CS657 – Design and Analysis of Algorithms

Unit 1 – Individual Project

01/09/2019

Instructor: Professor Amit Dave

The Project code, Time Complexity, and Space Complexity Analysis can be found at my GitHub at:

https://github.com/sgtlm19/Analysis-of-Algorithms-Class

Python Code (Python Central, 2017)**:**

```
def reverseList(array):

    left = 0
    right = len(array) - 1


    while left < right:
        temp = array[left]
        array[left] = array[right]
        array[right] = temp


        left += 1
        right -= 1


    return array
```

Time Complexity Analysis:

The Time Complexity of the reversal algorithm is dependent on the number of swaps needed to reverse the order. The number of swaps needed to reverse the order of the array is a function of the length of the array: $swaps = \left\lfloor \frac{n}{2} \right\rfloor$. This function holds true for worst case scenario.

```
def reverseList(array):

    left = 0                      1 Step (Executed 1 time)
    right = len(array) - 1        1 Step (Executed 1 time)

    while left < right:
        temp = array[left]        1 Step (Executed $\left\lfloor \frac{n}{2} \right\rfloor$ times)
        array[left] = array[right] 1 Step (Executed $\left\lfloor \frac{n}{2} \right\rfloor$ times)
        array[right] = temp       1 Step (Executed $\left\lfloor \frac{n}{2} \right\rfloor$ times)

        left += 1                 1 Step (Executed $\left\lfloor \frac{n}{2} \right\rfloor$ times)
        right -= 1                1 Step (Executed $\left\lfloor \frac{n}{2} \right\rfloor$ times)

    return array
```

1.) t(n) = $2 + 4\left\lfloor \frac{n}{2} \right\rfloor$ (Expression of time complexity as a function of the length of array (n))

2.) t(n) = $2 + \left\lfloor \frac{4n}{2} \right\rfloor$ (Multiply term)

The term with the highest power is 4n so the time complexity is $\emptyset(n)$ for a worst case scenario.

Space Complexity Analysis:

Assuming each int is one memory location. The space complexity is a function of the length of the array. So the space complexity of this algorithm is $\emptyset(\boldsymbol{n})$**.**

References

Python Central. (2017, Oct 18). *List in Python: How To Implement in Place Reversal*. Retrieved from Python Central: https://www.pythoncentral.io/python-reverse-list-place/