

---

## DSC 40B - Discussion 03

---

**Problem 1.**

Solve the following recurrence relations.

a)  $T(n) = T(n-1) + n$   
 $T(0)=0$

b)  $T(n)=4T(n/4) + n$   
 $T(1)=1$

**Problem 2.**

Determine the recurrence relation describing the time complexity of each of the recursive algorithms below.

a) `def fact(n):  
 if(n <= 1)  
 return 1  
 else  
 return n*fact(n-1)`

b) `def max_arr(arr):  
 if(len(arr) == 1):  
 return arr[0]  
 mid = len(arr)//2  
 left_max = max_arr(arr[:mid])  
 right_max= max_arr(arr[mid:])  
 if(left_max>right_max):  
 return left_max  
 else:  
 return right_max`

**Problem 3.**

Determine whether each piece of code is correct or incorrect.

a) `def max_arr(arr):  
 max1 = arr[0]  
 max2 = max_arr(arr[1:])  
 if(max1 > max2):  
 return max1  
 else:  
 return max2`

b) `def fib(n):  
 if (n==1):  
 return 1  
 return fib(n-1)+fib(n-2)`

**Problem 4.**

We'll consider an array of **Trues** and **Falses** to be sorted if all of the **Falses** come before any of the **Trues**, like in `[False, False, True, True, True]`.

The function `find_first_true(arr, start, stop)` below is a generalization of the binary search we saw in lecture. It accepts a sorted array of **Trues** and **Falses** and returns the index of the first **True** (if there is one) within `arr[start:stop]`; if the array contains all **Falses**, it returns `stop`.

```
def find_first_true(arr, start, stop):
    if stop - start == 1:
        if arr[start]:
            return start
        else:
            return stop

    if arr[middle]:
        return find_first_true(arr, start, middle)
    else:
        return find_first_true(arr, middle, stop)
```

Modify this function so that it takes in a sorted array of floats and a number  $a$  and returns the index of the first element that is  $\geq a$ .

### Problem 5.

Given a sorted array of distinct integers  $A[1 \dots n]$ , give an algorithm to find out whether there is an index  $i$  for which  $A[i] = i$ . Analyze the time complexity of the algorithm.

(Hint : There is a solution that runs in better than linear  $\Theta(n)$  time!)