Name: Harshit Jain
Access ID: hmj5262
Recitation: 8

Problem 0

Points:

Acknowledgements

(a) I did not work in a group.

(b) I did not consult without anyone my group members.

(c) I did not consult any non-class materials.

Problem 1 **Points:**

Divide-and-Conquer

(a) We will first have the function which returns the frequency of the element in the given list. This function will take $O(n)$ time.

```
def frequencyCalculator(Array, element):
    count = 0
    for ele in Array:
        if (element == ele):
            count += 1
    return count

```

Now, to start, we will split the array A into 2 subarrays A_1 and A_2 of half the size. Then we will calculate the majority elements of A_1 and A_2. The algorithm is as follows:

```
def majorityElement(Array, low, high):

    subArray = Array[low:high+1]

    # base case
    if (len(subArray)==1):
        return subArray[0]

    mid = (low+high)//2
    leftMajorityElement = majorityElement(Array, low, mid)
    rightMajorityElement = majorityElement(Array, mid+1, high)

    if (leftMajorityElement == rightMajorityElement):
        return leftMajorityElement

    leftFrequency = frequencyCalculator(subArray, leftMajorityElement)
    rightFrequency = frequencyCalculator(subArray, rightMajorityElement)

    if (leftFrequency > len(subArray)//2):
        return leftMajorityElement
    elif (rightFrequency > len(subArray)//2):
        return rightMajorityElement
    else:
        return -1 # no majority element found

```

(b)

(c) Here, we are choosing the majority element of sub-arrays after dividing them in half. Moreover, frequency calculation is using $O(n)$ time as showen above. Therefore, the recurrence relation for this algorithm is given by:

$$T(n) = 2T(n/2) + O(n)$$

According to Master's Theorem, the time complexity: $\underline{O(nlogn)}$

Problem 2

Points:

Reverse graph

(a)

(b)

Problem 3

Points:

Graph Basics

(a)

Adjacency-list

A	$\rightarrow B \rightarrow E$
B	$\rightarrow G \rightarrow D$
C	$\rightarrow I \rightarrow H$
D	$\rightarrow E$
E	$\rightarrow D$
F	$\rightarrow$
G	$\rightarrow D \rightarrow F$
H	$\rightarrow I$
I	$\rightarrow$

(b) The most number of edges that an undirected graph can have are : $\frac{|V|(|V|-1)}{2}$

(c) Since the degree of a vertex is the number of edges incident with that vertex, the sum of degree counts the total number of times an edge is incident with a vertex. Since every edge is incident with exactly two vertices, each edge gets counted twice, once at each end. Thus the sum of the degrees is equal twice the number of edges. Let n be the number of edges in a simple graph $G(E,V)$. We proceed our proof

(d)

(e)