# ACM-ICPC TEMPLATE


**SW2000**

**Last build at June 19, 2021**

cy1999txdy

# Contents

# 1 杂项

## 1.1 优先级

[] () . ->
- (类型) ++ – * & ! ~ sizeof
/ * %
+ -
< < > >
> >= < <=
== !=
&
^
|
&&
||
?:
= /= *= %= += -= «= »= &= ≙ |=
,

## 1.2 IO 优化

```
//适用于非负整数
template<class T>
void read(T&ret) {
    char c;
    ret=0;
    while((c=getchar())<'0'||c>'9');
    while(c>='0'&&c<='9')ret=ret*10+(c-'0'),c=getchar();
}
//适用于整数
template<class T>
bool read(T&ret) {
    char c;
    int sgn;
    if(c=getchar(),c==EOF)return 0;//EOF
    while(c!='-'&&(c<'0'||c>'9'))c=getchar();
    sgn=(c=='-')?-1:1;
    ret=(c=='-')?0:(c-'0');
    while(c=getchar(),c>='0'&&c<='9')ret=ret*10+(c-'0');
    ret*=sgn;
    return 1;
}
//适用于整数,(int,long long,float,double)
template<class T>
bool read(T&ret) {
    char c;
    int sgn;
    T bit=0.1;
    if(c=getchar(),c==EOF)return 0;
    while(c!='-'&&c!='.'&&(c<'0'||c>'9'))c=getchar();
    sgn=(c=='-')?-1:1;
    ret=(c=='-')?0:(c-'0');
    while(c=getchar(),c>='0'&&c<='9')ret=ret*10+(c-'0');
    if(c==' '||c=='\n') {
        ret*=sgn;
        return 1;
    }
    while(c=getchar(),c>='0'&&c<='9')ret+=(c-'0')*bit,bit/=10;
```

```
    ret*=sgn;
    return 1;
}
//输出外挂
void out(int x) {
    if(x>9)out(x/10);
    putchar(x%10+'0');
}
cout<<fixed<<setprecision(2)<<3.0<<endl;
cout<<setfill('0')<<setw(2)<<3<<endl;
```

## 1.3 STL

### 1.3.1 string

```
getline(cin,str);//带空格输入字符串
str.c_str();//把string类型转换为char*
str.substr(p0,len);//其中len可以不填，默认取到末尾。
str.erase(p0,len);//其中len可以不填，默认取到末尾。
str1.insert(p0,str2);//从p0处开始插入str2
str1.replace(p0,len0,str2);//将p0开始的len0个字符换成str2
str1.find(str2,pos);//从pos开始查找str2第一次出现的(int)下标，失败返回(int)(-1)
str1.rfind(str2,pos);//反向查找

to_string(n)//把n转化为string类型
stoi(s)//把string转换成int，超过int范围会RE
atoi(s)//把char*转换成int

//流输入
getline(cin, str);
istringstream iss;
iss.str(str);
while(iss >> x)dosomething();
```

### 1.3.2 vector

```
v.front();// 传回第一个数据。
v.back();// 传回最后一个数据，不检查这个数据是否存在。
v.erase(pos);// 删除pos位置的数据，传回下一个数据的位置。
v.erase(beg,end);//删除[beg,end)区间的数据，传回下一个数据的位置。
v.insert(pos,elem);// 在pos位置插入一个elem拷贝，传回新数据位置。
v.insert(pos,n,elem);// 在pos位置插入n个elem数据。无返回值。
v.insert(pos,beg,end);// 在pos位置插入在[beg,end)区间的数据。无返回值。
```

### 1.3.3 优先队列

```
priority_queue<int>q;//top是最大值
priority_queue<int,vector<int>,greater<int> >q;//top是最小值
bool operator<(Node a,Node b){return a.x<b.x;}//放到结构体里面时需要加const
struct cmp{bool operator()(int a,int b){return a<b;}};
priority_queue<int,vector<int>,cmp_key>q;
```

### 1.3.4 rope

块状链表，方便插入删除，可以作为可持久化数组，以下所有操作包括赋值，都是 $O(sqrtn)$ 复杂度（包括赋值）

```cpp
#include <ext/rope>
using namespace __gnu_cxx;
rope<int>r;
r[x];//访问第x个元素，下标从0开始
r.push_back(x);//在末尾添加x
r.substr(p, len); //提取从pos开始的x个的rope
r.insert(p, x); //在r[p]后面插入x，x可以是块状链表
r.erase(p, x); //从r[p]开始删除x个
r.replace(p, (len), x); //把第p个位置(到len个)用x代替，x可以是块状链表
```

### 1.3.5 bitset

```cpp
bitset<5>b;//坐标从后往前计数，高位在前
bitset<5>b(13);
bitset<5>b(”1101”);
b.count();//count函数用来求bitset中1的位数，一共3
b.size();//size函数用来求bitset的大小，一共5
b.any();//any函数检查bitset中是否有 1
b.none();//none函数检查bitset中是否没有 1
b.all();//all函数检查bitset中是全部为 1
foo.flip();//flip函数不指定参数时，将bitset每一位全部取反
foo.set();//set函数不指定参数时，将bitset的每一位全部置为 1
foo.reset();//reset函数不传参数时将bitset的每一位全部置为 0
string s = foo.to_string();//将bitset转换成string类型
unsigned long a = foo.to_ulong();//将bitset转换成unsigned long类型
unsigned long long b = foo.to_ullong();//将bitset转换成unsigned long long类型
```

### 1.3.6 随机

```cpp
mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());
double rnd(double l, double r){return uniform_real_distribution<double>(l,r)(rng);}
int rnd(int l,int r){return uniform_int_distribution<int>(l,r)(rng);}
```

### 1.3.7 Hash 表

```cpp
struct Hash_Table{
    static const int n=10000007;
    int cnt,head[n];
    struct A{int val,idx,ne;}e[n];//下标为[1,cnt]
    void init(){while(cnt)head[e[cnt--].idx%n]=0;};
    void add(int idx,int v){
        int x=idx%n;
        for(int i=head[x];i;i=e[i].ne)if(e[i].idx==idx){e[i].val+=v;return;}
        e[++cnt]={v,idx,head[x]};head[x]=cnt;
        return;
    }
    int get(int idx){
        for(int i=head[idx%n];i;i=e[i].ne)if(e[i].idx==idx)return e[i].val;
        return 0;
    }
};
```

### 1.3.8 builtin

```
int __builtin_ffs (unsigned int x)
返回x的最后一位1的是从后向前第几位，比如7368（1110011001000）返回4。
int __builtin_ctz (unsigned int x)
返回后面的0的个数，和__builtin_clz相对。
int __builtin_popcount (unsigned int x)
返回二进制表示中1的个数。

此外，这些函数都有相应的usigned long和usigned long long版本，只需要在函数名后面
    加上l或ll就可以了，比如int __builtin_clzll。
```

## 1.4 高精度

```
struct Num{
    const static int MOD=10000,DLEN=4;
    vector<int>a;
    Num(int v=0){
        a.clear();
        do{a.push_back(v%MOD);v/=MOD;}while(v);
    }
    Num(string s){//不可为空
        a.clear();
        for(int i = s.size(); i; i -= DLEN){
            a.push_back(0);
            for(int j = max(0, i - DLEN); j < i; j++)
                a.back() = a.back() * 10 + s[j] - '0';
        }
    }
    bool operator<(Num b){
        if(a.size() < b.a.size())return true;
        else if(a.size() > b.a.size())return false;
        else{
            int ln = a.size() - 1;
            while(a[ln] == b.a[ln] && ln >= 0)ln--;
            if(ln >= 0 && a[ln] < b.a[ln])return true;
            else return false;
        }
    }
    Num operator+(Num &b){
        Num ret;
        ret.a.assign(max(a.size(), b.a.size())+1,0);
        for(auto &i:ret.a)i=0;
        for(int i = 0; i < ret.a.size()-1; i++){
            ret.a[i]+=((i<a.size())?a[i]:0)+((i<b.a.size())?b.a[i]:0);
            ret.a[i+1]+=ret.a[i]/MOD;
            ret.a[i]%=MOD;
        }
        if(ret.a.back()==0)ret.a.pop_back();
        return ret;
    }
    Num operator-(Num b){//不支持负数

        Num ret = *this;
        for(int i = 0; i < b.a.size(); i++){
            ret.a[i] -= b.a[i];
            if(ret.a[i] < 0){
                if(i+1==a.size())assert(1);
                ret.a[i+1]--;
                ret.a[i]+=MOD;
            }
```

```
        }
        while(ret.a.back() == 0 && ret.a.size() > 1)ret.a.pop_back();
        return ret;
    }
    Num operator*(Num b){//la*lb复杂度
        Num ret;
        ret.a.assign(a.size()+b.a.size(),0);
        for(int i = 0; i < a.size(); i++){
            int up = 0;
            for(int j = 0; j < b.a.size(); j++){
                int tmp = a[i] * b.a[j] + ret.a[i + j] + up;
                ret.a[i + j] = tmp % MOD;
                up = tmp / MOD;
            }
            if(up)ret.a[i + b.a.size()] += up;
        }
        while(ret.a.back() == 0 && ret.a.size() > 1)ret.a.pop_back();
        return ret;
    }
    Num operator/(int b){
        Num ret=*this;
        int down = 0;
        for(int i = a.size() - 1; i >= 0; i--){
            ret.a[i] = (a[i] + down * MOD) / b;
            down = a[i] + down * MOD - ret.a[i] * b;
        }
        while(ret.a.back() == 0 && ret.a.size() > 1)ret.a.pop_back();
        return ret;
    }
    int operator%(int b){
        int ret = 0;
        for(int i = a.size()-1; i >= 0; i--)
            ret = ((ret * MOD) % b + a[i]) % b;
        return ret;
    }
    void o(){
        cout << a.back();
        for(int i=(int)a.size()-2;i>=0;i--)
            cout << setw(DLEN) << setfill('0') << a[i];
    }
};
```

# 2 数据结构

## 2.1 主席树

区间第 k 大

本模板是离散后对权值建树

```
#include<bits/stdc++.h>
#define mid (l+r>>1)
using namespace std;
const int N=2e5+10;
struct TR{
    int sum,lo,ro;
}tr[N<<5];
int tr_cnt;//之后需要初始化=0
int n,m,q,arr[N],brr[N],rt[N];//m是权值的数量
void build(int &o,int l=1,int r=m){
    o=++tr_cnt;
    //tr[o].sum=0;
```

```
    if(l==r)return;
    build(tr[o].lo,l,mid);
    build(tr[o].ro,mid+1,r);
}
void update(int p,int v,int pre,int &o,int l=1,int r=m){
    o=++tr_cnt;
    tr[o]=tr[pre];
    tr[o].sum+=v;//都是+1
    if(l==r)return;
    if(p<=mid)update(p,v,tr[pre].lo,tr[o].lo,l,mid);
    else update(p,v,tr[pre].ro,tr[o].ro,mid+1,r);
}
//u和v是两个线段树的根，相减后的线段树求第k个的下标位置
int query(int k,int u,int v,int l=1,int r=m){
    if(l==r)return l;
    int tmp=tr[tr[v].lo].sum-tr[tr[u].lo].sum;
    if(tmp>=k)return query(k,tr[u].lo,tr[v].lo,l,mid);
    else return query(k-tmp,tr[u].ro,tr[v].ro,mid+1,r);
}
int main(){
    int n,q;scanf("%d%d",&n,&q);
    for(int i=1;i<=n;i++)scanf("%d",arr+i),brr[i]=arr[i];
    sort(brr+1,brr+1+n);
    m=unique(brr+1,brr+1+n)-brr-1;
    build(rt[0]);
    for(int i=1;i<=n;i++){
        int p=lower_bound(brr+1,brr+1+m,arr[i])-brr;
        update(p,1,rt[i-1],rt[i]);
    }
    while(q--){
        int u,v,k;scanf("%d%d%d",&u,&v,&k);
        printf("%d\n",brr[query(k,rt[u-1],rt[v])]);
    }
    return 0;
}
```

## 2.2 RMQ

O(20*n) 复杂度,O(1) 求 1 n 的最大最小值 ma[j][i] 代表 i i+(1«j)-1 的最值

```
struct RMQ{
    static const int N=5e5+10;
    int n,ma[20][N],lg[N];
    void build(int n){
        lg[0]=-1;
        for(int i=1;i<=n;i++){
            lg[i]=lg[i-1]+(i&i-1?0:1);
            ma[0][i]=arr[i].se;
        }
        for(int j=1;j<=lg[n];j++)for(int i=1;i<=n-(1<<j)+1;i++)
            ma[j][i]=max(ma[j-1][i], ma[j-1][i+(1<<j-1)]);
    }
    int query(int l, int r){
        int k=lg[r-l+1];
        return max(ma[k][l], ma[k][r-(1<<k)+1]);
    }
};
```

## 2.3 线段树合并分裂

可重集合并分裂操作

```cpp
#include <bits/stdc++.h>
#define mid (l+r>>1)
using namespace std;
typedef long long ll;
const int N = 2e5 + 10;
int n,m,cnt,seq = 1,bac[N<<5],lo[N<<5],ro[N<<5],rt[N];
ll val[N << 5];
int newnod(){return bac[cnt--];}//取出一个下标
void del(int o){bac[++cnt]=o, lo[o]=ro[o]=val[o]=0;}//回收下标
//单点加操作
void add (int p, int v, int &o, int l=1, int r=n){
    if (!o)o = newnod();
    val[o] += v;
    if (l == r)return;
    if (p <= mid)add(p, v, lo[o], l, mid);
    else add(p, v, ro[o], mid + 1, r);
}
//查询区间和
ll query (int ql, int qr, int o, int l = 1, int r = n){
    if (ql <= l && r <= qr)return val[o];
    ll ret = 0;
    if(ql <= mid)ret = query(ql, qr, lo[o], l, mid);
    if(qr > mid)ret += query(ql, qr, ro[o], mid + 1, r);
    return ret;
}
//查询o处线段树第k小，本题不用离散化
int kth (int k, int o, int l = 1, int r = n){
    if (l == r)return l;
    if (val[lo[o]] >= k)return kth(k, lo[o], l, mid);
    else return kth(k - val[lo[o]], ro[o], mid + 1, r);
}
//合并下标x和y节点处的线段树，返回合并后的下标
int merge (int x, int y){
    if (!x||!y)return x+y;
    val[x]+=val[y];
    lo[x]=merge(lo[x], lo[y]);
    ro[x]=merge(ro[x], ro[y]);
    del(y);
    return x;
}
//把x的前k个保留，剩下的分给y
void split (int x, int &y, ll k){
    if (!x)return;
    y = newnod();
    ll v = val[lo[x]];
    if (k>v)split(ro[x], ro[y], k - v);
    else swap(ro[x], ro[y]);
    if (k<v)split(lo[x], lo[y], k);
    val[y]=val[x]-k,val[x]=k;
    return;
}
int main ()
{
    for(int i = 1; i<(N<<5); i++)bac[++cnt]=i;//把下标存进库中
    cin>>n>>m;
    for(int i = 1,a; i <= n; i++){cin>>a;add(i, a, rt[1]);}
    for(int i = 1,op,p,x,l,r,q,t,k; i <= m; i++){
        cin>>op;
        //将可重集p中大于等于l且小于等于r的值放入一个新的可重集中
        if(op==0){
            cin>>p>>l>>r;
            ll k1=query(1, r, rt[p]), k2=query(l, r, rt[p]);
```

```
            int tmp;
            split(rt[p], rt[++seq], k1-k2);//先把l~n分裂给新集合
            split(rt[seq], tmp, k2);//再把r~n分裂出来给tmp
            rt[p] = merge(rt[p], tmp);//把tmp还给p
        }
        //将可重集 t 中的数放入可重集 p，且清空可重集 t
        else if(op==1){
            cin>>p>>t;
            rt[p]=merge(rt[p], rt[t]);
        }
        //在 p 这个可重集中加入 x 个数字 q
        else if(op==2){
            cin>>p>>x>>q;
            add(q, x, rt[p]);
        }
        //查询可重集 p 中大于等于 x 且小于等于 y 的值的个数。
        else if(op==3){
            cin>>p>>l>>r;
            cout<<query(l, r, rt[p])<<endl;
        }
        //查询在 p 这个可重集中第 k 小的数，不存在时输出 -1
        else if(op==4){
            cin>>p>>k;
            cout<<(val[rt[p]] < k?-1:kth(k, rt[p]))<<endl;
        }
    }
}
```

## 2.4 Splay

本模板是离散后对权值建树

```
#include<bits/stdc++.h>
const int N = 1e5+5;
using namespace std;
int n;
int tot,root;
int w[N],num[N],sz[N],fa[N],son[N][2];
void update(int x) {
    sz[x]=sz[son[x][0]]+sz[son[x][1]]+num[x];
}
//void pushdown(int x) {
//  do something...
//}
void rotate(int x) {//单旋
//    pushdown(fa[x]);pushdown(x);
    int y=fa[x],z=fa[y],t=(son[y][0]==x);
    fa[y]=x;fa[x]=z;
    if(z) son[z][son[z][1]==y]=x;
    son[y][!t]=son[x][t];fa[son[x][t]]=y;
    son[x][t]=y;
    update(y);update(x);
}
void splay(int x,int f) {//双旋
//    pushdown(x);
    while(fa[x]!=f) {
        int y=fa[x],z=fa[y];
        if(z!=f) {
            if(son[y][0]==x^son[z][0]==y)rotate(x);
            else rotate(y);
        }
        rotate(x);
```

```
    }
    if(!f)root=x;
}
//插入val
void insert(int val,int &x=root,int f=0) {
    if(!x) {
        x=++tot;fa[x]=f;
        son[x][0]=son[x][1]=0;
        w[x]=val;sz[x]=num[x]=1;
        splay(x,0);
        return;
    }
    if(val==w[x]) {
        sz[x]++;num[x]++;
        splay(x,0);
        return;
    }
    insert(val,son[x][val>w[x]],x);
    update(x);
}
//得到val的节点下标
int get(int val) {
    int x=root;
//    pushdown(x);
    while(x&&w[x]!=val) {
        x=son[x][val>w[x]];
//    pushdown(x);
    }
    return x;
}
//删除一个大小为w的值
void delet(int w) {
    int x=get(w);
    if(!x)return;
    splay(x,0);
    if(num[x]>1) {
        num[x]--;sz[x]--;
        return;
    }
    if(!son[x][0]||!son[x][1])root=son[x][0]+son[x][1];
    else {
        int y=son[x][1];
        while(son[y][0])y=son[y][0];
        splay(y,x);
        fa[son[x][0]]=y;
        son[y][0]=son[x][0];
        root=y;
        son[x][0]=son[x][1]=0;
    }
    fa[root]=0;
    update(root);
}
//查询val是第几大
int getrank(int val) {
    int x=root,ret=0,last=0;
    while(x) {
        last=x;
        if(val>w[x]) {
            ret+=sz[son[x][0]]+num[x];
            x=son[x][1];
        } else if(val==w[x]) {
            ret+=sz[son[x][0]];
            break;
```

```
94          }
95          else {
96              x=son[x][0];
97          }
98      }
99      if(last)splay(last,0);
100     return ret+1;
101 }
102 //第k大
103 int kth(int k) {
104     int x=root;
105 //    pushdown(x);
106     while(k<=sz[son[x][0]]||k>sz[son[x][0]]+num[x]) {
107         if(k<=sz[son[x][0]])x=son[x][0];
108         else k-=sz[son[x][0]]+num[x],x=son[x][1];
109 //    pushdown(x);
110     }
111     return w[x];
112 }
113 //第一个小于val的值
114 int getpre(int val) {
115     int x=root,ret=0,last=0;
116     while(x) {
117         last=x;
118         if(val>w[x]) {
119             ret=w[x];
120             x=son[x][1];
121         } else {
122             x=son[x][0];
123         }
124     }
125     if(last)splay(last,0);
126     return ret;
127 }
128 //第一个大于val的值
129 int getne(int val) {
130     int x=root,ret=0,last=0;
131     while(x) {
132         last=x;
133         if(val<w[x]) {
134             ret=w[x];
135             x=son[x][0];
136         } else {
137             x=son[x][1];
138         }
139     }
140     if(last)splay(last,0);
141     return ret;
142 }
143 int main() {
144     scanf("%d",&n);
145     for(int i=1,x,y; i<=n; ++i) {
146         scanf("%d%d",&x,&y);
147         if(x==1)insert(y);
148         if(x==2)delet(y);
149         if(x==3)printf("%d\n",getrank(y));
150         if(x==4)printf("%d\n",kth(y));
151         if(x==5)printf("%d\n",getpre(y));
152         if(x==6)printf("%d\n",getne(y));
153     }
154 }
```

## 2.5 整体二分

主席树区间第 k 大的模板
前题必须要可以离线，常数大于主席树

```cpp
#include<bits/stdc++.h>
#define mid (l+r>>1)
using namespace std;
const int N=2e5+10;
const int inf=0x3f3f3f3f;
int n,m,cnt,ans[N];
struct A{
    int l,r,k,id,op;
}q[N<<1],q1[N<<1],q2[N<<1];
int c[N];
int lb(int a){return a&-a;}
void add(int p,int v){for(;p<=n;p+=lb(p))c[p]+=v;}
int query(int p){
    int ret=0;
    for(;p;p-=lb(p))ret+=c[p];
    return ret;
}
//通过分治对数组和第k大的答案进行排序
void solve(int l,int r,int ql,int qr)
{
    if(ql>qr)return;
    if(l==r){
        for(int i=ql;i<=qr;i++)if(q[i].op==2)ans[q[i].id]=l;
        return;
    }

    int cnt1=0,cnt2=0;
    for(int i=ql;i<=qr;i++){
        if(q[i].op==1){
            if(q[i].l<=mid)q1[++cnt1]=q[i],add(q[i].id,1);
            else q2[++cnt2]=q[i];
        }
        else{
            int t=query(q[i].r)-query(q[i].l-1);
            if(q[i].k<=t) q1[++cnt1]=q[i];
            else q[i].k-=t,q2[++cnt2]=q[i];
        }
    }
    for(int i=1;i<=cnt1;i++)if(q1[i].op==1)add(q1[i].id,-1);//清空BT
    for(int i=1;i<=cnt1;i++)q[i+ql-1]=q1[i];
    for(int i=1;i<=cnt2;i++)q[ql+cnt1+i-1]=q2[i];
    solve(l,mid,ql,ql+cnt1-1);
    solve(mid+1,r,ql+cnt1,qr);
}
int main()
{
    cin>>n>>m;
    for(int i=1;i<=n;i++){
        cnt++;
        cin>>q[cnt].l;//op=1时用不到r
        q[cnt].id=i,q[cnt].op=1;
    }
    for(int i=1;i<=m;i++){
        cnt++;
        cin>>q[cnt].l>>q[cnt].r>>q[cnt].k;
        q[cnt].id=i,q[cnt].op=2;
    }
    solve(-inf,inf,1,cnt);
```

```
    for(int i=1;i<=m;i++)cout<<ans[i]<<endl;
    return 0;
}
```

## 2.6 可撤销并查集

```
struct UFS {
    struct A{int tp,id,w;};
    stack<A> stk;
    vector<int>fa,rnk;
    UFS(int n) {
        fa.assign(n+1,0);
        rnk.assign(n+1,0);
        for (int i = 0; i <= n; ++i) fa[i] = i;
    }
    int Find(int x) {
        while(x^fa[x]) x = fa[x];
        return x;
    }
    int Merge(int x, int y) {   //返回本次合并栈的增加数量
        int ret=0;
        x = Find(x), y = Find(y);
        if(x == y) return 0;
        ans++;
        if(rnk[x] <= rnk[y]) {
            stk.push({1, x, fa[x]});ret++;
            fa[x] = y;
            if(rnk[x] == rnk[y]) {
                stk.push({2, y, rnk[y]});ret++;
                rnk[y]++;
            }
        }
        else {
            stk.push({1, y, fa[y]});ret++;
            fa[y] = x;
        }
        return ret;
    }
    void Undo() {
        A a=stk.top();stk.pop();
        if(a.tp==1)fa[a.id]=a.w,ans--;
        else rnk[a.id]=a.w;
    }
};
```

# 3 字符串

## 3.1 KMP

```
char a[N], b[N];
int nxt[N];
int main()
{
    cin >> a + 1 >> b + 1; //字符串从1开始读
    int la = strlen(a + 1), lb = strlen(b + 1);
    for(int i = 2, j = 0; i <= lb; i++){ //j是能和i-1匹配的前缀
        while(j && b[i] != b[j + 1]) j = nxt[j];
        if(b[i] == b[j + 1]) j++;
        nxt[i] = j;
```

```
    }
    for(int i = 1, j = 0; i <= la; i++){
        while(j && a[i] != b[j + 1]) j = nxt[j];
        if(a[i] == b[j + 1]) j++;
        if(j == lb){//此时的a可以和完整的b匹配
            cout << i - lb + 1 << endl;
            j = nxt[j];
        }
    }
    return 0;
}
/*
 * ababa
 * i:      1 2 3 4 5
 * nxt[i]: 0 0 1 2 3
 */
```

## 3.2 Manacher

最长回文串模板

```
char Ma[N<<1];
int Mp[N<<1];
void Manacher(char s[]){
    int l=0,len=strlen(s);
    Ma[l++]='$';
    Ma[l++]='#';
    for(int i=0;i<len;i++){
        Ma[l++]=s[i];
        Ma[l++]='#';
    }
    Ma[l]=0;
    int mx=0,id=0;//mx是最右端，id是中间值
    for(int i=0;i<l;i++){
        Mp[i]=mx>i?min(Mp[2*id-i],mx-i):1;
        while(Ma[i+Mp[i]]==Ma[i-Mp[i]])Mp[i]++;
        if(i+Mp[i]>mx){
            mx=i+Mp[i];
            id=i;
        }
    }
}
/*
 * abaaba
 * i:     0 1 2 3 4 5 6 7 8 9 10 11 12 13
 * Ma[i]: $ # a # b # a # a # b  #  a  #
 * Mp[i]: 1 1 2 1 4 1 2 7 2 1 4  1  2  1
 */
```

## 3.3 回文树

最长回文串模板

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 1e6+10;
struct node {
    int next[26];
    int len;
    int sufflink;
};
```

```
9
10  int len;
11  char s[maxn];
12  node tree[maxn];
13  int num;            // node 1 - root with len -1, node 2 - root with len 0
14  int suff;           // max suffix palindrome
15
16  bool addLetter(int pos) {
17      int cur = suff, curlen = 0;
18      int let = s[pos] - 'a';
19
20      while (true) {
21          curlen = tree[cur].len;
22          if (pos - 1 - curlen >= 0 && s[pos - 1 - curlen] == s[pos])
23              break;
24          cur = tree[cur].sufflink;
25      }
26      if (tree[cur].next[let]) {
27          suff = tree[cur].next[let];
28          return false;
29      }
30
31      num++;
32      suff = num;
33      tree[num].len = tree[cur].len + 2;
34      tree[cur].next[let] = num;
35
36      if (tree[num].len == 1) {
37          tree[num].sufflink = 2;
38          return true;
39      }
40
41      while (true) {
42          cur = tree[cur].sufflink;
43          curlen = tree[cur].len;
44          if (pos - 1 - curlen >= 0 && s[pos - 1 - curlen] == s[pos]) {
45              tree[num].sufflink = tree[cur].next[let];
46              break;
47          }
48      }
49      return true;
50  }
51
52  void initTree() {
53      num = 2; suff = 2;
54      tree[1].len = -1; tree[1].sufflink = 1;
55      tree[2].len = 0; tree[2].sufflink = 1;
56  }
57
58  int main() {
59      scanf("%s", s);
60      len = strlen(s);
61
62      initTree();
63
64      int ans=1;
65      for (int i = 0; i < len; i++) {
66          addLetter(i);
67          ans=max(ans,tree[suff].len);
68      }
69      cout << ans << endl;
70      return 0;
71  }
```

## 3.4 后缀数组

### 3.4.1 后缀数组

求后缀字典序板子

```
char s[N];
int n,m,rk[N],sa[N],tp[N],tmp[N],Height[N];
//m:桶高
//rk: 第i处的后缀的排名
//sa: 后缀下标从大到小排序
//tp:长度为2w的后缀中，第二关键字排名为i的后缀的位置
//tmp:基数排序中的桶
//Height[i]:lcp(sa[i],sa[i-1])排名为i和i-1的最长前缀
void Bsort(){//基数排序
    for(int i=0;i<=m;i++)tmp[i]=0;
    for(int i=1;i<=n;i++)tmp[rk[i]]++;
    for(int i=1;i<=m;i++)tmp[i]+=tmp[i-1];
    for(int i=n;i>=1;i--)sa[tmp[rk[tp[i]]]--]=tp[i];
}
void SuffixSort()
{
    scanf("%s",s+1);
    n=strlen(s+1);
    m=75;//'z'-'0'=74
    for(int i=1;i<=n;i++){
        rk[i]=s[i]-'0';
        tp[i]=i;
    }
    Bsort();
    for(int w=1,p=0;p<n;m=p,w<<=1){
        p=0;
        for(int i=1;i<=w;i++)tp[++p]=n-w+i;
        for(int i=1;i<=n;i++)if(sa[i]>w)tp[++p]=sa[i]-w;
        Bsort();
        swap(tp,rk);//相当于把老rk赋值给tp，再更新rk，节省一个数组
        rk[sa[1]]=p=1;
        for(int i=2;i<=n;i++)
        rk[sa[i]]=(tp[sa[i-1]]==tp[sa[i]]&&tp[sa[i-1]+w]==tp[sa[i]+w])?p:++p;
    }
}
void GetHeight(){
    for(int i=1,k=0;i<=n;i++){
        if(k)k--;
        int j=sa[rk[i]-1];
        while(s[i+k]==s[j+k])k++;
        Height[rk[i]]=k;
    }
}
```

### 3.4.2 sam

```
#include<bits/stdc++.h>
#define intt long long
using namespace std;
const int maxx = 5e6+1314;
int lin[maxx],len;
const int mx= 4e7;
struct edge{int y,next;}e[maxx];
int read(){int k;scanf("%d",&k);return k;}
void init(int x,int y){e[++len].y=y;e[len].next=lin[x];lin[x]=len;}
int dp[maxx];
```

```
11 int idx(char a){return a-'a'+1;}
12 struct SAM{
13     int tr[maxx][27],cnt,now,par[maxx],mx[maxx],flag[maxx];
14     void clear(){
15         for(int i=1;i<=cnt;i++) memset(tr[i],0,sizeof(tr[i])),par[i]=mx[i]=0,lin
               [i]=0,dp[i]=0;
16         cnt=now=1;len=0;
17     }
18     void extend(int x,int len=0){
19         int np=++cnt,p=now;
20         mx[np]=mx[p]+1;
21         now=np;
22         while(p&&!tr[p][x])tr[p][x]=np,p=par[p];
23         if(!p)par[np]=1;
24         else{
25             int q=tr[p][x];
26             if(mx[p]+1==mx[q])par[np]=q;
27             else {
28                 int nq=++cnt;mx[nq]=mx[p]+1;
29                 par[nq]=par[q];par[np]=par[q]=nq;
30                 memcpy(tr[nq],tr[q],sizeof(tr[nq]));
31                 while(p&&tr[p][x]==q)tr[p][x]=nq,p=par[p];
32             }
33         }
34         flag[np]=len;
35     }
36 }S,T;
37 char s[maxx],t[maxx];
38 int LL,RR;
39 int Q,lent,lens;
40 int ls[mx],rs[mx],sum[mx],root[mx];
41 #define mid (l+r>>1)
42 int tot=0;
43 int findx(int l,int r,int x,int L,int R){
44     if(r<L||l>R)return 0;
45     if(!sum[x])return 0;
46     if(l==r){
47         return r;
48     }
49     if(L<=l&&R>=r){
50         if(sum[rs[x]])  return findx(mid+1,r,rs[x],L,R);
51         else return findx(l,mid,ls[x],L,R);
52     }
53     return max(findx(l,mid,ls[x],L,R),findx(mid+1,r,rs[x],L,R));
54 }
55 void find(int L,int R){
56     int now1=1,now2=1;
57     int r=0,len=0;
58     int n=strlen(t+1);
59     while(r<n){
60         int flag=0;
61         int id=idx(t[r+1]);
62         int w=findx(1,lens,root[S.tr[now1][id]],L,R);
63         while(!S.tr[now1][id]||w-L+1<=S.mx[S.par[S.tr[now1][id]]]||w-L<len)
64         {
65             if(now1==1){flag=1;break;}
66             now1=S.par[now1];
67             w=findx(1,lens,root[S.tr[now1][id]],L,R);
68             len=min(S.mx[now1],w-L);
69         }
70         if(flag==1){now1=1,now2=1,r++,len=0;continue;}
71         int X=0;
72         while(len<=T.mx[T.par[now2]]&&now2!=1){
```

```
            now2=T.par[now2];
            X=now2;
        }   //Í¬Ê±ÔÚÁ½¸ö×Ô¶¯¾¦ÉÏÌø
        now1=S.tr[now1][id];
        now2=T.tr[now2][id];
        len++;r++;
        dp[now2]=max(dp[now2],len);
    }
}
void dfs(int x){
    for(int i=lin[x];i;i=e[i].next){
        int y=e[i].y;
        dfs(y);
        dp[x]=max(dp[x],dp[y]);
    }
    return ;
}
void slove(){
    int len=strlen(t+1);
    for(int i=1;i<=len;i++)T.extend(idx(t[i]));
    intt ans=0;
    for(int i=1;i<=T.cnt;i++){
        ans+=T.mx[i]-T.mx[T.par[i]];
        init(T.par[i],i);
    }
    find(LL,RR);
    dfs(1);
    for(int i=2;i<=T.cnt;i++){
        if(dp[i]){
            intt Ans=min(dp[i],T.mx[i])-T.mx[T.par[i]];
            ans -=Ans;
        }
    }
    cout<<ans<<endl;
    return ;
}
void insert(int l,int r,int &x,int id){
    if(!x)x=++tot;
    sum[x]++;
    if(l==r){return ;}
    if(id<=mid)insert(l,mid,ls[x],id);
    else insert(mid+1,r,rs[x],id);
}
int mager(int x,int y){
    int o=++tot;
    if(x*y==0)return x+y;
    sum[o]=sum[x]+sum[y];
    ls[o]=mager(ls[x],ls[y]);
    rs[o]=mager(rs[x],rs[y]);
    return o;
}
void dfsx(int x){
    for(int i=lin[x];i;i=e[i].next){
        int y=e[i].y;
        dfsx(y);
        root[x]=mager(root[x],root[y]);
    }
}
int main(){
    scanf(”%s”,s+1);
    S.clear();
    lens=strlen(s+1);
    for(int i=1;i<=lens;i++)S.extend(idx(s[i]),i);
```

```
    for(int i=1;i<=S.cnt;i++){
        if(S.flag[i])
            insert(1,lens,root[i],S.flag[i]);
    }
    for(int i=1;i<=S.cnt;i++)init(S.par[i],i);
    dfsx(1);
    intt ans=0;
    for(int i=1;i<=S.cnt;i++)   ans+=S.mx[i]-S.mx[S.par[i]],lin[i]=0;
    Q=read();
    for(int i=1;i<=Q;i++){
        scanf("%s",t+1);
        LL=read();RR=read();
        T.clear();
        slove();
    }
    return 0;
}
```

## 3.5 AC 自动机

模板

根下标从 0 开始

s 是 t 的后缀等价于 t 串终止节点能通过 fail 指针走到 s 终止节点，即 t 串终止节点是 s 终止节点在 fail 树上的孩子。

```
const int N=1e6+10,M=26;
int tr[N][M],fail[N],w[N],tr_cnt;//都要初始化为0
void build(){
    int n;cin>>n;
    for(int i=0;i<n;i++){
        string s;cin>>s;
        int o=0;
        for(auto j:s){
            if(!tr[o][j-'a'])tr[o][j-'a']=++tr_cnt;
            o=tr[o][j-'a'];
        }
        w[o]++;
    }
    queue<int>q;
    for(int i=0; i<M; i++)if(tr[0][i])q.push(tr[0][i]);
    while(!q.empty()){
        int o=q.front();q.pop();
        for(int i=0; i<M; i++){
            if(tr[o][i]){
                fail[tr[o][i]]=tr[fail[o]][i];
                q.push(tr[o][i]);
            }
            else tr[o][i]=tr[fail[o]][i];
        }
    }
    //for(int i=1;i<=tr_cnt;i++)edg[fail[i]].push_back(i);
}
int query(string s){
    int o=0, ret=0;
    for(auto i:s){
        o=tr[o][i-'a'];
        for(int t=o; t&&~w[t]; t=fail[t])
            ret+=w[t],w[t]=-1;
    }
    return ret;
}
```

字符集很大（不止 26）个时，需要用可持久化数组来维护每个节点的 trie 树，下面是主席树版本和 rope 版本。

```
int n,fail[N];
ll w[N];
vi edg[N];
map<int,int>tr[N];
struct A{int w,lo,ro;}nd[N<<5];
int rt[N],tot;
void build(int &o,int l=1,int r=1e5){
    o=tot++;
    if(l==r){
        if(tr[0].find(l)!=tr[0].end())nd[o].w=tr[0][l];
        return;
    }
    build(nd[o].lo,l,mid);
    build(nd[o].ro,mid+1,r);
}
void update(int p,int v,int &o,int l=1,int r=1e5){
    nd[tot]=nd[o];
    o=tot++;
    if(l==r){nd[o].w=v;return;}
    if(p<=mid)update(p,v,nd[o].lo,l,mid);
    else update(p,v,nd[o].ro,mid+1,r);
}
int ask(int p,int o,int l=1,int r=1e5){
    if(l==r)return nd[o].w;
    if(p<=mid)return ask(p,nd[o].lo,l,mid);
    else return ask(p,nd[o].ro,mid+1,r);
}
void build(){
    for(int i=0;i<n;i++){
        int u,v,c;scanf("%d%d%d",&u,&v,&c);
        tr[u][c]=v;//用map<int,int>建立trie树
    }
    queue<int>q;
    build(rt[0]);
    for(auto i:tr[0])q.push(i.se);
    while(!q.empty()){
        int o=q.front();q.pop();
        rt[o]=rt[fail[o]];
        for(auto i:tr[o]){
            update(i.fi,i.se,rt[o]);
            fail[i.se]=ask(i.fi,rt[fail[o]]);
            q.push(i.se);
        }
    }
    for(int i=1;i<=n;i++)edg[fail[i]].push_back(i);
}
```

```
#include<bits/stdc++.h>
#include<ext/rope>
using namespace std;
using namespace __gnu_cxx;
typedef long long ll;
const int N=1e5+10;

ll ans[N];
int fail[N],q[N],t;
vector<pair<int, int>>g[N];
rope<int>go[N];
int main() {
    int n;cin>>n;
```

```
    for(int i=0; i<n; i++) {
        int u,v,c; scanf("%d%d%d",&u,&v,&c);
        g[u].emplace_back(v, c);
    }
    go[0]=rope<int>(N, 0);
    q[t++]=0;
    for(int h=0;h<t; h++) {
        int u=q[h];
        go[u]=go[fail[u]];
        for(auto [v,c]:g[u]){
            fail[v]=go[u][c];
            go[u].replace(c,v);
            q[t++]=v;
        }
    }
    for(int i=t-1;i;i--) {
        int u=q[i];
        ans[u]=1;
        for(auto [v,c]:g[u]) {
            ans[u]+=ans[v];
        }
    }
    for(int i=t-1;i;i--){
        int u=q[i];
        ans[fail[u]]+=ans[u];
    }
    for(int i=1;i<=n;i++)printf("%lld\n",ans[i]);
}
```

## 3.6 哈希

多少对字符串只有一个位置不同，需要用双哈希

常用素数：311 1949 2027 314159 220000607 1e9+97 410102200006070017 1e18+2049

```
const int mod=1e9+97;
typedef pair<int, int>pii;
int n,m,t,ans;
pii hs[N][210],base[210]={{1,1},{233,220000607}};
pii operator*(pii a,pii b){return {(ll)a.fi*b.fi%mod,(ll)a.se*b.se%mod};}
pii operator+(pii a,pii b){return {(a.fi+b.fi)%mod,(a.se+b.se)%mod};}
pii operator-(pii a,pii b){return {(a.fi-b.fi+mod)%mod,(a.se-b.se+mod)%mod};}
int main(){
    for(int i=2;i<210;i++)base[i]=base[i-1]*base[1];
    cin>>n>>m>>t;
    for(int i=0;i<n;i++){
        char s[210];cin>>s+1;
        for(int j=1;j<=m;j++)hs[i][j]=hs[i][j-1]*base[1]+(pii){s[j],s[j]};
    }
    for(int i=1;i<=m;i++){
        map<pii,int>mp;
        for(int j=0;j<n;j++){
            pii a=hs[j][m]-hs[j][i]*base[m-i]+hs[j][i-1]*base[m-i];
            ans+=mp[a];
            mp[a]++;
        }
    }
    cout<<ans;
    return 0;
}
```

# 4 图论

## 4.1 最小生成树

### 4.1.1 Prim

Prim 求无向图最小生成树
无堆优化 O($n^2$), 空间 O($n^2$), 适用于稠密图
耗费矩阵 cost[n][n]，标号从 0 开始，0~n-1
矩阵 cost[n][n]，初始化为 inf，注意重边问题
返回最小生成树的权值，返回 -1 表示原图不连通

```
struct Prim{
    static const int N=5000+10;
    int n;vector<bool>vis;vector<vi>cost;vi lowc;
    void init(int _n){
        n=_n;vis.assign(n,0);lowc.assign(n,inf);
        cost.assign(n,vi(n,inf));
    }
    void add_edg(int u,int v,int w){cost[u][v]=cost[v][u]=min(cost[u][v],w);}
    ll solve(){
        ll ret=0;
        vis[0]=true;
        for(int i=1;i<n;i++)lowc[i]=cost[0][i];
        for(int i=1;i<n;i++){
            int mi=inf,id=-1;
            for(int j=0;j<n;j++)if(!vis[j]&&mi>lowc[j])
                mi=lowc[j],id=j;
            if(mi == inf)return -1;//原图不连通
            ret+=mi;vis[id]=1;
            for(int j=0;j<n;j++) if(!vis[j]&&lowc[j]>cost[id][j])
                lowc[j]=cost[id][j];
        }
        return ret;
    }
};
```

### 4.1.2 Kruskal

Kruskal 算法求 MST
适用于稀疏图，O(Elog(E))

```
struct Kruskal{
    int n;vector<array<int,3>>edg;vi f;
    void init(int _n){n=_n;edg.clear();f.resize(n);iota(f.begin(),f.end(),0);}
    void add_edg(int a, int b, int c){edg.push_back({c,a,b});}
    int ff(int a){return f[a]==a?a:f[a]=ff(f[a]);}
    ll solve(){
        sort(edg.begin(),edg.end());
        ll ret=0,cnt=0;
        for(auto i:edg){
            if(ff(i[1])!=ff(i[2]))ret+=i[0],f[ff(i[1])]=ff(i[2]),cnt++;
            if(cnt==n-1)return ret;
        }
        return -1;
    }
};
```

### 4.1.3 Kruskal 重构树

```cpp
#include<bits/stdc++.h>
#define bin(i)      (1 << (i))
using namespace std;
const int maxx = 1000000;
struct edge{int y,a,v,next;}e[maxx];
int len=1,tot=0,lin[maxx];
namespace IO {

    const int BUF = bin(20);

    char buf[BUF], *fs, *ft;

    inline char getc(){
        if (fs == ft) {
            fs = buf;
            ft = fs + fread(buf, 1, BUF, stdin);
        }
        return *fs++;
    }

    inline int read(){
        char ch = getc(); int x = 0, f = 1;
        while (ch > '9' || ch < '0') {if (ch == '-') f = -1; ch = getc();}
        while (ch >= '0' && ch <= '9') {x = x * 10 + ch - '0'; ch = getc();}
        return x * f;
    }

}
using IO::read;
void init(int x,int y,int v,int a){e[++len].y=y;e[len].next=lin[x];e[len].v=v;e[
    len].a=a;lin[x]=len;}
struct X{int x,y,a;}xx[maxx*2];
int fa[maxx],dis[maxx],Q,K,S;
struct Node{
    int x,y;
    Node (int x_=0,int y_=0){x=x_;y=y_;}
    friend bool operator<(Node a,Node b){return a.y>b.y;}
};
int Mindis[maxx];
priority_queue<Node>q;
int vis[maxx];
int dp[maxx][21];
int dpx[maxx][21];
int ans=0,n,m;
void dijkstar(int st=1){
    memset(vis,0,sizeof(vis));memset(dis,10,sizeof(dis));
    q.push(Node(1,0));dis[st]=0;vis[st]=1;
    for(int i=1;i<n;i++){
        Node tmp;
        while(!q.empty()){
            tmp=q.top();
            q.pop();
            if(!vis[tmp.x])break;
        }
        int x=tmp.x;
        vis[x]=1;
        for(int i=lin[x];i;i=e[i].next){
            int y=e[i].y;
            if(dis[y]>dis[x]+e[i].v){
                dis[y]=dis[x]+e[i].v;
                q.push(Node(y,dis[y]));
            }
```

```
        }
    }
    while(!q.empty())q.pop();
}
int getfather(int x){
    if(x==fa[x])return x;
    return fa[x]=getfather(fa[x]);
}
void HH(int x,int y){fa[x]=y;}
bool my(X A,X B){return A.a>B.a;}
void build(){
    sort(xx+1,xx+m+1,my);
    memset(lin,0,sizeof(lin));len=1;tot=n;
    for(int i=1;i<=n+m+1;i++)fa[i]=i;
    for(int i=1;i<=m;i++){
        int x=xx[i].x,y=xx[i].y;
        x=getfather(x);
        y=getfather(y);
        if(x==y)continue;
        HH(x,++tot);
        HH(y,tot);
        init(tot,x,0,xx[i].a);
        init(tot,y,0,xx[i].a);
    }
}
void dfs(int x,int fa){
    Mindis[x]=dis[x];dpx[x][0]=fa;
    for(int i=1;i<=20;i++)dpx[x][i]=dpx[dpx[x][i-1]][i-1];
    for(int i=1;i<=20;i++)
    {
        dp[x][i]=min(dp[x][i-1],dp[dpx[x][i-1]][i-1]);
    }
    for(int i=lin[x];i;i=e[i].next){
        int y=e[i].y;
        if(y==fa)continue;
        dp[y][0]=e[i].a;
        dfs(y,x);
        Mindis[x]=min(Mindis[x],Mindis[y]);
    }
}
void Make(int x,int p){
    for(int i=20;i>=0;i--)
    if(dp[x][i]>p){x=dpx[x][i];}
    ans=Mindis[x];
    printf("%d\n",ans);
}
void initx(){
    memset(lin,0,sizeof(lin));len=1;
    n=read();m=read();
    for(int i=1;i<=m;i++){
        int x=read();int y=read();int v=read();int a=read();
        init(x,y,v,a);init(y,x,v,a);
        xx[i].x=x;xx[i].y=y;xx[i].a=a;
    }
}
void slove(){
    memset(dp,127,sizeof(dp));
    memset(dpx,0,sizeof(dpx));
    dfs(tot,0);
    ans=0;
    Q=read();K=read();S=read();
    while(Q--){
        int v=read();int p=read();
```

```
        v=(v+K*ans-1)%n+1;
        p=(p+K*ans)%(S+1);
        Make(v,p);
    }
}
int main(){
    int T=read();
    while(T--){
        initx();
        dijkstar();
        build();
        dfs(tot,0);
        slove();
    }
    return 0;
}
```

## 4.2 最短路

下标都是从 0 开始

### 4.2.1 Dij

```
struct Dij{
    int n,s;
    vector<bool>vis;
    vector<ll>dis;
    vector<vector<pair<int,ll> > >edg;
    Dij(int n,int s=0){
        this->n=n;this->s=s;
        vis.assign(n,0);dis.assign(n,linf);edg.assign(n,{});
    }
    void add_edg(int u,int v,ll w){edg[u].emplace_back(v,w);};
    void solve(){
        priority_queue<pair<ll,int>>q;
        q.push({0,s});dis[s]=0;
        while(!q.empty()){
            int u=q.top().se;q.pop();
            if(vis[u])continue;
            vis[u]=1;
            for(auto i:edg[u]){
                int v=i.fi,w=i.se;
                if(vis[v]||dis[u]+w>=dis[v])continue;
                dis[v]=dis[u]+w;
                q.emplace(-dis[v],v);
            }
        }
    }
};
```

### 4.2.2 SPFA

```
struct SPFA{
    vector<bool>vis;vector<ll>dis;vector<int>cnt;
    vector<vector<pair<int,ll>>>edg;
    int n,s;
    void init(int _n,int _s=0){
        n=_n;s=_s;
```

```
        vis.assign(n,0);dis.assign(n,inf);edg.assign(n,{});cnt.assign(n,0);
    }
    void add_edg(int u,int v,ll w){edg[u].emplace_back(v,w);}
    bool solve(){
        queue<int>q;
        q.push(s);vis[s]=cnt[s]=1;dis[s]=0;
        while(!q.empty()){
            int u=q.front();q.pop();vis[u]=0;
            for(auto i:edg[u]){
                int v=i.first,w=i.second;
                if(dis[v]>dis[u]+w){
                    dis[v]=dis[u]+w;
                    if(!vis[v]){
                        q.push(v);vis[v]=1;
                        if(++cnt[v]>n)return 0;
                    }
                }
            }
        }
        return 1;
    }
};
```

## 4.3 网络流

### 4.3.1 Dinic

洛谷板子题，两点间最大流

时间复杂度最坏情况 $O(m \times n^2)$, 二分图 $O(m \times n^{0.5})$

```
struct Dinic{
    struct tri{int a;ll b;int c;};
    vector<tri>e;
    int n,s,t;//点下标1~n，边下标0~e_cnt-1
    vector<int>head,head2,dep;
    Dinic(int _n,int _s,int _t):n(_n),s(_s),t(_t){head.assign(n+1,-1);}
    void add_edg(int a,int b,ll c){
        e.push_back({b,c,head[a]});head[a]=e.size()-1;
        e.push_back({a,0,head[b]});head[b]=e.size()-1;
    }
    bool bfs(){
        queue<int>q;q.push(s);
        dep.assign(n+1,0);dep[s] = 1;
        while(!q.empty()){
            int u=q.front();q.pop();
            for(int i=head[u];~i;i=e[i].c){
                int v=e[i].a;
                if(e[i].b&&!dep[v]){
                    dep[v]=dep[u]+1;
                    if(v==t)return 1;
                    q.push(v);
                }
            }
        }
        return 0;
    }
    ll dfs(int u,ll low){
        if(u==t||!low)return low;
        ll ret=0;
        for(int &i=head2[u];~i;i=e[i].c){
            int v=e[i].a;
            if(dep[v]==dep[u]+1&&e[i].b){
```

```
                ll flow=dfs(v,min(low,e[i].b));
                if(!flow)continue;
                e[i].b-=flow;e[i^1].b+=flow;
                low-=flow;ret+=flow;
                if(!low)break;
            }
        }
        return ret;
    }
    ll solve(){
        ll ret=0;e.resize(e.size());
        while(bfs())head2=head,ret+=dfs(s,1e18);
        return ret;
    }
};
```

### 4.3.2 SPFA 费用流

洛谷板子题，最小费用最大流
时间复杂度最坏情况 $O(m \times n^2)$, 二分图 $O(m \times n^{0.5})$
就是把 EK 算法中的 bfs 换成以费用为标准的 SPFA
据说最坏复杂度为 $O(n^5)$，但无法证明，玄学

```
struct EK{
    vector<int>head,pre,d,inq;
    int n,s,t,cnt;
    ll ans1,ans2;
    struct Edg{int ne,to,w,f;};//w是容量，f是费用
    vector<Edg>e;
    void add_edg(int x,int y,int w,int f){
        e.push_back({head[x],y,w,f});head[x]=e.size()-1;
        e.push_back({head[y],x,0,-f});head[y]=e.size()-1;
    }
    void init(int _n,int _s,int _t){
        n=_n,s=_s,t=_t;ans1=ans2=cnt=0;
        head.assign(n,-1);pre.assign(n,-1);
    }
    bool spfa(){
        d.assign(n,2e9);
        inq.assign(n,0);
        queue<int>q;
        d[s]=0;q.push(s);inq[s]=1;
        while(!q.empty()){
            int x=q.front();q.pop();inq[x]=0;
            for(int i=head[x];~i;i=e[i].ne){
                int y=e[i].to;
                if(d[y]>d[x]+e[i].f&&e[i].w>0){
                    d[y]=d[x]+e[i].f;
                    pre[y]=i;
                    if(!inq[y])inq[y]=1,q.push(y);
                }
            }
        }
        return d[t]!=2e9;
    }
    void dinic(){
        int flow=2e9;
        for(int i=pre[t];~i;i=pre[e[i^1].to])flow=min(flow,e[i].w);
        for(int i=pre[t];~i;i=pre[e[i^1].to])e[i].w-=flow,e[i^1].w+=flow;
        ans1+=flow;
        ans2+=flow*d[t];
```

```
    }
    void solve(){
        while(spfa())dinic();
    }
};
```

### 4.3.3 匈牙利匹配

时间复杂度：O（VE)
适用于稠密图，DFS 找增广路

```
struct Hungary{
    static const int N=1000+10;
    int n,m,linker[N];//左右两边的点集数量，下标从0开始
    bool vis[N];
    vi edg[N];
    void add_edg(int u,int v){edg[u].push_back(v);}
    void init(int _n,int _m){
        n=_n,m=_m;
        for(int i=0;i<n;i++)edg[i].clear();
    }
    bool dfs(int u){
        for(auto v:edg[u])if(!vis[v]){
            vis[v]=1;
            if(linker[v]==-1||dfs(linker[v]))return linker[v]=u,1;
        }
        return 0;
    }
    int solve(){
        int ret=0;
        memset(linker,-1,sizeof(linker));
        for(int i=0;i<n;i++){
            memset(vis,0,sizeof(vis));
            if(dfs(i))ret++;
        }
        return ret;
    }
};
```

## 4.4 Tarjan

### 4.4.1 强连通

有向图求强连通分量
下标从 0 开始

```
struct Tarjan{
    int n,scc,index;
    vector<int>stk,dfn,low,in_stk,belong;
    vector<vector<int> >edg;
    void init(int _n){
        n=_n;index=scc=0;edg.assign(n,{});in_stk.assign(n,0);
        dfn.assign(n,0);low.assign(n,0);belong.assign(n,0);
    }
    void add_edg(int u,int v){edg[u].push_back(v);}
    void tarjan(int u){
        dfn[u]=low[u]=++index;
        stk.push_back(u);in_stk[u]=1;
        for(auto v:edg[u]){
            if(!dfn[v]){
```

```
                tarjan(v);
                low[u]=min(low[u],low[v]);
            }
            else if(in_stk[v])low[u]=min(low[u],dfn[v]);
        }
        if(dfn[u]==low[u]){
            scc++;
            while(1){
                int v=stk.back();stk.pop_back();in_stk[v]=0;
                belong[v]=scc;
                if(v==u)break;
            }
        }
    }
    void solve(){for(int i=0;i<n;i++)if(!dfn[i])tarjan(i);}
};
```

### 4.4.2 边双

带重边联通无向图中割去一条边，使两边点权差最小
建双向边，注意连通性

```
struct Tarjan{
    int n,index;
    vector<int>dfn,low;
    vector<vector<int> >edg;
    void init(int _n){
        n=_n;index=0;edg.assign(n,{});dfn.assign(n,0);low.assign(n,0);
    }
    void add_edg(int u,int v){edg[u].push_back(v);}
    void tarjan(int u,int f=-1){
        dfn[u]=low[u]=++index;
        bool fg=1;//处理父亲是重边的情况
        for(auto v:edg[u]){
            if(v==f&&fg){fg=0;continue;}
            if(!dfn[v]){
                tarjan(v,u);
                low[u]=min(low[u],low[v]);
                if(dfn[u]<low[v]);//u-v为割边
            }
            else low[u]=min(low[u],dfn[v]);
        }
    }
    void solve(){for(int i=0;i<n;i++)if(!dfn[i])tarjan(i);}
};
```

### 4.4.3 点双

无向非联通图求割点数量
建双向边，注意连通性

```
struct Tarjan{
    int n,index,st;
    vector<int>dfn,low;
    vector<vector<int> >edg;
    void init(int _n){
        n=_n;index=0;edg.assign(n,{});dfn.assign(n,0);low.assign(n,0);
    }
    void add_edg(int u,int v){edg[u].push_back(v);edg[v].push_back(u);}
    void tarjan(int u,int f=-1){
```

```
        dfn[u]=low[u]=++index;
        bool fg=0;//头节点的子树个数
        for(auto v:edg[u])if(v!=f){
            if(!dfn[v]){
                tarjan(v,u);
                low[u]=min(low[u],low[v]);
                if(u==st){
                    if(fg);//u是割点
                    fg=1;
                }
                else if(dfn[u]<=low[v]);//u是割点
            }
            else low[u]=min(low[u],dfn[v]);
        }
    }
    void solve(){for(st=0;st<n;st++)if(!dfn[st])tarjan(st);}
};
```

### 4.4.4 2-sat

给出 n 个集合，每个集合有两个元素，已知若干个 <a,b>，表示 a 与 b 矛盾（其中 a 与 b 属于不同的集合）。然后从每个集合选择一个元素，判断能否一共选 n 个两两不矛盾的元素。

Tarjan 法，复杂度 n+m，输出方案为 belong 小的，不唯一

```
struct Tarjan sw;
int n,m;
bool solve(){
    sw.solve();
    for(int i=0;i<2*n;i+=2)if(sw.belong[i]==sw.belong[i+1])return 0;
    return 1;
}
int main(){
    cin>>n>>m;
    sw.init(n*2);
    for(int i=0;i<m;i++){
        int a,b,c,d;cin>>a>>b>>c>>d;
        sw.add_edg(a*2+c,b*2+d^1);
        sw.add_edg(b*2+d,a*2+c^1);
    }
    cout<<(solve()?"YES":"NO")<<endl;
    return 0;
}
```

暴力染色法，复杂度 n(n+m)，方案为最小字典序

```
struct Twosat
{
    int n;//一共n对
    vi s;
    vector<vi>edg;
    vector<bool>mark;
    Twosat(int _n):n(_n){
        edg.assign(2*n, {});
        mark.assign(2*n, 0);
        s.reserve(2*n);
    }
    bool dfs(int u){
        if (mark[u^1]) return false;
        if (mark[u]) return true;
        mark[u] = true;
        s.push_back(u);
        for(auto v:edg[u])if(!dfs(v))return false;
```

```
        return true;
    }
    void add_clause(int x, int y){
        edg[x].push_back(y ^ 1);// 选了 x 就必须选 y^1
        edg[y].push_back(x ^ 1);// 选了 y 就必须选 x^1
    }
    bool solve(){
        for (int i=0; i<2*n; i+=2){
            if(mark[i]||mark[i+1])continue;
            s.clear();
            if (!dfs(i)){
                for(auto j:s)mark[j] = false;
                if (!dfs(i + 1)) return false;
            }
        }
        return true;
    }
};
int main()
{
    int n,m;cin>>n>>m;
    Twosat sw(n);
    for(int i=0;i<m;i++){
        int a,b,c,d;cin>>a>>b>>c>>d;
        sw.add_clause(a*2-2+b^1, c*2-2+d^1);
    }
    if(!sw.solve())cout<<"IMPOSSIBLE"<<endl;
    else{
        cout<<"POSSIBLE"<<endl;
        for(int i=0;i<n*2;i+=2)cout<<!sw.mark[i]<<' ';
    }
    return 0;
}
```

## 4.5 树

### 4.5.1 树链剖分

无向有根树，路径、子树的修改、查询
重链剖分到根的虚边最多 $\log(n)$ 个
长链剖分到根的虚边最多 (根号 n) 个
长链剖分多处理根层有关的问题，只需在 dfs1 中修改一处

```
struct T{
    static const int N=1e5+10;
    int n,m,r=1,w[N];
    vector<int>edg[N];
    int dep[N],sz[N],fa[N],son[N];//节点深度、子节点数量、父亲、重儿子
    int id[N],rk[N],top[N];//线段树下标、id的相反数组、重链顶
    void dfs1(int u, int f, int d){
        sz[u]=1;dep[u]=d;fa[u]=f;
        for(auto v:edg[u])if(v!=f){
            dfs1(v, u, d + 1);
            sz[u] += sz[v];
            if(sz[v] > sz[son[u]]) son[u] = v;
        }
    }
    int dfs_cnt = 0;
    void dfs2(int u){
        id[u] = ++dfs_cnt;
        if(son[fa[u]] == u) top[u] = top[fa[u]];
```

```
        else top[u] = u;
        rk[id[u]] = u;
        if(son[u]) dfs2(son[u]);
        for(auto v:edg[u])if(v!=son[u]&&v!=fa[u])dfs2(v);
    }
    ll sum[N << 2], lazy[N << 2];
    void pushdown(int o, int l, int r){
        sum[lo]+=lazy[o]*(mid-l+1),sum[ro]+=lazy[o]*(r-mid);
        lazy[lo] += lazy[o],lazy[ro] += lazy[o];
        lazy[o] = 0;
    }
    void build(int o, int l, int r){
        if(l == r){sum[o]=w[rk[l]];return;}
        build(lo, l, mid);
        build(ro, mid + 1, r);
        sum[o] = sum[lo] + sum[ro];
    }
    ll query(int ql, int qr, int o, int l, int r){
        if(ql <= l && r <= qr) return sum[o];
        pushdown(o, l, r);
        ll ret = 0;
        if(ql <= mid) ret += query(ql, qr, lo, l, mid);
        if(qr > mid) ret += query(ql, qr, ro, mid + 1, r);
        return ret;
    }
    void add(int ql, int qr, ll v, int o, int l, int r){
        if(ql <= l && r <= qr){sum[o]+=v*(r-l+1); lazy[o]+=v; return;}
        pushdown(o, l, r);
        if(ql <= mid) add(ql, qr, v, lo, l, mid);
        if(qr > mid) add(ql, qr, v, ro, mid + 1, r);
        sum[o] = sum[lo] + sum[ro];
    }
    ll querys(int x, int y){
        int fx = top[x], fy = top[y], ret = 0;
        while(fx != fy){
            if(dep[fx] < dep[fy])swap(fx, fy), swap(x, y);
            ret += query(id[fx], id[x], 1, 1, n);
            x = fa[fx], fx = top[x];
        }
        if(id[x] > id[y]) swap(x, y);
        ret+=query(id[x], id[y], 1, 1, n);//不算顶点时改成 id[x]+1
        return ret;
    }
    void adds(int x, int y, ll z){
        int fx = top[x], fy = top[y];
        while(fx != fy){
            if(dep[fx] < dep[fy])swap(fx, fy), swap(x, y);
            add(id[fx], id[x], z, 1, 1, n);
            x = fa[fx], fx = top[x];
        }
        if(id[x] > id[y])swap(x, y);
        add(id[x], id[y], z, 1, 1, n);
    }
    void init(){
        cin>>n;
        for(int i=0;i<n-1;i++){
            int a,b;cin>>a>>b;
            edg[a].push_back(b);edg[b].push_back(a);
        }
        dfs1(r,0,1);dfs2(r);build(1,1,n);
    }
};
```

### 4.5.2 LCA

倍增求 LCA

```
struct LCA{
    static const int N=5e5+10;
    vi edg[N];
    int n, fa[N][20], lg[N]{-1}, dep[N];
    void init(int _n){
        n=_n;
        for(int i=1;i<=n;i++){
            lg[i]=lg[i-1]+(i&i-1?0:1);
            edg[i].clear();
            memset(fa[i],0,sizeof(fa[i]));
        }
    }
    void dfs(int u, int f, int d){
        dep[u]=d;
        fa[u][0]=f;
        for(int i=1; d>1<<i; i++)fa[u][i]=fa[ fa[u][i-1] ][i-1];
        for(int v:edg[u])if(v!=f)dfs(v, u, d+1);
    }
    int lca(int a, int b){
        if(dep[a] < dep[b])swap(a, b);
        while(dep[a] > dep[b])a = fa[a][lg[dep[a] - dep[b]]];
        if(a == b)return a;
        for(int i=lg[dep[a]]; ~i; i--)
            if(fa[a][i]!=fa[b][i]) a=fa[a][i], b=fa[b][i];
        return fa[a][0];
    }
    void build(int r){dfs(r,0,1);}
};
```

### 4.5.3 点分治

模板题，求带边权树上是否存在距离恰好为 k 的点对

找到重心，以重心为根做 dfs 进行分治

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1e4+10;
int n,m,k[110],ans[110],cnt,head[N];
int rt,mx[N]{0x3f3f3f3f},sz[N],sum;
bool vis[N];
struct nd{int v,w,ne;} e[N<<1];
void add_edg(int u,int v,int w){e[++cnt]={v,w,head[u]};head[u]=cnt;}
void getrt(int u,int f){//找重心
    sz[u]=1;mx[u]=0;
    for(int i=head[u]; i; i=e[i].ne){
        int v=e[i].v;
        if(v==f||vis[v])continue;
        getrt(v,u);
        sz[u]+=sz[v];
        mx[u]=max(mx[u],sz[v]);
    }
    mx[u]=max(mx[u],sum-sz[u]);
    if(mx[rt]>mx[u])rt=u;
}
unordered_set<int>sv,su;//存储子树层数
void get(int u,int f,int d){//获得分治后子树中的层数
    sz[u]=1;
    if(d<=1e7)sv.insert(d);
```

```
    for(int i=head[u]; i; i=e[i].ne){
        int v=e[i].v,w=e[i].w;
        if(v==f||vis[v])continue;
        get(v,u,d+w);
        sz[u]+=sz[v];
    }
}
void dfs(int u){//分治过程
    vis[u]=1;
    su.clear();
    for(int i=head[u]; i; i=e[i].ne){
        int v=e[i].v,w=e[i].w;
        if(vis[v])continue;
        sv.clear();
        get(v,u,w);
        for(int id=1;id<=m;id++)for(auto j:sv)
            if(j==k[id]||su.count(k[id]-j))ans[id]=1;
        for(auto j:sv)su.insert(j);
    }
    for(int i=head[u]; i; i=e[i].ne){
        int v=e[i].v,w=e[i].w;
        if(vis[v])continue;
        sum=sz[v];rt=0;getrt(v,-1);
        dfs(rt);
    }
}
int main(){
    cin>>n>>m;
    for(int i=0;i<n-1;i++){
        int u,v,w;cin>>u>>v>>w;
        add_edg(u,v,w);
        add_edg(v,u,w);
    }
    for(int i=1;i<=m;i++)cin>>k[i];
    sum=n;rt=0;getrt(1,-1);
    dfs(rt);
    for(int i=1;i<=m;i++){
        cout<<(ans[i]?"AYE":"NAY")<<endl;
    }
    return 0;
}
```

## 4.6 三四元环计数

模板题，无向图三元环技术，复杂度 $O(m\sqrt{m})$

```
int n,m,id[N],rk[N];
vi e[N],e2[N];
int cycle3(){//最大范围为O(m*sqrt(m))
    int ret=0;
    int vis[N]{};
    for(int u=1;u<=n;u++){
        for(auto v:e2[u])vis[v]=u;
        for(auto v:e2[u])for(auto w:e2[v])if(vis[w]==u)ret++;
    }
    return ret;
}
ll cycle4(){
    ll ret=0;
    int cnt[N]{};
    for(int u=1;u<=n;u++){
```

```
        for(auto v:e[u])for(auto w:e2[v])if(id[w]>id[u])ret+=cnt[w]++;
        for(auto v:e[u])for(auto w:e2[v])if(id[w]>id[u])cnt[w]=0;
    }
    return ret;
}
int main(){
    cin>>n>>m;
    for(int i=0;i<m;i++){
        int u,v;cin>>u>>v;
        e[u].push_back(v);e[v].push_back(u);
    }
    iota(rk+1,rk+1+n,1);
    sort(rk+1,rk+1+n,[](int a,int b){return e[a].size()<e[b].size();});
    for(int i=1;i<=n;i++)id[rk[i]]=i;
    for(int u=1;u<=n;u++)for(auto v:e[u])if(id[u]<id[v])e2[u].push_back(v);
    cout<<cycle3<<' '<<cycle4();
    return 0;
}
```

# 5 数学

## 5.1 线性筛

```
vi prm,mu(N),phi(N);
void prepare(){
    bitset<N>vis;
    mu[1]=phi[1]=1;
    for(int i=2;i<N;i++){
        if(!vis[i])prm.push_back(i),mu[i]=-1,phi[i]=i-1;
        for(auto j:prm)if(i*j>=N)break;else{
            vis[i*j]=1;
            if(i%j){
                mu[i*j]=-mu[i];
                phi[i*j]=phi[i]*(j-1);
            }
            else{
                mu[i*j]=0;
                phi[i*j]=phi[i]*j;
                break;
            }
        }
    }
}
```

## 5.2 中国剩余定理

```
struct CRT{
    vector<ll>mod,r;
    //a*x+b*y==1
    //|x|<b  |y|<a
    ll exgcd(ll a,ll b,ll &x,ll &y){
        if(!b){
            x=1,y=0;
            return a;
        }
        ll gcd=exgcd(b,a%b,y,x);
        y-=a/b*x;
        return gcd;
    }
```

```
    ll EXCRT(){
        ll lcm=mod[0],last_r=r[0];
        ll lcm_a,x,y,k;
        for(int i=1; i<mod.size(); i++){
            lcm_a=((r[i]-last_r)%mod[i]+mod[i]);
            k=lcm;
            ll gcd=exgcd(lcm,mod[i],x,y);
            x=(x*lcm_a/gcd%(mod[i]/gcd)+(mod[i]/gcd))%(mod[i]/gcd);
            lcm=lcm*mod[i]/gcd;
            last_r=(last_r+k*x)%lcm;
        }
        return (last_r%lcm+lcm)%lcm;
    }
};
```

## 5.3 高斯消元

### 5.3.1 浮点数高斯消元

```
struct Mat{
    static const int N=200;
    int equ,var,i,j,k,r;//i为列号,k为行号
    double a[N][N],ans[N],t;//一个大小为 equ*(var+1)的矩阵
    int Gauss() {
        for (i=0,k=0; k<equ&&i<var; k++,i++) {
            for(r=k,j=k+1; j<equ; j++) if(fabs(a[j][i])>fabs(a[r][i]))r=j;
            if (fabs(a[r][i])<1e-8) {k--;continue;}//列全为0
            if (r!=k)for(j=i; j<=var; j++)swap(a[k][j],a[r][j]);
            for(r=k+1; r<equ; r++)for(t=a[r][i]/a[k][i],j=i; j<=var; j++)
                a[r][j]-=a[k][j]*t;
        }
        for(i=k; i<equ ; i++)if(fabs(a[i][var])>1e-8) return -1;//无解
        if(k < var) return var - k;  //自由元个数
        for(i =var-1; ~i; i--) { //回带求解
            for(ans[i]=a[i][var],j=i+1; j<var; j++)ans[i]-=ans[j]*a[i][j];
            ans[i] /= a[i][i];
        }
        return 0;
    }
};
```

### 5.3.2 整数高斯消元

```
struct Mat{
    static const int N=200+10;
    int Lcm( int a, int b ) { return a / __gcd( a, b ) * b; }
    int a[N][N];
    int x[N];//n个解
    int Gauss( int n, int m )//求解n*(m+1)的矩阵
    {
        int k, col, max_r;
        for ( k = 0, col = 0 ; k < n && col < m ; k++, col++ ){
            max_r = k;
            for(int i=k+1;i<n;i++)if( abs(a[i][col])>abs(a[max_r][col]) )max_r=i
                ;
            if (max_r!=k)for(int i=col;i<=m;i++)swap(a[k][i], a[max_r][i]);
            if ( a[k][col] == 0 ){k--;continue;}
            for ( int i = k + 1 ; i < n ; i++ )if(a[i][col]){
                int LCM = Lcm(abs(a[i][col]),abs(a[k][col]));
```

```
                int ta = LCM / abs( a[i][col] );
                int tb = LCM / abs( a[k][col] );
                if ( a[i][col]*a[k][col] < 0 )tb = -tb;
                for(int j=col;j<=m;j++)a[i][j] = a[i][j] * ta - a[k][j] * tb;
            }
        }
        for ( int i = k ; i < n ; i++ )if ( a[i][m] != 0 )return -1;//无解
        if ( k < m )return m - k;//自由元/无限解
        for ( int i = n - 1 ; i >= 0 ; i-- ){
            int tmp = a[i][m];
            for(int j=i+1;j<m;j++)if(a[i][j]!=0)tmp-=a[i][j]*x[j];
            if( tmp % a[i][i] != 0 )return -2;//无整数解
            x[i] = tmp / a[i][i];
        }
        return 0;//有解
    }
};
```

### 5.3.3 01 高斯消元

```
struct Mat{
    static const int N=200+10;
    int equ,var,i,j,k,r;//i为列号,k为行号
    bool a[N][N],ans[N];//一个大小为 equ*(var+1)的矩阵
    int Gauss(){
        for (i=0,k=0; k<equ&&i<var; k++,i++) {
            for(r=k,j=k+1; !a[r][i]&&j<equ; j++)if(a[j][i])r=j;
            if (!a[r][i]) {k--;continue;}//列全为0
            if (r!=k)for(j=i; j<=var; j++)swap(a[k][j],a[r][j]);
            for(r=k+1; r<equ; r++)if(a[r][i])
            for(j=i; j<=var; j++)a[r][j]^=a[k][j];
        }
        for(i=k; i<equ; i++)if(a[i][var])return -1;//无解
        if (k<var) return var - k;//自由元个数
        for (i=var-1; ~i; i--)for (ans[i]=a[i][var],j=i+1; j<var; j++)
                ans[i] ^= ans[j] & a[i][j];//回带求解
        return 0;
    }
};
```

### 5.3.4 线性基

```
ll d[100];//d[i]代表最高数位为i+1的线性基的值
void add(ll a){
    for(int i = 50; ~i; i--) if(a & 1ll << i){
        if(d[i]) a ^= d[i];
        else{d[i] = a;break;}
    }
}
```

### 5.3.5 矩阵树定理

这个定理共分为三个部分:
1. 给出无向图，求这个图的生成树个数。
mat[i][i]=i 度数,mat[i][j]=-(ij 边数)
2. 给出有向图和其中的一个点，求以这个点为根的生成外向树个数。
mat[i][i]=i 入度,mat[i][j]=-(ij 边数)，求根点的余子式

3. 给出有向图和其中一个点，求以这个点为根的生成内向树个数。
mat[i][i]=i 出度,mat[i][j]=-(ij 边数)，求根点的余子式

```
struct MT{
    static const int N=200+10;
    ll mat[N][N], n, mod = 1e9 + 7;
    ll solve(){//求1~n-1的矩阵行列式
        ll ret = 1;
        for(int i = 1, j; i < n; i++){
            for(j = i; !mat[j][i]; j++)//找到第i列第一个非零所在行
                if(j == n) return 0;
            if(j != i){//交换这两行的值
                for(int k = i; k < n; k++)swap(mat[i][k], mat[j][k]);
                ret *= -1;
            }
            for(j = i + 1; j < n; j++)while(mat[j][i]){
                if(abs(mat[j][i]) < abs(mat[i][i])){ //类似欧几里得
                    for(int k = i; k < n; k++)swap(mat[i][k], mat[j][k]);
                    ret *= -1;
                }
                int t = mat[j][i] / mat[i][i];
                for(int k=i; k<n; k++)mat[j][k]=(mat[j][k]-mat[i][k]*t)%mod;
            }
            ret = ret * mat[i][i] % mod;
        }
        return ret + (ret < 0 ? mod : 0);
    }
}mt;
```

## 5.4 多项式

### 5.4.1 FFT

模板

```
#include<bits/stdc++.h>
using namespace std;
const double pi = acos(-1.0);
vector<int>rev;
struct C {
    double x, y;
    C (double _x = 0, double _y = 0) {x = _x, y = _y;}
};
C operator + (C a, C b) { return C(a.x + b.x , a.y + b.y);}
C operator - (C a, C b) { return C(a.x - b.x , a.y - b.y);}
C operator * (C a, C b) { return C(a.x * b.x - a.y * b.y , a.x * b.y + a.y * b.x
    );}
void fft(vector<C>&a, int f) {
    int n = a.size();
    if (rev.size() != n) {
        int k = __builtin_ctz(n) - 1;
        rev.resize(n);
        for (int i = 0; i < n; ++i)
            rev[i] = rev[i >> 1] >> 1 | (i & 1) << k;
    }
    for (int i = 0; i < n; i++)
        if (i < rev[i]) swap(a[i], a[rev[i]]); //求出要迭代的序列
    for (int mid = 1; mid < n; mid <<= 1) { //待合并区间的长度的一半
        C Wn( cos(pi / mid) , f * sin(pi / mid) ); //单位根
        for (int R = mid << 1, j = 0; j < n; j += R) { //R是区间的长度，j表示前
            已经到哪个位置了
            C w(1, 0); //幂
```

```
            for (int k = 0; k < mid; k++, w = w * Wn) { //枚举左半部分
                C x = a[j + k], y = w * a[j + mid + k]; //蝴蝶效应
                a[j + k] = x + y;
                a[j + mid + k] = x - y;
            }
        }
    }
}
vector<C> operator*(vector<C>a,vector<C>b)
{
    int sz=1,tot=a.size()+b.size()-1;
    while (sz < tot)sz <<= 1;
    a.resize(sz);
    b.resize(sz);
    fft(a,1);fft(b,1);
    for (int i = 0; i < sz; ++i)a[i] = a[i] * b[i];
    fft(a,-1);
    a.resize(tot);
    for(auto &i:a)i.x=floor(i.x/sz+0.5);
    return a;
}
int main() {
    int n,m;cin>>n>>m;
    vector<C>a(n+1),b(m+1);
    for (int i = 0; i <= n; i++) cin>>a[i].x;
    for (int i = 0; i <= m; i++) cin>>b[i].x;
    for (auto i:a*b)cout<<(int)i.x<<' ';
    return 0;
}
```

### 5.4.2 NTT

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
constexpr int mod = 998244353;
typedef vector<int>vi;
vi rev, roots{0, 1};
int qm(int a, int b) {
    int ret = 1;
    for (; b; b >>= 1, a = (ll)a * a % mod)
        if (b & 1) ret = (ll) ret * a % mod;
    return ret;
}
void dft(vi &a) {
    int n = a.size();
    if (int(rev.size()) != n) {
        int k = __builtin_ctz(n) - 1;
        rev.resize(n);
        for (int i = 0; i < n; ++i)
            rev[i] = rev[i >> 1] >> 1 | (i & 1) << k;
    }
    for (int i = 0; i < n; ++i)
        if (rev[i] < i)swap(a[i], a[rev[i]]);
    if (int(roots.size()) < n) {
        int k = __builtin_ctz(roots.size());
        roots.resize(n);
        while ((1 << k) < n) {
            int e = qm(3, (mod - 1) >> (k + 1));
            for (int i = 1 << (k - 1); i < (1 << k); ++i) {
                roots[2 * i] = roots[i];
```

```
                roots[2 * i + 1] = 1ll * roots[i] * e % mod;
            }
            ++k;
        }
    }
    for (int k = 1; k < n; k *= 2) {
        for (int i = 0; i < n; i += 2 * k) {
            for (int j = 0; j < k; ++j) {
                int u = a[i + j];
                int v = 1ll * a[i + j + k] * roots[k + j] % mod;
                int x = u + v;
                if (x >= mod)x -= mod;
                a[i + j] = x;
                x = u - v;
                if (x < 0)x += mod;
                a[i + j + k] = x;
            }
        }
    }
}
void idft(vi &a) {
    int n = a.size();
    reverse(a.begin() + 1, a.end());
    dft(a);
    int inv = qm(n, mod - 2);
    for (int i = 0; i < n; ++i)
        a[i] = (ll)a[i] * inv % mod;
}
vi operator*(vi a,vi b)
{
    int sz = 1,tot=a.size()+b.size()-1;
    while (sz < tot)sz <<= 1;
    a.resize(sz);
    b.resize(sz);
    dft(a);dft(b);
    for (int i = 0; i < sz; ++i)
        a[i] = (ll)a[i] * b[i] % mod;
    idft(a);
    a.resize(tot);
    return a;
}
int main() {
    int n,m;cin>>n>>m;
    vector<int>a(n+1),b(m+1);
    for(int i=0;i<=n;i++)cin>>a[i];
    for(int i=0;i<=m;i++)cin>>b[i];
    for(auto i:a*b)cout<<i<<' ';
    return 0;
}
```

### 5.4.3 全家桶

```
struct Poly {
    vector<int> a;
    Poly() {}
    Poly(int a0) {
        if (a0)
            a = {a0};
    }
    Poly(const vector<int> &a1) : a(a1) {
        while (!a.empty() && !a.back())
```

```
            a.pop_back();
    }
    int size() const {
        return a.size();
    }
    int operator[](int idx) const {
        if (idx < 0 || idx >= size())
            return 0;
        return a[idx];
    }
    Poly mulxk(int k) const {
        auto b = a;
        b.insert(b.begin(), k, 0);
        return Poly(b);
    }
    Poly modxk(int k) const {
        k = min(k, size());
        return Poly(vector<int>(a.begin(), a.begin() + k));
    }
    Poly divxk(int k) const {
        if (size() <= k)
            return Poly();
        return Poly(vector<int>(a.begin() + k, a.end()));
    }
    friend Poly operator+(const Poly a, const Poly &b) {
        vector<int> ret(max(a.size(), b.size()));
        for (int i = 0; i < int(ret.size()); ++i) {
            ret[i] = a[i] + b[i];
            if (ret[i] >= mod)
                ret[i] -= mod;
        }
        return Poly(ret);
    }
    friend Poly operator-(const Poly a, const Poly &b) {
        vector<int> ret(max(a.size(), b.size()));
        for (int i = 0; i < int(ret.size()); ++i) {
            ret[i] = a[i] - b[i];
            if (ret[i] < 0)
                ret[i] += mod;
        }
        return Poly(ret);
    }
    friend Poly operator*(Poly a, Poly b) {
        int sz = 1, tot = a.size() + b.size() - 1;
        while (sz < tot)
            sz *= 2;
        a.a.resize(sz);
        b.a.resize(sz);
        dft(a.a);
        dft(b.a);
        for (int i = 0; i < sz; ++i)
            a.a[i] = 1ll * a[i] * b[i] % mod;
        idft(a.a);
        return Poly(a.a);
    }
    Poly &operator+=(Poly b) {
        return (*this) = (*this) + b;
    }
    Poly &operator-=(Poly b) {
        return (*this) = (*this) - b;
    }
    Poly &operator*=(Poly b) {
        return (*this) = (*this) * b;
```

```
    }
    Poly deriv() const {
        if (a.empty())
            return Poly();
        vector<int> ret(size() - 1);
        for (int i = 0; i < size() - 1; ++i)
            ret[i] = 1ll * (i + 1) * a[i + 1] % mod;
        return Poly(ret);
    }
    Poly integr() const {
        if (a.empty())
            return Poly();
        vector<int> ret(size() + 1);
        for (int i = 0; i < size(); ++i)
            ret[i + 1] = 1ll * a[i] * qm(i + 1, mod - 2) % mod;
        return Poly(ret);
    }
    Poly inv(int m) const {
        Poly x(qm(a[0], mod - 2));
        int k = 1;
        while (k < m) {
            k *= 2;
            x = (x * (2 - modxk(k) * x)).modxk(k);
        }
        return x.modxk(m);
    }
    Poly log(int m) const {
        return (deriv() * inv(m)).integr().modxk(m);
    }
    Poly exp(int m) const {
        Poly x(1);
        int k = 1;
        while (k < m) {
            k *= 2;
            x = (x * (1 - x.log(k) + modxk(k))).modxk(k);
        }
        return x.modxk(m);
    }
    Poly sqrt(int m) const {
        Poly x(1);
        int k = 1;
        while (k < m) {
            k *= 2;
            x = (x + (modxk(k) * x.inv(k)).modxk(k)) * ((mod + 1) / 2);
        }
        return x.modxk(m);
    }
    Poly mulT(Poly b) const {
        if (b.size() == 0)
            return Poly();
        int n = b.size();
        reverse(b.a.begin(), b.a.end());
        return ((*this) * b).divxk(n - 1);
    }
    vector<int> eval(vector<int> x) const {
        if (size() == 0)
            return vector<int>(x.size(), 0);
        const int n = max(int(x.size()), size());
        vector<Poly> q(4 * n);
        vector<int> ans(x.size());
        x.resize(n);
        function<void(int, int, int)> build = [&](int p, int l, int r) {
            if (r - l == 1) {
```

```
                q[p] = vector<int>{1, (mod - x[l]) % mod};
            } else {
                int m = (l + r) / 2;
                build(2 * p, l, m);
                build(2 * p + 1, m, r);
                q[p] = q[2 * p] * q[2 * p + 1];
            }
        };
        build(1, 0, n);
        function<void(int, int, int, const Poly &)> work = [&](int p, int l, int
            r, const Poly &num) {
            if (r - l == 1) {
                if (l < int(ans.size()))
                    ans[l] = num[0];
            } else {
                int m = (l + r) / 2;
                work(2 * p, l, m, num.mulT(q[2 * p + 1]).modxk(m - l));
                work(2 * p + 1, m, r, num.mulT(q[2 * p]).modxk(r - m));
            }
        };
        work(1, 0, n, mulT(q[1].inv(n)));
        return ans;
    }
};
```

## 5.5 大数分解

```
ll gcd(ll a,ll b){
    ll t;
    while(b){t=a;a=b;b=t%b;}
    if(a>=0)return a;
    else return -a;
}
//ret=(a*b)%c  0<=a,b<2^63  0<c<2^62
ll mm(ll a,ll b,ll c){
    ll ret = 0;
    for(a%=c;b;b>>=1){
        if(b&1){ret+=a;if(ret>c)ret-=c;}
        a<<=1;if(a>c)a-=c;
    }
    return ret;
}
//ret = (a^b)%c
ll pm(ll a,ll b,ll c){
    ll ret = 1;
    for(a%=c;b;b>>=1,a=mm(a,a,c))
        if(b&1)ret=mm(ret,a,c);
    return ret;
}
// 通过 a^(n-1)=1(mod n)来判断 n 是不是素数
// n - 1 = x * 2
// t 中间使用二次判断
// 是合数返回 true, 不一定是合数返回 false
bool check(ll a,ll n,ll x,ll t){
    ll ret=pm(a,x,n), last=ret;
    for(int i = 1; i <= t; i++){
        ret = mm(ret,ret,n);
        if(ret == 1 && last != 1 && last != n-1)return true;//合数
        last = ret;
    }
    if(ret != 1)return true;
```

```
    else return false;
}
// 是素数返回 true,(可能是伪素数)
// 不是素数返回 false
bool Miller_Rabin(ll n){
    if( n < 2)return false;
    if( n == 2)return true;
    if( (n&1) == 0)return false;//偶数
    ll x=n-1, t=0;
    while( (x&1)==0 )x>>=1, t++;
    srand(time(NULL));/* *************** */
    for(int i = 0; i < 8; i++){//8~10次
        ll a = rand()%(n-1) + 1;
        if( check(a,n,x,t) )return false;
    }
    return true;
}
map<ll,int>fac;//质因素分解结果
//找出一个因子
ll pollard_rho(ll x,ll c){
    ll i = 1, k = 2;
    srand(time(NULL));
    ll x0 = rand()%(x-1) + 1;
    ll y = x0;
    while(1){
        i ++;
        x0 = (mm(x0,x0,x) + c)%x;
        ll d = gcd(y - x0,x);
        if( d != 1 && d != x)return d;
        if(y == x0)return x;
        if(i == k)y = x0,k += k;
    }
}
//将n素因子存入 fac. k 设置为 107 左右即可
void findfac(ll n,int k=107)
{
    if(n == 1)return;
    if(Miller_Rabin(n)){fac[n]++;return;}
    ll p = n;int c = k;
    while( p >= n)p = pollard_rho(p,c--);//值变化, 防止死循环
    findfac(p,k);findfac(n/p,k);
}
//POJ 1811
//给出一个N>1,素数输出"Prime",否则输出最小的素因子
int main(){
    int _;cin>>_;
    while(_--){
        ll n;cin>>n;
        fac.clear();
        findfac(n);
        if(fac.begin()->first==n)cout<<"Prime"<<endl;
        else cout<<fac.begin()->first<<endl;
    }
    return 0;
}
```

## 5.6 公式

调和级数公式：$f_n = ln(n) + \frac{1}{2n} + C;$（欧拉常数值：C 0.57721566490153286060651209）

狄尔沃斯定理 (Dilworth's theorem)：亦称偏序集分解定理，是关于偏序集的极大极小的定理，

该定理断言：对于任意有限偏序集，其最大反链中元素的数目必等于最小链划分中链的数目。此定理的对偶形式亦真，它断言：对于任意有限偏序集，其最长链中元素的数目必等于其最小反链划分中反链的数目

威尔逊定理：$p$ 是质数 $\Leftrightarrow (p-1)! \equiv -1(modp)$

$n$ 个小球每个小球发生的概率为 $p_i$，取到的小球数量平方的期望等同于取出一对的方案数的期望 $E(X^2) = \sum_{i!=j} p_i p_j + \sum_{i=1}^{n} p_i = (\sum_{i=1}^{n} p_i)^2 - \sum_{i=1}^{n} p_i^2 + \sum_{i=1}^{n} p_i$

# 6 动态规划

## 6.1 插头 dp

12*12 的网格图中走哈密顿回路的方案数

```
#include<bits/stdc++.h>
#define fi first
#define se second
typedef long long ll;
using namespace std;
int n,m,boa[14][14],endx,endy,bits[14];
ll Plug(){
    ll ret=0;
    unordered_map<int,ll>u,v;
    u[0]=1;
    for (int i=1; i<=n; i++){
        for(auto j:u)v[j.fi<<2]=j.se;
        swap(u,v);
        v.clear();

        for (int j=1; j<=m; j++){
            for(auto k:u){
                int s=k.fi;
                ll w=k.se;//get previous state&answer
                int is_d=s/bits[j]%4,is_r=s/bits[j-1]%4;//get current plugs

                if (!boa[i][j]){//case 0
                    if (!is_r && !is_d) v[s]+=w;
                }
                else if (!is_r && !is_d){//case 1
                    if (boa[i+1][j] && boa[i][j+1])
                        v[s+bits[j-1]+2*bits[j]]+=w;
                }
                else if (is_r && !is_d){//case 2
                    if (boa[i+1][j]) v[s]+=w;//go down
                    if (boa[i][j+1]) v[s-is_r*bits[j-1]+is_r*bits[j]]+=w;//go
                        right
                }
                else if (!is_r && is_d){//case 3
                    if (boa[i][j+1]) v[s]+=w;//go right
                    if (boa[i+1][j]) v[s-is_d*bits[j]+is_d*bits[j-1]]+=w;//go
                        down
                }
                else if (is_r==1 && is_d==1){//case 4
                    int cnt=1;
                    for (int l=j+1; l<=m; l++){
                        if (s/bits[l]%4==1) cnt++;
                        else if (s/bits[l]%4==2) cnt--;
                        if (!cnt){
```

```
                            v[s-bits[l]-bits[j]-bits[j-1]]+=w;
                            break;
                        }
                    }
                }
                else if (is_r==2 && is_d==2){//case 5
                    int cnt=1;
                    for (int l=j-2; l>=0; l--){
                        if (s/bits[l]%4==1) cnt--;
                        else if (s/bits[l]%4==2) cnt++;
                        if (!cnt){
                            v[s-2*bits[j]-2*bits[j-1]+bits[l]]+=w;
                            break;
                        }
                    }
                }
                else if (is_r==2 && is_d==1)//case 6
                    v[s-2*bits[j-1]-bits[j]]+=w;
                else if (is_r==1 && is_d==2)//case 7
                    if (i==endx && j==endy) ret+=w;
            }
            swap(u,v);
            v.clear();
        }
    }
    return ret;
}
int main(){
    for(int i=0;i<14;i++)bits[i]=1<<(i<<1);
    cin>>n>>m;
    for(int i=1;i<=n;i++){
        char str[14];cin>>str+1;
        for(int j=1;j<=m;j++){
            if(str[j]=='*')boa[i][j]=0;
            else boa[i][j]=1,endx=i,endy=j;
        }
    }
    cout<<Plug();
    return 0;
}
```

## 6.2 cdq 分治

```
#include <bits/stdc++.h>
#define lb(x) (x&-x)
#define mid (l+r>>1)
using namespace std;
const int N=1e5+10;
int n,m,c[N<<1],ans[N],cnt;

struct A{
    int a,b,c,w,f;
}e[N],t[N];

void add(int x,int y){
    for(;x<=m;x+=lb(x)) c[x]+=y;
}
int sum(int x){
    int ret=0;
    for(;x;x-=lb(x)) ret+=c[x];
    return ret;
```

```
}

void CDQ(int l,int r){
    if(l==r) return ;
    CDQ(l,mid);CDQ(mid+1,r);
    int p=l,q=mid+1,tot=l;
    while(p<=mid&&q<=r){
        if(e[p].b<=e[q].b) add(e[p].c,e[p].w),t[tot++]=e[p++];
        else e[q].f+=sum(e[q].c),t[tot++]=e[q++];
    }
    while(p<=mid) add(e[p].c,e[p].w),t[tot++]=e[p++];
    while(q<=r) e[q].f+=sum(e[q].c),t[tot++]=e[q++];
    for(int i=l;i<=mid;i++) add(e[i].c,-e[i].w);
    for(int i=l;i<=r;i++) e[i]=t[i];
}

int main()
{
    cin>>n>>m;
    for(int i=1;i<=n;i++)
        cin>>e[i].a>>e[i].b>>e[i].c,e[i].w=1;
    sort(e+1,e+n+1,[](A a,A b){
         if(a.a!=b.a)return a.a<b.a;
         else if(a.b!=b.b)return a.b<b.b;
         else return a.c<b.c;
    });
    cnt=1;
    for(int i=2;i<=n;i++){
        if(e[i].a==e[cnt].a&&e[i].b==e[cnt].b&&e[i].c==e[cnt].c) e[cnt].w++;
        else e[++cnt]=e[i];
    }
    CDQ(1,cnt);
    for(int i=1;i<=cnt;i++) ans[e[i].f+e[i].w-1]+=e[i].w;
    for(int i=0;i<n;i++) cout<<ans[i]<<endl;
    return 0;
}
```

# 7 计算几何

## 7.1 Point

```
const double PI = acos(-1);
const double eps = 1e-8;
int sgn(double x) {return x <= -eps ? -1 : (x < eps ? 0 : 1);}
inline double sqr(double x) {return x * x;}
/*************************************************************/
struct Point
{
    double x, y;
    void i(){scanf("%lf%lf",&x,&y);}
    void o(){printf("%.6lf %.6lf\n",x,y);}
    Point() {}
    Point(double _x, double _y): x(_x), y(_y) {}
    bool operator==(Point b) {return !sgn(x - b.x) && !sgn(y - b.y);}
    Point operator+(Point b) {return Point(x + b.x, y + b.y);}
    Point operator-(Point b) {return Point(x - b.x, y - b.y);}
    Point operator*(double k) {return Point(x * k, y * k);}
    Point operator/(double k) {return Point(x / k, y / k);}
    bool operator<(Point b)const{return sgn(x-b.x) == 0 ? sgn(y-b.y)<0 : x<b.x;}
    double operator^(Point b){return x*b.y-y*b.x;}//叉积，逆时针为正
    double operator*(Point b){return x*b.x+y*b.y;}//点积，求投影长度
```

```
21      double len2() {return x * x + y * y;}//到原点距离的平方
22      double len() {return sqrt(len2());}//到原点的距离
23      double dis(Point p) {return (*this - p).len();}
24      double rad(Point a, Point b)
25      {//计算pa和pb的夹角（弧度制）
26          Point p = *this;
27          return fabs(atan2(fabs((a - p) ^ (b - p)), (a - p) * (b - p) ));
28      }
29      Point trunc(double r)
30      {//更改为长度为r的向量，可以负数改变方向
31          double l = len();
32          if(!sgn(l))return *this;
33          r /= l;
34          return Point(x * r, y * r);
35      }
36      Point rotl() {return Point(-y, x);}//逆时针旋转 90 度
37      Point rotr() {return Point(y, -x);}//顺时针旋转 90 度
38      Point rotate(Point p, double angle)
39      {//绕着p点逆时针旋转angle（弧度制）
40          Point v = (*this) - p;
41          double c = cos(angle), s = sin(angle);
42          return Point(p.x + v.x * c - v.y * s, p.y + v.x * s + v.y * c);
43      }
44      double cross(Point b, Point c) {return (b - *this) ^ (c - *this);}//ab^ac
45      double dot(Point b, Point c) {return (b - *this) * (c - *this);}//ab*ac
46  };
```

## 7.2 Line

```
1   struct Line{
2       Point u,v;//两点式存储，可以表示直线或线段
3       void i(){u.i(),v.i();}
4       Line() {}
5       Line(Point _u, Point _v): u(_u), v(_v) {}
6       Line(Point p, double angle)
7       {//从p开始的，弧度制为angle[0,PI)的线
8           u = p;
9           if(sgn(angle - PI / 2) == 0)v = (u + Point(0, 1));
10          else v = (u + Point(1, tan(angle)));
11      }
12      Line(double a, double b, double c)
13      {//ax+by+c=0
14          if(sgn(a) == 0)u = Point(0, -c / b),v = Point(1, -c / b);
15          else if(sgn(b) == 0)u = Point(-c / a, 0),v = Point(-c / a, 1);
16          else u = Point(0, -c / b),v = Point(1, (-c - a) / b);
17      }
18      double len() {return u.dis(v);}//求线段长度
19      double angle()
20      {//返回直线倾斜角[0,PI)
21          double k = atan2(v.y - u.y, v.x - u.x);
22          if(sgn(k) < 0)k += PI;
23          if(sgn(k - PI) == 0)k -= PI;
24          return k;
25      }
26      //点p是否在直线上
27      bool pointonline(Point p){return sgn((p - u) ^ (v - u)) == 0;}
28      //点p是否在线段上
29      bool pointonseg(Point p){return sgn((p - u) ^ (v - u)) == 0 && sgn((p - u) *
            (p - v)) <= 0;}
30      //两直线（线段）是否平行或重合
31      bool parallel(Line t){return sgn((v - u) ^ (t.v-t.u)) == 0;}
```

```
32      int segcrosseg(Line t)
33      {//两线段相交判断(0-不相交)(1-非规范相交，部分重合或有顶点在线上)(2-规范相交)
34          int d1 = sgn((v - u) ^ (t.u - u));
35          int d2 = sgn((v - u) ^ (t.v-u));
36          int d3 = sgn((t.v - t.u) ^ (u - t.u));
37          int d4 = sgn((t.v - t.u) ^ (v - t.u));
38          if((d1 ^ d2) == -2 && (d3 ^ d4) == -2)return 2;
39          return (d1 == 0 && sgn((t.u - u) * (t.u - v)) <= 0) ||
40                 (d2 == 0 && sgn((t.v - u) * (t.v - v)) <= 0) ||
41                 (d3 == 0 && sgn((u - t.u) * (u - t.v)) <= 0) ||
42                 (d4 == 0 && sgn((v - t.u) * (v - t.v)) <= 0);
43      }
44      int linecrossseg(Line t)
45      {//该直线和线段v相交判断(0-不相交)(1-非规范相交，部分重合或有顶点在线上)(2-规范相交)
46          int d1 = sgn((v - u) ^ (t.u - u));
47          int d2 = sgn((v - u) ^ (t.v - u));
48          if(d1 * d2 < 0)return 2;
49          return (d1 == 0 || d2 == 0);
50      }
51      int linecrossline(Line t)
52      {//两直线关系(0-平行)(1-重合)(2-相交)
53          if(this->parallel(t))return t.pointonline(u);
54          else return 2;
55      }
56      Point crosspoint(Line t)
57      {//求两直线的交点，要求保证两直线不平行或重合
58          double a1 = (t.v - t.u) ^ (u - t.u);
59          double a2 = (t.v - t.u) ^ (v - t.u);
60          return Point((u.x * a2 - v.x * a1) / (a2 - a1), (u.y * a2 - v.y * a1) / (a2 - a1));
61      }
62      //点到直线的距离
63      double dispointtoline(Point p){return fabs((p - u) ^ (v - u)) / len();}
64      double dispointtoseg(Point p)
65      {//点到线段的距离（直线距离或者到端点距离
66          if(sgn((p - u) * (v - u)) < 0 || sgn((p - v) * (u - v)) < 0)
67              return min(p.dis(u), p.dis(v));
68          return dispointtoline(p);
69      }
70      double dissegtoseg(Line t)
71      {//返回线段到线段的距离，需要保证不相交
72          return min(min(dispointtoseg(t.u), dispointtoseg(t.v)),
73                     min(t.dispointtoseg(u), t.dispointtoseg(v)));
74      }
75      //返回p在直线上的投影
76      Point lineprog(Point p){return u + ( ((v - u) * ((v - u) * (p - u))) / ((v - u).len2()) );}
77      Point symmetrypoint(Point p)
78      {//返回点p关于直线的对称点
79          Point q = lineprog(p);
80          return Point(2 * q.x - p.x, 2 * q.y - p.y);
81      }
82  };
```

## 7.3 Cir

```
1   struct Cir{
2       Point o;//圆心
3       double r;//半径
```

```
4       Cir() {}
5       Cir(Point _o, double _r){o = _o;r = _r;}
6       Cir(double x, double y, double _r){o = Point(x, y);r = _r;}
7       Cir(Point a, Point b, Point c)
8       {//三角形的外接圆，利用两条边的中垂线得到圆心
9           Line u = Line((a + b) / 2, ((a + b) / 2) + ((b - a).rotl()));
10          Line v = Line((b + c) / 2, ((b + c) / 2) + ((c - b).rotl()));
11          o = u.crosspoint(v);
12          r = o.dis(a);
13      }
14      Cir(Point a, Point b, Point c, bool t)
15      {//三角形的内切圆，参数 bool t 没有作用，只是为了和上面外接圆函数区别
16          Line u, v;
17          double m = atan2(b.y - a.y, b.x - a.x), n = atan2(c.y - a.y, c.x - a.x);
18          u.u = a;
19          u.v = u.u + Point(cos((n + m) / 2), sin((n + m) / 2));
20          v.u = b;
21          m = atan2(a.y - b.y, a.x - b.x), n = atan2(c.y - b.y, c.x - b.x);
22          v.v = v.u + Point(cos((n + m) / 2), sin((n + m) / 2));
23          o = u.crosspoint(v);
24          r = Line(a, b).dispointtoseg(o);
25      }
26      bool operator == (Cir v){return o == v.o && sgn(r - v.r) == 0;}
27      double area() {return PI * r * r;}//面积
28      double circumference() {return 2 * PI * r;}//周长
29      int relation(Point b)
30      {//点和圆的关系(0-圆外)(1-圆上)(2-圆内)
31          double dst = b.dis(o);
32          if(sgn(dst - r) < 0)return 2;
33          else if(sgn(dst - r) == 0)return 1;
34          return 0;
35      }
36      int relationseg(Line v)
37      {//线段和圆的关系，比较的是圆心到线段的距离和半径的关系
38          double dst = v.dispointtoseg(o);
39          if(sgn(dst - r) < 0)return 2;
40          else if(sgn(dst - r) == 0) return 1;
41          return 0;
42      }
43      int relationLine(Line v)
44      {//直线和圆的关系，比较的是圆心到直线的距离和半径的关系
45          double dst = v.dispointtoline(o);
46          if(sgn(dst - r) < 0)return 2;
47          else if(sgn(dst - r) == 0)return 1;
48          return 0;
49      }
50      int relationCir(Cir v)
51      {//两圆的关系(1-内含)(2-内切)(3-相交)(4-外切)(5-相离)
52          double d = o.dis(v.o);
53          if(sgn(d - r - v.r) > 0)return 5;
54          if(sgn(d - r - v.r) == 0)return 4;
55          double l = fabs(r - v.r);
56          if(sgn(d - r - v.r) < 0 && sgn(d - l) > 0)return 3;
57          if(sgn(d - l) == 0)return 2;
58          return 1;
59      }
60      int PointCrossCir(Cir v, Point &p1, Point &p2)
61      {//求两个圆的交点(0-表示没有交点)(1-是一个交点)(2-是两个交点)
62          int rel = relationCir(v);
63          if(rel == 1 || rel == 5)return 0;
64          double d = o.dis(v.o);
65          double l = (d * d + r * r - v.r * v.r) / (2 * d);
66          double h = sqrt(r * r - l * l);
```

```
67          Point tmp = o + (v.o - o).trunc(l);
68          p1 = tmp + ((v.o - o).rotl().trunc(h));
69          p2 = tmp + ((v.o - o).rotr().trunc(h));
70          if(rel == 2 || rel == 4)return 1;
71          return 2;
72      }
73      int PointCrossLine(Line v, Point &p1, Point &p2)
74      {//求直线和圆的交点，返回交点个数
75          if(!(*this).relationLine(v))return 0;
76          Point a = v.lineprog(o);
77          double d = v.dispointtoline(o);
78          d = sqrt(r * r - d * d);
79          if(sgn(d) == 0){p1 = a;p2 = a;return 1;}
80          p1 = a + (v.v-v.u).trunc(d);
81          p2 = a - (v.v-v.u).trunc(d);
82          return 2;
83      }
84      int tangentline(Point q, Line &u, Line &v)
85      {//过一点作圆的切线
86          int x = relation(q);
87          if(x == 2)return 0;
88          if(x == 1)
89          {
90              u = Line(q, q + (q - o).rotl());
91              v = u;
92              return 1;
93          }
94          double d = o.dis(q);
95          double l = r * r / d;
96          double h = sqrt(r * r - l * l);
97          u = Line(q, o + ((q - o).trunc(l) + (q - o).rotl().trunc(h)));
98          v = Line(q, o + ((q - o).trunc(l) + (q - o).rotr().trunc(h)));
99          return 2;
100     }
101     double areaCir(Cir v)
102     {//求两圆相交的面积
103         int rel = relationCir(v);
104         if(rel >= 4)return 0.0;
105         if(rel <= 2)return min(area(), v.area());
106         double d = o.dis(v.o);
107         double hf = (r + v.r + d) / 2.0;
108         double ss = 2 * sqrt(hf * (hf - r) * (hf - v.r) * (hf - d));
109         double a1 = acos((r * r + d * d - v.r * v.r) / (2.0 * r * d));
110         a1 = a1 * r * r;
111         double a2 = acos((v.r * v.r + d * d - r * r) / (2.0 * v.r * d));
112         a2 = a2 * v.r * v.r;
113         return a1 + a2 - ss;
114     }
115     double areatriangle(Point a, Point b)
116     {//求圆和三角形 pab 的相交面积
117         if(sgn((o - a) ^ (o - b)) == 0)return 0;
118         Point q[5];
119         int len = 0;
120         q[len++] = a;
121         Line l(a, b);
122         Point p1, p2;
123         if(PointCrossLine(l, q[1], q[2]) == 2)
124         {
125             if(sgn((a - q[1]) * (b - q[1])) < 0)q[len++] = q[1];
126             if(sgn((a - q[2]) * (b - q[2])) < 0)q[len++] = q[2];
127         }
128         q[len++] = b;
129         if(len==4 && sgn((q[0]-q[1])*(q[2]-q[1]))>0)swap(q[1], q[2]);
```

```
130         double ret = 0;
131         for(int i = 0; i < len - 1; i++)
132         {
133             if(relation(q[i]) == 0 || relation(q[i + 1]) == 0)
134             {
135                 double arg = o.rad(q[i], q[i + 1]);
136                 ret += r * r * arg / 2.0;
137             }
138             else ret += fabs((q[i] - o) ^ (q[i + 1] - o)) / 2.0;
139         }
140         return ret;
141     }
142 };
143 int getCir(Point a, Point b, double r, Cir &c1, Cir &c2)
144 {//经过点a,b、半径r
145     Cir x(a, r), y(b, r);
146     int t = x.PointCrossCir(y, c1.o, c2.o);
147     if(!t)return 0;
148     c1.r = c2.r = r;
149     return t;
150 }
151 int getCir(Line u, Point q, double r, Cir &c1, Cir &c2)
152 {//与直线l相切；经过点q；半径r
153     double dis = u.dispointtoline(q);
154     if(sgn(dis - r * 2) > 0)return 0;
155     if(sgn(dis) == 0)
156     {
157         c1.o = q + ((u.v-u.u).rotl().trunc(r));
158         c2.o = q + ((u.v-u.u).rotr().trunc(r));
159         c1.r = c2.r = r;
160         return 2;
161     }
162     Line u1=Line((u.u+(u.v-u.u).rotl().trunc(r)), (u.v+(u.v-u.u).rotl().trunc(r)
        ));
163     Line u2=Line((u.u+(u.v-u.u).rotr().trunc(r)), (u.v+(u.v-u.u).rotr().trunc(r)
        ));
164     Cir cc = Cir(q, r);
165     Point p1, p2;
166     if(!cc.PointCrossLine(u1, p1, p2))cc.PointCrossLine(u2, p1, p2);
167     c1 = Cir(p1, r);
168     if(p1 == p2){c2 = c1;return 1;}
169     c2 = Cir(p2, r);
170     return 2;
171 }
172 int getCir(Line u, Line v, double r, Cir &c1, Cir &c2, Cir &c3, Cir &c4)
173 {//与直线 u,v 相切，半径r
174     if(u.parallel(v))return 0;//两直线平行
175     Line u1 = Line(u.u+(u.v-u.u).rotl().trunc(r), u.v+(u.v-u.u).rotl().trunc(r))
        ;
176     Line u2 = Line(u.u+(u.v-u.u).rotr().trunc(r), u.v+(u.v-u.u).rotr().trunc(r))
        ;
177     Line v1 = Line(v.u+(v.v-v.u).rotl().trunc(r), v.v+(v.v-v.u).rotl().trunc(r))
        ;
178     Line v2 = Line(v.u+(v.v-v.u).rotr().trunc(r), v.v+(v.v-v.u).rotr().trunc(r))
        ;
179     c1.r = c2.r = c3.r = c4.r = r;
180     c1.o = u1.crosspoint(v1);
181     c2.o = u1.crosspoint(v2);
182     c3.o = u2.crosspoint(v1);
183     c4.o = u2.crosspoint(v2);
184     return 4;
185 }
186 int getCir(Cir cx, Cir cy, double r, Cir &c1, Cir &c2)
```

```
187 {//与不相交圆 cx,cy 相切, 半径r
188     Cir x(cx.o, r + cx.r), y(cy.o, r + cy.r);
189     int t = x.PointCrossCir(y, c1.o, c2.o);
190     if(!t)return 0;
191     c1.r = c2.r = r;
192     return t;
193 }
```

## 7.4 Pol

```
1  struct Pol{
2      vector<Point>p;//标准顺序是从左下角逆时针转一圈
3      Pol(int n=0){p.resize(n);}
4      void getconvex()
5      {//把Pol变成凸包, 返回标准顺序
6          sort(p.begin(),p.end());//把点排序后比较相邻三点间的叉积关系
7          if(p[0]==p.back()){p.resize(1);return;}//只有一个点的情况
8          vector<Point>u,d;//带左右端点的上下凸包
9          for(int i=0; i<p.size(); i++)//下方的凸包
10         {
11             while(d.size()>=2 && sgn(d[d.size()-2].cross(d.back(),p[i])<=0))d.
                   pop_back();
12             d.push_back(p[i]);
13         }
14         for(int i=p.size()-1; ~i; i--)//上方的凸包
15         {
16             while(u.size()>=2 && sgn(u[u.size()-2].cross(u.back(),p[i])<=0))u.
                   pop_back();
17             u.push_back(p[i]);
18         }
19         p=d;
20         for(int i=1;i<(int)u.size()-1;i++)p.push_back(u[i]);
21     }
22     vector<Line>l;
23     void getline()
24     {//得到多边形的每一条边
25         l.resize(p.size());
26         for(int i=0;i<l.size();i++)l[i]=Line(p[i], p[(i+1)%p.size()]);
27     }
28     int relationpoint(Point q)
29     {//判断点和任意多边形的关系(0-外部)(1-内部)(2-边上)(3-点上)
30         for(auto i:p)if(i==q)return 3;
31         getline();
32         for(auto i:l)if(i.pointonseg(q))return 2;
33         int cnt = 0;
34         for(int i = 0; i < p.size(); i++)
35         {
36             int j = (i + 1) % p.size();
37             int k = p[j].cross(q,p[i]);
38             int u = sgn(p[i].y - q.y);
39             int v = sgn(p[j].y - q.y);
40             if(k > 0 && u < 0 && v >= 0)cnt++;
41             if(k < 0 && v < 0 && u >= 0)cnt--;
42         }
43         return cnt != 0;
44     }
45     void cutconvex(Line t)
46     {//返回直线 t 切割凸多边形左侧, 直线方向是u->v
47         Pol ret;
48         for(int i = 0; i < p.size(); i++)
49         {
```

```
50              int d1 = sgn(t.u.cross(t.v,p[i]));
51              int d2 = sgn(t.u.cross(t.v,p[(i+1)%p.size()]));
52              if(d1>=0)ret.p.push_back(p[i]);
53              if(d1*d2<0)ret.p.push_back(t.crosspoint(Line(p[i], p[(i+1)%p.size()
                    ])));
54          }
55          *this=ret;
56      }
57      double circum()
58      {//得到周长
59          double sum = 0;
60          for(int i = 0; i < p.size(); i++)
61              sum += p[i].dis(p[(i + 1) % p.size()]);
62          return sum;
63      }
64      double area()
65      {//得到面积
66          double sum = 0;
67          for(int i = 0; i < p.size(); i++)
68              sum += (p[i] ^ p[(i + 1) % p.size()]);
69          return fabs(sum) / 2;
70      }
71      bool getdir()
72      {//得到方向，1 表示逆时针，0 表示顺时针
73          double sum = 0;
74          for(int i = 0; i < p.size(); i++)
75              sum += p[i] ^ p[(i + 1) % p.size()];
76          return sgn(sum) > 0;
77      }
78      Point getbarycentre()
79      {//得到重心
80          Point ret(0, 0);
81          double area = 0;
82          for(int i = 1; i < p.size() - 1; i++)
83          {
84              double tmp = p[0].cross(p[i],p[i+1]);
85              if(!sgn(tmp))continue;
86              area += tmp;
87              ret.x += (p[0].x + p[i].x + p[i + 1].x) / 3 * tmp;
88              ret.y += (p[0].y + p[i].y + p[i + 1].y) / 3 * tmp;
89          }
90          if(sgn(area))ret = ret / area;
91          return ret;
92      }
93      double areaCir(Cir c)
94      {//多边形和圆交的面积，可以处理凹多边形，顺时针逆时针都可以
95          double ret = 0;
96          for(int i = 0; i < p.size(); i++)
97          {
98              int j = (i + 1) % p.size();
99              if(sgn(p[j]-c.o ^ p[i]-c.o) >= 0)ret += c.areatriangle(p[i], p[j]);
100             else ret -= c.areatriangle(p[i], p[j]);
101         }
102         return fabs(ret);
103     }
104     int relationCir(Cir c)
105     {//多边形和圆关系(0-圆完全在多边形内)(1-圆在多边形里面，碰到了多边形边界)(2-
            其它)
106         getline();
107         int ret = 0;
108         if(relationpoint(c.o) != 1)return 2;//圆心不在内部
109         for(auto i:l)
110         {
```

```
111             if(c.relationseg(i) == 2)return 2;
112             if(c.relationseg(i) == 1)ret = 1;
113         }
114         return ret;
115     }
116     double minRectangleCover()
117     {//最小矩形面积覆盖，此时必须是凸包（而且是逆时针顺序）
118         if(p.size() < 3)return 0;
119         int n=p.size();
120         p.push_back(p[0]);
121         double ret = -1;
122         int r = 1, w = 1, q;
123         for(int i = 0; i < n; i++)
124         {
125             while( sgn( p[i].cross(p[i + 1], p[r + 1]) - p[i].cross(p[i + 1], p[
                    r]) ) >= 0)
126                 r = (r + 1) % n;//卡出离边 A.p[i] - A.p[i+1] 最远的点
127             while(sgn( p[i].dot(p[i + 1], p[w + 1]) - p[i].dot(p[i + 1], p[w]) )
                     >= 0 )
128                 w = (w + 1) % n;//卡出 A.p[i] - A.p[i+1] 方向上正向 n 最远的点
129             if(!i)q = w;
130             while(sgn( p[i].dot(p[i + 1], p[q + 1]) - p[i].dot(p[i + 1], p[q]))
                    <= 0)
131                 q = (q + 1) % n;//卡出 A.p[i] - A.p[i+1] 方向上负向最远的点
132             double d = (p[i] - p[i + 1]).len2();
133             double tmp=p[i].cross(p[i+1],p[r])*(p[i].dot(p[i+1],p[w])-p[i].dot(p
                    [i+1],p[q]))/d;
134             if(ret < 0 || ret > tmp)ret = tmp;
135         }
136         p.pop_back();
137         return ret;
138     }
139 };
```

## 7.5 3D

```
1  const double PI = acos(-1);
2  const double eps = 1e-8;
3  int sgn(double x) {return x <= -eps ? -1 : (x < eps ? 0 : 1);}
4  struct Point3
5  {
6      double x, y, z;
7      void i(){scanf("%lf%lf%lf",&x,&y,&z);}
8      void o(){printf("%lf %lf %lf\n",x,y,z);}
9      Point3(double _x = 0, double _y = 0, double _z = 0): x(_x), y(_y), z(_z) {}
10     bool operator ==(Point3 b) {return !sgn(x - b.x) && !sgn(y - b.y) && !sgn(z
           - b.z);}
11     bool operator <(Point3 b) {return !sgn(x - b.x) ? (!sgn(y - b.y) ? sgn(z - b
           .z) < 0 : y < b.y) : x < b.x;}
12     Point3 operator -(Point3 b) {return Point3(x - b.x, y - b.y, z - b.z);}
13     Point3 operator +(Point3 b) {return Point3(x + b.x, y + b.y, z + b.z);}
14     Point3 operator *(double k) {return Point3(x * k, y * k, z * k);}
15     Point3 operator /(double k) {return Point3(x / k, y / k, z / k);}
16     double operator *(Point3 b) {return x * b.x + y * b.y + z * b.z;}//点乘
17     Point3 operator ^(Point3 b) {return Point3(y * b.z - z * b.y, z * b.x - x *
           b.z, x * b.y - y * b.x);}//叉乘
18     double len2() {return x * x + y * y + z * z;}
19     double len() {return sqrt(len2());}
20     double dis(Point3 b) {return (*this - b).len();}
21     Point3 cross(Point3 b,Point3 c) {return (b-*this)^(c-*this);}//ab^ac
22     double rad(Point3 a, Point3 b)
```

```
23      {//计算pa和pb的夹角（弧度制）
24          Point3 p = *this;
25          return acos( ((a - p) * (b - p)) / (a.dis(p) * b.dis(p)) );
26      }
27      Point3 trunc(double r)
28      {//更改为长度为r的向量，可以负数改变方向
29          double l = len();
30          if(!sgn(l)) return *this;
31          r /= l;
32          return Point3(x * r, y * r, z * r);
33      }
34  };
35  struct Line3
36  {
37      Point3 s, e;
38      Line3() {}
39      Line3(Point3 _s, Point3 _e): s(_s), e(_e) {}
40      bool operator ==(Line3 v) {return (s == v.s) && (e == v.e);}
41      double length() {return s.dis(e);}
42      //点到直线距离
43      double dispointtoline(Point3 p){return ((e - s) ^ (p - s)).len() / s.dis(e)
            ;}
44      double dispointtoseg(Point3 p)
45      {//点到线段距离
46          if(sgn((p - s) * (e - s)) < 0 || sgn((p - e) * (s - e)) < 0)
47              return min(p.dis(s), e.dis(p));
48          return dispointtoline(p);
49      }
50      //返回点 p 在直线上的投影
51      Point3 lineprog(Point3 p){return s + ( ((e - s) * ((e - s) * (p - s))) / ((e
            - s).len2()) );}
52      Point3 rotate(Point3 p, double ang)
53      {//p 绕此向量逆时针 arg 角度
54          if(sgn(((s - p) ^ (e - p)).len()) == 0) return p;
55          Point3 f1 = (e - s) ^ (p - s);
56          Point3 f2 = (e - s) ^ (f1);
57          double len = ((s - p) ^ (e - p)).len() / s.dis(e);
58          f1 = f1.trunc(len);
59          f2 = f2.trunc(len);
60          Point3 h = p + f2;
61          Point3 pp = h + f1;
62          return h + ((p - h) * cos(ang)) + ((pp - h) * sin(ang));
63      }
64      //点在直线上
65      bool pointonseg(Point3 p){return !sgn((s-p^e-p).len())&&!sgn((s-p)*(e-p));}
66  };
67  struct Plane
68  {
69      Point3 a, b, c, o; //平面上的三个点，以及法向量
70      Plane() {}
71      Plane(Point3 _a, Point3 _b, Point3 _c):a(_a),b(_b),c(_c){o = pvec();}
72      Point3 pvec() {return (b - a) ^ (c - a);}
73      Plane(double _a, double _b, double _c, double _d)
74      {//ax+by+cz+d = 0
75          o = Point3(_a, _b, _c);
76          if(sgn(_a) != 0)a = Point3((-_d - _c - _b) / _a, 1, 1);
77          else if(sgn(_b) != 0)a = Point3(1, (-_d - _c - _a) / _b, 1);
78          else if(sgn(_c) != 0)a = Point3(1, 1, (-_d - _a - _b) / _c);
79      }
80      //点在平面上的判断
81      bool pointonplane(Point3 p){return sgn((p - a) * o) == 0;}
82      //两平面夹角
83      double angleplane(Plane f){return acos(o * f.o) / (o.len() * f.o.len());}
```

```
84      int crossline(Line3 u, Point3 &p)
85      {//平面和直线的交点，返回值是交点个数
86          double x = o * (u.e-a);
87          double y = o * (u.s - a);
88          double d = x - y;
89          if(sgn(d) == 0)return 0;
90          p = ((u.s * x) - (u.e * y)) / d;
91          return 1;
92      }
93      Point3 pointtoplane(Point3 p)
94      {//点到平面最近点（也就是投影）
95          Line3 u = Line3(p, p + o);
96          crossline(u, p);
97          return p;
98      }
99      int crossplane(Plane f, Line3 &u)
100     {//平面和平面的交线
101         Point3 oo = o ^ f.o;
102         Point3 v = o ^ oo;
103         double d = fabs(f.o * v);
104         if(sgn(d) == 0) return 0;
105         Point3 q = a + (v * (f.o * (f.a - a)) / d);
106         u = Line3(q, q + oo);
107         return 1;
108     }
109 };
110
111 struct Pol
112 {
113     struct face
114     {
115         int a,b,c;//表示凸包一个面上的三个点的编号
116         bool ok;//表示该面是否属于最终的凸包上的面
117     };
118     int n;//初始顶点数
119     Point3 P[N];//多面体所有的点
120     int num;//凸包表面的三角形数
121     face F[8*N];//凸包表面的三角形，法向量朝外，下标从0开始
122     int g[N][N];//方向为i->j的边所在的表面三角形的编号
123     //2*三角形面积
124     double area(Point3 a,Point3 b,Point3 c) {return a.cross(b,c).len();}
125     //6*四面体有向面积
126     double volume(Point3 a,Point3 b,Point3 c,Point3 d) {return a.cross(b,c)*(d-a
        );}
127     //点p和面f形成的四面体的有向面积，当点和面同向时为正
128     double dblcmp(Point3 p,face f){return volume(P[f.a],P[f.b],P[f.c],p);}
129     void deal(int p,int a,int b)
130     {
131         int f = g[a][b];
132         face add;
133         if(F[f].ok)
134         {
135             if(dblcmp(P[p],F[f]) > eps)dfs(p,f);
136             else
137             {
138                 add={b,a,p,true};
139                 g[p][b] = g[a][p] = g[b][a] = num;
140                 F[num++] = add;
141             }
142         }
143     }
144     void dfs(int p,int now)
145     {//递归搜索所有应该从凸包内删除的面
```

```
146         F[now].ok = false;
147         deal(p,F[now].b,F[now].a);
148         deal(p,F[now].c,F[now].b);
149         deal(p,F[now].a,F[now].c);
150     }
151     void getconvex()
152     {//增量法求三维凸包
153 //******************此段是为了保证前四个点不共面******************
154         bool flag = true;
155         for(int i = 1; i < n; i++)if(!(P[0] == P[i]))
156             {swap(P[1],P[i]);flag = false;break;}
157         if(flag)return;flag = true;
158         for(int i = 2; i < n; i++)if(area(P[0],P[1],P[i]) > eps)
159             {swap(P[2],P[i]);flag = false;break;}
160         if(flag)return;flag = true;
161         for(int i = 3; i < n; i++)if(fabs(volume(P[0],P[1],P[2],P[i])) > eps)
162             {swap(P[3],P[i]);flag = false;break;}
163         if(flag)return;
164 //**************************************************************
165
166         num = 0;
167         face add;
168         for(int i = 0; i < 4; i++)
169         {
170             add={(i+1)%4,(i+2)%4,(i+3)%4,true};
171             if(dblcmp(P[i],add) > 0)swap(add.b,add.c);
172             g[add.a][add.b] = g[add.b][add.c] = g[add.c][add.a] = num;
173             F[num++] = add;
174         }
175         for(int i = 4; i < n; i++)for(int j = 0; j < num; j++)
176             if(F[j].ok && dblcmp(P[i],F[j]) > eps){dfs(i,j);break;}
177         int tmp = num;
178         num = 0;
179         for(int i = 0; i < tmp; i++)
180             if(F[i].ok)F[num++] = F[i];
181     }
182     double area()
183     {//多边体的表面积
184         if(n == 3)return area(P[0],P[1],P[2])/2;
185         double ret = 0;
186         for(int i = 0; i < num; i++)
187             ret += area(P[F[i].a],P[F[i].b],P[F[i].c]);
188         return ret/2.0;
189     }
190     double volume()
191     {//多边体的体积
192         double ret = 0;
193         Point3 tmp = Point3(0,0,0);
194         for(int i = 0; i < num; i++)
195             ret += volume(tmp,P[F[i].a],P[F[i].b],P[F[i].c]);
196         return fabs(ret/6);
197     }
198     int polygon()
199     {//表面多边形个数，判断多边形F[i]和F[j]是否在同一平面
200         int ret = 0;
201         for(int i = 0; i < num; i++)
202         {
203             bool f=1;
204             for(int j = 0; j < i; j++)
205                 if(!sgn(dblcmp(P[F[i].a],F[j]))&&
206                    !sgn(dblcmp(P[F[i].b],F[j]))&&
207                    !sgn(dblcmp(P[F[i].c],F[j])))f=0;
208             ret+=f;
```

```
        }
        return ret;
    }
    Point3 barycenter()
    {//多面体的重心
        Point3 ret = Point3(0,0,0);
        Point3 o = Point3(0,0,0);
        double all = 0;
        for(int i = 0; i < num; i++)
        {
            double vol = volume(o,P[F[i].a],P[F[i].b],P[F[i].c]);
            ret = ret + (((o+P[F[i].a]+P[F[i].b]+P[F[i].c])/4)*vol);
            all += vol;
        }
        ret = ret/all;
        return ret;
    }
    double ptoface(Point3 p,int i)
    {//点到面的距离
        double tmp1 = fabs(volume(P[F[i].a],P[F[i].b],P[F[i].c],p));
        double tmp2 = area(P[F[i].a],P[F[i].b],P[F[i].c]);
        return tmp1/tmp2;
    }
};
```

## 7.6 半平面交

半平面交
测试 POJ3335 POJ1474 POJ1279

```
struct halfplane:public Line {
    double angle;
    halfplane() {}
//表示向量 s->e 逆时针 (左侧) 的半平面
    halfplane(Point _s,Point _e) {
        s = _s;
        e = _e;
    }
    halfplane(Line v) {
        s = v.s;
        e = v.e;
    }
    void calcangle() {
        angle = atan2(e.y-s.y,e.x-s.x);
    }
    bool operator <(const halfplane &b)const {
        return angle < b.angle;
    }
};
struct halfplanes {
    int n;
    halfplane hp[N];
    Point p[N];
    int que[N];
    int st,ed;
    void push(halfplane tmp) {
        hp[n++] = tmp;
    }
//去重
    void unique() {
        int m = 1;
        for(int i = 1; i < n; i++) {
```

```
33              if(sgn(hp[i].angle-hp[i-1].angle) != 0)
34                  hp[m++] = hp[i];
35              else if(sgn( (hp[m-1].e-hp[m-1].s)^(hp[i].s-hp[m-1].s)
36                          ) > 0)
37                  hp[m-1] = hp[i];
38          }
39          n = m;
40      }
41      bool halfplaneinsert() {
42          for(int i = 0; i < n; i++)
43              hp[i].calcangle();
44          sort(hp,hp+n);
45          unique();
46          que[st=0] = 0;
47          que[ed=1] = 1;
48          p[1] = hp[0].crosspoint(hp[1]);
49          for(int i = 2; i < n; i++) {
50              while(st<ed && sgn((hp[i].e-hp[i].s)^(p[ed]-hp[i].s))<0)
51                  ed--;
52              while(st<ed && sgn((hp[i].e-hp[i].s)^(p[st+1]-hp[i].s))<0)
53                  st++;
54              que[++ed] = i;
55              if(hp[i].parallel(hp[que[ed-1]]))
56                  return false;
57              p[ed]=hp[i].crosspoint(hp[que[ed-1]]);
58          }
59          while(st<ed && sgn((hp[que[st]].e-hp[que[st]].s)^(p[ed]-hp[que[st]].s))
                <0)
60              ed--;
61          while(st<ed && sgn((hp[que[ed]].e-hp[que[ed]].s)^(p[st+1]-hp[que[ed]].s)
                )<0)
62              st++;
63          if(st+1>=ed)
64              return false;
65          return true;
66      }
67  //得到最后半平面交得到的凸多边形
68  //需要先调用 halfplaneinsert() 且返回 true
69      void getconvex(Pol &con) {
70          p[st] = hp[que[st]].crosspoint(hp[que[ed]]);
71          con.n = ed-st+1;
72          for(int j = st,i = 0; j <= ed; i++,j++)
73              con.p[i] = p[j];
74      }
75  };
76  struct Cirs
77  {
78      Cir c[N];
79      double ans[N];//ans[i] 表示被覆盖了 i 次的面积
80      double pre[N];
81      int n;
82      circles() {}
83      void add(Cir cc)
84      {
85          c[n++] = cc;
86      }
87  //x 包含在 y 中
88      bool inner(Cir x,Cir y)
89      {
90          if(x.relationCir(y) != 1)
91              return 0;
92          return sgn(x.r-y.r)<=0?1:0;
93      }
```

```
94  //圆的面积并去掉内含的圆
95      void init_or()
96      {
97          bool mark[N] = {0};
98          int i,j,k=0;
99          for(i = 0; i < n; i++)
100         {
101             for(j = 0; j < n; j++)
102                 if(i != j && !mark[j])
103                 {
104                     if( (c[i]==c[j])||inner(c[i],c[j]) )
105                         break;
106                 }
107             if(j < n)
108                 mark[i] = 1;
109         }
110         for(i = 0; i < n; i++)
111             if(!mark[i])
112                 c[k++] = c[i];
113         n = k;
114     }
115 //圆的面积交去掉内含的圆
116     void init_add()
117     {
118         int i,j,k;
119         bool mark[N] = {0};
120         for(i = 0; i < n; i++)
121         {
122             for(j = 0; j < n; j++)
123                 if(i != j && !mark[j])
124                 {
125                     if( (c[i]==c[j])||inner(c[j],c[i]) )
126                         break;
127                 }
128             if(j < n)
129                 mark[i] = 1;
130         }
131         for(i = 0; i < n; i++)
132             if(!mark[i])
133                 c[k++] = c[i];
134         n = k;
135     }
136 //半径为 r 的圆，弧度为 th 对应的弓形的面积
137     double areaarc(double th,double r)
138     {
139         return 0.5*r*r*(th-sin(th));
140     }
141 //测试 SPOJVCIRCLES SPOJCIRUT
142 //SPOJVCIRCLES 求 n 个圆并的面积，需要加上 init_or() 去掉重复圆（否则WA）
143 //SPOJCIRUT 是求被覆盖 k 次的面积，不能加 init_or()
144 //对于求覆盖多少次面积的问题，不能解决相同圆，而且不能 init_or()
145 //求多圆面积并，需要 init_or，其中一个目的就是去掉相同圆
146     void getarea()
147     {
148         memset(ans,0,sizeof(ans));
149         vector<pair<double,int> >v;
150         for(int i = 0; i < n; i++)
151         {
152             v.clear();
153             v.push_back(make_pair(-PI,1));
154             v.push_back(make_pair(PI,-1));
155             for(int j = 0; j < n; j++)
156                 if(i != j)
```

```
157                 {
158                     Point q = (c[j].p - c[i].p);
159                     double ab = q.len(),ac = c[i].r, bc = c[j].r;
160                     if(sgn(ab+ac-bc)<=0)
161                     {
162                         v.push_back(make_pair(-PI,1));
163                         v.push_back(make_pair(PI,-1));
164                         continue;
165                     }
166                     if(sgn(ab+bc-ac)<=0)
167                         continue;
168                     if(sgn(ab-ac-bc)>0)
169                         continue;
170                     double th = atan2(q.y,q.x), fai = acos((ac*ac+ab*ab-bc*bc)
                        /(2.0*ac*ab));
171                     double a0 = th-fai;
172                     if(sgn(a0+PI)<0)
173                         a0+=2*PI;
174                     double a1 = th+fai;
175                     if(sgn(a1-PI)>0)
176                         a1-=2*PI;
177                     if(sgn(a0-a1)>0)
178                     {
179                         v.push_back(make_pair(a0,1));
180                         v.push_back(make_pair(PI,-1));
181                         v.push_back(make_pair(-PI,1));
182                         v.push_back(make_pair(a1,-1));
183                     }
184                     else
185                     {
186                         v.push_back(make_pair(a0,1));
187                         v.push_back(make_pair(a1,-1));
188                     }
189                 }
190             sort(v.begin(),v.end());
191             int cur = 0;
192             for(int j = 0; j < v.size(); j++)
193             {
194                 if(cur && sgn(v[j].first-pre[cur]))
195                 {
196                     ans[cur] += areaarc(v[j].first-pre[cur],c[i].r);
197                     ans[cur] += 0.5*(Point(c[i].p.x+c[i].r*cos(pre[cur]),c[i].p.
                        y+c[i].r*sin(pre[cur]))^Point(c[i].p.x+c[i].r*cos(v[j].
                        first),c[i].p.y+c[i].r*sin(v[j].first)));
198                 }
199                 cur += v[j].second;
200                 pre[cur] = v[j].first;
201             }
202         }
203         for(int i = 1; i < n; i++)
204             ans[i] -= ans[i+1];
205     }
206 };
```

## 7.7 平面最近点对

```
1 typedef pair<double, double>Point;
2 double dis(Point a, Point b)
3 {
4     return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
5 }
```

```
Point p[N], tmp[N];
double Closest_Pair(int l, int r)
{
    if(l == r)
        return 1e20;
    if(l + 1 == r)
        return dis(p[l], p[r]);
    double d = min(Closest_Pair(l, mid), Closest_Pair(mid + 1, r));
    int cnt = 0;
    for(int i = l; i <= r; i++)
    {
        if(fabs(p[mid].x - p[i].x) <= d)
            tmp[cnt++] = p[i];
    }
    sort(tmp, tmp + cnt, [](Point a, Point b)
    {return a.y == b.y ? a.x < b.x : a.y < b.y;});
    for(int i = 0; i < cnt; i++)
    {
        for(int j = i + 1; j < cnt && tmp[j].y - tmp[i].y < d; j++)
            d = min(d, dis(tmp[i], tmp[j]));
    }
    return d;
}
int main()
{
    int n; cin >> n;
    for(int i = 0; i < n; i++)
        cin >> p[i].x >> p[i].y;
    sort(p, p + n);
    cout << Closest_Pair(0, n - 1);
    return 0;
}
```

## 7.8 三角剖分

```
struct Delaunay{
    int sgn(double x) {return x <= -eps ? -1 : (x < eps ? 0 : 1);}
    struct Point{
        double x, y;
        int id;
        void o(){printf("%lf %lf\n",x,y);}
        Point(double a = 0, double b = 0) : x(a), y(b) {}
        bool operator<(Point a){return x < a.x || (fabs(x - a.x) < eps && y < a.
            y);}
        bool operator==(Point a){return fabs(x - a.x) < eps && fabs(y - a.y) <
            eps;}
        double cross(Point a, Point b) {return (a.x-x)*(b.y-y)-(a.y-y)*(b.x-x);}
        double dis2(Point b){return (x - b.x) * (x - b.x) + (y - b.y) * (y - b.y
            );}
    };
    struct Point3{
        double x, y, z;
        Point3(double _x = 0, double _y = 0, double _z = 0): x(_x), y(_y), z(_z)
             {}
        Point3(Point p){x=p.x, y=p.y, z=p.x*p.x+p.y*p.y;}
        Point3 operator-(Point3 b){return Point3(x - b.x, y - b.y, z - b.z);}
        double operator*(Point3 b){return x * b.x + y * b.y + z * b.z;}
        Point3 operator^(Point3 b){return Point3(y*b.z-z*b.y,z*b.x-x *b.z,x*b.y-
            y*b.x);}
        Point3 cross(Point3 b,Point3 c) {return (b-*this)^(c-*this);}
    };
```

```
22      int inCircle(Point a, Point b, Point c, Point p){
23          if (a.cross(b, c) < 0) swap(b, c);
24          Point3 a3(a), b3(b), c3(c), p3(p);
25          b3 = b3 - a3, c3 = c3 - a3, p3 = p3 - a3;
26          Point3 f = b3^c3;
27          return sgn(p3*f);  // check same direction, in: < 0, on: = 0, out: > 0
28      }
29      int intersection(Point a, Point b, Point c,Point d){//seg(a,b)and seg(c,d)
30          return sgn(a.cross(c, b)) * sgn(a.cross(b, d)) > 0 &&
31                 sgn(c.cross(a, d)) * sgn(c.cross(d, b)) > 0;
32      }
33      struct Edg{
34          int id;
35          list<Edg>::iterator c;
36          Edg(int _id=0):id(_id){}
37      };
38      list<Edg> head[N];  // graph
39      Point p[N];
40      int n;
41
42      void addEdg(int u, int v){
43          head[u].push_front(Edg(v));
44          head[v].push_front(Edg(u));
45          head[u].begin()->c = head[v].begin();
46          head[v].begin()->c = head[u].begin();
47      }
48      void divide(int l, int r){
49          if (r - l <= 2)    {// #point <= 3
50              for (int i = l; i <= r; i++)
51                  for (int j = i + 1; j <= r; j++) addEdg(i, j);
52              return;
53          }
54          int mid = (l + r) / 2;
55          divide(l, mid);
56          divide(mid + 1, r);
57          int nowl = l, nowr = r;
58          // find left and right convex, lower common tangent
59          for(int update=1;update;){
60              update = 0;
61              Point ptL = p[nowl], ptR = p[nowr];
62              for (auto i:head[nowl]){
63                  Point t = p[i.id];
64                  double v = ptR.cross(ptL, t);
65                  if(sgn(v)>0 || (sgn(v)==0 && ptR.dis2(t)<ptR.dis2(ptL))){
66                      nowl = i.id, update = 1;
67                      break;
68                  }
69              }
70              if (update) continue;
71              for (auto i:head[nowr]){
72                  Point t = p[i.id];
73                  double v = ptL.cross(ptR, t);
74                  if (sgn(v) < 0 || (sgn(v) == 0 && ptL.dis2(t) < ptL.dis2(ptR))){
75                      nowr = i.id, update = 1;
76                      break;
77                  }
78              }
79          }
80          addEdg(nowl, nowr); // add tangent
81          for(int update=1;;){
82              update = 0;
83              Point ptL = p[nowl], ptR = p[nowr];
84              int ch = -1, side = 0;
```

```
            for (auto it = head[nowl].begin(); it != head[nowl].end(); it++)
                if (sgn(ptL.cross(ptR,p[it->id]))>0&&(ch==-1||inCircle(ptL,ptR,p
                    [ch],p[it->id])<0))
                    ch = it->id, side = -1;
            for (auto it = head[nowr].begin(); it != head[nowr].end(); it++)
                if (sgn(ptR.cross(p[it->id],ptL))>0&&(ch==-1||inCircle(ptL,ptR,p
                    [ch],p[it->id])<0))
                    ch = it->id, side = 1;
            if (ch == -1) break;  // upper common tangent
            if (side == -1){
                for (auto it = head[nowl].begin(); it != head[nowl].end();)
                    if (intersection(ptL, p[it->id], ptR, p[ch])){
                        head[it->id].erase(it->c);
                        head[nowl].erase(it++);
                    } else it++;
                nowl = ch;
                addEdg(nowl, nowr);
            }
            else{
                for (auto it = head[nowr].begin(); it != head[nowr].end();)
                    if (intersection(ptR, p[it->id], ptL, p[ch])){
                        head[it->id].erase(it->c);
                        head[nowr].erase(it++);
                    }else it++;
                nowr = ch;
                addEdg(nowl, nowr);
            }
        }
    }
    void solve(){
        scanf("%d",&n);
        for(int i=0;i<n;i++){
            p[i].id=i+1;
            scanf("%lf%lf",&p[i].x,&p[i].y);
        }
        sort(p, p + n);
        divide(0, n - 1);
        for (int i = 0; i < n; i++)for(auto j:head[i])
            arr.push_back({p[i].id, p[j.id].id,(ll)p[i].dis2(p[j.id])});
    }
};
```

## 7.9 旋转卡壳

找三个点形成最小三角形

```
#include <bits/stdc++.h>
using namespace std;
const int N=1000+10;
typedef long long ll;
struct Point
{
    ll x, y;
    Point() {}
    Point(ll _x, ll _y): x(_x), y(_y) {}
    void in(){cin>>x>>y;}
    bool operator<(Point b)const{return x < b.x || ( x == b.x && y < b.y );}
    Point operator-(Point b) {return Point(x - b.x, y - b.y);}
    ll operator^(Point b) {return x * b.y - y * b.x;}
} p[N];
struct Line
{
```

```
    int x, y;
    double k;
    bool operator<(Line b)const{return k < b.k;}
};

ll ans = 0x3f3f3f3f3f3f3f3f;
int n,rk[N],id[N];
vector<Line>l;
int main()
{
    cin>>n;
    for( int i = 1 ; i <= n ; i++ )p[i].in();
    sort( p + 1, p + n + 1 );
    for( int i = 1 ; i <= n ; i++ ) id[ i ] = rk[ i ] = i;
    for( int i = 1 ; i < n ; i++ )//k属于[-pi/2,pi/2)
        for( int j = i + 1 ; j <= n ; j++ )
            l.push_back({i,j,atan2(p[j].y-p[i].y,p[j].x-p[i].x)});
    sort(l.begin(),l.end());
    for(auto i:l)
    {
        int a=i.x,b=i.y;
        if( id[ a ] > id[ b ] ) swap( a, b );
        if(id[a]!=1)ans=min(ans, abs( (p[b]-p[a])^(p[b]-p[rk[id[a]-1]]) ));
        swap(id[a],id[b]);
        swap(rk[ id[a] ],rk[ id[b] ]);
    }
    cout<<ans/2<<(ans%2?".50":".00");
    return 0;
}
```

## 7.10 公式

Pick 定理：给定顶点均为整点的多边形，皮克定理说明了其面积 S 和内部格点数目 a、边上格点数目 b 的关系：S=a+b/2+1

已知圆锥表面积 S 求最大体积 V：$V = S \times \sqrt{\frac{S}{72\pi}}$

atan2(y,x) 函数：返回 $\frac{y}{x}$ 的反正切值，以弧度表示，从左下逆时针到左上取值范围为 $(-\pi, \pi]$