

代码库

上海交通大学

2016 年 10 月 16 日

目录

| | | |
|----------|-----------------------------|-----------|
| 1 | 数论 | 2 |
| 1.1 | 中国剩余定理 | 2 |
| 1.2 | FFT | 2 |
| 1.3 | 组合数学（卢卡斯定理、线性筛逆元） | 3 |
| 1.4 | 辛普森自适应积分 | 3 |
| 1.5 | 线性筛 | 3 |
| 2 | 图论 | 4 |
| 2.1 | Tarjan | 4 |
| 2.2 | 最大流 | 5 |
| 2.3 | 最小费用流 | 7 |
| 2.4 | 平面图转对偶图 | 7 |
| 3 | 数据结构 | 14 |
| 3.1 | K-D 树 | 14 |
| 3.2 | 可持久化 Trie | 16 |
| 3.3 | Link-Cut-Tree | 17 |
| 3.4 | 可持久化线段树 | 18 |
| 3.5 | 左偏树 | 19 |
| 4 | 字符串 | 20 |
| 4.1 | AC 自动机 | 20 |
| 4.2 | 后缀数组 | 21 |
| 4.3 | 后缀自动机 | 22 |
| 4.4 | 回文自动机 | 24 |
| 4.5 | Manacher 算法 | 24 |
| 5 | 博弈 | 25 |
| 5.1 | 二分图博弈 | 25 |

1 数论

1.1 中国剩余定理

```
typedef long long LL;
LL a[N], r[N], n;
void exgcd(LL a, LL b, LL &d, LL &x, LL &y) {
    if (!b) {d = a; x = 1; y = 0;}
    else {exgcd(b, a%b, d, y, x); y -= (a / b) * x;}
}

LL ex_CRT(LL *m, LL *r, int n) {
    LL M = m[1], R = r[1], x, y, d;
    for(int i = 2; i <= n; i++) {
        exgcd(M, m[i], d, x, y);
        if ((r[i] - R) % d) return -1;
        x = (r[i] - R) / d * x % (m[i] / d);
        R += x * M;
        M = M / d * m[i];
        R %= M;
    }
    return R > 0 ? R : R + M;
}
```

1.2 FFT

```
struct comp {
    double r, i;
    comp(double r = 0.0, double i = 0.0) : r(r), i(i) {}
    comp operator + (const comp &b) const {return comp(r + b.r, i + b.i);}
    comp operator - (const comp &b) const {return comp(r - b.r, i - b.i);}
    comp operator * (const comp &b) const
        {return comp(r * b.r - i * b.i, r * b.i + i * b.r);}
}a[N], b[N], c[N];
void FFT(comp *a, int n, int type) {
    for(int i = 1; j = 0; i < n - 1; i++) {
        for(int s = n; j ^= s >= 1, ~j&s);
        if (i < j) swap(a[i], a[j]);
    }
    for(int m = 1; m < n; m <= 1) {
        double u = pi / m * type;
        comp wm(cos(u), sin(u));
        for(int i = 0; i < n; i += (m < 1)) {
            comp w(1, 0);
```

```

    for(int j = 0; j < m; j++) {
        comp &A = a[i + j + m], &B = a[i + j], t = w * A;
        A = B - t; B = B + t; w = w * wm;
    }
}
}
if (type == -1) for(int i = 0; i < n; i++) a[i].r /= n;
}

```

1.3 组合数学（卢卡斯定理、线性筛逆元）

```

//Lucas
int C(int n, int m) {
    if (n < m) return 0;
    if (n < P && m < P) return fac[n] * inv[m] % P * inv[n - m] % P;
    return C(n % P, m % P) * C(n / P, m / P);
}

//inv
for(int i = 1; i <= n; i++)
    inv[i] = (LL)inv[P % i] * (P - P / i) % P;

```

1.4 辛普森自适应积分

```

double getf(double x) {
}

double calc(double l, double fl, double fmd, double fr) {
    return (fl + fmd * 4 + fr) * 1 / 6;
}

double simpson(double l, double mid, double r, double fl, double fmd, double fr, double s) {
    double m1 = (l + mid) / 2, m2 = (mid + r) / 2;
    double f1 = getf(m1), f2 = getf(m2);
    double g1 = calc(mid - l, fl, f1, fmd), g2 = calc(r - mid, fmd, f2, fr);
    if (fabs(g1 + g2 - s) < eps) return g1 + g2;
    return simpson(l, m1, mid, fl, f1, mid, g1) +
        simpson(mid, m2, r, fmd, f2, fr, g2);
}

```

1.5 线性筛

```

int tot, prime[N], mu[N], sum[N];
bool check[N];
void getmu() {
    mu[1] = 1;
    for(int i = 2; i < n; i++) {

```

```

    if (!check[i]) {
        prime[++ tot] = i;
        mu[i] = -1;
    }
    for(int j = 1; j <= tot; j ++) {
        if (i * prime[j] > n) break;
        check[i * prime[j]] = 1;
        if (i % prime[j]) mu[i * prime[j]] = -mu[i];
        else {
            mu[i * prime[j]] = 0;
            break;
        }
    }
}
for(int i = 1; i < n; i ++) sum[i] = sum[i - 1] + mu[i];
}

```

2 图论

2.1 Tarjan

```

//SCC
int dfn[N], low[N], dfs_clock, belong[N], SCC, size[N];
int st[N], top;
bool in[N];
void tarjan(int x) {
    dfn[x] = low[x] = ++dfs_clock;
    st[top++] = x;
    in[x] = 1;
    for(int i = head[x]; i; i = nxt[i])
        if (!dfn[to[i]]) {
            tarjan(to[i]);
            low[x] = min(low[x], low[to[i]]);
        } else if (in[to[i]]) low[x] = min(low[x], dfn[to[i]]);
    if (low[x] == dfn[x]) {
        SCC ++; size[SCC] = 0;
        for(int y = 0; y != x;) {
            y = st[--top];
            in[y] = 0;
            belong[y] = num;
            size[num] ++;
        }
    }
}
}

```

```

}

//BCC
int dfn[N], low[N], dfs_clock, bccno[N], have[N], size[N], bcc, top;
bool iscut[N];
void tarjan(int x, int fa) {
    low[x] = dfn[x] = ++dfs_clock;
    int child = 0;
    for(int i = head[x]; i; i = nxt[i]) {
        if (!dfn[to[i]]) {
            child ++;
            tarjan(to[i], x);
            low[x] = min(low[to[i]], low[x]);
            if (low[to[i]] >= dfn[x]) iscut[x] = 1;
        } else low[x] = min(low[x], dfn[to[i]]);
    }
    if (fa < 0 && child == 1) iscut[x] = 0;
}

bool vis[N];
void dfs(int x) {
    vis[x] = 1; size[bcc] ++;
    for(int i = head[x]; i; i = nxt[i])
        if (!vis[to[i]]) {
            if (!iscut[to[i]]) dfs(to[i]);
            else if (bccno[to[i]] != bcc)
                bccno[to[i]] = bcc, have[bcc] ++;
        }
}

```

2.2 最大流

```

const int N = 1010, M = 100010;
const int INF = 1e9;
struct edge{
    int to, v;
}E[M << 1];
int head[N], nxt[M << 1], cnt = 1;

void add(int x, int y, int z) {
    E[++ cnt] = (edge){y, z}; nxt[cnt] = head[x]; head[x] = cnt;
    E[++ cnt] = (edge){x, 0}; nxt[cnt] = head[y]; head[y] = cnt;
}

```

```

int S, T, d[N], cur[N];
bool mklevel() {
    memset(d, -1, sizeof d);
    Q.push(S);
    d[S] = 0;
    while(!Q.empty()) {
        int x = Q.front(); Q.pop();
        for(int i = head[x]; i; i = nxt[i])
            if (d[E[i].to] == -1 && E[i].v) {
                d[E[i].to] = d[x] + 1;
                Q.push(E[i].to);
            }
    }
    return d[T] != -1;
}

int dfs(int x, int a) {
    if (x == T || a == 0) return a;
    int flow = 0;
    for(int &i = cur[x]; i; i = nxt[i])
        if (d[E[i].to] == d[x] + 1 && E[i].v) {
            int f = dfs(E[i].to, min(E[i].v, a - flow));
            E[i].v -= f;
            E[i ^ 1].v += f;
            flow += f;
            if (f == a) break;
        }
    if (!flow) d[x] = -1;
    return flow;
}

int Dinic() {
    int ans = 0;
    while(mklevel()) {
        for(int i = 0; i <= T; i++) cur[i] = head[i];
        ans += dfs(S, INF);
    }
    return ans;
}

```

2.3 最小费用流

```
int cost, flow;
struct edge {
    int from, to, v, c;
}E[M];
void ins(int x, int y, int z, int c) {
    E[++ cnt] = (edge){x, y, z, c};
    nxt[cnt] = heda[x]; head[x] = cnt;
}
void add(int x, int y, int z, int c) {
    ins(x, y, z, c); ins(y, x, 0, -c);
}
int S, T, d[N], from[N], Q[M];
bool inq[N];
bool spfa() {
    int l = 0, r = -1;
    for(int i = 0; i <= T; i ++)
        if (d[E[i].to] > d[x] + E[i].c && E[i].v) {
            d[E[i].to] = d[x] + E[i].c;
            from[E[i].to] = i;
            if (!inq[E[i].to]) {
                Q[++ r] = E[i].to;
                inq[E[i].to] = 1;
            }
        }
    return d[T] != INF;
}
void mcf() {
    int x = INF;
    for(int i = from[T]; i; i = from[E[i].from])
        x = min(x, E[i].v);
    for(int i = from[T]; i; i = from[E[i].from]) {
        E[i].v -= x;
        E[i ^ 1].v += x;
    }
    cost += x * d[T];
    flow += x;
}
```

2.4 平面图转对偶图

```
//bzoj 2965
#include<cstdio>
```

```

#include<cstring>
#include<iostream>
#include<algorithm>
#include<vector>
#include<cmath>
#define N 5010
#define M 500010
#define INF 0x3f3f3f3f
#define LL long long
using namespace std;
int p,n,m;

namespace Max_Flow{
    struct Edge{
        int x,y,v;
        Edge(){}
        Edge(int a,int b,int c){
            x=a;
            y=b;
            v=c;
        }
    }e[M];
    int cnt;

    int S,T;

    int head[N],next[M],edge[M],f[M],tot=1;
    void connect(int a,int b,int c){
        ++tot;next[tot]=head[a];head[a]=tot;edge[tot]=b;f[tot]=c;
        ++tot;next[tot]=head[b];head[b]=tot;edge[tot]=a;f[tot]=0;
    }

    void init(){
        tot=1;
        memset(head,0,sizeof(head));
    }

    void rebuild(){
        for(int i=1;i<=cnt;i++){
            connect(e[i].x,e[i].y,e[i].v);
        }
    }
}

```



```

int l,r,q[N],d[N];
bool bfs(){
    l=r=0;
    memset(d,0,sizeof(d));
    d[S]=1;
    q[++r]=S;
    while(l!=r){
        int u=q[++l];
        for(int i=head[u];i;i=next[i]){
            int v=edge[i];
            if(f[i] && !d[v]){
                d[v]=d[u]+1;
                if(v==T) return true;
                q[++r]=v;
            }
        }
    }
    return false;
}

int dinic(int u,int flow){
    if(u==T) return flow;

    int tmp=flow,k;
    for(int i=head[u];i;i=next[i]){
        int v=edge[i];
        if(tmp && f[i] && d[v]==d[u]+1){
            k=dinic(v,min(f[i],tmp));
            if(!k) d[v]=0;
            f[i]-=k;
            f[i^1]+=k;
            tmp-=k;
        }
    }
    return flow-tmp;
}

}

namespace Graph{
    int _index;

```

```

struct Point{
    double x,y;
    Point (){}
    Point (double a,double b){
        x=a;
        y=b;
    }
    friend Point operator + (const Point &a,const Point &b){
        return Point (a.x+b.x,a.y+b.y);
    }
    friend Point operator - (const Point &a,const Point &b){
        return Point (a.x-b.x,a.y-b.y);
    }
    friend double operator * (const Point &a,const Point &b){
        return a.x*b.y-a.y*b.x;
    }
    friend double operator ^ (const Point &a,const Point &b){
        return a.x*b.x+a.y*b.y;
    }

    inline void read(){
        scanf("%lf%lf",&x,&y);
    }
}_point[30],point[N];

struct Line{
    Point s,t;
    Line(){}
    Line(const Point &a,const Point &b){
        s=a;
        t=b;
    }
    bool online(const Point &b)const{
        return ((b-s)*(t-s)==0 && (b.x-s.x)*(b.x-t.x)<=0 && (b.y-s.y)*(b.y-t.y)<=0);
    }
};

struct Link{
    int to,v,belong;
    double alpha;
    Link *another;
    Link(){}

```

```

Link(int a,int b,double c){
    another=0x0;
    to=a;
    v=b;
    belong=0;
    alpha=c;
}
};

bool cmp(Link *a,Link *b){
    return a->alpha<b->alpha;
}

vector<Link*> a[N];
vector<Point> polygon;

void add(int x,int y,int z){
    Point p=point[y]-point[x];
    Link *temp1=new Link(y,z,atan2(p.y,p.x));
    Link *temp2=new Link(x,z,atan2(-p.y,-p.x));
    temp1->another=temp2;
    temp2->another=temp1;
    a[x].push_back(temp1);
    a[y].push_back(temp2);
}

void dfs(Link *p1){
    Link *p2=p1->another;
    int y=p1->to;
    int x=p2->to;

    p1->belong=_index;
    polygon.push_back(point[x]);

    vector<Link*>::iterator t;
    t=upper_bound(a[y].begin(),a[y].end(),p2,cmp);
    if(t==a[y].end()) t=a[y].begin();

    if((*t)->belong) return;
    dfs(*t);
}

```

```

double Get_Area(){
    int n=polygon.size();
    double s=polygon[0].y*(polygon[n-1].x-polygon[1].x)+polygon[n-1].y*(polygon[n-2].x-polygon[
    for(int i=1;i<n-1;i++){
        s+=polygon[i].y*(polygon[i-1].x-polygon[i+1].x);
    }
    return s/2;
}

bool Cross(Line l1,Line l2){
    return(
        (max(l1.s.x,l1.t.x)>=min(l2.s.x,l2.t.x))&&
        (min(l1.s.x,l1.t.x)<=max(l2.s.x,l2.t.x))&&
        (max(l1.s.y,l1.t.y)>=min(l2.s.y,l2.t.y))&&
        (min(l1.s.y,l1.t.y)<=max(l2.s.y,l2.t.y))&&
        (((l1.s-l2.t)*(l2.s-l2.t))*((l1.t-l2.t)*(l2.s-l2.t))<=0)&&
        (((l2.s-l1.t)*(l1.s-l1.t))*((l2.t-l1.t)*(l1.s-l1.t))<=0)&&
        !l1.online(l2.s) && !l1.online(l2.t) && !l2.online(l1.s) && !l2.online(l1.t)
    );
}

bool In_Polygon(const Point &p){
    Line l1=Line(p,p);
    l1.t.x=1e10;

    int intersection=0,n=polygon.size();
    for(int i=0;i<n;i++){
        Line l2=Line(polygon[i],polygon[(i+1)%n]);
        if(l2.online(p)) return false;
        if(Cross(l1,l2)||
            (l1.online(polygon[(i+1)%n])&&
             (!l1.online(polygon[(i+2)%n])&&
              ((polygon[i]-p)*(polygon[(i+1)%n]-p))*((polygon[(i+2)%n]-p)*(polygon[(i+1)%n]-p))<0)||
             (l1.online(polygon[(i+2)%n])&&
              ((polygon[i]-p)*(polygon[(i+1)%n]-p))*((polygon[(i+3)%n]-p)*(polygon[(i+1)%n]-p))<0)
            )
        ) intersection++;
    }
    if(intersection&1) return true;
    return false;
}
}

```

```

int ans[N],pos[30];
int main(){
    using namespace Graph;
    using namespace Max_Flow;

    int x,y,z;
    scanf("%d%d%d",&p,&n,&m);
    for(int i=1;i<=p;i++){
        _point[i].read();
    }
    for(int i=1;i<=n;i++){
        point[i].read();
    }

    for(int i=1;i<=m;i++){
        scanf("%d%d%d",&x,&y,&z);
        add(x,y,z);
    }

    for(int i=1;i<=n;i++){
        sort(a[i].begin(),a[i].end(),cmp);
    }

    T=0;
    for(int i=1;i<=n;i++){
        vector<Link*>::iterator t;
        for(t=a[i].begin();t!=a[i].end();t++){
            if(!(*t)->belong){
                polygon.clear();

                ++_index;
                dfs(*t);

                if(Get_Area(>0){
                    e[++cnt]=Edge(_index,T,INF);
                    continue;
                }

                for(int j=1;j<=p;j++){
                    if(In_Polygon(_point[j])){
                        pos[j]=_index;

```

```

        }
    }
}
}
for(int i=1;i<=n;i++){
    vector<Link*>::iterator t;
    for(t=a[i].begin();t!=a[i].end();t++){
        e[++cnt]=Edge((*t)->belong,(*t)->another->belong,(*t)->v);
    }
}

S=_index+1;

memset(ans,0x3f,sizeof(ans));

for(int i=1;i<(1<<p);i++){
    int digit=0;
    init();
    rebuild();
    for(int j=0;j<p;j++){
        if(i & (1<<j)){
            digit++;
            connect(S,pos[j+1],INF);
        }
    }
    int maxflow=0;
    while(bfs()) maxflow+=dinic(S,INF);
    ans[digit]=min(ans[digit],maxflow);
}

for(int i=1;i<=p;i++) printf("%d\n",ans[i]);
return 0;
}

```

3 数据结构

3.1 K-D 树

```

const int N = 100010;
int D, p[N], tot, root;
struct node {
    int d[2], mn[2], mx[2], l, r, D, size, sum, v;

```

```

    int& operator [] (int x) {return d[x];}
}t[N], now;

inline bool cmp(int x, int y) {return t[x][D] < t[y][D];}
#define L t[o].l
#define R t[o].r
#define mid ((l + r) >> 1)
inline void Push_up(int o) {
    for(int i = 0; i < 2; i++) {
        t[o].mn[i] = min(t[o].mn[i], min(t[L].mn[i], t[R].mn[i]));
        t[o].mx[i] = max(t[o].mx[i], max(t[L].mx[i], t[R].mx[i]));
    }
    t[o].sum = t[L].sum + t[R].sum + t[o].v;
    t[o].size = t[L].size + t[R].size + 1;
}
inline int build(int l, int r, int dir) {
    D = dir;
    nth_element(p + 1, p + mid, p + r + 1, cmp);
    int o = p[mid];
    for(int i = 0; i < 2; i++) t[o].mn[i] = t[o].mx[i] = t[o][i];
    t[o].sum = t[o].v;
    L = l < mid ? build(l, mid - 1, dir ^ 1) : 0;
    R = mid < r ? build(mid + 1, r, dir ^ 1) : 0;
    Push_up(o);
    return o;
}
inline void dfs(int o){
    if (!o) return;
    dfs(L);
    p[++cnt] = o;
    dfs(R);
}
inline void rebuild(int &o) {
    cnt = 0;
    dfs(o);
    o = bulid(1, cnt, t[o].D);
}
inline void Insert(int &o, int dir) {
    if (!o) {
        o = ++tot; t[o] = now;
        for(int i = 0; i < 2; i++) t[o].mn[i] = t[o].mx[i] = t[o][i];
        t[o].D = dir;
    }
}

```

```

    t[o].size = 1;
    t[o].sum = t[o].v;
    return;
}
if (now[dir] < t[o][dir]) {
    Insert(L, dir ^ 1);
    Push_up(o);
    if ((double)t[L].size > (double)t[o].size * 0.7) rebuild(o);
} else {
    Insert(R, dir ^ 1);
    Push_up(o);
    if ((double)t[R].size > (double)t[o].size * 0.7) rebuild(o);
}
}
inline double getans(int o, int k) {
    if (!o) return INF;
}
inline double calc(int o, double k) {
    if (!o) return INF;
    double ans = ....;
}
inline void query(int o, double k) {
    if (!o) return 0;
    double dl = calc(L, k), dr = calc(R, k), d0 = getans(o, k);
    ans = max(ans, d0);
    if (dl < dr) {
        if (dr > ans && R) query(R, k);
        if (dl > ans && L) query(L, k);
    } else {
        if (dl > ans && L) query(L, k);
        if (dr > ans && R) query(R, k);
    }
}
}

```

3.2 可持久化 Trie

```

int t[M][2], rt[N], id[M], tot;

inline void Ins(int pre, int x, int k) {
    int now = rt[k] = ++tot; id[tot] = k;
    for(int i = 30; i >= 0; i--) {
        int j = (x >> i) & 1;
        t[now][j ^ 1] = t[pre][j ^ 1];
    }
}

```



```

    t[now][j] = ++tot; id[tot] = k;
    now = tot;
    pre = t[pre][j];
}
}
inline int query(int l, int r, int x) {
    int ans = 0, now = rt[r];
    for(int i = 30; i >= 0; i --) {
        int j = ((x >> i) & 1) ^ 1;
        if (id[t[now][j]] >= 1) ans |= 1 << i; else j ^= 1;
        now = t[now][j];
    }
    return ans;
}

```

3.3 Link-Cut-Tree

```

struct LCT {
    int c[N][2], fa[N], v[N], mx[N];
    bool rev[N];
    int st[N], top;
#define L c[x][0]
#define R c[x][1]
    void Push_up(int x) {
        mx[x] = x;
        if (v[mx[L]] > v[mx[x]]) mx[x] = mx[L];
        if (v[mx[R]] > v[mx[x]]) mx[x] = mx[R];
    }
    void Push_down(int x) {
        if (rev[x]) rev[x] = 0, rev[L] ^= 1, rev[R] ^= 1, swap(L, R);
    }
    bool not_root(int x) {
        return c[fa[x]][0] == x || c[fa[x]][1] == x;
    }
    void Rotate(int x) {
        int y = fa[x], z = fa[y], l = (c[y][1] == x), r = l ^ 1;
        if (not_root(y)) c[z][c[z][1] == y] = x;
        fa[x] = z; fa[y] = x; fa[c[x][r]] = y;
        c[y][l] = c[x][r]; c[x][r] = y;
        Push_up(y);
    }
    void preview(int x) {
        top = 0; st[++top] = x;
    }
}

```

```

    for(;not_root(x); x = fa[x]) st[++top] = fa[x];
    for(int i = top; i; i --) Push_down(st[i]);
}

void splay(int x, int y = 0) {
    for(peek(x); not_root(x); Rotate(x))
        if (not_root(y = fa[x]))
            Rotate(c[y][1] == x ^ c[fa[y][1]] == y ? x : y);
    Push_up(x);
}

void access(int x, int y = 0) {
    for(;x;splay(x), c[x][1] = y, y = x, x = fa[x]);
}

void makeroot(int x) {
    access(x); splay(x); rev[x] ^= 1;
}

void link(int x, int y) {
    makeroot(x); fa[x] = y;
}

void cut(int x, int y) {
    makeroot(x); access(y); splay(y);
    if (c[y][0] == x) c[y][0] = fa[x] = 0;
}

int query(int x, int y) {
    makeroot(x); access(y); splay(y);
    return mx[y];
}
};

```

3.4 可持久化线段树

```

struct node {
    int cnt, l, r;
}t[N * 30];

void update(int &o, int l, int r, int pos) {
    t[++ cnt] = t[o], o = cnt, ++ t[o].cnt;
    if (l == r) return;
    if (pos <= mid) update(t[o].l, l, mid, pos);
    else update(t[o].r, mid + 1, r, pos);
}

int query(int i, int j, int rank) {
    i = root[i], j = root[j];

```

```

int l = 1, r = n;
while(l != r) {
    if (t[t[j].l].cnt - t[t[i].l].cnt >= rank)
        r = mid, i = t[i].l, j = t[j].l;
    else{
        rank -= t[t[j].l].cnt - t[t[i].l].cnt;
        l = mid + 1, i = t[i].r, j = t[j].r;
    }
}
return l;
}

```

3.5 左偏树

```

struct node{
    int l, r, pnt, v, dist;
}a[N * 17 + (N << 1)];
struct Heap{
    int root;
    int size;
    int merge(int x, int y){
        if(!(x && y)) return x ^ y;
        if(a[x].v < a[y].v) swap(x, y);
        a[x].r = merge(a[x].r, y);
        if(a[x].r) a[a[x].r].pnt = x;
        if(a[a[x].l].dist < a[a[x].r].dist) swap(a[x].l, a[x].r);
        a[x].dist = a[a[x].r].dist + 1;
        return x;
    }
    inline int newnode(int x, int p){
        int res = id(x, p);
        a[res].pnt = a[res].l = a[res].r = 0;
        a[res].dist = 1;
        a[res].v = d[x][p];
        return res;
    }
    inline void push(int x, int p){
        size++;
        int t = newnode(x, p);
        root = merge(root, t);
    }
    inline void pop(int x, int p){
        size--;
    }
}

```

```

    int res = id(x, p);
    int fa = a[res].pnt;
    int t = merge(a[res].l, a[res].r);
    a[t].pnt = fa;
    if(!fa) root = t;
    if(a[fa].l == res) a[fa].l = t;
    if(a[fa].r == res) {a[fa].r = t; a[fa].dist = a[t].dist + 1;}
}

inline int top(){return a[root].v;}

inline int newnode2(int x, int v){
    int res = h(x);
    a[res].dist = 1;
    a[res].l = a[res].r = a[res].pnt = 0;
    a[res].v = v;
    return res;
}

inline void push(int x){
    size++;
    root = merge(root, x);
}

inline void pop(int x){
    size--;
    int fa = a[x].pnt;
    int t = merge(a[x].l, a[x].r);
    a[t].pnt = fa;
    if(!fa) root = t;
    if(a[fa].l == x) a[fa].l = t;
    else if(a[fa].r == x) a[fa].r = t;
}

inline void change(int x, int v){
    pop(h(x));
    push(newnode2(x, v));
}

}q[N << 1], Q;

```

4 字符串

4.1 AC 自动机

```

int cnt = 1;
struct Trie {
    int ch[26], cnt, fail;
    bool sign;

```

```

}T[N];
inline int id(char c) {return c - 'A';}
void Ins(char *s) {
    int x = 1, y, l = strlen(s);
    for(int i = 0; i < l; i++) {
        y = id(s[i]);
        if (!T[x].ch[y]) T[x].ch[y] = ++cnt;
        x = T[x].ch[y];
    }
    T[x].sign = 1;
}
int Q[N];
void make_fail() {
    int l = 0, r = -1;
    Q[++r] = 1;
    while(l <= r) {
        int x = Q[l++], y, j;
        for(int i = 0; i < 26; i++) {
            j = T[x].fail;
            T[x].sign |= T[j].sign;
            while(j && !T[x].ch[i]) j = T[j].fail;
            if (T[x].ch[i]) {
                y = T[x].ch[i];
                T[y].fail = j ? T[j].ch[i] : 1;
                Q[++r] = y;
            } else T[x].ch[i] = j ? T[j].ch[i] : 1;
        }
    }
}
}

```

4.2 后缀数组

```

int n, m, sa[N], c[N], wa[N], wb[N], rank[N], height[N];
inline bool cmp(int *r, int a, int b, int l) {
    return r[a] == r[b] && r[a + l] == r[b + l];
}

void DA(char *s, int *sa, int n, int m) {
    int *x = wa, *y = wb;
    for(int i = 0; i < m; i++) c[i] = 0;
    for(int i = 0; i < n; i++) c[x[i] = s[i]]++;
    for(int i = 1; i < m; i++) c[i] += c[i - 1];
    for(int i = n - 1; i >= 0; i--) sa[--c[x[i]]] = i;
}

```

```

for(int j = 1; p = 0; p < n; j <= 1, m = p) {
    for(int i = n - j; i < n; i++) y[p++] = i;
    for(int i = 0; i < n; i++) if (sa[i] >= j) y[p++] = sa[i] - j;

    for(int i = 0; i < m; i++) c[i] = 0;
    for(int i = 0; i < n; i++) c[x[y[i]]]++;
    for(int i = 1; i < m; i++) c[i] += c[i - 1];
    for(int i = n - 1; i >= 0; i--) sa[--c[x[y[i]]]] = y[i];
    swap(x, y); p = 1; x[sa[0]] = 0;
    for(int i = 1; i < n; i++) x[sa[i]] = cmp(y, sa[i - 1], sa[i], j) ? p - 1 : p++;
}
}

void calcheight(char *s, int *sa, int n) {
    int k = 0;
    for(int i = 1; i <= n; i++) rank[sa[i]] = i;
    for(int i = 0; i < n; i++) {
        if (k) k--;
        int j = sa[rank[i] - 1];
        while(s[i + k] == s[j + k]) k++;
        height[rank[i]] = k;
    }
}

int main() {
    n = strlen(s); m = size_of_character_set;
    DA(s, sa, n + 1, 39);
    calcheight(s, sa, n);
}

```

4.3 后缀自动机

```

struct Suffix_Automation{
    struct node{
        int son[26];
        int pre,cnt,len,right;
    }a[N<<2];
    int tail,tot;

    void insert(int key){
        int np,nq,p,q;
        if(a[tail].son[key]){
            if(a[q=a[tail].son[key]].len==a[tail].len+1) tail=q;

```

```

else{
    a[nq=++tot]=a[q];
    a[nq].right=0;
    a[nq].len=a[tail].len+1;
    a[q].pre=nq;
    for(p=tail;a[p].son[key]==q;p=a[p].pre) a[p].son[key]=nq;
    tail=nq;
}
}
else{
    a[np=++tot].len=a[tail].len+1;
    for(p=tail;p && !a[p].son[key];p=a[p].pre) a[p].son[key]=np;
    if(!a[p].son[key]){
        a[p].son[key]=np;
        a[np].pre=p;
    }
    else{
        if(a[q=a[p].son[key]].len==a[p].len+1) a[np].pre=q;
        else{
            a[nq=++tot]=a[q];
            a[nq].len=a[p].len+1;
            a[nq].right=0;
            a[q].pre=a[np].pre=nq;
            for(;a[p].son[key]==q;p=a[p].pre) a[p].son[key]=nq;
        }
    }
    tail=np;
}
a[tail].right++;
}

void bfs(){
    static int cnt[N],l,r,q[N];
    for(int i=1;i<=tot;i++){
        cnt[a[i].pre]++;
        a[i].cnt=1;
    }
    l=r=0;
    for(int i=1;i<=tot;i++){
        if(!cnt[i]) q[++r]=i;
    }
}

```

```

while(l!=r){
    int u=q[++l];
    int p=a[u].pre;
    a[p].right+=a[u].right;
    a[p].cnt+=a[u].cnt+a[u].right;
    if(!--cnt[p]) q[++r]=p;
}
}
}sam;

```

4.4 回文自动机

```

int ch[N][26], fail[N], len[N], tot, cnt[N];
void ready() {
    len[0] = 0; len[1] = -1;
    fail[0] = 1; fail[1] = -1;
}
void Insert(char *s, int *cnt) {
    int now = 1, l = strlen(s), x, y, tmp;
    for(int i = 0; i < l; ++ i) {
        x = s[i] - 'a';
        while(s[i] != s[i - len[now] - 1]) now = fail[now];
        if (!ch[now][x]) {
            ch[now][x] = ++ tot;
            len[tot] = len[now] + 2;
        }
        y = ch[now][x];
        tmp = fail[now];
        if (tmp == -1) fail[y] = 0;
        else {
            while(s[i] != s[i - len[tmp] - 1]) tmp = fail[tmp];
            fail[y] = ch[tmp][x];
        }
        now = y;
        cnt[now] ++;
    }
}

```

4.5 Manacher 算法

```

//求串 S 中最长的回文子串的长度
char s[N];
int a[N];
int manacher(char *s) {

```



```

memset(p, 0, sizeof p);
int n = strlen(s);
for(int i = 1; i <= n; i++) a[i << 1] = b[i - 1];
n = n << 1 | 1;
int id = 0, mx = 0, ans = 0;
for(int i = 1; i <= n; i++) {
    if (mx > i) p[i] = min(p[2 * id - i], mx - i);
    while(i - p[i] - 1 > 0 && i + p[i] + 1 <= n
        && a[i - p[i] - 1] == a[i + p[i] + 1]) p[i] ++;
    if (p[i] + i > mx) mx = p[i] + i, id = i;
    if (p[i] > ans) ans = p[i];
}
return ans;
}

```

5 博弈

5.1 二分图博弈

```

#include<cstdio>
#include<cstring>
#include<iostream>
#include<algorithm>
#include<vector>
#define M 101
#define N 101010
#define INF 0x3f3f3f3f
#define LL long long
using namespace std;
int n, m, n1, n2, pos[M][M];
bool map[M][M];
int dx[] = {1, 0, -1, 0};
int dy[] = {0, 1, 0, -1};
vector<int> edge[N];
int match[N];
void dfs(int u, bool ans[]){
    ans[u] = true;
    for(int i = 0; i < edge[u].size(); i++){
        if(match[edge[u][i]] != -1 && !ans[match[edge[u][i]]]){
            dfs(match[edge[u][i]], ans);
        }
    }
}
}

```

```

bool mark[N], vis[N];
bool find(int u){
    for(int i = 0; i < edge[u].size(); i++){
        int v = edge[u][i];
        if(!vis[v]){
            vis[v] = 1;
            if(match[v] == -1 || find(match[v])){
                match[v] = u; return true;
            }
        }
    }
    return false;
}

int hungary(int cnt){
    int res = 0;
    memset(match, -1, sizeof(match));
    memset(mark, 0, sizeof(mark));
    for(int i = 1; i <= cnt; i++){
        memset(vis, 0, sizeof(vis));
        if(!find(i)){
            mark[i] = 1;
            res++;
        }
    }
    return res;
}

bool ans[2][N];
int main(){
    char c;
    scanf("%d%d", &n, &m);
    for(int i = 1; i <= n; i++){
        for(int j = 1; j <= m; j++){
            scanf(" %c", &c);
            if(c == '.'){
                map[i][j] = 1;
                if((i & 1) ^ (j & 1) == 0){
                    ++ n1;
                    pos[i][j] = n1;
                }
                else{
                    ++ n2;
                    pos[i][j] = n2;
                }
            }
        }
    }
}

```

```

        }
    }
    else map[i][j] = 0;
}
}
for(int i = 1; i <= n; i++){
    for(int j = (i & 1 ? 1 : 2); j <= m; j += 2){
        if(map[i][j]){
            for(int k = 0; k < 4; k++){
                int xx = i + dx[k], yy = j + dy[k];
                if(0 < xx && xx <= n && 0 < yy && yy <= m && map[xx][yy]){
                    edge[pos[i][j]].push_back(pos[xx][yy]);
                }
            }
        }
    }
}
int tmp = hungary(n1);
if(tmp == n1 && tmp == n2){
    puts("LOSE");
    return 0;
}
for(int i = 1; i <= n1; i++){
    if(mark[i]) dfs(i, ans[0]);
}
for(int i = 1; i <= n1; i++) edge[i].clear();
for(int i = 1; i <= n; i++){
    for(int j = (i & 1 ? 2 : 1); j <= m; j += 2){
        if(map[i][j]){
            for(int k = 0; k < 4; k++){
                int xx = i + dx[k], yy = j + dy[k];
                if(0 < xx && xx <= n && 0 < yy && yy <= m && map[xx][yy]){
                    edge[pos[i][j]].push_back(pos[xx][yy]);
                }
            }
        }
    }
}
hungary(n2);
for(int i = 1; i <= n2; i++){
    if(mark[i]) dfs(i, ans[1]);
}

```

```

puts("WIN");
for(int i = 1; i <= n; i++){
    for(int j = 1; j <= m; j++){
        if(map[i][j] && ans[(i & 1)^(j & 1)][pos[i][j]]){
            printf("%d %d\n", i, j);
        }
    }
}
return 0;
}
|

```