Gebze Technical University
Computer Engineering

CSE 222 - 2019 Spring

HOMEWORK 4 REPORT

TAYLAN ÖNDER
151044015

Course Assistant: Ayşe Şerbetçi TURAN

Question 1

1.a

```
public Node countIncreasingElements() {
    int curr_len = 1, max_len = 1;
    int total_count = 1, res_index = 0;
    Node temp_result=null;
    Node result=null;
    Node head=null;
    for (Node curr = head_node; curr.next != null; curr = curr.next) {
        if (curr.data < curr.next.data) {
            Node newNode = new Node();
            newNode.data = curr.data;
            if (curr_len == 1) {
                if (temp_result == null) {
                    temp_result = newNode;
                    result = temp_result;
                }
            }
            curr_len++;
            temp_result.next = newNode;
            temp_result = temp_result.next;
            newNode = new Node();
            newNode.data = curr.next.data;
            temp_result.next = newNode;
        }
        else {
                // compare maximum length with len.
                if (max_len < curr_len) {
                    head = result;
                    temp_result=new Node();
                    temp_result=null;
                    max_len = curr_len;
                }
                curr_len = 1;
            }
        total_count++;
    }

    if (max_len < curr_len) {
        head = result;
        temp_result=new Node();
        temp_result=null;
        max_len = curr_len;
    }
    return head;
}
```

This function time complexity is O(n).Because loop turn end of node.If and else conditions are constant effect time complexity.Complexity is approximately n+26.But 26 is a constant.

1.b

```
public Node RecursivecountIncreasingElements(Node head,Node curr,Node re-
sult,Node temp_result,int curr_len,int total_count,int max_len) {
    while(curr!=null) {
        if(curr.next!=null) {
            if (curr.data < curr.next.data) {
                Node newNode = new Node();
                newNode.data = curr.data;
                if (curr_len == 1) {
                    if(temp_result==null){
                        temp_result=newNode;
                        result=temp_result;
                    }
                }
                curr_len++;
                temp_result.next = newNode;
                temp_result=temp_result.next;
                newNode = new Node();
                newNode.data=curr.next.data;
                temp_result.next=newNode;
            } else {
                if (max_len < curr_len) {
                    head = result;
                    temp_result=new Node();
                    temp_result=null;
                    max_len = curr_len;
                }
                curr_len = 1;
            }
            total_count++;
        }
       if(curr!=null) {
            return RecursivecountIncreasingElements(head, curr.next, result,
temp_result, curr_len, total_count, max_len);
        }
    }
    if (max_len < curr_len) {
        head = result;
        temp_result=new Node();
        temp_result=null;
        max_len = curr_len;
    }
    return head;

}
```

Recurrence relation is T(n) = O(n) + T(n+1).It means T(n)=T(n+1)+n.
In master theorem a=1,b=1,d=1.So rule is a=b^d.It means 1=1^1.Thus time complexity is (n^1)*logn.So complexity is nlogn.

Induction prove :

Prove T(n) is O(nlogn) ie by definition

T(n) <= cn for all n >= n0

Assume:

T(n - 1) <= c(n - 1).log(n-1) then

T(n) = T(n - 1) + O(1)

T(n) <= c(n - 1)*log(n) + d

T(n) <= c(n - 1)*log(n) + d - c

T(n) <= c(n - 1)*log(n) for c >= d

Use T(1) as a base case to complete the proof, giving you n0 = 1

Question 2

```
public void array_element_sum(){
    int array1[]={1,5,9,13,15,16};
    int x=28;
    int i=0;
    int j=array1.length-1;
    while(i<=j){
        if(array1[i]+array1[j]==x){
            System.out.println("first  :"+i+"  second :"+j);
            return;
        }
        if(array1[i]+array1[j]<x){
            i++;
        }
        else{
            j--;
        }
    }
}
```

Θ (n) is time complexity of function.Main loop turn according to array length.It means n.If and else functions are constant effectively time complexity.

Question 3

for (i=2*n; i>=1; i=i-1)	-> 2n complexity turn 2n
for (j=1; j<=i; j=j+1)	->2n complexity
for (k=1; k<=j; k=k*3)	->log(n/3)complexity
print("hello")	->O(1)

When we calculate first loop connected n.Second loop connected i so i connected n.Third loop connected j, this j connected i, this i connected n.So this functions complexity is 2n*2n*log(n/3).Time complexity is O(n^2log(n)).

Question 4

```
float aFunc(myArray,n){
    if (n==1){
        return myArray[0];
    }
     //let myArray1,myArray2,myArray3,myArray4 be predefined arrays
    for (i=0; i <= (n/2)-1; i++){
        for (j=0; j <= (n/2)-1; j++){
            myArray1[i] = myArray[i];
            myArray2[i] = myArray[i+j];
            myArray3[i] = myArray[n/2+j];
            myArray4[i] = myArray[j];
        }
    }
    x1 = aFunc(myArray1,n/2);
    x2 = aFunc(myArray2,n/2);
    x3 = aFunc(myArray3,n/2);
    x4 = aFunc(myArray4,n/2);

    return x1*x2*x3*x4;
}
```

Loops complexity n^2

Recursion call 4 times

Recurrence relation T(n) is O(n^2)+4*T(n/2).According to master theorem a=4,b=2,d=2.So rule is a=b^d -> 4=2^2.It means (n^d)*logn , thus time complexity is n^2*log(n).