CSE222 HOMEWORK 4

MUHAMMET ÇAĞRI YILMAZ

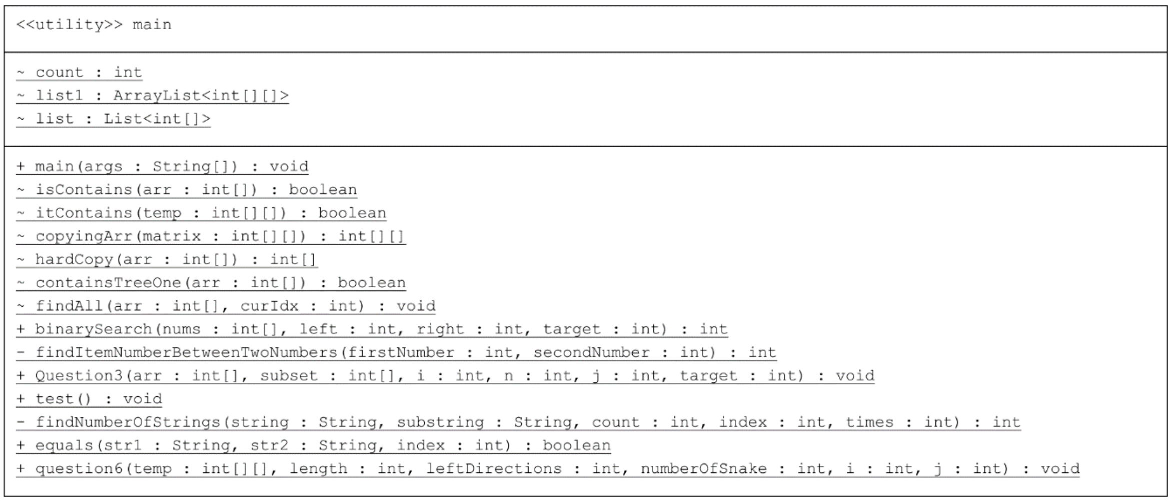
1901042694

System Requirements

Before I started, I defined problems and find a way to solve them.

- Determine the base case for each one
- Write a prototype for the recursive function.
- Use the solutions of the smaller problem to solve the large problem
- Write a comment line which describes what the function does

Class Diagrams



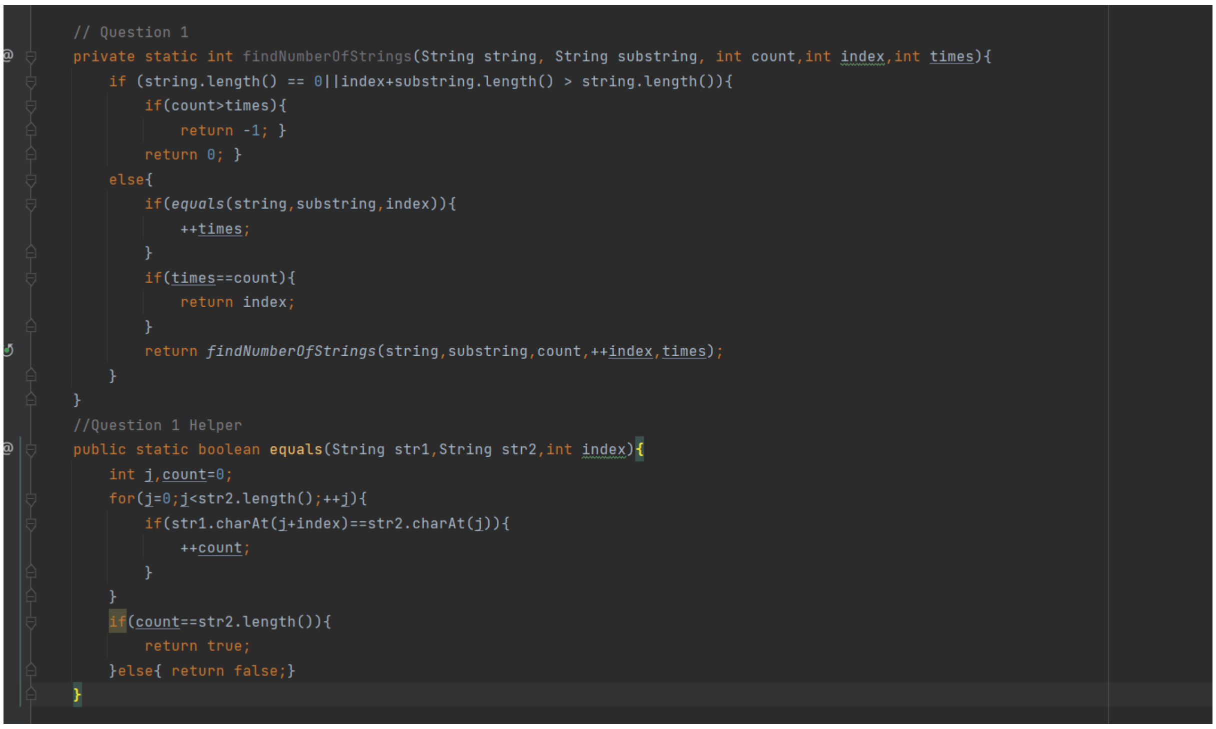
```
<<utility>> main

~ count : int
~ list1 : ArrayList<int[][]>
~ list : List<int[]>

+ main(args : String[]) : void
~ isContains(arr : int[]) : boolean
~ itContains(temp : int[][]) : boolean
~ copyingArr(matrix : int[][]) : int[][]
~ hardCopy(arr : int[]) : int[]
~ containsTreeOne(arr : int[]) : boolean
~ findAll(arr : int[], curIdx : int) : void
+ binarySearch(nums : int[], left : int, right : int, target : int) : int
- findItemNumberBetweenTwoNumbers(firstNumber : int, secondNumber : int) : int
+ Question3(arr : int[], subset : int[], i : int, n : int, j : int, target : int) : void
+ test() : void
- findNumberOfStrings(string : String, substring : String, count : int, index : int, times : int) : int
+ equals(str1 : String, str2 : String, index : int) : boolean
+ question6(temp : int[][], length : int, leftDirections : int, numberOfSnake : int, i : int, j : int) : void
```

Problem Solutions Approach

Question1:

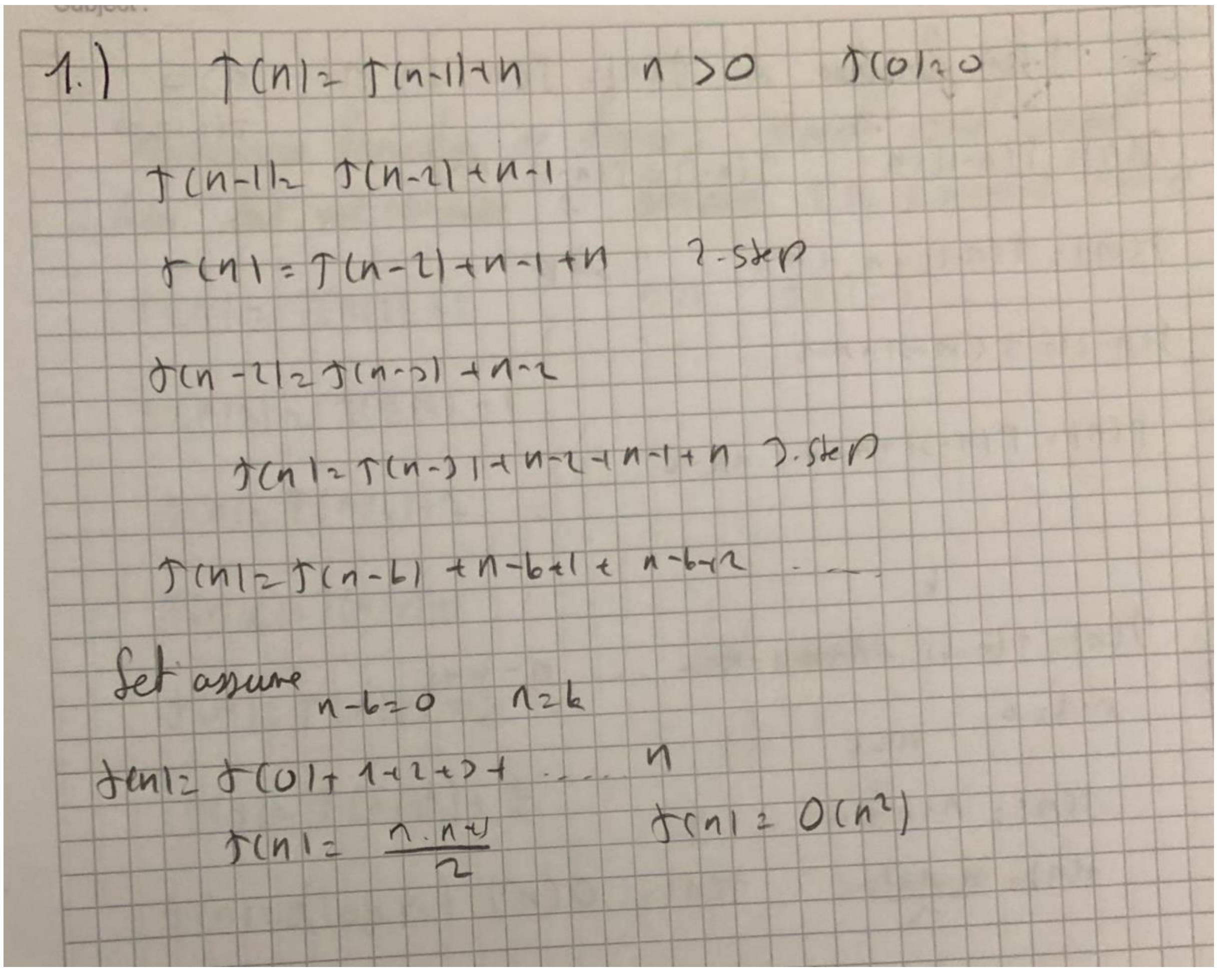
The question is that: Write a recursive function to search a given string in another given bigger string. The function should return the index of the *ith* occurrence of the query string and return -1 when the query string doesn't occur in the big string or the number of occurences is less than *i*.

We have a helper function that find the substring and part of the strings are equal or not.
İf it is equal we increase times one time.
In recursive function we have a index parameter and it goes until a condition .



```java
// Question 1
private static int findNumberOfStrings(String string, String substring, int count,int index,int times){
    if (string.length() == 0||index+substring.length() > string.length()){
        if(count>times){
            return -1; }
        return 0; }
    else{
        if(equals(string,substring,index)){
            ++times;
        }
        if(times==count){
            return index;
        }
        return findNumberOfStrings(string,substring,count,++index,times);
    }
}
//Question 1 Helper
public static boolean equals(String str1,String str2,int index){
    int j,count=0;
    for(j=0;j<str2.length();++j){
        if(str1.charAt(j+index)==str2.charAt(j)){
            ++count;
        }
    }
    if(count==str2.length()){
        return true;
    }else{ return false;}
}
```

Solution:



1.) $T(n) = T(n-1) + n \quad n > 0 \quad T(0) = 0$

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

$T(n) = T(n-2) + n-1 + n$ 2-step

$T(n-2) = T(n-3) + n-2$

$T(n) = T(n-3) + n-2 + n-1 + n$ 3.step

$T(n) = T(n-k) + n-k+1 + n-k+2 \ \ldots$

Set assume $n-k=0 \quad n=k$

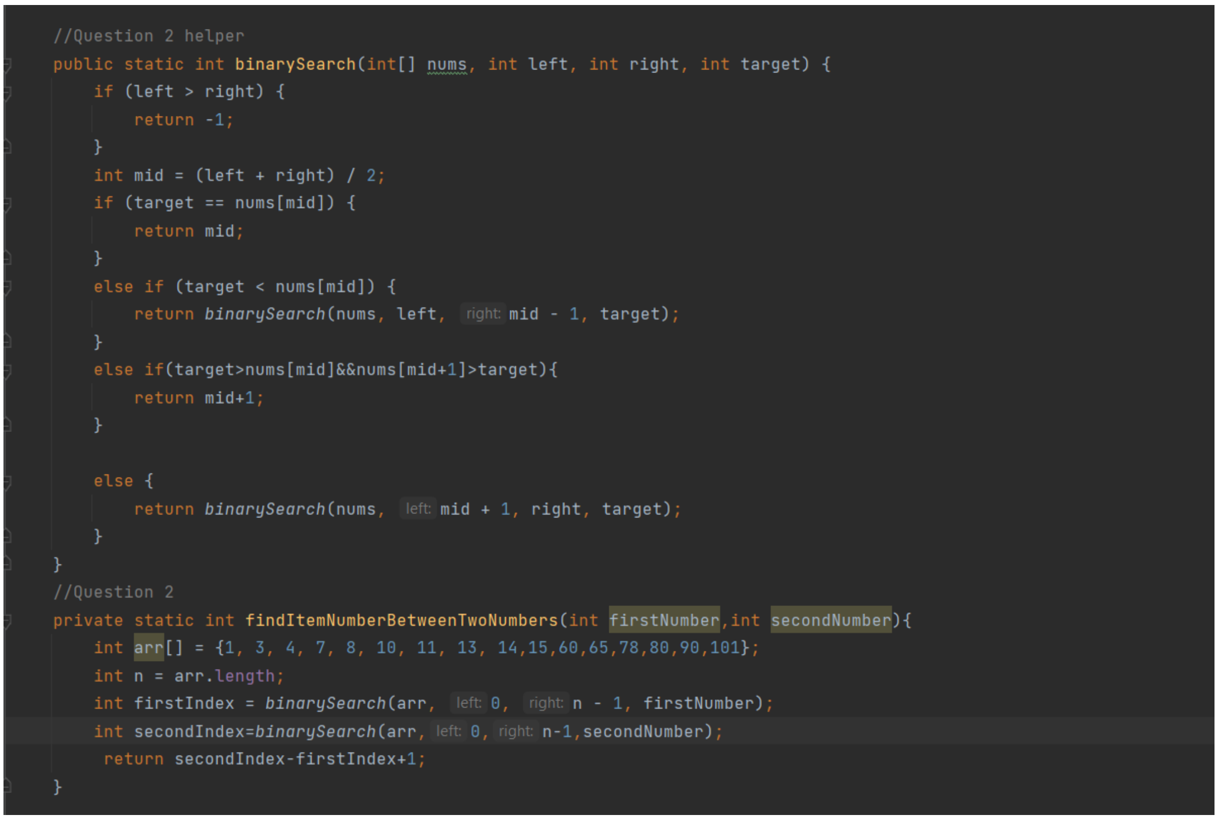
$T(n) = T(0) + 1+2+3+ \ldots n$

$T(n) = \frac{n.(n+1)}{2} \qquad T(n) = O(n^2)$

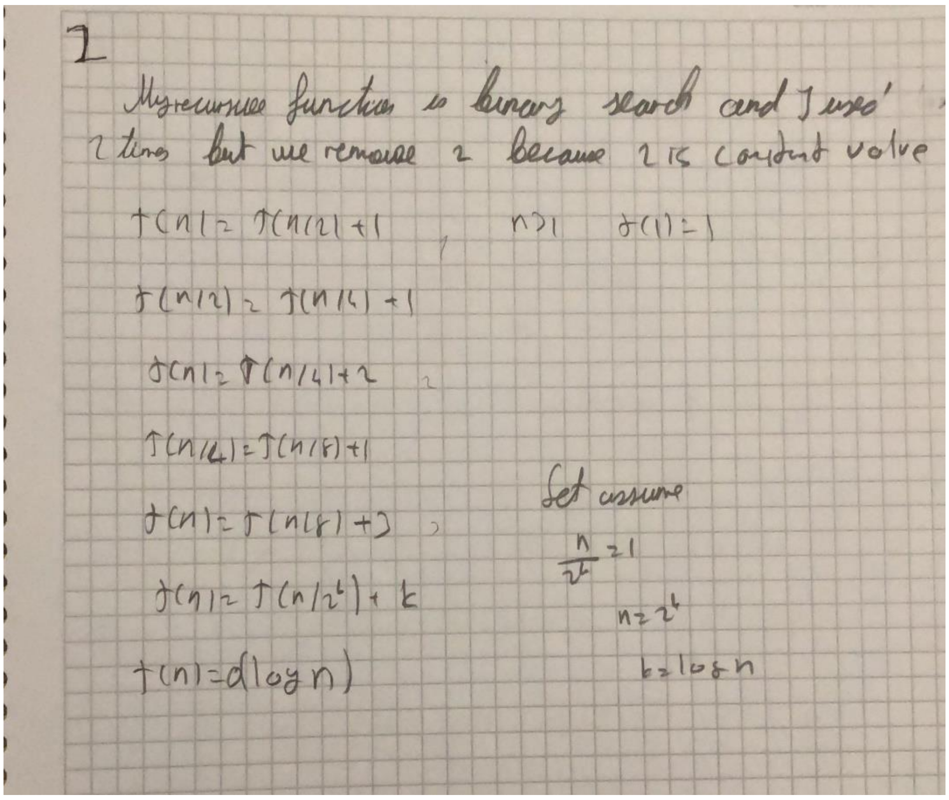
Question 2

Suppose that you are given a sorted integer array. Suggest a recursive algorithm to find the number of items in the array between two given integer values. Hint: Consider an approach like binary search algorithm; compare given two integers with the middle element.

This time I called a recursive function 2 times one of them finds the smallest number the other one finds the biggest number. After that , find difference between them.



```java
//Question 2 helper
public static int binarySearch(int[] nums, int left, int right, int target) {
    if (left > right) {
        return -1;
    }
    int mid = (left + right) / 2;
    if (target == nums[mid]) {
        return mid;
    }
    else if (target < nums[mid]) {
        return binarySearch(nums, left,  right: mid - 1, target);
    }
    else if(target>nums[mid]&&nums[mid+1]>target){
        return mid+1;
    }

    else {
        return binarySearch(nums,  left: mid + 1, right, target);
    }
}
//Question 2
private static int findItemNumberBetweenTwoNumbers(int firstNumber,int secondNumber){
    int arr[] = {1, 3, 4, 7, 8, 10, 11, 13, 14,15,60,65,78,80,90,101};
    int n = arr.length;
    int firstIndex = binarySearch(arr,  left: 0,  right: n - 1, firstNumber);
    int secondIndex=binarySearch(arr,  left: 0,  right: n-1,secondNumber);
     return secondIndex-firstIndex+1;
}
```



2

My recursive function is binary search and I used 2 times but we remove 2 because 2 is constant value

$T(n) = T(n/2) + 1$, $n>1$ $T(1) = 1$

$T(n/2) = T(n/4) + 1$

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

$T(n/4) = T(n/8) + 1$

$T(n) = T(n/8) + 3$ 3

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

$T(n) = O(\log n)$

Let assume

$\frac{n}{2^k} = 1$

$n = 2^k$

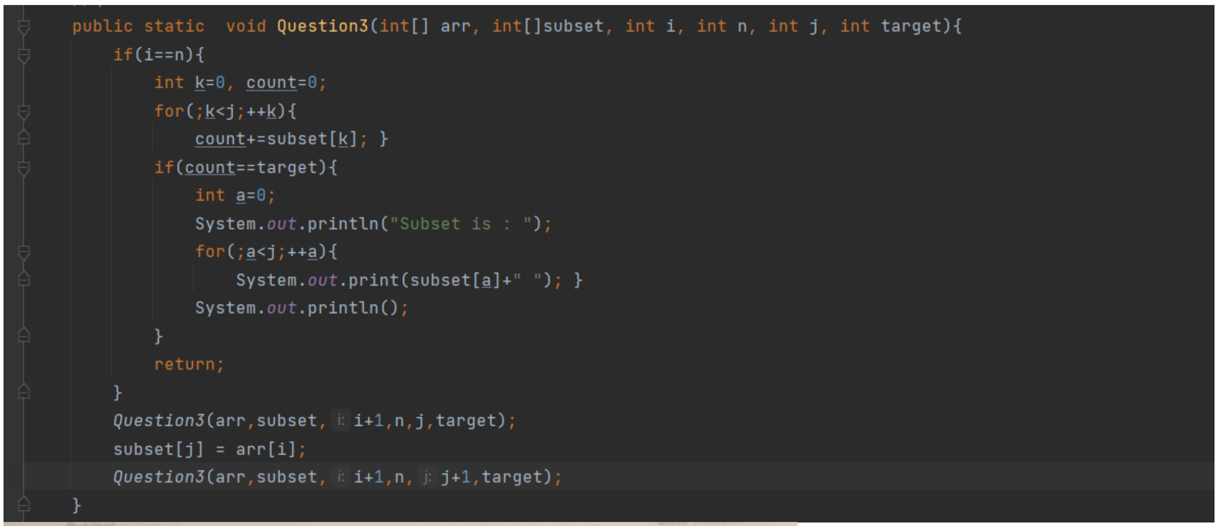
$k = \log n$

Question 3

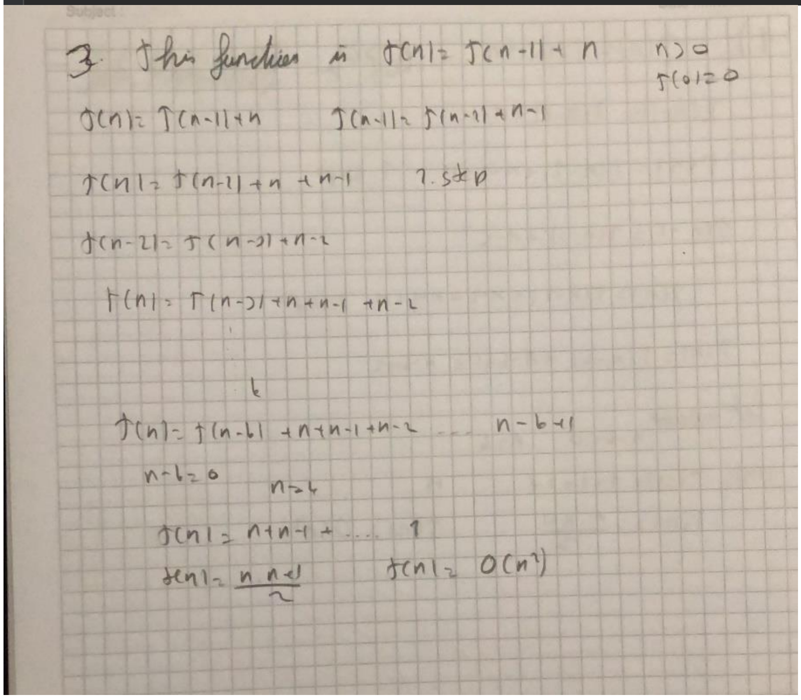
The question 3 is that:

Suppose that you are given an unsorted integer array. Propose a recursive solution to find contiguous subarray/s that the sum of its/theirs items is equal to a given integer value. Hint: Consider the following approach; the first element can be a part of the contiguous subarray or not. You could perform two recursive calls based on these two cases.

In this part, I try to find all subset and after that something is available to our condition I print them.



```java
public static  void Question3(int[] arr, int[]subset, int i, int n, int j, int target){
    if(i==n){
        int k=0, count=0;
        for(;k<j;++k){
            count+=subset[k]; }
        if(count==target){
            int a=0;
            System.out.println("Subset is : ");
            for(;a<j;++a){
                System.out.print(subset[a]+" "); }
            System.out.println();
        }
        return;
    }
    Question3(arr,subset, i+1,n,j,target);
    subset[j] = arr[i];
    Question3(arr,subset, i+1,n, j+1,target);
}
```



3. This function is $T(n) = T(n-1) + n$ $\quad n>0$

$T(0)=0$

$T(n) = T(n-1)+n \qquad T(n-1) = T(n-2) + n-1$

$T(n) = T(n-2) + n + n-1 \qquad$ 1. step

$T(n-2) = T(n-3) + n-2$

$T(n) = T(n-3) + n + n-1 + n-2$

k

$T(n) = T(n-k) + n + n-1 + n-2 \dots \quad n-k+1$

$n-k=0$

$n=k$

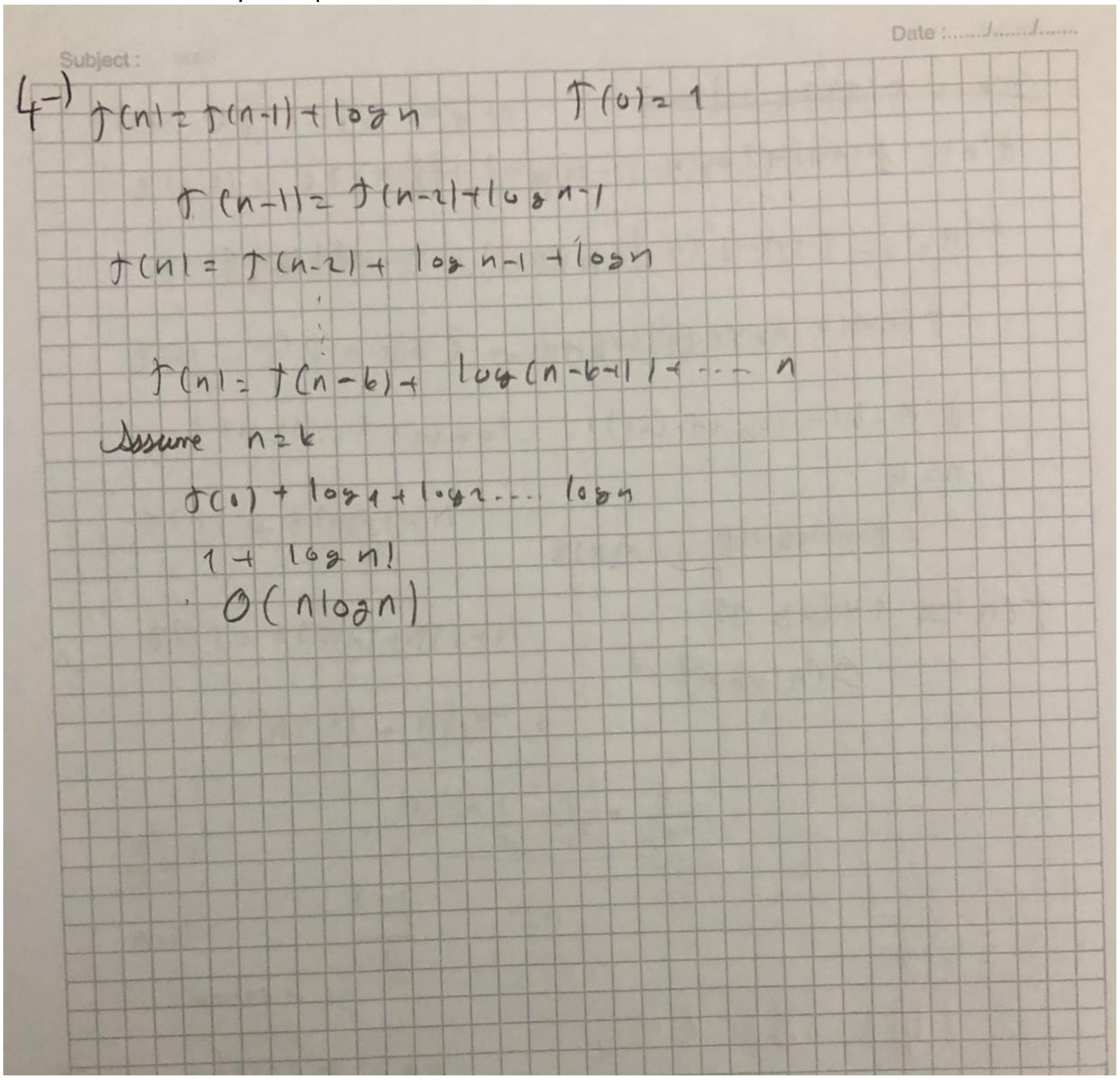
$T(n) = n + n-1 + \dots 1$

$T(n) = \frac{n(n+1)}{2} \qquad T(n) = O(n^2)$

Question 4

The fourth question is that

Explain the output of the given recursive function and how it works in detail. Analyze the time complexity of this function by analyzing the number of multiplication operations (which is the basic operation) at the base case.
Question 4 is actually multiplication



Subject :

Date :....../....../......

4-) $T(n) = T(n-1) + \log n$ $T(0) = 1$

$T(n-1) = T(n-2) + \log n-1$

$T(n) = T(n-2) + \log n-1 + \log n$

$\vdots$

$T(n) = T(n-k) + \log(n-k-1) + \dots n$

Assume $n = k$

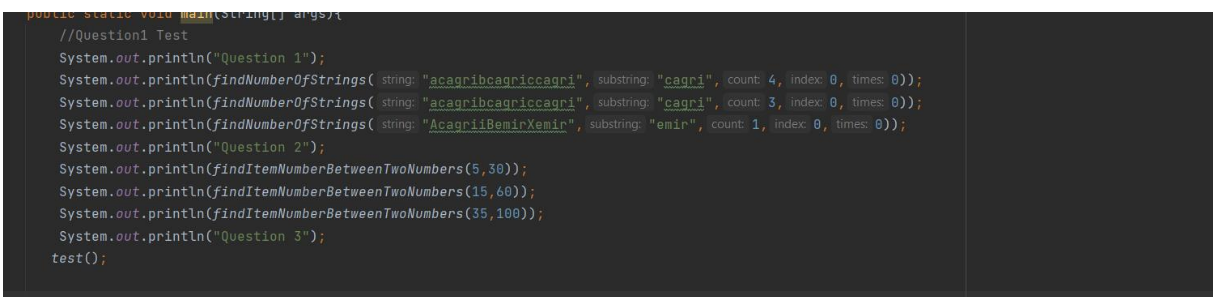
$T(0) + \log 1 + \log 2 \dots \log n$

$1 + \log n!$

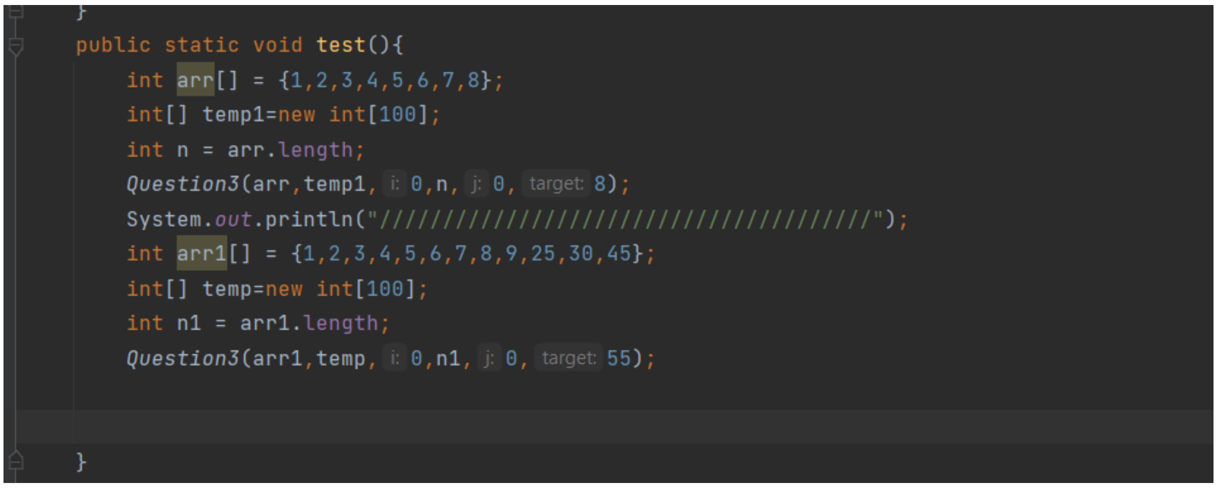
$O(n \log n)$

Test Cases and Results

Here is the first 3 questions

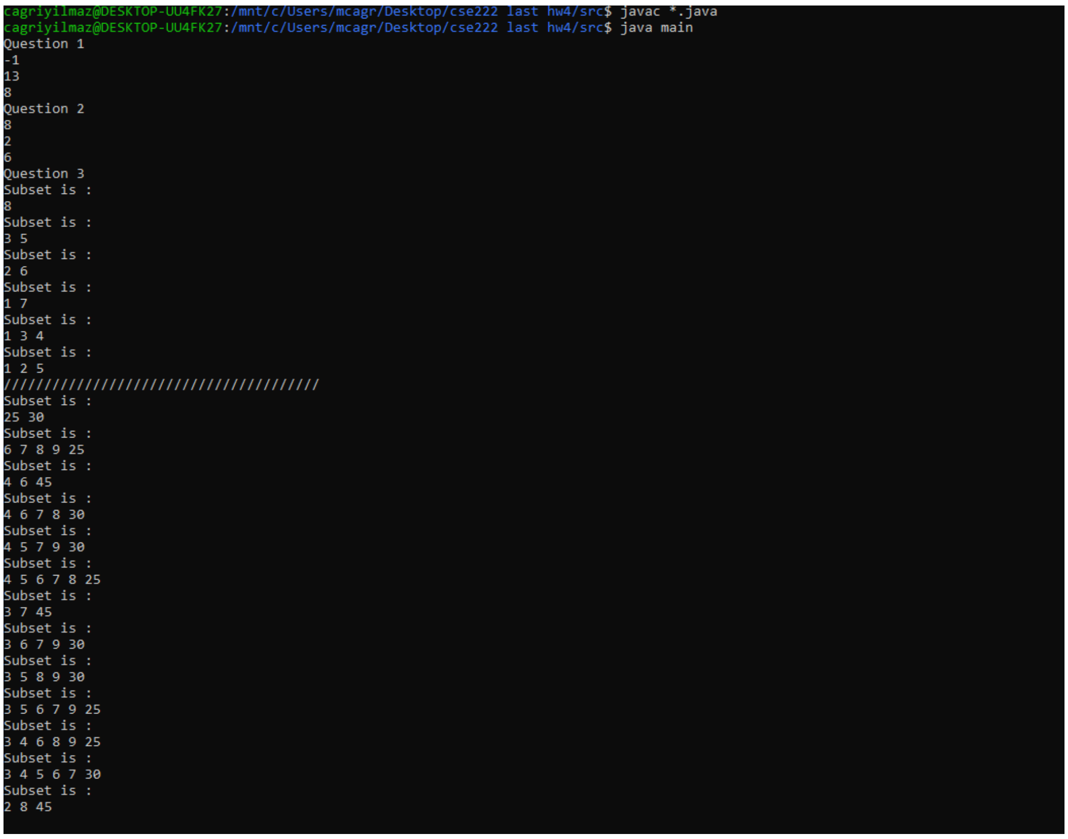


```
public static void main(String[] args){
    //Question1 Test
    System.out.println("Question 1");
    System.out.println(findNumberOfStrings( string: "acagribcagriccagri", substring: "cagri", count: 4, index: 0, times: 0));
    System.out.println(findNumberOfStrings( string: "acagribcagriccagri", substring: "cagri", count: 3, index: 0, times: 0));
    System.out.println(findNumberOfStrings( string: "AcagriiBemirXemir", substring: "emir", count: 1, index: 0, times: 0));
    System.out.println("Question 2");
    System.out.println(findItemNumberBetweenTwoNumbers(5,30));
    System.out.println(findItemNumberBetweenTwoNumbers(15,60));
    System.out.println(findItemNumberBetweenTwoNumbers(35,100));
    System.out.println("Question 3");
   test();
```



```
    }
    public static void test(){
        int arr[] = {1,2,3,4,5,6,7,8};
        int[] temp1=new int[100];
        int n = arr.length;
        Question3(arr,temp1, i: 0,n, j: 0, target: 8);
        System.out.println("///////////////////////////////////////");
        int arr1[] = {1,2,3,4,5,6,7,8,9,25,30,45};
        int[] temp=new int[100];
        int n1 = arr1.length;
        Question3(arr1,temp, i: 0,n1, j: 0, target: 55);

    }
```

Test Results



```
cagriyilmaz@DESKTOP-UU4FK27:/mnt/c/Users/mcagr/Desktop/cse222 last hw4/src$ javac *.java
cagriyilmaz@DESKTOP-UU4FK27:/mnt/c/Users/mcagr/Desktop/cse222 last hw4/src$ java main
Question 1
-1
13
8
Question 2
8
2
6
Question 3
Subset is :
8
Subset is :
3 5
Subset is :
2 6
Subset is :
1 7
Subset is :
1 3 4
Subset is :
1 2 5
///////////////////////////////////////
Subset is :
25 30
Subset is :
6 7 8 9 25
Subset is :
4 6 45
Subset is :
4 6 7 8 30
Subset is :
4 5 7 9 30
Subset is :
4 5 6 7 8 25
Subset is :
3 7 45
Subset is :
3 6 7 9 30
Subset is :
3 5 8 9 30
Subset is :
3 5 6 7 9 25
Subset is :
3 4 6 8 9 25
Subset is :
3 4 5 6 7 30
Subset is :
2 8 45
```