Jacob Archer
Prof. Shen CS 210
6/6/24

Program 2 Complexity Analysis

Push Operation:

When we look at the push method, it involves adding a new jersey to our stack. Essentially, we're creating a new PromotedJersey object and then checking if the stack is empty. If it is, we initialize the current minimum and maximum prices to the price of the new jersey. If not, we update the current minimum and maximum prices by comparing the new price with the last min and max prices in the stack. Finally, we add the new jersey along with its current min and max prices to the stack. All these operations involve a few comparisons and assignments, which are constant-time operations. So, the time complexity is O(1)O(1), meaning the time it takes to execute the push operation doesn't increase with the size of the stack. For space complexity, since we're only using a fixed amount of space for the new PromotedJersey object and updating the min and max prices, it remains constant as well, giving us O(1)O(1) auxiliary space complexity.

Pop Operation:

For the pop method, it’s pretty straightforward. We first check if the stack is empty and if it is, we throw a logic_error exception. If not, we simply remove and return the top jersey from the stack. Removing the last element from a vector is a constant-time operation, so the time complexity here is O(1)O(1). As for auxiliary space, we're only dealing with the space required to hold the removed PromotedJersey object and handling the exception, which doesn't grow with the stack size. Therefore, the auxiliary space complexity is O(1)O(1).

Peek Operation:

The peek method is designed to return the top jersey without removing it. We start by checking if the stack is empty, throwing a logic_error if it is. If the stack isn’t empty, we simply return the top element. Accessing the last element of a vector is a constant-time operation, so the time complexity is O(1)O(1). The space complexity is also constant, as we’re only holding a reference to the top PromotedJersey object and not using any additional space that grows with the stack size. This means the auxiliary space complexity is O(1)O(1).

Get Highest Priced Promoted Jersey:

For the getHighestPricedPromotedJersey method, we first check if the stack is empty and throw a logic_error if it is. Otherwise, we retrieve the current max price from the last element in the stack. We then iterate through the stack to find the jersey with the max price. However, since we maintain the max price in constant time during the push operation, retrieving the highest priced jersey is optimized to be constant time, making the time complexity O(1)O(1). The space used to return the jersey and handle exceptions does not depend on the stack size, so the auxiliary space complexity remains O(1)O(1).

Get Lowest Priced Promoted Jersey:

Similarly, the getLowestPricedPromotedJersey method involves checking if the stack is empty and throwing a logic_error if it is. We retrieve the current min price from the last element in the stack and iterate through the stack to find the jersey with the min price. Like the max price, we maintain the min price in constant time during the push operation, optimizing retrieval to constant time. Hence, the time complexity is O(1)O(1). The space required to return the jersey and handle exceptions does not change with the stack size, giving us O(1)O(1) auxiliary space complexity.