Cargo Explanation

Samuel Kent 22704037

Given an array containing Q elements of class Query and number of stops N. We must produce an array of size Q that holds the result of each corresponding query in the same order that the queries are given.

The effects of each query are persistent and will impact the results of later queries. Due to this we must maintain an array of size N that holds the values of each stop after each query has been applied (cargo_arr). For example, cargo_arr at position [2] will hold the value of the stop [2] after a Query has been applied.

So, each Query Q will produce an array cargo_arr of size N and a result of size 1 given by cargo_arr[collect] where the value of collect is given by the current Query. The result is stored in the final array total_cargo in the same order that the queries are given from [0] to [Q].

Initialize an array of size N. By default, each element is 0. Is done with O(N) space complexity but time complexity O(1) since the array is declared as being of static duration and it is initialized by the runtime-environment not Arrays.fill().

Therefore, we loop through and access every Query in the same order they are given in Query[ ]. This always takes O(Q) time regardless of the case as we must use every Query.

Then for the current Query access the three elements that it stores. The static Query class has its elements pointed to by the variables “cargoMass”, “collect” and “deliver”. Accessing each element from Query therefore is done in O(1).

Then we must traverse and modify each element in the cargo_arr that is in an array position greater than or equal to “collect” and less than “deliver”. In the worst case when collect = 0 and deliver = N+1 (or cargoMass = 0) we need to traverse every element in the array taking O(N) time. Modifying each element depending on the value of cargoMass as we traverse the array, is done in O(1).

After the cargo_arr array has been traversed and modifed return cargo_arr[collect] and place it in the final array total_cargo at the same position as the current Query in Query[ ]. I.e for Query[i] result is placed in total_cargo[i]. This is done in O(1) as we are accessing a single element in a fixed position. The array total_cargo[ ] is then returned after all Query’s from queries[ ] have been accessed.

All operations done in constant time become negligible. For each Query in queries[ ] of size Q we are traversing an array of size N. Therefore, the worst case time complexity for this solution is O(Q) x O(N) = O(NQ)