Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 4 REPORT

EDA BAHRİOĞLU
131044055

Course Assistant:

INTRODUCTION

1 Problem Definition

The general purpose of this assignment is to write iterative functions as recursive.I find the maximum length sorted sublist in this list for first question.I find two subsets of an int array whose sums equal a given target in second question.I calculated the running time for third question and I analized recurrence relation for fourth question.Finally questions a program is a matrix action.

2 System Requirements

For the last part of this project, intelig ide is used. And this program is tested in Java 11.2.
In this point for this project is writed a matrixRecursion function. Of This function purpose tasked with coding an iterator class for these data that will traverse a given 2D array spirally clockwise starting at the top left element.

```
①(a) public List<int> maxLength Sublist ( Node head)
{     int length = 1 , max = 1;
      List<int> List ;
      int count = 1 ;
      int max-index = 0;
      int index ;
      Node x = head;
      while ( x.next != null)
      {   if ( x.next.data > x.data)
              {   length ++;
              }
          else
          {   if (Length > max)
                  {  max = length;
                     max-index = count - length ;
                  }
                  Length = 1 ;
          }
            count ++ ;
            x = x.next ;
      }
      if (Length > max)
      {    max = length;
           max.index = count - max ;
      }
      index=0;
      Node y = head ;
      While (y != null)
      {  if (index == max-index)
          {  while (max > 0)
               {  List . add.( y.data);
                  y = y.next;
                  max -- ;
               }
               break;
          }
          y = y.next;
          index + +;
      }
   return List ;
}
```

O(n)

O(n)

O(n) = n^2

$T(n) = \boxed{n^2} + n$ → biggest root

$T(n) = O(n^2)$

① b.)

```
public static int maxSublist(int head, List<Integer> tail, int ListSize)
{
    if (tail == null)
      return 0;
    else if (taill.size() == 0)
      return List.Size;
    if (head <= tail.get(0))
    {
        ListSize ++;
    }
    else
    {   ListSize = 1;
    }
    int next = maxSublist (tail).get(0), tail.sublist (1, tail.size()), ListSize);
    int max = Math.max (ListSize, next);
    if (tail.sublist (0, tail.size() -1). size() == max)
       System.out.println (tail).sublist (0, tail.size() - 1);
    return max;
}
```

Loop will return until to size

size = n

So this code complexity is $O(n)$ //

① Complexity Analysis for Master Teorem

$T(n) = a\,T(n/b) + f(n)$

$T(n) = T(n)$

$a = 1 \quad b = 1 \quad d = 0$

$T(n) = \Theta(n^d \log n)$ if $a = b^d \rightarrow$ the equal provided.

$1 = 1$

$T(n) = \Theta(n \log n)$ //

② for Induction method

$T(n) = T(n) \qquad f(n) = 0$

$0 \leq O(n) \qquad c \geq 1$

$0 \leq c(n) \qquad n \geq 1$

$0 \leq n$

for $n = 1$

$0 \leq 1$ ✓

for $n = k$

$0 \leq k$ accepted true

for $n = k + 1$

$0 \leq k + 1$ ✓ it is true

```
② public static void sumOfNumber( List<Integer> List , int [] array,
                                   int sumTarget , int index)
{
    if (sumTarget <0)
      System.out.Println(" Wrong value ");
    else if( sumTarget == 0)
    { if(list.size() == 2)
        { System.out.Println(List);
        }
    }
    else if (index > array.length -1 || index<0)
      System.out.Println(" Wrong index ");
    for(int j=index; j<array.length ; j++)
array length
    {    List.add (array [j];
         sumOfNumber( List, array , sum - array[j] , j+1 );
         List remove (new Integer (array [j ]));
    }
}
```

$T(n) = O(\text{arraylength})$;

$n = \text{array.length}$.

$T(n) = O(n)$;

```
(3) for(i=2*n; i>1; i=i-1)        -> O(2n)   }+ O(n^2)
      for(j=1; j<=i; j=j+1)       -> O(2n)   }            ) +
        for(k=1; k<=j; k=k*3)     -> O(logn)    O(logn)   )
          print("hello");         -> O(1)
```

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

```
(4) void afunc(myArray, n) {  ------> T(n)
  for(i=0; i<=(n/2)-1; i++)
  { for (j=0; i<(n/2)-1; i++){
      for(j=0; j<=(n/2)-1; j++){
                                    -> (n/2-1)^2 -> O(n^2)
      }
  }
      x1=afunc(myArray1, n/2); -> T(n/2) }
      x2=afunc(myArray2, n/2); -> T(n/2) }
      x3=afunc(myArray3, n/2); -> T(n/2) }   4T(n/2)
      x4=afunc(myArray4, n/2); -> T(n/2) }
```

$T(n) = 4T(n/2) + \underbrace{((n/2)-1)^2} \rightarrow n^2$ (coefficients and constants not important)

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

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

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

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

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

$\boxed{T(n) = 4^2T(n/2^2) + 2n^2}$

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

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

$\boxed{T(n) = 4^3T(n/2^3) + 3n^2}$

$$\begin{cases} 1 & n=1 \\ T(n)=4T(n/2)+n^2 & n>1 \end{cases}$$

$T(n) = 4^kT(n/2^k) + kn^2$

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

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

$T(n) = 4^k(T1) + kn^2$ → coefficient and constants not imp.

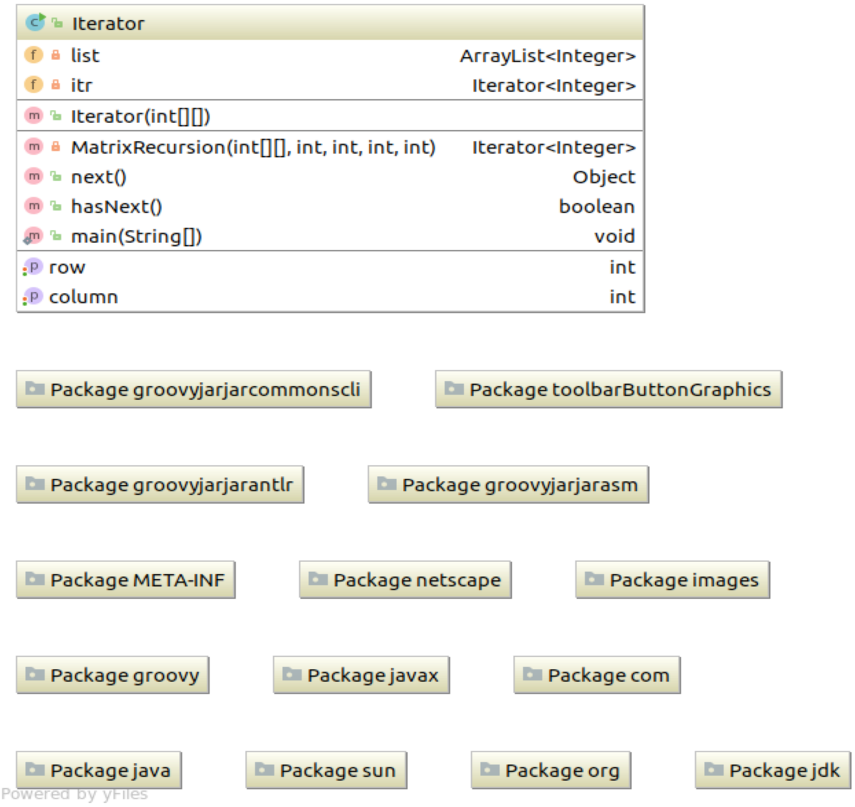
$n^2 \times 1 + \boxed{n^2 \log n}$

$\boxed{T(n) = O(n^2 \log n)}$ //

METHOD

1 Class Diagrams

For 5. question



Iterator	
list	ArrayList<Integer>
itr	Iterator<Integer>
Iterator(int[][])	
MatrixRecursion(int[][], int, int, int, int)	Iterator<Integer>
next()	Object
hasNext()	boolean
main(String[])	void
row	int
column	int

2 Use Case Diagrams

3 Problem Solution Approach

1 Recursion is often used for this assignment . Complexity is Computed for each question separately.The first part of the first question returns the sublist as iterative. Second part of the first question does not return the list , but prints the largest sub-list to the screen as recursion. The complexity analysis was performed according to the Master theorem and induction. In question 2, recursion was used because of complexity O(n).For the third question, the recurrence relation approach is used.

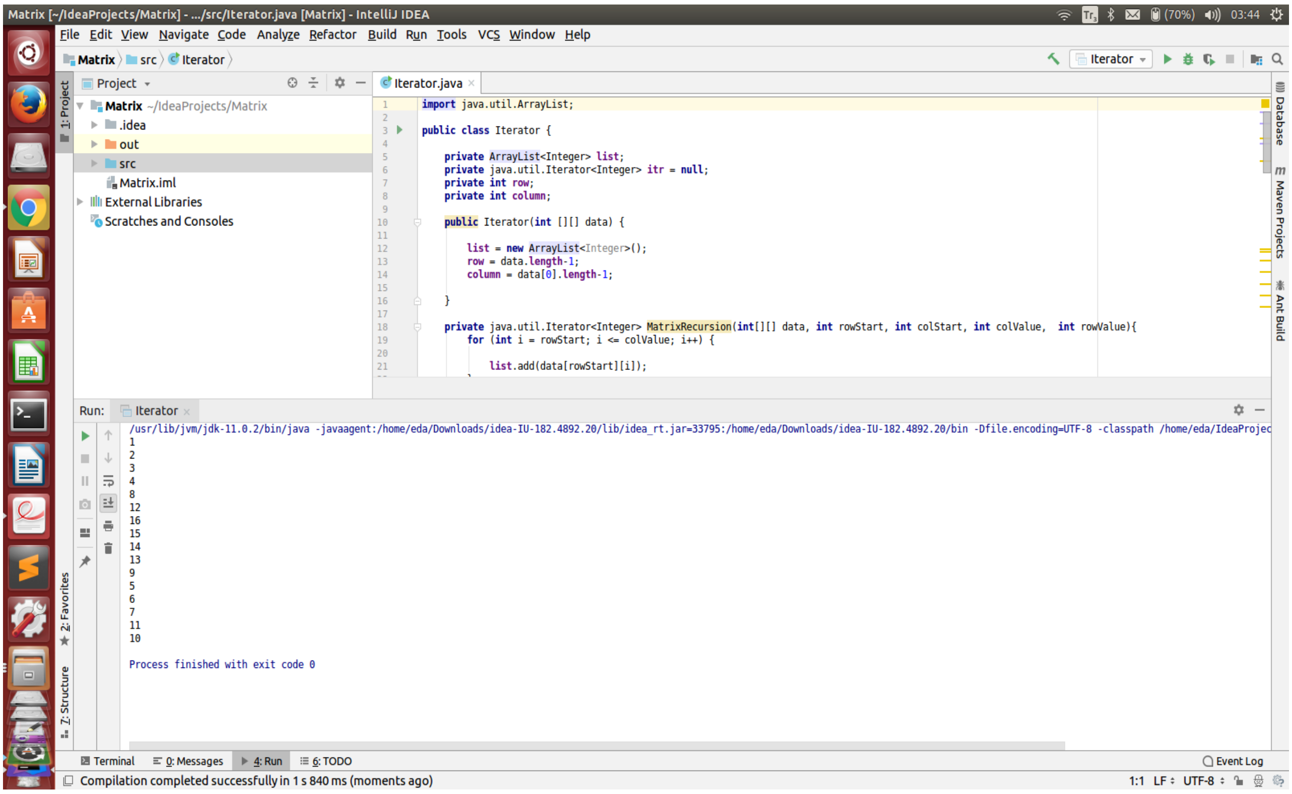
RESULT

1 Test Cases

This program is tested according java 11.2 in Intellig ide.

2 Running Results

For 5. question.



```java
import java.util.ArrayList;

public class Iterator {

    private ArrayList<Integer> list;
    private java.util.Iterator<Integer> itr = null;
    private int row;
    private int column;

    public Iterator(int [][] data) {

        list = new ArrayList<Integer>();
        row = data.length-1;
        column = data[0].length-1;

    }

    private java.util.Iterator<Integer> MatrixRecursion(int[][] data, int rowStart, int colStart, int colValue,  int rowValue){
        for (int i = rowStart; i <= colValue; i++) {

            list.add(data[rowStart][i]);
```

```
/usr/lib/jvm/jdk-11.0.2/bin/java -javaagent:/home/eda/Downloads/idea-IU-182.4892.20/lib/idea_rt.jar=33795:/home/eda/Downloads/idea-IU-182.4892.20/bin -Dfile.encoding=UTF-8 -classpath /home/eda/IdeaProjec
1
2
3
4
8
12
16
15
14
13
9
5
6
7
11
10

Process finished with exit code 0
```

Compilation completed successfully in 1 s 840 ms (moments ago)

1:1 LF UTF-8