Problem 1: Asymptotic Analysis Practice

(a) [5 points] Prove or disprove that $\log_k n \in O(\lg n)$ for any $k > 1$. (Note that lg refers to $\log_2$)

Answer:- Now according to the definition of the log and by the definition of big-O, we have
f(n) <= c.g(n) where n>k, c>0 and k>0

According to the problem statement, we have, $\log_k n \in O(\lg n)$ for any $k > 1$

This can be re-written as given below:-
f(n) = $\log_k n$ g(n) = lg_n

In the problem statement, it has already been mentioned that k > 1 and n > 1 Therefore, we can say that:-
$\log_k n$ <= c.lg_n
Now as we know that $lg_n = \log_2 n$
Replacing this in the given equation, we have,
$\log_k n$ <= c.$\log_2 n$
taking the logs on one side, we have $\frac{\log_2 n}{\log_k n}$
From this equation, we can say that c > 1
This means that $\log_k n > 1$
Hence , we can say that,

$\frac{\log_2 n}{\log_k n} \leq c \log_2 n$
By using this logarithmic expression, we can say that $\log_k n \in O(\lg n)$ for any $k > 1$ is true
Hence Proved

b) [5 points] The following recurrence relation solves to $O(n \log^2 n)$. Prove this using some variety of substitution. Do not use the Master method.
T(n) = $2T\left(\frac{n}{2}\right) + n \lg n$
T(1) = 0

Answer:- The equation given is:- T(n) = $2T\left(\frac{n}{2}\right) + n \lg n$
Now for using substitution method, we have:-
Replace n by $\left(\frac{n}{2}\right)$ Therefore, we have, T($\left(\frac{n}{2}\right)$) = $2T\left(\frac{n}{4}\right) + \left(\frac{n}{2}\right) \lg \left(\frac{n}{2}\right)$

Similarly, we can also Replace n by $\left(\frac{n}{4}\right)$
This makes the equations as:-

T($\left(\frac{n}{4}\right)$) = $2T\left(\frac{n}{8}\right) + \left(\frac{n}{4}\right) \lg \left(\frac{n}{4}\right)$

$T(n) = 2[2T\frac{(}{n}4) + \frac{n}{2} \log \frac{n}{2}]n \log n$

T(n) = $4[2T\left(\frac{n}{8}\right) + \left(\frac{n}{4}\right) \lg \left(\frac{n}{4}\right)] + n \lg \left(\frac{n}{2}\right) + n \lg n$

T(n) = $8T(\left(\frac{n}{8}\right)) + n \lg \left(\frac{n}{4}\right) + n \lg \left(\frac{n}{2}\right) + n \lg n$

$T(n) = 8T(\frac{n}{8}) + n[\log n - \log 4] + n[\log n - \log 2] + n \log n$

T(n) = $8T(\left(\frac{n}{8}\right)) + n[\log n$ - $\log 4]$ + n$[\log n$ - $\log 2]$ + n

$T(n)$=8T$(\left(\frac{n}{8}\right)) + n[\log n$ - 2] + n$[\log n$ - 1] + n

$T(n)$=8T$(\left(\frac{n}{8}\right)) + n\log n$ - 2n + n $\log n$ - n + n

$T(n)$=8T$(\left(\frac{n}{8}\right)) - 3n + 3n\log n$

T(n)=$2^k T(\frac{n}{2^k}) + kn[\log n - 1]$

T(n) = $0 + n\log n[\log n]$

T(n) = $n(\log n)^2$

$T(n) = n\log^2 n$

Hence Proved

(c) [5 points] Suppose that $f(n)$ and $g(n)$ are non-negative functions. Prove or disprove the following: if $f(n) \in O(g(n))$ then $2^{f(n)} \in O(2^{g(n)})$.

Answer :-
According to question, $f(n)$ and $g(n)$ are non-negative functions
$f(n) = 2^{f(n)}$ and $g(n) = 2^{g(n)}$
Now, since we know that $f(n) \leq cg(n)$
This means, we can write:-
$2^{f(n)} \leq 2^{cg(n)}$
$= (2^c)^{g(n)}$
Now Since we know that $c > 1$,
This means , The time complexity will be of the terms:-
$2^{g(n)}$. So f(n)=2n, g(n)=n.
This can be written as:-
$2^{f(n)} \leq 2^{cg(n)}$
$2^{2n} \leq 2^n$
From the above condition, we can say that the condition is not satisfied.
Hence we can say that $2^{f(n)} \in O(2^{g(n)})$ is not true.
Hence Disproved

Problem 2: Peak-finding Given a set of real numbers stored in an array A find the index of a *Peak*, where a *Peak* is defined as an element that is larger or equal to the both the elements on its sides. (Note: you only need to return *a* peak, not the highest one.)

Example array:

-2	6	-1	4	9	-5	5

Assuming the array one based it has peaks at indices $\{2, 5, 7\}$.

(a) [5 points] Give a linear time algorithm to solve this problem. (This should be obvious)

For this problem, we have an array, We will iterate through the array from array[0] and array[n-1] where n is the length of the array.
Starting from the end elements of the array, we have:-

```
First Checking the condition:
  if(n==0):
    return 0
  if(array[0]>=array[1]):
    return 0
```

```
  if(array[n-1]>=array[n-2]):
    return n-1


    Here, firstly we check the condition for the number of elements after which the second condition ch

Now, for finding the peak , we have:-

   if(array[i]>= array[i+1] and array[i]>=array[i-1])
     return i

Since here we iterate only once in a loop from array[0] to array[n-1] where comparison
is done among the array elements hence here the complexity of the algorithm is O(n)
```

(b) [10 points] Give a $O(\log n)$ time algorithm to solve this problem.
Answer:-

```
For finding the peak and the complexity be $O(\log n)$, it should follow an approach which is similar t

The first step that we will perform is initialization of the variables.

left->0

right->length_of_array-1

array->{-2,6,-1,4,9,-5,5}

The next step would be to find the middle element.

This can be done as follows:-

middle -> left + (right-left)/2

In the next step, we will check the corner elements.
If the corner element is the middle element, then we will return the middle element.

if(middle == 0 || array[middle-1] <= array[middle] && middle = n-1  || array[middle+1 ] <= array[middle]
return the middle element.

After the above step, if the middle element is not found, then check for peak elements,

This can be done by checking the element to the left of the middle element and if middle is greater tha

if (middle>0 and array[middle-1] >= array[middle]
return the subarray from left to middle-1
else
return the subarray middle+1 to right

This is how we divide the array into smaller pieces and using this, solve the required problem.

Time Complexity:-
Since we are finding the middle element and subdividing till we find the peak, this algorithm takes tim

Pseudo code:-
mid = low + (high-low)/2
```

```
if ((mid==0 or array[mid-1] <= array[mid]) and
(mid==n-1 or array[mid+1]<=array[mid]))
    return mid

else if(mid>0 and array[mid-1]>array[mid]
    return find(array,low,mid-1,n)

else
    return find(array,mid+1,high,n)
```

Q2(C)

What if instead of a simple array you are given a square matrix, where a *Peak* is now defined as an element larger or equal to its four neighbours. Give a $O(n\ logn)$ solution to this variant of the problem.

Answer:-

```
For the square matrix, we will have to perform the following steps to find the peak element.

We will check the middle column and then find the max element accordingly,

The first step that we will perform is initialization of the variables.

left->0

right->length_of_array-1

array->2-d matrix

The first step is to check the number of columns in the array
if (column=1, exit)
else

The second step is to find the middle element
middle -> set as pivot

The next step is to check the maximum element in this column.
if(array[left] < array[right && middle]
maximum element found

The next step is to find the peak by checking this with the elements present on either side of the colu

pseudocode:-

length_of_column -> check
if column == 0
Not a 2-d Array
else
middle = column/2
Find peak
check:- middle>column index, or middle<0
not found,
middle=column and array[index][middle] >array [index][middle-1]
found middle
```

```
else
if array[index][mid]>array[index][mid+1] and array[index][mid]>array[index][mid-1]
found middle and peak
else
middle=begin value+column /2
found peak
```

Assignment-1

Parth Parashar (923928157)

1