GEBZE TECHNICAL UNIVERSITY
COMPUTER ENGINEERING

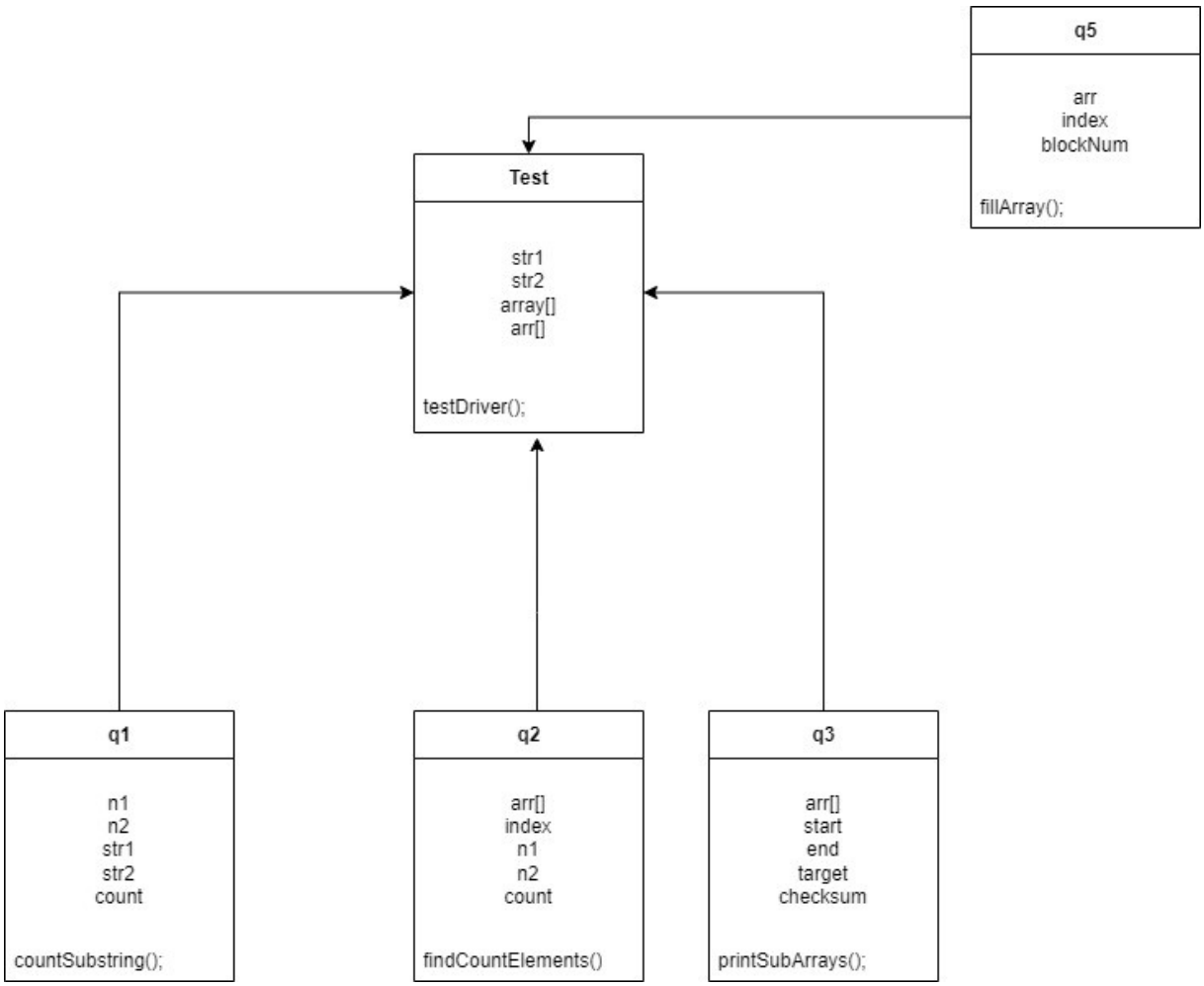
CSE 222
HOMEWORK 4 REPORT

ERAY ALÇİN
1801042687

1. SYSTEM REQUIREMENTS

As a system requirement, we need to know recursive functions. We need to do time complexity analysis. We need to prove our solutions with the induction method. So first we need to find our formula using recurrence relation and then we need to prove it with induction method.

2. CLASS DIAGRAM



3. PROBLEM SOLUTION APPROACH

I determined what I needed to do in the system requirements and created the algorithms. I did it by using recursive methods while solving the problem. I learned the use of recursive functions and learned for what purpose they were written. I found the time complexity of the programs I wrote.

4. TEST

```
------------TESTING QUESTION 1 --------------
Index :6
Index :5
------------TESTING QUESTION 2 --------------
Count  :13
Count  :2
Count  :7
------------TESTING QUESTION 3 --------------
[2,3][5]
[3,4]
[2,1][3]

------------TESTING QUESTION 5 --------------
1110000000
1110111000
1111000000
1111011110
1111100000
1111110000
1111111000
1111111100
1111111110
1111111111
```

```
     * @param args
     */
    public static void testDriver()
    {
        System.out.println("--------------TESTING QUESTION 1 ----------------");
        String str1 = "erayalcin123computer",
        str2 = "com";
        System.out.println("Index :"+q1.countSubstrig(str1,str2,0));

        String str3 = "denemedeneme123eraydenemeeray",
        str4 = "eray";
        System.out.println("Index :"+q1.countSubstrig(str3,str4,0));


        System.out.println("--------------TESTING QUESTION 2 ----------------");
        int array[]={-15,-7,-6,-3,-1,0,1,2,3,4,5,6,7,8,9,10};
        System.out.println("Count  :"+q2.findCountElements(array,0,-7,10,0));


        int array1[]={-15,-7,-6,-3,-1,0,1,2,3,4,5,6,7,8,9,10};
        System.out.println("Count  :"+q2.findCountElements(array1,0,5,8,0));


        int array2[]={-15,-7,-6,-3,-1,0,1,2,3,4,5,6,7,8,9,10};
        System.out.println("Count  :"+q2.findCountElements(array2,0,-2,6,0));
        System.out.println("--------------TESTING QUESTION 3 ----------------");
        int target=5;
        int []arr = {0,1, 2, 3,4,5,6};
        q3.printSubArrays(arr, 0, 0,target);
        System.out.println();
        int target1=7;
        int []arr1 = {0,2, 1, 3,4};
        q3.printSubArrays(arr1, 0, 0,target1);
        System.out.println();
        int target2=3;
        int []arr2 = {2,1,3,4,0};
        q3.printSubArrays(arr2, 0, 0,target2);
        System.out.println();


        System.out.println("\n--------------TESTING QUESTION 5 ----------------");


        int array3[]={0,0,0,0,0,0,0,0,0,0};
        for(int i=3;i<=array.length;i++)
        {
            q5.fillArray(array3,0,i);
        }
    }
    public static void main(String[] args) {
        testDriver();


    }
```

Q1

```java
public class q1 {

    /**
     * @str1 bigger string
     * @str2 search word
     * @count that that holds the index we will return.
     */
    static int countSubstrig(String str1,String str2,int count)
    {
        int n1 = str1.length();
        int n2 = str2.length();

        /**
         * Base Case
         */
        if (n1 == 0 || n1 < n2)
            return -1;
        /**
         *  Recursive Case
         *  Checking if the first
         *  substring matches
         */
        if (str1.substring(0, n2).equals(str2))
            return count;
        /**
         *  Otherwise, return the count
         *  from the remaining index
         */

        return countSubstrig(str1.substring(n2 - 1),str2,count+1);
    }


}
```

Q2

```java
public class q2 {

    /**
     * @arr array
     * @index array's index
     * @n1 Values range
     * @n2 Values Range
     * @count number of items between n1 and n2
     */


    static int  findCountElements(int arr[],int index,int n1,int n2,int count)
    {
        /**
         * Base Case
         */
        if(arr.length < index || index < 0)
        {
            return -1;
        }
        /**
         * Recursive Case
         */
        if(arr[index]==n1 || (arr[index]<n1 && arr[index+1]>n1)) return findCountElements(arr,index+1,n1,n2,0);

        if(arr[index]==n2) return count;

        return findCountElements(arr,index+1,n1,n2,count+1);

    }

}
```

Q3

```
/**
 * base case
 * Stop if we have reached the end of the array
 */
if (end == arr.length) O(1)
    return;
/**
 * Recursive case
 * Increment the end point and start from 0
 */
else if (start > end)
    printSubArrays(arr, 0, end + 1,target);
/**
 * Recursive case
 * Print the subarray and increment the starting point
 */
else
{
     /**
      * It collects the data between the start index and the end index and then checks
      * if it is equal to the target and if it is, it prints that start and end subarray to the screen
      */


    int checksum=0; O(1)
    for (int i = start; i <= end; i++){ } O(n)
        checksum+=arr[i]; O(1)


    }

    if(checksum==target)
    {
        System.out.print("["); O(1)
        for (int i = start; i <= end; i++){
            if(i==end) System.out.print(arr[i]) O(1)
            else System.out.print(arr[i]+","); O(1) } O(n)


        }

        System.out.print("]"); O(1)
    }

    printSubArrays(arr, start + 1, end,target);
}

return;
```

Q5

```
static public void fillArray(int arr[],int index,int blockNum)
{

    /**
     * Base Case
     */
    if(index+blockNum>arr.length || blockNum>arr.length) O(1)
    {
        return; O(1)
    }
    /**
     * Recursive Case
     * The length of the block to be added and the index should not exceed the length of the array.
     */
    if(index+blockNum<=arr.length)
    {

        for(int i=0;i<blockNum;i++) } O(n)
        {
            arr[index+i]=1;
        }
        for(int i=0;i<arr.length;i++) } O(n)
        {
            System.out.print(arr[i]);
        }
        System.out.println(); O(1)

        fillArray(arr,index+1+blockNum,blockNum);

    }
    /**
     * Array cleaning
     */
    for(int i=0;i<arr.length;i++) } O(n)
    {
        arr[i]=0;
    }
```