

**Gebze Technical University  
Computer Engineering**

**CSE 222 - 2019 Spring**

**HOMEWORK 4 REPORT**

**NEVRA GÜRSES  
161044071**

**Course Assistant: Ayşe Şerbetçi Turan**

# QUESTION 1

## Problem Definition

There are a single linked list. We have to find the sublist of this single linked list that is maximum length sorted. We have to find this sublist as iteratively in first part, then we have to find it recursively in second part of this question. And also make complexity analysis of every part.

## Problem Solution Approach for part a

a) For first part, I keep two variable. First is keep that keeps size of max length sublist. Second is maxSize that is current maxSize. Then, I assign them one. Then I start to traverse the linked list. If data of next index is bigger than data of current index, I increase maxSize. If not, I control whether maxSize is bigger keep or not. If it is bigger, I assign maxSize to keep. And I keep starting index of this sorted sublist. I do this control when I traverse all elements. After, I create a temporary linked list, then I add element of maximum length sorted sublist in this linked list. Then, I return this temporary linkedlist.



## Solution

## Java Code Of This Function With Iteratively

```
public static LinkedList findMaxLenghtSubList(LinkedList list){
    int keep=1,index=0;
    int maxSize=1,i=1;
    LinkedList temp = new LinkedList(); //temporary LinkedList object created.
    int prev ;
    int forward;
    while(i< list.getSize()) {
        prev = list.get(i-1);
        forward=list.get(i);
        if (prev<forward ) {
            maxSize++
        }
        else {
            if (maxSize > keep) {
                keep = maxSize; //uploades length of max length sublist.
                index=i-keep; //keeps start index of max length sublist.
            }
        }
        i++;
    }
    return temp;
```

$n^2$  because  
get method  $\rightarrow n$   
and traversing  $\rightarrow n$   
linked list.

```

    }
    maxSize = 1; //to control length of another sublists.
    }
    ++i;
}
//Controls size of last sublist.
if(maxSize>keep){  1
    keep=maxSize;
    index=list.getSize()-keep;
}
int j=index;
//This loop provide to add max length sublist in temporary linkedlist.
while(j<keep+index){  n * m -> Note m is length of sublist.n is to get method.
    temp.add(list.get(j));
    ++j;
}
return temp; //returns temporary LinkedList.
}

```

**Note:** This method is in my own writed LinkedList class and its methods.

## a) Calculation Time complexity of Iteratively Version

Time complexity of this function is  $O(n^2)$ .Analyze of this:

First while loop I control data of current index with data of next index.To do this,I traverse linked list with this control at one time.This traversing  $O(n)$  time.But in while loop I using get method of linked list and this method has  $O(n)$  time complexity.There is if - else statements and time complexity of this statements is  $O(1)$ .So the total time complexity of first while loop is  $O(n*n) = O(n^2)$

Another while loop assigning element of sublists in temporary linked list and time complexity of this while loop  $O(m*n)$ . ==>If m is equal to n , it is  $O(n^2)$

$O(n^2) + O(m*n) = O(n^2)$  .So the result of time complexity of this function  $O(n^2)$ .


## Problem Solution Approach for part b

**b)**In this part,the parameter of function is actual list,temporary list that keeps max length sorted list, variable keep to keep length of max length sorted sublist, variable maxIndex to keep length of current sublist and variable i to keep size.The termination condition is i is equal to size of actual list.If i is equal it,I add max length sorted sublist in temporary list.If i is smaller than size,I comparision current data in index i and data of index i-1.If current data bigger than previous data,I increase maxLength.If not , I update keep as some

condition than I increase i and call this method again. So I traverse all list recursively and return temporary list.

## Java Code Of This Function With Recursively

```
public static LinkedList find(LinkedList list, LinkedList temp, int i, int maxIndex, int keep, int index){
    if(i==list.getSize()){
        if(keep < maxIndex){
            keep=maxIndex;
        }
        for(int j=index; j<keep+index; j++){  $\rightarrow n$ 
            temp.add(list.get(j));  $\rightarrow n$ 
        }
        return temp ;
    }
    else{
        if(list.get(i-1) < list.get(i)){  $\rightarrow n$ 
            ++ maxIndex ;
        }
        else{
            if(keep< maxIndex){  $\rightarrow 1$ 
                keep=maxIndex;
                index = i -keep;
            }
            maxIndex=1;
        }
        return recursion(list,temp,i+1,maxIndex,keep,index);  $\rightarrow 1/2T(n-1)$ 
    }
}
```



## Calculation Time complexity of Recursively Version

With master theorem :  $T(n) = 1/2T(n-1) + n^2$

$$a=1/2$$

$$b=1$$

$$d=2$$

$a < b^d$  because  $1/2 < 1^2$  so Master Theorem Format should be  $\Theta(n^d)$  format.

So, time complexity of this recursive function is :  $\Theta(n^2)$

### With Induction:

$$T(n) = 1/2T(n-1) + n^2$$

Suppose  $T(n) = \Theta(n^2)$

So  $c_1n^2 \leq T(n) \leq c_2n^2$  by rule of  $\Theta$  notation.

By Induction, assume  $T(n-1)$  is true.  $\Rightarrow$  So  $c_1(n-1)^2 \leq T(n-1) \leq c_2(n-1)^2$

We put  $T(n-1)$  in equation  $T(n)$  :

$$\frac{1}{2}(c_2(n-1)^2) + n^2 \leq T(n) \leq \frac{1}{2}(c_2(n-1)^2) + n^2$$

$$\frac{3}{2}c_1n^2 - c_1n + \frac{c_1}{2} \leq T(n) \leq \frac{3}{2}c_2n^2 - c_2n + \frac{c_2}{2}$$

In complexity calculations constants and low order terms can ignore, So;

$c_1n^2 \leq T(n) \leq c_2n^2$  . So  $T(n)$  is also true  $\Rightarrow T(n) = \Theta(n^2)$

## QUESTION 2

### Problem Definition

There are a sorted array, we have to find two numbers in this array that their sum is exactly equal to given number. And also have to make and analyze its time complexity as  $\Theta(n)$ .

### Problem Solution Approach

This array is sorted, so I don't need to sort it. After I keep 2 variable as left and right for first index and for last index. Then I start to add value in left index and value in right index. If they are equal, I keep and record this two values in stringbuler to print all two numbers that are in this array and their sum is equal to given number. If their sum is less than given sum number, I increase index left, on the other hand if, their sum is more than given sum number, I decrease index right. And if I traverse all numbers in array and there are no sum as equal to given number, I give a warn message.

Because, I traverse all numbers in this array, time complexity of this function is  $\Theta(n)$ .

## Solution

### Java Code Of This Function:

```
public static String find(int [] arr,int sum){
```

```
    StringBuilder build=new StringBuilder();
```

```
    int right=arr.length-1;
```

```
    int left=0,stop=0 , count=0;
```

```
    while(left<right && stop==0 ){
```

```
        if(arr[left]+arr[right]==sum){ —————> n
```

```
            build.append("First number:" +arr[left]+ "Second number:" +arr[right]);
```

```
            if(left <right ){ —————> 1
```

```
                ++left;
```

```
                ++ count;
```

```
            }
```

```
            //This statement provide to exit from loop.
```

```
            else{ —————> 1
```

```
                stop=1;
```

```
            }
```

```
        }
```

```
        else if(arr[left]+arr[right]<sum){ —————> 1
```

```
            left++;
```

```
        }
```

```
        else{ —————> 1
```

```
            right --;
```

```
        }
```

```
    }
```

```
    //If there are no two numbers that their sum is equal given number:
```

```
    if(count==0){ —————> 1
```

```
        System.out.println("There are no numbers for this sum!");
```

```
    }
```

```
    return build.toString();
```

```
}
```

n

## Pseudocode Of This Function

def find(Array,sum)

create two variable as right and left.

assign to left is 0, assign to right is size of Array – 1.

create variables as sum ,count, stop and assign them 0.

While( left is less than right and stop is equal to 0 )

if ( Array(left) + Array(right) is equal to sum )

append in stringbuilder array(left) and array(right)

if ( left is less than right)

increase left and count

else

assign stop is 1

else if ( Array(left) + Array(right) is less than sum)

increase left

else

decrease right

if ( count is 0 )

print warning message.

## Time Algorithm:

Briefly, This function prints all two numbers that their sum is equal to exactly x. Because traversing all numbers in this array one time, time complexity of this function  $\Theta(n)$ .

## QUESTION 3

### Problem Definition

There are a code snippet, we have to calculate running time of this code.

### Problem Solution Approach

There are 3 for loop. So I calculate running time of each for loop separately. After that, I multiply them.

### Solution

```
for (i=2*n; i>=1; i=i-1) -----> Running time of this for loop is 2n
    for (j=1; j<=i; j=j+1) -----> Running time of this for loop is 2n
        for (k=1; k<=j; k=k*3) ----> Running time of this for loop is  $\log_3 2^n$ 
            print("hello") -> Running time is constant 1
```

So, total running time of this code snippet is:

$$T(n) = O(2n * 2n * \log_3 2^n * 1)$$
$$= O(4n^2 * \log_3 2^n)$$

Constants can be ignored for running time calculations. So The result is:

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



## QUESTION 4

### Problem Definition

There are a recursive function. We have to write recurrence relation of it and then we have to analyze its time complexity.

### Problem Solution Approach

I find recurrence relation of this recursive function by following statements and then if Master Theorem appropriate to calculate its total complexity, I calculate it by using Master Theorem. If not appropriate, I use different solutions.

### Solution

```
float aFunc(myArray,n){
    if (n==1){
        return myArray[0]; → Constant time -> O(1)
    }
    //let myArray1,myArray2,myArray3,myArray4 be predefined arrays
    for (i=0; i <= (n/2)-1; i++){ → O( n/2) - 1
        for (j=0; j <= (n/2)-1; j++){ → O( n/2) - 1
            myArray1[i] = myArray[i];
            myArray2[i] = myArray[i+j];
            myArray3[i] = myArray[n/2+j];
            myArray4[i] = myArray[j]; → Time complexity of this assignments is O(1) -> constant time.
        }
    }
    x1 = aFunc(myArray1,n/2); → T ( n/2 )
    x2 = aFunc(myArray2,n/2); → T (n/2)
    x3 = aFunc(myArray3,n/2); → T (n/2)
    x4 = aFunc(myArray4,n/2); → T (n/2)

    return x1*x2*x3*x4; → Constant time -> O(1)
}
```

So Recurrence Relation of this function is:

$$T(n) = 4 T(n/2) + n^2$$

This relation is appropriate for Master Theorem. According to Master Theorem:

$$a = 4$$

$$b = 2$$

$$f(n) = n^2$$

$$n^{\log_b a} \implies n^{\log_2 4} \implies n^2$$

$$n^2 = f(n) \text{ So ; } T(n) \text{ must be } \Theta ( n^{\log_b a} * \log n ) \text{ format. } \implies \Theta ( n^{\log_2 4} * \log n )$$

And the time complexity of this function:

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

## QUESTION 5

### Problem Definition

There are a 2D array that includes non-negative numbers. We have to write a iterator class to traverse this 2D array as spirally clockwise starting at the top left element. This iterator class must be recursive and it have to called recursively.

### Problem Solution Approach

Firstly, next method of iterator class start to return first top left element. Then it traverse and return elements of this array left to right, top to bottom, right to left and bottom to top. By doing this, I traverse out big rectangle by spirally. Then, I update starting positions and also column and row numbers. After that I call this next method again and so, I traverse small rectangles as spirally. Until, I traversing all elements in this 2D array, this recursive calls and operations process. So I traverse all elements in this 2D array as spirally clockwise. After I create a method. This method calls next method as recursively, so it prints elements of this array until hasNext is not null.

## System Requirements

This part is writing as java so it require IntelliJ idea. So it requires Jdk 11.0.2

IntelliJ system require :

Microsoft Windows 10/8/7/Vista/2003/XP (incl.64-bit),

2 GB RAM minimum, 4 GB RAM recommended.

1.5 GB hard disk space + at least 1 GB for caches.

1024x768 minimum screen resolution.

## Running Results

```
"C:\Program Files\Java\jdk-11.0.2\bin\java.exe"
```

```
THE GIVEN 2D ARRAY:
```

```
1  2  3  4
5  6  7  8
9  10 11 12
13 14 15 16
```

```
THE ITERATOR ORDER IS:
```

```
1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10
```

```
Process finished with exit code 0
```

```
"C:\Program Files\Java\jdk-11.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA\bin\idea_rt.jar=5000:C:\Program Files\Java\jdk-11.0.2\bin" -Dfile.encoding=UTF-8
```

```
THE GIVEN 2D ARRAY:
```

```
1  2  3  4  5  6
7  8  9  10 11 12
12 14 15 16 17 18
17 18 19 20 21 22
23 24 25 26 27 28
```

```
THE ITERATOR ORDER IS:
```

```
1 2 3 4 5 6 12 18 22 28 27 26 25 24 23 17 12 7 8 9 10 11 17 21 20 19 18 14 15 16
```

```
"C:\Program Files\Java\jdk-11.0.2\bin\java.exe" "-javaagent:
```

```
THE GIVEN 2D ARRAY:
```

```
1   2   3   4   5   6  
7   8   9   10  11  12  
12  14  15  16  17  18
```

```
THE ITERATOR ORDER IS:
```

```
1 2 3 4 5 6 12 18 17 16 15 14 12 7 8 9 10 11
```