ITCS 6114 - Assignment (Extra Credit)

Rishant Dutt, 801104239

PROBLEM STATEMENT:

Design an O(log N) algorithm to find the first and last occurrence of a given number in a sorted array.

For example,

Input:

array1 = [2, 5, 5, 5, 6, 6, 8, 9, 9, 9]

target = 5

The task is to find the first and last occurrence of given target 5 in the given array1.

Output:

First occurrence of element 5 is found at index 1

Last occurrence of element 5 is found at index 3

Input:

array1 = [2, 5, 5, 5, 6, 6, 8, 9, 9, 9]

target = 4

Output:

Element not found in the array

LANGUAGE OF CHOICE: Python 3.7

TIME COMPLEXITY ANALYSIS:

Size: N, [len(input) in code]

Basic Operation: input[mid] == target

Recurrence Relation: $T(N) = T(N/2) + 1$

Using the Master Theorem,
Here, $a = 1$, $b = 2$, $f(n) = 1$
also, $c = \log_b{}^a = \log_2{}^1 = 0$

because, $f(n) = 1 = n^c \log^k n = n^0 \log^0 n$, where $k = 0$
therefore,
Using the Case 2 of Master Theorem,

$$T(n) = O(n^c \log^{k+1} n) = O(n^0 \log^{0+1} n) = O(\log n)$$

PROOF OF CORRECTNESS:

Both the algorithms used in this assignment, namely 'find_first' and 'find_last', work in a similar manner to one another, and are based off of the recursive variant of 'binary search' algorithm. They only differ from binary search in that they consider only the first, or last, occurence of the target as a positive match, respectively.

Hence, their correctness is solely based on the correctness of the binary search algorithm, which I will now proceed to prove.

1.) Base case:
    Let, size of n == 1, that is, there is only 1 element in the sorted sequence. then, either A[mid] == target (where mid = 0), or the sequence 'A' does not contain the target.

2.) Assumption:
    Let us assume that binary search is correct for sorted sequences of length less than or equal to 'k', where k >= 1. Then we prove that binary search must be true for a sequence of length 'k+1'.

3.) Induction:
    For a sorted sequence, 'A', of length 'k+1', and a target, 'X'.

    Case 1: A[mid] == X
        here, the algorithm is clearly correct according to the problem definition.

    Case 2: A[mid] > X
        because, 'A' is a sorted sequence, for 'X' to be present in the sequence it must lie between A[0..mid-1] and because, that

sequence is smaller than ‘k’, therefore, due to our prior assumption, our algorithm is correct.

Case 3: A[mid] < X

because, ‘A’ is a sorted sequence, for ‘X’ to be present in the sequence it must lie between A[mid+1..k+1] and because, that sequence is smaller than ‘k’, therefore, due to our prior assumption, our algorithm is correct.