A.D.E.H 签到题

B.NEO

题意:

有一个$N*M$的矩阵，找出一个最大的子矩阵，使得这个子矩阵的每一个行列数都不为1的子子矩阵，满足左上角+右下角的值$\leq$ 左下角+右上角的值

题解:

结论：如果相邻的两个$2*2$的小矩阵是合法的，那么这两个小矩阵组成的大矩阵也是合法的

证明：对于任意一个$2*3$的矩阵

$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}$$

$$两个小矩阵是合法的，所以：$$

$$a+e \leq b+d$$

$$b+f \leq c+e$$

$$两式相加得：$$

$$a+f \leq c+d$$

所以大矩阵也是合法的。这个结论可以推广到更大的情况。

于是问题就转化成了求一个最大的子矩阵，他的每一个$2*2$的子矩阵都是合法的，那么我们用另一个$(N-1)*(M-1)$的矩阵B，其中(之前说过，[一个表达式]表示如果这个表达式为真时值是1，否则值是0)

$$B_{i,j}=[\begin{Bmatrix} A_{i,j} & A_{i,j+1} \\ A_{i+1,j} & A_{i+1,j+1} \end{Bmatrix}为合法矩阵]$$

这样，题目有转化成求最大全1子矩阵的问题,就是我们要求B的一个最大的子矩阵，这个子矩阵里全是1（也就是全是合法的$2*2$子矩阵），这是一个经典问题，网上有详细做法，我用的是悬线法+单调栈

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll; typedef double db;
typedef pair<int, int> pii; typedef pair<ll, ll> pll;
typedef pair<int, ll> pil; typedef pair<ll, int> pli;
#define Fi first
#define Se second
#define _Out(a) cerr<<#a<<" = "<<(a)<<endl
const int INF = 0x3f3f3f3f, MAXN =1e3 + 50;
const ll LINF = 0x3f3f3f3f3f3f3f3f, MOD = 1e9+7;
const db Pi = acos(-1), EPS = 1e-6;
namespace Fast_IO { //orz laofu
    const int MAXL((1 << 18) + 1); int iof, iotp;
    char ioif[MAXL], * ioiS, * ioiT, ioof[MAXL], * iooS = ioof, * iooT = ioof +
MAXL - 1, ioc, iost[55];
    char Getchar() {
        if (ioiS == ioiT) {
```

```
            ioiS = ioif; ioiT = ioiS + fread(ioif, 1, MAXL, stdin); return (ioiS
== ioiT ? EOF : *ioiS++);
        }
        else return (*ioiS++);
    }
    void Write() { fwrite(ioof, 1, iooS - ioof, stdout); iooS = ioof; }
    void Putchar(char x) { *iooS++ = x; if (iooS == iooT)Write(); }
    inline int read() {
        int x = 0; for (iof = 1, ioc = Getchar(); ioc<'0' || ioc>'9';)iof = ioc
== '-' ? -1 : 1, ioc = Getchar();
        for (x = 0; ioc <= '9' && ioc >= '0'; ioc = Getchar())x = (x << 3) + (x
<< 1) + (ioc ^ 48); return x * iof;
    }
    inline long long read_ll() {
        long long x = 0; for (iof = 1, ioc = Getchar(); ioc<'0' || ioc>'9';)iof
= ioc == '-' ? -1 : 1, ioc = Getchar();
        for (x = 0; ioc <= '9' && ioc >= '0'; ioc = Getchar())x = (x << 3) + (x
<< 1) + (ioc ^ 48); return x * iof;
    }
    template <class Int>void Print(Int x, char ch = '\0') {
        if (!x)Putchar('0'); if (x < 0)Putchar('-'), x = -x; while
(x)iost[++iotp] = x % 10 + '0', x /= 10;
        while (iotp)Putchar(iost[iotp--]); if (ch)Putchar(ch);
    }
    void Getstr(char* s, int& l) {
        for (ioc = Getchar(); ioc <'a' || ioc>'z';)ioc = Getchar();
        for (l = 0; ioc <= 'z' && ioc >= 'a'; ioc = Getchar())s[l++] = ioc; s[l]
= 0;
    }
    void Putstr(const char* s) { for (int i = 0, n = strlen(s); i < n;
++i)Putchar(s[i]); }
} // namespace Fast_IO
using namespace Fast_IO;
typedef pii TP;
struct MyQ
{
#define MAXS 1000060
    TP que[MAXS];int frt,rear;
    void init() { frt = 1; rear = 0; }
    TP front() {return frt <= rear ? que[frt] : pii(-1,-1);}
    TP back() {return frt <= rear ? que[rear] : pii(-1,-1);}
    void push_back(TP x){que[++rear] = x;}
    void pop_back(){rear--;}
    void pop_front(){frt++;}
    bool empty(){return frt > rear;}
    int size(){return rear-frt+1;}
}Q;
int a[MAXN][MAXN];
int G[MAXN][MAXN];
int len[MAXN];
void work()
{
    int n=read(),m=read();
    for(int i=1;i<=n;i++)for(int j=1;j<=m;j++)a[i][j]=read();
    for(int i=1;i<n;i++)for(int j=1;j<m;j++)
        if(a[i][j]+a[i+1][j+1]<=a[i+1][j]+a[i][j+1])
            G[i][j]=1;//就是求题解中的B[i][j]
```

```
    n--;//因为B矩阵是n-1行,m-1列的，不过这里m没有减，是一个小技巧，因为G[i][m]默认是0，所以最后就可以计算还留在单调栈里的答案
    int ans=0;
    for(int i=1;i<=n;i++)
    {
        Q.init();
        Q.push_back({-1,0});
        for(int j=1;j<=m;j++)
        {
            if(G[i][j])len[j]++;//悬线法，记到从第i行开始第j列上面还有多少个连续的1
            else len[j]=0;
            while(len[j]<=Q.back().first)
            {
                int h=Q.back().first;Q.pop_back();
                if(h==0||(j-Q.back().second==1))continue;//pop时计算答案，但如果行和列长度有一个是0就不算答案了
                ans=max(ans,(j-Q.back().second)*(h+1));
            }
            Q.push_back({len[j],j});//总之最大全1子矩阵网上的题解更详细，不会的话建议去学一下
        }
    }
    printf("%d\n",ans);
}
int main(){
    work();
}
```

--ztc

C.ZGODAN

题意:

定义一个数是合法的，当且仅当这个数中每两个连续的数位的奇偶性是不同的，给一个数，求这个数最近的合法的数

思路:

让$a[i]$表示这个数从左往右数第i位的数

先考虑比当前的数大的那个合法的数怎么求，对于第一个不合法的位置i，即$a[i]\%2 == a[i+1]\%2$，理论上只要$a[i+1]+=1$，然后后面的位01交替就行了，但是还要考虑到进位的问题，所以要分类讨论：

1.如果$a[i+1] == 9$

此时$a[i+1]$如果直接+1，那么会产生进位，于是$a[i]$也变成了偶数，那么这是$a[i+1]$就需要再+1,不过a[i]再发生进位的话是不会出现问题的（动脑筋想一想，为什么）

2.否则$a[i+1]$就直接加1

于是比当前数还要大的数就求好了，小的数也可以类似的进行讨论，可能要注意一下由于进位导致位数增加的情况

```
#pragma comment(linker, "/STACK:142400000,142400000")
#include<bits/stdc++.h>
```

```
using namespace std;
typedef long long ll; typedef double db;
typedef pair<int, int> pii; typedef pair<ll, ll> pll;
typedef pair<int, ll> pil; typedef pair<ll, int> pli;
#define Fi first
#define Se second
#define _Out(a) cerr<<#a<<" = "<<(a)<<endl
const int INF = 0x3f3f3f3f, MAXN = 5e3 + 50;
const ll LINF = 0x3f3f3f3f3f3f3f3f, MOD = 1e9+7;
const db Pi = acos(-1), EPS = 1e-6;
struct Bign
{
    short b[2005], len, f;
    void reset(){while (len > 1 && !b[len - 1])len--; if (len == 1 && b[0] ==
0)f = 1; }
    Bign() { memset(b, 0, sizeof(b)); len = 1; f = 1; }
    Bign operator =(const char* num)
    {
        memset(b, 0, sizeof(b)); int x = 0;
        if (num[0] == '-') { f = -1; num++; }
        else f = 1;
        while(num[0]=='0')num++;
        len = strlen(num);
        for (int i = 0; i < len; i++)b[i] = num[len - 1 - i] - '0';
        reset();
        return *this;
    }
    Bign operator =(int num)
    {
        char s[32]; sprintf(s, "%d", num);
        *this = s;
        return *this;
    }
    Bign operator =(ll num){
        char s[32]; sprintf(s, "%lld", num);
        *this = s;
        return *this;
    }
    Bign& operator =(const Bign & c){
        memcpy(b, c.b, sizeof(b));
        len = c.len;
        f = c.f;
        return *this;
    }
    Bign(char* num) { *this = num; }
    Bign(int num) { *this = num; }
    void print(){
        if (f == -1)printf("-");
        for (int i = len - 1; i >= 0; i--)
            printf("%d", b[i]);
    }
    bool operator <(const Bign & c)const{
        if (f > 0)
        {
            if (c.f < 0)return false;
            else
            {
                if (len != c.len)return len < c.len;
```

```
                for (int i = len - 1; i >= 0; i--)
                    if (b[i] != c.b[i])return b[i] < c.b[i];
                return false;
            }
        }else if (f < 0){
            if (c.f > 0)return true;
            else
            {
                Bign d = *this, e = c; d.f = 1; e.f = 1;
                return e < d;
            }
        }

    }
    bool operator >(const Bign & c)const{
        return c < *this;
    }
    bool operator <=(const Bign & c)const{
        return !(c < *this);
    }
    bool operator >=(const Bign & c)const{
        return !(*this < c);
    }
    bool operator !=(const Bign & c)const{
        return c<*this || c> * this;
    }
    bool operator ==(const Bign & c)const{
        return !(c != *this);
    }
    Bign operator +(const Bign & c)
    {
        Bign l = *this, r = c; int i;
        if (l.f == r.f)
        {
            for (i = 0; i < r.len; i++)
            {
                l.b[i] += r.b[i];
                if (l.b[i] > 9) { l.b[i] -= 10; l.b[i + 1]++; }
            }
            while (l.b[i] > 9)
            {
                l.b[i++] %= 10;
                l.b[i]++;
            }
            l.len = max(len, r.len);
            if (l.b[i] && l.len <= i)l.len = i + 1;
            return l;
        }
        else{
            if (l.f < 0)
            {
                l.f = 1;
                l = r - l;
                return l;
            }
            else
            {
                r.f = 1;
```

```
                l = l - r;
                return l;
            }
        }

    }
    Bign operator -(const Bign & c)
    {
        Bign l = *this, r = c; int i;
        if (l.f > 0)
        {
            if (r.f < 0)
            {
                r.f = 1;
                l = l + r;
                return l;
            }
            else
            {
                if (l >= r)
                {
                    for (i = 0; i < r.len; i++)
                    {
                        l.b[i] -= r.b[i];
                        if (l.b[i] < 0) { l.b[i] += 10; l.b[i + 1]--; }
                    }
                    while (l.b[i] < 0) {
                        l.b[i++] += 10; l.b[i]--;
                    }
                    l.reset();
                    return l;
                }
                else
                {
                    l = r - l;
                    l.f = -1;
                    return l;
                }
            }
        }
        else
        {
            if (r.f > 0)
            {
                r.f = -1;
                l = l + r;
                return l;
            }
            else
            {
                l.f = 1;
                l = r - l;
                return l;
            }
        }
    }
    Bign operator *(const Bign & c)
    {
```

```
        int i, j; Bign l = *this, r = c, ans; ans.len = l.len + r.len; ans.f =
l.f * r.f;
        for (j = 0; j < r.len; j++)for (i = 0; i < l.len; i++)ans.b[i + j] +=
l.b[i] * r.b[j];
        for (i = 0; i < ans.len; i++) { ans.b[i + 1] += ans.b[i] / 10; ans.b[i]
%= 10; }
        ans.reset();
        return ans;
    }
    Bign operator / (const Bign & c)
    {
        int i, j;
        Bign l = *this, r = c, a = 0; l.f *= r.f; r.f = 1;
        for (i = len - 1; i >= 0; i--)
        {
            a = a * 10 + b[i];
            for (j = 0; j < 10; j++)if (a < r * (j + 1))break;
            l.b[i] = j;
            a = a - r * j;
        }
        l.reset();
        return l;
    }
};//大一时搞得高精度，虽然很全但很慢
char str[MAXN],sml[MAXN],big[MAXN];
void work()
{
    scanf("%s",str+1);
    int n=strlen(str+1);
    sml[0]=big[0]=str[0]='0';
    sml[1]=big[1]=str[1];
    for(int i=2;str[i];i++)
    {
        sml[i]=big[i]=str[i];
        if((str[i]-'0')%2==(str[i-1]-'0')%2)
        {
            if(str[i]=='9'){
                if(str[i-1]=='9')
                    sml[i-2]++,sml[i-1]='0',sml[i-1]='1';
                else
                    sml[i-1]++,sml[i]='1';
            }
            else sml[i]++;
            bool nxt=((sml[i]-'0')%2==0);
            for(int j=i+1;str[j];j++)
            {
                sml[j]='0'+nxt;
                nxt^=1;
            }

            if(str[i]=='0'){
                if(str[i-1]=='0')
                    big[i-2]--,big[i-1]='9',big[i]='8';
                else big[i-1]--,big[i]='8';
            }else big[i]--;
            nxt=((big[i]-'0')%2==0);
            for(int j=i+1;str[j];j++)
            {
```

```
                big[j]='8'+nxt;
                nxt^=1;
            }
            break;
        }
    }
    Bign sm=sml,bg=big,now=str;
    if(sm-now>now-bg)
    {
        bg.print();
    }else if(sm-now<now-bg)
    {
        sm.print();
    }
    else bg.print(),printf(" "),sm.print();
}
int main(){
    work();
}
```

--ztc

F.

题意

有n个模式串,Q次操作,操作有两种,第一种为增加一个文本串,第二种为查询某个模式串在多少个文本串中出现了

思路

先按模式串建个AC自动机出来

增加文本串就直接在自动机上跑,把经过的节点叫做关键点

文本串中出现的模式串只会是关键点和关键点的$fail$树上的祖先

把这些点都更新一下的话肯定会T,因为祖先可能会很多很多

这样就只能更新关键点,然后查询的时候就查那个模式串对应节点的子树和

子树和可以用线段树求,更新关键点也是简单的单点修改

但是这样有一个问题,在加入文本串时,如果某个节点的后代中有多个关键点,那它在查询时会被重复计算

只要按dfs序更新,更新时把节点相应的lca也处理一下就好了

比如在u,v处$+1$,就在$lca(u,v)$处-1

虽然看起来也很暴力,但复杂度应该是对的,处理的节点应该不会超过关键点个数的两倍

然后线段树就一直T,改成树状数组就过了

卡常时,在更新处加了个特判,如果更新序列是u, v且u是v的祖先,就直接无视u,可以省一点时间

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef double db;
const int MOD = 998244353;
```

```
const int N = 1000005;
ll power(const ll & x,const ll & mi)
{
    ll s1=1LL,s2=x,m=mi;
    while (m)
    {
        if (m&1) s1=s1*s2%MOD;
        s2=s2*s2%MOD;
        m>>=1;
    }
    return s1;
}
inline int read()
{
    char ch=getchar();
    int x=0;
    while (ch<'0' || ch>'9') ch=getchar();
    while ('0'<=ch && ch<='9') x=(x<<3)+(x<<1)+ch-'0',ch=getchar();
    return x;
}
#define pb push_back
vector <int> adj[2000005];
queue <int> q;
struct Trie
{
    struct Node
    {
        int ch[26];
        int fail,dfn,sz;
    }node[2000005];
    int st[2000005][21],dep[2000005];
    int node_cnt,rt,ind;
    void init()
    {
        node_cnt=ind=0;
        rt=++node_cnt;
    }
    int extend(int p,int c)
    {
        if (node[p].ch[c]) return node[p].ch[c];
        ++node_cnt;
        node[p].ch[c]=node_cnt;
        return node_cnt;
    }
    int findc(int p,int c)
    {
        if (node[p].ch[c]) return node[p].ch[c];
        if (p==0) return rt;
        return node[p].ch[c]=findc(node[p].fail,c);
    }
    void get_fail()
    {
        q.push(rt);
        while (!q.empty())
        {
            int x=q.front();q.pop();
            for (int c=0;c<26;++c)
            {
```

```
                if (node[x].ch[c])
                {
                    node[node[x].ch[c]].fail=findc(node[x].fail,c);
                    q.push(node[x].ch[c]);
                }else node[x].ch[c]=findc(node[x].fail,c);
            }
        }
    }
    void dfs(int x)
    {
        node[x].dfn=++ind;
        node[x].sz=1;
        st[x][0]=node[x].fail;
        dep[x]=dep[st[x][0]]+1;
        for (auto y:adj[x])
        {
            dfs(y);
            node[x].sz+=node[y].sz;
        }
    }
    void get_dfsid()
    {
        for (int i=rt+1;i<=node_cnt;++i) adj[node[i].fail].pb(i);
        dfs(rt);
    }
    void get_st()
    {
        for (int i=1;i<21;++i) for (int j=1;j<=node_cnt;++j) st[j][i]=st[st[j]
[i-1]][i-1];
    }
    int lca(int x,int y)
    {
        if (dep[x]<dep[y]) swap(x,y);
        for (int i=20;i>=0;--i) if (dep[st[x][i]]>=dep[y]) x=st[x][i];
        if (x==y) return x;
        for (int i=20;i>=0;--i) if (st[x][i]!=st[y][i]) x=st[x][i],y=st[y][i];
        return st[x][0];
    }
}trie;
string st;
int stk[2000005],top;
int flag[2000005];
int tr[8000005];
#define lowbit(x) x&-x
void tr_add(int x,int t)
{
    for (int i=x;i<=trie.node_cnt;i+=lowbit(i)) tr[i]+=t;
}
int tr_calc(int x)
{
    int res=0;
    for (int i=x;i;i^=lowbit(i)) res+=tr[i];
    return res;
}
int tr_query(int l,int r)
{
    return tr_calc(r)-tr_calc(l-1);
}
```

```
int id[N];
bool cmp(int x,int y)
{
    return trie.node[x].dfn<trie.node[y].dfn;
}
int main()
{
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    trie.init();
    int n;
    cin>>n;
    for (int i=1;i<=n;++i)
    {
        cin>>st;
        int len = st.size(),lst=trie.rt;
        for (int j=0;j<len;++j)
        {
            lst=trie.extend(lst,st[j]-'a');
        }
        id[i]=lst;
    }
    trie.get_fail();
    trie.get_dfsid();
    trie.get_st();
    int Q,op;
    cin>>Q;
    while (Q--)
    {
        cin>>op;
        if (op==1)
        {
            cin>>st;
            int lst=trie.rt,len=st.size(),x;
            for (int i=0;i<len;++i)
            {
                lst=trie.findc(lst,st[i]-'a');
                if (!flag[lst]) stk[++top]=lst,flag[lst]=1;
            }
            sort(stk+1,stk+1+top,cmp);
            for (int i=1,lst=0,lc;i<=top;++i)
            {
                x=stk[i];
                flag[x]=0;
                if (i==top || trie.node[x].dfn+trie.node[x].sz-
1<trie.node[stk[i+1]].dfn)
                {
                    tr_add(trie.node[x].dfn,1);
                    if (lst)
                    {
                        lc=trie.lca(x,lst);
                        tr_add(trie.node[lc].dfn,-1);
                    }
                    lst=x;
                }
            }
            top=0;
        }
        if (op==2)
```

```
        {
            int x;
            cin>>x;
            x=id[x];
            cout<<tr_query(trie.node[x].dfn,trie.node[x].dfn+trie.node[x].sz-1)
<<endl;
        }
    }
    return 0;
}
```

--yzh

I.METEOR

题意:

给了个n*m的图，其中X表示陨石，#表示地面，陨石每秒垂直往下掉一格，求陨石和地面接触那一瞬间的图像

思路:

就是一个模拟题，但如果直接一秒一秒模拟肯定是会T的，所以先要知道陨石往下掉几格，然后整体往下移那么多格就可以了.

由于题目保证每一块陨石的上面不可能右地面，于是可以先求出每一列最高的地面的高度，然后对于每一块陨石，显然离地面最近的那一个就是最早就是最早碰到地面的那个，于是下落的距离也求出来了。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll; typedef double db;
typedef pair<int, int> pii; typedef pair<ll, ll> pll;
typedef pair<int, ll> pil; typedef pair<ll, int> pli;
#define Fi first
#define Se second
#define _Out(a) cerr<<#a<<" = "<<(a)<<endl
const int INF = 0x3f3f3f3f, MAXN = 3e3 + 50;
const ll LINF = 0x3f3f3f3f3f3f3f3f, MOD = 1e9 + 7;
const db Pi = acos(-1), EPS = 1e-6;
char str[MAXN][MAXN];
int h[MAXN];
void work()
{
    int n, m; scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++)
    {
        scanf("%s", str[i] + 1);
    }
    int minv = INF;
    for (int i = n; i >= 1; i--)
    {
        for (int j = 1; j <= m; j++)
        {
            if (str[i][j] == '#')h[j] = i;
            if(str[i][j]=='X')minv=min(minv,h[j]-i-1);
```

```
        }
    }
    for (int i = n; i >= 1; i--)
    {
        for (int j = 1; j <= m; j++)
        {
            if (str[i][j] == 'X')
            {
                str[i + minv][j] = 'X';
                str[i][j] = '.';
            }
        }
    }
    for (int i = 1; i <= n; i++)printf("%s\n", str[i] + 1);
}
int main(){
    work();
}
```

--ztc

K.WTF

想知道的单独问我。 --ztc

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll; typedef double db;
typedef pair<int, int> pii; typedef pair<ll, ll> pll;
typedef pair<int, ll> pil; typedef pair<ll, int> pli;
#define Fi first
#define Se second
#define _Out(a) cerr<<#a<<" = "<<(a)<<endl
const int INF = 0x3f3f3f3f, MAXN = 3e3 + 50;
const ll LINF = 0x3f3f3f3f3f3f3f3f, MOD = 1e9+7;

int a[MAXN];
int n, r;
int val(int i,int ind)
{
    return a[(ind-r*(i-1)%n+n+n-1)%n+1];
}
int dp[MAXN][MAXN],pre[MAXN][MAXN];
pii premax[MAXN][MAXN],sufmax[MAXN][MAXN];
void updmax(int i)
{
    for(int j=1;j<=n-1;j++)
    {
        int now=val(i,j)+dp[i][j];
        if(j==1||now>premax[i][j-1].first)premax[i][j]={now,j};
        else premax[i][j]=premax[i][j-1];
    }
    for(int j=n-1;j>=1;j--)
    {
        int now=-val(i,j+1)+dp[i][j];
        if(j==n-1||now>sufmax[i][j+1].first)sufmax[i][j]={now,j};
```

```
        else sufmax[i][j]=sufmax[i][j+1];
    }
}
void work()
{
    memset(dp,0xcf,sizeof dp);
    scanf("%d%d",&n,&r);
    for(int i=1;i<=n;i++)scanf("%d",a+i);
    for(int i=1;i<=n-1;i++)dp[1][i]=0;
    updmax(1);
    for(int i=2;i<=n+1;i++)
    {
        for(int j=1;j<=n-1;j++)
        {
            dp[i][j]=premax[i-1][j].first-val(i-1,j+1);
            pre[i][j]=premax[i-1][j].second;
            int now=sufmax[i-1][j].first+val(i-1,j);
            if(now>dp[i][j])dp[i][j]=now,pre[i][j]=sufmax[i-1][j].second;
        }
        updmax(i);
    }
    int now=1;
    for(int i=2;i<=n-1;i++)if(dp[n+1][i]>dp[n+1][now])now=i;
    printf("%d\n",dp[n+1][now]);
    /*stack<int>stk;
    stk.push(now);
    int i=n;
    while(i)
    {
        now=pre[i+1][now];
        stk.push(now);
        i--;
    }
    while(!stk.empty())
    {
        printf("%d ",stk.top());stk.pop();
    }*/
    //构造方案，理论上是对的
}
int main(){
    work();
}
```

L.NIKO

题意:

有m个人，给出这m个人可以当什么角色，然后有n个问题问是否存在某种方案。

题解:

可以暴力贪心，也可以dp.

其中$dp[n][O][V][N]$表示前n个人能否构造出$O-V-N$,然后类似背包的转移即可。

```
#pragma comment(linker, "/STACK:142400000,142400000")
#include<bits/stdc++.h>
```

```
using namespace std;
typedef long long ll; typedef double db;
typedef pair<int, int> pii; typedef pair<ll, ll> pll;
typedef pair<int, ll> pil; typedef pair<ll, int> pli;
#define Fi first
#define Se second
#define _Out(a) cerr<<#a<<" = "<<(a)<<endl
const int INF = 0x3f3f3f3f, MAXN =1e3 + 50;

int O[33],V[33],N[33];
bool dp[12][12][12],tmp[12][12][12];//滚动数组
char str[33];
void work()
{
    int n,m;
    scanf("%d",&n);

    for(int i=1;i<=n;i++)
    {
        scanf("%d-%d-%d",&O[i],&V[i],&N[i]);
    }
    scanf("%d",&m);
    dp[0][0][0]=1;
    for(int i=1;i<=m;i++)
    {
        scanf("%s",str);
        bool o=0,v=0,nn=0;
        for(int j=0;str[j];j++)
        {
            if(str[j]=='O')o=1;
            if(str[j]=='V')v=1;
            if(str[j]=='N')nn=1;
        }

        if(o)
        {
            for(int x=10;x>=1;x--)
            {
                for(int j=0;j<=10;j++)for(int k=0;k<=10;k++)
                {
                    tmp[x][j][k]|=dp[x-1][j][k];
                }
            }
        }
        if(v)
        {
            for (int x = 10; x >= 1; x--)
            {
                for (int j = 0; j <= 10; j++)for (int k = 0; k <= 10; k++)
                {
                    tmp[j][x][k] |= dp[j][x-1][k];
                }
            }
        }
        if(nn)
        {
            for (int x = 10; x >= 1; x--)
            {
```

```
                for (int j = 0; j <= 10; j++)for (int k = 0; k <= 10; k++)
                {
                    tmp[k][j][x] |= dp[k][j][x-1];
                }
            }
        }
        memcpy(dp,tmp,sizeof tmp);
    }
    for(int i=1;i<=n;i++)
    {
        printf("%s\n",dp[O[i]][V[i]][N[i]]?"DA":"NE");
    }
}
int main(){
    work();
}
```

--ztc