

Experiment 5 (Graph)

Student Name: SANSKAR AGRAWAL

Branch: BE CSE

Semester: 5th

Subject Name: Competitive Coding

UID: 20BCS5914

Section/Group: 20BCSMM-806B

Date of Performance: 18.10.2022

Subject Code: 20CSP_314

1. Aim/Overview of the practical:

- a. Journey to the moon.
- b. Frog in the maze.

2. Task to be done/ Which logistics used:

- a. Complete the journeyToMoon function in the editor below.

journeyToMoon has the following parameter(s):

int n: the number of astronauts

int astronaut[p][2]: each element astronaut[i] is a 2 element array that represents the ID's of two astronauts from the same country

- b. Alef the Frog is in an $n \times m$ two-dimensional maze represented as a table. The maze has the following Characteristics:
 - a. Each cell can be free or can contain an obstacle, an exit, or a mine.
 - b. Any two cells in the table considered adjacent if they share a side.
 - c. The maze is surrounded by a solid wall made of obstacles.
 - d. Some pairs of free cells are connected by a bidirectional tunnel.

- e. Each cell can be free or can contain an obstacle, an exit, or a mine.
- f. Any two cells in the table considered adjacent if they share a side.
- g. The maze is surrounded by a solid wall made of obstacles.
- h. Some pairs of free cells are connected by a bidirectional tunnel.

3. Steps for experiment/practical/Code:

a. Journey to the Moon:

```
import java.io.*;
import java.util.*;
public class Solution {
    static void numSeclection(LinkedList<Integer>[] links){
        int n = links.length;
        int[] group = new int[n];
        long[] count = new long[n+1];
        LinkedList<Integer> q = new LinkedList();
        q.add(0);
        group[0] = 1; count[1] = 1;
        int curGroup = 1;
        int unassignedNode = 1;
        while (!q.isEmpty()){
            int cur = q.removeFirst();
            for (int next:links[cur])
                if (group[next]==0){
                    group[next] = curGroup;
                    q.add(next);
                    count[curGroup]++;
                }
        }
        if (q.isEmpty()){
            while(unassignedNode<n && group[unassignedNode]!=0) unassignedNode++;
            if (unassignedNode<n){
```

Discover. Learn. Empower.

```
        curGroup++;
        group[unassignedNode] = curGroup;
        q.add(unassignedNode);
        count[curGroup]++;
        unassignedNode++;
    }
}
}
long result = 0;
long total = 0;
for (int i=0; i<=curGroup; i++)
    total += count[i];
for (int i=0; i<=curGroup; i++){
    total -= count[i];
    result += total*count[i];
}
System.out.print(result);
}

public static void main(String[] args) {
    /* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution. */
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
    int m = sc.nextInt();
    LinkedList<Integer>[] links = new LinkedList[n];
    for (int i=0; i<n; i++)
        links[i] = new LinkedList();
    for (int i=0; i<m; i++){
        int x = sc.nextInt();
        int y = sc.nextInt();
        links[x].add(y);
        links[y].add(x);
    }
    numSeclection(links);
}
}
```

b. Frog in maze:

```
import java.util.Arrays;

public class Solution002 {
    static final int EXIT = Integer.MAX_VALUE;
    public static void main(String[] args) {
        java.util.Scanner sc = new java.util.Scanner(System.in);
        int n = sc.nextInt(), m = sc.nextInt(), k = sc.nextInt();
        sc.nextLine();
        int[][] nextAry2 = new int[n + 2][m + 2];
        int[][] ids = new int[n + 2][m + 2];
        int ax = -1, ay = -1, id = 0;
        for (int i = 1; i <= n; ++i) {
            char[] typeLine = sc.nextLine().toCharArray();
            for (int j = 1; j <= m; ++j) {
                switch (typeLine[j - 1]) {
                    case '*':
                        nextAry2[i][j] = 1;
                        break;
                    case '#':
                        nextAry2[i][j] = 0;
                        break;
                    case '%':
                        nextAry2[i][j] = EXIT;
                        break;
                    case 'A':
                        ax = i;
                        ay = j;
                        break;
                    default:
                        nextAry2[i][j] = (i << 16) | j;
                }
            }
        }
        for (int i = 0; i < k; ++i) {
            int x0 = sc.nextInt(), y0 = sc.nextInt(), x1 = sc.nextInt(), y1 = sc.nextInt();
            nextAry2[x0][y0] = (x1 << 16) | y1;
        }
    }
}
```

```

        nextAry2[x1][y1] = (x0 << 16) | y0;
    }
    for (int i = 1; i <= n; ++i)
        for (int j = 1; j <= m; ++j)
            ids[i][j] = nextAry2[i][j] > 1 ? id++ : -1;

double[][] T = new double[id][id];
for (int i = 1; i <= n; ++i) {
    int[] nextAry2i = nextAry2[i];
    int[] idi = ids[i];
    for (int j = 1; j <= m; ++j) {
        int cid = idi[j];
        if (idi[j] < 0) continue;
        int v = nextAry2i[j];
        if (v != EXIT) {
            int a=v>>16,b=v&0xffff;
            if(a!=i || b!=j) {
                a = i;
                b = j;
            }
            int w0 = nextAry2[a][b - 1], w1 = nextAry2[a - 1][b], w2 = nextAry2[a][b + 1], w3 =
nextAry2[a + 1][b];

            int c = (w0 > 0 ? 1 : 0) + (w1 > 0 ? 1 : 0) + (w2 > 0 ? 1 : 0) + (w3 > 0 ? 1 : 0);
            if (c == 0) continue;
            double c1 = 1.0 / c;
            if(w0==EXIT) T[cid][ids[a][b-1]] = c1;
            else if(w0 > 1) T[cid][ids[w0 >> 16][w0 & 0xffff]] = c1;
            if(w1==EXIT) T[cid][ids[a-1][b]] = c1;
            else if (w1 > 1) T[cid][ids[w1 >> 16][w1 & 0xffff]] = c1;
            if(w2==EXIT) T[cid][ids[a][b+1]] = c1;
            else if (w2 > 1) T[cid][ids[w2 >> 16][w2 & 0xffff]] = c1;
            if(w3==EXIT) T[cid][ids[a+1][b]] = c1;
            else if (w3 > 1) T[cid][ids[w3 >> 16][w3 & 0xffff]] = c1;
            continue;
        }
        T[cid][cid] = 1.0;
    }
}

```

```
    }  
    }  
    print(T);  
    double[][] TP = pow(T, id, 0x10000L);  
    int ida = ids[ax][ay];  
    double rs = 0;  
    for (int i = 1; i <= n; ++i)  
        for (int j = 1; j <= m; ++j)  
            if (nextAry2[i][j] == EXIT) rs += TP[ida][ids[i][j]];  
    print(TP);  
    System.out.println(rs);  
}  
public static void print(double[][] x) {  
    System.out.println("");  
    for(int i=0;i<x.length;++i) {  
        if(i!=0) {  
            System.out.print(",");  
        }  
        System.out.println(Arrays.toString(x[i]));  
    }  
    System.out.println("");  
  
    for (int i = 0; i < x.length; ++i) {  
        if (i > 0) {  
            System.out.println("\n");  
        }  
        for (int j = 0; j < x[i].length; ++j) {  
            if (j > 0) {  
                System.out.print(' ');  
            }  
            System.out.print(String.format("%.20f", x[i][j]));  
        }  
    }  
  
    System.out.println();  
    System.out.println("----- ");
```

```
        System.out.println();
    }

    static void print(Object...args) {
        System.out.println(Arrays.toString(args));
    }

    static void mul(double[][] A, double[][] B, double[][] R, int n) {
        for (int i = 0; i < n; i++) {
            double[] Ri = R[i], Ai = A[i];
            for (int j = 0; j < n; j++)
                for (int k = 0; k < n; k++) Ri[j] += Ai[k] * B[k][j];
        }
    }

    static double[][] pow(double[][] A, int n, long p) {
        double[][] C = new double[n][n], R = new double[n][n], t = null;
        for (int i = 0; i < n; i++) R[i][i] = 1;
        while (p != 0) {
            if (p % 2 == 1) {
                mul(A, R, C, n);
                t = C;
                C = R;
                R = t;
            }
            mul(A, A, C, n);
            t = C;
            C = A;
            A = t;
            p >>= 1;
        }
        return R;
    }
}
```

4. Result/Output/Writing Summary:

a. Journey to the Moon:

✓ Test case 0	Compiler Message
✓ Test case 1	Success
✓ Test case 2	Input (stdin) Download
✓ Test case 3	1 5 3
✓ Test case 4	2 0 1
✓ Test case 5	3 2 3
✓ Test case 6	4 0 4
	Expected Output Download
	1 6

b. Frog in the Maze:

✓ Test case 0	Compiler Message
✓ Test case 1	Success
✓ Test case 2	Input (stdin) Download
✓ Test case 3	1 3 6 1
✓ Test case 4	2 ###*00
✓ Test case 5	3 0#0A%0
✓ Test case 6	4 ###*00
✓ Test case 7	5 2 3 2 1
	Expected Output Download
	1 0.25

Learning outcomes (What I have learnt):

- Learnt about vectors and hashing.
- Learnt about graphs.
- Got an overview of the type of questions on hacker-rank.
- Get to know about crucial test cases.



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			