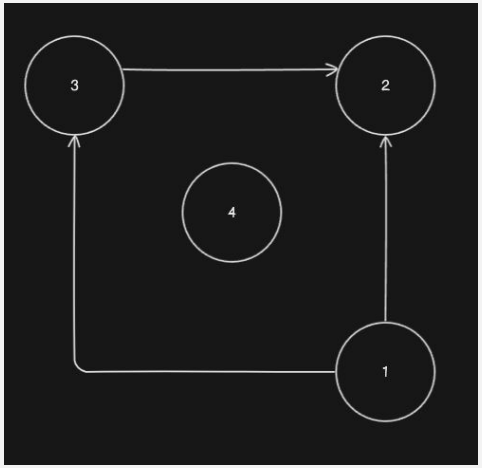



```
Page Rank:
[0.167, 0.167, 0.167, 0.167, 0.167, 0.167]
```

Implementation

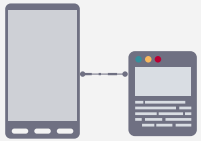



```
Page Rank:
[0.175, 0.325, 0.325, 0.175]
```




```
Page Rank:
[0.116, 0.602, 0.166, 0.116]
```

Thanks!



Feedback

Speaker: Logan Roche

Reviewer 1 (Sloane Tribble): Great slides and presentation. You clearly understand the algorithm well and seemed comfortable explaining it. The intuition was hard to follow without a visual aid, but once you moved on to the next slides it all came together and made sense. Suggestion – could be a good idea to give a more thorough explanation of the math involved in the pseudocode. Loved the implementation slides. Overall good quality and entertaining presentation – nice work!
Reviewer 2 (David Wilson): Very formal presentation slideshow where I could tell the speaker was well versed in the content he was explaining. It was very easy to understand, and was interesting to hear about the history of the algorithm, including how it was used by Google in the past.
Reviewer 3 (Jonathan Tawadros):
The explanation + walking through how the algorithm runs through everything was very good for learning. Looks like implementation for the project is also already done or mostly completed. Seems like you are on your way to finishing this project strong!

Reviewer 4 (Mohsen Amiri): The format of the presentation was really impressive, the presentation was great as well, I loved how you provided examples to demonstrate how it works.

Feedback Impact:

- I took the feedback that was given and gave a more concrete explanation of how the math works, within the presentation I skimmed over it pretty quick thus this comment makes sense.