博弈论

<[891. Nim游戏 - AcWing题库](#)>

给定 n 堆石子，两位玩家轮流操作，每次操作可以从任意一堆石子中拿走任意数量的石子（可以拿完，但不能不拿），最后无法进行操作的人视为失败。

问如果两人都采用最优策略，先手是否必胜。

输入格式

第一行包含整数 n。

第二行包含 n 个数字，其中第 i 个数字表示第 i 堆石子的数量。

输出格式

如果先手方必胜，则输出 `Yes`。

否则，输出 `No`。

数据范围

$1 \leq n \leq 10^5$,
$1 \leq$ 每堆石子数 $\leq 10^9$

分析见下：

证明

- 操作到最后时，每堆石子数都是0，$0 \oplus 0 \oplus \ldots 0 = 0$
- 在操作过程中，如果 $a_1 \oplus a_2 \oplus \ldots \oplus a_n = x \neq 0$。那么玩家必然可以通过拿走某一堆若干个石子将异或结果变为0。
  证明：不妨设x的二进制表示中最高一位1在第k位，那么在$a_1, a_2, \ldots, a_n$中，必然有一个数a_i，它的第k为时1，且 $a_i \oplus x < a_i$，那么从第i堆石子中拿走$(a_i - a_i \oplus x)$个石子，第i堆石子还剩$a_i - (a_i - a_i \oplus x) = a_i \oplus x$，此时 $a_1 \oplus a_2 \oplus \ldots \oplus a_i \oplus x \oplus \ldots \oplus a_n = x \oplus x = 0$。
- 在操作过程中，如果 $a_1 \oplus a_2 \oplus \ldots \oplus a_n = 0$，那么无论玩家怎么拿，必然会导致最终异或结果不为0。
  反证法：假设玩家从第i堆石子拿走若干个，结果仍是0。不妨设还剩下a'个，因为不能不拿，所以$0 \leq a' < a_i$，且 $a_1 \oplus a_2 \oplus \ldots \oplus a' \oplus \ldots \oplus a_n = 0$。那么
  $(a_1 \oplus a_2 \oplus \ldots \oplus a_i \oplus \ldots a_n) \oplus (a_1 \oplus a_2 \oplus \ldots \oplus a' \oplus \ldots \oplus a_n) = a_i \oplus a' = 0$，则 $a_i = a'$，与假设 $0 \leq a' < a_i$矛盾。

基于上述三个证明：
1. 如果先手面对的局面是$a_1 \oplus a_2 \oplus \ldots \oplus a_n \neq 0$，那么先手总可以通过拿走某一堆若干个石子，将局面变成$a_1 \oplus a_2 \oplus \ldots \oplus a_n = 0$。如此重复，最后一定是后手面临最终没有石子可拿的状态。先手必胜。
2. 如果先手面对的局面是$a_1 \oplus a_2 \oplus \ldots \oplus a_n = 0$，那么无论先手怎么拿，都会将局面变成 $a_1 \oplus a_2 \oplus \ldots \oplus a_n \neq 0$，那么后手总可以通过拿走某一堆若干个石子，将局面变成 $a_1 \oplus a_2 \oplus \ldots \oplus a_n = 0$。如此重复，最后一定是先手面临最终没有石子可拿的状态。先手必败。

AC代码:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n ;
    int ans = 0 ;
    cin >> n ;
    for(int i = 0 ; i < n ; i ++ )
```

```
    {
        int x ;
        cin >> x;
        ans ^= x ;
    }
    if(ans ) puts("Yes");
    else puts("No") ;
    return 0 ;
}
```

例题2

<893. 集合-Nim游戏 - AcWing题库>

给定 n 堆石子以及一个由 k 个不同正整数构成的数字集合 S。

现在有两位玩家轮流操作，每次操作可以从任意一堆石子中拿取石子，每次拿取的石子数量必须包含于集合 S，最后无法进行操作的人视为失败。

问如果两人都采用最优策略，先手是否必胜。

输入格式

第一行包含整数 k，表示数字集合 S 中数字的个数。

第二行包含 k 个整数，其中第 i 个整数表示数字集合 S 中的第 i 个数 s_i。

第三行包含整数 n。

第四行包含 n 个整数，其中第 i 个整数表示第 i 堆石子的数量 h_i。

输出格式

如果先手方必胜，则输出 `Yes`。

否则，输出 `No`。

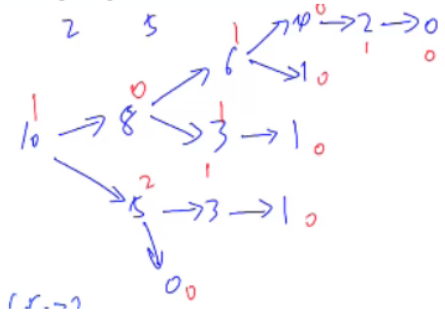
数据范围

$1 \le n, k \le 100$,
$1 \le s_i, h_i \le 10000$

前置知识:

将每一个$h[i]$的所有方案看做是一张有向图，例

若$S=[2,5],h=10$,则有如下展开形式：



$mex()$：设集合S是一个非负整数集合，定义$mex(S)$为求出不属于S的最小非负整数的运算，即：
$mes(S)=min[x]$,其中x属于自然数，且x不属于S**（用人话说就是不存在S集合中的数中，最小的那个数）**

$SG()$：在有向图中，对于每个节点x，设从x出发共有k条有向边，分别达到节点y_1，$y_2\ldots\ldots y_k$，定义$SG(x)$为x的后继节点的SG值构成的集合执行$mex()$运算后的值
即：$SG(x)=mex(SG(y_1),SG(y_2)\ldots SG(y_k))$；（用人话说就是比后继节点的$SG$都小的值）
特别的整个图G的SG值被定义为起点s的SG值，即$SG(G)=SG(s)$
上图标红的值就是每一个节点的SG值
性质：$1.SG(i)=k$，则i最大能到达的点的SG值为$k-1$。
$2.$非0可以走向0
3.0只能走向非0

定理：
对于一个图G，如果$SG(G)!=0$，则先手必胜，反之必败

证明：
若$SG(G)=!0$,
1.根据性质2，先手必可以走向0，
2.因此留给后手的是0，根据性质3，后手只能走向非0
3.以此类推，后手始终无法走向0，后手永远处于非0，当先手到达终点的0时，先手获胜
（由此我们可以知道，有些事是命中注定的~~~）
反之同理，必败

定理：
对于n个图，如果$SG(G_1)$ ^ $SG(G_2)$ ^ ... $SG(G_n)!=0$,则先手必胜，反之必败

证明（类似与Nim游戏）：
①当$SG(G_i)=0$ 时， $xor=0$， 显然先手必败
（PS：结束状态必是状态①，但状态①不一定是结束状态）

②当$xor=x!=0$ 时，因为肯定存在一个$SG(x_i)$^x $< SG(x_i)$,而根据$SG()$的性质1可知，$SG(k)$可以走到$0-k-1$的任何一个状态，
因此，必定可以从$SG(x_i)->SG(x_i)$^x ， 于是使得$xor=0$

③当$xor=0$时，当移动任何一个节点时，对应的SG值必然减小，可以证明：$xor!=0$
下证：$xor!=0$
假设:$xor=0$,则说明移动的那个节点的值并没有变化，即从$SG(k)$变成了k，但是这与SG函数的性质1相矛盾，因此不成立

证得：若先手面对的状态是$xor!=0$,则先手方总能使$xor=0$，即使后手面对的永远是必败态直到结束状态①，因此先手必胜！
反之，必败！

对于性质1 ， 2 ， 3 的解释：

1. 若 `SG(i) = k` , 则 `i` 最大能到达的SG值为 `k - 1` .
   - 若SG(i) = k , 那么一定可以走到0 ~ k - 1 的任一个值，但大于k的值可能走到也可能走不到。因为SG(i)的值是根据其子节点的SG值得来的。
2. 非0可以走向0， 如果一个点的SG值不为零，说明其后继节点的SG值一定有一个是0.
3. 零只能走向非零，反证法：如果其后继节点有0，那么根据mex运算的性质当前点的SG的值一定不是0。

AC代码

```
#include <bits/stdc++.h>
using namespace std;
const int N = 10010 ;
int s[N] , f[N]  ; // f[x] 是一个记忆化搜索 ，能够有效地降低时间复杂度
int n , m ;
int SG(int x )
{
    if(f[x] != -1 ) return f[x] ;
    unordered_set<int> S ;
    for(int i = 0; i < m ; i ++ ) // 通过递归实现对其子节点的SG值的插入。
    {
        int sum = s[i] ;
        if(x >= sum ) S.insert(SG(x - sum) ) ;
    }
    for(int i = 0 ; ; i ++ )  // 实现mex() 运算
    {
        if(!S.count(i))
        {
            f[x] = i ;
            return f[x] ;
        }
    }
}
```

```cpp
int main()
{
    cin >> m;
    for(int i = 0 ; i < m ; i ++ ) cin >> s[i] ;
    cin >> n ;
    int res = 0 ;
    memset(f , -1 , sizeof f ) ;
    for(int i = 0 ; i < n ; i ++ )
    {
        int x ;
        cin >> x ;
        res ^= SG(x) ;
    }
    if(res) puts("Yes") ;
    else puts("No") ;
    return 0 ;
}
```

博弈论知识补充:

如果有一堆石子数为n的石子，每一次能从其中取`1 ~ m`个石子，那么如果：

- 若 n % (m + 1) == 0 ，那么先手必败。
- 若 n %(m + 1) != 0　那么先手必胜。

例题三

Alice 和 Bob 在玩一个游戏。

首先，给定一个长度为 n 的正整数数列 $a_1, a_2, \ldots, a_n$。

随后，两人轮流展开行动，由 Alice 先手行动。

当轮到一人采取行动时，如果 $a_1 = 0$，则该玩家输掉游戏，否则该玩家需要：

1. 在 $[2, n]$ 范围内选择一个整数 i。
2. 将 a_1 的值减少 1。
3. 交换 a_1 和 a_i 的值。

假设双方都采取最优策略，请你判断谁将获胜。

输入格式

第一行包含整数 T，表示共有 T 组测试数据。

每组数据第一行包含整数 n。

第二行包含 n 个整数 $a_1, a_2, \ldots, a_n$。

输出格式

每组数据输出一行结果，如果 Alice 获胜，则输出 `Alice`，如果 Bob 获胜，则输出 `Bob`。

数据范围

前 3 个测试点满足 $1 \le T \le 10$，$2 \le n \le 3$。
所有测试点满足 $1 \le T \le 2 \times 10^4$，$2 \le n \le 10^5$，$1 \le a_i \le 10^9$，一个测试点的所有的 n 相加之和不超过 2×10^5。

思路:

最小值是最快变成 0 的，如果第一个元素是最小值则先手必败，否则后手必败

若 第一个元素 是 最小值 且不是 0 (是 0 直接输了，没必要讨论):

- 则先手 $Alice$ 减 1 后，不管与谁交换值，交换后都比之前大， 由于第一个元素不是 0 , 所以交换后轮到 Bob 也必然不是 0。而 Bob 只需要做一件事，不断将最小值扔回给 $Alice$ ， 则 $Alice$ 必然最先达到 0， 必败。

若第一个元素不是最小值

- 则 $Alice$ 在减 1 后， 直接选择最小值扔给 Bob，转变为 Bob 先手拿到最小值，则转变上面那种情况，必败。

AC代码:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int T ;
    cin >> T;
    while(T -- )
    {
        int n;1
        cin >> n ;
        int minv = 2e9  ;
        vector<int> a(n) ;
        for(int i = 0 ; i < n ; i ++ )
        {
            cin >> a[i] ;
            minv = min(a[i] , minv ) ;
        }
        if(minv == a[0] ) puts("Bob") ;
        else puts("Alice") ;
    }
    return 0 ;
}
```