Karim Itani
ECE 36800

PA06 - REPORT

In this assignment, I had to find the shortest path from one side of a river to the other side, given there is a symmetric matrix-type arrangement of poles, with preexisting boards.

To do this I had to first read the inputs, and stored them in a 2-D array. **This has a time complexity of O(V).**

Then I made a graph of all the poles, with each pole being a struct that has pointers to the surrounding poles, plus details like distance from the start point, the side that it was entered from etc. **Creating the graph has a time complexity of O(V).**

After that I created a priority queue that contained all the nodes within itself. The priority queue was built using heapsort/ downward heapify. **Creating the priority queue has a time complexity of O(VlogV).**

Therefore the overall time complexity for the initiation steps is O(VlogV).

To perform Dijkstras, I repeatedly dequeued the minimum distance pole, and updated the distances of the poles around it.

Extracting the minimum pole has a **time complexity of V * logV = O(VlogV).**

Updating the priority queue has a **time complexity of O(ElogV).**

Therefore the calculation part of my code has a time complexity of O((E+V)logV).

Therefore the overall time complexity of my code is O((E+V+V)logV).

Hence the overall time complexity is equal to O(ElogV).

----->small note on E and V. V is the number of poles in the river, and E is equal to 4*V. This is because each pole has an edge to all the 4 poles around it.