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.

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"))

{

startR = i;

startC = j;

graphADT.get(startR).get(startC).setDistance(0);

break;

}

}

}

}

public static void findDistance(LinkedList<LinkedList<Node>> graphADT, int length, int row, int column)

{

if ((row < graphADT.size()) && (row > -1) && (column < graphADT.size()) && (column > -1))

{

switch (graphADT.get(row).get(column).getKey())

{

case "E":

if (SD > length + 1)

{

SD = length + 1;

graphADT.get(row).get(column).setDistance(SD);

}

break;

case "S":

findDistance(graphADT, length, row + 1, column);

findDistance(graphADT, length, row - 1, column);

findDistance(graphADT, length, row, column + 1);

findDistance(graphADT, length, row, column - 1);

break;

case "P":

if (graphADT.get(row).get(column).getDistance() == 1000)

{

length = length + 1;

graphADT.get(row).get(column).setDistance(length);

findDistance(graphADT, length, row + 1, column);

findDistance(graphADT, length, row - 1, column);

findDistance(graphADT, length, row, column + 1);

findDistance(graphADT, length, row, column - 1);

}

break;

case "T":

break;

}

}

}

static void printRules()

{

System.out.println();

System.out.println("Enter below key words to describe the path matrix");

System.out.println("START == S");

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");

}

}

}







