UCF Local Contest 2015
解题报告

东华大学 ACM 集训队

2020 年 2 月

A. Find the Twins

题目大意

给定长度为 10 的数列，若数列中只含有 18，输出 Mack。若只含有 17，输出 Zack。若同时含有 18 和 17，输出 both。若不含 18 和 17，输出 none。

题解

签到题，略。

Code

```
#include <iostream>
#include <algorithm>
#include <cstring>
#include <cstdio>
using namespace std;

int main(){
    int N;
    scanf("%d",&N);
    while(N--){
        bool Mack=false,Zack=false;
        for(register int i=1;i<=10;++i){
            int x;scanf("%d",&x);
            printf("%d",x);
            if(i<10) printf(" ");
            if(x==18) Mack=true;
            else if(x==17) Zack=true;
        }
        printf("\n");
        if(Mack && Zack) printf("both\n");
        else if(!Mack && !Zack) printf("none\n");
```

```
        else if(Mack) printf("mack\n");
        else printf("zack\n");
        printf("\n");
    }

    return 0;
}
```

B. Medal Ranking

题目大意

在奥运会上，有金银铜三种奖牌。比较两个国家的名次时有两种规则。
第一种(count)：所有奖牌的总数多者胜。
第二种(color)：金牌多者胜。若金牌数目相同，则银牌多者胜。若银牌数目仍相同，则铜牌多者胜。
现在分别给出美国和俄罗斯金银铜牌的数量，若按两种规则都是美国胜，输出 both。若只有按第一种规则美国胜，输出 count。若只有按第二种规则美国胜，输出 color。若按两种规则美国都败，输出 none。

题解

直接模拟即可。

Code

```
#include <iostream>
#include <algorithm>
#include <cstring>
#include <cstdio>
using namespace std;

int USA[3],Russia[3];
int T;

int main(){
    scanf("%d",&T);
    while(T--){
        scanf("%d%d%d",&USA[0],&USA[1],&USA[2]);
        scanf("%d%d%d",&Russia[0],&Russia[1],&Russia[2]);
```

```
        printf("%d %d %d %d %d %d\n",USA[0],USA[1],USA[2],Russia[0],Russia[
1],Russia[2]);
        bool Count=false,Color=true;
        if(USA[0]+USA[1]+USA[2]>Russia[0]+Russia[1]+Russia[2]) Count=true;
        if(USA[0]<Russia[0]) Color=false;
        else if(USA[0]==Russia[0] && USA[1]<Russia[1]) Color=false;
        else if(USA[0]==Russia[0] && USA[1]==Russia[1] && USA[2]<=Russia[2]
) Color=false;
        if(Count && Color) printf("both\n");
        else if(Color) printf("color\n");
        else if(Count) printf("count\n");
        else printf("none\n");
        printf("\n");
    }
    return 0;
}
```

C. Brownies vs. Candies vs. Cookies

题目大意

有 N 块蛋糕要依次分给 M 组学生，若当前组学生的人数小于等于蛋糕的数量，就要把每块蛋糕切成两半，若仍不够分，就再把每块蛋糕切成两半，如此重复直到蛋糕数量大于当前组的学生人数为止。分别输出分给每组学生蛋糕后，剩余的蛋糕数量。

题解

直接模拟即可。

Code

```
#include <iostream>
#include <algorithm>
#include <cstring>
#include <cstdio>
using namespace std;

int T,Case=0;

int main(){
    scanf("%d",&T);
```

```
    while(T--){
        int Student,Num,M;
        scanf("%d%d",&Student,&Num);
        printf("Practice #%d: %d %d\n",++Case,Student,Num);
        scanf("%d",&M);
        while(M--){
            int x;scanf("%d",&x);
            while(Num<=x) Num<<=1;
            Num-=x;
            printf("%d %d\n",x,Num);
        }
        printf("\n");
    }
    return 0;
}
```

D. Lemonade Stand

题目大意

你在经营一家卖柠檬水的店。制作一杯柠檬水需要消耗x个柠檬和s盎司糖。
柠檬是一个个卖的，糖是 5 磅每包卖的(1 磅 = 16 盎司)。
给出接下来的$d(d \le 1000)$天里，每天将要卖出的柠檬水的数量，以及这一天的每个柠檬和每包糖的价格，你可以在任意天采购任意数量的柠檬和糖，并且在一天之内是先采购柠檬和糖再售卖柠檬水的。询问满足这d天的需求的情况下，购入柠檬和糖的最少总价。

题解

贪心。若在当天原材料够了，则不再购入。否则在当天及以前该原材料价格最低的时候购入恰好满足当天需求的量。时间复杂度$O(d)$。

Code

```
#include <iostream>
#include <algorithm>
#include <cstring>
#include <cstdio>
using namespace std;

int Num[1005],A[1005],B[1005];
int T,Day,N,M,Ans;
```

```
int main(){
    scanf("%d",&T);
    while(T--){
        Ans=0;
        scanf("%d%d%d",&Day,&N,&M);
        int CountA=0,CountB=0;
        int MinA=2147483647,MinB=2147483647;
        for(register int i=1;i<=Day;++i){
            scanf("%d%d%d",&Num[i],&A[i],&B[i]);
            MinA=min(MinA,A[i]);
            MinB=min(MinB,B[i]);
            if(CountA<Num[i]*N){
                int AddA=Num[i]*N-CountA;
                CountA+=AddA;Ans+=AddA*MinA;
            }
            if(CountB<Num[i]*M){
                int AddB=(Num[i]*M-CountB-1)/80+1;
                CountB+=AddB*80;Ans+=AddB*MinB;
            }
            CountA-=Num[i]*N;
            CountB-=Num[i]*M;
        }
        printf("%d\n",Ans);
    }
    return 0;
}
```

E. Rain Gauge

题目大意

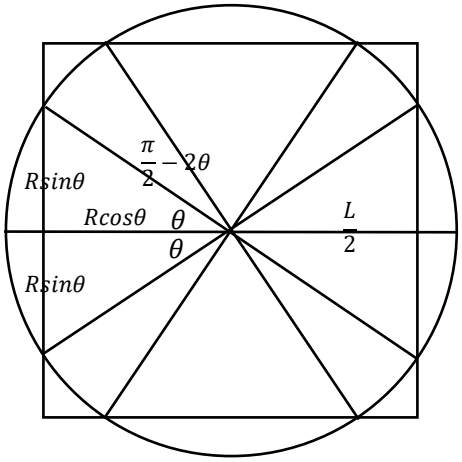
给定一个长为L的正方形和一个半径为R的圆形，它们的中心重合。求它们重合部分的面积。

题解

当$2R \geq \sqrt{2}L$时，重合部分的面积等于L^2。

当$2R \leq L$时，重合部分的面积为πR^2。

其它情况，



$$S=\pi R^2-4\left(\frac{2\theta}{2\pi}\pi R^2-\frac{1}{2}Rcos\theta\times 2Rsin\theta\right)$$

$$\theta=arccos\frac{L}{2R}$$

Code

```
#include <iostream>
#include <algorithm>
#include <cstring>
#include <cstdio>
#include <cmath>
using namespace std;

const double PI=3.14159265358979;
int T,L,R;

int main(){
    scanf("%d",&T);
    while(T--){
        scanf("%d%d",&L,&R);
        if(2*R*R>=L*L) {printf("%d.00\n",L*L);continue;}
        if(2*R<=L){printf("%.2f\n",PI*(double)(R*R));continue;}
        double theta=(double)L/(double)(2*R);
        theta=acos(theta);
        double S=PI*(double)R*R-
4.0*theta*(double)(R*R)+(double)(2*L*R)*sin(theta);
        printf("%.2f\n",S);
    }
    return 0;
}
```

F. Balanced Strings

题目大意

给定一个长为$N(N \leq 100)$的字符串，其中未知的字母用'?'表示。

现在认为 a,e,i,o,u,y 是元音字母，其它的是辅音字母。

如果一个字符串的所有偶数长度(可以为 0)的子串中元音字母与辅音字母的个数相同，则认为该字符串是平衡的。

求把'?'用单个字母替换，所能得到的平衡的字符串的方案数。

题解

一个字符串的所有偶数长度(可以为 0)的子串中元音字母与辅音字母的个数相同，则认为该字符串是平衡的。容易想到，该字符串一定是“元辅元辅元辅”这样元音和辅音交错的形式，不然必不满足要求。根据原字符串的非'?'部分，我们可以判定每个位置上是元音还是辅音，或者不符合刚刚的构造，答案为 0。若原字符串全为'?'，则答案为$6^{\lfloor \frac{N}{2} \rfloor} \times 20^{N-\lfloor \frac{N}{2} \rfloor} + 20^{\lfloor \frac{N}{2} \rfloor} \times 6^{N-\lfloor \frac{N}{2} \rfloor}$。否则设应当是元音的'?'数量为$x$，应当是辅音的'?'数量为$y$，答案为$6^x \times 20^y$。

Code

```
#include <iostream>
#include <algorithm>
#include <cstring>
#include <cstdio>
using namespace std;

#define LL long long
string S;
int T,Num=0;

LL EXP(LL b,LL n){
    if(n==0LL) return 1LL;
    LL x=1,Power=b;
    while(n){
        if(n&1) x*=Power;
        Power*=Power;
        n>>=1;
    }
    return x;
}
```

```cpp
int main(){
    ios::sync_with_stdio(false);
    cin>>T;
    while(T--){
        cin>>S;++Num;
        cout<<"String #"<<Num<<": ";
        int Case=-1,Len=S.size();;
        for(int i=0;i<Len;++i){
            if(S[i]=='?') continue;
            else if(S[i]=='a' || S[i]=='e' || S[i]=='i'
                || S[i]=='o' || S[i]=='u' || S[i]=='y'){
                Case=i%2;
                break;
            }
            else{Case=1-i%2;break;}
        }
        if(Case==-1){
            cout<<EXP(6,Len/2)*EXP(20,Len-Len/2)+
                EXP(20,Len/2)*EXP(6,Len-Len/2)<<endl<<endl;
            continue;
        }
        bool flag=false;
        int Count0=0,Count1=0;
        for(register int i=0;i<Len;++i){
            if(S[i]=='?'){
                if(i%2==0) ++Count0;
                else ++Count1;
            }
            else if(S[i]=='a' || S[i]=='e' || S[i]=='i'
                || S[i]=='o' || S[i]=='u' || S[i]=='y'){
                if(i%2!=Case) {flag=true;break;}
            }
            else if(i%2!=1-Case){flag=true;break;}
        }
        if(flag){cout<<0<<endl<<endl;continue;}
        if(Case==0) cout<<EXP(6,Count0)*EXP(20,Count1)<<endl;
        else cout<<EXP(20,Count0)*EXP(6,Count1)<<endl;
        cout<<endl;
    }
```

```
    return 0;
}
```

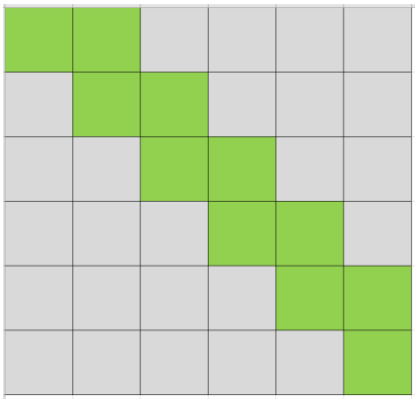
G. Towers of Hanoi Grid

题目大意

有一种 Hanoi 网格游戏，游戏规则是这样的：
柱子排列成棋盘，棋盘大小是$N \times N(2 \leq N \leq 100)$。
左上角的柱子上一开始有$d(2 \leq d \leq 100)$个圆盘，圆盘的大小从上往下依次递增。
每次只能把一个圆盘移动到该柱子左边或下边的柱子，且除了左上角的柱子和右下角的柱子，每个柱子上最多同时只能有一个圆盘。右下角的柱子，大圆盘只能放在小圆盘的下面。
求把左上角柱子上的圆盘全部移动到右下角的柱子上的最少移动步数。

题解

最终肯定是中间的柱子都放上圆盘，然后依次按圆盘从大到小的顺序移动到右下角的柱子上。又因为中间的柱子上同时只能放一个圆盘，而最大的圆盘是最后一个离开左上角的柱子，第一个移动到右下角的柱子，所以中间的柱子不可能全部都放满圆盘，一定要给最大的圆盘留下一条通往右下角的路(如下图绿色部分，这些位置不能放圆盘)。容易发现，一定存在一种移动方式，使得左上角的圆盘能够从小到大地移动到灰色区域，灰色区域的圆盘又能够从大到小地移动到右下角。而且，因为只能向左和向下移动圆盘，所以每个圆盘移动到右下角的距离一定是$2 \times (N-1)$。所以当$d \geq N \times N - 2 \times (N-1)$时，无解。否则，答案为$d \times 2 \times (N-1)$。



Code

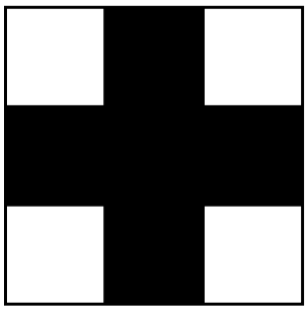
```
#include <iostream>
#include <algorithm>
#include <cstring>
#include <cstdio>
using namespace std;
```

```
int T,D,N,Case=0;

int main(){
    scanf("%d",&T);
    while(T--){
        printf("Grid #%d: ",++Case);
        scanf("%d%d",&D,&N);
        if(D>N*N-2*(N-1)){printf("impossible\n\n");continue;}
        else printf("%d\n\n",D*2*(N-1));
    }
    return 0;
}
```

H. Reach for the Stars

题目大意



有一个如左图形状的印章，黑色部分会蘸有涂料。

给你一个r行c列的图像，要求用该印章敲出这个图像，求最少的操作步数。

$$1 \le r, c \le 9$$

题解

看到数据范围，马上想到应该是搜索或者状压 dp。但是它不太“细长”，不像状压 dp，所以应该是搜索。要注意印章的中心是不能在边界上的，不然会越界。

Code

```
#include <iostream>
#include <algorithm>
#include <cstring>
#include <cstdio>
using namespace std;

char Map[10][10];
int T,N,M,Ans,Case=0;
```

```
void DFS(int i,int j,int Step,int Count){
                                        //考虑以第 i 行 j 列为中心是否要印章
                                        //且操作了 Step 步，还剩 Count 个格子没涂色
    if(Step>=Ans) return;//当前步数大于等于已有的答案，没有意义
    if(Count==0){Ans=min(Ans,Step);return;}//所有黑色部分都被涂上，更新答案
    if(i==1 || i>=N || j==1 || j>=M) return;//印章的中心点在边界上，颜色会越界
    if((Count-1)/5+1+Step>=Ans) return;//可行性剪枝
    char temp[5];
    int Cnt=0;bool flag=false;
    temp[0]=Map[i][j];
    temp[1]=Map[i-1][j];temp[2]=Map[i+1][j];
    temp[3]=Map[i][j-1];temp[4]=Map[i][j+1];
    for(register int k=0;k<5;++k){
        if(temp[k]=='#') ++Cnt;//应该被涂色而未被涂色的格子数量
        if(temp[k]=='.'){flag=true;break;}//存在白色的格子，不能涂色
    }
    if(!flag && Cnt){//不存在白色的个子，且有格子没涂
        Map[i][j]=Map[i-1][j]=Map[i+1][j]=Map[i][j-1]=Map[i][j+1]='@';
        if(j+1<M) DFS(i,j+1,Step+1,Count-Cnt);//右边一格没越界
        else DFS(i+1,2,Step+1,Count-Cnt);//右边一格越界，到下一行
        Map[i][j]=temp[0];
        Map[i-1][j]=temp[1];Map[i+1][j]=temp[2];
        Map[i][j-1]=temp[3];Map[i][j+1]=temp[4];
    }
    if(i==2 && Map[1][j]=='#') return;
    if(j==2 && Map[i][1]=='#') return;
    if(i==N-1 && Map[N][j]=='#') return;
    if(j==M-1 && Map[i][M]=='#') return;//这些情况如果此时不涂，以后就涂不了了
    if(j+1<M) DFS(i,j+1,Step,Count);
    else DFS(i+1,2,Step,Count);
    return;
}

int main(){
    ios::sync_with_stdio(false);
    cin>>T;
    while(T--){
        ++Case;
```

```
        cout<<"Image #"<<Case<<": ";
        cin>>N>>M;
        Ans=2147483647;
        int Count=0;
        for(register int i=1;i<=N;++i){
            for(register int j=1;j<=M;++j){
                cin>>Map[i][j];
                if(Map[i][j]=='#') ++Count;
            }
        }
        if(Map[1][1]=='#'||Map[1][M]=='#'||Map[N][1]=='#'||Map[N][M]=='#'){
            cout<<"impossible"<<endl<<endl;//四角上存在要涂色的格子，无解
            continue;
        }
        DFS(2,2,0,Count);//从 2 行 2 列开始搜
        if(Ans==2147483647) cout<<"impossible"<<endl;
        else cout<<Ans<<endl;
        cout<<endl;
    }

    return 0;
}
```

I. Longest Path

题目大意

给出一个$n(n \leq 500)$个点的有向图，任意两个点之间有且仅一条有向边。
求一条不重复经过一个点的最长的简单路径。

题解

任意两个点之间有且仅一条有向边，这是一个竞赛图。
竞赛图一定存在哈密顿路径，所以n个点的竞赛图的不重复经过一个点的最长的简单路径的长度为n。
我们可以根据如下方法构造一条哈密顿路径：
维护一个 list，整个 list 相当于当前维护的路径。
每次加入一个点u，
若u连向 list 中的所有点，则把u加入 list 的头部。
若 list 中的所有点连向u，则把u加入 list 的尾部。
否则，list 维护的路径中一定存在一条边$< v,w >$，且存在边$< v,u >,< u,w >$，于是把u插

入至v和w之间。

时间复杂度$O(N^2)$。

Code

```
#include <iostream>
#include <algorithm>
#include <cstring>
#include <cstdio>
#include <list>
using namespace std;

list<int> List;
int Matrix[505][505];
int T,N;

template<typename elemType>
inline void Read(elemType &T){
    elemType X=0,w=0; char ch=0;
    while(!isdigit(ch)) {w|=ch=='-';ch=getchar();}
    while(isdigit(ch)) X=(X<<3)+(X<<1)+(ch^48),ch=getchar();
    T=(w?-X:X);
}

int main(){
    Read(T);
    while(T--){
        Read(N);
        List.clear();
        for(register int i=1;i<=N;++i)
            for(register int j=1;j<=N;++j)
                Read(Matrix[i][j]);
        List.push_back(1);
        for(register int i=2;i<=N;++i){
            int CountIn=0,CountOut=0;
            for(auto it:List){
                if(Matrix[i][it]) ++CountOut;
                else ++CountIn;
            }
            if(CountOut==i-1) List.push_front(i);
            else if(CountIn==i-1) List.push_back(i);
            else{
                auto it=List.begin();
```

```
                for(;it!=List.end();++it){
                    auto Next=it;++Next;
                    if(Next==List.end()) break;
                    if(Matrix[*it][i] && Matrix[i][*Next]) break;
                }
                List.insert(++it,i);
            }
        }
        int Count=0;
        for(auto it:List){
            ++Count;
            printf("%d",it);
            if(Count<N) printf(" ");
        }
        printf("\n");
    }
    return 0;
}
```

J. You Shall Pass

题目大意

有$n(n \leq 50)$个学生，要分配到两个班级。给出每个学生在两个班级中能通过考试的概率。
而且若学生i和j在同一个班级中，学生i通过考试的概率将会增加a_{ij}。
给出每个a_{ij}，且保证无论怎样分配，任意一个学生通过考试的概率都在[0,1]范围内。

题解

相当于划分成两个集合，可以转化为最小割问题。
设源点为S，汇点为T。
S向每个学生连一条容量为该学生分到第一个班级中能通过考试的概率的边。
每个学生向T连一条容量为该学生分到第二个班级中能通过考试的概率的边。
对于每个a_{ij}，拆成一个入点和一个出点。
S向a_{ij}的入点连一条容量为a_{ij}的边，a_{ij}的出点向T连一条容量为a_{ij}的边。
a_{ij}的入点分别向第i个学生和第j个学生连一条容量为 INF 的边。
第i个学生和第j个学生分别向a_{ij}的出点连一条容量为 INF 的边。
记录下所有流量不为 INF 的边的容量之和，减去最小割即为答案。

Code

```
#include <iostream>
#include <algorithm>
```

```cpp
#include <cstring>
#include <cstdio>
#include <queue>
using namespace std;

const int INF=1<<29;
struct edge{int next,to,c;};
edge G[500005];
int head[6005];
int Deep[6005];
int current[6005];
int Num[6005];
int Pre[6005];
int cnt=2,N,M,S,T,NN,Test;

inline void add_edge(int u,int v,int c){
    G[cnt].c=c;
    G[cnt].to=v;
    G[cnt].next=head[u];
    head[u]=cnt++;
    return;
}

void ISAP_BFS(int t){
    queue<int> Q;
    for(register int i=1;i<=N;i++)
        Deep[i]=N;
    Deep[t]=0;Q.push(t);
    while(!Q.empty()){
        int now=Q.front();Q.pop();
        for(int i=head[now];i;i=G[i].next){
            if(G[i^1].c && Deep[G[i].to]>Deep[now]+1){
                Deep[G[i].to]=Deep[now]+1;
                Q.push(G[i].to);
            }
        }
    }
    return;
}

int Add_Flow(){
    int Res=INF,x=T;
    while(x!=S){
        x=Pre[x];
```

```
        Res=min(Res,G[current[x]].c);
    }
    x=T;
    while(x!=S){
        x=Pre[x];
        G[current[x]].c-=Res;
        G[current[x]^1].c+=Res;
    }
    return Res;
}

int ISAP(){
    fill(Num,Num+N+2,0);
    int now=S,MaxFlow=0;
    ISAP_BFS(T);
    for(register int i=1;i<=N;i++){
        ++Num[Deep[i]];
        current[i]=head[i];
    }
    while(Deep[S]<N){
        if(now==T){MaxFlow+=Add_Flow();now=S;}
        bool has_find=false;
        for(int i=current[now];i;i=G[i].next){
            if(Deep[now]!=Deep[G[i].to]+1 || G[i].c==0) continue;
            has_find=true;
            current[now]=i;
            Pre[G[i].to]=now;
            now=G[i].to;
            break;
        }
        if(!has_find){
            int minn=N-1;
            for(int i=head[now];i;i=G[i].next)
                if(G[i].c) minn=min(minn,Deep[G[i].to]);
            if((--Num[Deep[now]])==0) break;
            Num[Deep[now]=minn+1]++;
            current[now]=head[now];
            if(now!=S) now=Pre[now];
        }
    }
    return MaxFlow;
}

int main(){
```

```
    scanf("%d",&Test);
    while(Test--){
        scanf("%d",&NN);
        S=NN+2*NN*NN+1;T=S+1;N=T;int Sum=0;
        cnt=2;fill(head,head+N+2,0);
        for(register int i=1;i<=NN;++i){
            double x;scanf("%lf",&x);
            int c=round(x*100);Sum+=c;
            add_edge(S,i,c);
            add_edge(i,S,0);
        }
        for(register int i=1;i<=NN;++i){
            double x;scanf("%lf",&x);
            int c=round(x*100);Sum+=c;
            add_edge(i,T,c);
            add_edge(T,i,0);
        }
        for(register int i=1;i<=NN;++i){
            for(register int j=1;j<=NN;++j){
                double x;scanf("%lf",&x);
                int c=round(x*100);Sum+=2*c;
                add_edge(S,NN+(i-1)*NN+j,c);
                add_edge(NN+(i-1)*NN+j,S,0);
                add_edge(NN+(i-1)*NN+j,i,INF);
                add_edge(i,NN+(i-1)*NN+j,0);
                add_edge(NN+(i-1)*NN+j,j,INF);
                add_edge(j,NN+(i-1)*NN+j,0);
                add_edge(i,NN+NN*NN+(i-1)*NN+j,INF);
                add_edge(NN+NN*NN+(i-1)*NN+j,i,0);
                add_edge(j,NN+NN*NN+(i-1)*NN+j,INF);
                add_edge(NN+NN*NN+(i-1)*NN+j,j,0);
                add_edge(NN+NN*NN+(i-1)*NN+j,T,c);
                add_edge(T,NN+NN*NN+(i-1)*NN+j,0);
            }
        }
        printf("%.2f\n",(double)(Sum-ISAP())/100.0);
    }

    return 0;
}
```

K. Turing's Challenge

题目大意

给定正整数X和N，定义集合$T=\{T_i|1\le i\le N+1\},T_i=\binom{N}{i-1}X^{i-1}$。
要求选出T的一个非空子集S，使得$\prod_{i\in S}T_i\equiv 2(mod4)$，并最大化$\sum_{i\in S}i$。
输出最大的$\sum_{i\in S}i$。若不存在，输出 0。

$$1\le X,N<2^{31}$$

题解

当$X\ mod\ 4=0$时，$\forall S\subseteq T,\prod_{i\in S}T_i=\prod_{i\in S}\binom{N}{i-1}X^{i-1}\equiv 0(mod4)$，无解。

当$X\ mod\ 4=2$时，$\forall i\ge 3,X^{i-1}\equiv 0(mod4)$，故只需考虑$i=1$和$i=2$时的情况。

$T_1=\binom{N}{0}=1,T_2=\binom{N}{1}X=NX$。当$N$为偶数时，$NX\equiv 0(mod4)$，无解。当$N$为奇数时，$NX\equiv 2(mod4)$，取$S=\{T_1,T_2\}$，答案为 3。

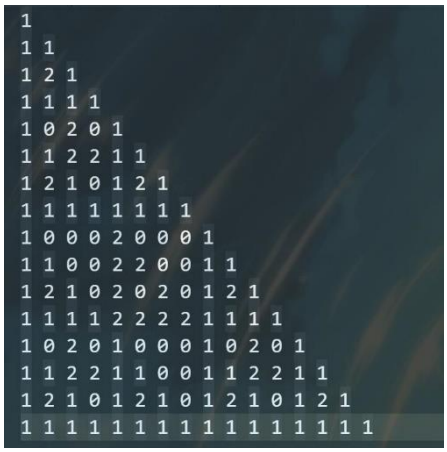
当$X\ mod\ 4=1$或$X\ mod\ 4=3$时，$\prod_{i\in S}X^{i-1}\equiv 1(mod4)$或$\prod_{i\in S}X^{i-1}\equiv 3(mod4)$。

对于$\prod_{i\in S}\binom{N}{i-1}$部分，若$\binom{N}{i-1}$是奇数，可以随便取，模数仍为1或3，因为要求$\sum_{i\in S}i$最大，所以对于奇数的$\binom{N}{i-1}$，全部都取。然后对于$\binom{N}{i-1}$是偶数的情况，只需要再取一个$\binom{N}{i-1}$，且$\binom{N}{i-1}\equiv 2(mod4)$，最终才能使得$\prod_{i\in S}\binom{N}{i-1}X^{i-1}\equiv 2(mod4)$，且只能取一个，不能不取，否则无解。

$X\ mod\ 4=0$和$X\ mod\ 4=2$时都是$O(1)$的。

接下来的问题就是对于$X\ mod\ 4=1$和$X\ mod\ 4=3$时，如何快速去计算$\sum i\left[\binom{N}{i-1}mod2=1\right]$，以及满足$\binom{N}{i-1}\equiv 2(mod4)$的最大的$i$。

对于计算$\sum i\left[\binom{N}{i-1}mod2=1\right]$，我们不妨打印出杨辉三角形模 2 的情形，发现是一个类似于谢尔宾斯基三角形的分形图。
我们针对的是其中的第N行(从 0 行开始)，可以在$O(logN)$时间内递归计算出这一行有多少个 1，然后再用$O(logN)$时间计算出所有为 1 的位置上i的和。



对于去寻找满足$\binom{N}{i-1} \equiv 2(mod4)$的最大的$i$，我们不妨打印出杨辉三角形模2等于1和模4等于2的情形，发现仍是一个分形图。我们又可以用$O(logN)$时间去递归出答案。

综上，最终此题的时间复杂度为$O(logN)$。

Code

```
#include <iostream>
#include <algorithm>
#include <cstring>
#include <cstdio>
using namespace std;

#define LL long long
LL N,X,T;

LL Count1(LL n){ //统计杨辉三角第 n 行模 2 等于 1 的位置的个数
    LL pos=1;
    while((pos<<1)<=n+1) pos<<=1;
    if(pos==n+1LL) return n+1LL;
    return (Count1(n-pos))<<1;
}

LL Calc(LL n){ //统计杨辉三角第 n 行模 2 等于 1 的位置上的 i 的和
    LL pos=1;
    while((pos<<1)<=n+1) pos<<=1;
    if(pos==n+1LL) return ((n+1LL)*(n+2LL))>>1;
    return (Calc(n-pos)<<1)+pos*Count1(n-pos);
}

LL Rightist(LL n){ //统计杨辉三角第 n 行模 4 等于 2 的最右位置
    LL pos=1;
    while((pos<<1)<=n+1) pos<<=1;
    if(pos==n+1LL) return -1;
    LL temp=Rightist(n-pos);
    if(temp==-1){
        if(n+1LL-pos<=(pos>>1)) return (pos>>1)+n+1LL-pos;
```

```
        return -1;
    }
    return temp+pos;
}

int main(){
    cin>>T;
    while(T--){
        cin>>X>>N;
        if(X%4==0) cout<<0<<endl;
        else if(X%4==2){
            if(N%2) cout<<3<<endl;
            else cout<<0<<endl;
        }
        else{
            LL Ans=Calc(N),temp=Rightist(N);
            if(temp==-1) cout<<0<<endl;
            //找不到使得 C(N,i-1) mod 4=2 的 i，无解
            else cout<<Ans+temp<<endl;
        }
    }
    return 0;
}
```