

# 随机？随机~

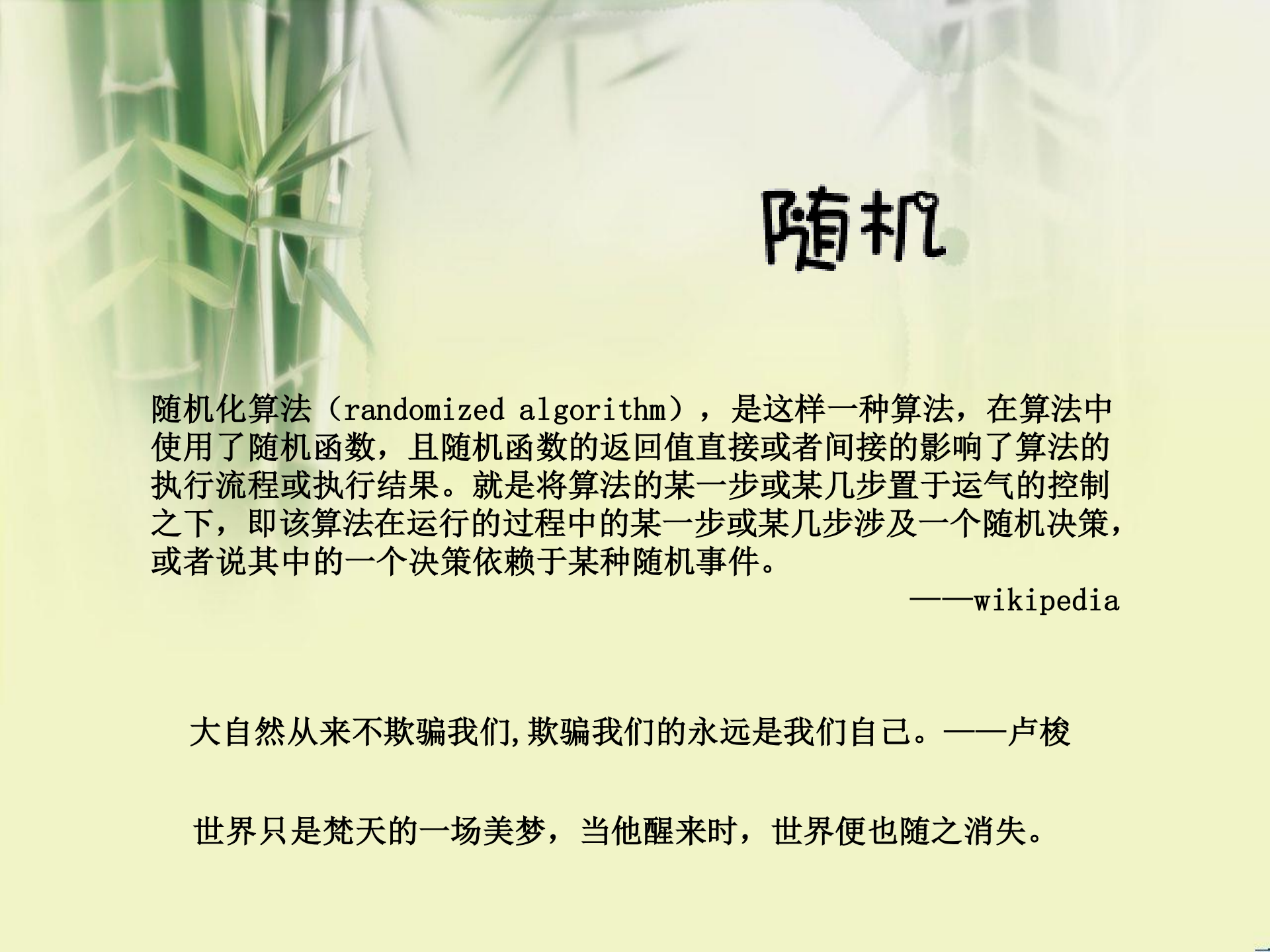
Shi\*han\*yuan\*



15216677522



shihanyuan1995@gmail.com



# 随机

随机化算法（randomized algorithm），是这样一种算法，在算法中使用了随机函数，且随机函数的返回值直接或者间接的影响了算法的执行流程或执行结果。就是将算法的某一步或某几步置于运气的控制之下，即该算法在运行的过程中的某一步或某几步涉及一个随机决策，或者说其中的一个决策依赖于某种随机事件。

——wikipedia

大自然从来不欺骗我们, 欺骗我们的永远是我们自己。——卢梭

世界只是梵天的一场美梦，当他醒来时，世界便也随之消失。

## 快速排序

```
void Qsort(int a[], int low, int high) {  
    if(low >= high) return;  
    int first = low, last = high, key = a[first];  
    while(first < last) {  
        while(first < last && a[last] >= key) --last;  
        a[first] = a[last];  
        while(first < last && a[first] <= key) ++first;  
        a[last] = a[first];  
    }  
    a[first] = key;  
    Qsort(a, low, first-1);  
    Qsort(a, first+1, high);  
}
```

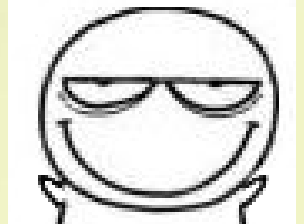
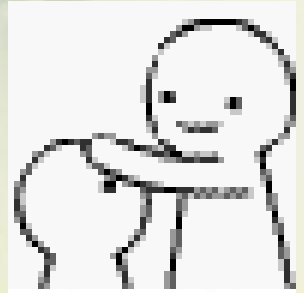


```
void Qsort(int a[], int low, int high) {  
    if(low >= high) return;  
    swap(a[first], a[(first + last) >> 1]);  
    int first = low, last = high, key = a[first];  
    while(first < last) {  
        while(first < last && a[last] >= key) --last;  
        a[first] = a[last];  
        while(first < last && a[first] <= key) ++first;  
        a[last] = a[first];  
    }  
    a[first] = key;  
    Qsort(a, low, first-1);  
    Qsort(a, first+1, high);  
}
```



```
void Qsort(int a[], int low, int high) {  
    if(low >= high) return;  
    int first = low, last = high, key = a[first];  
    while(first < last) {  
        while(first < last && a[last] >= key) --last;  
        a[first] = a[last];  
        while(first < last && a[first] <= key) ++first;  
        a[last] = a[first];  
    }  
    a[first] = key;  
    Qsort(a, low, first-1);  
    Qsort(a, first+1, high);  
}
```

**random\_shuffle(a, a + n);**



## Treap

随机为二叉搜索树中的每个点赋一个权值

随机射线

判断一个点是不是在多边形内

当看到数据随机的时候，你会想到什么？



Warming up



猴子最讨厌什么线？



布和纸怕什么？





# 红眼睛与蓝眼睛

在很远很远的地方，有一个神奇的岛屿。岛屿上住着一群可爱的人，他们的眼睛有红和蓝两种颜色，这群人虔诚地信奉属于他们的宗教。在一块古老的石碑上，首任部落长老刻下了以下教义：

- 1、如果一个教徒知道了自己是红眼睛，那么他必须在那个夜晚自行了结自己的生命；
- 2、每个人都不能告诉别人他们眼睛的颜色；
- 3、每个人都不能通过任何途径发现自己眼睛的颜色；

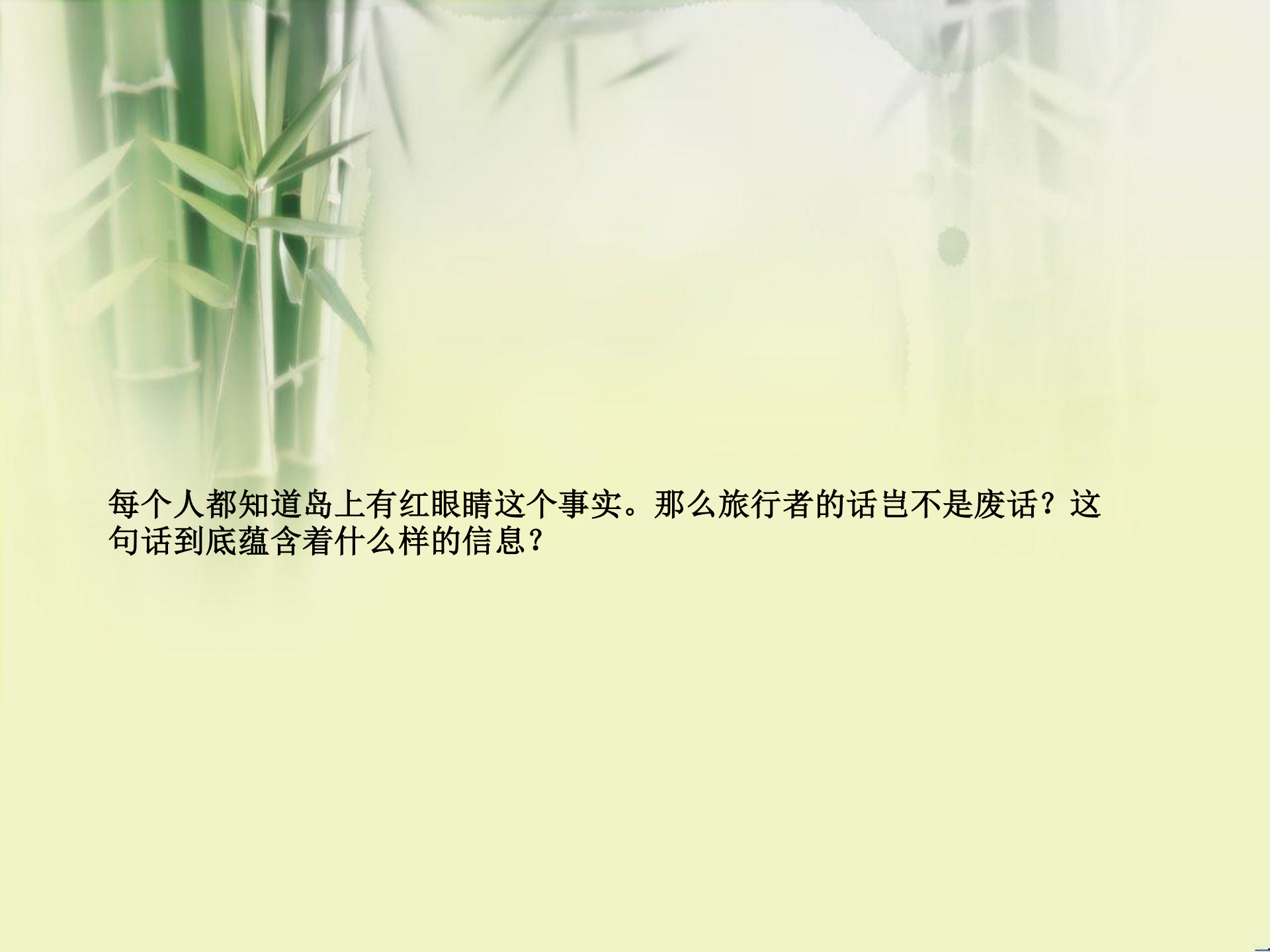
这群岛民就这样幸福地生活着。

直到有一天。来了一位异乡的旅行者，他受到了所有岛民热情的欢迎，然而他对于岛上的情况并不知情，他对着众人说：原来你们这里有红眼睛呀！话音刚落，他就继续踏上了旅途。

第一个夜晚，风平浪静。第二个夜晚，依旧风平浪静。

第三天，三个人自杀了。

请问发生了什么？



每个人都知道岛上有红眼睛这个事实。那么旅行者的话岂不是废话？这句话到底蕴含着什么样的信息？

## **POJ3318**

### **Description**

给定三个 $n * n$ 的矩阵**A**、**B**、**C**，问是否有 $\mathbf{A} * \mathbf{B} = \mathbf{C}$ 成立。

### **Input**

$1 \leq n \leq 1000$



## Solution

矩阵乘法暴力？  $O(n^3)$

如果你的矩乘算法足够优秀， **maybe is ok;**

但是如果我把范围开大一些呢？

## **Solution**

随机一个由 $n$ 个元素构成的行向量 $H$ ,

如果 $AB = C$ 成立, 根据矩阵乘法的结合律,  $H(AB) = (HA)B = HC$ 成立;

一个行向量与矩阵的乘法复杂度为 $O(n^2)$ , nice!

喂等等，这个算法不靠谱啊！事实是在 $AB \neq C$ 的前提下，满足 $HAB = HC$ 的 $H$ 依旧多的数不清；

该怎办呢？

那么就多随几次 $H$ 呗。

## POJ1686

### Description

A math instructor is too lazy to grade a question in the exam papers in which students are supposed to produce a complicated formula for the question asked. Students may write correct answers in different forms which makes grading very hard. So, the instructor needs help from computer programmers and you can help. You are to write a program to read different formulas and determine whether or not they are arithmetically equivalent.

### Input

The first line of the input contains an integer  $N$  ( $1 \leq N \leq 20$ ) that is the number of test cases. Following the first line, there are two lines for each test case. A test case consists of two arithmetic expressions, each on a separate line with at most 80 characters. There is no blank line in the input. An expression contains one or more of the following:

- Single letter variables (case insensitive).

- Single digit numbers.

- Matched left and right parentheses.

- Binary operators  $+$ ,  $-$  and  $*$  which are used for addition, subtraction and multiplication respectively.

- Arbitrary number of blank or tab characters between above tokens.

Note: Expressions are syntactically correct and evaluated from left to right with equal precedence (priority) for all operators. The coefficients and exponents of the variables are guaranteed to fit in 16-bit integers.





## Solution

暴力。直接把表达式树建出来，然后计算每一个项出现的系数。

能不能随机呢？

给变量随机赋值，把两个表达式的值计算出来进行比较。

把上述操作进行若干次，若每次都相等，则两个表达式等价。



## Description

给出若干点的四元组；

给出若干组询问，每组询问对应一个矩形，试问以下结论是否成立：对于任意四元组，都恰好有两个点落在矩形内？

## Input

所有的数字均为整数，且其绝对值小于等于 $10^9$ ；

四元组数 $n$ 小于等于 $10^5$ ；

询问数 $m$ 小于等于 $10^5$ ；



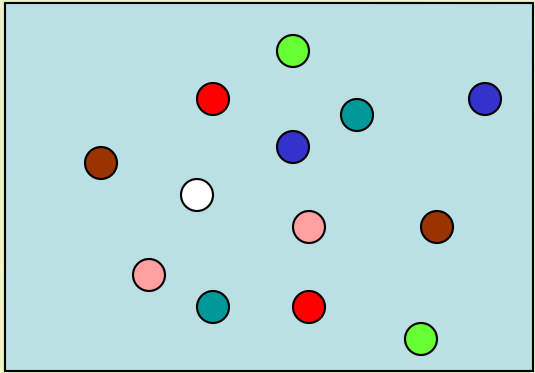
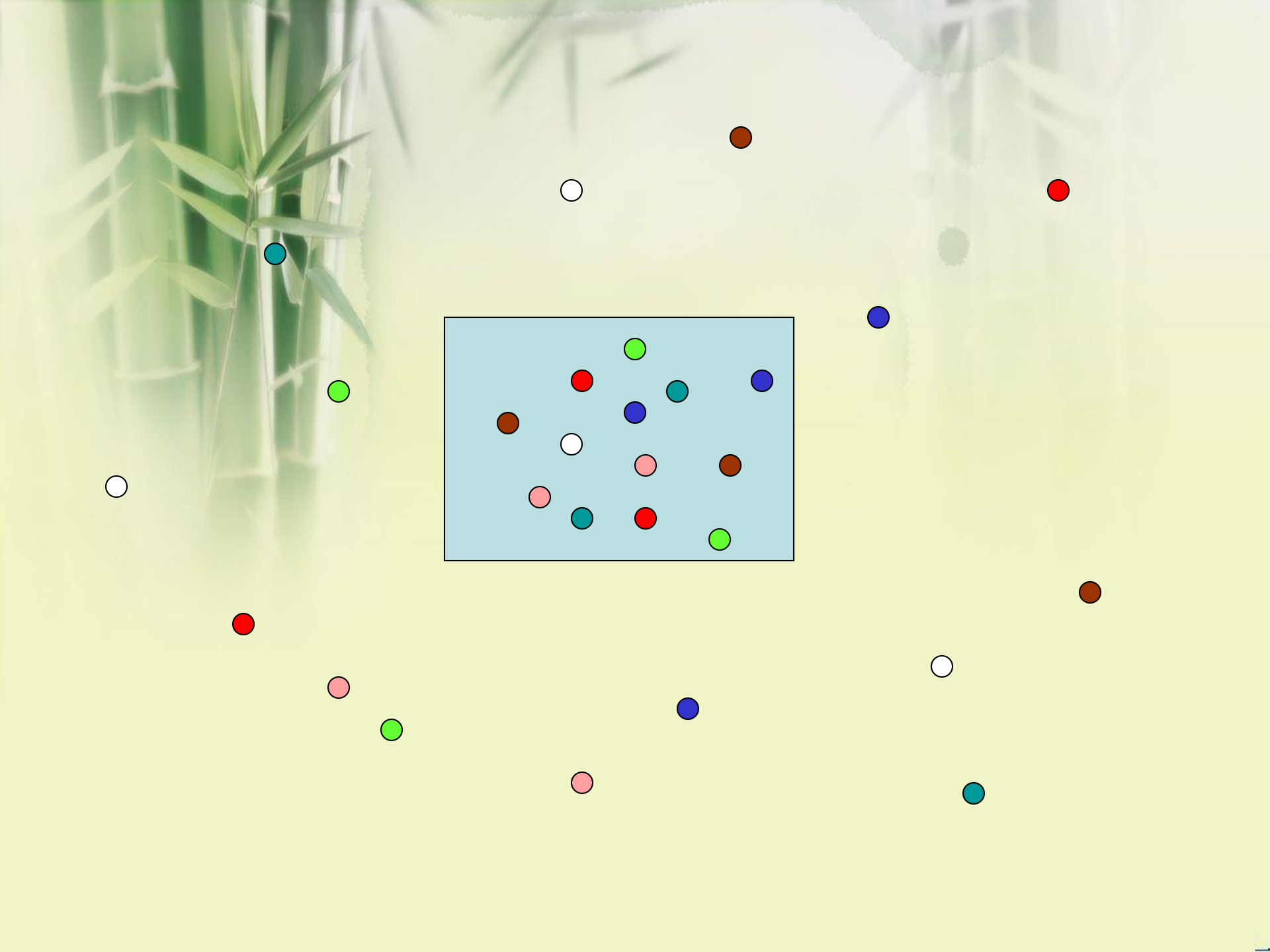
## Solution

暴力。 $O(n*m)$ 。祈祷奇迹发生？

很多时候暴力并出不了奇迹。

对于每组询问随机**check**若干点对？

对于随机数据也许可以。





## **Solution**

**Step1:** 对于每组点的四元组赋一个权值。

**Step2:** 对于每个询问，查询在对应矩阵内的点权和，若恰好等于总权值的一半，那么题目结论成立。

## **POJ2576**

### **Description**

**A tug of war is to be arranged at the local office picnic. For the tug of war, the picnickers must be divided into two teams. Each person must be on one team or the other; the number of people on the two teams must not differ by more than 1; the total weight of the people on each team should be as nearly equal as possible.**

### **Input**

**The first line of input contains n the number of people at the picnic. n lines follow. The first line gives the weight of person 1; the second the weight of person 2; and so on. Each weight is an integer between 1 and 450. There are at most 100 people at the picnic.**



## **Solution**

**Step1:** 把 $n$ 个人随机分为两组;

**Step2:** 从每组中随机出一个人, 若交换两人会使得答案变优秀, 则交换;

**Step3:** 不停尝试**Step2**, 若多次尝试未果, 则重新开始**Step1**;

**Step4:** 多次重复以上操作, 输出答案。



## **POJ2454**

### **Description**

In the newest census of Jersey Cows and Holstein Cows, Wisconsin cows have earned three stalls in the Barn of Representatives. The Jersey Cows currently control the state's redistricting committee. They want to partition the state into three equally sized voting districts such that the Jersey Cows are guaranteed to win elections in at least two of the districts. Wisconsin has  $3 \cdot K$  ( $1 \leq K \leq 60$ ) cities of 1,000 cows, numbered  $1..3 \cdot K$ , each with a known number (range:  $0..1,000$ ) of Jersey Cows. Find a way to partition the state into three districts, each with  $K$  cities, such that the Jersey Cows have the majority percentage in at least two of districts.

All supplied input datasets are solvable.

### **Input**

Line 1: A single integer,  $K$

Lines  $2..3 \cdot K + 1$ : One integer per line, the number of cows in each city that are Jersey Cows. Line  $i + 1$  contains city  $i$ 's cow census.



## **Solution**

**Step1:** 去除最小的 $k$ 个数;

**Step2:** 把剩下的 $2k$ 个数分成两组;

**Step3:** 不停交换两组数, 直至满足题目要求为止;



## Description

给定一幅由 $n$ 个点 $m$ 条边构成的图和 $S$ 、 $T$ 两点，询问是否有从 $S$ 至 $T$ 的长度恰好为7的简单路径。

## Input

$n \leq 100$



## Solution

搜索。Maybe 可以。

折半算法？

如果范围大一些呢，这个算法还work不work？



## **Solution**

**Step1:** 对于每个点，我们随机赋予它一个1-7的颜色，并且保证S，T的颜色并不一样。

**Step2:** 找寻从S到T的通过恰好7种不一样的颜色的路径。

**Step3:** 重复多次上述过程，如果存在一次找到了路径，那么我们就找到了题目要求的路径。

## Prove

在最糟糕的情况下，只有一条从S到T的满足条件的路径。

则对于其上七个点染色的总方案数为 $7^7$ ，用我们的方法可以搜到解的方案数为 $7!$ ，则随机一次失败的概率高达 $1 - 7!/7^7 = 0.99388!$

随10次呢？失败的概率为 $(1 - 7!/7^7)^{10} = 0.94$ 。

随100次呢？失败的概率为 $(1 - 7!/7^7)^{100} = 0.54$ 。

随1000次呢？失败的概率为 $(1 - 7!/7^7)^{1000} = 0.002$ 。

随2000次呢？失败的概率为 $(1 - 7!/7^7)^{2000}$  几乎等于0。

复杂度为 $O(2^5 * n * \text{随机次数})$



## Description

给定一个由 $n$ 个数构成的环，要求你把它分成 $m$ 个连续的部分，对于每个分法 $\Sigma$ ，定义一个也许不是很简单的函数 $f(\Sigma)$ （例如每部分和的最小值的最大值，或是每部分平方和的最小值的最大值），我们假设这个函数的值可以简单计算并可以dp，现在我们询问 $f(\Sigma)$ 的极值。

## Input

$n \leq 10000$ ,  $m \leq 1000$



## Solution

枚举环中的每一个结点作为第一部分的结点，用  $f[i][j]$  表示  $dp$  的状态，枚举转移到那个  $k$ 。

复杂度  $O(n * n * m)$



## Solution

随机若干次起始点。

随机几次呢？

$n/m$ 的常数倍 $a(n/m)$ 次。

随机一次错误的概率为 $1 - m/n$ ,

总错误概率为 $(1 - m/n)^{a(n/m)} \rightarrow 0$



## Description

给定一幅由 $n$ 个点 $m$ 条边构成的无向图，现在问你是否存在一种方案，使得你删除其中 $k$ 个点和其对应的边后，剩下的点恰好构成一颗树？

## Input

$10 < n \leq 10000$ ,  $2 * n \leq m$ ,  $k < 8$



## Solution

随机一条边，然后随机选择其中一个端点（未删除过的）删除。经历此过程 $k$ 次。**Check。**

随机若干次。

## Prove

删掉的点带走的边至少到达总边数的一半。

即有至少一半的边要删掉。

即随出一条边并且他是需要删掉的概率至少为 $1/2$ 。

随出的端点是需要删掉的概率至少为 $1/2$ 。

即随出一条边一个点正确的概率至少为 $1/4$ 。

那么随机一次的错误概率为 $1 - (1/4)^k \rightarrow 1$

但是假如我们随机**10000**次，情况就大为不同了。

有一天我的室友（）买了 $n$ 箱同一品牌同一型号肥皂，每箱里面有 $n$ 块肥皂，然而其中有一箱里面的肥皂每块的重量都比标准轻一点点，已知这一点点是多少以及标准的重量是多少，问如何通过一次过秤操作确定是哪一箱有质量问题？



最小割

Dinic

Sap

随机也可以！

定义

给定一个加（正）权无向图  $G = (V, E, w)$ ，任何一个关于顶点集合  $V$  的划分  $[A, B]$  ( $V = A \cup B$ ;  $B = V - A$ ;  $A, B$  非空) 称为该图的一个割。

对于图中两个不同的顶点  $s, t \in V$ ，若  $[A, B]$  满足  $s \in A, t \in B$ ，则称  $[A, B]$  是一个  $s - t$  割。

对于任何  $[A, B]$ ，定义割的容量

$$C[A, B] = \sum_{(u,v) \in E, u \in A, v \in B} w(u, v)$$



定义**G**中的最小割为所有割[A, B]中容量最小者。给定不同的s, t ∈ V, 定义**G**中的最小s - t割为所有s - t割中的容量最小者。即

$$MinCut(G) = \min_{[A,B]} \{C(A, B)\}$$

$$MinCut_{s-t}(G) = \min_{[A,B], s \in A, t \in B} \{C(A, B)\}$$

$$MinCut(G) = \min_{s \neq t \in G} \{MinCut_{s-t}(G)\}$$

## Solution

**Step1:** 在 $G$ 中随机选择一条边 $(u, v)$ ;

**Step2:** 收缩 $(u, v)$ , 保留重边, 但删除自环。即如果图中存在两条边 $(u, y)$ 和 $(v, y)$ , 在收缩后这两条边都被保留;

**Step3:** 如果图中还有多于2个顶点, 返回(1)进行循环, 否则输出两个点所代表的集合, 作为最小割 $[A, B]$ ;

**Step4:** 将上述操作进行 $cn^2 \log(n)$ 次, 取最值。

**Prove**

**Idea1:** 假设原图的最小割为 $[A0, B0]$ , 那么在算法中, 只要不选择跨越 $[A0, B0]$ 的边进行收缩,  $[A0, B0]$ 将一直是一个合法的割。

**Idea2:** 在算法的循环过程中, 新图的任何一个割都一定是原图的割(把新图中的点展开即可)。因而在算法执行过程中, 图中的最小割容量一定不会减少。

**Idea3:** 假设原图中最小割的容量是 $k$ , 那么在算法执行过程中, 新图的最小割容量至少也是 $k$ 。于是新图中每一个点的度至少是 $k$ , 假设这是第 $i(1 \leq i < n - 1)$ 次循环, 新图中有 $(n - i + 1)$ 个顶点, 那么图中至少有 $(n - i + 1)k / 2$ 条边。

**Idea4:** 于是, 确定原图中的一个最小割 $C[A0, B0] = k$ , 在算法循环的第 $i$ 轮中,  $[A0, B0]$ “存活”下来的概率至少是

$$1 - \frac{k}{(n - i + 1)k / 2} = \frac{n - i - 1}{n - i + 1}$$

那么在算法结束时（第**(n - 1)**轮后），**[A0, B0]**作为输出的概率至少是

$$\begin{aligned} & \prod_{i=1}^{n-2} \frac{n-i-1}{n-i+1} \\ &= \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdots \frac{2}{4} \cdot \frac{1}{3} \\ &= \frac{2}{n(n-1)} \end{aligned}$$

运行一次简单算法失败的概率最多是 $1 - 2/(n^2)$ ，那么运行 $cn^2 \log(n)$ 次该算法，失败的概率最多为

$$\begin{aligned} & \left(1 - \frac{2}{n^2}\right)^{cn^2 \ln n} \\ & \leq e^{-2c \ln n} \\ & = \frac{1}{n^{2c}} \end{aligned}$$

## AHOI2013 连通图

### Description

给定一个连通的无向图和若干个小集合，每个小集合包含一些边。对于每个集合，你需要确定将集合中的边从原来的无向图中删除后该图是否保持连通。一个图是连通的当且仅当任意两个不同的点之间存在一条路径连接他们。

### Input

输入的第一行包含两个整数 $n$ 和 $m$  ( $1 \leq n \leq 100000, 1 \leq m \leq 200000$ )，表示无向图的点数和边数，每个点从1到 $n$ 标号。

接下来的 $m$ 行表示图的每条边，每行包含两个整数 $a$ 和 $b$ ——一条边连接的两个端点的标号。保证每对顶点最多被一条边连接。没有一条边连接两个相同的顶点。每条边按照输入的顺序标号为1到 $m$ 。

接下来的一行包含一个整数 $k$  ( $1 \leq k \leq 100000$ )，表示需要测试的小集合的个数，接下来的 $k$ 行每行描述一个小集合。每行的第一个数 $c$  ( $1 \leq c \leq 4$ ) 表示集合中边的个数，接下来有 $c$ 个整数表示集合中边的标号，保证集合中的整数互不相同。





## Solution

**Step1:** 随便建立一颗生成树；（**simple**, 并查集 is ok）

**Step2:** 对于每一条不在生成树上的边，随机一个**64**位整数作为这条边的权值；（**simple**）

**Step3:** 对于每条在生成树上的边，定义它的权值为连接他的两个端点对应部分的所有非树边的权值的**xor**和；（**not too difficult**）

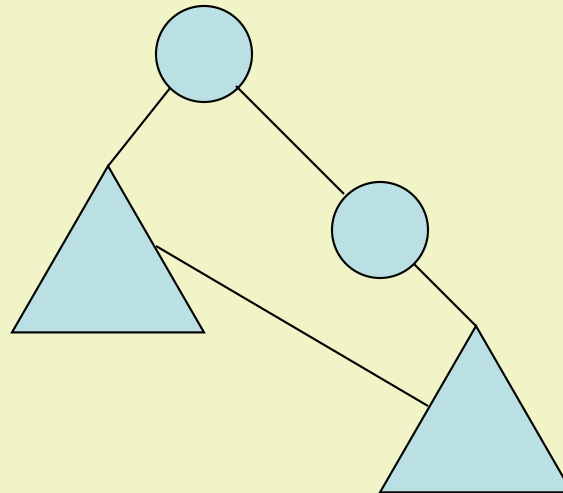
**Step4:** 对于每一组询问，枚举询问中边的子集，如果某个子集所对应的边的权值**xor**和等于**0**，则我们可以断言，去掉这些边以后图不连通；否则，去掉这些边不影响图的连通性。（**simple**）



## Prove

**Idea1:** 直观上讲，一些64位的随机数的xor和在他不应该等于0的情况下等于0的概率与中彩票的概率差不多；

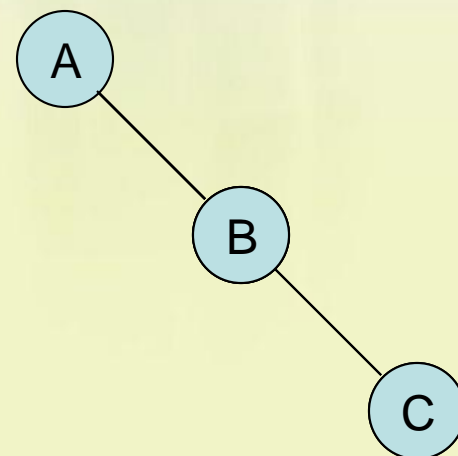
**Idea2:** 假设我们在只删除一条树边的情况下想要使图不连通，那么我们必须删除所有连接这条边两个端点相应部分的边，而我们刚刚定义的操作恰恰满足了这个条件。

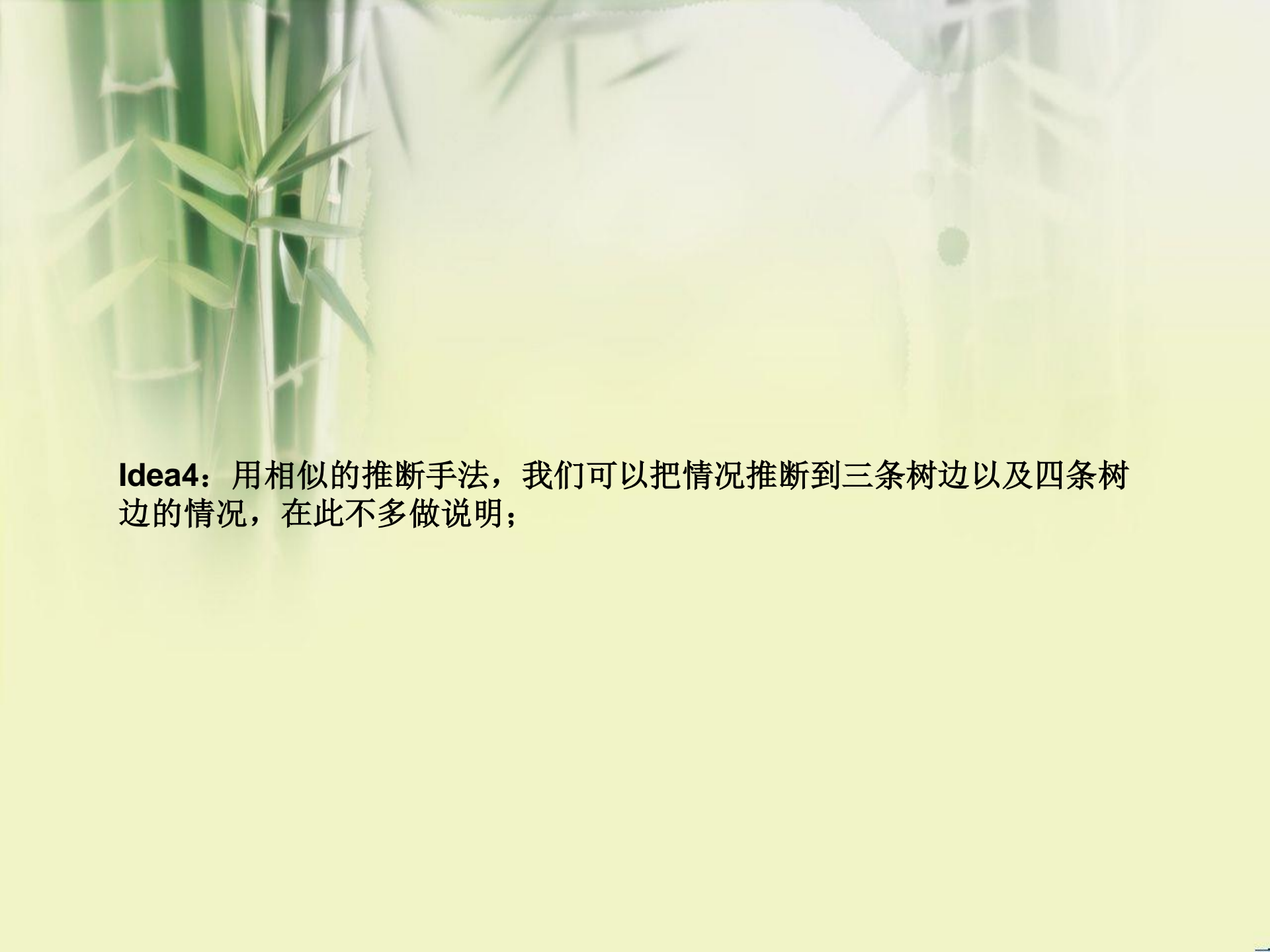


**Idea3:** 现在来考虑两条树边的情况，  
不妨假设两条树边相连，  
若**TA**与**TC**之间没有边相连，  
那么下列两种情况必然出现一个：

- 1、枚举边**AB**、**BC**时图不连通；
- 2、**TA**与**TB**、**TB**与**TC**之间均有边，  
则删除这两条树边不影响连通性；  
否则，**TA**与**TC**之间有边相连，

那么根据**xor**的性质，对**AB**与**BC**的权值进行**xor**的过程中，**TA**与**TC**之间的边的权值两两抵消；则问题又变成了**TA**与**TB**、**TB**与**TC**是否有边相连





**Idea4:** 用相似的推断手法，我们可以把情况推断到三条树边以及四条树边的情况，在此不多做说明；

## Problem Description

At the children's day, the child came to Picks's house, and messed his house up. Picks was angry at him. A lot of important things were lost, in particular the favorite sequence of Picks.

Fortunately, Picks remembers how to repair the sequence. Initially he should create an integer array  $a[1], a[2], \dots, a[n]$ . Then he should perform a sequence of  $m$  operations.

An operation can be one of the following:

- 1、Print operation  $l, r$ . Picks should write down the value of  $\text{sum}(l, r)$ .
- 2、Modulo operation  $l, r, x$ . Picks should perform assignment  $a[i] = a[i] \bmod x$  for each  $i$  ( $l \leq i \leq r$ ).
- 3、Set operation  $k, x$ . Picks should set the value of  $a[k]$  to  $x$  (in other words perform an assignment  $a[k] = x$ ).

Can you help Picks to perform the whole sequence of operations?

## Input

The first line of input contains two integer:  $n, m$  ( $1 \leq n, m \leq 10^5$ ). The second line contains  $n$  integers, separated by space:  $a[1], a[2], \dots, a[n]$  ( $1 \leq a[i] \leq 10^9$ ) — initial value of array elements.

Each of the next  $m$  lines begins with a number *type*.

If *type* = 1, there will be two integers more in the line:  $l, r$  ( $1 \leq l \leq r \leq n$ ), which correspond the operation 1.

If *type* = 2, there will be three integers more in the line:  $l, r, x$  ( $1 \leq l \leq r \leq n; 1 \leq x \leq 10^9$ ), which correspond the operation 2.

If *type* = 3, there will be two integers more in the line:  $k, x$  ( $1 \leq k \leq n; 1 \leq x \leq 10^9$ ), which correspond the operation 3.



感谢聆听~