Interactive Playground for PageRank Algorithm using Random Surfer and Power Iteration Method.

Team : 1) Rushidhar S (106124117)
2) Sanjay R (106124099)
3) Sai Nandu K (106124059)

Course Code : CSPE 01

Course Name : Combinatorics and Graph Theory

Department : Computer Science and Engineering

Section : CSE - A

Institute : NIT TRICHY

Aim of the project:

To design and implement an interactive simulation that demonstrates how the PageRank algorithm solves the importance of nodes in a directed graph. The project aims to help visualise the working of two approaches.

i. Random Surfer model
ii. Power Iteration method

Introduction:

PageRank is a graph-based ranking algorithm originally developed by Google to measure the importance of web pages. The importance of a page depends not only on the number of incoming links but also on the importance of those linking pages. This project demonstrates PageRank computation in two ways:

i. Random Surfer Model - a probabilistic approach.
ii. Power Iteration Method - a deterministic, matrix-based iterative method.

Both methods finally compute the same rank distribution over the graph nodes.

Algorithms Used:

1)Random Surfer Model (Monte Carlo Simulation):

Concept:

The Random Surfer Model explains PageRank using the idea of a person randomly browsing the web. At each step, the surfer either follows one of the links on the current page (with a damping factor, usually 0.85) or jumps to any random page in the network (with 0.15 probability). After running this process many times, the pages that get visited more frequently are considered to be more “important.”

- This method gives a good approximation of PageRank using the random walks.

Mathematical Expression:

If there are N nodes and damping factor d,

$$PR(i) = (1-d)/N + d \sum_{j \in In(i)} \frac{PR(j)}{L(j)}$$

Where,
N= total number of pages

d= damping factor (0.85, in general)

P_i= PageRank of page i

In(i) is the set of pages that link to i

$L(j)$ is the number of outgoing links from page j

Steps:

i. Represent the web as a directed graph.
ii. Start from a random node.
iii. At each step:
    - With probability d, move along a random outgoing link.
    - With probability 1 - d , teleport to a random node.
iv. Repeat for more iterations.
v. Normalize all the visit frequencies.

2) Power Iteration Method (Matrix Approach):

Concept:

This approach uses matrix operations to find the PageRank values more accurately. It treats the web as a network of probabilities where each link passes a fraction of a page's importance to the another. Starting with equal ranks, the algorithm repeatedly updates each page's rank based on incoming links until the values stop changing significantly. This represents the stable or steady-state PageRank distribution.

Mathematical Expression:

The PageRank vector PR converges as:

$$PR^{(t+1)} = d \cdot M \cdot PR^{(t)} + \frac{(1-d)}{N}$$

Where,
$PR^{(t)}$ is the PageRank vector at iteration t.

d is the damping factor (0.85).

M is the transition probability matrix.

N is the total number of nodes.

Steps:

i. Input the number of nodes and directed edges.
ii. Construct Transition Matrix,
For each node j:
- If it has outgoing links, distribute $\frac{1}{L(j)}$ among its neighbours.
- If there is no outgoing link (that means a dangling node), distribute $\frac{1}{L(j)}$ evenly among all nodes.

iii. Set $PR^{(0)} = \frac{1}{N}$,for all nodes.
iv. Compute $PR^{(t+1)} = d \cdot M \cdot PR^{(t)} + \frac{1-d}{N} \cdot e$.
v. If $| PR^{(t+1)} - PR^{(t)} | < \varepsilon$ for all nodes, stop.
vi. Final PageRank vector representing importance of each node.

Implementation:

- **Language Used:** C++
- **Input:** Number of nodes, number of directed edges, and list of edges.
- **Output:** PageRank values for each node.
- **Parameters:**
  - Damping factor (d): 0.85
  - Convergence threshold (ε): 1e-8
  - Random steps (for Monte Carlo): 1,000,000

Codes:

i)Random Surfer Model:

```cpp
#include<bits/stdc++.h>
using namespace std;

int main()
{
    int n, m;
    cin >> n >> m;
    vector<vector<int>> adj(n);
    for (int i = 0; i < m; i++) {
        int u, v; cin >> u >> v;
        adj[u].push_back(v);
    }
    double damping = 0.85;
    int steps = 1000000;
    vector<int> visits(n, 0);
    int current = rand() % n;
    for (int i = 0; i < steps; i++) {
        double r = (double)rand() / RAND_MAX;
        if (r < damping && !adj[current].empty())
            current = adj[current][rand() % adj[current].size()];
        else
            current = rand() % n;
        visits[current]++;
    }
    for (int i = 0; i < n; i++)
        cout << (double)visits[i]/steps << " ";

return 0;
}
```

ii)Power Iteration Method:

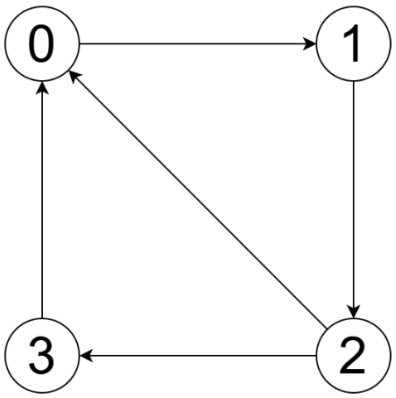
```cpp
#include<bits/stdc++.h>
using namespace std;

int main()
{
    int n, m;
    cin >> n >> m;
    vector<vector<int>> adj(n);
    for (int i = 0; i < m; i++) {
        int u, v; cin >> u >> v;
        adj[u].push_back(v);
    }
    double d = 0.85, eps = 1e-8;
    vector<double> pr(n, 1.0/n), new_pr(n);
    bool converged = false;
    while (!converged) {
        fill(new_pr.begin(), new_pr.end(), (1-d)/n);
        for (int j = 0; j < n; j++) {
            if (adj[j].empty())
                for (int i = 0; i < n; i++) new_pr[i] += d * pr[j] / n;
            else
                for (int dest : adj[j]) new_pr[dest] += d * pr[j] / adj[j].size();
        }
        converged = true;
        for (int i = 0; i < n; i++)
            if (fabs(new_pr[i] - pr[i]) > eps) converged = false;
        pr = new_pr;
    }
    for (double x : pr) cout << x << " ";
return 0;
}
```

Sample Inputs:

u, v = 4, 5

links:

i) 0→1
ii) 1→2
iii) 2→0
iv) 2→3
v) 3→0



Sample Outputs:

Node	Random Surfer	Power Iteration
0	0.371852	0.372628
1	0.200537	0.200365
2	0.245386	0.245009
3	0.182225	0.182000

GitHub Repository:

https://github.com/sai-nandu/CGT_PROJECT

Observation:

- Both algorithms produce nearly identical values (differences < 0.001).
- Random Surfer gives an approximation.(based on random walks).
- Power Iteration gives the exact steady-state values.
- The small difference occurs because the Random Surfer method is approximate and based on randomness, while the Power Iteration method gives precise, mathematically converged results.
- The ranking order of importance is the same for both:

  Node 0 > Node 2 > Node 1 > Node 3

Conclusion

The project successfully demonstrates the working of the PageRank algorithm using two distinct computational approaches. Both methods converge to the same steady-state probability distribution, illustrating how node importance can be determined purely from the link structure. This experiment deepened understanding of graph algorithms, probabilistic models, and iterative convergence methods.