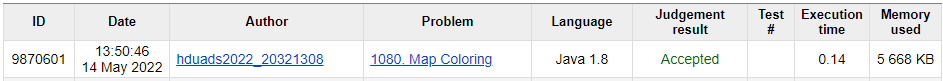
Laboratory work #7

Student: CAO Xinyang  
Student ID: 20321308  
Timus Name: hduads2022\_20321308

Mail: c.x\_yang@foxmail.com

Problem #1080

Screenshot from Timus:



Explanation of algorithm:

When a point is red, its next point is blue; And vice versa. So we can use either a depth-first algorithm (DFS) or a breadth-first algorithm (BFS) to traverse all points and color them in turn. When two adjacent points are of the same color in this process, it is judged to be impossible. This code uses the depth-first algorithm.

Computational complexity of algorithm:

T(N) = O(N^2)

Source code:

import java.util.Scanner;

public class App {

public static int[][] matrix;

public static int[] color;

public static boolean flag = false;

public static void DFS(int index, int size) {

if (flag)

return;

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

if (matrix[index][i] == 1) {

if (color[i] == -1) {

if(color[index] == 0){

color[i] = 1;

}else if(color[index] == 1){

color[i] = 0;

}

DFS(i, size);

} else if (color[i] == color[index]) {

flag = true;

return;

}

}

}

}

public static void main(String[] args){

Scanner scan = new Scanner(System.in);

int n = scan.nextInt();

matrix = new int[n][n];

color = new int[n];

for (int i = 0; i < n; i++) {

color[i] = -1;

}

for (int i = 0; i < n; i++) {

while (true) {

int index = scan.nextInt() - 1;

if (index == -1)

break;

matrix[i][index] = 1;

matrix[index][i] = 1;

}

}

for (int i = 0; i < n; i++) {

if (color[i] == -1) {

color[i] = 0;

DFS(i, n);

}

}

if (flag) {

System.out.println(-1);

} else {

for (int i = 0; i < n; i++) {

System.out.print(color[i]);

}

}

scan.close();

}

}