Q1. Write all possibilities to check palindrome. Also do time and space complexity analysis.

Ans:

So, to check for palindrome, we can use two approaches

1. Reverse the datatype and compare it with the original datatype
2. Two Pointer Approach

And

We can perform Palindrome for the datatypes:

1. Numbers
2. Strings
3. Arrays
4. Linked List

So

I. String Palindrome Algorithm:

1. First take two integers (left) and (right) that will point to the first and the last character of the string respectively
2. Then, we will use a while loop with condition being that (left) is less than or equal to (right)
3. Inside the while loop, we will use string.charAt(left) is equal to string.charAt(right)
   1. If it is true, then increase the left integer by 1 and decrease the right integer by 1
   2. If it is false , then the string is not a palindrome and break or return false
4. If the whole for loop gets executed, then the given string is a palindrome

Time Complexity: O(n)

Space Complexity: O(1)

II. Array Palindrome Algorithm:

1. First take two integers (left) and (right) that will point to the first and the last character of the array respectively
2. Then, we will use a while loop with condition being that (left) is less than or equal to (right)
3. Inside the while loop, we will check if the given array element at index left is equal to the array element at index right
   1. If it is true, then increase the left integer by 1 and decrease the right integer by 1
   2. If it is false , then the string is not a palindrome and break or return false
4. If the whole for loop gets executed, then the given array is a palindrome

Time Complexity: O(n)

Space Complexity: O(1)

III. Number Palindrome Algorithm:

1. We’ll take the number and reverse it
2. Then we will check if the reversed number is equal to the original number
   1. If true, then the given number is a palindrome
   2. If false, then the given number is not a palindrome

Time Complexity: O(n)

Space Complexity: O(1)

IV. Linked List Palindrome:

1. What we’ll do is, we will traverse through the linked list using a for loop and push the current node element into a stack
2. We will we also a maintain a head pointer that will point to the first node of the linked list
3. Then after iterating the linked list fully,
4. Iterate the linked list again using a for loop, but now this time compare the current node element of the linked list with the topmost element of the stack
   1. If both the elements are the same or equal, move the head pointer to the next node
   2. If they are not equal, the given linked list is not a palindrome, break or return false

If 4.a then, continue doing it till the for-loop finishes

If you have successfully executed the for loop, then the given linked list is a palindrome

Time Complexity: O(n)

Space Complexity: O(n)