#A、移动硬币

给定 n 枚硬币的位置（第 i 枚位置为 pi）与 n，对任意一枚 i 可做如下操作：

1. **pi+=2 或-=2，代价为 0**
2. **Pi+=1 或-=1，代价为 1**

要求求出将 n 枚硬币集中到一个格子所需的最小代价。

思路：

读懂题目后，会发现若以第一种方式移动不需要代价，但移动前后的位置奇偶性相同。那**如果要改变奇偶性，就必须付出代价。**

因为最开始各个棋子的奇偶性不尽相同，所以**如果只使用第一种操作，最多只能将其变为奇偶性不同的两堆。**

但题目要求最后都要到一个格子啊！

但我们可以先将它们全都用第一种方法变为两堆，再用第二种方法变为奇偶性相同的一堆。

那么是那堆变到那堆呢？

肯定是少的到多的啊！

又因为这两堆的数量即是位置为奇数的棋子数量和位

置为偶数的棋子数量，所以，这题就可以看作成一道统计奇数偶数的问题，统计奇数个数和偶数个数，再比较最小值，输出遍可以 AC。

#B、2048 游戏

给出 t，即为 t 组数据，对每组数据：给出 n 个 2 的幂次（int 范围内）和 n，可做如下操作：将两相同数拿出，将它们的和放入。求经过若干次操作能否得到 2048，可以输出“YES”，否则输出“NO”。

思路

1.创建一个数组用于存 2^n 的数量，每输入一个 2^n 就把数组 n 位置的数加一，最后一直让 i 循环到 11 让 **cnt[i]=cnt[i-1]/2**，如果 cnt[11]>0 则说明可以得到 2048，否则就是不能。

核心代码

```
while(k<=11){
    cnt[k]+=cnt[k-1]/2;
    k++;
}
```

2.也可使用递归实现该操作。

想想看，若要得到 2 的 n 次方，就要得到 2 个 2

的 n-1 次方，这就是可以递归的地方。

对于每一个布尔型函数，查找给出的要找的数，若找到，将其归 0（防止重复），并返回真，否则返回 **函数（查找数/2） && 函数（查找数/2）**；

那这样不就是无限循环了吗？（到最后会无限查找 0）

所以要添加结束条件，因为是 2 的幂次，所以至少为 1。那如果某函数的参数为 0，那就肯定找不到了，就返回假。

#c 修剪草地

给出某人在平面直角坐标系中的移动过程，每个数由方向和距离构成。

并且时间从 0 开始，每走一格便增加一个单位时间

经过的地方即被割草，但是草经过 x 个单位时间会重新长出（x 不一定）。

已知该人不会经过没草的地方，求 x 的最大值，若无上限，输出“-1”。

思路：

重要的是“不会经过没有草的格子”，意思就是：如果有重复经过的格子，那那边的草一定已经长出来了，故 **x <= 两次经过时间之差的绝对值**。

那如果有 n(n >= 0)个格子被重复经过，即有 n 个不等式，若 n 不为 0，**n 应该小于最小的差**，若 n =

0，即没有经过重复的格子，那 x 无法确定，即为-1。

那如何知道有没有重复和经过时的时间呢?

二维数组是个好东西。

该人最多走 1000 格，由于在中间开始，故数组的行列至少为 2000，于是便可在经过的格子上标记时间，若遇到不为 0 的格子就说明重复经过。

#D 矩阵游戏

有一个 n*m 的矩阵，在其中任意一个 h*w 的矩阵中找最大值。要求给出 h，w，使其中的最大值可以确定，并且 h*w 的值最小。

思路：

“使其中最大值可以确定”，因为一定有若干个矩阵中的最大值为整个矩阵的最大值，**所以要让任意 h*w 的矩阵包含最大值。**

那如何确定要多大的 h*w 才能保证一定包含某一个元素呢？

我们设该元素到第 0 行，第 n - 1 行，第 0 列，第 m - 1 列（包含该元素）的距离为 a1,a1,b1,b2。**那么既然要包含该行，就要考虑离该行的最远距离（只考虑行）**，即 a1，a2 中大的一个，作为 h，那么列同理，即得到 w。

那么如果遇到多个最大值怎么办呢？

我们可以存储目前的最大值和该最大值对应的 h*w，**若遇到一样大的，则比较 h*w，选择小的。**