题目概述

题目名称	最小生成链	穿越	树统计
源文件名	`msc.c/cpp/pas`	`across.c/cpp/pas`	`count.c/cpp/pas`
输入输出文件名	`msc.in/out`	`across.in/out`	`count.in/out`
时间限制	1 sec	2 sec	2 sec
空间限制	512 MB	512 MB	512 MB
题目类型	传统	传统	传统
是否开启 O2 优化	是	是	是

A. 最小生成链(msc)

题目描述

定义一张图的生成链是原图的一棵生成树，且这棵树退化成一条链。我们称一条生成链是原图的最小生成链，当且仅当它当中边权最大的边是原图的所有生成链中最小的。

现有一个 n 个点的完全图，点编号为 1 到 n。另给出一个长度为 n 的序列 a_i，完全图中第 i 个点与第 j 个点间的边的边权为 $a_i \oplus a_j$，其中 $\oplus$ 表示异或运算。

请你找出该完全图的最小生成链。但由于答案可能很多，你只需要输出这条最小生成链中边权最大的边的边权即可。

输入输出格式

输入格式

从文件 `msc.in` 中读入数据。

输入文件将严格遵循以下格式：

> n
> $a_1 \quad a_2 \quad \cdots \quad a_n$

第一行输入一个正整数 n $(2 \leq n \leq 2 \cdot 10^5)$，表示完全图的点数。
第二行输入 n 个非负整数 a_i $(0 \leq a_i < 2^{60})$，表示这个序列。

输出格式

输出到文件 `msc.out` 中。

输出文件请严格遵循以下格式：

> w

输出一行一个非负整数 w，表示这条生成链中边权最大值。

样例输入和输出

样例输入 1

```
3
1 2 3
```

样例输出 1

```
2
```

样例解释 1

最小生成链为 $1 \leftrightarrow 3 \leftrightarrow 2$，其中 $1 \leftrightarrow 3$ 这条边边权为 $1 \text{ xor } 3 = 2$，$3 \leftrightarrow 2$ 这条边边权为 $3 \text{ xor } 2 = 1$。容易证明不存在最大边权更小的生成链。

样例输入 2

```
4
9 10 13 14
```

样例输出 2

```
4
```

样例输入 3

```
2
1 1
```

样例输出 3

```
0
```

样例 4

见附加文件中的 `msc4.in` 与 `msc4.out` 。

子任务

本题使用子任务作为计分规则。对于一个子任务，你的程序需要通过该子任务下所有数据才能得到该子任务对应的分数。

Subtask #1 (20 points) ：$n \leq 8$。
Subtask #2 (20 points) ：$n \leq 17$。
Subtask #3 (20 points) ：$n \leq 1\,000$。
Subtask #4 (20 points) ：$a_i \in [0, 1]$。
Subtask #5 (20 points) ：无特殊性质。

B. 穿越(across)

题目描述

就在不久前，Z 国科学家发明了空间穿越器，它能快速地到达指定地点，但是它也有一个很大的缺陷，只要目的地被设定之后就再也不能更改目的地。而现在，空间穿越器已经被投入使用。

Z 国有一座首都，除此之外还有 n 座城市，为了便于描述，我们把首都编号为 0，其它城市从 1 编号到 n。由于空间穿越器成本高昂，每座城市只能使用至多两座空间穿越器，而首都城市作为 Z 国的行政中心，所有城市必须要有一座空间穿越器通向首都（包括首都）。除此之外，每座城市还会有一个空间穿越器通向 x_i 号城市。Z 国不会让另一座空间穿越器也通向首都，但不保证空间穿越器不会通向其所在城市。空间穿越器只负责将指定的东西送至目的地，并不能将目的地的东西接回来。

空间穿越器非常大地方便了 Z 国交通，同时也促进了 Z 国旅游业的发展。但是不久以后，旅行者们发现了问题：由于空间穿越器是单向的，这就导致他们不一定能很方便地回到自己的城市。为了调查这样的情况，Z 国国主需要你告诉他经过 m 次穿越，分别从 Z 国的 $(n+1)$ 座城市出发回到出发地有多少种穿越方法，两种方法不同当且仅当某一次穿越使用的空间穿越器不同。

经过 Z 国计算学会一个月的研究，他们发现答案可能很大，于是 Z 国国主只要你将答案模 998244353 后输出即可。

输入输出格式

输入格式

从文件 `across.in` 中输入数据。

第一行两个自然数 n,m，表示 Z 国首都之外的城市数量以及穿越次数。
第二行 $(n+1)$ 个正整数 x_i ，表示第 i 座城市其中一台空间穿越器的目的地。

输出格式

从文件 `across.out` 输出数据。

输出一行 $(n+1)$ 个自然数，表示从第 i 座城市出发经过 m 次穿越回到第 i 座城市的方案数。

样例输入和输出

样例输入 1

```
3 3
1 2 3 1
```

样例输出 1

```
4 3 2 1
```

样例输入 2

```
5 5
1 1 1 1 1
```

样例输出 2

```
16 16 0 0 0 0
```

样例输入 3

```
20 25
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 1
```

样例输出 3

```
16777216 8388616 4194308 2097154 1048577 524288 262144 131072 65536 32768 16384 8192
4096 2048 1024 512 256 128 64 32 16
```

样例 4

见选手文件下的 `across4.in` / `across4.ans` 。

数据范围

测试点编号	$n \leq$	$m \leq$	特殊性质
1	15	15	对于所有的 x_i , x_i 在 1 至 n 中随机生成
2	15	15	无特殊性质
3	30	30	对于所有的 x_i , x_i 在 1 至 n 中随机生成
4	30	30	无特殊性质
5	50	30	对于所有的 x_i , x_i 在 1 至 n 中随机生成
6	50	30	无特殊性质
7	100	100	对于所有的 x_i , x_i 在 1 至 n 中随机生成
8	100	100	无特殊性质
9	200	200	对于所有的 x_i , x_i 在 1 至 n 中随机生成
10	200	200	无特殊性质
11	300	200	对于所有的 x_i , x_i 在 1 至 n 中随机生成
12	300	200	无特殊性质
13	200	2000	对于所有的 x_i , x_i 在 1 至 n 中随机生成
14	200	2000	无特殊性质
15	2000	200	对于所有的 x_i , x_i 在 1 至 n 中随机生成
16	2000	200	无特殊性质
17	1000	1000	对于所有的 x_i , x_i 在 1 至 n 中随机生成
18	1000	1000	无特殊性质
19	1500	1500	对于所有的 x_i , x_i 在 1 至 n 中随机生成
20	1500	1500	无特殊性质
21	2000	2000	对于所有的 x_i , x_i 在 1 至 n 中随机生成
22	2000	2000	无特殊性质
23	4000	3000	对于所有的 x_i , x_i 在 1 至 n 中随机生成
24	4000	3000	无特殊性质
25	3000	4000	对于所有的 x_i , x_i 在 1 至 n 中随机生成
26	3000	4000	无特殊性质
27	5000	5000	对于所有的 x_i , x_i 在 1 至 n 中随机生成
28	5000	5000	无特殊性质
29	5000	5000	所有的 x_i 相等
30	5000	5000	对于所有的 $1 \leq i \leq n$,都有$x_i = i$

测试点编号	$n \leq$	$m \leq$	特殊性质
31	50000	50000	对于所有的 x_i , x_i 在 1 至 n 中随机生成
32	50000	50000	无特殊性质
33	100000	100000	对于所有的 x_i , x_i 在 1 至 n 中随机生成
34	100000	100000	无特殊性质
35	100000	100000	所有的 x_i 相等
36	100000	100000	对于所有的 $1 \leq i \leq n$,都有$x_i = i$
37	500000	500000	对于所有的 x_i , x_i 在 1 至 n 中随机生成
38	500000	500000	无特殊性质
39	200	100000	无特殊性质
40	200	100000	无特殊性质
41	200	100000	无特殊性质
42	200	100000	无特殊性质
43	200	1000000	无特殊性质
44	200	1000000	无特殊性质
45	200	1000000	无特殊性质
46	200	1000000	无特殊性质
47	1000000	1000000	对于所有的 x_i , x_i 在 1 至 n 中随机生成
48	1000000	1000000	无特殊性质
49	1000000	1000000	所有的 x_i 相等
50	1000000	1000000	对于所有的 $1 \leq i \leq n$,都有$x_i = i$

对于所有的数据，保证 $1 \leq x_i \leq n$ 。

C. 树统计(count)

题目描述

有一棵 n 个节点的树，以 1 号点为根。每个点有一个整数点权 a_i，点权可能为正也可能为负。

我们定义一棵树是美妙的，当且仅当这棵树所有点的点权都是非负整数。

原树不一定是美妙的。我们希望通过施加一些魔法使得这棵树成为一棵美妙的树。

一次魔法可任意选取选取一个点 u $(2 \leq u \leq n)$ 和一个正整数 w，设 u 点父亲节点是 f，那么进行一次魔法后，可以使得 u 点点权加上 w，而使 f 点点权减去 w。

我们需要你回答是否可能通过施加若干次魔法使这棵树成为一棵美妙的树。

相信你已经会做这道题了。但是本题作为此次比赛的压轴，肯定是不会这么简单的。

我们让这棵树进行 m 次变化，每次变化会指定一个点 x 和一个整数 w，这里 w 可能是正整数、零或负整数。意为将点 x 的点权加上 w。

你需要在每一次变化后回答是否可能通过施加魔法使这棵树称为一棵美妙的树。

注意：每次变化是永久有效的，前面对点权的变化会直接修改原树的点权。而每次询问时只询问是否可能，并不真正施加魔法。详见样例解释。

输入输出格式

输入格式

从文件 `count.in` 中读入数据。

输入文件将严格遵循以下格式：

$$
\begin{array}{ll}
n & \\
a_1 & a_2 \quad \cdots \quad a_n \\
u_1 & v_1 \\
u_2 & v_2 \\
\vdots & \\
u_{n-1} & v_{n-1} \\
m & \\
x_1 & w_1 \\
x_2 & w_2 \\
\vdots & \\
x_m & w_m
\end{array}
$$

第一行输入一个正整数 n $(2 \leq n \leq 10^5)$，表示树的点数。
接下来输入一行 n 个整数 a_i $(|a_i| \leq 10^9)$，表示每个点的初始点权。
接下来 $n-1$ 行，每行输入两个正整数 u_i, v_i $(1 \leq u_i < v_i \leq n)$，描述这棵树。
接下来一行输入一个非负整数 m $(0 \leq m \leq 10^5)$，表示树的变化次数。
接下来 m 行，每行输入两个整数 x_i, w_i $(1 \leq x_i \leq n, |w_i| \leq 10^9)$，表示将点 x_i 的点权加上 w_i。

输出格式

输出到文件 `count.out` 中。

输出文件请严格遵循以下格式：

> ans_0
> ans_1
> ans_2
> $\vdots$
> ans_m

第一行输出一个字符串 ans_0，表示原树是否可能通过施加魔法使这棵树称为一棵美妙的树。若可能，请输出 `Yes`；否则输出 `No`。
接下来 m 行，每行输出一个字符串 ans_i，表示当前的树是否可能通过施加魔法使这棵树称为一棵美妙的树。若可能，请输出 `Yes`；否则输出 `No`。

样例输入和输出

样例输入 1

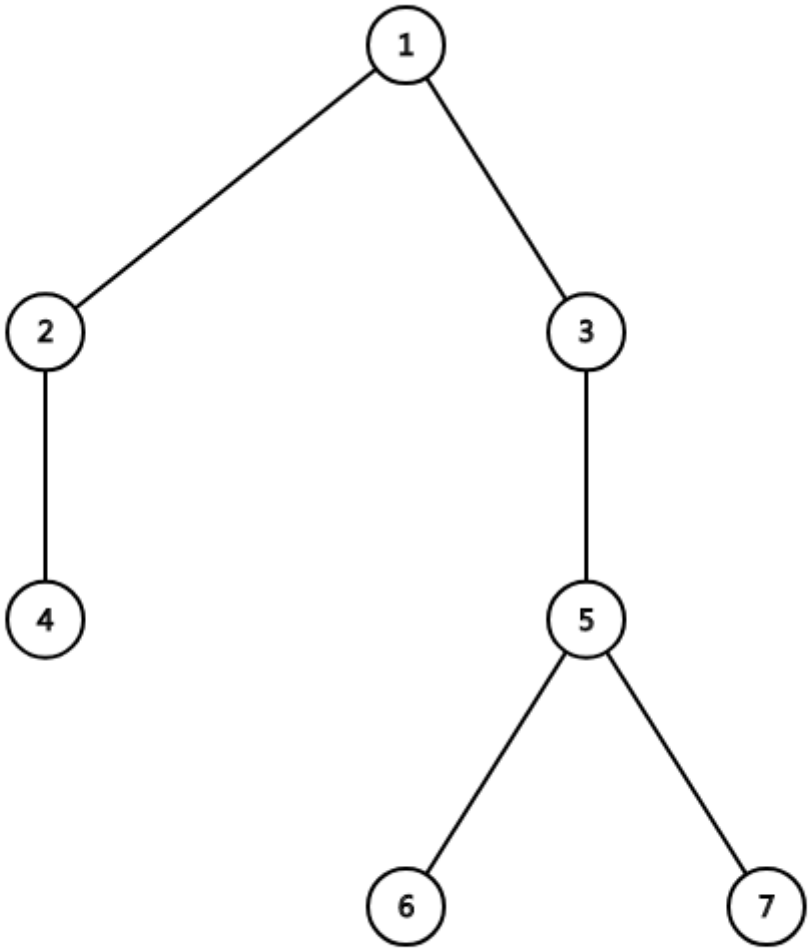
```
7
7 1 -1 -3 1 0 1
1 2
1 3
2 4
3 5
5 6
5 7
4
2 -4
4 -2
1 1
3 1
```

样例输出 1

```
Yes
Yes
No
No
Yes
```

样例解释 1

树的形态如下：



在原树上，可以选择 $u=4, w=1$，使得 2 号点点权为 0，4 号点点权变为 0。再选择 $u=3, w=4$，使得 3 号点点权变为 1，1 号点点权变为 3。至此，树中所有点点权均为非负整数，是美妙的。

树第一次变化后，点权为 $(7,-3,-1,-3,1,0,1)$。可以选择 $u=2, w=4$，使得 2 号点点权变为 1，1 号点点权变为 3。再选择 $u=3, w=3$，使得 3 号点点权变为 0，1 号点点权变为 0。再选择 $u=4, w=1$，使得 2 号点点权为 0，4 号点点权变为 0。至此，树中所有点点权均为非负整数，是美妙的。

树第二次变化后，点权为 $(7,-3,-1,-5,1,0,1)$。可以证明没有方案使得这棵树成为美妙的。

树第三次变化后，点权为 $(8,-3,-1,-5,1,0,1)$。可以证明仍然没有方案使得这棵树成为美妙的。

树第四次变化后，点权为 $(8,-3,0,-5,1,0,1)$。可以选择 $u=2, w=3$，使得 2 号点点权变为 0，1 号点点权变为 5。再选择 $u=4, w=5$，使得 4 号点点权变为 0，1 号点点权变为 0。至此，树中所有点点权均为非负整数，是美妙的。

样例 2

见附加文件中 `count2.in` 与 `count2.out`。这组样例满足 ***Subtask* #4** 的特殊性质——对于第 i 条边，$u_i=1, v_i=i+1$。

样例 3

见附加文件中 `count3.in` 与 `count3.out`。这组样例满足 ***Subtask* #5** 的特殊性质——m 次变化中，w_i 为正数的次数不超过 400 次。

样例 4

见附加文件中 `count4.in` 与 `count4.out`。

子任务

本题使用子任务作为计分规则。对于一个子任务，你的程序需要通过该子任务下所有数据才能得到该子任务对应的分数。

Subtask #1 (12 points) ：$n, m \leq 100$。
Subtask #2 (16 points) ：$n, m \leq 2\,000$。
Subtask #3 (20 points) ：$m = 0$。
Subtask #4 (20 points) ：对于第 i 条边，$u_i = 1, v_i = i + 1$。
Subtask #5 (20 points) ：m 次变化中，w_i 为正数的次数不超过 400 次。
Subtask #6 (12 points) ：无特殊性质。