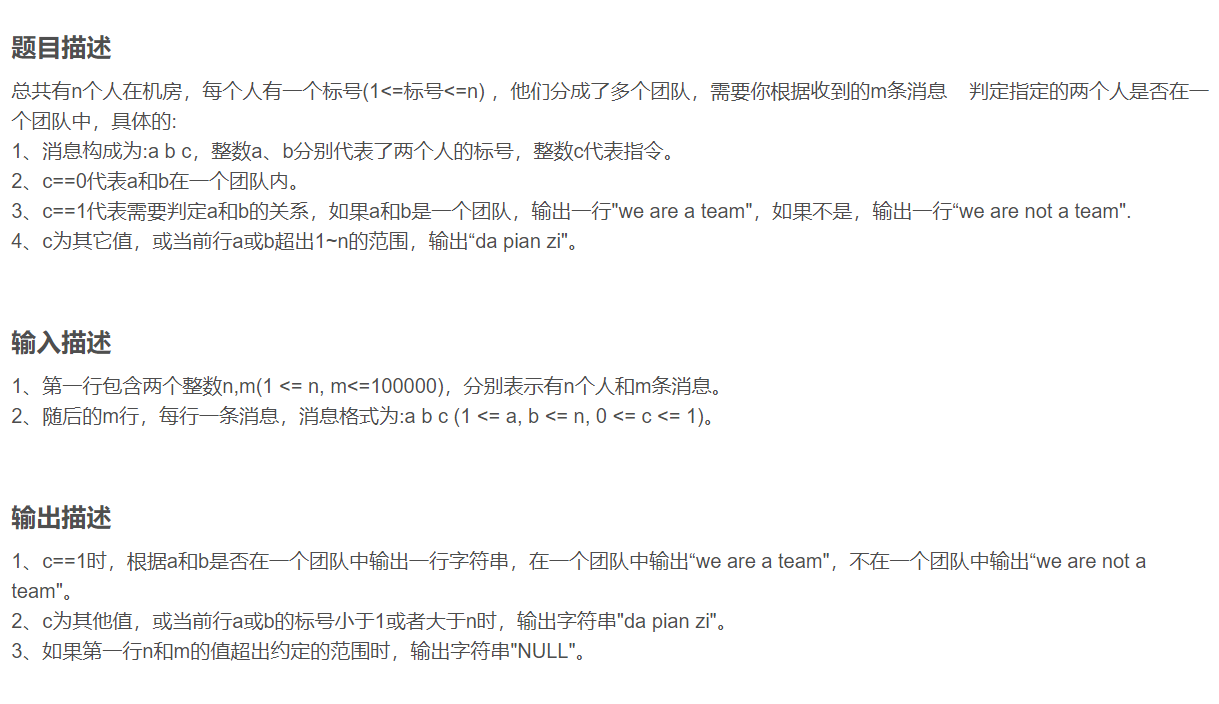
# **E卷-We Are A Team[200分]（ Java | Python3 | C++ | C语言 | JsNode | Go ）**





5 6

1 2 0

1 2 1

1 5 0

2 3 1

2 5 1

1 3 2

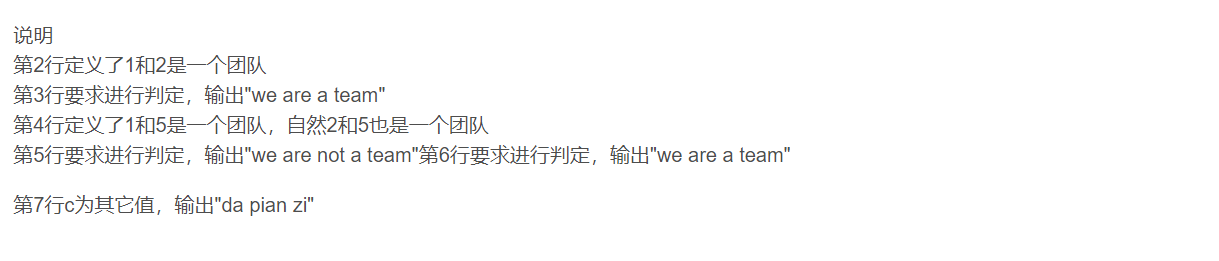


we are a team

we are not a team

we are a team

da pian zi





import java.util.Scanner;

public class Main {

static int[] f;

// 查找函数，查找x的团队根节点

static int getf(int x) {

if (x == f[x]) {

return x; // 如果x是自己的根节点，返回x

}

f[x] = getf(f[x]); // 路径压缩，直接连接到根节点

return f[x];

}

// 合并函数，将x和y所在的团队合并

static void merge(int x, int y) {

f[getf(y)] = getf(x); // 将y的根节点连接到x的根节点

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 读取输入的n和m，分别表示人数和消息数

int n = scanner.nextInt();

int m = scanner.nextInt();

// 检查n和m是否在约定范围内

if (n < 1 || n > 100000 || m < 1 || m > 100000) {

System.out.println("NULL"); // 如果不在范围内，输出"NULL"

return;

}

// 初始化并查集数组f，f[i]表示第i个人所在的团队

f = new int[n + 1];

for (int i = 1; i <= n; i++) {

f[i] = i; // 每个人初始时自己是自己的团队

}

// 处理每条消息

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

int a = scanner.nextInt();

int b = scanner.nextInt();

int c = scanner.nextInt();

// 检查a和b是否在有效范围内

if (a < 1 || a > n || b < 1 || b > n) {

System.out.println("da pian zi"); // 如果不在范围内，输出"da pian zi"

continue;

}

if (c == 0) {

merge(a, b); // 如果c为0，合并a和b的团队

} else if (c == 1) {

// 如果c为1，判断a和b是否在同一团队

if (getf(a) == getf(b)) {

System.out.println("we are a team"); // 在同一团队，输出"we are a team"

} else {

System.out.println("we are not a team"); // 不在同一团队，输出"we are not a team"

}

} else {

System.out.println("da pian zi"); // 如果c不是0或1，输出"da pian zi"

}

}

scanner.close();

}

}



import sys

# 设置递归深度限制，确保深度优先搜索不会超出限制

sys.setrecursionlimit(100005)

# 读取输入的n和m，分别表示人数和消息数

n, m = [int(\_) for \_ in input().split()]

# 检查n和m是否在约定范围内

if n < 1 or n > 100000 or m < 1 or m > 100000:

print("NULL") # 如果不在范围内，输出"NULL"

exit(0) # 退出程序

# 初始化并查集数组f，f[i]表示第i个人所在的团队

f = [0] \* (n + 1)

for i in range(1, n + 1):

f[i] = i # 每个人初始时自己是自己的团队

# 查找函数，查找x的团队根节点

def getf(x):

if x == f[x]: # 如果x是自己的根节点

return x # 返回x

f[x] = getf(f[x]) # 路径压缩，直接连接到根节点

return f[x] # 返回根节点

# 合并函数，将x和y所在的团队合并

def merge(x, y):

f[getf(y)] = getf(x) # 将y的根节点连接到x的根节点

# 处理每条消息

for i in range(m):

a, b, c = [int(\_) for \_ in input().split()] # 读取消息中的a, b, c

# 检查a和b是否在有效范围内

if a < 1 or a > n or b < 1 or b > n:

print("da pian zi") # 如果不在范围内，输出"da pian zi"

continue # 跳过当前消息

if c == 0:

merge(a, b) # 如果c为0，合并a和b的团队

elif c == 1:

# 如果c为1，判断a和b是否在同一团队

if getf(a) == getf(b):

print("we are a team") # 在同一团队，输出"we are a team"

else:

print("we are not a team") # 不在同一团队，输出"we are not a team"

else:

print("da pian zi") # 如果c不是0或1，输出"da pian zi"



#include <iostream>

#include <vector>

using namespace std;

vector<int> f;

// 查找函数，查找x的团队根节点

int getf(int x) {

if (x == f[x]) {

return x; // 如果x是自己的根节点，返回x

}

f[x] = getf(f[x]); // 路径压缩，直接连接到根节点

return f[x];

}

// 合并函数，将x和y所在的团队合并

void merge(int x, int y) {

f[getf(y)] = getf(x); // 将y的根节点连接到x的根节点

}

int main() {

int n, m;

cin >> n >> m; // 读取输入的n和m，分别表示人数和消息数

// 检查n和m是否在约定范围内

if (n < 1 || n > 100000 || m < 1 || m > 100000) {

cout << "NULL" << endl; // 如果不在范围内，输出"NULL"

return 0;

}

// 初始化并查集数组f，f[i]表示第i个人所在的团队

f.resize(n + 1);

for (int i = 1; i <= n; i++) {

f[i] = i; // 每个人初始时自己是自己的团队

}

// 处理每条消息

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

int a, b, c;

cin >> a >> b >> c; // 读取消息中的a, b, c

// 检查a和b是否在有效范围内

if (a < 1 || a > n || b < 1 || b > n) {

cout << "da pian zi" << endl; // 如果不在范围内，输出"da pian zi"

continue;

}

if (c == 0) {

merge(a, b); // 如果c为0，合并a和b的团队

} else if (c == 1) {

// 如果c为1，判断a和b是否在同一团队

if (getf(a) == getf(b)) {

cout << "we are a team" << endl; // 在同一团队，输出"we are a team"

} else {

cout << "we are not a team" << endl; // 不在同一团队，输出"we are not a team"

}

} else {

cout << "da pian zi" << endl; // 如果c不是0或1，输出"da pian zi"

}

}

return 0;

}



#include <stdio.h>

#include <stdlib.h>

int\* f;

// 查找函数，查找x的团队根节点

int getf(int x) {

if (x == f[x]) {

return x; // 如果x是自己的根节点，返回x

}

f[x] = getf(f[x]); // 路径压缩，直接连接到根节点

return f[x];

}

// 合并函数，将x和y所在的团队合并

void merge(int x, int y) {

f[getf(y)] = getf(x); // 将y的根节点连接到x的根节点

}

int main() {

int n, m;

scanf("%d %d", &n, &m); // 读取输入的n和m，分别表示人数和消息数

// 检查n和m是否在约定范围内

if (n < 1 || n > 100000 || m < 1 || m > 100000) {

printf("NULL\n"); // 如果不在范围内，输出"NULL"

return 0;

}

// 初始化并查集数组f，f[i]表示第i个人所在的团队

f = (int\*)malloc((n + 1) \* sizeof(int));

for (int i = 1; i <= n; i++) {

f[i] = i; // 每个人初始时自己是自己的团队

}

// 处理每条消息

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

int a, b, c;

scanf("%d %d %d", &a, &b, &c); // 读取消息中的a, b, c

// 检查a和b是否在有效范围内

if (a < 1 || a > n || b < 1 || b > n) {

printf("da pian zi\n"); // 如果不在范围内，输出"da pian zi"

continue;

}

if (c == 0) {

merge(a, b); // 如果c为0，合并a和b的团队

} else if (c == 1) {

// 如果c为1，判断a和b是否在同一团队

if (getf(a) == getf(b)) {

printf("we are a team\n"); // 在同一团队，输出"we are a team"

} else {

printf("we are not a team\n"); // 不在同一团队，输出"we are not a team"

}

} else {

printf("da pian zi\n"); // 如果c不是0或1，输出"da pian zi"

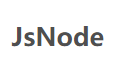
}

}

free(f); // 释放动态分配的内存

return 0;

}



const readline = require("readline");

// 创建接口以读取输入

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout,

});

let f = []; // 初始化并查集数组f

// 查找函数，查找x的团队根节点

function getf(x) {

if (x === f[x]) {

return x; // 如果x是自己的根节点，返回x

}

f[x] = getf(f[x]); // 路径压缩，直接连接到根节点

return f[x];

}

// 合并函数，将x和y所在的团队合并

function merge(x, y) {

f[getf(y)] = getf(x); // 将y的根节点连接到x的根节点

}

let firstLine = true;

rl.on("line", (line) => {

const input = line.split(" ").map(Number);

if (firstLine) {

// 读取输入的n和m，分别表示人数和消息数

const n = input[0];

const m = input[1];

// 检查n和m是否在约定范围内

if (n < 1 || n > 100000 || m < 1 || m > 100000) {

console.log("NULL"); // 如果不在范围内，输出"NULL"

rl.close();

return;

}

f = new Array(n + 1);

for (let i = 1; i <= n; i++) {

f[i] = i; // 每个人初始时自己是自己的团队

}

firstLine = false;

} else {

const a = input[0];

const b = input[1];

const c = input[2];

// 检查a和b是否在有效范围内

if (a < 1 || a > f.length - 1 || b < 1 || b > f.length - 1) {

console.log("da pian zi"); // 如果不在范围内，输出"da pian zi"

return;

}

if (c === 0) {

merge(a, b); // 如果c为0，合并a和b的团队

} else if (c === 1) {

// 如果c为1，判断a和b是否在同一团队

if (getf(a) === getf(b)) {

console.log("we are a team"); // 在同一团队，输出"we are a team"

} else {

console.log("we are not a team"); // 不在同一团队，输出"we are not a team"

}

} else {

console.log("da pian zi"); // 如果c不是0或1，输出"da pian zi"

}

}

});



package main

import (

"bufio"

"fmt"

"os"

"strconv"

"strings"

)

var f []int // 初始化并查集数组f

// 查找函数，查找x的团队根节点

func getf(x int) int {

if x == f[x] {

return x // 如果x是自己的根节点，返回x

}

f[x] = getf(f[x]) // 路径压缩，直接连接到根节点

return f[x]

}

// 合并函数，将x和y所在的团队合并

func merge(x, y int) {

f[getf(y)] = getf(x) // 将y的根节点连接到x的根节点

}

func main() {

reader := bufio.NewReader(os.Stdin)

line, \_ := reader.ReadString('\n')

line = strings.TrimSpace(line)

parts := strings.Split(line, " ")

n, \_ := strconv.Atoi(parts[0])

m, \_ := strconv.Atoi(parts[1])

// 检查n和m是否在约定范围内

if n < 1 || n > 100000 || m < 1 || m > 100000 {

fmt.Println("NULL") // 如果不在范围内，输出"NULL"

return

}

// 初始化并查集数组f，f[i]表示第i个人所在的团队

f = make([]int, n+1)

for i := 1; i <= n; i++ {

f[i] = i // 每个人初始时自己是自己的团队

}

// 处理每条消息

for i := 0; i < m; i++ {

line, \_ := reader.ReadString('\n')

line = strings.TrimSpace(line)

parts := strings.Split(line, " ")

a, \_ := strconv.Atoi(parts[0])

b, \_ := strconv.Atoi(parts[1])

c, \_ := strconv.Atoi(parts[2])

// 检查a和b是否在有效范围内

if a < 1 || a > n || b < 1 || b > n {

fmt.Println("da pian zi") // 如果不在范围内，输出"da pian zi"

continue

}

if c == 0 {

merge(a, b) // 如果c为0，合并a和b的团队

} else if c == 1 {

// 如果c为1，判断a和b是否在同一团队

if getf(a) == getf(b) {

fmt.Println("we are a team") // 在同一团队，输出"we are a team"

} else {

fmt.Println("we are not a team") // 不在同一团队，输出"we are not a team"

}

} else {

fmt.Println("da pian zi") // 如果c不是0或1，输出"da pian zi"

}

}

}