解题报告

By moonstaring

1. 下棋

原题是 洛谷 P2051 中国象棋

首先可以按行枚举，然后发现列有三种状态：放了两个棋子，放了一个棋子，没有放棋子。然后可以设出状态 f(i, j, k)表示到了第 i 行，有 j 个放了一个的，k 个放了两个的，从这往后的方案数，假设此时放了棋子的列都是已经有固定位置的。（放了两个棋子的列数量不需要记录，可以通过列数减去一个和两个的得到）

又因为一行最多放两个棋子，可以枚举是放在有 0 个的列里，还是有 1 个的列里。具体可以见图片。

```cpp
f[wi][fi][si]=(f[wi][fi][si]+finds(wi+1,fi,si))%M;
//放0个的情况
if(fi>0) f[wi][fi][si]=(f[wi][fi][si]+finds(wi+1,fi-1,si)*fi)%M;
//放在一个有1个的列里
if(si>0) f[wi][fi][si]=(f[wi][fi][si]+finds(wi+1,fi+1,si-1)*si)%M;
//放在一个有0个的列里
if(fi+si>=2){
    if(fi>1) f[wi][fi][si]=(f[wi][fi][si]+finds(wi+1,fi-2,si)*fi*(fi-1)/2)%M;
    //放两个，都在有1个的列里
    if(fi>0&&si>0) f[wi][fi][si]=(f[wi][fi][si]+finds(wi+1,fi,si-1)*fi*si)%M;
    //放两个，一个在有1个的列里，一个在有0的列里
    if(si>1) f[wi][fi][si]=(f[wi][fi][si]+finds(wi+1,fi+2,si-2)*si*(si-1)/2)%M;
    //放两个，都在有0个的列里
}
```

2.抓老鼠

原题 CF148D Bag of mice

其实就是硬模拟……然后加个记忆化。

k(i,j)表示此时有 i 只白老鼠，j 只黑老鼠，然后就是按题意模拟

```
double find(int a,int b)
{
    if(a==0) return 0;
    if(b==0) return 1;
    if(k[a][b]!=0) return k[a][b];
    k[a][b]=a/q(a+b);
    //小A第一次就抓到白老鼠
    if(b>1){
        k[a][b]=k[a][b]+b/q(a+b)*(b-1)/q(a+b-1)*a/q(a+b-2)*find(a-1,b-2);
        //小A第一次抓到黑老鼠，小B第一次抓到黑老鼠，跑出来白老鼠
        if(b>2)k[a][b]=k[a][b]+b/q(a+b)*(b-1)/q(a+b-1)*(b-2)/q(a+b-2)*find(a,b-3)
        //小A第一次抓到黑老鼠，小B第一次抓到黑老鼠，跑出来黑老鼠
    }
    return k[a][b];
}
```

3.观光电梯

原题 CF983C

可以简单地设出状态 dp(i,a0,a1,a2,a3)表示还没进来的第一个人是 i,电梯里人的目的地是 a0,a1,a2,a3(没人就是 0),可以简单地通过搜索转移。可以通过30%的数据。

但是，在100%的数据下，空间会爆炸。可以考虑将a3改为一个值表示i-1是否还在电梯里，效果是等价的，开的数组是4e7级别的int，在限制为512M时是可以接受的。

有个小优化就是由于电梯里的人是无序的，可以提前给每个人都目的地排序，减少搜索次数

```
int dfs(int nt,int wi,int a[3],int wo){
    int b[3];
    sort(a,a+3);
    for(int i=0;i<3;++i) b[i]=a[i];
    if(nt==n+1&&wo==0&&a[2]==0){
        return 0;
    }
    if(dp[nt][wi][a[0]][a[1]][a[2]][wo]>0) return dp[nt][wi][a[0]][a[1]][a[2]][wo];
    int ans=1e8;
    //前面是一些初始化信息
    if((a[0]==0||wo==0)&&nt!=n+1){//去接下一个人,要保证有空位并且下一位有人
        if(wo==0) {
            ans=min(ans,dfs(nt+1,que[nt].ai,a,1)+abs(que[nt].ai-wi)+1);
            for(int i=0;i<3;++i) a[i]=b[i];//由于函数里传的是引用，必须重新赋给原来的值
        }
        else {
            a[0]=que[nt-1].bi;//把本来存在wo里的信息转移
            ans=min(ans,dfs(nt+1,que[nt].ai,a,1)+abs(que[nt].ai-wi)+1);
            for(int i=0;i<3;++i) a[i]=b[i];
        }
    }
```

```
ss.cpp
    for(int i=0;i<3;++i){//去放下电梯里的人
        if(a[i]==0) continue;//这里得有人
        a[i]=0;
        ans=min(ans,dfs(nt,b[i],a,wo)+abs(b[i]-wi)+1);
        for(int i=0;i<3;++i) a[i]=b[i];
    }
    if(wo){//如果第i-1个人在，去放下他
        ans=min(ans,dfs(nt,que[nt-1].bi,a,0)+abs(que[nt-1].bi-wi)+1);
        for(int i=0;i<3;++i) a[i]=b[i];
    }
    dp[nt][wi][a[0]][a[1]][a[2]][wo]=ans;
    return ans;
}
```

4.收集
原题 P3303 淘金

乍一看数据范围很没有头绪，再仔细想想最后有值的一定是 2,3,5,7 的倍数，这肯定不多，可以枚举（实际上粗估一下 40*30*20*20 完全可以接受，肯定比这个小）。然后就是一个数位 dp。

设 dp(wi,0/1,a0,a1,a2,a3),wi 表示当前正在枚举第 wi 位，如果是 1 的话就是上一位贴着上界，0 的话就是可以随便取，a0,a1,a2,a3 分别表示接下去的位数了需要给 2,3,5,7 分配多少个数（其实就是记录接下去位数的各位之积），然后枚举这一位选什么转移。

总体的思路就是说枚举每一个最后有值的数，然后对这个数进行 dp，得到这一位的金子数。最后进行一次简单的贪心，这里就不赘述了。

实现细节详见代码和注释

```
ll dfs(int wi,bool c,int a[4]){
    //边界条件和预处理
    if(wi==0){
        for(int i=0;i<4;++i){
            if(a[i]!=0) return 0;//边界条件，要分配的值都是0时return 1
        }
        return 1;
    }
    int b[4];
    for(int i=0;i<4;++i) {
        if(a[i]<0) return 0;
        b[i]=a[i];
    }
    //转移部分
    ll ans=0;
    if(c){
        if(s[wi]>0){//特判！如果贴位并且为0就不能直接搜该位的值
            ch(s[wi],a);
            ans+=dfs(wi-1,1,a);
        }
        for(int i=0;i<4;++i) a[i]=b[i];
        for(int i=s[wi]-1;i>=1;--i){//位数上不能有0
            ch(i,a);
            ans+=dfs(wi-1,0,a);
            for(int i=0;i<4;++i) a[i]=b[i];
        }
        return ans%M;
    }
    //转移基本同上，只不过不限制最高位
    if(yf[wi][a[0]][a[1]][a[2]][a[3]]>0) return dp[wi][a[0]][a[1]][a[2]][a[3]];
    for(int i=9;i>=1;--i){
        ch(i,a);
        ans+=dfs(wi-1,0,a);
        for(int i=0;i<4;++i) a[i]=b[i];
    }
    ans%=M;
    yf[wi][a[0]][a[1]][a[2]][a[3]]=1;
    dp[wi][a[0]][a[1]][a[2]][a[3]]=ans;
    return ans;
}
```

以及前面的枚举

```cpp
for(int i=0;i<=40;++i,aa*=2){
    if(aa>n) break;
    for(int j=0;j<=30;++j,bb*=3){
        if(aa*bb>n) break;
        for(int w=0;w<=20;++w,cc*=5){
            if(aa*bb*cc>n) break;
            for(int q=0;q<=20;++q,dd*=7){//枚举2,3,5,7的次方数
                if(aa*bb*cc*dd>n) break;
                ans=0;
                df[0]=i,df[1]=j,df[2]=w,df[3]=q;
                for(int g=yo;g>=1;--g){//枚举有多少位
                    if(g==yo)ans+=dfs(g,1,df);//在最高位就算贴位
                    else ans+=dfs(g,0,df);
                }
                ccd[++ct]=ans%M;
            }
            dd=1;
        }
        cc=1;
    }
    bb=1;
}
```