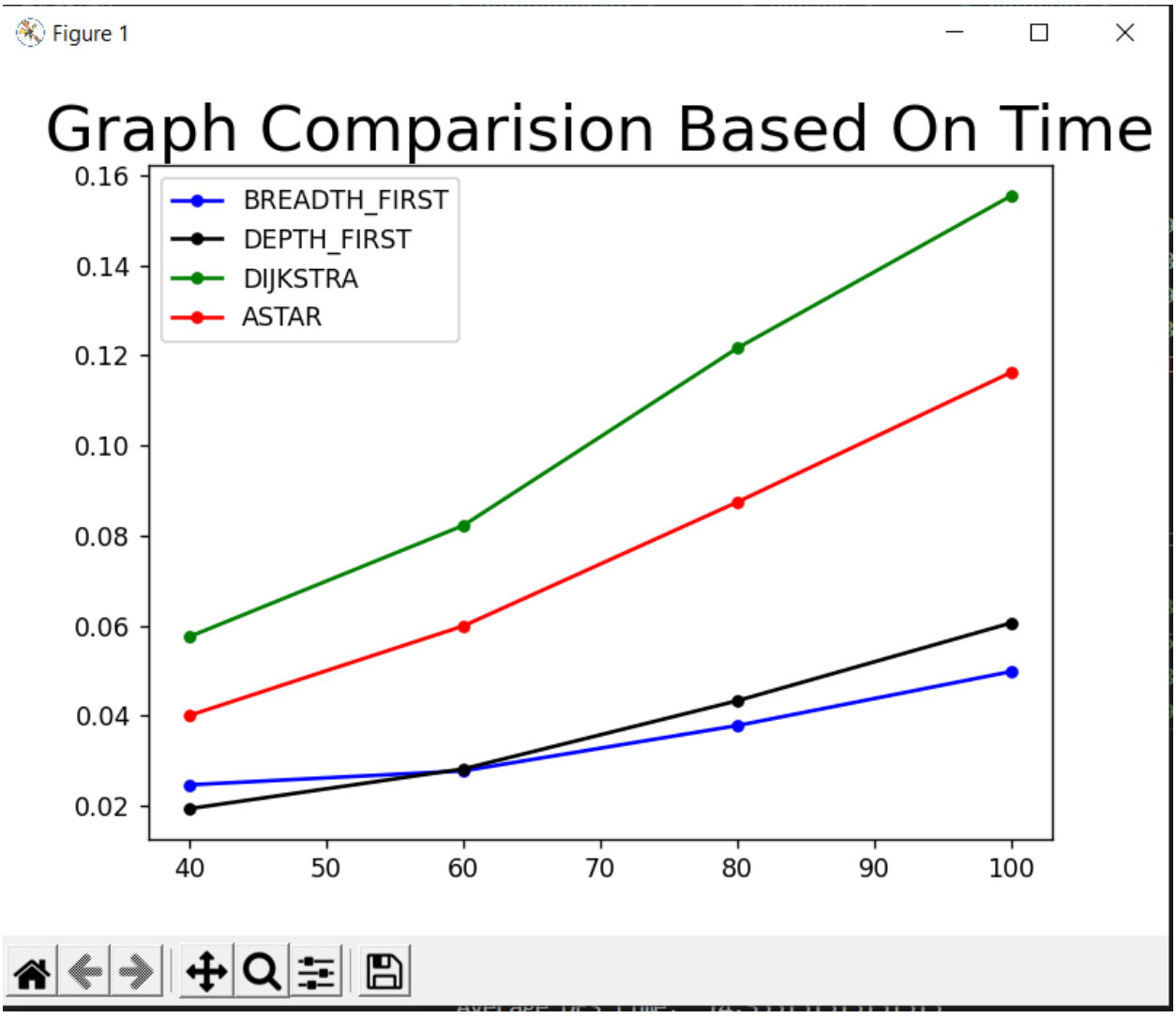
Contents

Procedure Taken

As required by the bonus question, four additional graph files were created by the name graph1x, graph2x, graph3x and graph4x along with their position files position1x, position2x, position3x and position4x respectively. This graphs where then inserted into our graph implementation code and the test algorithms where tested again for each graph files. The graphs, as indicated in their names, where filled with nodes that are 1x, 2x, 3x and 4x the original numbers of nodes and the effect that this increase in nodes and edges bring was tested and recorded. The general result of these graphs compared to the initial result of the original graph were found to be different. The average time and number of hops where found to be increased and logically this was correct. Increase in number of nodes and connections will both increase the time to traverse and the number of nodes passed during traversal. The average length, however, was found to be less than the initial graph. Based on the weights that the new edges have, this made sense since most of the new edges have lower costs and new edges create the opportunity for lower paths between previous nodes resulting in a lower average length in our case. This may not have been true if we had used longer edges few connections. Other than the initial graph, the graphs had showed difference in average time, length and hops amongst each other. This differences are shown below.

Average Time

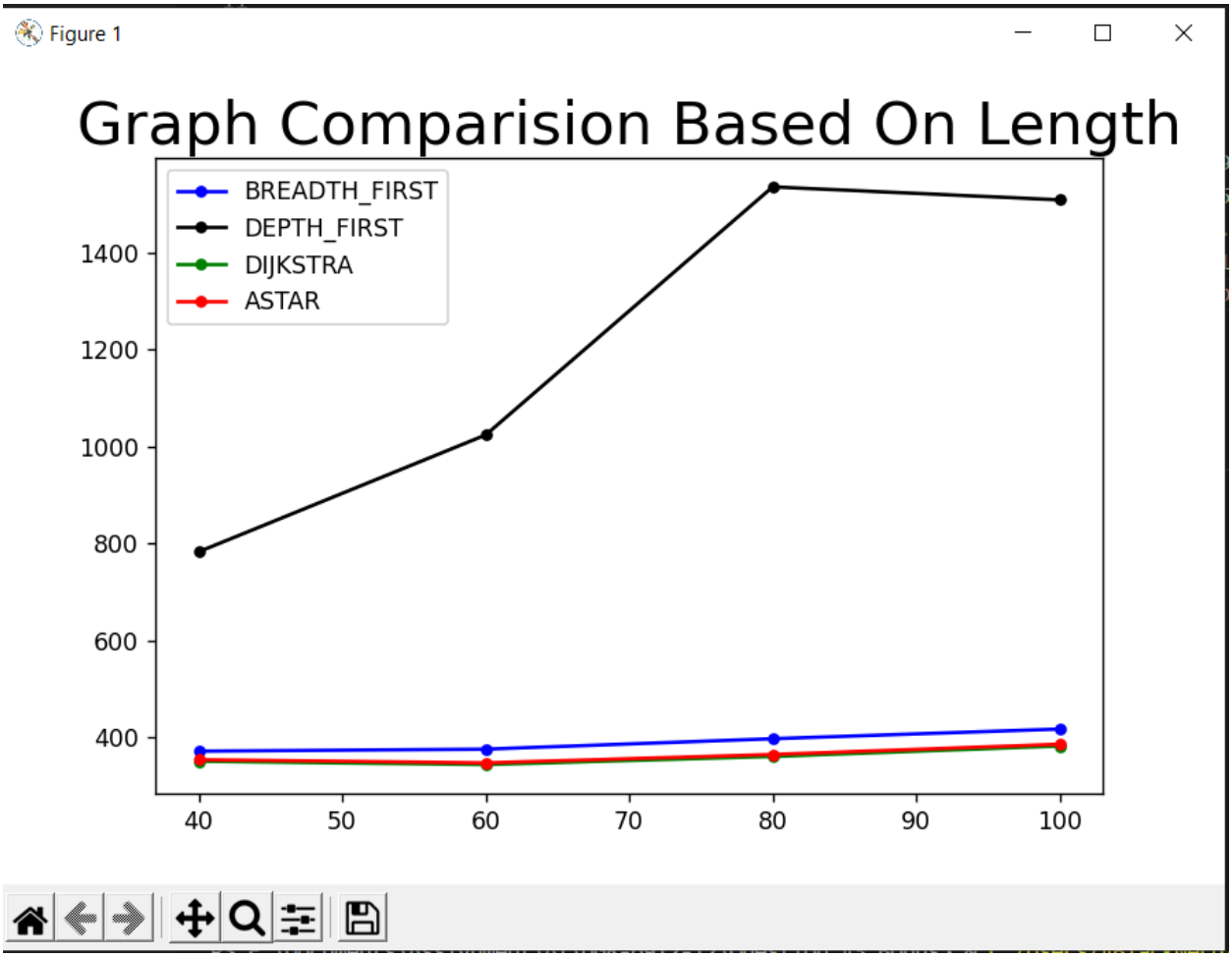


Compared to the initial graph average BFS and DFS time, the new graphs' average BFS and DFS time showed a big difference. In the initial graph the values where found to be very similar although DFS was a little faster than BFS but now it started mixing up.

In graph1x, the average DFS time was found to be smaller than the BFS just like our initial graph, but then in graph2x the average BFS and DFS time were found to be identical. In graph3x, the BFS was found to be faster than DFS which continued to graph4x. this result was due to our increase in number of nodes per to number of edges compared to the initial value. This higher percentage addition of node resulted in a more branched graph which made the BFS algorithm favorable.

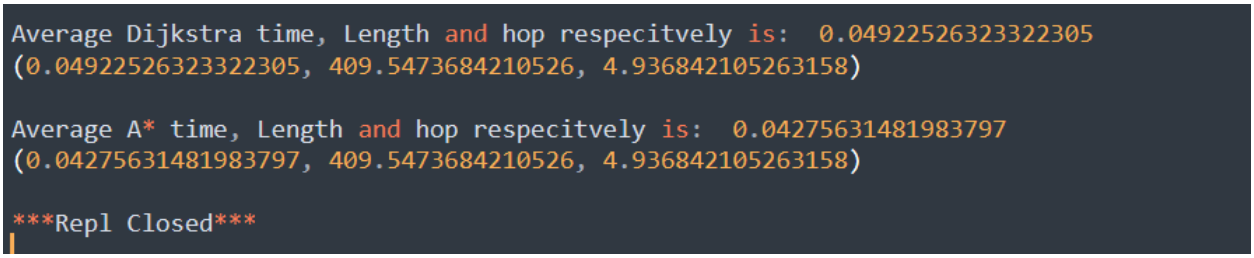
The dijkstra and A* algorithm maintained their property from initial graph just with a higher difference. The average time take to find a path using dijkstra was found to be growing further and further away from the A* as the number of nodes increase. The reason for this remains the same as the reason described in the initial graph. The A* algorithm uses the heuristic value to quickly find the shortest path significantly reducing the time take by the dijkstra's algorithm of finding nearest nodes in the right order without considering the target node.

Average Length



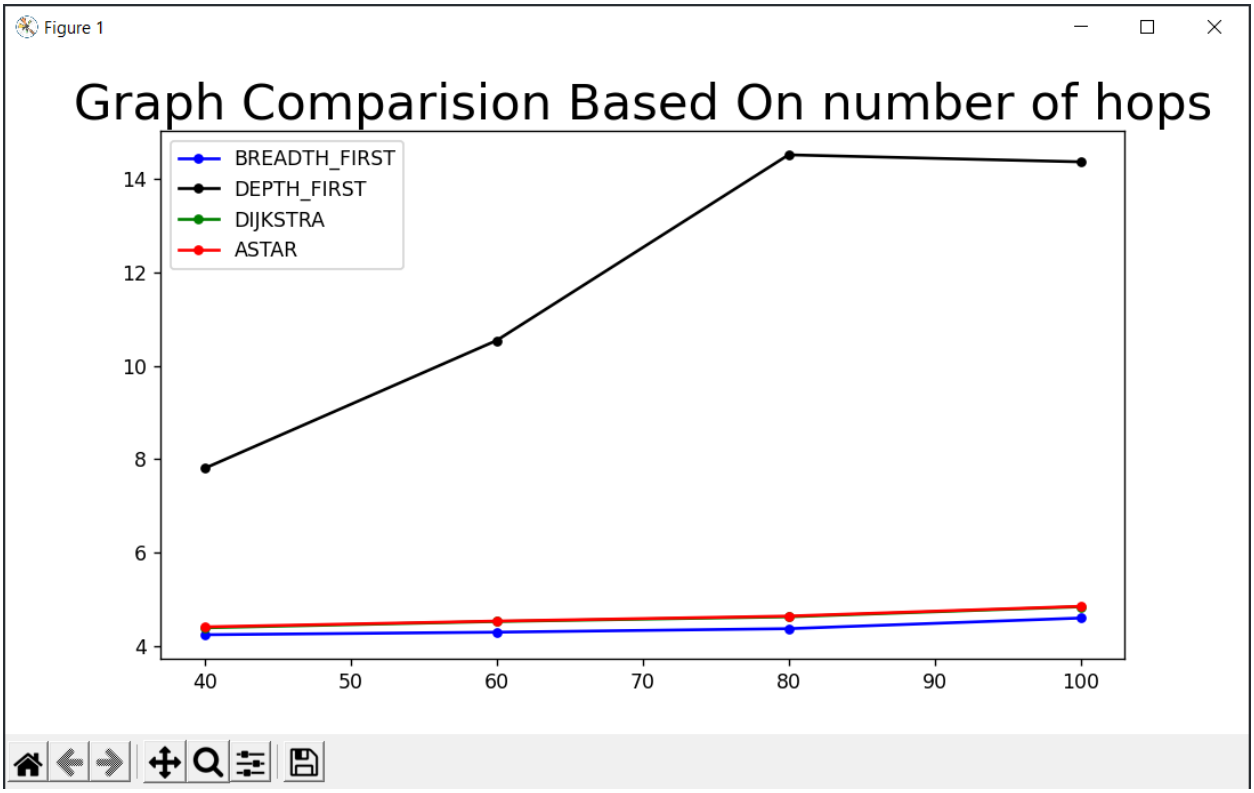
In comparison with the initial graph, the overall property of the length was found to be unchanged. The Dijkstra's and A* algorithms where still found to provide the shortest average length and the BFS was still found to have a shorter average length than the DFS. The DFS was found to be increasing exponentially as the number of edges were increased and the branches on the same layer grew more and more. **graph4x was intentionally made to add just one node and one connection with an existing node from graph3x and making it more layered with fewer branches**. This was expected to make the average DFS length smaller as it would be easier and shorter to find this additional nodes using DFS. This was approved by our implementation as the average DFS distance decreased in graph4x as compared to graph3x. it is important to note that the DFS might have been faster for graph1x, 2x, and 3x if the conditions have been reversed.

In the initial graph, the heuristic value of the A* algorithm was seen to cause a difference of 0.2 km from the dijkstra's value. It was concluded if a better heuristic was used, the difference will decrease and the dijkstra's and A*'s average length will be very very close. This was tested with a new heuristic value and was found to be true.



As it can be seen from the screenshot the average lengths of dijkstra's and A* algorithms were found to be identical for the new heuristic value. But this heuristic value did not hold fully accurate for the new graphs with added nodes. It was found to have more and more average length as the number of nodes increased from graph1x to graph4x. but, this difference was so marginal that it became unnoticeable when the graph was plotted. in the end, it was understood that this heuristic value was better but it still was not perfect.

Average number of hops



The number of hopes was also found to preserve its property from the initial value, The BFS algorithm was found to have the smallest number of hop compared to the other algorithms. The Dijkstra and A* algorithms were also found to have nearly identical number of hops across all graphs. The DFS had the largest number of hops compared to the others. The less edged addition in graph4x also affected the average number of hops made by the DFS algorithm. As the layers increased and the graph started becoming more layered than branched the DFS algorithm started finding most of the nodes easily as most of them became available on different layers rather than different branches. This intern decreased the average number of hops just like it did with the average number of length.

The Dijkstra and A* algorithms average number of hops was found to show a bit odd behavior. The number of hops where very close to each other but they had marginal difference. The fact that made it unpredictable was that it was not consistently increasing as the number of nodes increased from graph1x to graph4x just like it did with average length. It was actually found to be inconsistent. For example :- the difference in average number of hops between dijkstra and A* in graph1x where found to be 0.02. it decreased to 0.01 in graph2x. it then increased to 0.02 in graph3x just to decrease to 0.01 again in graph4x. it was understood that this inconsistency was being created due to the alternate paths created by adding the nodes. The node addition in

graph2x and graph4x created an alternative paths that reduced the average number of hops of the A* algorithms in the previous ones compared to the current ones. This resulted in an inconsistent difference between the two algorithms. All in all, it was observed that all of these marginal and inconsistent differences in the A* and Dijkstras algorithms were mainly being caused due to the quality of the heuristic value.

Conclusion

- BFS will always have the smallest number of hops no matter what the numbers of nodes and edges are.
- The difference in time between DFS and BFS highly depend on the structure of the graph.
- Dijkstra's algorithm will always find the shortest path but it takes much more time to find the paths specially as the size of the graph increases.

- The A* algorithm is the best and the fastest way to find the shortest algorithm between two nodes but it highly depends on the quality of the heuristic value.