洪水

根据时间进行模拟过程，可以同时模拟更新出洪水当前淹没的状态和果老师当前的移动情况。

用两个BFS,分别维护这两个部分即可。

别墅晚餐

有个比较容易想到的办法，预处理出二维前缀和（表示一个矩形的`x`的个数），再n^4枚举矩形的两个顶点，然后判断矩形内的`x`个数是否为0，更新最优解即可。

但是$O(n^4)$通过不了。

上述思路可以优化，我们预处理每行的前缀和$sum[i][j]$表示i行到了j列的`x`的个数，更改枚举方法，考虑枚举l,r表示矩形长的左右边界，要使周长最大，就得在此基础上最大化宽度，贪心的从第1行开始判断每行的`x`的个数，记录能连续的最大合法长度就是当前的宽，然后更新周长就好了。

时间复杂度：$O(n^3)$

写BUG

本题就是要找满足最大中心对称的正方形，为了求边长的最大值，我们二分边长的一半（注意奇偶性讨论），在`check`函数中，如果正反hash相等就返回1。

直角三角形

如果我们设符合要求的三角形直角顶点坐标为$A(x_1,y_1)$，那么剩下两点坐标一定为 $B(x_1,yi)$、$C(x_j,y_1)$。所以，如果我们确定一个点为直角顶点，那么可以与之配合成为目标三角形的只有是横、纵坐标都与之相同的点。

由此，我们可以考虑用两个数组 $sumx$、 $sumy$来记录一个横坐标或纵坐标上的点的个数，明显的，当直角顶点为 $A(x_1,y_1)$时，可以组成的目标三角形个数为$(sumx-1)*(sumy-1)$

由此，我们可以枚举每个点，取出它所在的横、纵坐标，用上面的公式计算出在这个点可以组成的目标三角形个数，累加起来，就是答案。

取数游戏

考虑破环成链，$dp[i][j]$ 表示 $i-j$这条链可以从两边开始选，先手能比后手多拿到最多的奇数个数，显然$dp[i][j]=max(dp[i][i]-dp[i+1][j],dp[j][j]-dp[i][j-1])$，即枚举取左边还是右边。

然后当 $dp[i][i]-dp[i+1][i+n-1]>1$时这个点可行，直接判断即可。

```
#include <bits/stdc++.h>
using namespace std;
int n, a[110], dp[210][210];
int main() {
    cin >> n;
```

```
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
        a[i] %= 2;
        dp[i][i] = dp[i + n][i + n] = a[i];
    }
    for (int l = 2; l <= n; l++) {
        for (int i = 1; i <= (n << 1); i++) {
            int j = i + l - 1;
            if (j > (n << 1)) {
                break;
            }
            dp[i][j] = max(dp[i][i] - dp[i + 1][j], dp[j][j] - dp[i][j - 1]);
        }
    }
    int ans = 0;
    for (int i = 1; i <= n; i++) {
        if (dp[i][i] - dp[i + 1][i + n - 1] > 0) {
            ans++;
        }
    }
    cout << ans;
}
```

数据还原

输入数据，可以使用 $getline$ 解决，一次读入一行，在字符串中处理数据。

然后暴力循环，不要修改原来没有被污损的数据。

我们就可以判断（当然可以用表达式），如果不是 ?，就按照他的，是的话, 0 到 100 循环。

传奇厨师

二分答案

对于每个需要的食材，我们先二分能做多少菜，

然后可以使用**贪心**算出它去除已有部分还需用钱买的量

只要枚举大包食材的袋数，从而计算出小包用量，选取最小值