Nim游戏

> 你和你的朋友面前有一堆石子，你们轮流拿，一次至少拿一颗，最多拿三颗，谁拿走最后一颗石子谁获胜。
> 假设你们都很聪明，由你第一个开始拿，请你写一个算法，输入一个正整数 n，返回你是否能赢（true 或 false）。

解决这种问题的思路一般都是反着思考： 如果我能赢，那么最后轮到我取石子的时候必须要剩下 1~3 颗石子，这样我才能一把拿完。 如何营造这样的一个局面呢？显然，如果对手拿的时候只剩 4 颗石子，那么无论他怎么拿，总会剩下 1~3 颗石子，我就能赢。 如何逼迫对手面对 4 颗石子呢？要想办法，让我选择的时候还有 5~7 颗石子，这样的话我就有把握让对方不得不面对 4 颗石子。 如何营造 5~7 颗石子的局面呢？让对手面对 8 颗石子，无论他怎么拿，都会给我剩下 5~7 颗，我就能赢。 这样一直循环下去，我们发现只要踩到 4 的倍数，就落入了圈套，永远逃不出 4 的倍数，而且一定会输。所以这道题的解法非常简单：

```
bool canWeNim(int n) {
  return n % 4 != 0;
}
```

石头游戏

> 你和你的朋友面前有一排石头堆，用一个数组 piles 表示，piles[i] 表示第 i 堆石子有多少个。你们轮流拿石头，一次拿一堆，但是只能拿走最左边或者最右边的石头堆。所有石头被拿完后，谁拥有的石头多，谁获胜。

分析： 题目条件： 一是石头总共有偶数堆，石头的总数是奇数。 而作为第一个拿石头的人，你可以控制自己拿到所有偶数堆，或者所有的奇数堆。 你最开始可以选择第 1 堆或第 4 堆。如果你想要偶数堆，你就拿第 4 堆，这样留给对手的选择只有第 1、3 堆，他不管怎么拿，第 2 堆又会暴露出来，你就可以拿。同理，如果你想拿奇数堆，你就拿第 1 堆，留给对手的只有第 2、4 堆，他不管怎么拿，第 3 堆又给你暴露出来了。 也就是说，你可以在第一步就观察好，奇数堆的石头总数多，还是偶数堆的石头总数多，然后步步为营，就一切尽在掌控之中了。

```
class Solution {
public:
  bool stoneGame(vector<int>& piles) {
    return true;
  }
};
```

点灯开关问题

有 n 盏电灯，最开始时都是关着的。现在要进行 n 轮操作：
第 1 轮操作是把每一盏电灯的开关按一下（全部打开）。
第 2 轮操作是把每两盏灯的开关按一下（就是按第 2，4，6... 盏灯的开关，它们被关闭）。
第 3 轮操作是把每三盏灯的开关按一下（就是按第 3，6，9... 盏灯的开关，有的被关闭，比如 3，有的被打开，比如 6）...
如此往复，直到第 n 轮，即只按一下第 n 盏灯的开关。
现在给你输入一个正整数 n 代表电灯的个数，问你经过 n 轮操作后，这些电灯有多少盏是亮的?

分析： 首先，因为电灯一开始都是关闭的，所以某一盏灯最后如果是点亮的，必然要被按奇数次开关。 我们假设只有 6 盏灯，而且我们只看第 6 盏灯。需要进行 6 轮操作对吧，请问对于第 6 盏灯，会被按下几次开关呢？这不难得出，第 1 轮会被按，第 2 轮，第 3 轮，第 6 轮都会被按。 为什么第 1、2、3、6 轮会被按呢？因为 6=1×6=2×3。一般情况下，因子都是成对出现的，也就是说开关被按的次数一般是偶数次。但是有特殊情况，比如说总共有 16 盏灯，那么第 16 盏灯会被按几次? 16=1×16=2×8=4×4 其中因子 4 重复出现，所以第 16 盏灯会被按 5 次，奇数次。现在你应该理解这个问题为什么和平方根有关了吧？ 不过，我们不是要算最后有几盏灯亮着吗，这样直接平方根一下是啥意思呢？稍微思考一下就能理解了。 就假设现在总共有 16 盏灯，我们求 16 的平方根，等于 4，这就说明最后会有 4 盏灯亮着，它们分别是第 1×1=1 盏、第 2×2=4 盏、第 3×3=9 盏和第 4×4=16 盏。 我们不是想求有多少个可开方的数吗，4 是最大的平方根，那么小于 4 的正整数的平方都是在 1~16 内的，是会被按奇数次开关，最终亮着的灯。 就算有的 n 平方根结果是小数，强转成 int 型，也相当于一个最大整数上界，比这个上界小的所有整数，平方后的索引都是最后亮着的灯的索引。所以说我们直接把平方根转成整数，就是这个问题的答案。

```
class Solution {
public:
  int bulbSwitch(int n) {
    return (int)sqrt(n);
  }
};
```