

Sia Sharma

Collaborators: Om Italiya, Nikita Salkar

Code Output:

```
● siasharma@crc-dot1x-nat-10-239-81-24 hw10main % cargo run
  Compiling hw10main v0.1.0 (/Users/siasharma/Desktop/rusthw/hw10/hw10main)
  Finished dev [unoptimized + debuginfo] target(s) in 0.48s
  Running `target/debug/hw10main`
vertex 603: approximate PageRank 0.0020
vertex 596: approximate PageRank 0.0019
vertex 886: approximate PageRank 0.0014
vertex 177: approximate PageRank 0.0014
vertex 650: approximate PageRank 0.0014
○ siasharma@crc-dot1x-nat-10-239-81-24 hw10main %
```

When it is run again, this is the output:

```
● siasharma@crc-dot1x-nat-10-239-81-24 hw10main % cargo run
  Finished dev [unoptimized + debuginfo] target(s) in 0.11s
  Running `target/debug/hw10main`
vertex 596: approximate PageRank 0.0021
vertex 603: approximate PageRank 0.0018
vertex 198: approximate PageRank 0.0015
vertex 744: approximate PageRank 0.0014
vertex 354: approximate PageRank 0.0014
○ siasharma@crc-dot1x-nat-10-239-81-24 hw10main %
```

Analysis on my program:

- My program calculates PageRank values for vertices in a graph. The program begins by reading graph data from a file, where the first line specifies the number of vertices, and subsequent lines represent edges between vertices. Using a random walk algorithm, I simulated transitions between vertices to approximate PageRank values. In the `random_moves` function, I conducted random walks on the graph, with each walk starting from a vertex and moving to neighboring vertices based on the probabilities given in the HW set. During each walk, I counted the number of times each vertex is visited, storing these counts in a vector. After simulating a large number of random walks, I calculated the PageRank values for each vertex based on the visit counts.
- To identify the top vertices with the highest PageRank values, I sorted the PageRank values in descending order and associated each value with its corresponding vertex. I then printed the top vertices along with their respective PageRank values.
- I also implemented error handling when reading the graph file, which printed error messages if necessary, for which I used help from <https://chat.openai.com/>. My test function makes sure the graph reading process is correct.
- Throughout the code, I used the `rand` crate for generating random numbers and making random choices during the random walk simulation.