蓝桥杯选拔赛第一场 A正解 && G题解

A 题：lrq学长的神奇数列

正解：

前置知识：快速幂、矩阵乘法

注意到上面的暴力使用的是递推的思想，但是由于项数 *N* 很大，一项一项的递推想要在 1s 的时限内把答案跑出来是不可能的

类似的问题我们在学习快速幂的时候遇到过，求 a^b % p 时，当b很大时，按照定义，把 a 乘上 b 次是很慢的

于是我们把取幂的任务按照 b 的 **二进制表示** 来分割任务。

快速幂：

```
long long qpow(long long a,long long b,int p){
    long long res = 1;
    while(b){
        //数很大 每一步都需要取模
        if(b & 1) res *= a, res %= p;
        a = a * a % p;
        b >>= 1;
    }
    return res;
}
```

在这题里面也可以用快速幂的方法，只不过数乘变成了矩阵乘

设初始状态矩阵为 *F*

$$[a_{i-3}\ a_{i-2}\ a_{i-1}]$$

以及递推至下一项的矩阵 *P*

$$[a_{i-2}\ a_{i-1}\ a_i]$$

我们要构建一个矩阵 *A* 使得 $F\times A = P$

由递推式

$$a_i = a_{i-1} + a_{i-3}$$

易得 *A*

$$\begin{bmatrix}0 & 1 & 0\\0 & 0 & 1\\1 & 0 & 1\end{bmatrix}$$

于是往后递推一位可以等价于将 *F* 乘上一次 *A* ,接下来就是矩阵乘法上的快速幂问题了

时间复杂度：O(log n) 能允许跑2^{10^7} 以内的数据以及多次询问

```cpp
#include<bits/stdc++.h>
const int mod = 1e9+7;
long long f[5], a[5][5];
long long n;
void axa(){ // 矩阵A自乘
    long long w[5][5];
    memset(w,0,sizeof w);
    for(int i = 1; i <= 3; i++)
        for(int k = 1; k <=3 ; k++)
            if(a[i][k])
                for(int j = 1; j <= 3; j++)
                    if(a[k][j])
                        //数很大 每一次操作都要取模一次
                        w[i][j] += a[i][k] * a[k][j], w[i][j] %= mod;
    memcpy(a, w, sizeof a);
}
void fxa(){ //F和当前的A相乘
    long long w[5];
    memset(w, 0, sizeof w);
    for(int i = 1; i <= 3; i++)
        for(int j = 1;j <= 3; j++)
            w[i] += f[j] * a[j][i], w[i] %= mod;
    memcpy(f,w,sizeof f);
}
void mpow(long long k){ //快速幂
    while(k){
        if(k & 1) fxa();
        axa();
        k >>= 1;
    }
}
signed main(){
    long long t; std:: cin >> t; //询问次数
    while(t--){
        //初始化矩阵
        f[1] = 1; f[2] = 1; f[3] = 1;
        a[1][1] = 0; a[1][2] = 1; a[1][3] = 0;
        a[2][1] = 0; a[2][2] = 0; a[2][3] = 1;
        a[3][1] = 1; a[3][2] = 0; a[3][3] = 1;
        std::cin >> n; //询问第n项
        if(n <= 3){
            std::cout << 1 << '\n';
            continue;
        }
        mpow(n - 2);
        std::cout << f[1] << '\n';
    }
}
```

G 题：分配

一道博弈问题

1个人时，拥有分配权的人可以获得所有分配池中的分数

2个人时，拥有分配权的人可以获得所有池中的分数，因为在第一个人投同意的情况下，第二个人的反对不影响方案通过；

3个人时，拥有分配权的人为了能让自己不被投死，必须收买1个人，根据上面的情况可以看出，只需要拿出1分收买最后一个人（如果第一个人被投死，最后一个人只能再得0分），那么自己在有两票的情况下就不会被投死。

以此类推……我们会发现，在分配池中的分数足够多的情况下，拥有分配权的人只要收买(n-1)/2个人，每个人每个只给他们1分，自己的方案就能通过。

那么，假如拥有分配权的人，分配池中的分数足够收买后面的人，且收买之后还有分数剩余，那么分配者获得最高分数；

假如拥有分配权的人，分配池中的分数刚好收买后面的人，即收买之后没有分数剩余，那么被收买者获得最高分数3分；

假如拥有分配权的人，分配池中的分数不够收买后面的人，分配者被投死，分数进入分配池，可以进行递归。

于是确立边界条件、写一个简单的递归就可以了。

```
#include<iostream>
int dfs(int i,int k){
    if(i == 1 || i == 2) return 2+k;
    if(k > (i - 1) / 2) return 2 + k - (i - 1) / 2;
    else if( k == (i - 1) / 2) return 3;
    else return dfs(i - 1,k + 2);
}
signed main(){
    int n, m;
    std::cin >> n >> m;
    int ans = dfs(n,m-2*n);
    std::cout << ans * 100 << '\n';
    return 0;
}
```