**Data Structures & Algorithm Analysis Assignment 1 README**

Name: Daniel Agu

NetID: DCA190003

Section #: SE 3345.006

**ArrayStack:**

For my ArrayStack program I implemented the provided BKStack interface to right a stack implementation using arrays. The program creates an array for a stack and keeps track of the top of the stack while being able to push, pop, peek, and check if the stack is empty.

isEmpty:

Text

Description automatically generated

The method isEmpty checks the top pointer of the stack to see if it is zero which is what the pointer would be pointing to if there is nothing in the stack. It then returns true if it is empty and false otherwise. The complexity of the function is O(1)

push:

A screenshot of a computer

Description automatically generated with medium confidence

The method push takes in a double and before it enters It into the array stack it makes sure the array stack isn’t full. If it is we create a new array stack five times the length and copy all the old elements of the stack into the new one and set the old array stack variable to the new one so we can continue using the same class variable. After that we set the new top element to the given value and increase the index of top to the next open space. At the best case where we don’t have to create a new array stack and copy the elements over the complexity is O(1) but if we have to create a new stack the complexity will be O(N) due to the for loop so the complexity will be O(N).

pop:

A picture containing graphical user interface

Description automatically generated

The pop method first checks to see if the array stack is empty using the isEmpty method because we shouldn’t pop an element of the stack if its empty. Once we confirmed the stack is not empty we grab the top elements values from the method peek and then move top index down one. We then return the popped element. The complexity of this function is O(1).

peek:

A screenshot of a computer

Description automatically generated with medium confidence

The peek method first checks to see if the array stack is empty using the isEmpty method because we shouldn’t pop an element of the stack if its empty. Once we confirmed the stack is not empty we grab the top – 1 element which is the top element because top index is the next available spot. The method then returns the top element . the complexity for this method is O(1).

With all the methods together the complexity of the implementation ArrayStack is O(N) at worst case.

**ListStack:**

For my ListStack program I implemented the provided BKStack interface to right a stack implementation using a linked list. The program contains a subclass for the linked list node called StackNode and a top node variable that represents the top node. The program is able to push, pop, peek, and check if the stack is empty.

StackNode:

Text

Description automatically generated

The stack node class contains class variables for the node value which is a double and the previous node which is an instance of itself. The class also contains a constructor that allows the user to enter double for the node value and then stack node for the previous node. The rest of the methods are getters and setters for the two class variables, and they all have the complexity of O(1).

isEmpty:

Text

Description automatically generated

The method isEmpty checks the top node of the stack to see if it is null which is what the node would be equivalent to if there is nothing in the stack. It then returns true if it is empty and false otherwise. The complexity of the function is O(1).

push:

Text

Description automatically generated

The method push takes in a double and creates a node using the given double d as the value and the class variable top as the previous node. The method then sets the new top to the new created stack node. The complexity of the method is O(1).

pop:

Text

Description automatically generated

The pop method first checks to see if the list stack is empty using the isEmpty method because we shouldn’t pop an element of the stack if its empty. Once we confirmed the stack is not empty we grab the top elements values from the method peek and then set the new top node to the tops previous node using the get method. We then return the popped element. The complexity of this function is O(1).

peek:

Shape, rectangle

Description automatically generated with medium confidence

The peek method first checks to see if the list stack is empty using the isEmpty method because we shouldn’t pop an element of the stack if its empty. Once we confirmed the stack is not empty we grab the top elements value using the get method and return it. the complexity for this method is O(1).

With all the methods together the complexity of the implementation ListStack is O(1) at worst case.

Because the implementation of the ArrayStack has a complexity of O(N) at worst case and ListStack has a complexity of O(1) we can assume the that ListStack runs more efficiently than ArrayStack.