CITS2200 Project

Name: Ciaran Engelbrecht
Student ID: 23169641

Shallows:

My implementation of Shallows uses Dijkstra's shortest path algorithm with a priority queue to further increase efficiency, as well as using an adjacency list. However, this algorithm had to be adjusted, to not find the shortest path, but the path with the minimum depth across the ports for the ship to travel. The priority queue prioritises this ideal path of maximum depth across the route. This requires traversing all vertices of the graph, and in the priority queue store the maximum depth vertex from the set of not yet included vertices. Where P is the number of ports and L is the number of lanes, the time complexity for this implementation is O((L+P)Log P), which is essentially O(L Log P). This is due to the priority queue using binary heap, taking O(Log n) time. Since there will never be a negative weighted edge in this scenario, Dijkstra's algorithm will always work for this. It works by beginning at the origin port, traversing through the graph to find the route/path along the nodes (in this case ports) to find the path with the greatest overall depth, and ensuring that this depth does not increase after passing a port with a lower depth, as this will be the maximum for the route. This depth value is kept track of and is updated if there is a route with greater depth.

https://www.codetd.com/en/article/13887935
https://dzenanhamzic.com/2016/12/14/dijkstras-algorithm-implementation-in-java/
https://www.freecodecamp.org/news/dijkstras-shortest-path-algorithm-visual-introduction/
https://medium.com/@l__j__l/implementing-dijkstra-in-java-69bef4d54d4b