**LAB 2 REPORT**

**ADJACENCY LISTS:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adjacency List | DFS Iterative | DFS Recursive | BFS Iterative | BFS Recursive | Dijkstra | A\* |
| Nodes in Path | 7.2456 | 7.2456 | 6.453 | 6.453 | 5.3245 | n/a |
| Nodes Explored | 12.243 | 12.243 | 17.3421 | 17.3421 | 12.33392 | n/a |
| Execution Time | 0.000081904 | 0.000145632 | 0.000095463 | 0.00023722 | 0.00009732 | n/a |
| Distance | 15.2523 | 15.2523 | 12.364 | 12.364 | 11.264 | n/a |
| Cost | 9.9345 | 9.9345 | 8.453291 | 8.453291 | 6.943 | n/a |

**ADJACENCY MATRIX:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adjacency Matrix | DFS Iterative | DFS Recursive | BFS Iterative | BFS Recursive | Dijkstra | A\* |
| Nodes in Path | 8.3612 | 8.3612 | 7.221 | 7.221 | 6.1145 | n/a |
| Nodes Explored | 13.2399 | 13.2399 | 17.998 | 17.998 | 16.4712 | n/a |
| Execution Time | 0.000021904 | 0.00099632 | 0.00001472 | 0.00024554 | 0.0000797 | n/a |
| Distance | 17.45365 | 17.45365 | 11.1342 | 11.1342 | 9.9976 | n/a |
| Cost | 10.1245 | 10.1245 | 7.9876 | 7.9876 | 7.621 | n/a |

**EVALUATION:**

It is clear that after seeing this data results, the best my program did functionally was with the Dijkstra algorithm. This is because it is and algorithm that finds the shortest path rather than the quickest. Although sometimes it could lose to the DFS or BFS, overall after normalizing the data, it was clear that it was the most accurate, meaning lowest cost. But with this cost came the price of time. Dijkstra was also the slowest algorithm because it consequently had to explore more nodes in order to be more accurate. Another observation that can be taken from this data, is how the recursive function call performed slower in both DFS and BFS algorithms. This is because when doing recursive function calls, it takes more time rather than just looping through a simple while loop. If I was able to get the A\* algorithm to work though, it was clear by just making pseudocode diagrams, that it had the potential to be the best. This is because it uses a heuristic technique of trying to predict future cost and basing its current moves off that. So basically, it’s like a well-informed version of Dijkstra. Also, when looking at my data when it came down to the speed and efficiency of adjacency lists vs adjacency matrixes, the list came in better in most categories. I think this is due to the efficiency on which their complexity is measured. Because matrixes are based on NxN dimensions, they are O(n^2), but adjacency lists are only dependent on edges, so even in their best, it can be O (1). This goes along way when you have to continually sort and parse through the data structures.