GTU Department of Computer Engineering
CSE 222/505 - Spring 2023
Homework #08 Report

Selim Aynigül
200104004004

Comparison of Algorithms

Djikstra ***Time Complexity: O(V + E.logV)***

While loop: The while loop iterates until the queue is empty. In the worst case scenario, all nodes will be visited, so the loop will execute a maximum of V times, where V is the number of nodes.

Queue operations: Inside the for loop, the distance array is updated, and nodes are added to the priority queue. The offer operation for the priority queue has a time complexity of O(log V) in the worst case, where V is the number of nodes.

Combining these factors, the overall time complexity of the algorithm can be approximated as O(V + E log V), where V is the number of nodes and E is the number of edges in the graph.

BFS ***Time Complexity: O(V + E)***

While loop: The while loop iterates until the queue is empty. In the worst case scenario, all nodes in the graph will be visited. Therefore, the while loop will execute a maximum of V times, where V is the number of nodes in the graph.

For loop: Inside the while loop, there is a for loop that iterates over the neighbors of the current node. The number of neighbors that a node has is typically determined by the graph's structure and can vary. Let's denote the average number of neighbors as E.

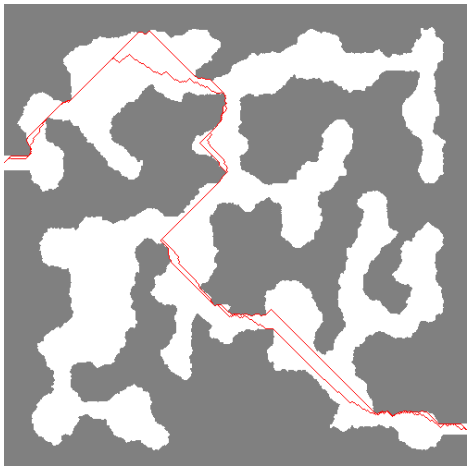
In the worst case, we may need to explore all vertices and traverse all edges. Therefore, the time complexity of BFS is O(V + E), where V represents the number of vertices and E represents the number of edges in the graph

Comparison

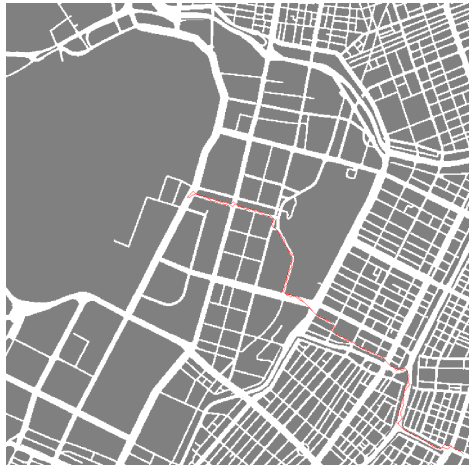
BFS findPath() algorithm works faster in general but as the size of the input file grows, djikstra algorithm becomes faster. In our inputs while bfs works faster for 500x500 files, djikstra gave better results with 1000x1000 files.

Output PNG Examples

Map03.txt



tokyo.txt



There will be 3 files for all the maps as outputs. 1 png image that shows the shortest path on the map for bfs and djikstra, and also 2 txt files that show coordinates of the paths for bfs and djikstra.

Running Command and Results

Compiling commands:

cd homwork8
javac *.java

Running commands:

java Main.java

```
C:\Users\Administrator\Desktop\aa>java Main.java
Map is Map08.txt with X_SIZE 500 and Y_SIZE 500
Map is Map04.txt with X_SIZE 500 and Y_SIZE 500
Map is map01.txt with X_SIZE 500 and Y_SIZE 500
Map is Map09.txt with X_SIZE 500 and Y_SIZE 500
Map is Map10.txt with X_SIZE 500 and Y_SIZE 500
Map is Triumph.txt with X_SIZE 1000 and Y_SIZE 1000
Map is Vatican.txt with X_SIZE 1000 and Y_SIZE 1000
Map is Map06.txt with X_SIZE 500 and Y_SIZE 500
Map is Map05.txt with X_SIZE 500 and Y_SIZE 500
Map is Tokyo.txt with X_SIZE 1000 and Y_SIZE 1000
Map is Map02.txt with X_SIZE 500 and Y_SIZE 500
Map is Map07.txt with X_SIZE 500 and Y_SIZE 500
Map is Pisa.txt with X_SIZE 1000 and Y_SIZE 1000
Map is Map03.txt with X_SIZE 500 and Y_SIZE 500
Map04.txt Dijkstra Path: 673
Map04.txt BFS Path: 673
Map07.txt Dijkstra Path: 709
Map07.txt BFS Path: 709
Map08.txt Dijkstra Path: 640
Map08.txt BFS Path: 640
Map10.txt Dijkstra Path: 478
Map10.txt BFS Path: 478
Map05.txt Dijkstra Path: 599
Map05.txt BFS Path: 599
Map06.txt Dijkstra Path: 506
Map06.txt BFS Path: 506
Map03.txt Dijkstra Path: 760
Map03.txt BFS Path: 760
Map02.txt Dijkstra Path: 666
Map02.txt BFS Path: 666
map01.txt Dijkstra Path: 991
map01.txt BFS Path: 991
Map09.txt Dijkstra Path: 957
Map09.txt BFS Path: 957
Tokyo.txt Dijkstra Path: 890
Tokyo.txt BFS Path: 890
Triumph.txt Dijkstra Path: 1059
Triumph.txt BFS Path: 1059
Vatican.txt Dijkstra Path: 1412
Vatican.txt BFS Path: 1412
Pisa.txt Dijkstra Path: 1642
Pisa.txt BFS Path: 1642
```