Experiment No.10
PAGE RANK Algorithm
Date of Performance: 01/10/25
Date of Submission: 08/10/25
Name: Sumit Metkari
Div/Roll no.: 2 / 32

Aim: To implement Page Rank Algorithm

Objective:-Develop a program to implement page rank algorithm

Theory:-PageRank (PR) is an algorithm used by Google Search to rank web pages in their search engine results. PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. Page Rank Algorithm is designed to increase the effectiveness of search engines and improve their efficiency. It is a way of measuring the importance of website pages. Page rank is used to prioritize the pages returned from a traditional search engine using keyword searching. Page rank is calculated based on the number of pages that point to it. The value of the page rank is the probability will be between 0 and 1. A web page is a directed graph having two important components: nodes and connections. The pages are nodes and hyperlinks are the connections, the connection between two nodes.

Page rank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important website are likely to receive more links from other websites. The page rank value of individual node in a graph depends on the page rank value of all the nodes which connect to it and those nodes are cyclically connected to the nodes whose ranking we want; we use converging iterative method for assigning values to page rank. In short page rank is a vote, by all the other pages on the web, about how important a page is. A link to a page count as a vote of support. If there is no link, there is no support.

We assume that page A has pages B…….N which point to it.

Page rank of a page A is given as follows:

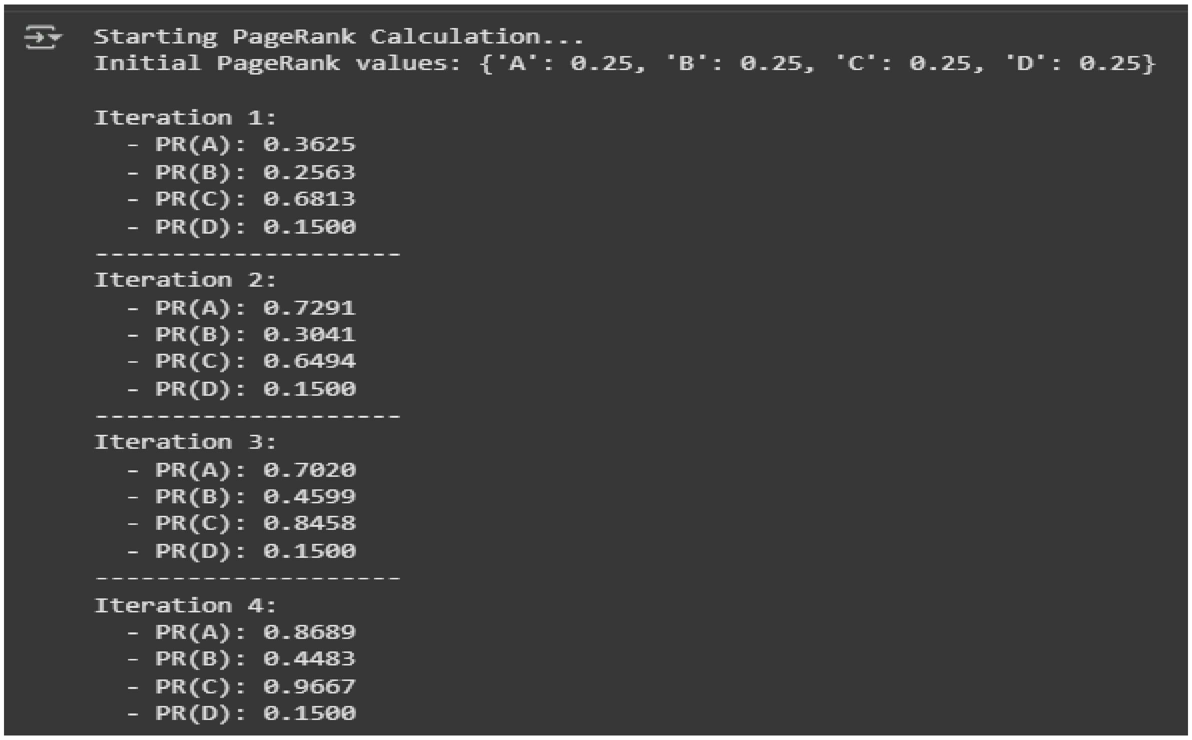
$PR(A)=(1-\beta) + \beta \, (\, (PR(B)/c_{out}(B) \,) + \, (PR(C \,)/c_{out}(C \,) \,) + \text{-----} + (PR(N)/c_{out}(N) \,) \,)$

Parameter β is a teleportation factor which can be set between 0 and 1.
$C_{out}(A)$ is defined as the number of links going out of page A.

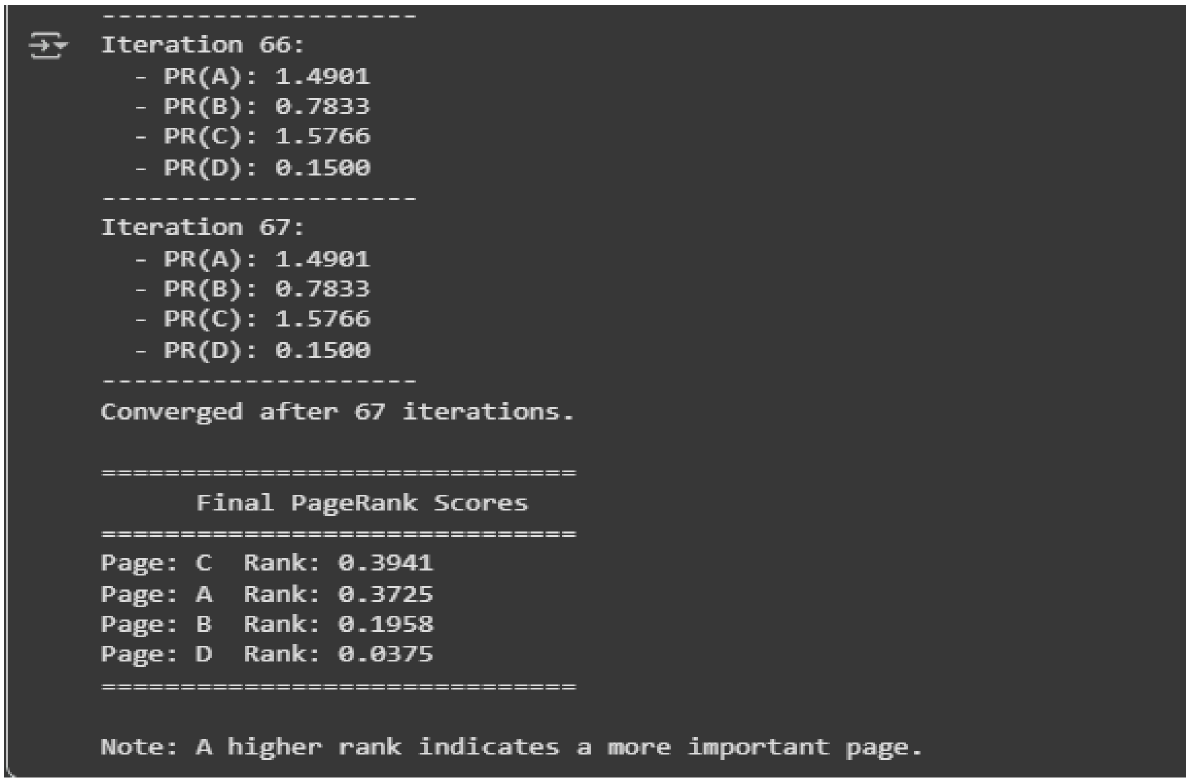
Code:

```
def calculate_page_rank(graph, damping_factor=0.85, max_iterations=100,
min_delta=0.00001):
    nodes = list(graph.keys())
    num_nodes = len(nodes)
    if num_nodes == 0:
        return {}
    page_rank = {node: 1.0 / num_nodes for node in nodes}
    out_links = {node: len(graph[node]) for node in nodes}

    for _ in range(max_iterations):
        previous_page_rank = page_rank.copy()
        for node in nodes:
            rank_sum = 0
            for linking_node in nodes:
                if node in graph[linking_node]:
                    rank_sum += previous_page_rank[linking_node] /
out_links[linking_node]
            new_rank = (1 - damping_factor) + damping_factor * rank_sum
            page_rank[node] = new_rank
        delta = sum(abs(page_rank[node] - previous_page_rank[node]) for node
in nodes)

        if delta < min_delta:

            break

    total_rank = sum(page_rank.values())

    if total_rank > 0:

        normalized_page_rank = {node: rank / total_rank for node, rank in
page_rank.items()}

    else:

        normalized_page_rank = page_rank

    return normalized_page_rank


if __name__ == "__main__":

    web_graph = {

        'A': ['B', 'C'],

        'B': ['C'],

        'C': ['A'],

        'D': ['C']

    }

    final_ranks = calculate_page_rank(web_graph, damping_factor=0.85,
max_iterations=100)

    sorted_ranks = sorted(final_ranks.items(), key=lambda item: item[1],
reverse=True)

    for page, rank in sorted_ranks:

        print(f"Page: {page}\t Rank: {rank:.4f}")
```

Output:



```
Starting PageRank Calculation...
Initial PageRank values: {'A': 0.25, 'B': 0.25, 'C': 0.25, 'D': 0.25}

Iteration 1:
  - PR(A): 0.3625
  - PR(B): 0.2563
  - PR(C): 0.6813
  - PR(D): 0.1500
--------------------
Iteration 2:
  - PR(A): 0.7291
  - PR(B): 0.3041
  - PR(C): 0.6494
  - PR(D): 0.1500
--------------------
Iteration 3:
  - PR(A): 0.7020
  - PR(B): 0.4599
  - PR(C): 0.8458
  - PR(D): 0.1500
--------------------
Iteration 4:
  - PR(A): 0.8689
  - PR(B): 0.4483
  - PR(C): 0.9667
  - PR(D): 0.1500
```



```
--------------------
Iteration 66:
  - PR(A): 1.4901
  - PR(B): 0.7833
  - PR(C): 1.5766
  - PR(D): 0.1500
--------------------
Iteration 67:
  - PR(A): 1.4901
  - PR(B): 0.7833
  - PR(C): 1.5766
  - PR(D): 0.1500
--------------------
Converged after 67 iterations.

==============================
      Final PageRank Scores
==============================
Page: C  Rank: 0.3941
Page: A  Rank: 0.3725
Page: B  Rank: 0.1958
Page: D  Rank: 0.0375
==============================

Note: A higher rank indicates a more important page.
```

Conclusion:
The implemented PageRank algorithm successfully calculated the relative importance of each page within the given web graph based on the number and quality of incoming links. The output ranks the pages in order of their significance, reflecting how likely a user is to visit each page. This ranking aligns with the core principle of PageRank—that pages with more inbound links from important pages receive higher scores. The algorithm's iterative approach ensures convergence to stable rank values, effectively simulating the behavior of a random web surfer.