GIT Department of Computer Engineering

CSE 222/505 - Spring 2022

Homework 4 Report

Süleyman Burak Yaşar

1901042662

Q1)

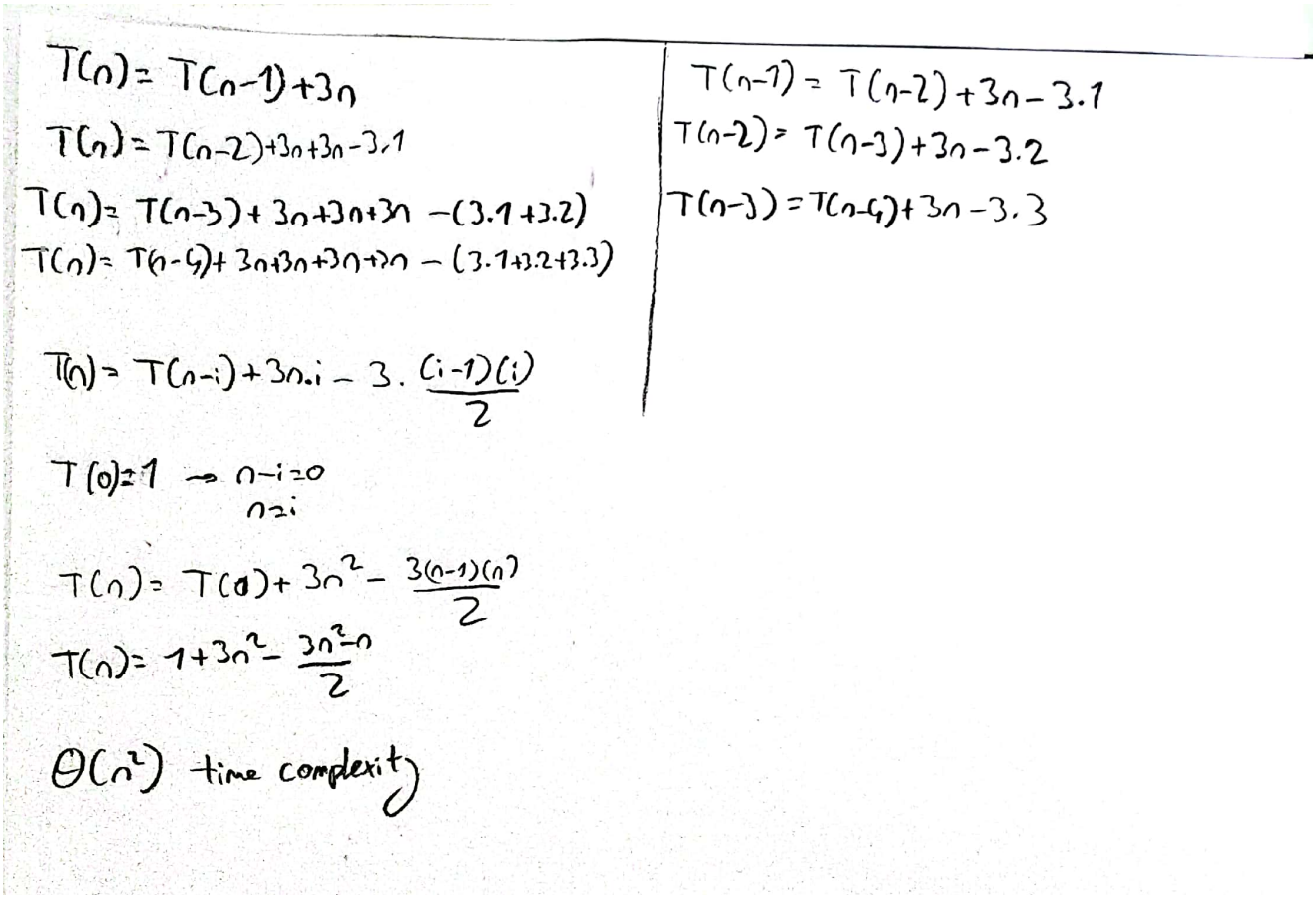
```
private static int searchSub(String small,String big,int index,int n){

    if(big.length()<small.length()){
        return -1;  θ(1) time
    }
                        θ(n) time        θ(n) time              θ(1) time
    else if( small.equals( big.substring(0,small.length()) ) && index-- == 1){
        return n;
    }

    else                                        θ(n) time
        return searchSub(small,big.substring(1),index,n+1);
}
```

```
public static int searchSub(String small,String big,int index){
    return searchSub(small,big,index,0);
}
```

Best Case θ(1)



Q2)

```
private static double binarySearch(int[] array,int target,int first,int last,boolean isFirst){

    if(last <= first){

        int middle = (first+last)/2;

        if(array[middle] == target){
            if(isFirst){
                return middle-0.50;
            }

            else return middle+0.50;
        }
        if(array[middle]>=target)
            return middle - 0.50;
        else return middle + 0.50;
    }

    else{
        int middle = (first+last)/2;

        if(array[middle] == target){
            if(isFirst){
                return middle-0.50;
            }

            else return middle+0.50;
        }

        else if(array[middle]<target)
            return binarySearch(array,target,middle+1,last,isFirst);
        else
            return binarySearch(array,target,first,middle-1,isFirst);
    }

}
public static int numOfItemArrBetween(int[] array,int firstNumber,int secondNumber){
    return (int)(binarySearch(array,secondNumber,0,array.length-1,false) - binarySearch(array,firstNumber,0,array.length-1,true));
}
```

θ(1) time

θ(1) time

Best Case θ(1)

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

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

$T(n) = T(n/2^3) + 1 + 1 + 1$

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

$T(n) = T(n/2^i) + 1 \cdot i$

$T(1) = 1$

$\frac{n}{2^i} = 1 \rightarrow n = 2^i \rightarrow \log_2 n = i$

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

$T(n/2^2) = T(n/2^3) + 1$

$T(n/2^3) = T(n/2^4) + 1$

$T(n) = T(1) + \log_2 n$

$T(n) = 1 + \log_2 n$

$O(\log n)$ time complexity

Q3)

```java
public static int[] sumCheck(int[] array,int[] sub,int target,int index,int sum){

    if(index == array.length){
        return null;
    }

    else{

        sum +=array[index];

        int[] temp = Arrays.copyOf(sub,sub.length+1);

        temp[sub.length] = array[index];



        if(sum == target) return temp;

        else if(sum > target) return null;

        else return sumCheck(array,temp,target,index+1,sum);
    }
}
```

θ(1) time

θ(1) time

θ(n) time

θ(1) time

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

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

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

$T(n) = T(n-4) + n + n + n + n - n + 2 + 2 + 2 + 2 - (1+2+3)$

$T(n) = T(n-i) + i \cdot n + 2 \cdot i - \frac{(i-1)(i)}{2}$

$T(0) = 1 \rightarrow n = i$

$T(n) = T(0) + n^2 + 2n - \frac{n^2 - n}{2}$

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

$\Theta(n^2)$ time complexity

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

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

$T(n-3) = T(n-4) + n - 3 + 2$

```java
public static ArrayList<int[]> findSubArrays(int[] array,ArrayList<int[]> subs,int target,int n){

    if(n == array.length){
        return subs;
    }

    else{

        int[] tempSub = new int[0];
        tempSub = sumCheck(array,tempSub,target,n,0);

        if(tempSub !=null){
            subs.add(tempSub);
        }

        return findSubArrays(array,subs,target,n+1);

    }
}
```

θ(1) time

θ(1) time

θ(n^2) time

θ(1) time

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

$T(n) = T(n-2) + n^2 + n^2 - 1 + n + 2 \cdot 2 + 1$

$T(n) = T(n-3) + 3n^2 + 2(1+2)n + 3 \cdot 2 + 1^2 + 2^2$

$T(n) = T(n-4) + 4n^2 + 2 \cdot (1+2+3)n + 4 \cdot 2 + 1^2 + 2^2 + 3^2$

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

$T(n-2) = T(n-3) + n^2 + 4n + 4 + 2$

$T(n-3) = T(n-4) + n^2 + 6n + 9 + 2$

$$T(n) = T(n-i) + in^2 + 2 \cdot \frac{(i-1)(i)}{2} \cdot n + in + \frac{(i-1)(i)(2i-1)}{6}$$

$T(0) = 1 \rightarrow n = i \rightarrow$

$$T(n) = \underset{1}{T(0)} + n^3 + \frac{n^3 - n^2}{2} + n^2 + \frac{(n^2 - n)(2n-1)}{6}$$

$\Theta(n^3)$ time complexity

```java
public static ArrayList<int[]> findSubArrays(int[] array,int target){

    ArrayList<int[]> subs= new ArrayList<int[]>();

    return findSubArrays(array,subs,target,0);
}
```