LAB 07 - GRAPH

EC 4070

DATA STRUCTURES AND ALGORITHMS

JAYASINGHE PA

2020/E/058

GROUP B15

SEMESTER 4

18 APRIL 2023

PROBLEM SPECIFICATION

The user is trapped in a jungle and cannot find the way out. Divise an algorithm that is guaranteed to find the way out to go out of the jungle as fast as you can before it gets dark. The input starts with a number N and then the matrix of size N x N filled with S, E, T, and P which is our map. Map contains a single S representing the start point, and single E representing the end point and P representing the path and T representing the Tree.

IMPLEMENTATION

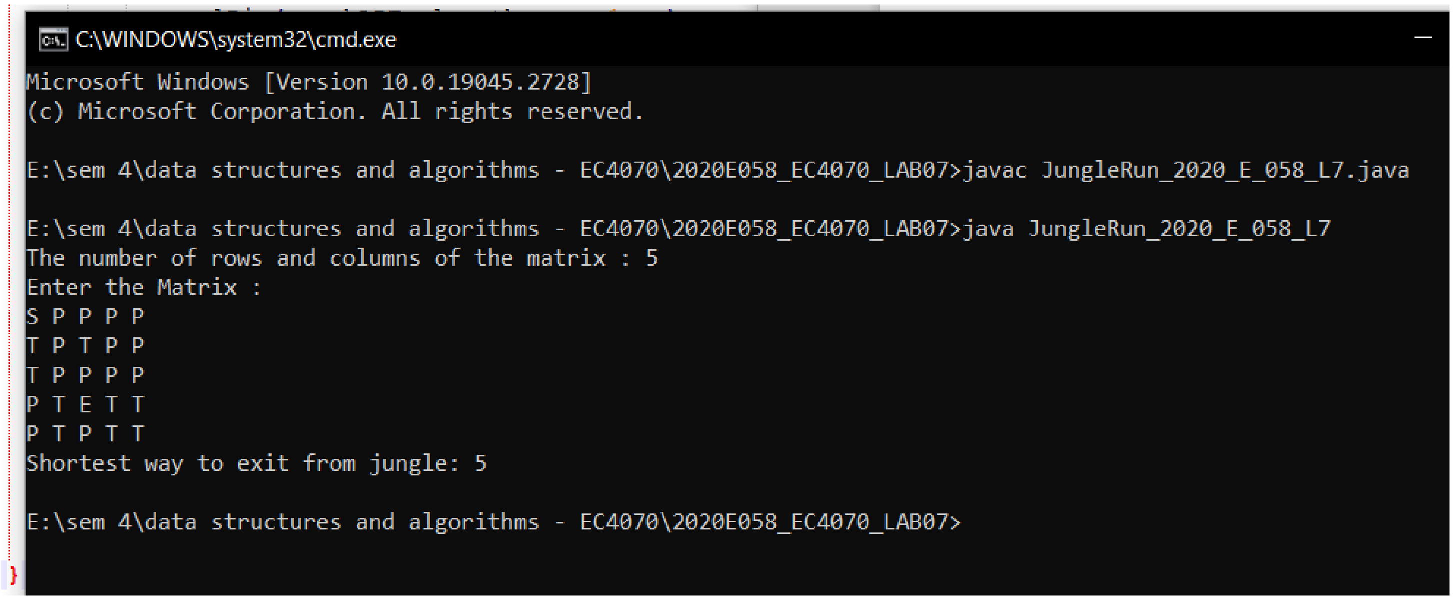
```
import java.util.*;
import java.util.Scanner;

public class JungleRun_2020_E_058_L7
{
  public static int firstRow;
  public static int firstcoloumn;;
  public static int Distance = 30*30;


  static class Node // Node class is created
  {
        int r; // r= row
        int c; // c= coloumns
        int dis; // dis = distance
        String k; // k= key

        Node(String k,int r,int c ) // constructor
        {
                this.k=k;
                this.r=r;
                this.c=c;
                dis = 900;
        }

        public void setDis(int dis)
        {
                this.dis= dis;

        }
```

```
            public int getDis() //function for getting the distance
            {
                        return dis;
            }
            public String getK()//function for getting the key
            {
                        return k;
            }
            public int getR()//function for getting the rows
            {
                        return r;
            }
            public int getC()//function for getting the coloumns
            {
                        return c;
            }
   }

          public static void searchStart(LinkedList<LinkedList<Node>>
graphADT, int size) // function which shows the
          {
      for (int i = 0; i < size; i++) {

       for (int j = 0; j < size; j++) {

            if (graphADT.get(i).get(j).getK().equals("S")) {
                firstRow = i;
                firstcoloumn = j;
                graphADT.get(firstRow).get(firstcoloumn).setDis(0);
                break;
             }
          }
       }
    }


    public static void calDis(LinkedList<LinkedList<Node>> graphADT,
int length, int r, int c)
          {

       if ((r < graphADT.size()) && (r > -1) && (c < graphADT.size()) &&
(c > -1)){
```

```
            switch (graphADT.get(r).get(c).getK()) {
                case "E":
                    if (Distance > length + 1) {
                        Distance = length + 1;
                        graphADT.get(r).get(c).setDis(Distance);
                    }
                    break;

                                        case "S":
                    calDis(graphADT, length, r + 1, c);
                    calDis(graphADT, length, r - 1, c);
                    calDis(graphADT, length, r, c + 1);
                    calDis(graphADT, length, r, c - 1);
                    break;

                case "P":
                    if (graphADT.get(r).get(c).getDis() == 900)
                                                  {
                        length = length + 1;
                        graphADT.get(r).get(c).setDis(length);
                        calDis(graphADT, length, r + 1, c);
                        calDis(graphADT, length, r - 1, c);
                        calDis(graphADT, length, r, c + 1);
                        calDis(graphADT, length, r, c - 1);
                    }
                    break;

                case "T":
                    break;
            }
        }
    }

public static void main (String args[])
        {

                    LinkedList<LinkedList<Node>> graphADT = new
LinkedList<>();

                    // getting the matrix data from the user
                    System.out.print("The number of rows and columns
of the matrix : ");
```

```
                    Scanner myobj = new Scanner(System.in);
                    int matrix_size = myobj.nextInt();

                    System.out.println("Enter the Matrix : ");

                    for (int i=0; i<matrix_size; i++)
                    {
                              graphADT.add(new LinkedList<>()); // adding
new linked list
                              for (int j=0; j<matrix_size; j++)
                              {
                                        graphADT.get(i).add(new
Node(myobj.next(),i,j)); // adding new node to the linked list
created
                              }
                    }

                    // finding size of linkedlist
                    int size=graphADT.size();

                    // findind the place which we should start in the
matrix
                    searchStart(graphADT, size);

                    calDis(graphADT, 0, firstRow, firstcoloumn);


                    if (Distance != 30*30)
                    {
          System.out.println("Shortest way to exit from jungle: " +
Distance );
      }
      else
                    {
          System.out.println("You can't escape from the jungle");
                    }


         }
}
```

OUTPUT SCREENSHOT



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.2728]
(c) Microsoft Corporation. All rights reserved.

E:\sem 4\data structures and algorithms - EC4070\2020E058_EC4070_LAB07>javac JungleRun_2020_E_058_L7.java

E:\sem 4\data structures and algorithms - EC4070\2020E058_EC4070_LAB07>java JungleRun_2020_E_058_L7
The number of rows and columns of the matrix : 5
Enter the Matrix :
S P P P P
T P T P P
T P P P P
P T E T T
P T P T T
Shortest way to exit from jungle: 5

E:\sem 4\data structures and algorithms - EC4070\2020E058_EC4070_LAB07>
```

CONCLUSION

- In this laboratory the main aim was to implement graph in the given tasks.h
- The given task was a bit challenging which required to go through articles about graph.
- The size of the map should be considered while taling the size of the matrixes.