Adam Winders

CS-215-ON

11 September 2022

Assignment 2.2

1. Write a short summary explaining how you thought through the problem of using a stack to accomplish the task.

**My order of thinking:**

**To check if a String is a valid palindrome, the String would have to be chopped up into its individual characters. From there, each character would need to be added (*push*) to the stack in the order of arrangement in the word.**

|  |
| --- |
| **E** |
| **L** |
| **P** |
| **U** |
| **T** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **T** | **U** | **P** | **L** | **E** |

**“TUPLE”**

**After the new stack is created, the chars need to be removed (*pop*) from the stack, one at a time, from top to bottom, to form a new word that is now the inverse of the original.**

|  |
| --- |
| **E** |
| **L** |
| **P** |
| **U** |
| **T** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **E** | **L** | **P** | **U** | **T** |

**“ELPTUT”**

5a. What is the basic operation in your code?

**The basic operation of my code is a “for” loop.**

5b. What input determines how much time the basic operation takes place?

**The total time for my basic operations is dependent on the String input of *n*.**

5c. Express the number of times the basic operation occurs in terms of n. What does n represent?

*for (int 1 = 0; I < input.length(); i++) {  
 inputStack.push(input.charAt());  
 }*

|  |  |
| --- | --- |
| **i = 0;** | **1 time** |
| **i < input.length** | ***n* + 1 times** |
| **i++;** | ***n* times** |

***n* represents the length of the input String.**

5d. What is the computational complexity of the code in terms of Big O? Explain why.

**The complexity of my code is O(n). This is due to the fact that the processing time for the algorithm grows proportionally to the input, making it a linear time complexity.**

6. Explain how a stack data structure differs from a bag data structure. Explain how you used the stack in your code to solve the palindrome problem.

**Stack data structures differ from bags in one critical way. Bags do not allow you to remove items once they have been added – only the ability to collect items and then iterate through them. Stacks, however, allow you to remove (*pop*) items by removing the top entry, one at a time, until the condition is satisfied, or the stack is empty.**

**As the String was parsed up into its individual chars, they were being added to the stack one at a time, from top to bottom. This allowed me to basically have the String input already in the reverse order that I needed to perform a palindrome validity test. All I needed to do then was to pull out the chars one at a time, until the stack was empty, and assemble them into the inverse of their input.**

Text

Description automatically generated