MISE Summer Programming Camp

Class 7 Worksheet - Efficient Programming

This worksheet has 2 sections, which should take you around 1 hour to complete. Section 1 contains a series of questions about what we did in class. You'll have to reason about code and run some code in your IDE to see the result and write it down. In Section 2 you'll have to write some programs and submit them to CodeForces.

1. Review questions

1. For each function, write down the time complexity of each one (note, you don't have to understand what the code is doing, you only need to think about the time complexity using the notion of big-O notation you learned in class):

A.

```
def maximum(arr):
    max_so_far = None
    for item in arr:
        if max_so_far is None:
            max_so_far = item
        else:
            max_so_far = max(item, max_so_far)
    return max_so_far
```

B.

```
def insertion_sort(arr):
    for i in range(len(arr)):
        if i == 0:
            continue
        else:
            curr = i
            while curr > 0:
                if arr[curr] < arr[curr-1]:
                    temp = arr[curr]
                    arr[curr] = arr[curr-1]
                    arr[curr-1] = temp
                else:
                    break
```

```
                curr-=1
    return arr
```

C. (Challenge/Optional)

```
def binary_search(arr,item):
    if len(arr) == 0:
        return None
    elif len(arr) == 1:
        if arr[0] == item:
            return 0
        else:
            return None
    elif len(arr) == 2:
        if arr[0] == item:
            return 0
        elif arr[1] == item:
            return 1
        else:
            return None
    else:
        mid = len(arr)//2
        if arr[mid] == item:
            return mid
        elif arr[mid] < item:
            pos = binary_search(arr[mid:],item)
            if pos:
                return  pos + mid
            else:
                return None
        else:
            pos = binary_search(arr[:mid],item)
            if pos:
                return pos
            else:
                return None
```

Hint: Don't worry if part C is hard! Determining the complexity of a recursive function can be difficult. This function is a popular one for finding the index/position of an item in a sorted array. The way it works is by first looking at the middle item. If the middle item happens to be the item

we are looking for, we are done. If on the other hand, the middle item is less than the item we are looking for, then if the item is in the array, it will be to the right of the middle item since the array is sorted. And if the middle item is less than the item we are looking for then we search in the right half.
Tackle this problem by first doing a few simple examples to convince yourself that you understand what the program is doing before you start trying to figure out the complexity.

2. Coding Questions

You should see a new problem set called “Class 7 Problems” (https://codeforces.com/group/K1Fxw6skwV/contest/385740). You should solve at least the first 2 problems (Odd sums and Even products). The last 2 problems are much harder and are intended to be challenge problems for the more experienced students (Colorful Stamp and Maximum Crossings).