总共有 n 个人在机房，每个人有一个标号（1<=标号<=n），他们分成了多个团队，需要你根据收到的 m 条消息判定指定的两个人是否在一个团队中，具体是：

1. 消息格式为 a b c，整数 a、b 分别代表两个人的标号，整数 c 代表指令
2. c == 0 代表 a 和 b 在一个团队内
3. c == 1 代表需要判定 a 和 b 的关系，如果 a 和 b 是一个团队，输出一行'we are a team',如果不是，输出一行'we are not a team'
4. c 为其他值，或当前行 a 或 b 超出 1~n 的范围，输出'da pian zi'

### 输入描述

1. 第一行包含两个整数 n，m(1<=n,m<=100000),分别表示有 n 个人和 m 条消息
2. 随后的 m 行，每行一条消息，消息格式为：a b c(1<=a,b<=n,0<=c<=1)

### 输出描述

1. c == 1 根据 a 和 b 是否在一个团队中输出一行字符串，在一个团队中输出'we are a team',不在一个团队中输出'we are not a team'
2. c 为其他值，或当前行 a 或 b 的标号小于 1 或者大于 n 时，输出字符串'da pian zi'
3. 如果第一行 n 和 m 的值超出约定的范围时，输出字符串"Null"。

### 用例

| | |
|---|---|
| 输入 | 5 7<br>1 2 0<br>4 5 0<br>2 3 0<br>1 2 1<br>2 3 1<br>4 5 1<br>1 5 1 |
| 输出 | We are a team<br>We are a team<br>We are a team<br>We are not a team |
| 说明 | 无 |

| | |
|---|---|
| 输入 | 3 0<br>1 2 0<br>1 2 1<br>1 5 0<br>2 3 1<br>2 5 1<br>1 3 2 |
| 输出 | we are a team<br>we are not a team<br>we are a team<br>da pian zi |
| 说明 | 无 |

### 题目解析

本题其实就是构造 无向图 ，并根据无向图确定两个顶点的连接关系，因此可以使用并查集实现。

JavaScript算法源码

```
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const lines = [];
let n, m;
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 1) {
    [n, m] = lines[0].split(" ").map(Number);
  }

  if (n && lines.length === m + 1) {
    lines.shift();
    const msgs = lines.map((line) => line.split(" ").map(Number));

    print(msgs, n, m);

    lines.length = 0;
  }
});

function print(msgs, n, m) {
  if (n < 1 || n >= 100000 || m < 1 || m >= 100000) return console.log("Null");

  const ufs = new UnionFindSet(n);
  msgs
    .sort(([a1, b1, c1], [a2, b2, c2]) => c1 - c2)
    .forEach((msg) => {
      const [a, b, c] = msg;

      if (a < 1 || a > n || b < 1 || b > n) {
        return console.log("da pian zi");
      }

      if (c === 0) {
        ufs.union(a, b);
      } else if (c === 1) {
        const res =
          ufs.find(a) === ufs.find(b) ? "We are a team" : "We are not a team";
        console.log(res);
      } else {
        console.log("da pian zi");
      }
    });
}

class UnionFindSet {
  constructor(n) {
    this.fa = new Array(n + 1).fill(0).map((_, idx) => idx);
  }

  find(x) {
    if (this.fa[x] !== x) {
      this.fa[x] = this.find(this.fa[x]);
      return this.fa[x];
    }
    return x;
  }

  union(x, y) {
    let x_fa = this.find(x);
    let y_fa = this.find(y);

    if (x_fa !== y_fa) {
      this.fa[y_fa] = x_fa;
    }
  }
}
```

Java算法源码

```
import java.util.Arrays;
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    int n = sc.nextInt();
    int m = sc.nextInt();

    int[][] msgs = new int[m][3];
    for (int i = 0; i < m; i++) {
      msgs[i][0] = sc.nextInt();
      msgs[i][1] = sc.nextInt();
      msgs[i][2] = sc.nextInt();
    }

    getResult(msgs, n, m);
  }

  public static void getResult(int[][] msgs, int n, int m) {
    if (n < 1 || n >= 100000 || m < 1 || m >= 100000) {
      System.out.println("Null");
      return;
    }

    UnionFindSet ufs = new UnionFindSet(n + 1);

    Arrays.sort(msgs, (x, y) -> x[2] - y[2]);

    for (int[] msg : msgs) {
      int a = msg[0], b = msg[1], c = msg[2];

      if (a < 1 || a > n || b < 1 || b > n) {
        System.out.println("da pian zi");
        continue;
      }

      if (c == 0) {
        ufs.union(a, b);
      } else if (c == 1) {
        System.out.println(ufs.find(a) == ufs.find(b) ? "We are a team" : "We are not a team");
      } else {
        System.out.println("da pian zi");
      }
    }
  }
}

class UnionFindSet {
  int[] fa;

  public UnionFindSet(int n) {
    this.fa = new int[n];
    for (int i = 0; i < n; i++) fa[i] = i;
  }

  public int find(int x) {
    if (this.fa[x] != x) {
      return this.fa[x] = this.find(this.fa[x]);
    }
    return x;
  }

  public void union(int x, int y) {
    int x_fa = this.find(x);
    int y_fa = this.find(y);

    if (x_fa != y_fa) {
      this.fa[y_fa] = x_fa;
    }
  }
}
```

Python算法源码

```
# 输入获取
n, m = map(int, input().split())
msgs = [list(map(int, input().split())) for i in range(m)]


# 并查集实现
class UnionFindSet:
    def __init__(self, n):
        self.fa = [i for i in range(n)]

    def find(self, x):
        if self.fa[x] != x:
            self.fa[x] = self.find(self.fa[x])
            return self.fa[x]
        return x

    def union(self, x, y):
        x_fa = self.find(x)
        y_fa = self.find(y)

        if x_fa != y_fa:
            self.fa[y_fa] = x_fa


# 算法入口
def getResult(n, m, msgs):
    if n < 1 or n >= 100000 or m < 1 or m >= 100000:
        print("Null")
        return

    ufs = UnionFindSet(n + 1)

    msgs.sort(key=lambda x: x[2])

    for a, b, c in msgs:
        if a < 1 or a > n or b < 1 or b > n:
            print("da pian zi")
            continue

        if c == 0:
            ufs.union(a, b)
        elif c == 1:
            print("We are a team" if ufs.find(a) == ufs.find(b) else "We are not a team")
        else:
            print("da pian zi")


# 算法调用
getResult(n, m, msgs)
```