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（初赛）

网络赛题解

B - East Ninth Road

SOLUTION

用线段树维护区间最大子段，包含左端点的最大子段，包含最右端的最大子段即可

STDCODE

```cpp
#include<iostream>
#include<cstdio>
using namespace std;
struct data1{
        int lm,rm,mx;
        int sum;
        int l,r;
        }tree[2000001];
int n,m;
int a[500001];
void update(int k)
{
     tree[k].sum=tree[k<<1].sum+tree[k<<1|1].sum;
     tree[k].lm=max(tree[k<<1].lm,tree[k<<1].sum+tree[k<<1|1].lm);
     tree[k].rm=max(tree[k<<1|1].rm,tree[k<<1].rm+tree[k<<1|1].sum);
     tree[k].mx=max(max(tree[k<<1].mx,tree[k<<1|1].mx),tree[k<<1].rm+tree[k<<1|1].lm);
 }
void build(int k,int s,int t)
{
     tree[k].l=s;tree[k].r=t;
     if(s==t){tree[k].sum=tree[k].lm=tree[k].rm=tree[k].mx=a[s];return;}
     int mid=(s+t)>>1;
     build(k<<1,s,mid);
     build(k<<1|1,mid+1,t);
     update(k);
}
data1 ask(int k,int p,int q)
{
    data1 g,h,a;
    int l=tree[k].l,r=tree[k].r;
    if(l==p&&q==r)return tree[k];
    int mid=(l+r)>>1;
    if(q<=mid) return ask(k<<1,p,q);
    else if(p>mid) return ask(k<<1|1,p,q);
    else{
    g=ask(k<<1,p,mid);
    h=ask(k<<1|1,mid+1,q);
    a.mx=max(max(g.mx,h.mx),g.rm+h.lm);
    a.rm=max(h.sum+g.rm,h.rm);
    a.lm=max(g.lm,g.sum+h.lm);
    return a;
   }
}
void change(int k,int x,int y)
{
     int l=tree[k].l,r=tree[k].r;
     if(l==r){tree[k].sum=tree[k].lm=tree[k].rm=tree[k].mx=y;return;}
     int mid=(l+r)>>1;
     if(x<=mid)change(k<<1,x,y);
     else if(x>mid)change(k<<1|1,x,y);
     update(k);
 }
int main()
{
    freopen("input.in","r",stdin);
    freopen("output.txt","w",stdout);
    while(scanf("%d%d",&n,&m)==2){
            for(int i=1;i<=n;i++)
       scanf("%d",&a[i]);
    build(1,1,n);
    for(int i=1;i<=m;i++)
    {
            int t,x,y;
            scanf("%d%d%d",&t,&x,&y);
            if(t==1)
            {
                    if(x>y)swap(x,y);
                    printf("%d\n",ask(1,x,y).mx);
                    }
            if(t==2) change(1,x,y);
            }
    }

    return 0;
}
```

C - Rubik

SOLUTION

模拟题，写出 12 种旋转变换函数，注意细节即可。详见代码。

STDCODE

```c
#include <stdio.h>
#include <string.h>

char str[1000];
char res[30];
int msk[30];

void ClockRotate(int n, int *arr)
{
    int i, t = res[arr[n - 1]];
    for (i = n - 2; i >= 0; i--)
        res[arr[i + 1]] = res[arr[i]];
    res[arr[0]] = t;
}

void U()
{
    int i;
    msk[0] = 1;     msk[1] = 2;
    msk[2] = 4;     msk[3] = 3;
    ClockRotate(4, msk);
    msk[0] = 5;     msk[1] = 13;
    msk[2] = 9;     msk[3] = 17;
    ClockRotate(4, msk);
    msk[0] = 6;     msk[1] = 14;
    msk[2] = 10;    msk[3] = 18;
    ClockRotate(4, msk);
}

void F()
{
    int i;
    msk[0] = 5;     msk[1] = 6;
    msk[2] = 8;     msk[3] = 7;
    ClockRotate(4, msk);
    msk[0] = 3;     msk[1] = 17;
    msk[2] = 22;    msk[3] = 16;
    ClockRotate(4, msk);
    msk[0] = 4;     msk[1] = 19;
    msk[2] = 21;    msk[3] = 14;
    ClockRotate(4, msk);
}

void B()
{
    int i;
    msk[0] = 9;     msk[1] = 10;
    msk[2] = 12;    msk[3] = 11;
    ClockRotate(4, msk);
    msk[0] = 1;     msk[1] = 15;
    msk[2] = 24;    msk[3] = 18;
    ClockRotate(4, msk);
    msk[0] = 2;     msk[1] = 13;
    msk[2] = 23;    msk[3] = 20;
    ClockRotate(4, msk);
}

void L()
{
    int i;
    msk[0] = 13;    msk[1] = 14;
    msk[2] = 16;    msk[3] = 15;
    ClockRotate(4, msk);
    msk[0] = 1;     msk[1] = 5;
    msk[2] = 21;    msk[3] = 12;
    ClockRotate(4, msk);
    msk[0] = 3;     msk[1] = 7;
    msk[2] = 23;    msk[3] = 10;
    ClockRotate(4, msk);
}

void R()
{
    int i;
    msk[0] = 17;    msk[1] = 18;
    msk[2] = 20;    msk[3] = 19;
    ClockRotate(4, msk);
    msk[0] = 6;     msk[1] = 2;
    msk[2] = 11;    msk[3] = 22;
    ClockRotate(4, msk);
    msk[0] = 8;     msk[1] = 4;
    msk[2] = 9;     msk[3] = 24;
    ClockRotate(4, msk);
}

void D()
{
    int i;
    msk[0] = 21;    msk[1] = 22;
    msk[2] = 24;    msk[3] = 23;
    ClockRotate(4, msk);
    msk[0] = 7;     msk[1] = 19;
    msk[2] = 11;    msk[3] = 15;
    ClockRotate(4, msk);
    msk[0] = 8;     msk[1] = 20;
    msk[2] = 12;    msk[3] = 16;
    ClockRotate(4, msk);
}

void Init()
{
    int i;
    for (i = 0; i < 4; i++)
    {
        res[i + 1] = 'Y';
        res[i + 5] = 'R';
        res[i + 9] = 'O';
        res[i + 13] = 'B';
        res[i + 17] = 'G';
        res[i + 21] = 'W';
    }
}

void Print()
{
    for (int i = 1; i <= 24; i++)
        printf("%c", res[i]);
    printf("\n");
}

void Solve()
{
    int n, i;
    char c;
    scanf("%d%s", &n, str);
    Init();
    for (i = 0; i < n; i++)
    {
        c = str[i];
        if (c == 'U') { U(); }
        if (c == 'D') { D(); }
        if (c == 'L') { L(); }
        if (c == 'R') { R(); }
        if (c == 'F') { F(); }
        if (c == 'B') { B(); }
        if (c == 'u') { U(); U(); U(); }
        if (c == 'd') { D(); D(); D(); }
        if (c == 'l') { L(); L(); L(); }
        if (c == 'r') { R(); R(); R(); }
        if (c == 'f') { F(); F(); F(); }
        if (c == 'b') { B(); B(); B(); }
    }
    Print();
}

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

D - Triangle Plus

SOLUTION

对于问题一

C(n, 3) 便是答案

对于问题二

首先我们应该知道最大的三角形的三个点一定是在凸包上，现在问题变成在凸包上找三个点组成一个面积最大的三角形。由于 N 是相当大的，故不能枚举。

于是我们可以用旋转卡壳

时间复杂度为 nlogn

STDCODE

```cpp
#include<bits/stdc++.h>

#define  IN  freopen("in.txt","r",stdin);
#define  OUT freopen("out.txt","w",stdout);

using namespace std;
const int maxn = 100000 + 10;

struct Point
{
    double x, y;
    Point(){}
    Point(double x_, double y_):x(x_),y(y_){}
} p[maxn], ch[maxn];
typedef Point Vector;
int n;

Vector operator + (Vector a, Vector b)  { return Vector(a.x + b.x, a.y + b.y); }
Vector operator - (Point a, Point b)    { return Point(a.x - b.x, a.y - b.y); }
Vector operator * (Vector a, double p)  { return Vector(a.x * p, a.y * p); }
Vector operator / (Vector a, double p)  { return Vector(a.x / p, a.y / p); }

bool operator < (const Point& a, const Point& b)
{
    return a.x < b.x || (a.x == b.x && a.y < b.y);
}

const double eps = 1e-10;
int dcmp(double x)
{
    if(fabs(x) < eps)    return 0;
    else                 return x < 0 ? -1 : 1;
}

bool operator == (const Point& a, const Point& b)
{
    return dcmp(a.x - b.x) == 0 && dcmp(a.y - b.y) == 0;
}

double dot(Vector a, Vector b)      { return a.x * b.x + a.y * b.y; }
double length(Vector a)             { return sqrt(dot(a, a)); }
double angle(Vector a, Vector b)    { return acos(dot(a, b) / length(a) / length(b)); }
double angle(Vector v)              { return atan2(v.y, v.x); }
double cross(Vector a, Vector b)    { return a.x * b.y - b.x * a.y; }
double dist(Point p1,Point p2)      { return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y)); }

int convexHull(Point* p, int n, Point* ch)
{
    sort(p, p + n);
    int m = 0;
    for(int i = 0; i < n; i++) {
        while(m > 1 && cross(ch[m-1] - ch[m-2], p[i] - ch[m-2]) <= 0) m--;
        ch[m++] = p[i];
    }
    int k = m;
    for(int i = n-2; i >= 0; i--) {
        while(m > k && cross(ch[m-1] - ch[m-2], p[i] - ch[m-2]) <= 0) m--;
        ch[m++] = p[i];
    }
    if(n > 1)    m--;
    return m;
}

double area(Point a,Point b,Point c){
    return (a.x-c.x)*(b.y-c.y)-(b.x-c.x)*(a.y-c.y);
}

```

```
double solve()
{
    int m = convexHull(p, n, ch);
    int a = 1, b = 2;
    double res = 0;
    for(int i = 0; i < m; i++) {
        while(area(ch[i], ch[a], ch[(b+1)%m]) > area(ch[i], ch[a], ch[b]))
            b = (b + 1) % m;
        res = max(res, area(ch[i], ch[a], ch[b]) / 2.0);
        while(area(ch[i], ch[(a+1)%m], ch[b]) > area(ch[i], ch[a], ch[b]))
            a = (a + 1) % m;
        res = max(res, area(ch[i], ch[a], ch[b]) / 2.0);
    }
    return res;
}

int main()
{
    //IN
    //OUT
    while(scanf("%d", &n) != EOF) {
        for(int i = 0; i < n; i++) {
            scanf("%lf%lf", &p[i].x, &p[i].y);
        }

        long long num = (long long) n;
        long long sum = (long long)((num * (num - 1) * (num - 2)) / 6);
        printf("%I64d\n",sum);
        printf("%.2f\n", solve());
    }
    return 0;
}
```

E - Money

SOLUTION

0-1 背包问题，求出最大的 n 使得 n 为偶数且 n/2 和 n 都能够由钞票组成，最终答案为（sum-n）＋n/2 sum 为钞票面值总和

STDCODE

```cpp
#include<cstdio>
#include<cstring>
using namespace std;
#define maxn 500
#define maxm 100000
int a[maxn+10];
bool dp[maxm+10];
int main() {
    //freopen("data9.in","r",stdin);
    //freopen("data9.out","w",stdout);
    int n;
    scanf("%d",&n);
    int sum=0;
    for(int i=1;i<=n;i++)
        scanf("%d",&a[i]),sum+=a[i];
    memset(dp,false,sizeof(dp)); dp[0]=true;
    for(int i=1;i<=n;i++)
        for(int j=sum;j>=a[i];j--)
            if(dp[j-a[i]]) dp[j]=true;
    /*for(int i=1;i<=sum;i++)
        if(dp[i]) printf("true_%d\n",i);*/
    int ans=0;
    for(int i=sum;i>=0;i--)
        if(!(i&1) && dp[i] && dp[i>>1]) {
            ans=i>>1; break;
        }
    //printf("%d\n",ans);
    printf("%d\n",sum-ans);
    return 0;
}
```

F- Random at Random

SOLUTION

纯数学题

最标准的做法是推公式，见“矩母函数”

答案是 E(N)Var(X)+((E(X))^2)Var(N)

其中 E(N)是 N 的期望

E(X)是 X 的期望

Var(N)是 N 的方差

Var(X)是 X 的方差

其中要用到一个基本关系是，对于随机变量 X 有：

Var(X)=E(X^2)-(E(X))^2

打表可能也能看出来

STDCODE

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

int T, en, vn, ex, vx, ans;
int main()
{
    //freopen("a.in","r",stdin);
    //freopen("a.out","w",stdout);
    scanf("%d",&T);
    while(T--)
    {
        scanf("%d%d%d%d",&en,&vn,&ex,&vx);
        ans = en*vx + ex*ex*vn;
        printf("%d\n",ans);
    }
}
```

G – Sorting

SOLUTION

在原串中求一个最长子串使得它是排序后的串的连续子串，答案是原串减去该串的长度。

用一个队列来维护这个最长子串，假设当前队列中已经维护了最大数字为 j 的最长子串，我们先考虑求到最大数字为 j+1 的最长子串，从右到左遍历数字 j+1 在原串中出现的位置，然后处理下就可以求出。再把数字 j+1 加入到队列里面。

复杂度 O(max(Ai))

STDCODE

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<vector>
#include<map>
#include<iomanip>
#include<queue>
#include<set>
using namespace std;
const int SIZE = 100010;
int a[SIZE], q[SIZE];
int n, m, ans;
vector<int> b[SIZE];

int main()
{
    freopen("A.in","r",stdin);
    freopen("A.out","w",stdout);
    int t;
    scanf("%d",&t);
    for (int cas=1;cas<=t;++cas)
    {
        m=0;
        ans=0;
        scanf("%d",&n);
        for (int i = 1; i <= n; i++)
        {
            scanf("%d", &a[i]);
            b[a[i]].push_back(i);
            m = max(m, a[i]);
        }
        int l = 1, r = 0;
        for (int i = 1; i <= m; i++)
        {
            reverse(b[i].begin(), b[i].end());
            for (int j = 0; j < b[i].size(); j++)
            {
                int k = b[i][j];
                while (l <= r && q[r] > k)
                {
                    while (l < r && a[q[l]] < a[q[r]]) l++;
                    r--;
                }
                ans = max(ans, r - l + 2 + j);
            }
            for (int j = b[i].size() - 1; j >= 0; j--)
            {
                q[++r] = b[i][j];
            }
        }
        printf("%d\n",n-ans);
        for (int i = 1; i <= n; i++)
        {
            b[a[i]].clear();
        }
    }
}
```

H - Smooth Project

SOLUTION

裸并查集。给一条边然后把两个点加入到集合中，以后的每条边得点如果能在前面找到，就加入那个集合，否则自创一个集合，最后总计一共有多少个集合，把总数减一即可。

STDCODE

```
#include<cstdio>
#include<iostream>
#include<cstring>
#define MAXN 1005
using namespace std;

int pre[MAXN],total;

int Find(int x){
    int r=x;
    while(pre[r]!=r)
    r=pre[r];
    int i=x,j;
    while(i!=r){
        j=pre[i];
        pre[i]=r;
        i=j;
    }
    return r;
}

void join(int x,int y){
    int fx=Find(x),fy=Find(y);
    if(fx!=fy){
        pre[fy]=fx;
        total--;
    }
}

int main(){
    //freopen("data.txt","r",stdin);
    //freopen("dataout.txt","w",stdout);
    int n,m,a,b,i;
    while(scanf("%d",&n) && n){
        scanf("%d",&m);
        total=n-1;
        for(i=1;i<=n;i++)
        pre[i]=i;
        while(m--){
            scanf("%d %d",&a,&b);
            join(a,b);
        }
        printf("%d\n",total);
    }
```

```cpp
    return 0;
}
```

I - Key

SOLUTION

建立 AC 自动机，进行匹配，记录每个节点能不能在匹配的时候遍历到。然后在 trie 上进行一次遍历，对于每个节点记录从根到他自己这条路径上能在匹配过程中被访问到的节点的最大深度 h。对于每个 Servant Key，其最长能在 Master Key 中匹配到的前缀的长度就是他在 trie 上记录的 h。

STDCODE

```cpp
#include <cstdio>
#include <cstring>

using namespace std;

int B[256], N, M;

struct tnode {
    tnode *c[4], *fail;
    int len;
    bool vis;
} Buf[1000001], *R[10000], *Q[1000001];
int Cnt;

char KM[1000002], S[102];
int KK[1000001];

inline tnode *tnew() {
    memset(&Buf[Cnt], 0, sizeof(tnode));
    return &Buf[Cnt++];
}

int main () {
    B['N'] = 0;
    B['S'] = 1;
    B['W'] = 2;
    B['E'] = 3;
    int nCases;
    scanf("%d\n", &nCases);
    for (int iCase = 1; iCase <= nCases; ++iCase) {
        fgets(KM, 1000002, stdin);
        M = 0;
        for (char *c = KM; *c && *c != '\n'; ++c)
            KK[M++] = B[(int) *c];
        Cnt = 0;
        tnode *root = tnew();
        scanf("%d\n", &N);
        for (int i = 0; i < N; ++i) {
            fgets(S, 102, stdin);
            R[i] = root;
            for (char *c = S; *c && *c != '\n'; ++c) {
                if (!R[i]->c[B[(int) *c]])
                    R[i]->c[B[(int) *c]] = tnew();
                R[i] = R[i]->c[B[(int) *c]];
            }
        }
        int ql = 0;
        for (int i = 0; i < 4; ++i)
            if (root->c[i]) {
                root->c[i]->fail = root;
                Q[ql++] = root->c[i];
            }
        for (int qf = 0; qf < ql; ++qf)
            for (int i = 0; i < 4; ++i) {
                if (Q[qf]->c[i]) {
                    tnode *fail = Q[qf]->fail;
                    while (fail != root && !fail->c[i])
                        fail = fail->fail;
                    Q[qf]->c[i]->fail = fail->c[i] ? fail->c[i] : root;
                    Q[ql++] = Q[qf]->c[i];
                }
            }
        tnode *j = root;
        for (int i = 0; i < M; ++i) {
            while (!j->c[KK[i]] && j != root)
                j = j->fail;
            j = j->c[KK[i]];
            if (!j)
                j = root;
            for (tnode *k = j; k != root && !k->vis; k = k->fail)
                k->vis = true;
        }
        Q[0] = root;
        for (int qf = 0, ql = 1; qf < ql; ++qf)
            for (int i = 0; i < 4; ++i)
                if (Q[qf]->c[i]) {
                    Q[qf]->c[i]->len = Q[qf]->len;
                    if (Q[qf]->c[i]->vis)
                        ++Q[qf]->c[i]->len;
                    Q[ql++] = Q[qf]->c[i];
                }
        printf("Case #%d:\n", iCase);
        for (int i = 0; i < N; ++i)
            printf("%d\n", R[i]->len);
    }
    return 0;
}
```

J - Lucky Sum

SOLUTION

按位 DP。详解见标程。

STDCODE

```
#include <stdio.h>
#include <iostream>
#include <string.h>
#include <string>
#include <assert.h>
#include <algorithm>

using namespace std;

//standard code for Lucky Sum
//by sheep
//2011.12.06

```

```
int f[2][2][2][2][4];
char a[300007],b[300007];

void init(int n)
{
    for (int i = 0;i < 2;++i)
        for (int j = 0;j < 2;++j)
            for (int r = 0;r < 2;++r)
                for (int k = 0;k < 4;++k)
                    f[n][i][j][r][k] = -0x7f7f7f7f;
}

void update(int &a,int b)
{
    a = max(a,b);
}

int main()
{
    while (scanf("%s%s",a,b) != EOF)
    {
        int l1 = strlen(a),l2 = strlen(b);
        assert(l1 <= l2);
        if (l1 == l2) assert(strcmp(a,b)<=0);
        int n = l2;

        for (int i = 0;i < l1;++i)
            a[i] -= '0';
        for (int i = 0;i < l2;++i)
            b[i] -= '0';
        reverse(a,a+l1);
        reverse(b,b+l2);
        reverse(a,a+n);
        reverse(b,b+n);
        init(0);
        f[0][0][0][0][3] = 0;
      //  cout<<n<<endl;
        for (int i = 0;i < n;++i)
        {
            int nowi = i & 1,nexti = nowi^1;
            init(nexti);
            for (int j = 0;j < 2;++j)
                for (int k = 0;k < 2;++k)
                    for (int z = 0;z < 2;++z)
```

```
                        for (int r = 0;r < 4;++r)
                        {
                            if (f[nowi][j][k][z][r] < 0) continue;
                            for (int now = 0;now < 10;++now)
                            {
                                int nextj = j,nextk = k,nextr = r+1;
                                if (!j && !k)
                                {
                                    if (now < a[i]) continue;
                                    if (now > b[i]) continue;
                                    if (now > a[i]) nextj = 1;
                                    if (now < b[i]) nextk = 1;
                                }
                                if (!j && k)
                                {
                                    if (now < a[i]) continue;
                                    if (now > a[i]) nextj = 1;
                                }
                                if (!k && j)
                                {
                                    if (now > b[i]) continue;
                                    if (now < b[i]) nextk = 1;
                                }
                                if (now == 4 || now == 8)
                                    nextr = 0;
                                if (z)
                                {
                                    if (nextr < 4)
                                        update(f[nexti][nextj][nextk][z][nextr],f[nowi][j][k][z][r]+now);
                                }
                                else
                                {
                                    assert(f[nowi][j][k][z][r] == 0);
                                    assert(r == 3);
                                    if (now)
                                    {
                                        if (nextr < 4)
                                            update(f[nexti][nextj][nextk][1][nextr],f[nowi][j][k][z][r]+now);
                                    }
                                    else
                                    {
                                        assert(nextr == 4);
                                        update(f[nexti][nextj][nextk][z][3],f[nowi][j][k][z][r]+now);
                                    }
                                }
                            }
                        }
        }
        int ans = 0;
        for (int i = 0;i < 2;++i)
            for (int j = 0;j < 2;++j)
                for (int k = 0;k < 2;++k)
                    update(ans,f[n&1][i][j][k][0]);
        printf("%d\n",ans);
    }
    return 0;
}
```

K - Orderliness

SOLUTION

见标程注释。

STDCODE

```
/* ************************************************
Author :111qqz
Created Time :2016年10月04日 星期二 23时59分11秒
File Name :code/cf/problem/558E.cpp
 ************************************************ */
#include <cstdio>
#include <cstring>
#include <iostream>
#include <algorithm>
#include <vector>
#include <queue>
#include <set>
#include <deque>
#include <map>
#include <string>
#include <cmath>
#include <cstdlib>
```

```
#include <bitset>
#include <ctime>
#define fst first
#define sec second
#define lson l,m,rt<<1
#define rson m+1,r,rt<<1|1
#define ms(a,x) memset(a,x,sizeof(a))
typedef long long LL;
#define pi pair < int ,int >
#define MP make_pair
using namespace std;
const double eps = 1E-8;
const int dx4[4]={1,0,0,-1};
const int dy4[4]={0,-1,1,0};
const int inf = 0x3f3f3f3f;
const int N=1E5+7;
int tree[27][N<<2];//26 棵线段树，表示 26 个字母在对应区间中的个数。
int lazy[27][N<<2];//lazy 存储的是某段区间是否被某个字母覆盖. -1 表示初始没有标记，0 表示该区间被重置，1 表示该区间被某种颜色覆盖。
int n,q;
char st[N];
int cnt[27]; //计数排序.
void PushUp(int rt,int id)
{
    tree[id][rt] = tree[id][rt<<1] + tree[id][rt<<1|1];
}
void PushDown(int l,int r,int rt,int id)
{
    if (lazy[id][rt]==-1) return ;
    if (l==r) return;
    lazy[id][rt<<1] = lazy[id][rt<<1|1] = lazy[id][rt]; //可以把一段区间被某种字母覆盖看做被染色。。。不过我们不需要知道被哪种字母染色，只需要知道是否被染色。
    if (lazy[id][rt]) //如果一段区间被某种字母覆盖，那么该区间某种字母的数目就是区间长度，这也是 PushDown 需要传递区间端点 l,r 进来的原因。
    {
    int m = (l+r)>>1;
    tree[id][rt<<1] = m-l+1;
    tree[id][rt<<1|1] = r-m; //区间长度
    }
    else tree[id][rt<<1] = tree[id][rt<<1|1] = 0 ; //如果被重置了，清空线段树数组。
    lazy[id][rt] = -1;
    //关于 lazy 标记的思考：lazy 标记是为了延迟更新。。。因此 PushDown 操作的内容其实和 update 时候的东西是非常相似的。。。
```

```
}
void build ( int l,int r,int rt)
{
    if (l==r)
    {
    int id = st[l]-'a'+1;
    tree[id][rt] = 1;
    lazy[id][rt] = 1;
    return;
    }
    int m = (l+r)>>1;
    build(lson);
    build(rson);
    for ( int i = 1 ; i <= 26 ; i++) //26 棵线段树，每一层向上更新的时候每一棵都要记得更新。
    PushUp(rt,i);
}
void update(int L,int R,int sc,int l,int r,int rt,int id)
{
    if (r<L||l>R) return;
    if (L<=l&&r<=R)
    {
    lazy[id][rt] = sc; //sc 为 0 表示重置区间，sc 为 1 表示该区间被 id+'a'-1 的字母覆盖。
    if (sc) tree[id][rt] = r-l+1;
    else tree[id][rt] = 0;
    return;
    }
    PushDown(l,r,rt,id);
    int m = (l+r)>>1;
    update(L,R,sc,lson,id);
    update(L,R,sc,rson,id);
    PushUp(rt,id);
}
int query(int L,int R,int l,int r,int rt,int id)
{
    if (r<L||l>R) return 0;
    if (L<=l&&r<=R) return tree[id][rt];
    PushDown(l,r,rt,id);
    int m = (l+r)>>1;
    //int ret = 0 ;
//    if (L<=m) ret = ret + query(L,R,lson,id);
//    if (R>=m+1) ret = ret + query(L,R,rson,id);
 //   PushUp(rt,id);
```

```
    return query(L,R,lson,id) + query(L,R,rson,id);
}
int main()
{
#ifndef  ONLINE_JUDGE
    //freopen("in.txt","r",stdin);
#endif
    ms(tree,0);
    ms(lazy,-1);
    cin>>n>>q;
    scanf("%s",st+1);
    build(1,n,1);
    while (q--)
    {
    int l,r,k;
    scanf("%d%d%d",&l,&r,&k);
    for ( int i = 1 ; i <= 26 ; i++)
    {
        cnt[i] = query(l,r,1,n,1,i);
      update(l,r,0,1,n,1,i); //初始要把[l,r]区间的每个字母的线段树区间重置，因为排序会打乱，在下面按照 k 的不同以不同方式（正序，逆序）更新。
    }
    if (k==1)
    {
        int pos = l;
        for ( int i = 1 ; i <= 26 ; i++)
        {
        if (cnt[i]==0) continue;
        update(pos,pos+cnt[i]-1,1,1,n,1,i);
        pos = pos + cnt[i];
        }
    }
    else
    {
        int pos = l;
        for ( int i = 26 ; i >= 1 ; i--)
        {
        if (cnt[i]==0) continue;
        update(pos,pos+cnt[i]-1,1,1,n,1,i);
        pos = pos + cnt[i];
        }
    }
    }
    for ( int i = 1 ; i <= n ; i++)
    {
    for ( int j = 1 ; j <= 26 ; j++)
    {
        if (query(i,i,1,n,1,j))  //在第 i 个位置 26 个字母只可能有一种值为 1，其余都为 0，输出的时候 query 就好了。
        {
        printf("%c",j+'a'-1);
        break;
        }
    }
    }
    return 0;
}
```

L - LianLianKan

SOLUTION

题意：连连看游戏，给定一个局面判断最后能不能消完。

思路：用 dfs 来确定消去哪一个格子（x，y），在 dfs 中进行 bfs，看（x，y）周围有多少格子可以和（x，y）相消，找到之后再用 dfs 枚举与哪一个相消或者当前（x，y）不消。

注意一个剪枝，存在下面这种情况的肯定不符合题意：

```
*********
***AB***
***BA***
*********
```

另外，注意题意的消去方法，连线最多只转两个弯。

STDCODE

```
#include <iostream>
#include <cstdio>
```

```
#include <cstring>
#include <algorithm>
#include <cmath>
#include <string>
#include <map>
#include <stack>
#include <vector>
#include <set>
#include <queue>
#pragma comment (linker,"/STACK:102400000,102400000")
#define maxn 1005
#define MAXN 2005
#define mod 1000000009
#define INF 0x3f3f3f3f
#define pi acos(-1.0)
#define eps 1e-6
#define lson rt<<1,l,mid
#define rson rt<<1|1,mid+1,r
#define FRE(i,a,b)  for(i = a; i <= b; i++)
#define FREE(i,a,b) for(i = a; i >= b; i--)
#define FRL(i,a,b)  for(i = a; i < b; i++)
#define FRLL(i,a,b) for(i = a; i > b; i--)
#define mem(t, v)   memset ((t) , v, sizeof(t))
#define sf(n)       scanf("%d", &n)
#define sff(a,b)    scanf("%d %d", &a, &b)
#define sfff(a,b,c) scanf("%d %d %d", &a, &b, &c)
#define pf          printf
#define DBG         pf("Hi\n")
typedef long long ll;
using namespace std;

struct Node
{
    int x,y,turn,w,d; //turn 是走到当前（x，y）处已经转了多少个弯，d 是（x，y）是有上一步哪个方向来的
};

int dir[4][2]={1,0,0,1,0,-1,-1,0};
int mp[11][11],num[4];
int n,m;
char str[11];
bool flag;

bool isok(int x,int y)
```

```
{
    if (x>=0&&x<n&&y>=0&&y<m) return true;
    return false;
}

bool isOK()
{
    for (int i=0;i<n;i++)
    {
        for (int j=0;j<m;j++)
        {
            if (mp[i][j]!=-1&&mp[i][j+1]!=-1&&num[mp[i][j]]==num[mp[i][j+1]]&&num[mp[i][j]]==2)
            {
                if (mp[i][j]==mp[i+1][j+1]&&mp[i][j+1]==mp[i+1][j])
                    return true;
            }
        }
    }
    return false;
}

void bfs(int x,int y,int v,int s[25][2],int &nn)
{
    nn=0;
    queue<Node>Q;
    Node st,now;
    st.x=x; st.y=y; st.w=-1; st.turn=0; st.d=-1;
    bool vis[11][11];
    memset(vis,false,sizeof(vis));
    vis[x][y]=true;
    Q.push(st);
    while (!Q.empty())
    {
        st=Q.front();Q.pop();
        if (st.w==v)
        {
            s[nn][0]=st.x;      //记录下找到的可以相消的格子的坐标
            s[nn++][1]=st.y;
            continue;
        }
        for (int i=0;i<4;i++)
        {
            now.x=st.x+dir[i][0];
```

```
           now.y=st.y+dir[i][1];
           if (isok(now.x,now.y)&&!vis[now.x][now.y])
           {
               if (mp[now.x][now.y]!=-1&&mp[now.x][now.y]!=v) continue;
               if (st.d==i||st.d==-1)
                   now.turn=st.turn;
               else
                   now.turn=st.turn+1;
               if (now.turn<=2)
               {
                   now.w=mp[now.x][now.y];
                    now.d=i;
                    vis[now.x][now.y]=true;
                    Q.push(now);
                }
            }
        }
    }
}

void dfs(int cnt)
{
    if (flag) return ;
    if (cnt==0)
    {
        flag=true;
        return;
    }
    if (isOK()) return ;        //剪枝
    for (int i=0;i<n;i++)
    {
        for (int j=0;j<m;j++)
        {
            if (mp[i][j]!=-1)
            {
                int s[25][2];
                int sum;
                int v=mp[i][j];
                bfs(i,j,v,s,sum);
                num[v]-=2;
                mp[i][j]=-1;
                for (int k=0;k<sum;k++)
                {
                    int x=s[k][0];
                    int y=s[k][1];
                    mp[x][y]=-1;
                    dfs(cnt-2);
                    mp[x][y]=v;
                }
                num[v]+=2;
                mp[i][j]=v;
            }
        }
    }
}

int main()
{
    freopen("out.txt","r",stdin);
    freopen("ans.txt","w",stdout);
    int i,j;
    while (scanf("%d%d",&n,&m))
    {
        if (n==0&&m==0) break;
        flag=false;
        mem(num,0);
        mem(mp,-1);
        int all=0;
        for (i=0;i<n;i++)
        {
            scanf("%s",str);
            for (j=0;j<m;j++)
            {
                if (str[j]=='*') mp[i][j]=-1;
                else if (str[j]=='A'){
                    num[0]++;
                    mp[i][j]=0;
                    all++;
                }
                else if (str[j]=='B')
                {
                    num[1]++;
                    mp[i][j]=1;
                    all++;
                }
                else if (str[j]=='C')
                {
                    num[2]++;
                    mp[i][j]=2;
                    all++;
                }
                else{
                    num[3]++;
                    mp[i][j]=3;
                    all++;
                }
            }
        }
        if (num[0]%2||num[1]%2||num[2]%2||num[3]%2) //存在奇数时肯定不合要求
        {
            pf("no\n");
            continue;
        }
        dfs(all);
        if (flag)
            pf("yes\n");
        else
            pf("no\n");
    }
    return 0;
}
```

M - Breaking News

SOLUTION

首先一看数据范围就要离散化所有的点.

然后利用扫描线的思想将所有行最左边的点作为事件开始点，最右边的点作为结束点。

以 x 为关键字排序所有点，再一路扫描。对于两个 x 坐标相同的点(x,y1),(x,y2),

线段树或者树状数组查询(y1+1,y2-1)区间有多少个事件点，累加答案。

在做 x 列前，先扫描该列的点，将事件点加入线段树或者从线段树上消除。

复杂度 O(nlogn)。

STDCODE

```cpp
#include <iostream>
#include <algorithm>
#include <cstring>
#include <cstdio>
#include <algorithm>
#include<ctime>
using namespace std;
const int maxn=200000+100;
struct node
{
    int x;
    int y;
}p[maxn];
int a[maxn];
int b[maxn];
int n,cur;
bool vis[maxn],l[maxn],r[maxn];
bool cmp(node u,node v)
{
    if(u.y==v.y)
        return u.x<v.x;
    return u.y<v.y;
}
int low(int k)
{
    return k&(-k);
}
void update(int k,int v)
{
    while(k<maxn)
    {
        a[k]+=v;
        k+=low(k);
    }
}
long long sum(int k)
{
    long long ans=0;
    while(k>0)
    {
        ans+=a[k];
        k-=low(k);
    }
    return ans;
}
void init()
{
    memset(a,0,sizeof(a));
    sort(b,b+cur);
    cur=unique(b,b+cur)-b;
    for(int i=0;i<n;i++)
    {
        p[i].x=lower_bound(b,b+cur,p[i].x)-b+1;
        p[i].y=lower_bound(b,b+cur,p[i].y)-b+1;
    }
    sort(p,p+n,cmp);
    memset(vis,0,sizeof(vis));
    for(int i=0;i<n;i++)
    {
       if(!vis[p[i].x])
       {
           l[i]=1;
           vis[p[i].x]=1;
       }
       else
       l[i]=0;
    }
    memset(vis,0,sizeof(vis));
    for(int i=n-1;i>=0;i--)
    {
        if(!vis[p[i].x])
        {
            r[i]=1;
            vis[p[i].x]=1;
        }
        else
        r[i]=0;
    }
}
void solve()
{
    long long ans=0;
    for(int i=0;i<n;)
    {
        int j=i;
       while(j<n-1&&p[j].y==p[j+1].y)
       j++;
       for(int k=i;k<=j;k++)
       {
           if(!l[k]&&r[k])
           {
               update(p[k].x,-1);
           }
       }
       if(i!=j)
       ans+=(sum(p[j].x-1)-sum(p[i].x));
       if(i<j)
       {
           for(int k=i+1;k<j;k++)
            if(!l[k]&&!r[k])
            ans--;
        }
        while(i<=j)
        {
            if(l[i]&&!r[i])
            update(p[i].x,1);
            i++;
        }
    }

    printf("%lld\n",ans+n);
}
int main()
{
    //double start=clock();
    //freopen("in.txt","r",stdin);
    scanf("%d",&n);
    cur=0;
    for(int i=0;i<n;i++)
    {
        scanf("%d%d",&p[i].x,&p[i].y);
        b[cur++]=p[i].x;
        b[cur++]=p[i].y;
    }
    init();
    solve();
    //double end=clock();
    //printf("single case time:%lf\n",(end-start)/CLOCKS_PER_SEC);
    return 0;
}
```