Chessica Nation

CS-215-ON

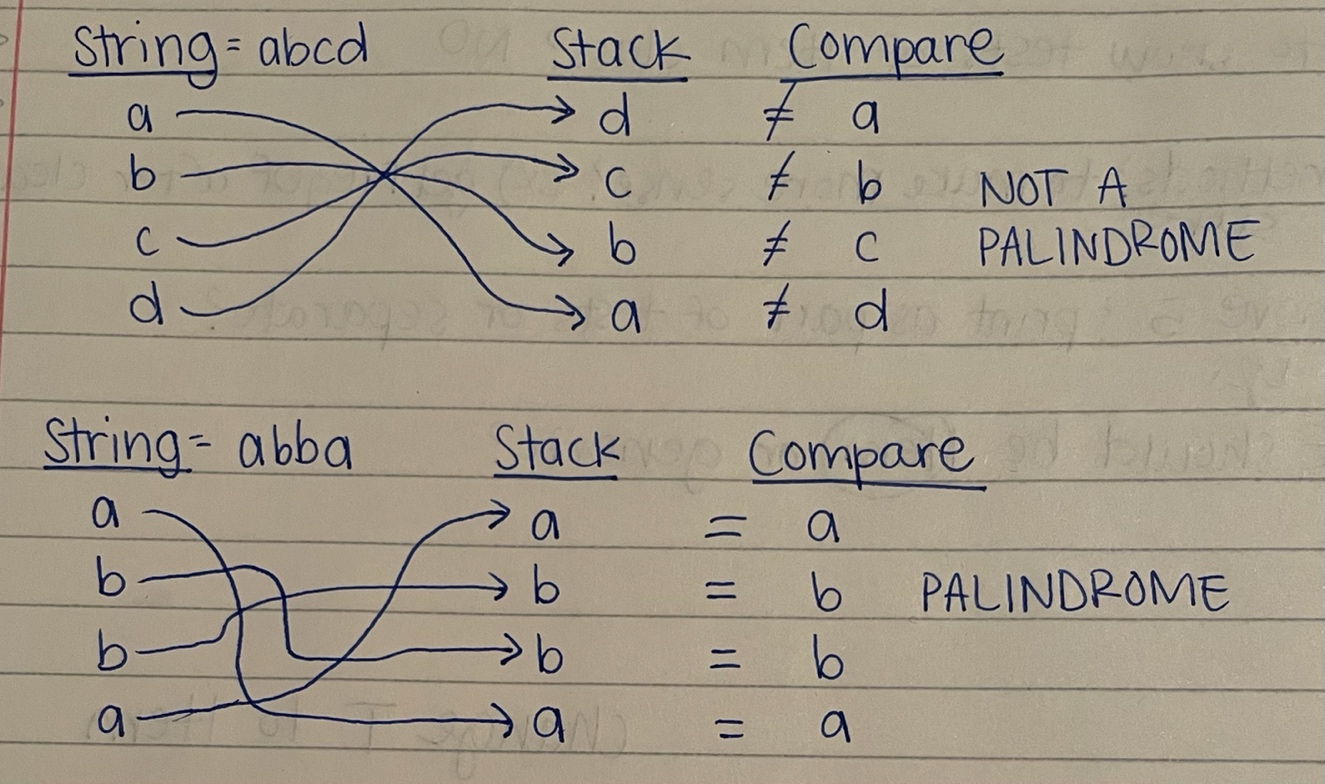
Data Structures

11 Sept 2022

Assignment 2.2

1. Before you begin writing code, think about the palindrome problem. Write a short summary explaining how you thought through the problem of using a stack to accomplish the task. Discuss your thoughts and conclusions. Use a scratchpad and paper and write a very short palindrome, showing how the stack would be used to check it.

While thinking through the palindrome problem, I knew I needed to see if the string was the same backwards as it was forwards. Since I knew stacks ordered data “last in, first out,” I thought I could enter the characters of the string into a stack to reverse them, and then I could pop the characters out to compare them to the original string. If they matched, that would mean the string was a palindrome; if not, it was not a palindrome.



2. Thinking in terms of the material regarding computational complexity that we just finished covering answer the following in the document portion of your homework:

a. What is the basic operation in your code?

The basic operations in my code are push and pop of the stack.

b. What input determines how many times the basic operation takes place?

The size of the string entered by the user determines how many times the push and pop operations will take place.

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

The push and pop operations will occur n times because the loop must execute n times. N represents the number of characters in the string entered by the user. If the string is four characters long, the loop will have to execute four times to push all four characters onto the stack. Similarly, the loop will have to execute four times to pop all four characters out of the stack.

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

The computational complexity of the code is O(n). The push and pop operations are O(1) because they only operate once, on the top entry of the stack. Both loops in the code are O(n) because they must operate n times, the number of characters in the string entered by the user. Since there are two loops, the computational complexity would be 2n; however, since we only care about the largest factor, we simplify the Big Oh notation to O(n).

3. 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.

A stack data structure and a bag data structure are similar in that they can contain items; however, their biggest difference is that the items in a bag are not ordered, whereas the items in a stack have a specific order. Items in stacks are ordered “last in, first out” meaning that the first item added to the stack is placed at the bottom and each additional item that is added to the stack gets placed on the top. While items in a bag can be iterated through or traversed to view or remove, items in a stack can only be accessed or removed from the top. If you wanted to see each item in a stack, you would have to do so by removing each one from the top.

I capitalized upon the stack’s “last in, first out” order to solve the palindrome problem. Since I essentially needed a reverse copy of the string to compare it to, I pushed each character of my string onto a stack. The characters were then in reverse order, so I popped them into a new string. I then used the equals method to compare both strings, the original and the reverse from the stack, to see if they matched. If so, that meant the string was the same in reverse as forward – a palindrome.