GEBZE TECHNICAL UNIVERSITY
CSE222 HW8 REPORT

Name: Emre Oytun
Student Number: 200104004099

1) Complexity Analysis of Graph Algorithms:

1.1) Breadth First Search Algorithm:

The algorithm uses a queue to traverse the graph level by level, this queue keeps the nodes that will be visited in order. It also keeps the visited elements in an hash set and keeps the parents of each nodes in a hash map.

Firstly, it initializes the queue, hash set and hash map in O(1) time. It puts the start node into the queue. Secondly, it enters a while loop that continues as long as the queue is not empty or the end node is not found. Inside the loop, it checks if the current node is the end node. If the end node is found, then it exits the loop otherwise it traverse through the all edges of the current node and puts the unvisited nodes into the queue and updates their parents.

In the worst-case, the algorithm traverses all the vertexes and finds the other unvisited vertexes by traversing the edges of the vertexes. Therefore, it takes O(V+E) running time complexity.

1.2) Dijsktra's Algorithm:

The algorithm uses a priority queue to keep the nodes with their distances in ascending order by using a comparator to find the minimum distance node at each iteration in O(logV) time. It also keeps an hash set for visited elements, hash map for their parents, hash map for distances of the nodes.

Firstly, it initializes the data structures that it keeps. Initializing the priority queue, visited set and parent map takes only O(1) and initializing the distance map takes O(V) time because the distances of the nodes are infinite at the beginning.

Secondly, it sets the distance of the start node to 0 and adds it to the priority queue in O(1) time before entering the while loop. The loop continues as long as the queue is not empty. Inside the loop, it polls the minimum distance node from the priority queue in O(logV) time and marks it as visited in O(1) time. After that, it traverses all edges of the node and finds the neighbor of this node. It updates the distances of the neighbors in O(1) time if it is needed such that the new distance with this node plus the edge weight is smaller than the current distance of the neighbor, after this update check it adds the neighbors to the priority queue in (logV) time.

In the worst-case such that the start node is connected with all of the vertexes, the outer while loop works in O(V) time and inside it takes O(logV) time to poll the minimum distance node from the queue and adding neighbors to the queue. As a result, it takes O(VlogV) running time complexity.

2) Experimental Results with Running Times:

2.1) Experimental Results:

Map / Time Result (ms)	Breadth First Search Algorithm	Dijstra’s Algorithm
Map01	430	659
Map02	363	666
Map03	371	650
Map04	376	1034
Map05	418	287
Map06	367	441
Map07	273	503
Map08	356	603
Map09	268	514
Map10	74	544
tokyo	359	804
triumph	461	748
vatican	501	959
pisa	356	583

2.2) Comments on Experimental Results:

The first observation as seen from the result table that dijkstra’s algorithm takes more time to compute the resulting path than the breadth first search algorithm. This is related with their complexities since the dijkstra’s algorithm takes O(VlogV) and breadth first search algorithm takes O(V+E), it is expected for breadth first search algorithm to be faster than dijkstra’s algorithm in such kind of unweighted or equal weighted graph.

The second observation is the fact that both algorithms take more time as the number of vertexes increases.