Project 2C

“min-hop path finding problem”

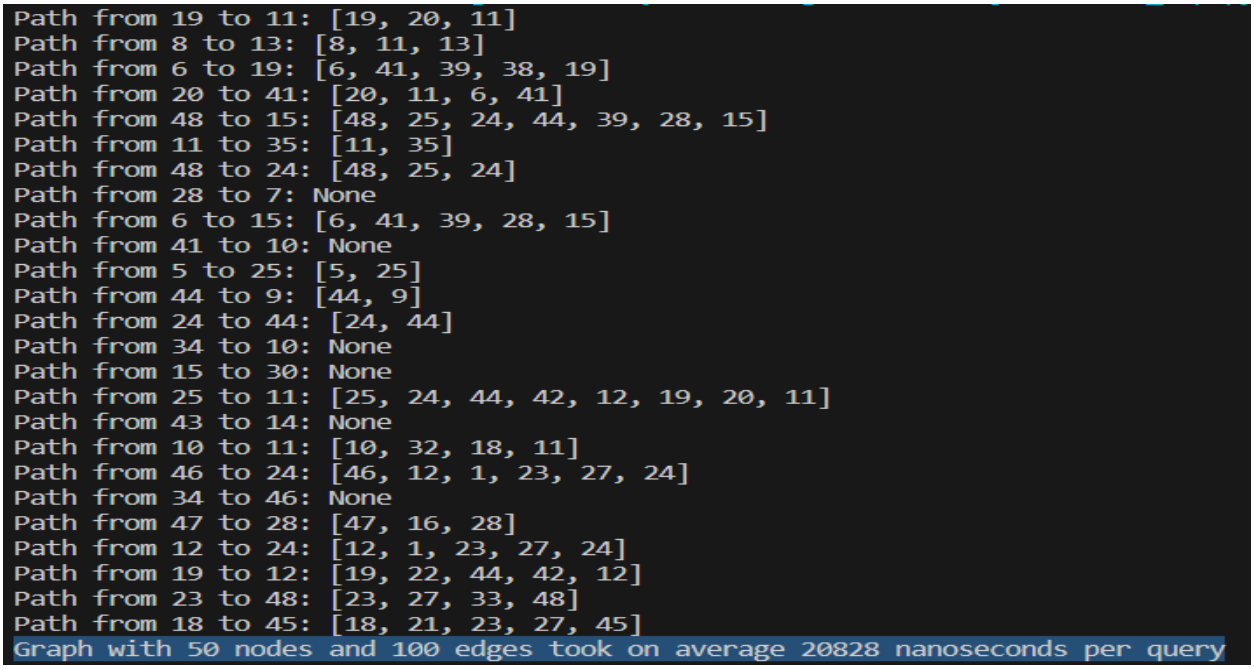
Mehrdad Yadollahi

Contents

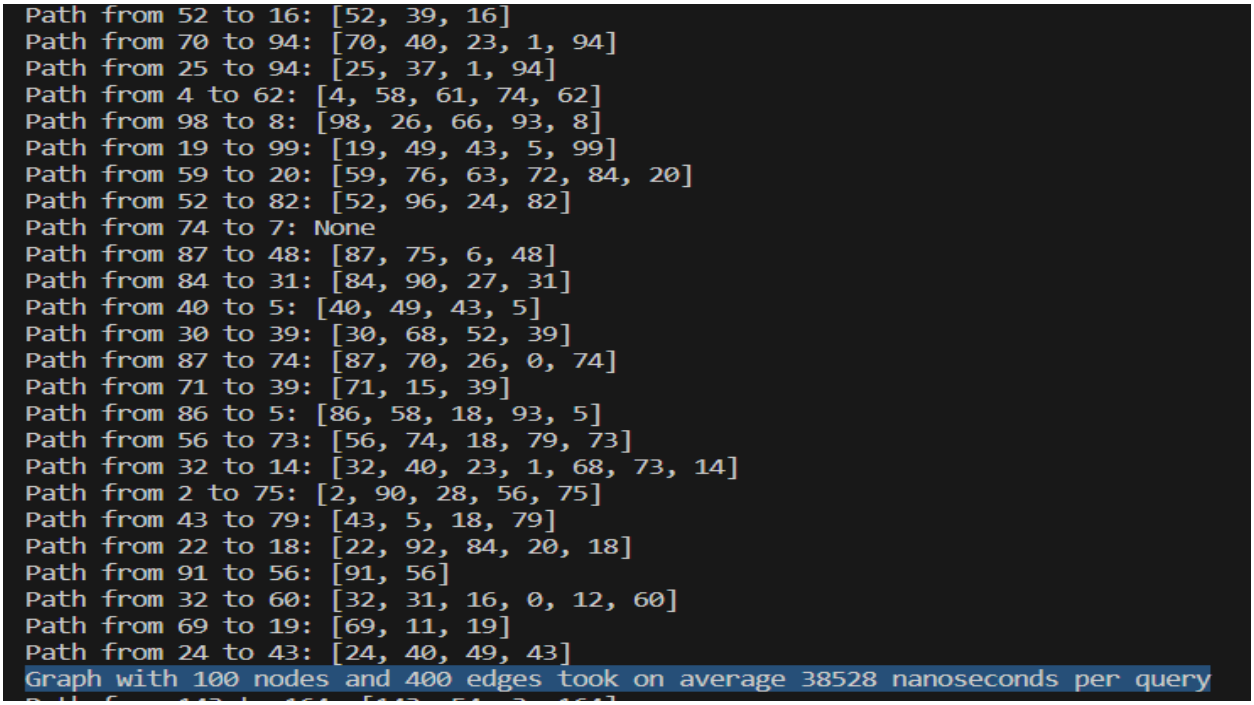
Output for all test data

You can find the requested output based on 4 types of input format.

G1: 50 nodes and 100 edges (pairs=25)



G2: 100 nodes and 400 edges (pairs=25)



G3: 200 nodes and 1600 edges (pairs=25)

```
Path from 143 to 164: [143, 54, 3, 164]
Path from 122 to 188: [122, 199, 183, 188]
Path from 88 to 162: [88, 1, 84, 162]
Path from 88 to 129: [88, 1, 172, 129]
Path from 19 to 107: [19, 63, 22, 107]
Path from 119 to 10: [119, 33, 10]
Path from 77 to 122: [77, 24, 173, 113, 122]
Path from 125 to 144: [125, 53, 5, 144]
Path from 29 to 56: [29, 130, 56]
Path from 173 to 166: [173, 104, 191, 166]
Path from 129 to 159: [129, 10, 159]
Path from 38 to 178: [38, 142, 178]
Path from 8 to 135: [8, 45, 135]
Path from 80 to 39: [80, 6, 7, 39]
Path from 73 to 104: [73, 3, 143, 104]
Path from 3 to 47: [3, 143, 104, 47]
Path from 195 to 94: [195, 109, 94]
Path from 81 to 159: [81, 131, 141, 159]
Path from 95 to 82: [95, 69, 158, 82]
Path from 31 to 120: [31, 108, 185, 120]
Path from 156 to 84: [156, 86, 84]
Path from 115 to 129: [115, 6, 197, 129]
Path from 79 to 92: [79, 38, 92]
Path from 137 to 142: [137, 67, 38, 142]
Path from 120 to 157: [120, 157]
Graph with 200 nodes and 1600 edges took on average 70176 nanoseconds per query
```

G4: 400 nodes and 3200 edges (pairs=25)

```
Path from 49 to 104: [49, 33, 383, 104]
Path from 275 to 322: [275, 373, 31, 322]
Path from 243 to 17: [243, 378, 379, 17]
Path from 343 to 42: [343, 92, 242, 42]
Path from 270 to 151: [270, 273, 326, 55, 151]
Path from 89 to 46: [89, 35, 43, 46]
Path from 82 to 54: [82, 63, 108, 54]
Path from 299 to 178: [299, 144, 138, 217, 178]
Path from 230 to 367: [230, 219, 313, 367]
Path from 163 to 392: [163, 203, 189, 392]
Path from 332 to 312: [332, 105, 396, 227, 312]
Path from 323 to 257: [323, 257]
Path from 204 to 287: [204, 287]
Path from 235 to 71: [235, 276, 313, 71]
Path from 169 to 107: [169, 235, 18, 145, 107]
Path from 75 to 207: [75, 10, 201, 207]
Path from 304 to 187: [304, 298, 284, 187]
Path from 367 to 312: [367, 175, 227, 312]
Path from 86 to 323: [86, 295, 323]
Path from 82 to 361: [82, 188, 7, 361]
Path from 188 to 105: [188, 130, 218, 105]
Path from 225 to 54: [225, 287, 318, 54]
Path from 104 to 313: [104, 159, 12, 313]
Path from 355 to 356: [355, 84, 356]
Path from 297 to 29: [297, 381, 306, 210, 29]
Graph with 400 nodes and 3200 edges took on average 149156 nanoseconds per query
```

Analysis of expected and theoretical performance

Time Complexity of BFS: Theoretically, the time complexity of BFS is $O(V+E)$. This is because each vertex and each edge will be explored in the worst case.

Expected Performance in the Min-Hop Problem

- The performance will vary based on the size and density (number of edges relative to the number of nodes) of the graph.
- The existence of a path from the source to the destination node also affects performance. If no path exists, the algorithm might end up exploring the entire graph, leading to longer execution times.
- Since the test graphs are randomly generated, the average performance might vary.

Comparison to Theoretical Performance

- **Best-Case Scenario**: In the best case, where the destination node is one of the immediate neighbors of the source node, the algorithm would have a linear performance relative to the number of neighbors, effectively $O(k)$ where k is the degree of the source node.
- **Average Case**: The average-case performance can be hard to predict due to the random nature of the graph generation and the distribution of nodes.
- **Worst-Case Scenario**: The worst-case scenario occurs related to size of graph or when the destination node is one of the last nodes to be explored, leading to a full exploration of all vertices and edges.

Analysis and Discussion of the results

When the size of the graph increases, the average time will increase. In fact, the time complexity of the graph depends on the input density of directed graphs (G1, G2, G3, G4). According to following table:

	Average Time1 (nanoseconds)	**Average Time2 (nanoseconds)**	**Average Time3 (nanoseconds)**
G1	**9392**	**8088**	**16229**
G2	**23116**	**25492**	**44668**
G3	**113500**	**64184**	**146784**
G4	**176308**	**145940**	**261235**

A graph with more nodes and edges typically requires more time, but the increase should align with the O (V + E) complexity.

The high-level analysis of the BFS implementation for the min-hop path finding problem should provide a clear understanding of the algorithm's performance under varying conditions and its practical applications. This analysis not only demonstrates the effectiveness of BFS in certain scenarios but also highlights areas for further research and optimization.