EC4070: Data Structures and Algorithms

LAB 07

K.J.M.U.G.S. Eranda Jayasinghe

2021/E/075

SEMESTER 4

EC4070

15.11.2023

Q1.

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

public class JungleRun
{
    public static int startR;
    public static  int startC;
    public static int SD=30*30;

    static class Node
    {
        int row;
        int column;
        int dis;
        String key;

        Node(String key, int row, int column)
        {
            this.key = key;
            this.row = row;
            this.column = column;
            dis = 1000;
        }

        public void setDistance(int dis)
        {
            this.dis = dis;
        }
        public int getDistance()
        {
            return dis;
        }
        public String getKey()
        {
            return key;
        }
        public int getRow()
        {
            return row;
        }
        public int getColumn()
        {
            return column;
        }

    }

    public static void findStart(LinkedList<LinkedList<Node>> graphADT, int size)
    {
        for (int i = 0; i < size; i++)
        {
            for (int j = 0; j < size; j++)
            {
                if (graphADT.get(i).get(j).getKey().equals("S"))
                {
```

```java
 55                         {
 56                             startR = i;
 57                             startC = j;
 58                             graphADT.get(startR).get(startC).setDistance(0);
 59                             break;
 60                         }
 61                     }
 62                 }
 63             }
 64
 65         public static void findDistance(LinkedList<LinkedList<Node>> graphADT, int length, int row, int column)
 66         {
 67
 68             if ((row < graphADT.size()) && (row > -1) && (column < graphADT.size()) && (column > -1))
 69             {
 70
 71                 switch (graphADT.get(row).get(column).getKey())
 72                 {
 73                     case "E":
 74                         if (SD > length + 1)
 75                         {
 76                             SD = length + 1;
 77                             graphADT.get(row).get(column).setDistance(SD);
 78                         }
 79                         break;
 80                     case "S":
 81                         findDistance(graphADT, length, row + 1, column);
 82                         findDistance(graphADT, length, row - 1, column);
 83                         findDistance(graphADT, length, row, column + 1);
 84                         findDistance(graphADT, length, row, column - 1);
 85                         break;
 86                     case "P":
 87                         if (graphADT.get(row).get(column).getDistance() == 1000)
 88                         {
 89                             length = length + 1;
 90                             graphADT.get(row).get(column).setDistance(length);
 91                             findDistance(graphADT, length, row + 1, column);
 92                             findDistance(graphADT, length, row - 1, column);
 93                             findDistance(graphADT, length, row, column + 1);
 94                             findDistance(graphADT, length, row, column - 1);
 95                         }
 96                         break;
 97                     case "T":
 98                         break;
 99                 }
100             }
101         }
102
103
104         static void printRules()
105         {
106             System.out.println();
107             System.out.println("Enter below key words to describe the path matrix");
108             System.out.println("START == S");
109             System.out.println("END   == E");
```

```java
                    System.out.println("END   == E");
                    System.out.println("PATH  == P");
                    System.out.println("TREE  == T");
        }

        public static void main (String args[])
        {
            LinkedList<LinkedList<Node>> graphADT = new LinkedList<>();

            Scanner sc=new Scanner(System.in);
            System.out.print("Enter the SIZE of the matrix: ");
            int s=sc.nextInt();
            System.out.println("Matrix size is: " + s + "*" + s);

            printRules();
            int k=1;

            for (int i=0; i<s; i++)
            {
                graphADT.add(new LinkedList<>());
                while (k<=s){
                    System.out.println("Enter elements of " +k+ " raw in the matrix: ");
                    for (int j=0; j<s; j++){
                        graphADT.get(i).add(new Node(sc.next(),i,j));
                        }
                        k++;
                        break;
                }
            }
            int size=graphADT.size();
            findStart(graphADT, size);
            findDistance(graphADT, 0, startR, startC);


            if (SD != 30*30)
            {
                System.out.println("Shortest way to exit from jungle: " + SD );
            }
            else
            {
                System.out.println("Sorry, No way to escape from the jungle");
            }


        }
}
```

```
C:\Users\2021e075\OneDrive - University of Jaffna\lab 7\lab>javac JungleRun.java

C:\Users\2021e075\OneDrive - University of Jaffna\lab 7\lab>java JungleRun
Enter the SIZE of the matrix: 5
Matrix size is: 5*5

Enter below key words to describe the path matrix
START == S
END   == E
PATH  == P
TREE  == T
Enter elements of 1 raw in the matrix:
S
P
P
P
P
Enter elements of 2 raw in the matrix:
T
P
T
P
P
Enter elements of 3 raw in the matrix:
T
P
P
P
P
Enter elements of 4 raw in the matrix:
P
T
E
T
T
Enter elements of 5 raw in the matrix:
P
T
P
T
T
Shortest way to exit from jungle: 5

C:\Users\2021e075\OneDrive - University of Jaffna\lab 7\lab>
```