2017 年华中科技大学 ACM 程序设计竞赛
暨武汉地区高校邀请赛（决赛）

现场赛题解

A-Special Painting

SOLUTION

迭代加深搜索，枚举刻度数量，判断是否能包含所需求的长度

STDCODE

```
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <set>
#include <map>
#include <algorithm>
#include <ctime>

using namespace std;
const int maxn = 55;
#define time_ (printf("%.6f\n", double(clock())/CLOCKS_PER_SEC))
int n;
int length[2*maxn];
int scale[maxn], ptr1;
int que[maxn*maxn*maxn], ptr2;
map<int, int> M;
set<int> S;
int res;

bool dfs(int u, int d, int D) {
    if(d > D) return false;
    if(!res) return true;
    bool flag = false;
    for(int i = u; !flag && i < n-1; i++) {
        int len = length[i];
        for(int j = 0; j < ptr1; j++) {
            int len2 = abs(len - scale[j]);
            if(S.count(len2)) {
                if(!M.count(len2)){
                    M[len2] = 1;
                    res--;
                }
                else {
                    if(!M[len2]) res--;
```

```
                    M[len2]++;
                }
            }
            que[ptr2++] = len2;
        }
        scale[ptr1++] = len;
        flag = dfs(i+1, d+1, D);
        if(!flag) {
            ptr1--;
            for(int j = 0; j < ptr1; j++) {
                ptr2--;
                int len2 = que[ptr2];
                if(M.count(len2)) {
                    M[len2]--;
                    if(!M[len2]) res++;
                }
            }
        }
    }
    return flag;
}

int solve() {
    int i;
    for(i = 2; i <= 7; i++) {
        ptr1 = 2;
        ptr2 = 1;
        M.clear();
        M[que[ptr2-1]] = 1;
        res = S.size()-1;
        if(dfs(0, 2, i)) break;
    }
    return i;
}

int main() {

    while(scanf("%d", &n) == 1 && n) {
        for(int i = 0; i < n; i++) scanf("%d", &length[i]);
        sort(length, length+n);
        n = unique(length, length+n) - length;
        ptr1 = ptr2 = 0;
        S.clear();
        for(int i = 0; i < n; i++)
```

```
            S.insert(length[i]);
        scale[ptr1++] = 0;
        scale[ptr1++] = length[n-1];
        que[ptr2++] = length[n-1];
        int k = 0;
        for(int i = 0; i < n-1; i++, k++)
            length[n+k] = length[n-1] - length[i];
        sort(length, length+n+k);
        n = unique(length, length+n+k) - length;
        int ans = solve();
        printf("%d\n", ans);
    }
    //time_;
    return 0;
}
```

B-Prefix and Suffix

SOLUTION

后缀自动机：

如果一个子串出现 2 次以上，那么取最左出现的位置 p 和最右的位置 q，而且子串长度 x 在 L 到 R 之间，那么子串 s[p,q+x-1]符合条件，所以建好自动机，遍历每个节点记录 right 集合最大最小值

和节点的长度，复杂度 O(n)

后缀数组：

后缀数组思路比较麻烦，对于 lcp(p,q)=k，如果 k>=l，那么子串 s[p,q+min(k,r)+1]符合条件，而根据性质，lcp(p,q)=min(height[x],height[y])，所以转而枚举 t，令 height[t]为 height[x]~height[y]的最小值，求出 x 集合的最大

最小值和 y 集合的最大最小值，这个可以用单调栈实现，复杂度 O(n)，所以为了保持 O(n)的复杂度，后缀数组应使用 dc3 算法

STDCODE

```
#include <bits/stdc++.h>
const int MAXN = 5e6;
using namespace std;

#define ll long long
#define up(i,j,n)   for(int i=j;i<=n;i++)
#define down(i,j,n) for(int i=j;i>=n;i--)
#define cmax(a,b)   a=max(a,b)
#define cmin(a,b)   a=min(a,b)
#define clr(m) memset(m, 0, sizeof(m))

namespace SAM{
    int p,np,q,nq,now=1,cnt=1,n;
    int pre[MAXN],son[MAXN][26],step[MAXN],pos[MAXN],pos2[MAXN];
    int mark[MAXN],rnk[MAXN];
    bool isr[MAXN];
    void reset() {
        now=1;cnt=1;
        clr(son);clr(isr);
        clr(mark);
    }
    void extend(int poss,int nxt){
        p=now;np=++cnt;now=np;
        step[np]=step[p]+1;pos[np]=pos2[np]=poss;
        isr[np]=true;
        for(;!son[p][nxt]&&p;p=pre[p])son[p][nxt]=np;
        if(!p)pre[np]=1;
        else{
            q=son[p][nxt];
            if(step[q]==step[p]+1)pre[np]=q;
            else{
                nq=++cnt;
                step[nq]=step[p]+1;
                pos[nq]=pos2[nq]=pos[q];
                memcpy(son[nq],son[q],sizeof(son[q]));
                pre[nq]=pre[q];
                pre[np]=pre[q]=nq;
```

```
                for(;son[p][nxt]==q;p=pre[p])son[p][nxt]=nq;
            }
        }
    }
    void topsort(){
        up(i,1,cnt)mark[step[i]]++;
        up(i,1,n)mark[i]+=mark[i-1];
        down(i,cnt,1)rnk[mark[step[i]]--]=i;
    }
}

namespace Solution {
    using namespace SAM;
    int l,r;
    char s[MAXN];
    void init(){
        scanf("%d%d%s",&l,&r,s+1);
        //reset();
        n=strlen(s+1);
        up(i,1,n)extend(i,s[i]-'a');
        topsort();
    }
    void solve() {
        int ans=0;
        down(i,cnt,2){
            int p=rnk[i],q=pre[p];
            cmin(pos[q], pos[p]);
            cmax(pos2[q], pos2[p]);
            if (l<=step[p] && step[q]+1<=r && pos2[p]!=pos[p])
                cmax(ans,pos2[p]-pos[p]+min(r,step[p]));
        }
        printf("%d\n",ans);
    }
}

int main(){
    Solution::init();
    Solution::solve();
    return 0;
}
#include <bits/stdc++.h>
const int N = 5e6;
using namespace std;

```

```
#define ll long long
#define up(i,j,n)   for(int i=j;i<=n;i++)
#define down(i,j,n) for(int i=j;i>=n;i--)
#define cmax(a,b)   a=max(a,b)
#define cmin(a,b)   a=min(a,b)
#define clr(m) memset(m, 0, sizeof(m))

namespace SA{
    int wa[N*3],wb[N*3],wv[N*3],ws[N*3];
    int r[N],sa[N];
    char s[N];
    int n;
    #define c0(r,a,b) r[a]==r[b]&&r[a+1]==r[b+1]&&r[a+2]==r[b+2]

    int c12(int k,int *r,int a,int b)
    {
        if(k==2) return r[a]<r[b]||r[a]==r[b]&&c12(1,r,a+1,b+1);
        else return r[a]<r[b]||r[a]==r[b]&&wv[a+1]<wv[b+1];
     }

     void sort(int *r,int *a,int *b,int n,int m)
     {
         for(int i=0;i<n;i++) wv[i]=r[a[i]];
         for(int i=0;i<m;i++) ws[i]=0;
         for(int i=0;i<n;i++) ws[wv[i]]++;
         for(int i=1;i<m;i++) ws[i]+=ws[i-1];
         for(int i=n-1;i>=0;i--) b[--ws[wv[i]]]=a[i];
     }

     void dc3(int *r,int *sa,int n,int m)
     {
         #define F(x) ((x)/3+((x)%3==1?0:tb))
         #define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)

         int *rn=r+n,*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;
         r[n]=r[n+1]=0;
         for(int i=0;i<n;i++) if(i%3!=0) wa[tbc++]=i;

         sort(r+2,wa,wb,tbc,m);
         sort(r+1,wb,wa,tbc,m);
         sort(r,wa,wb,tbc,m);

         rn[F(wb[0])]=0; p=1;
         for(int i=1;i<tbc;i++) rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;
```

```
126.
127.         if(p<tbc) dc3(rn,san,tbc,p);
128.         else for(int i=0;i<tbc;i++) san[rn[i]]=i;
129.         /**以上是第一部分计算完毕*/
130.
131.         for(int i=0;i<tbc;i++) if(san[i]<tb) wb[ta++]=san[i]*3;
132.         if(n%3==1) wb[ta++]=n-1;
133.         sort(r,wb,wa,ta,m);
134.         /**以上是第二部分计算完毕*/
135.
136.         /**合并*/
137.         for(int i=0;i<tbc;i++) wv[wb[i]=G(san[i])]=i;
138.         int i=0,j=0;
139.         for(p=0;i<ta&&j<tbc;p++)
140.         {
141.             sa[p]=c12(wb[j]%3,r,wa[i],wb[j])?wa[i++]:wb[j++];
142.         }
143.         while(i<ta) sa[p++]=wa[i++];
144.         while(j<tbc) sa[p++]=wb[j++];
145.     }
146.     int ra[N],h[N];
147.     void getheight()
148.     {
149.         for(int i=0;i<n;i++) r[i]=s[i]-'a'+1;
150.         r[n+1]=0;
151.         dc3(r,sa,n+1,27);
152.         up(i,1,n)ra[sa[i]]=i;
153.         up(i,0,n-1) if (ra[i]>1) {
154.             int &p=h[ra[i]] ; p=max(0,i>0?h[ra[i-1]]-1:0) ;
155.             for (int x=sa[ra[i]-
    1];x+p<n && i+p<n && s[x+p]==s[i+p];) ++p ;
156.         }
157.         //up(i,1,n)
158.         //    printf("%d %s %d %d\n",i, s+sa[i], sa[i], h[i]);
159.     }
160. };
161. namespace Solution {
162.     using namespace SA;
163.     int l,r;
164.     int top,stk[N],mi[N],ma[N];
165.     int p[N],q[N];
166.     void init(){
167.         scanf("%d%d%s",&l,&r,s);
168.         n=strlen(s);
```

```
        getheight();
    }
    void solve() {
        int ans=0,top=-1;
        top=-1;
        up(i,2,n){
            p[i]=q[i]=sa[i-1];
            while(top>=0&&h[stk[top]]>=h[i]) {
                cmin(p[i],mi[top]);
                cmax(q[i],ma[top]);
                --top;
            }
            stk[++top]=i;
            mi[top]=p[i];
            ma[top]=q[i];
        }
        top=-1;
        int pp,qq;
        down(i,n,2){
            pp=qq=sa[i];
            while(top>=0&&h[stk[top]]>=h[i]) {
                cmin(pp,mi[top]);
                cmax(qq,ma[top]);
                --top;
            }
            stk[++top]=i;
            mi[top]=pp;
            ma[top]=qq;
            if (h[i]>=l) {
                cmax(ans,abs(pp-q[i])+min(r,h[i]));
                cmax(ans,abs(p[i]-qq)+min(r,h[i]));
            }
        }
        printf("%d\n",ans);
    }
}

int main(){
    Solution::init();
    Solution::solve();
    return 0;
}
```

C-Portal

SOLUTION

主算法是 BFS，搜索从 X 结点到 Y 结点的最短路。因为原先关闭的通道需要钥匙才能开启，注意处理不同位置的结点时各个通道的状态可能不同，因此在将某一结点加入队列的同时也要记录此刻各个通道的状态。处理方法为给每个结点设置一个十位二进制数 state，每一位表示相应通道的状态（0 表示关闭，1 表示开启），然后将 state 与结点的行号 r 和列号 c 编码，即 $v = state*n*m + r*m + c$

细节详见标程。

STDCODE

```
#include <stdio.h>
#include <string.h>
#define MAXP 15000000
#define MAXK 12
#define MAXN 33

int queue[MAXP], qs, qe;
int visit[MAXP];
int msk[MAXK];
char mat[MAXN][MAXN];
int portal[MAXK][2][2];
int n, m, k;

int go[4][2] =
{
    {1, 0},
    {0, 1},
    {-1, 0},
    {0, -1}
};

```

```
22. void Get(int v, int &r, int &c, int &st)
23. {
24.     st = v / (n * m);
25.     v %= (n * m);
26.     r = v / m;
27.     v %= m;
28.     c = v;
29. }
30.
31. int Set(int r, int c, int st)
32. {
33.     return st * n * m + r * m + c;
34. }
35.
36. bool CanGo(int r, int c)
37. {
38.     if (r < 0 || r >= n || c < 0 || c >= m)
39.         return false;
40.     return mat[r][c] != '#';
41. }
42.
43. bool PortalCanUse(int idx, int st)
44. {
45.     return (((st>>idx) & 1) == 1);
46. }
47.
48. void FindPortal(int r, int c, int idx, int &nr, int &nc)
49. {
50.     if (r == portal[idx][0][0] && c == portal[idx][0][1])
51.     {
52.         nr = portal[idx][1][0];
53.         nc = portal[idx][1][1];
54.     }
55.     else
56.     {
57.         nr = portal[idx][0][0];
58.         nc = portal[idx][0][1];
59.     }
60. }
61.
62. int bfs(int xr, int xc, int yr, int yc, int state)
63. {
64.     int v, r, c, st, nr, nc, nst, i, idx, tmp;
65.     qs = qe = 0;
```

```
66.     memset (visit, 0x0a, sizeof visit);
67.     v = Set(xr, xc, state);
68.     visit[v] = 0;
69.     queue[qe++] = v;
70.     while (qs < qe)
71.     {
72.         v = queue[qs++];
73.         Get(v, r, c, st);
74.         // printf("%d: %d %d %d, %d\n", v, r, c, st, visit[v]);
75.         if (r == yr && c == yc)
76.             return visit[v];
77.         for (i = 0; i < 4; i++)
78.         {
79.             nr = r + go[i][0];
80.             nc = c + go[i][1];
81.             nst = st;
82.             tmp = Set(nr, nc, nst);
83.             if (CanGo(nr, nc) && visit[tmp] > visit[v] + 1)
84.             {
85.                 // printf(" %d --> %d\n", v, tmp);
86.                 visit[tmp] = visit[v] + 1;
87.                 queue[qe++] = tmp;
88.             }
89.         }
90.         idx = mat[r][c] - 'A';
91.         if (idx >= 0 && idx < k && PortalCanUse(idx, st))
92.         {
93.             FindPortal(r, c, idx, nr, nc);
94.             nst = (st | msk[idx]);
95.             tmp = Set(nr, nc, nst);
96.             if (CanGo(nr, nc) && visit[tmp] > visit[v] + 1)
97.             {
98.                 // printf(" %d ---> %d\n", v, tmp);
99.                 visit[tmp] = visit[v] + 1;
100.                 queue[qe++] = tmp;
101.             }
102.         }
103.     }
104.     return -1;
105. }
106.
107. void Solve()
108. {
109.     char str[10];
```

```
110.     int i, j, t, state, idx, cnt, res;
111.     int xr, xc, yr, yc;
112.     memset (portal, -1, sizeof portal);
113.     memset (msk, 0, sizeof msk);
114.     scanf("%d%d%d", &n, &m, &k);
115.     for (i = 0; i < n; i++)
116.     {
117.         scanf("%s", mat[i]);
118.         for (j = 0; j < m; j++)
119.         {
120.             if (mat[i][j] == 'X') {
121.                 xr = i; xc = j;
122.             } else if (mat[i][j] == 'Y') {
123.                 yr = i; yc = j;
124.             } else if (mat[i][j] - 'A' >= 0 && mat[i][j] - 'A' < k) {
125.                 idx = mat[i][j] - 'A';
126.                 if (portal[idx][0][0] == -1) {
127.                     portal[idx][0][0] = i;
128.                     portal[idx][0][1] = j;
129.                 } else {
130.                     portal[idx][1][0] = i;
131.                     portal[idx][1][1] = j;
132.                 }
133.             }
134.         }
135.     }
136.     scanf("%d", &cnt);
137.     for (i = state = 0; i < cnt; i++)
138.     {
139.         scanf("%s", str);
140.         idx = str[0] - 'A';
141.         state = (state | (1<<idx));
142.     }
143.     for (i = 0; i < k; i++)
144.     {
145.         scanf("%d", &cnt);
146.         for (j = 0; j < cnt; j++)
147.         {
148.             scanf("%s", str);
149.             idx = str[0] - 'A';
150.             msk[i] |= (1<<idx);
151.         }
152.     }
153.     // printf("%d %d %d %d %d\n", xr, xc, yr, yc, state);
```

```
    // for (i = 0; i < k; i++)
        // printf("(%d, %d) -
   - (%d, %d) %d\n", portal[i][0][0], portal[i][0][1],
            // portal[i][1][0], portal[i][1][1], msk[i]);
    res = bfs(xr, xc, yr, yc, state);
    if (res >= 0)
        printf("%d\n", res);
    else
        printf("Unreachable\n");
}

int main()
{
    int cas, i;
    scanf("%d", &cas);
    for (i = 1; i <= cas; i++)
    {
        printf("Case #%d: ", i);
        Solve();
    }
    return 0;
}
```

D-Bipartite Graph

SOLUTION

签到题。二分图并查集。

STDCODE

```
#include <stdio.h>
#include <stack>
#include <list>
#include <map>
#include <set>
#include <iostream>

using namespace std;

const double eps = 1e-8;
const int MAXN = 300007;
```

```
12.
13. int u[MAXN], v[MAXN];
14. int fa[MAXN * 2];
15.
16. int my_get(int u)
17. {
18.     if (fa[u] == u)
19.         return u;
20.     return fa[u] = my_get(fa[u]);
21. }
22.
23. bool same(int u, int v)
24. {
25.     return my_get(u) == my_get(v);
26. }
27.
28. void my_uni(int u, int v)
29. {
30.     fa[my_get(u)] = my_get(v);
31. }
32.
33. char buffer[200];
34.
35. int main()
36. {
37.     int n, m;
38.     while(scanf("%d%d", &n, &m) != EOF) printf("%d\n", n + m);
39.     return 0;
40. }
```

E-Stations

SOLUTION

以车站和用户群建点，由用户群向车站连边，问题转化为最大权闭合子图。用网络流求解。

STDCODE

```
1.  #include <iostream>
2.  #include <cstdio>
3.  #include <algorithm>
4.  #include <cstring>
```

```cpp
#include <queue>
#include <cmath>
#define rep(i,l,r) for(int i=l; i<=r; i++)
#define clr(x,y) memset(x,y,sizeof(x))
#define travel(x) for(int i=last[x]; i!=-1; i=edge[i].pre)
const int INF = 0x7fffffff;
const int maxn = 55010;
using namespace std;
struct Edge{
    int pre,to,cost;
}edge[350000];
int n,m,x,y,z,s,t,now,tot=-1,ans=0,total=0,last[maxn],cur[maxn],d[maxn];
queue <int> q;
inline int read(){
    int ans = 0, f = 1;
    char c = getchar();
    while (!isdigit(c)){
        if (c == '-') f = -1;
        c = getchar();
    }
    while (isdigit(c)){
        ans = ans * 10 + c - '0';
        c = getchar();
    }
    return ans * f;
}
inline void addedge(int x,int y,int z){
    edge[++tot].pre = last[x];
    edge[tot].to = y;
    edge[tot].cost = z;
    last[x] = tot;
}
bool bfs(){
    while (!q.empty()) q.pop();
    clr(d,-1); d[s] = 0; q.push(s);
    while (!q.empty()){
        now = q.front(); q.pop();
        travel(now){
            if (d[edge[i].to] == -1 && edge[i].cost > 0){
                d[edge[i].to] = d[now] + 1;
                q.push(edge[i].to);
                if (edge[i].to == t) return 1;
            }
        }
```

```
    }
    return 0;
}
int dfs(int x,int flow){
    if (x == t || (!flow)) return flow; int w = 0;
    for (int i=cur[x]; i!=-1 && w<flow; i=edge[i].pre){
        if (d[edge[i].to] == d[x] + 1 && edge[i].cost > 0){
            int delta = dfs(edge[i].to,min(flow-w,edge[i].cost));
            edge[i].cost -= delta;
            edge[i^1].cost += delta;
            w += delta;
            if (edge[i].cost) cur[x] = i;
        }
    }
    if (w < flow) d[x] = -1;
    return w;
}
int main(){
    int T = read();
    while (T--) {
    memset(edge, 0, sizeof(edge));
    memset(last, 0, sizeof(last));
    memset(cur, 0, sizeof(cur));
    memset(d, 0, sizeof(d));
    tot=-1;ans=0;total=0;
    n = read(); m = read(); clr(last,-1);
    s = 0; t = n + m + 1;
    rep(i,1,n){
        x = read();
        addedge(s,i,x); addedge(i,s,0);
    }
    rep(i,1,m){
        x = read(); y = read(); z = read();
        total += z;
        addedge(i+n,t,z); addedge(t,i+n,0);
        addedge(x,i+n,INF); addedge(i+n,x,0);
        addedge(y,i+n,INF); addedge(i+n,y,0);
    }
    while (bfs()){
        rep(i,0,n+m+1) cur[i] = last[i];
        int tans = dfs(s,INF);
        ans += tans;
    }
    printf("%d\n",total-ans);
```

```
    }
    return 0;
}
```

F-Fighting and fighting

SOLUTION

这个题显然是要求找一天从左到右的路径，使这条路径上的最小的过道最大

最小值最大，于是可以二分。

其次我们发现，如果能找到这么一条路径，则表示可以将上下边界分成两个不同的集合，于是我们将上边界看作一个点，下边界看作一个点，每条线段看作一个点，于是得到 n+2 个点。

我们二分答案 ans，当每个点之间的距离小于等于 ans 时就合并两个集合，最后判断上边界和下边界是否在同一个集合

时间复杂度为 n*n*log(1000000000).

STDCODE

```
#include <bits/stdc++.h>
using namespace std;

const double pi=acos(-1.0);
const double eps=1e-8;
const int maxn = 5000 + 10;
const int inf = 0x3f3f3f3f;

struct Point{
    double x,y;
    Point(double x=0,double y=0):x(x),y(y){}
};
```

```
13.
14. typedef Point Vector;
15. Vector operator + (Vector A,Vector B){return Vector(A.x+B.x,A.y+B.y);}
16. Vector operator - (Vector A,Vector B){return Vector(A.x-B.x,A.y-B.y);}
17. Vector operator * (Vector A,double p){return Vector(A.x*p,A.y*p);}
18. Vector operator / (Vector A,double p){return Vector(A.x/p,A.y/p);}
19. bool operator <(const Point &a,const Point &b){return (a.x<b.x || (a.x==b.x
    && a.y<b.y));}
20.
21. int dcmp(double x){
22.     if (fabs(x)<eps) return 0;
23.     else return x<0?-1:1;
24. }
25.
26. bool operator == (const Point &a,const Point &b){return (dcmp(a.x-
    b.x)==0 && dcmp(a.y-b.y)==0);}
27. double Dot(Vector A,Vector B){return A.x*B.x+A.y*B.y;}
28. double Length(Vector A){return sqrt(Dot(A,A));}
29. double Cross(Vector A,Vector B){return A.x*B.y-A.y*B.x;}
30.
31. double DistanceToSegment(Point P,Point A,Point B)
32. {
33.     if (A==B) return Length(P-A);
34.     Vector v1=B-A, v2=P-A, v3=P-B;
35.     if (dcmp(Dot(v1,v2))<0) return Length(v2);
36.     else if (dcmp(Dot(v1,v3))>0) return Length(v3);
37.     else return fabs(Cross(v1,v2))/Length(v1);
38. }
39.
40. bool SegmentProperIntersection(Point a1,Point a2,Point b1,Point b2)
41. {
42.     double c1=Cross(a2-a1,b1-a1);
43.     double c2=Cross(a2-a1,b2-b1);
44.     double c3=Cross(b2-b1,a1-b1);
45.     double c4=Cross(b2-b1,a2-b1);
46.     return (dcmp(c1)*dcmp(c2)<0 && dcmp(c3)*dcmp(c4)<0);
47. }
48.
49. double distanceLineToLine(Point a, Point b, Point c, Point d)
50. {
51.     if (SegmentProperIntersection(a,b,c,d)) return 0;
52.     return min(min(DistanceToSegment(a,c,d), DistanceToSegment(b,c,d)),min(D
    istanceToSegment(c,a,b), DistanceToSegment(d,a,b)));
53. }
```

```
54.
55. struct Line
56. {
57.     Point a,b;
58. };
59.
60. Line line[maxn];
61. int p[maxn];
62. double dis[maxn][maxn];
63.
64. int f(int x) {
65.     return x==p[x]?x:p[x]=f(p[x]);
66. }
67.
68. void getdistance(int n, double M)
69. {
70.     dis[0][0]=dis[n+1][n+1]=0;
71.     dis[0][n+1]=inf;
72.     for (int i=1;i<=n;i++) {
73.         dis[i][i]=0;
74.         dis[0][i] = dis[i][0] = min(line[i].a.y, line[i].b.y);
75.         dis[i][n+1] = dis[n+1][i] = M - max(line[i].a.y, line[i].b.y);
76.         for (int j=i+1;j<=n;j++) {
77.             dis[i][j]=distanceLineToLine(line[i].a,line[i].b,line[j].a,line[
    j].b);
78.             dis[j][i]=dis[i][j];
79.         }
80.     }
81. }
82.
83. bool ok(double x, int n)
84. {
85.     for (int i=0;i<=n+1;i++) p[i]=i;
86.     for (int i=0;i<=n+1;i++) {
87.         for (int j=i+1;j<=n+1;j++)
88.             if (dis[i][j] + eps < x) {
89.                 int ii = f(i);
90.                 int jj = f(j);
91.                 if (ii != jj) p[ii] = jj;
92.             }
93.     }
94.     return f(0) != f(n+1);
95. }
96.
```

```
int main()
{
    //freopen("in.txt","r",stdin);
     // freopen("out.txt","w",stdout);
     int n;
     double M;
     while(scanf("%d",&n)!=EOF)
     {
         double L=0, R=0;
         scanf("%lf",&M);
         for (int i=1;i<=n;i++){
             scanf("%lf%lf%lf%lf",&line[i].a.x, &line[i].a.y, &line[i].b.x, &line[i].b.y);
             R = max(R, max(line[i].a.y, line[i].b.y));
         }
         getdistance(n, M);
         while(L+eps<R) {
             double mid = (L+R)/2.0;
             if (ok(mid, n)) L = mid;
             else R=mid;
         }
         printf("%.2f\n",L);
     }
     return 0;
}
```

G-Old Printer

SOLUTION

根据题意，我们可以得到 dp 方程

dp[i][j]=min(dp[k][j-1]+(sum[i]-sum[k])^2)

其中 0<=k<i，时间复杂度为 O(n^2*m)，空间复杂度为 O(n*m)。

由于给定的数据比较大，我们需要对 dp 进行优化。

化简 dp 表达式发现 dp[i][j]=min(dp[k][j-1]+sum[i]^2+sum[k]^2-2*sum[i]*sum[k]);那么我们只需要找出前 i 个数中 dp[k][j-1]+sum[k]^2-2*sum[k]*sum[i]中最小的一个。

设 k<ok，要 ok 比 k 优只需

dp[k][j-1]+sum[k]^2-2*sum[k]*sum[i]>dp[ok][j-1]+sum[ok]^2-2*sum[ok]*sum[i]

即

（dp[ok][j-1]+sum[ok]^2-dp[k][j-1]-sum[k]^2）/(2*sum[ok]-2*sum[k])<sum[i]

设 y[i]=dp[i][j-1]+sum[i]^2，x[i]=2*sum[i]

则上式为

（y[ok]-y[k]）/（x[ok]-x[k]）<sum[i]

根据上式我们可以采用斜率优化 dp，我们记 g[ok][k]=(y[ok]-y[k])/(x[k]-x[ok])那么 g[ok][k]<sum[i]表示对于 i 而言，用 ok 来更新比用 k 来更新更优。

对 g[k][l]>g[ok][k]

① 若 g[ok][ok]<sum[i]，则 ok 比 k 优

② 若 g[k][l]>=g[ok][k]>=sum[i],则 l 比 k 优

综上，对于 g[k][l]>g[ok][k]而言，k 总是不优的情况，所以我们可以去掉 k，那么我们相当于维护一个下凸包。因为 sum[i]

随 i 的增大而增大，所以下凸包上任意相邻两点 i，j 有若此时 j 比 i 优，则以后都是 j 比 i 优。

由于空间不足，并且 dp[i][j]只与 dp[k][j-1]有关，我们可以采用滚动数组来优化空间。

STDCODE

```
#include<cstdio>
#include<cstring>
using namespace std;
#define inf 1000000000000000000ll
#define maxn 10000
#define maxm 5000
#define LL long long
LL dp[maxn+10][2];
LL sum[maxn+10];
LL y[maxn+10],x[maxn+10];
int c[maxn+10];
int q[maxn+10];
LL Min(LL v,LL u) {
    if(v<u) return v;
    return u;
}
LL Square(LL x) {
    return x*x;
}
bool check1(int i,int l,int r) {
    if(l==r) return false;
    if(dp[q[l]][0]+Square(sum[i]-sum[q[l]]) > dp[q[l+1]][0]+Square(sum[i]-sum[q[l+1]])) return true;
    return false;
}
bool check2(int l,int r) {
    if(l+1==r) return false;
    if((y[q[r]]-y[q[r-1]])*(x[q[r-1]]-x[q[r-2]]) <= (y[q[r-1]]-y[q[r-2]])*(x[q[r]]-x[q[r-1]])) return true;
    return false;
}
void Function(int n,int m) {
    sum[0]=0;
    for(int i=1;i<=n;i++)
        scanf("%d",&c[i]),sum[i]=sum[i-1]+c[i];
```

```
    for(int i=1;i<=n;i++)
        dp[i][0]=inf;

    for(int j=1;j<=m;j++) {
        int l=1,r=1;
        q[1]=0;
        for(int i=1;i<=n;i++) {
            while(check1(i,l,r)) l++;
            dp[i][1]=dp[q[l]][0]+Square(sum[i]-sum[q[l]]);
            q[++r]=i; y[i]=dp[i][0]+Square(sum[i]); x[i]=2*sum[i];
            while(check2(l,r)) q[--r]=q[r+1];
        }
        for(int i=1;i<=n;i++)
            dp[i][0]=dp[i][1];
    }
    printf("%lld\n",dp[n][0]);
    return ;
}
int main() {
    freopen("data0.in","r",stdin);
    freopen("data0.out","w",stdout);
    memset(dp,0,sizeof(dp));
    int n,m;
    while(scanf("%d%d",&n,&m)!=EOF) Function(n,m);
    return 0;
}
```

H-Cookies

SOLUTION

组合数学+DP

STDCODE

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <vector>
#include <queue>
#include <stack>
```

```cpp
#include <set>
#include <map>
#include <algorithm>

using namespace std;
#define MOD 1000000007

long long f[255][255],fsum[255][255],dp[255555],dpsum[255555],g[255][255];
int x[1010],q[1010],qsum[1010],b[255555];

int main()
{
    //freopen("B.in","r",stdin);
    //freopen("B.out","w",stdout);
    int T;
    scanf("%d",&T);
    for (int cas=1;cas<=T;++cas)
    {
        memset(f,0,sizeof(f));
        memset(fsum,0,sizeof(fsum));
        memset(dp,0,sizeof(dp));
        memset(dpsum,0,sizeof(dpsum));
        memset(g,0,sizeof(g));
        memset(b,0,sizeof(b));
        memset(qsum,0,sizeof(qsum));
        int n,t;
        long long ans=0;
        scanf("%d%d",&n,&t);
        for (int i=1;i<=n;++i) scanf("%d%d",q+i,x+i);
        for (int i=0;i<=250;++i) f[0][i]=1,fsum[0][i]=fsum[0][i-1]+f[0][i];
        for (int i=1;i<=250;++i) f[1][i]=i,fsum[1][i]=fsum[1][i-1]+f[1][i];
        for (int i=2;i<=250;++i)
            for (int j=i;j<=250;++j)
                if (j>=t+1)
                {
                    f[i][j]=fsum[i-1][j-t-1];
                    fsum[i][j]=(fsum[i][j-1]+f[i][j])%MOD;
                }
        for (int i=1;i<=250;++i)
            for (int j=0;j<=250;++j)
            {
                if (j==0) g[j][i]=f[j][i];
```

```
50.                 else g[j][i]=(g[j-1][i]+f[j][i])%MOD;
51.             }
52.         for (int i=1;i<=n;++i)
53.         {
54.             qsum[i]=qsum[i-1]+q[i];
55.             for (int j=qsum[i-1]+1;j<=qsum[i];++j)
56.                 b[j]=i;
57.         }
58.         for (int i=1;i<=qsum[1];++i)
59.         {
60.             if (q[1]==x[1]) dp[i]=0;
61.             else
62.             {
63.                 if (q[1]==x[1]+1) dp[i]=1;
64.                 else
65.                 {
66.                     if (i-1-t<=0) dp[i]=1;
67.                     else dp[i]=g[q[1]-x[1]-1][i-1-t];
68.                 }
69.             }
70.             dpsum[i]=(dpsum[i-1]+dp[i])%MOD;
71.         }
72.         for (int i=qsum[1]+1;i<=qsum[n];++i)
73.         {
74.             if (q[b[i]]==x[b[i]])
75.                 dp[i]=0;
76.             else
77.             {
78.                 for (int j=qsum[b[i]-1];j>=0&&j>qsum[b[i]-1]-t;--j)
79.                 {
80.                     if (i-j-1<t) continue;
81.                     if (i-qsum[b[i]-1]-1-t-(j+t-qsum[b[i]-1])>0)
82.                         dp[i]=(dp[i]+dp[j]*g[q[b[i]]-x[b[i]]-1][i-qsum[b[i]-
    1]-1-t-(j+t-qsum[b[i]-1])])%MOD;
83.                     else dp[i]=(dp[i]+dp[j])%MOD;
84.                 }
85.                 if (qsum[b[i]-1]-t>0)
86.                 {
87.                     if (q[b[i]]-x[b[i]]-1==0) dp[i]=(dp[i]+dpsum[qsum[b[i]-
    1]-t])%MOD;
88.                     else if (i-qsum[b[i]-1]-1-
    t>0) dp[i]=(dp[i]+dpsum[qsum[b[i]-1]-t]*g[q[b[i]]-x[b[i]]-1][i-qsum[b[i]-1]-
    1-t])%MOD;
89.                     else dp[i]=(dp[i]+dpsum[qsum[b[i]-1]-t])%MOD;
```

```
                }
                if (i-qsum[b[i]-1]-t-1<=0) dp[i]=(dp[i]+1)%MOD;
                else dp[i]=(dp[i]+g[q[b[i]]-x[b[i]]-1][i-qsum[b[i]-1]-t-
1])%MOD;
            }
            dpsum[i]=(dpsum[i-1]+dp[i])%MOD;
        }
        for (int i=1;i<=qsum[n];++i) ans=(ans+dp[i])%MOD;
        ans=(ans+1)%MOD;
        cout<<ans<<endl;
    }
     return 0;
}
```

I-RAID

SOLUTION

首先是位运算，奇校验所有数字加起来为奇数，偶校验所有数字加起来为偶数，偶校验中当实际数据中“1”的个数为偶数的时候，这个校验位就是“0”，否则这个校验位就是“1”，奇校验正好相反。4 个 bit 转化为一个十六进制。可以每一个十六进制位输出一次便可。

STDCODE

```
#include <bits/stdc++.h>
using namespace std;

int d, s, b, kase = 0;
char type;
char tab[8][8000];

bool Fix()
{
    for(int i = 0; i < s * b; ++i){
        int sum = 0, cnt = 0, x_no;
        for(int j = 0; j < d; ++j){
            if(tab[j][i]=='1') ++sum;
            if(tab[j][i]=='x') ++cnt, x_no = j;
```

```
        }
        sum %= 2;
        if(cnt >= 2) return false;
        else if(cnt == 1){
            if(sum)
                if(type == 'E') tab[x_no][i] = '1';
                else tab[x_no][i] = '0';
            else
                if(type == 'E') tab[x_no][i] = '0';
                else tab[x_no][i] = '1';
        }
        else if(cnt == 0){
            if(type == 'E' && sum == 1) return false;
            if(type == 'O' && sum == 0) return false;
        }
    }
    return true;
}

void out_data()
{
    int sum = 0, bit_cnt = 0;
    for(int i = 0; i < b; ++i){
        int except = i % d;
        for(int j = 0; j < d; ++j){
            if(j == except) continue;
            for(int k = i * s; k < i * s + s; ++k){
                bit_cnt = (bit_cnt + 1) % 4;
                if(tab[j][k] == '0') sum *= 2;
                else sum = sum * 2 + 1;
                if(!bit_cnt){
                    printf("%X", sum);
                    sum = 0;
                }
            }
        }
    }
    if(bit_cnt){
        int over = 4 - bit_cnt;
        sum = sum * (1<<over);
        printf("%X", sum);
    }
    printf("\n");
}
```

```
59.
60. int main()
61. {
62.     ios::sync_with_stdio(false);
63.     while(memset(tab, 0, sizeof(tab)), cin >> d && d){
64.         cin >> s >> b >> type;
65.         for(int i = 0; i < d; ++i)
66.             cin >> tab[i];
67.         if(!Fix())
68.             printf("Disk set %d is invalid.\n", ++kase);
69.         else{
70.             printf("Disk set %d is valid, contents are: ", ++kase);
71.             out_data();
72.         }
73.     }
74.     return 0;
75. }
```

J-Rent House

SOLUTION

这道题既可以用容斥原理做，也是裸的莫比乌斯反演。

令 f(x)为 GCD(a, b, c, d) = x 的四元组个数，F(x) 为 GCD(a, b, c, d) = n*x(x 的倍数)的四元组个数，首先预处理对每个数分解质因子，对于每个 x，设有 k 个数是 x 的倍数，则 F(x) = C（k, 4）

所以 ans = f(1) = mob[1] * F[1] + .. +mob[m] * F[m], m = max(a1, a2, ..., an)

STDCODE

```
1. #include <cstdio>
2. #include <cmath>
3. #include <cstring>
4. #include <algorithm>
```

```
5.
6.
7.  #define LL long long
8.  #define MAXN 10005
9.
10. using namespace std;
11. int M;
12. bool prime[MAXN];
13. int mobi[MAXN], p[MAXN], cnt[MAXN], num[MAXN];
14.
15. int Mobius(){
16.     memset(prime, 1, sizeof(prime));
17.     int num = 0;
18.     mobi[1] = 1;
19.     for (int i = 2; i < MAXN; ++i){
20.         if (prime[i]){
21.             p[num++] = i;
22.             mobi[i] = -1;
23.         }
24.         for (int j = 0; j < num && i * p[j] < MAXN; ++j){
25.             prime[i * p[j]] = false;
26.             if (i % p[j] == 0){
27.                 mobi[i * p[j]] = 0;
28.                 break;
29.             }
30.             mobi[i * p[j]] = -mobi[i];
31.         }
32.     }
33.     return 0;
34. }
35.
36. LL Work(){
37.     memset(num, 0, sizeof(num));
38.     for (int i = 1; i <= M; ++i){
39.         for (int j = i; j <= M; j += i){
40.             num[i] += cnt[j];
41.         }
42.     }
43.     LL ans = 0;
44.     for (int i = 1; i <= M; ++i){
45.         int x = num[i];
46.         if (x >= 4){
47.             ans += (LL)mobi[i] * x * (x - 1) * (x - 2) * (x - 3) / 24;
48.         }
```

```
    }
    return ans;
}

int main(){
    freopen("data.in", "r", stdin);
    freopen("data.out","w",stdout);
    int n, m;
    Mobius();
    while (~scanf("%d%d", &n, &m)){
        M = 0;
        memset(cnt, 0, sizeof(cnt));

        for (int i = 0; i < n * m; ++i){
            int x;
            scanf("%d", &x);
            ++cnt[x];
            M = max(M, x);
        }
        if (n * m < 4){
            puts("0");
            continue;
        }
        printf("%lld\n", Work());
    }
}
```

K-Multisets

SOLUTION

使用并查集维护每个位置的数属于哪一个 Multiset。用一个 Splay 来维护一个 Multiset。Multiset 不一定要 splay，平衡树也可以，只要能实现合并操作即可。

STDCODE

```
#include <algorithm>
#include <cstdio>
#include <utility>
```

```
4.
5.  using namespace std;
6.
7.  struct TNode {
8.      bool nul;
9.      int val;
10.     unsigned vsiz, vcnt, nsiz;
11.     TNode *child[2];
12.     void Update() {
13.         nsiz = child[0]->nsiz + 1 + child[1]->nsiz;
14.         vsiz = child[0]->vsiz + vcnt + child[1]->vsiz;
15.     }
16. } Buf[100000], *Stk[100000], *Root[100001], nil;
17.
18. unsigned N, M;
19. unsigned Count, StkTop;
20. unsigned Ans;
21. unsigned Ufs[100001];
22.
23. void SplGetKth(TNode *p, unsigned k) {
24.     StkTop = 0;
25.     for (;;) {
26.         Stk[StkTop++] = p;
27.         unsigned nsiz = p->child[0]->nsiz + 1;
28.         if (k < nsiz)
29.             p = p->child[0];
30.         else if (k > nsiz) {
31.             p = p->child[1];
32.             k -= nsiz;
33.         }
34.         else
35.             break;
36.     }
37. }
38.
39. void SplGetVal(TNode *p, int val) {
40.     StkTop = 0;
41.     while (!p->nul) {
42.         Stk[StkTop++] = p;
43.         if (val < p->val)
44.             p = p->child[0];
45.         else if (val > p->val)
46.             p = p->child[1];
47.         else
```

```
48.             break;
49.     }
50. }
51.
52. TNode *SplSplay() {
53.     TNode *p = Stk[--StkTop];
54.     while (StkTop) {
55.         if (StkTop == 1) {
56.             TNode *o = Stk[--StkTop];
57.             int d = o->child[1] == p;
58.             o->child[d] = p->child[d ^ 1];
59.             p->child[d ^ 1] = o;
60.             o->Update();
61.             p->Update();
62.         }
63.         else {
64.             TNode *o = Stk[--StkTop];
65.             TNode *oo = Stk[--StkTop];
66.             int d = o->child[1] == p;
67.             int dd = oo->child[1] == o;
68.             if (d == dd) {
69.                 oo->child[d] = o->child[d ^ 1];
70.                 o->child[d ^ 1] = oo;
71.                 o->child[d] = p->child[d ^ 1];
72.                 p->child[d ^ 1] = o;
73.             }
74.             else {
75.                 oo->child[dd] = p->child[d];
76.                 p->child[d] = oo;
77.                 o->child[d] = p->child[dd];
78.                 p->child[dd] = o;
79.             }
80.             oo->Update();
81.             o->Update();
82.             p->Update();
83.             if (StkTop) {
84.                 TNode *ooo = Stk[StkTop - 1];
85.                 ooo->child[ooo->child[1] == oo] = p;
86.             }
87.         }
88.     }
89.     return p;
90. }
91.
```

```
92. TNode *SplMerge(TNode *p1, TNode *p2) {
93.     if (p1->nul)
94.         return p2;
95.     if (p2->nul)
96.         return p1;
97.     SplGetKth(p1, (p1->nsiz + 1) >> 1);
98.     p1 = SplSplay();
99.     SplGetVal(p2, p1->val);
100.     p2 = SplSplay();
101.     unsigned vsiz = p1->child[1]->vsiz + p1->vcnt;
102.     if (p2->val > p1->val) {
103.         TNode *p3 = p2->child[0];
104.         p2->child[0] = &nil;
105.         p2->Update();
106.         Ans += vsiz * p3->vsiz;
107.         p1->child[0] = SplMerge(p1->child[0], p3);
108.         p1->child[1] = SplMerge(p1->child[1], p2);
109.     }
110.     else if (p1->val > p2->val) {
111.         TNode *p3 = p2->child[1];
112.         p2->child[1] = &nil;
113.         p2->Update();
114.         Ans += vsiz * p2->vsiz;
115.         p1->child[0] = SplMerge(p1->child[0], p2);
116.         p1->child[1] = SplMerge(p1->child[1], p3);
117.     }
118.     else {
119.         Ans += vsiz * p2->child[0]->vsiz;
120.         Ans += p1->child[1]->vsiz * p2->vcnt;
121.         p1->vcnt += p2->vcnt;
122.         p1->child[0] = SplMerge(p1->child[0], p2->child[0]);
123.         p2->child[1] = SplMerge(p1->child[1], p2->child[1]);
124.     }
125.     p1->Update();
126.     return p1;
127. }
128.
129. unsigned ufs(unsigned x) {
130.     static unsigned u[100000];
131.     int child = 0;
132.     while (Ufs[x] != x) {
133.         u[child++] = x;
134.         x = Ufs[x];
135.     }
```

```
    while (child)
        Ufs[u[--child]] = x;
    return x;
}

int main() {
    nil.nul = true;
    nil.child[0] = nil.child[1] = &nil;
    int nCases;
    scanf("%d", &nCases);
    for (int iCase = 1; iCase <= nCases; ++iCase) {
        scanf("%u%u", &N, &M);
        Count = 0;
        for (unsigned i = 1; i <= N; ++i) {
            Ufs[i] = i;
            Root[i] = &Buf[Count++];
            scanf("%d", &(Root[i]->val));
            Root[i]->child[0] = Root[i]->child[1] = &nil;
            Root[i]->vcnt = 1;
            Root[i]->Update();
        }
        printf("Case #%d:\n", iCase);
        while (M--) {
            unsigned a, b;
            scanf("%u%u", &a, &b);
            if (ufs(a) == ufs(b)) {
                puts("-1");
                continue;
            }
            Ans = 0;
            Root[Ufs[a]] = SplMerge(Root[Ufs[a]], Root[Ufs[b]]);
            Ufs[Ufs[b]] = Ufs[a];
            printf("%u\n", Ans);
        }
    }
    return 0;
}
```

L-Morse Code Breaking

SOLUTION

dp 题，设 dp[i]为前 i 位的不同译法，如果第 i - j 位到第 i 位可以有效翻译，则 dp[i] += dp[i-j]。

递推写法为：

```
dp[0] = 1;
for (i = 1; i <= n; i++)
    for (j = 1; j <= 4 && j <= i; j++)
    {
        if (IsValid(i, j))
        {
            dp[i] += dp[i - j];
            dp[i] %= MOD;
        }
    }STDCODE
```

```
#include <stdio.h>
#include <string.h>
#define MAXN 100020
#define MOD 1000000007

char str[MAXN];
int dp[MAXN];
int msk[1000] = {0};

int ttt(int x)
{
    // printf("%d ---> ", x);
    int res = 0, cube = 1, t;
    while (x)
    {
        t = x % 10;
        res += t * cube;
        cube *= 3;
```

```
        x /= 10;
    }
    // printf("%d\n", res);
    return res;
}

void PreProcess()
{
    msk[ttt(12)] = 1;
    msk[ttt(2111)] = 1;
    msk[ttt(2121)] = 1;
    msk[ttt(211)] = 1;
    msk[ttt(1)] = 1;
    msk[ttt(1121)] = 1;
    msk[ttt(221)] = 1;
    msk[ttt(1111)] = 1;
    msk[ttt(11)] = 1;
    msk[ttt(1222)] = 1;
    msk[ttt(212)] = 1;
    msk[ttt(1211)] = 1;
    msk[ttt(22)] = 1;
    msk[ttt(21)] = 1;
    msk[ttt(222)] = 1;
    msk[ttt(1221)] = 1;
    msk[ttt(2212)] = 1;
    msk[ttt(121)] = 1;
    msk[ttt(111)] = 1;
    msk[ttt(2)] = 1;
    msk[ttt(112)] = 1;
    msk[ttt(1112)] = 1;
    msk[ttt(122)] = 1;
    msk[ttt(2112)] = 1;
    msk[ttt(2122)] = 1;
    msk[ttt(2211)] = 1;

}

bool IsValid(int st, int len)
{
    int i, res, t, cube;
    for (i = res = 0, cube = 1; i < len; i++)
    {
        // printf("i = %d\n", i);
        t = ((str[st - i - 1] == '.') ? 1 : 2);
```

```
63.         res += t * cube;
64.         cube *= 3;
65.     }
66.     // printf("res = %d\n", res);
67.     return msk[res] == 1;
68. }
69.
70. void Solve()
71. {
72.     int i, j, n;
73.     scanf("%s", str);
74.     n = strlen(str);
75.     memset (dp, 0, sizeof dp);
76.     dp[0] = 1;
77.     for (i = 1; i <= n; i++)
78.         for (j = 1; j <= 4 && j <= i; j++)
79.         {
80.             if (IsValid(i, j))
81.             {
82.                 dp[i] += dp[i - j];
83.                 dp[i] %= MOD;
84.             }
85.         }
86.     printf("%d\n", dp[n]);
87. }
88.
89. int main()
90. {
91.     int cas, i;
92.     scanf("%d", &cas);
93.     PreProcess();
94.     for (i = 1; i <= cas; i++)
95.     {
96.         printf("Case #%d: ", i);
97.         Solve();
98.     }
99.     return 0;
100. }
```

M-Play Chess in Cafe

SOLUTION

显而易见的 SG 函数问题。

利用搜索可以得出每一个状态的 mex 值，计算每一盘棋的 sg 值再异或起来。

异或值为真则先手胜，为 0 则后手胜。

STDCODE

```
#include<iostream>
#include<cstring>
#include<map>

using namespace std;
const int maxr=4,maxn=8,maxm=6;
const int gx[maxm][25]={
    {3,1,1,1},
    {21,1,2,3,4,5,6,7,1,2,3,4,5,6,7,1,2,3,4,5,6,7},
    {7,1,2,3,4,5,6,7},
    {14,1,2,3,4,5,6,7,1,2,3,4,5,6,7},
    {4,1,2,2,1},
    {1,1}
};
const int gy[maxm][25]={
    {3,-1,0,1},
    {21,-1,-2,-3,-4,-5,-6,-7,0,0,0,0,0,0,0,1,2,3,4,5,6,7},
    {7,0,0,0,0,0,0,0},
    {14,-1,-2,-3,-4,-5,-6,-7,1,2,3,4,5,6,7},
    {4,2,1,1,2},
    {1,0}
};
class pos{
public:
    int x[maxm],y[maxm];
};
pos s;
bool board[maxr][maxn];
unsigned char sg[10010];
int n;
map<long long,unsigned char> mex;
//int mex[1<<24];

```

```
int encode(pos p)
{
    int code=0;
    for(int i=0;i<maxm;i++)
        code=code*maxr*maxn+p.x[i]*maxn+p.y[i];
    return code;
}
unsigned char dfs(pos p)
{
    bool dsg[30]={0};
    int x,y,xx,yy,nextcode;
    for(int i=0;i<maxm;i++)
    {
        int cnt=gx[i][0];
        x=p.x[i];y=p.y[i];
        for(int j=1;j<=cnt;j++)
        {
            xx=x+gx[i][j];yy=y+gy[i][j];
            if(xx<0||xx>=maxr||yy<0||yy>=maxn) continue;
            if(board[xx][yy]) continue;
            board[x][y]=0;
            board[xx][yy]=1;
            p.x[i]=xx;p.y[i]=yy;
            nextcode=encode(p);
            if(mex.find(nextcode)==mex.end())
            {
                mex[nextcode]=dfs(p);
            }
            dsg[mex[nextcode]]=true;
            p.x[i]=x;p.y[i]=y;
            board[xx][yy]=0;
            board[x][y]=1;
        }
    }
    for(unsigned char i=0;;i++)
        if(!dsg[i]) return i;
}

void init()
{
    cin >> n;
    memset(sg,0,sizeof(sg));
    for(int k=0;k<n;k++)
    {
```

```
78.         memset(board,0,sizeof(board));
79.         memset(&s,0,sizeof(s));
80.         for(int i=0;i<maxm;i++)
81.         {
82.             cin >> s.x[i] >> s.y[i];
83.             board[s.x[i]][s.y[i]]=1;
84.
85.         }
86.         sg[k]=dfs(s);
87.     }
88. }
89. void work()
90. {
91.     int sum=0;
92.     for(int i=0;i<n;i++)
93.         sum^=sg[i];
94.     if(sum)
95.         cout << "use_nature is lucky to win." << endl ;
96.     else cout << "Lalatina has a powerful brain." << endl ;
97. }
98. int main()
99. {
100.     ios::sync_with_stdio(false);
101.     int T;
102.     cin >> T;
103.     while(T--)
104.     {
105.         init();
106.         work();
107.     }
108.     return 0;
109. }
```