Edward Lang
04/06/2023

# Project 2

## Data Structures Used

The data structure I used to implement the graph is a map that stores a string as the key and a vector pair (string and integer) as the values of the map. The keys represent the URLs (vertices) in the map. The vector, which is values for the map, stores the string of a connected vertex and the index of the URL as an integer. I also used another map to store the page rank and contributions of the URL. The key is the URLs, and the first double is the page rank and the second doble is the contribution. The reason for using a map was for quicker search and insertion of URLs. Since we had to implement an adjacency list, maps were a great option for a sparse graph. Storing the page rank and contribution in a map allowed for efficient updating, calculating, and searching.

## Time Complexities

AddVertex: O(V)

AddVertex has a worst-case time complexity of O(log(V)), where V represents the number of vertices in the graph. Map's find function completes the search as a balanced binary search tree. This has a time complexity of O(log(V)).

InsertEdge: O(log(V))

InsertEdge also has a worst-case time complexity of O(V). InsertEdge calls AddVertex twice, so we get O(log(V)+log(V)) = O(2log(V)) = O(log(V)).

Outdegree: O(log(V))

Map's find function completes the search as a balanced binary search tree. This has a time complexity of O(log(V)).

CalculateContribution: O(V+E)

CalculateContribution iterates through all vertices and their connected edges. Outer loop iterates through all vertices and the inner loop iterates through that vertex's connected edges. This means that the total number of iterations through the inner loop is O(E), whilst the outer loop is O(V). O(V) + O(E) = O(V+E).

UpdatePageRank: O(V)

Iterates through all vertices, so O(V).

PrintPageRank: O(V)

Iterates through all vertices, so O(V).

PageRank: O(n * (V + E))

This one starts by iterating over the graph, so O(V). Then, we loop from 1 to the number of power iterations; n. This is O(n). Calculate Contribution has a worst-case time complexity of O(V+E) and UpdatePageRank has a worst-case of O(V). Finally, PrintPageRank has a time complexity of O(V). Adding this all together, we get: O(V) + O(n) * (O(V+E) + O(V)) + O(V) = O(V) + O(n) * (O(2V+E)) = O(n*(V+E)).

Main Method: O(E*log(V) + n*(V+E))

To explain, we start off by reading in 2 inputs: O(1).

Next, loop from 0 through no_of_lines: O(no_of_lines) since no_of_lines = number of edges, we get O(E). InsertEdge called in loop, InsertEdge is O(log(V)). So total for this loop is O(E*log(V)).

PageRank is called: O(n*(V+E)).

Adding all of these together, we get:
O(1)+O(E*log(V)) + O(n*(V+E)) =
O(1+E*log(V) + n*(V+E)) =
O(E*log(V) + n*(V+E)).

## What I learned

I learned that I shouldn't just add in the default or typical methods for a problem. For example, my first version of the project used all the methods I used to solve the Stepiks and Programming Quiz 8. This was a gigantic mistake, as it cost me a lot of time to effort contemplating how to implement these methods to the project. Instead, I should have focused on what I needed first. Then, as I built my project, I should add things as needed. This would have saved me a lot of time. The "junk" code I added had value, but not for this project. In the future, I will spend more time building a plan; figuring out what I need and adding them as necessary. Next time, I will also consider using an unordered map, as the time complexities for insertion into a hash table is O(1). This would greatly increase the speed of my program. Another thing I would have implemented would be a separate class for PageRanks. I unnecessarily added it to a map, when a struct would have sufficed. This would improve the modularity and readability of my code. It was tough to remember what keys are stored in what location when I have two maps with vectors. It felt needlessly complex. So to recap, I will plan better and think about the data structures I want to use and compare them to each other to find the best structure for an assignment.