LAB 07

GRAPHS

JAYAWARDHANA D.D.V.M.

2020/E/061

GROUP B

SEMESTER 4

18 APRIL 2023

Q1.

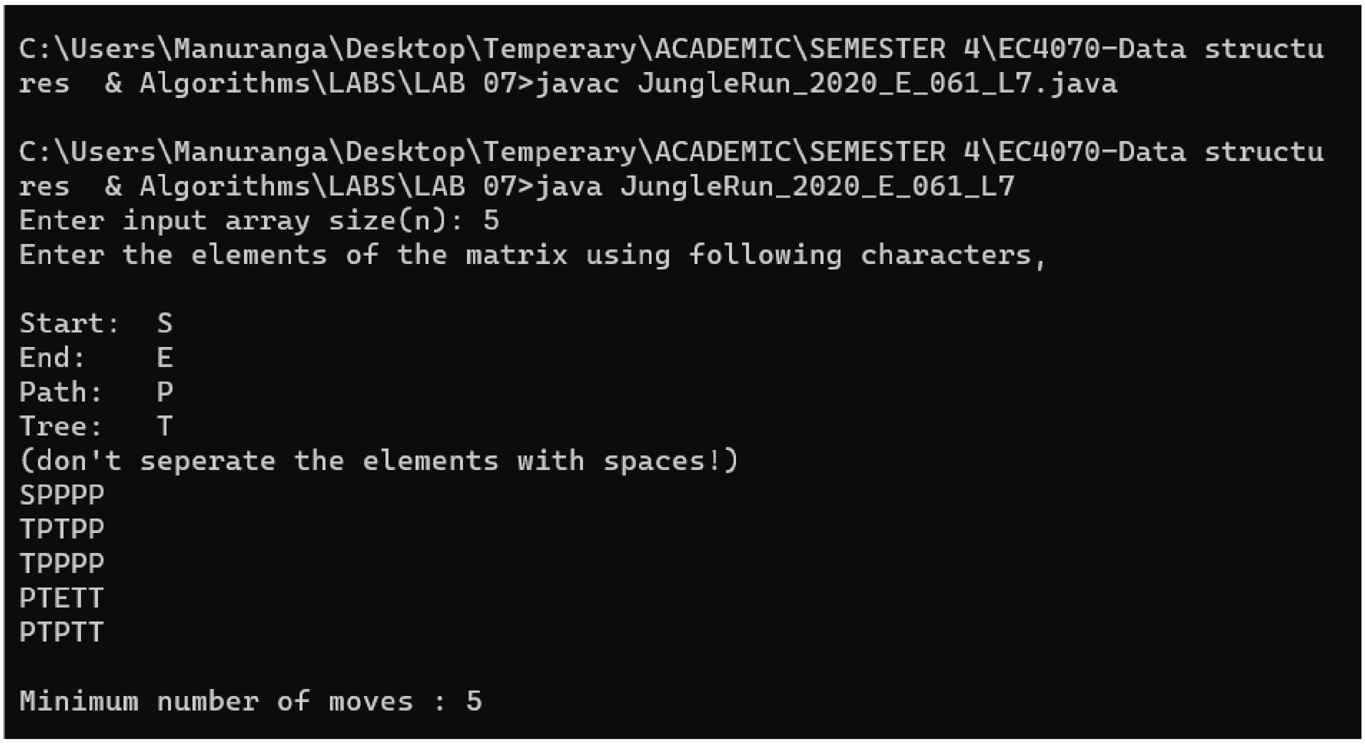
PROBLEM SPECIFICATION:

A person is in a jungle and he can’t find a way to get out of the jungle. And he needs to get out before the jungle gets dark. This problem asks to build an algorithm to find the shortest way out. The input consists of the number N, a size N x N matrix filled with the numbers S, E, T, and P, and finally our map. The map has a single S for the start point, a single E for the endpoint, a single P for the path, and a single T for the tree.

IMPLEMENTATION:

```
import java.util.Stack;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.BufferedReader;
public class JungleRun_2020_E_061_L7
{
        private Node startPoint, endPoint;
        Node[][] jungleMat;
        Stack<Node> stack = new Stack<>();

        public class Node
        {
                Node previous;
                int position;
                boolean isVisited = false;
                boolean isPath = false;
                boolean isSelected = false;
                char CurPos = ' ';
                Node(int position) {
                        this.position = position;
                }
        }
        JungleRun_2020_E_061_L7(char[][] map, int n)
        {
                jungleMat = new Node[n][n];
                for (int i = 0; i < n; i++)
                {
                        for (int j = 0; j < n; j++)
                        {
                                int tempPos = i * n + j;
                                jungleMat[i][j]=new Node(tempPos);
                                jungleMat[i][j].CurPos = map[i][j];
                                if (jungleMat[i][j].CurPos == 'P')
                                {
                                        jungleMat[i][j].isPath = true;
                                }
                                if (jungleMat[i][j].CurPos == 'S')
                                {
                                        this.startPoint = jungleMat[i][j];
                                }
                                if (jungleMat[i][j].CurPos == 'E')
                                {
                                        this.endPoint = jungleMat[i][j];
                                }
                        }
                }
        }

        public void addToStack(Node consNode, int n)
        {
                int consCol=consNode.position%n;
                int consRow=consNode.position/n;
                int[][] tempArr={{-1,0},{0,-1},{0,1},{1,0}};
                for (int[] a:tempArr)
                {
                        int tempRow=consRow + a[0];
                        int tempCol=consCol + a[1];
                        if
((tempRow>=0)&&(tempRow<n)&&(tempCol>=0)&&(tempCol<n)&&(!jungleMat[tempRow][tempCol].isVisited)
                                && ((jungleMat[tempRow][tempCol].CurPos
=='E')||(jungleMat[tempRow][tempCol].CurPos == 'P')))
                        {
                                jungleMat[consRow + a[0]][consCol + a[1]].previous =
jungleMat[consRow][consCol];
                                stack.push(jungleMat[consRow + a[0]][consCol + a[1]]);
                        }
                }
        }

        public int path()
        {
                Node consNode = startPoint;
                while (true)
                {
                        consNode.isVisited = true;
                        if (consNode==endPoint)
                        {
                                int numOfMoves=0;
                                while (consNode.previous != null)
                                {
                                        if ((consNode.CurPos!='S')&&(consNode.CurPos != 'E'))
                                        {
                                                consNode.CurPos=' ';
                                        }
                                        consNode=consNode.previous;
                                        numOfMoves++;
                                }
                                return numOfMoves;
                        }
                        addToStack(consNode,jungleMat.length);
                        if (!stack.empty())
                        {
                                consNode=stack.pop();
                        } else
                        {
                                return -1;
                        }
                }
        }
        public static void main(String[] args) throws IOException {
                System.out.print("Enter input array size(n): ");


                BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
                int n = Integer.parseInt(reader.readLine());
                char Arr[][] = new char[n][n];
                int inputSize = 0;
                String input="";
                System.out.println("Enter the elements of the matrix using following characters,\n\nStart:
S\nEnd:    E\nPath:   P\nTree:   T\n(don't seperate the elements with spaces!) ");


                for (int i = 0; i < n; i++)
                {
                        {
                                input = reader.readLine();
                                inputSize = input.length();
                        } while (inputSize != n);


                        for (int j = 0; j < n; j++) {
                                Arr[i][j] = input.charAt(j);
                        }
                }
                JungleRun_2020_E_061_L7 jungle1 = new JungleRun_2020_E_061_L7(Arr, n);
                System.out.println("\nMinimum number of moves : " + jungle1.path());
        }
}
```

OUTPUT:



```
C:\Users\Manuranga\Desktop\Temperary\ACADEMIC\SEMESTER 4\EC4070-Data structu
res  & Algorithms\LABS\LAB 07>javac JungleRun_2020_E_061_L7.java

C:\Users\Manuranga\Desktop\Temperary\ACADEMIC\SEMESTER 4\EC4070-Data structu
res  & Algorithms\LABS\LAB 07>java JungleRun_2020_E_061_L7
Enter input array size(n): 5
Enter the elements of the matrix using following characters,

Start:  S
End:    E
Path:   P
Tree:   T
(don't seperate the elements with spaces!)
SPPPP
TPTPP
TPPPP
PTETT
PTPTT

Minimum number of moves : 5
```

CONCLUSION

- This lab is about graphs.
- When implementing the program, there were many difficulties to face and much new knowledge to get.
- Some web references had to be used when implementing this program.