Eric Krivoy Sekler

COP3530

Project 2: Simplified Page Rank Documentation

1. Data Structure Used

I used an Adjacency List to represent the graph. This choice was made because an adjacency list is memory efficient, especially for sparse graphs, and allows easy iteration over all neighbors of a node. This efficiency is crucial for the PageRank algorithm, which involves frequent access to the neighbors of each node.

2. Computational Complexity

1. addEdge Method:
    - Time Complexity: O(1)
    - This is because adding an edge involves a constant-time operation to insert the edge into the adjacency list.
2. initializeRanks Method:
    - Time Complexity: O(V)
    - This involves initializing the rank for each of the V nodes, which is a linear operation.
3. computePageRank Method:
    - Time Complexity: O(IV + IE) = O(I(V + E))
    - Here, I is the number of iterations, V is the number of nodes, and E is the number of edges. For each iteration, we update the rank of each node based on its neighbors, which involves iterating over all edges.
4. PageRank Method:
    - Time Complexity: O(V + I(V + E))
    - This method first initializes ranks in O(V) and then calls computePageRank, resulting in the above complexity.

3. Main Method Complexity

- Time Complexity: O(V + E + I(V + E))
- In the main method, we read input edges (O(E)), initialize the graph (O(V)), and compute the PageRank (O(I(V + E))).

4. Learning and Reflection

From this assignment, I learned how to implement and optimize the PageRank algorithm using an adjacency list. If I had to start over, I would focus more on optimizing the computation of PageRank to see if I could make a more efficient code.