字符串

1.回文自动机
2.SAM 统计出现次数在[A,B]之间的串的个数
3.SAM 求第 k 小串
4.SAM 在线修改 right 集合
5.ac 自动机拓扑排序 DP

数学

1.java 高精度开方

图论

1.A*搜索
2.最长路

算法

1.斯坦纳树
2.中缀表达式计算

数据结构

1. CDQ 分治处理线段覆盖问题
2.cdq 分治动态求长方体内点的数量
3.CDQ 分治三维偏序问题
4.CDQ 分治二维偏序常见数据结构优化 DP
5.整体二分--静态区间第 k 小
6.整体二分--动态区间第 k 小
7.树状数组离线求区间不同数字个数
8.KDtree
9.树分治 treedp 计数
10.树分治计数

计算几何

1.基础
2.求凸包内核
3.卷包裹算法
4.安德鲁算法求凸包
5.求凸包宽度
6.Simpson 多圆面积并
7.多边形重心
8.最近点对
9.最远点对
10.凸包最大三角形
11.最小矩形覆盖

1.回文自动机

len[i]:节点 i 的回文串的长度
next[i][c]：节点 i 的回文串在两边添加字符 c 以后变成的回文串的编号(和字典树的 next 指针类似)
fail[i]：类似于 AC 自动机的 fail 指针，指向失配后需要跳转到的节点
cnt[i]：节点 i 表示的回文串在 S 中出现的次数（建树时求出的不是完全的，count()加上子节点以后才是正确的）
num[i]：以节点 i 回文串的末尾字符结尾的但不包含本条路径上的回文串的数目。(也就是 fail 指针路径的深度)
last：指向最新添加的回文结点
S[i]表示第 i 次添加的字符
p 表示添加的节点个数

空间复杂度为 O（N*字符集大小），时间复杂度为 O（N*log（字符集大小））

1.求串 S 前缀 0~i 内本质不同回文串的个数（两个串长度不同或者长度相同且至少有一个字符不同便是本质不同）

2.求串 S 内每一个本质不同回文串出现的次数

3.求串 S 内回文串的个数（其实就是 1 和 2 结合起来）

4.求以下标 i 结尾的回文串的个数

```
#define  LL long long
#define  ULL unsigned long long
using namespace std;
const int MAXN = 100005 ;
const int N = 26;
char s[MAXN];
struct Palindromic_Tree
{
    int next[MAXN][N] ;//next 指针，next 指针和字典树类似，指向的串为当前串两端加上同一个字符构成
    int fail[MAXN] ;//fail 指针，失配后跳转到 fail 指针指向的节点
    int cnt[MAXN] ;
    int num[MAXN] ; // 当前节点通过 fail 指针到达 0 节点或 1 节点的步数(fail 指针的深度)
    int len[MAXN] ;//len[i]表示节点 i 表示的回文串的长度
    int S[MAXN] ;//存放添加的字符
    int last ;//指向上一个字符所在的节点，方便下一次 add
    int n ;//字符数组指针
    int p ;//节点指针
    int newnode(int l)     //新建节点
    {
```

```
        for(int i = 0 ; i < N ; ++ i) next[p][i] = 0 ;
        cnt[p] = 0 ;
        num[p] = 0 ;
        len[p] = l ;
        return p ++ ;
    }
    void init()   //初始化
    {
        p = 0 ;
        newnode(0) ;
        newnode(-1) ;
        last = 0 ;
        n = 0 ;
        S[n] = -1 ;//开头放一个字符集中没有的字符，减少特判
        fail[0] = 1 ;
    }
    int get_fail(int x)     //和 KMP 一样，失配后找一个尽量最长的
    {
        while(S[n - len[x] - 1] != S[n]) x = fail[x] ;
        return x ;
    }
    void add(int c,int pos)
    {
        printf("%d:",p);
        c -= 'a';
        S[++ n] = c ;
        int cur = get_fail(last);   //通过上一个回文串找这个回文串的匹配位置
        printf("%d ",cur);
        if(!next[cur][c])     //如果这个回文串没有出现过，说明出现了一个新的本质不同的回
文串
        {
            int now = newnode(len[cur] + 2) ;   //新建节点
            fail[now] = next[get_fail(fail[cur])][c] ;   //和 AC 自动机一样建立 fail
指针，以便失配后跳转
            next[cur][c] = now ;
            num[now] = num[fail[now]] + 1 ;
            for(int i=pos-len[now]+1; i<=pos; ++i) printf("%c",s[i]);
        } last = next[cur][c] ;
        cnt[last] ++ ;
        putchar(10);
    }
    void count()
    {
        for(int i = p - 1 ; i >= 0 ; -- i) cnt[fail[i]] += cnt[i] ;
```

```
            //父亲累加儿子的 cnt，因为如果 fail[v]=u，则 u 一定是 v 的子回文串！
        }
} run;
int main()
{
    scanf("%s",&s);
    int n=strlen(s);
    run.init();
    for(int i=0; i<n; i++) run.add(s[i],i);
    run.count();
    return 0;
}
```

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 10;
const int maxn = 2e6+10;
const int mod = 1e9+7;
char s[maxn];

ll qpow(ll a,ll n){
    ll ret=1;
    while(n){
        if(n&1)ret=ret*a%mod;
        a=a*a%mod;
        n>>=1;
    }return ret;
}

struct pam{
    int next[maxn][N],fail[maxn];
    int len[maxn],S[maxn];
    int cnt[maxn],num[maxn];
    int ppos[maxn],number[maxn];
    int last,n,p;
    int newnode(int l){
        for(int i=0;i<N;i++)next[p][i]=0;
        cnt[p]=num[p]=0;
        len[p]=l;
        return p++;
    }
```

```
    void init(){
        p=0;
        newnode(0);newnode(-1);
        last=n=0;
        S[n]=-1;
        fail[0]=1;
    }
    int get_fail(int x){
        while(S[n-len[x]-1]!=S[n])x=fail[x];
        return x;
    }
    void add(int c,int pos){
        c-='0';
        S[++n]=c;
        int cur=get_fail(last);
        //printf("%d:%d ",p,cur);
        if(!next[cur][c]){
            int now=newnode(len[cur]+2);
            ppos[now]=pos;
            fail[now]=next[get_fail(fail[cur])][c];
            next[cur][c]=now;
            number[now] = ((10ll*number[cur])%mod+c)%mod;
            if(len[cur]>=0){
                number[now]=(number[now]+1ll*c*qpow(10,len[cur]+1)%mod)%mod;
            }
            num[now]=num[fail[now]]+1;
            //for(int i=pos-len[now]+1;i<=pos;i++)printf("%c",s[i]);
        }
        last = next[cur][c];
        cnt[last]++;
        //putchar(10);
    }
    void count(){
        ll ret=0;
        for(int i=p-1;i>=0;i--){
            cnt[fail[i]]+=cnt[i];
            ret=(ret+number[i])%mod;
        }cout<<ret<<endl;
    }
}run;

int hs[maxn],po[maxn];
ll geth(int l,int r){
    ll ret=hs[r]-1ll*hs[l-1]*po[r-l+1]%mod;
```

```
    ret%=mod;ret+=mod;ret%=mod;
    return ret;
}

int main(){
    po[0]=1;
    for(int i=1;i<maxn;i++)po[i]=1ll*po[i-1]*10%mod;
    while(scanf("%s",s+1)!=EOF){
        int len=strlen(s+1);
        run.init();
        for(int i=1;i<=len;i++){
            run.add(s[i],i);
            hs[i]=(1ll*hs[i-1]*10+(s[i]-'0'))%mod;
        }
        run.count();
//        ll ans=0;
//        for(int i=2;i<run.p;i++){
//            ans=(ans+geth(run.ppos[i]-run.len[i]+1,run.ppos[i]))%mod;
//        }
//        cout<<ans<<endl;
    }
    return 0;
}
```

2.SAM 统计出现次数在[A,B]之间的串的个数

```
/**统计出现次数在[A,B]之间的串的个数**/
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

const int maxn = 2e5+1000;
char s[maxn];
int pa[maxn<<1],son[maxn<<1][27];
int deep[maxn<<1],cnt,root,last;

int sum[maxn<<1],topo[maxn<<1];
int r[maxn<<1];

inline int newnode(int _deep){
    deep[++cnt]=_deep;
    return cnt;
}
```

```
inline void sam(int alp){
    int np=newnode(deep[last]+1);
    int u=last;
    memset(son[np],0,sizeof son[np]);
    while(u && !son[u][alp])son[u][alp]=np,u=pa[u];
    if(!u)pa[np]=root;
    else{
        int v=son[u][alp];
        if(deep[v]==deep[u]+1)pa[np]=v;
        else{
            int nv=newnode(deep[u]+1);
            memcpy(son[nv],son[v],sizeof son[v]);
            pa[nv]=pa[v],pa[v]=pa[np]=nv;
            while(u&&son[u][alp]==v)son[u][alp]=nv,u=pa[u];
        }
    }
    last=np;
}

inline void toposort(){
    for(int a=1;a<=deep[last];a++)sum[a]=0;
    for(int a=1;a<=cnt;a++)sum[deep[a]]++;
    for(int a=1;a<=deep[last];a++)sum[a]+=sum[a-1];
    for(int a=1;a<=cnt;a++)topo[sum[deep[a]]--]=a;
}

inline void pre(){
    cnt=0;
    root=last=newnode(0);
    memset(r,0,sizeof r);
}
int A,B;
int main(){
    while(scanf("%s%d%d",s,&A,&B)!=EOF){
    pre();
    memset(son[1],0,sizeof son[1]);
    int len=strlen(s);
    for(int a=0;a<len;a++){
        sam(s[a]-'A');
    }
    toposort();
    int tmp=root;
    for(int a=0;a<len;a++){
        tmp=son[tmp][s[a]-'A'];
```

```
        r[tmp]=1;
    }
    /**将所有状态的 right 集合大小赋值为 1**/
    for(int a=cnt;a>=1;a--)r[pa[topo[a]]] += r[topo[a]];
    long long ans=0;
    for(int a=cnt;a>=1;a--){
        if(r[topo[a]]>=A && r[topo[a]]<=B){
            ans+=deep[topo[a]] - deep[pa[topo[a]]];
            /**right 集合中不同子串个数为 deep[p] - deep[pa[p]]]**/
        }
    }
    cout<<ans<<endl;
    }
    return 0;
}
```

3.SAM 求第 k 小串

```
/**T 为 0 则表示不同位置的相同子串算作一个。T=1 则表示不同位置的相同子串算作多个**/
/**统计出现次数在[A,B]之间的串的个数**/
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

const int maxn = 1e6+1000;
char s[maxn];
int pa[maxn<<1],son[maxn<<1][27];
int deep[maxn<<1],cnt,root,last;

int sum[maxn<<1],topo[maxn<<1];
int r[maxn<<1],rr[maxn<<1];

inline int newnode(int _deep){
    deep[++cnt]=_deep;
    return cnt;
}

inline void sam(int alp){
    int np=newnode(deep[last]+1);
    int u=last;r[np]=1;
    memset(son[np],0,sizeof son[np]);
    while(u && !son[u][alp])son[u][alp]=np,u=pa[u];
```

```
    if(!u)pa[np]=root;
    else{
        int v=son[u][alp];
        if(deep[v]==deep[u]+1)pa[np]=v;
        else{
            int nv=newnode(deep[u]+1);
            memcpy(son[nv],son[v],sizeof son[v]);
            pa[nv]=pa[v],pa[v]=pa[np]=nv;
            while(u&&son[u][alp]==v)son[u][alp]=nv,u=pa[u];
        }
    }
    last=np;
}

inline void toposort(){
    for(int a=1;a<=cnt;a++)sum[deep[a]]++;
    for(int a=1;a<=deep[last];a++)sum[a]+=sum[a-1];
    for(int a=1;a<=cnt;a++)topo[sum[deep[a]]--]=a;
}

inline void pre(){
    cnt=0;
    root=last=newnode(0);
}
int T,K;
int main(){
    scanf("%s%d%d",s,&T,&K);
    pre();
    memset(son[1],0,sizeof son[1]);
    int len=strlen(s);
    for(int a=0;a<len;a++){
        sam(s[a]-'a');
    }
    toposort();
    for(int i=cnt;i;i--){
        if(T)r[pa[topo[i]]] += r[topo[i]];
        else r[topo[i]]=1;
    }
    r[root]=0;
    for(int i=cnt;i;i--){
        rr[topo[i]]=r[topo[i]];
        for(int j=0;j<26;j++){
            rr[topo[i]]+=rr[son[topo[i]][j]];
        }
```

```
    }

    int now=root;
    if(K>rr[root])return puts("-1");
    while(K){
        K-=r[now];
        if(K<=0)break;
        for(int i=0;i<26;i++){
            if(son[now][i]){
                if(rr[son[now][i]]>=K){
                    putchar(i+'a');
                    now=son[now][i];
                    break;
                }
                else K-=rr[son[now][i]];
            }
        }
    }
    puts("");
    return 0;
}



#include<cstdio>
#include<cstring>
#include<algorithm>
#include<vector>
using namespace std;
const int MAXN = 1e6 + 10;
inline int read() {
    char c = getchar(); int x = 0, f = 1;
    while(c < '0' || c > '9') {if(c == '-') f = -1; c = getchar();}
    while(c >= '0' && c <= '9') x = x * 10 + c - '0', c = getchar();
    return x * f;
}
char s[MAXN];
int opt, K, N;
int fa[MAXN], len[MAXN], ch[MAXN][27], siz[MAXN], right[MAXN], tot = 1, last = 1, root
= 1;
void insert(int x) {
    int now = ++tot, pre = last; last = now; len[now] = len[pre] + 1;
    right[now] = 1;
    for(; pre && !ch[pre][x]; pre = fa[pre]) ch[pre][x] = now;
    if(!pre) fa[now] = root;
```

```
    else {
        int q = ch[pre][x];
        if(len[q] == len[pre] + 1) fa[now] = q;
        else {
            int nows = ++tot; len[nows] = len[pre] + 1;
            memcpy(ch[nows], ch[q], sizeof(ch[q]));
            fa[nows] = fa[q]; fa[q] = fa[now] = nows;
            for(; pre && ch[pre][x] == q; pre = fa[pre]) ch[pre][x] = nows;
        }
    }
}
void Query(int K) {
    int now = root;
    if(K > siz[root]) return (void) printf("-1");
    while(K) {
        if(now != root) K -= right[now];
        if(K <= 0) break;
        for(int i = 0; i <= 25; i++)
            if(ch[now][i]) {
                if(siz[ch[now][i]] >= K) {putchar(i + 'a'); now = ch[now][i]; break; }
                else K -= siz[ch[now][i]];
            }
    }
    puts("");
}
void Topsort() {
    static int A[MAXN], a[MAXN];
    for(int i = 1; i <= tot; i++) A[len[i]]++;
    for(int i = 1; i <= N; i++)   A[i] += A[i - 1];
    for(int i = tot; i >= 1; i--) a[A[len[i]]--] = i;
    //for(int i = 1; i <= tot; i++) siz[i] = 1;

    if(opt == 1)
        for(int i = tot; i; i--) right[fa[a[i]]] += right[a[i]];
    if(opt == 0)
        for(int i = tot; i; i--) right[a[i]] = 1;
    for(int i = tot; i; i--) {
        siz[a[i]] = right[a[i]];
        for(int j = 0; j <= 25; j++)
            siz[a[i]] += siz[ch[a[i]][j]];
    }
}
int main() {
    scanf("%s", s + 1);
```

```
    opt = read(), K = read();
    N = strlen(s + 1);
    for(int i = 1; i <= N; i++) insert(s[i] - 'a');
    Topsort();
    Query(K);
    return 0;
}
```

4.SAM 在线修改 right 集合

```
/**统计出现次数在[A,B]之间的串的个数**/
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

const int maxn = 2e5+1000;
char s[maxn];
int pa[maxn<<1],son[maxn<<1][27];
int deep[maxn<<1],cnt,root,last;

int sum[maxn<<1],topo[maxn<<1];
int r[maxn<<1];

inline int newnode(int _deep){
    deep[++cnt]=_deep;
    return cnt;
}

inline void sam(int alp){
    int np=newnode(deep[last]+1);
    int u=last;
    memset(son[np],0,sizeof son[np]);
    while(u && !son[u][alp])son[u][alp]=np,u=pa[u];
    if(!u)pa[np]=root;
    else{
        int v=son[u][alp];
        if(deep[v]==deep[u]+1)pa[np]=v;
        else{
            int nv=newnode(deep[u]+1);
            memcpy(son[nv],son[v],sizeof son[v]);
            r[nv]=r[v];
            pa[nv]=pa[v],pa[v]=pa[np]=nv;
            while(u&&son[u][alp]==v)son[u][alp]=nv,u=pa[u];
```

```
        }
    }
    last=np;
    int temp=last;
    while(temp>1){
        r[temp]++;
        temp=pa[temp];
    }
}

inline void toposort(){
    for(int a=1;a<=deep[last];a++)sum[a]=0;
    for(int a=1;a<=cnt;a++)sum[deep[a]]++;
    for(int a=1;a<=deep[last];a++)sum[a]+=sum[a-1];
    for(int a=1;a<=cnt;a++)topo[sum[deep[a]]--]=a;
}

inline void pre(){
    cnt=0;
    root=last=newnode(0);
    memset(r,0,sizeof r);
}
int A,B;
int main(){
    while(scanf("%s%d%d",s,&A,&B)!=EOF){
    pre();
    memset(son[1],0,sizeof son[1]);
    int len=strlen(s);
    for(int a=0;a<len;a++){
        sam(s[a]-'A');
    }
    toposort();
    long long ans=0;
    for(int a=cnt;a>=1;a--){
        if(r[topo[a]]>=A && r[topo[a]]<=B){
            ans+=deep[topo[a]] - deep[pa[topo[a]]];
        }
    }
    cout<<ans<<endl;
    }
    return 0;
}
```

5.ac 自动机拓扑排序 DP

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

const int maxn = 2e5+10;
string s[maxn];
char ans[maxn];
int flag=0,ansnum=0,tt=0;
int G[maxn][2];
void addedge(int u,int v,int w){
    G[u][w]=v;
}

struct Trie{
    int next[maxn][2],fail[maxn],ed[maxn];
    int root,L;
    int newnode(){
        memset(next[L],-1,sizeof next[L]);
        ed[L++]=0;
        return L-1;
    }
    void init(){
        memset(fail,0,sizeof fail);
        memset(ed,0,sizeof ed);
        L=0;
        root=newnode();
    }
    void insert(string buf){
        int len=buf.length();
        int now=root;
        for(int i=0;i<len;i++){
            if(next[now][buf[i]-'0']==-1){
                next[now][buf[i]-'0']=newnode();
            }
            now=next[now][buf[i]-'0'];
        }
        ed[now]=1;
    }
    void buildfail(){
        queue<int> Q;
        fail[root]=root;
        for(int i=0;i<2;i++){
            if(next[root][i]==-1)
```

```
            next[root][i]=root;
        else{
            fail[next[root][i]]=root;
            Q.push(next[root][i]);
        }
    }
    while(!Q.empty()){
        int now=Q.front();Q.pop();
        if(ed[fail[now]])ed[now]=1;
        for(int i=0;i<2;i++){
            if(next[now][i]==-1){
                next[now][i]=next[fail[now]][i];
            }
            else{
                fail[next[now][i]]=next[fail[now]][i];
                Q.push(next[now][i]);
            }
        }
    }
}
int in[maxn],vis[maxn],inq[maxn],in2[maxn];
void dfs2(int now){
    for(int i=0;i<2;i++){
        int j=next[now][i];
        if(ed[j])continue;
        in[j]++;in2[j]++;
        if(in[j]==1)dfs2(j);
    }
}
bool topo(){
    memset(in,0,sizeof in);
    memset(in2,0,sizeof in2);
    memset(inq,0,sizeof inq);
    memset(vis,0,sizeof vis);
    dfs2(0);
    queue<int> q;
    if(in[0])return 1;
    q.push(0);
    while(!q.empty()){
        int now=q.front();q.pop();
        for(int i=0;i<2;i++){
            int j=next[now][i];
            if(ed[j])continue;
            in[j]--;
```

```
                vis[j]=1;
                if(!in[j]){
                    inq[j]=1;
                    q.push(j);
                }
            }
        }
        for(int i=0;i<L;i++){
            if(inq[i]!=vis[i])return true;
        }
        return false;
    }
    int dp[maxn],to[maxn],maxx=0;
    void dfs5(int now,int num){
        maxx=max(num,maxx);
        for(int i=0;i<2;i++){
            int j=next[now][i];
            if(ed[j])continue;
            dfs5(j,num+1);
        }
    }
    void dfs3(int now,int num){
        if(num==maxx){tt=1;return;}
        for(int i=0;i<2;i++){
            int j=next[now][i];
            if(ed[j])continue;
            dfs3(j,num+1);
            if(tt){
                ans[num]=i+'0';return;
            }
        }
    }
    void query(){
        if(topo()){
            puts("-1");return ;
        }
        memset(dp,0,sizeof dp);
        memset(to,0,sizeof to);
        queue<int>q;
        dfs5(0,0);
        dfs3(0,0);
        printf("%s",ans);
    }
}ac;
```

```
void init(){
    memset(ans,0,sizeof ans);
    flag=tt=ansnum=0;
}
int main(){
    int n;
    scanf("%d",&n);
    ac.init();
    init();
    for(int i=0;i<n;i++){
        cin>>s[i];
        ac.insert(s[i]);
    }
    ac.buildfail();
    ac.query();
    return 0;
}
```

1.java 高精度开方

```
import java.util.*;
import java.math.*;
public class Main{
    public static void main(String[] args){
        Scanner in = new Scanner(System.in);
        int t=in.nextInt();
        for(int z=0;z<t;z++){
            BigInteger x=in.nextBigInteger();
            int p=(x.toString().length()+1)/2;
            char tmp[]=new char[p+1];
            tmp[0]='1';
            for(int i=1;i<=p;i++)tmp[i]='0';
            String tt=String.valueOf(tmp);
            BigInteger r=new BigInteger(tt);
            while(true){
                BigInteger
dec=r.multiply(r).subtract(x).divide(r.multiply(BigInteger.valueOf(2)));
                if(dec.compareTo(BigInteger.ZERO)<=0)break;
                r=r.subtract(dec);
            }
            String rr=r.toString();
            int rrr=rr.charAt(rr.length()-1);
            if(rrr%2==0)System.out.println("1");
            else System.out.println("0");
        }
    }
}

import java.util.*;
import java.math.*;
public class Main{
    //二分开根板子
    public static BigInteger check(BigInteger n,BigInteger x) {
        BigInteger ans=BigInteger.valueOf(1);
        BigInteger a=BigInteger.valueOf(1);
        for(BigInteger i=BigInteger.ZERO;i.compareTo(n)<0;i=i.add(a)) {
            ans=ans.multiply(x);
        }
        return ans;
    }
    static BigInteger Get(BigInteger m) {
     BigInteger l=BigInteger.ZERO;
        BigInteger a=BigInteger.valueOf(2);
```

```
        BigInteger b=BigInteger.valueOf(1);
        BigInteger r=BigInteger.valueOf(1);
        BigInteger mid=BigInteger.ZERO;
        while(check(BigInteger.valueOf(2),r).compareTo(m)<=0) {
            l=r;
            r=r.multiply(a);
        }
        while(l.compareTo(r)<=0) {
            mid=l.add(r).divide(a);
            if(check(BigInteger.valueOf(2),mid).compareTo(m)<=0) l=mid.add(b);
            else r=mid.subtract(b);
        }
        return r;
    }
    public static void main(String[]args) {
     int T;
        Scanner sca=new Scanner(System.in);
        T=sca.nextInt();
        while(T--!=0) {
         BigInteger m=sca.nextBigInteger();
            BigInteger res1=Get(m);
            BigInteger n;
            n=m.multiply(m.subtract(BigInteger.ONE)).divide(BigInteger.valueOf(2));
            BigInteger res2=Get(n);
            BigInteger tmp1=res1.multiply(res1);
            BigInteger tmp2=res2.multiply(res2);
            if(tmp1.equals(m)&&tmp2.equals(n)) {
              System.out.println("Arena of Valor");
            }
            else if(tmp1.compareTo(m)==0&&tmp2.compareTo(n)!=0) {
              System.out.println("Hearth Stone");
            }
            else if(tmp1.compareTo(m)!=0&&tmp2.compareTo(n)==0) {
              System.out.println("Clash Royale");
            }
            else System.out.println("League of Legends");
        }
    }
}
```

1.A*搜索

```
#include<bits/stdc++.h>
using namespace std;
inline int read(){
    int x=0,f=1;char ch=getchar();
    while (ch<'0' || ch>'9'){if (ch=='-') f=-1;ch=getchar();}
    while (ch>='0' && ch<='9'){x=x*10+ch-'0';ch=getchar();}
    return x*f;
}
const int
    Point=1005,
    Edges=100005;
int n,m,start,send,kth;
int dist[Point],times[Point];
bool vis[Point];
struct Edge{
    int to,next,val;
}E[Edges],Eopp[Edges];     //Eopp means Eopposite
int head[Point],head_opp[Point];
struct A_Star_node{
    int p,g,h;
    bool operator < (A_Star_node x)const{
        return x.g+x.h<g+h;
    }
};        //means point  and a_Star:f(x)=g(x)+h(x);
priority_queue<A_Star_node>Q;
inline void add(int Ecnt,int u,int v,int w){
    E[Ecnt].next=head[u];
    E[Ecnt].to=v;
    E[Ecnt].val=w;
    head[u]=Ecnt;
}
inline void add_opposite(int EoppCnt,int u,int v,int w){
    Eopp[EoppCnt].next=head_opp[u];
    Eopp[EoppCnt].to=v;
    Eopp[EoppCnt].val=w;
    head_opp[u]=EoppCnt;
}
void dijkstra(int s,int e){
    memset(vis,0,sizeof(vis));
    memset(dist,127,sizeof(dist));
    int mini;    dist[e]=0;
    for (int i=1;i<=n;i++){
```

```
        mini=0;
        for (int j=1;j<=n;j++)
            if (!vis[j] && dist[mini]>dist[j])    mini=j;
        vis[mini]=1;
        for (int x=head_opp[mini];x;x=Eopp[x].next)
            dist[Eopp[x].to]=min(dist[Eopp[x].to],dist[mini]+Eopp[x].val);
    }
}
int A_Star(int s,int e){
    A_Star_node t1,tmp;
    memset(times,0,sizeof(times));
    t1.g=t1.h=0; t1.p=s;
    Q.push(t1);
    while (!Q.empty()){
        t1=Q.top();  Q.pop();
        times[t1.p]++;
        if (times[t1.p]==kth && t1.p==e) return t1.h+t1.g;
        if (times[t1.p]>kth) continue;
        for (int i=head[t1.p];i;i=E[i].next){
            tmp.p=E[i].to;
            tmp.g=dist[E[i].to];
            tmp.h=E[i].val+t1.h;
            Q.push(tmp);
        }
    }
    return -1;
}
int main(){
    n=read(),m=read(),kth=read(),start=read(),send=read();
    int x,y,z;
    memset(head,0,sizeof(head));
    memset(head_opp,0,sizeof(head_opp));
    for (int i=1;i<=m;i++){
        x=read(),y=read(),z=read();
        add(i,x,y,z);
        add_opposite(i,y,x,z);
    }
    dijkstra(start,send);
    if (start==send) kth++;
    int ans=A_Star(start,send);
    if (ans==-1) puts("No");
        elseprintf("%d\n",ans);
    return 0;
}
```

2.最长路

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define maxn 100005

struct node{
    int u,v,w;
    int id;
    node(int id,int u=0,int v=0,int w=0):id(id),u(u),v(v),w(w){}
};
vector<node> g[maxn],mp[maxn];
int u[maxn],v[maxn],w[maxn];

int ans=1;
int vis[maxn];
int dp[maxn];

int get(int u){
    if(vis[u])return dp[u];
    vis[u]=1;
    for(auto e:mp[u]){
        dp[u]=max(dp[u],get(e.v)+1);
    }
    return dp[u];
}

int main(){
    ll n,m;
    cin>>n>>m;
    for(int i=1;i<=m;i++){
        scanf("%d%d%d",u+i,v+i,w+i);
        g[u[i]].push_back(node(i,u[i],v[i],w[i]));
    }
    for(int i=1;i<=n;i++){
        for(auto e:g[i]){
            int j=e.v;
            for(auto ee:g[j]){
                if(e.w<ee.w && e.id<ee.id){
                    mp[e.id].push_back(node(e.id,e.id,ee.id,ee.w));
                }
            }
```

```
        }
    }
    memset(vis,0,sizeof(vis));
    for(int i=1;i<=m;i++){
            dp[i]=max(dp[i],get(i));
    }cout<<*max_element(dp+1,dp+1+m)+1;
     return 0;
}
```

1.斯坦纳树

```
/*
给一个有 n 个节点带权完全图，编号从 1~n，
现在有一个 0 节点，可以与 m 个节点相连，且给出了权值，
求 0 和这 m 个节点组成的最小斯坦纳树。
令 s 表示 m 个节点的子集，
dp[s][i]表示包含 s 中节点且以 i 为根的树的最小权值
dp[s][i]=min(dp[s1][i]+dp[s2][i],dp[s][j]+g[i][j]),
其中 s1+s2=s,
*/
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;
#define N 60
int g[N][N],a[N],dp[(1<<10)+10][N];
int n,m;
const int inf=0x3f3f3f3f;
int main()
{
    int ca;
    scanf("%d",&ca);
    while(ca--)
    {
        scanf("%d%d",&n,&m);
```

```
        for(int i=1;i<=n;++i)
            for(int j=1;j<=n;++j)
                scanf("%d",&g[i][j]);
        int x,w;
        for(int i=1;i<=n;++i)
            g[0][i]=g[i][0]=inf;
        for(int i=1;i<=m;++i)
        {
            scanf("%d%d",&x,&w);
            g[0][x]=g[x][0]=w;
            a[i]=x;
        }
        for(int k=0;k<=n;++k)
            for(int i=0;i<=n;++i)
                for(int j=0;j<=n;++j)
                    g[i][j]=min(g[i][j],g[i][k]+g[k][j]);
        memset(dp,0x3f,sizeof(dp));
        for(int i=1;i<=m;++i)
            for(int j=0;j<=n;++j)
                dp[1<<i-1][j]=g[a[i]][j];
        for(int s=1;s<(1<<m);++s)
        {
            if(s&(s-1)==0) continue;
            for(int i=0;i<=n;++i)
                for(int ss=s;ss;ss=(ss-1)&s)
                    dp[s][i]=min(dp[s][i],dp[ss][i]+dp[s-ss][i]);
            for(int i=0;i<=n;++i)
                for(int j=0;j<=n;++j)
                    dp[s][i]=min(dp[s][i],dp[s][j]+g[i][j]);
        }
        printf("%d\n",dp[(1<<m)-1][0]);
    }
    return 0;
}
```

2.中缀表达式计算

```
#include <cstdio>
#include <cstring>
#include <cstdlib>
#include <cmath>
#define N 1000
char st[N];
int slove(char *st,double *ans)
{
```

```
    double stai[N];
    char stac[N],stt[N];
    int ti,tc,i,l,ll;
    double x;
    l=strlen(st);
    ti=tc=ll=0;
    stt[0]='\0';
    for(i=0;i<l;++i)
    {
        if(('0'<=st[i]&&st[i]<='9')||st[i]=='.')
        {
            stt[ll++]=st[i];stt[ll]='\0';
        }
        else
        {
            if(stt[0]!='\0')
            {
                //puts(stt);
                x=atof(stt);stt[ll=0]='\0';
                stai[++ti]=x;
                //printf("%f\n",x);
            }
            if(st[i]=='+'||st[i]=='-')
            {

if(st[i]=='-'&&(i==0||st[i-1]<'0'||st[i-1]>'9')){stt[ll++]='-';stt[ll]='\0';}
                else
                {
                    while(tc&&stac[tc]!='(')
                    {
                        if(ti>1)
                        {
                            switch(stac[tc--])
                            {
                                case '+':stai[ti-1]+=stai[ti];break;
                                case '-':stai[ti-1]-=stai[ti];break;
                                case '*':stai[ti-1]*=stai[ti];break;
                                case '/':stai[ti-1]/=stai[ti];break;
                                case '^':stai[ti-1]=pow(stai[ti-1],stai[ti]);break;
                            }
                            --ti;
                        }
                        else return 1;
                    }
```

```
        stac[++tc]=st[i];
    }
}
else if(st[i]=='*'||st[i]=='/')
{
    while(tc&&stac[tc]=='*'||stac[tc]=='/'||stac[tc]=='^')
    {
        if(ti>1)
        {
            switch(stac[tc--])
            {
                case '*':stai[ti-1]*=stai[ti];break;
                case '/':stai[ti-1]/=stai[ti];break;
                case '^':stai[ti-1]=pow(stai[ti-1],stai[ti]);break;
            }
            --ti;
        }
        else return 2;
    }
    stac[++tc]=st[i];
}
else if(st[i]=='^'||st[i]=='(') stac[++tc]=st[i];
else if(st[i]==')')
{
    while(stac[tc]!='(')
    {
        if(tc==0) return 3;
        if(ti>1)
        {
            switch(stac[tc--])
            {
                case '+':stai[ti-1]+=stai[ti];break;
                case '-':stai[ti-1]-=stai[ti];break;
                case '*':stai[ti-1]*=stai[ti];break;
                case '/':stai[ti-1]/=stai[ti];break;
                case '^':stai[ti-1]=pow(stai[ti-1],stai[ti]);break;
            }
            --ti;
        }
        else return 4;
    }
    --tc;
}
else return 5;
```

```
        }
    }
    if(stt[0]!='\0') stai[++ti]=atof(stt);
    //puts(stt);
    //printf("%f %d %d %c\n",stai[ti],ti,tc,stac[tc]);
    while(tc)
    {
        if(ti>1)
        {
            switch(stac[tc--])
            {
                case '+':stai[ti-1]+=stai[ti];break;
                case '-':stai[ti-1]-=stai[ti];break;
                case '*':stai[ti-1]*=stai[ti];break;
                case '/':stai[ti-1]/=stai[ti];break;
                case '^':stai[ti-1]=pow(stai[ti-1],stai[ti]);break;
                default:return 6;
            }
            --ti;
        }
        else return 7;
    }
    if(ti==1){*ans=stai[1];return 0;}
    else return 8;
}
int main()
{
    double ans;
    gets(st);
    //puts(st);
    int k=slove(st,&ans);
    if(k==0) printf("%g\n",ans);
    else printf("error %d\n",k);
    return 0;
}
```

1．CDQ 分治处理线段覆盖问题

```
#include <algorithm>
#include <cstdio>
#include <cstring>
using namespace std;
typedef long long ll;

const int maxn = 1e5+10;
struct Node{
    int idx,l,r,delt;
    int kind;
    bool operator < (const Node &rhs)const{
        return l<rhs.l;
    }
}node[maxn];

int ans[maxn],del[maxn];
/*******反向 BIT*******/
inline int lowbit(int x){return x&(-x);}
int arr[maxn],MAX;
void add(int x,int d){
    while(x){
        arr[x]+=d;
        x-=lowbit(x);
    }
}
int sum(int x){
    int ret=0;
    while(x<=MAX){
        ret+=arr[x];
        x+=lowbit(x);
    }return ret;
}
/****离散化****/
int vec[maxn],vec_idx;
int hs(int x){
    return lower_bound(vec,vec+vec_idx,x)-vec+1;
}
/*****CDQ*****/
void cdq(int l,int r){
    if(l==r)return;
    int mid = (l+r)>>1;
    cdq(l,mid);
```

```
    cdq(mid+1,r);
    int j=l;
    for(int i=mid+1;i<=r;i++){
        if(node[i].kind==2){
            for(;j<=mid && node[j].l<=node[i].l;j++){
                if(node[j].kind==1){
                    add(hs(node[j].r),node[j].delt);
                }
            }
            ans[node[i].idx]+=sum(hs(node[i].r));
        }
    }
    for(int i=l;i<j;i++){
        if(node[i].kind==1){
            add(hs(node[i].r),-node[i].delt);
        }
    }
    inplace_merge(node+l,node+mid+1,node+r+1);
}
int vis[maxn];
int main(){
    int n,m;
    while(scanf("%d%d",&m,&n)!=EOF){
        int cnt=0;
        vec_idx=0;
        memset(arr,0,sizeof(arr));
        memset(ans,0,sizeof(ans));
        memset(vis,0,sizeof(vis));
        vector<int> v;
        char op[3];
        for(int i=1;i<=n;i++){
            scanf("%s",op);
            if(op[0]=='1'){
                scanf("%d%d",&node[i].l,&node[i].r);
                node[i].kind=1;
                node[i].idx=i;
                node[i].delt=1;
                vec[vec_idx++]=node[i].r;
                v.push_back(i);
            } else if(op[0]=='2'){
                scanf("%d%d",&node[i].l,&node[i].r);
                node[i].kind=2;
                node[i].idx=i;
                vec[vec_idx++]=node[i].r;
```

```
                vis[i]=1;
            } else{
                int tmp;
                scanf("%d",&tmp);
                node[i].kind=1;
                node[i].l=node[v[tmp-1]].l;
                node[i].r=node[v[tmp-1]].r;
                node[i].delt=-1;
                node[i].idx=i;
            }
        }
        sort(vec,vec+vec_idx);
        MAX = vec_idx+10;
        cdq(1,n);
        for(int i=1;i<=n;i++){
            if(vis[i]){
                printf("%d\n",ans[i]);
            }
        }
    }
    return 0;
}
```

2.cdq 分治动态求长方体内点的数量

```
/*
1 x y z 添加一个点
2 x1 y1 z1 x2 y2 z2 查找长方体内点的数量
CDQ 分治套 CDQ 分治套树状数组
第一重 CDQ 分治计算左边的修改对右边的询问的影响，
第二重 CDQ 分治处理三维偏序问题
*/
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;
#define N 50010
struct rec{
    int k,x,y,z,w,id;
    rec(){}
    rec(int k,int x,int y,int z,int w,int id):
        k(k),x(x),y(y),z(z),w(w),id(id){}
    bool operator==(const rec &a)
    {
```

```
        return x==a.x&&y==a.y&&z==a.z;
    }
}q[N*8],qq[N*8],tmp[N*8];
int ans[N],tree[N*2],a[N*2],b[N*2],c[N*2],kind[N];
int n,ta,tb,tc;
bool cmp1(rec &a,rec &b)
{
    if(a==b) return a.id<b.id;
    return a.x<b.x||a.x==b.x&&a.y<b.y||a.x==b.x&&a.y==b.y&&a.z<b.z;
}
bool cmp2(rec &a,rec &b)
{
    return a.y<b.y;
}
inline void add(int x,int w)
{
    for(;x<=tc;x+=x&-x) tree[x]+=w;
}
inline int getsum(int x)
{
    int ans=0;
    for(;x;x-=x&-x) ans+=tree[x];
    return ans;
}
void cdq(int l,int r)
{
    if(l==r) return;
    int mid=l+r>>1;
    for(int i=l;i<=r;++i) tmp[i]=qq[i];
    sort(qq+l,qq+mid+1,cmp2);
    sort(qq+mid+1,qq+r+1,cmp2);
    int ll=l;
    for(int i=mid+1;i<=r;++i)
    {
        while(ll<=mid&&qq[ll].y<=qq[i].y)
        {
            if(qq[ll].k==1) add(qq[ll].z,1);
            ++ll;
        }
        if(qq[i].k==2) ans[qq[i].id]+=getsum(qq[i].z)*qq[i].w;
    }
    for(int i=l;i<ll;++i)
        if(qq[i].k==1) add(qq[i].z,-1);
    for(int i=l;i<=r;++i) qq[i]=tmp[i];
```

```
    cdq(l,mid);cdq(mid+1,r);
}
void solve(int l,int r)
{
    if(l==r) return;
    int mid=l+r>>1,n=0;
    for(int i=l;i<=mid;++i)
        if(q[i].k==1) qq[++n]=q[i];
    for(int i=mid+1;i<=r;++i)
        if(q[i].k==2) qq[++n]=q[i];
    if(n)
    {
        sort(qq+1,qq+n+1,cmp1);
        cdq(1,n);
    }
    solve(l,mid);solve(mid+1,r);
}
int main()
{
    int ca;
    scanf("%d",&ca);
    while(ca--)
    {
        scanf("%d",&n);
        int nn=0,k,a1,b1,c1,a2,b2,c2;
        ta=tb=tc=0;
        for(int i=1;i<=n;++i)
        {
            scanf("%d%d%d%d",&k,&a1,&b1,&c1);
            kind[i]=k;
            if(k==1)
            {
                q[++nn]=rec(k,a1,b1,c1,1,i);
                c[++tc]=c1;
            }
            else
            {
                scanf("%d%d%d",&a2,&b2,&c2);
                --a1;--b1;--c1;
                c[++tc]=c1;c[++tc]=c2;
                q[++nn]=rec(k,a2,b2,c2,1,i);
                q[++nn]=rec(k,a1,b2,c2,-1,i);
                q[++nn]=rec(k,a2,b1,c2,-1,i);
                q[++nn]=rec(k,a2,b2,c1,-1,i);
```

```
                q[++nn]=rec(k,a1,b1,c2,1,i);
                q[++nn]=rec(k,a1,b2,c1,1,i);
                q[++nn]=rec(k,a2,b1,c1,1,i);
                q[++nn]=rec(k,a1,b1,c1,-1,i);
            }
        }
        sort(c+1,c+tc+1);tc=unique(c+1,c+tc+1)-c-1;
        for(int i=1;i<=nn;++i)
            q[i].z=lower_bound(c+1,c+tc+1,q[i].z)-c;
        memset(ans,0,sizeof(ans));
        solve(1,nn);
        for(int i=1;i<=n;++i)
            if(kind[i]==2) printf("%d\n",ans[i]);
    }
    return 0;
}
```

3.CDQ 分治三维偏序问题

```
/*
给定 N 个三维的点，问对于点 i，
有多少个点三个坐标都小于点 i(1<=i<=n)
*/
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;
#define N 100010
struct rec{
    int x,y,z,id;
    bool operator==(const rec &a)
    {
        return x==a.x&&y==a.y&&z==a.z;
    }
}a[N],tmp[N];
int ans[N],c[N];
int n;
bool cmp1(rec &a,rec &b)
{
    if(a==b) return a.id<b.id;
    return a.x<b.x||a.x==b.x&&a.y<b.y||a.x==b.x&&a.y==b.y&&a.z<b.z;
}
bool cmp2(rec &a,rec &b)
{
    return a.y<b.y||a.y==b.y&&a.id<b.id;
```

```
}
inline void add(int x,int w)
{
    for(;x<=100000;x+=x&-x) c[x]+=w;
}
inline int getsum(int x)
{
    int ans=0;
    for(;x;x-=x&-x) ans+=c[x];
    return ans;
}
void solve(int l,int r)
{
    if(l==r) return;
    int mid=l+r>>1;
    for(int i=l;i<=r;++i) tmp[i]=a[i];
    sort(a+l,a+mid+1,cmp2);
    sort(a+mid+1,a+r+1,cmp2);
    int ll=l;
    for(int i=mid+1;i<=r;++i)
    {
        while(ll<=mid&&a[ll].y<=a[i].y) add(a[ll++].z,1);
        ans[a[i].id]+=getsum(a[i].z);
    }
    for(int i=l;i<ll;++i) add(a[i].z,-1);
    for(int i=l;i<=r;++i) a[i]=tmp[i];
    solve(l,mid);solve(mid+1,r);
}
int main()
{
    int ca;
    scanf("%d",&ca);
    while(ca--)
    {
        scanf("%d",&n);
        for(int i=1;i<=n;++i)
        {
            scanf("%d%d%d",&a[i].x,&a[i].y,&a[i].z);
            a[i].id=i;
        }
        sort(a+1,a+n+1,cmp1);
        memset(ans,0,sizeof(ans));
        solve(1,n);
        for(int i=n-1;i>0;--i)
```

```
            if(a[i]==a[i+1]) ans[a[i].id]=ans[a[i+1].id];
        for(int i=1;i<=n;++i)
            printf("%d\n",ans[i]);
    }
    return 0;
}
```

4.CDQ 分治二维偏序常见数据结构优化 DP

```
#include <bits/stdc++.h>
using namespace std;
#define N 100010
struct rec{
    int x,y;
    rec(){}
    rec(int x,int y):x(x),y(y){}
}q[N],s,t;
int n;
int a[N];
bool cmp1(rec &a,rec &b)
{
    return a.y<b.y||a.y==b.y&&a.x<b.x;
}
bool cmp2(rec &a,rec &b)
{
    return a.y>b.y||a.y==b.y&&a.x<b.x;
}
int solve(int z)
{
    int cnt=0;
    for(int i=1;i<=z;++i)
    {
        int x=upper_bound(a+1,a+cnt+1,q[i].x)-a;
        if(x>cnt) a[++cnt]=q[i].x;
        else a[x]=q[i].x;
    }
    return cnt;
}
int main()
{
    scanf("%d",&n);
    scanf("%d%d",&s.x,&s.y);
    scanf("%d%d",&t.x,&t.y);
    if(s.x>t.x) swap(s,t);
    int y1=min(s.y,t.y),y2=max(s.y,t.y);
```

```
    int x,y,z=0;
    for(int i=1;i<=n;++i)
    {
        scanf("%d%d",&x,&y);
        if(s.x<=x&&x<=t.x&&y1<=y&&y<=y2)
            q[++z]=rec(x,y);
    }
    if(s.y<t.y) sort(q+1,q+z+1,cmp1);
    else sort(q+1,q+z+1,cmp2);
    printf("%d\n",solve(z));
    return 0;
}
```

5.整体二分--静态区间第 k 小

```
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;
#define N 100010
const int inf=1e9;
struct rec{
    int w,id;
    rec(){}
    rec(int w,int id):w(w),id(id){}
    bool operator<(const rec a)const
    {
        return w<a.w;
    }
}a[N];
struct interval{
    int l,r,k,id,cnt;
}q[N],tmp[N];
int n,m;
int c[N],ans[N];
inline void add(int x,int w)
{
    for(;x<=n;x+=x&-x) c[x]+=w;
}
inline int getsum(int x)
{
    int ans=0;
    for(;x;x-=x&-x) ans+=c[x];
    return ans;
}
```

```
void calc(int s,int t,int l,int r)
{
    int x=lower_bound(a+1,a+n+1,rec(l,0))-a;
    for(int i=x;i<=n&&a[i].w<=r;++i) add(a[i].id,1);
    for(int i=s;i<=t;++i)
        q[i].cnt=getsum(q[i].r)-getsum(q[i].l-1);
    for(int i=x;i<=n&&a[i].w<=r;++i) add(a[i].id,-1);
}
void solve(int s,int t,int l,int r)
{
    if(l==r)
    {
        for(int i=s;i<=t;++i) ans[q[i].id]=l;
        return;
    }
    int mid=l+(r-l>>1);
    calc(s,t,l,mid);
    int ss=s-1,tt=t+1;
    for(int i=s;i<=t;++i)
    {
        if(q[i].cnt>=q[i].k) tmp[++ss]=q[i];
        else q[i].k-=q[i].cnt,tmp[--tt]=q[i];
    }
    for(int i=s;i<=t;++i) q[i]=tmp[i];
    if(ss>=s) solve(s,ss,l,mid);
    if(tt<=t) solve(tt,t,mid+1,r);
}
bool cmp(rec a,rec b)
{
    return a.w<b.w;
}
int main()
{
    while(scanf("%d%d",&n,&m)!=EOF)
    {
        for(int i=1;i<=n;++i)
        {
            scanf("%d",&a[i].w);a[i].id=i;
        }
        sort(a+1,a+n+1,cmp);
        for(int i=1;i<=m;++i)
        {
            scanf("%d%d%d",&q[i].l,&q[i].r,&q[i].k);
            q[i].cnt=0;q[i].id=i;
```

```
        }
        solve(1,m,-inf,inf);
        for(int i=1;i<=m;++i) printf("%d\n",ans[i]);
    }
    return 0;
}
```

6.整体二分--动态区间第 k 小

```
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;
#define N 100010
struct rec{
    int x,y,k,cnt,id;
    rec(){}
    rec(int x,int y,int k,int cnt,int id):
        x(x),y(y),k(k),cnt(cnt),id(id){}
}q[N*3],q1[N*3],q2[N*3];
const int inf=1e9;
int a[N],ans[N*3],c[N];
int n,m,nn;
inline void add(int x,int w)
{
    for(;x<=n;x+=x&-x) c[x]+=w;
}
inline int getsum(int x)
{
    int ans=0;
    for(;x;x-=x&-x) ans+=c[x];
    return ans;
}
void calc(int s,int t,int l,int r)
{
    for(int i=s;i<=t;++i)
    {
        if(q[i].k) q[i].cnt=getsum(q[i].y)-getsum(q[i].x-1);
        else if(q[i].y<=r) add(q[i].x,q[i].cnt);
    }
    for(int i=s;i<=t;++i)
        if(!q[i].k&&q[i].y<=r) add(q[i].x,-q[i].cnt);
}
void solve(int s,int t,int l,int r)
```

```
{
    if(l==r)
    {
        for(int i=s;i<=t;++i)
            if(q[i].k) ans[q[i].id]=l;
        return;
    }
    int mid=l+(r-l>>1);
    calc(s,t,l,mid);
    int t1=0,t2=0;
    for(int i=s;i<=t;++i)
    {
        if(q[i].k)
        {
            if(q[i].cnt>=q[i].k) q1[++t1]=q[i];
            else q[i].k-=q[i].cnt,q2[++t2]=q[i];
        }
        else
        {
            if(q[i].y<=mid) q1[++t1]=q[i];
            else q2[++t2]=q[i];
        }
    }
    for(int i=1;i<=t1;++i) q[s+i-1]=q1[i];
    for(int i=1;i<=t2;++i) q[t-t2+i]=q2[i];
    if(t1) solve(s,s+t1-1,l,mid);
    if(t2) solve(s+t1,t,mid+1,r);
}
int main()
{
    while(scanf("%d",&n)!=EOF)
    {
        for(int i=1;i<=n;++i)
        {
            scanf("%d",&a[i]);
            q[i]=rec(i,a[i],0,1,i);
            ans[i]=0;
        }
        nn=n;
        scanf("%d",&m);
        int x,y,k,z;
        for(int i=1;i<=m;++i)
        {
            scanf("%d%d%d",&z,&x,&y);
```

```
            if(z==1)
            {
                ++nn;q[nn]=rec(x,a[x],0,-1,nn);
                ++nn;q[nn]=rec(x,y,0,1,nn);
                a[x]=y;
                ans[nn]=0;
            }
            else
            {
                scanf("%d",&k);
                ++nn;q[nn]=rec(x,y,k,0,nn);
            }
        }
        memset(ans,0,sizeof(ans));
        solve(1,nn,1,inf);
        for(int i=1;i<=nn;++i)
            if(ans[i]) printf("%d\n",ans[i]);
    }
    return 0;
}
```

7.树状数组离线求区间不同数字个数

```
#include <bits/stdc++.h>
using namespace std;
#define lowbit(x) (x&-x)
#define max(a,b) ((a)>(b)?(a):(b))

const int N=200005, M=200005;
int bit[N], a[N], n, m, ihead[100005], inext[N];
struct Q {
    int l, r, id, ans;
}q[M];
bool cmp1(const Q &a, const Q &b) { return a.l==b.l?a.r<b.r:a.l<b.l; }
bool cmp2(const Q &a, const Q &b) { return a.id<b.id; }

inline int read() {
    int ret=0; char c;
    for(c=getchar(); c<'0' || c>'9'; c=getchar());
    for(; c>='0' && c<='9'; c=getchar()) ret=ret*10+c-'0';
    return ret;
}
inline void add(int x, const int &y) { while(x<=n) bit[x]+=y, x+=lowbit(x); }
inline int sum(int x) { int ret=0; while(x>0) ret+=bit[x], x-=lowbit(x); return ret; }
```

```
int main() {
    while(scanf("%d%d",&n,&m)!=EOF){
    memset(bit,0,sizeof bit);
    memset(ihead,0,sizeof ihead);
    memset(inext,0,sizeof inext);

    int i, maxi=0, l=1;
    for(i=1; i<=n; ++i){
        a[i]=read(), maxi=max(maxi, a[i]);
        a[i+n]=a[i];
    }
    n*=2;
    for(int i=1;i<=n;i++){
        maxi=max(maxi,a[i]);
    }
    for(i=n; i>=0; --i){
        inext[i]=ihead[a[i]], ihead[a[i]]=i;
    }
    for(i=0; i<=maxi; ++i){
        if(ihead[i]) add(ihead[i], 1);
    }
    for(i=1; i<=m; ++i){
        q[i].l=read(), q[i].r=read(), q[i].id=i;
        int tmp=q[i].r;
        q[i].r=n/2+q[i].l;
        q[i].l=tmp;
        //cout<<q[i].l<<" "<<q[i].r<<endl;
    }
    sort(q+1, q+1+m, cmp1);
    for(i=1; i<=m; ++i) {
        while(l<q[i].l) {
            if(inext[l]) add(inext[l], 1);
            ++l;
        }
        q[i].ans=sum(q[i].r)-sum(q[i].l-1);
    }
    sort(q+1, q+1+m, cmp2);
    for(i=1; i<=m; ++i) printf("%d\n", q[i].ans);
}
return 0;
}
```

8.KDtree

```
/*
给定 n 个点，有 m 次询问
1 x y 插入一个节点
2 x y 询问与这个节点最近的点
*/
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;
#define N 500010
const int inf=0x3f3f3f3f;
int D;//当前维
struct Point{
    //d[0/1]表示一维/二维坐标
    //mn[0/1],mx[0/1]表示最小，最大一维/二维坐标
    //l,r 表示左儿子和右儿子
    int d[2],mn[2],mx[2],l,r;
    bool operator==(const Point a)const
    {
        return d[0]==a.d[0]&&d[1]==a.d[1];
    }
    bool operator<(const Point a)const
    {
        return d[D]<a.d[D];
    }
}p[N];
struct KDtree{
    Point t[N*2],P,Q;
    int rt,cnt,ans;
    void updata(int k)
    {
        int l=t[k].l,r=t[k].r;
        for(int i=0;i<2;++i)
        {
            t[k].mn[i]=t[k].mx[i]=t[k].d[i];
            if(l) t[k].mn[i]=min(t[k].mn[i],t[l].mn[i]);
            if(l) t[k].mx[i]=max(t[k].mx[i],t[l].mx[i]);
            if(r) t[k].mn[i]=min(t[k].mn[i],t[r].mn[i]);
            if(r) t[k].mx[i]=max(t[k].mx[i],t[r].mx[i]);
        }
    }
    int build(int l,int r,int f)
    {
```

```
    if(l>r) return 0;
    int mid=l+r>>1;
    D=f;nth_element(p+l,p+mid,p+r+1);
    t[mid]=p[mid];
    t[mid].l=build(l,mid-1,f^1);
    t[mid].r=build(mid+1,r,f^1);
    updata(mid);
    return mid;
}
void insert(int &k,int f,Point &now)
{
    if(!k)
    {
        k=++cnt;
        t[k].d[0]=t[k].mn[0]=t[k].mx[0]=now.d[0];
        t[k].d[1]=t[k].mn[1]=t[k].mx[1]=now.d[1];
        t[k].l=t[k].r=0;
    }
    if(t[k]==now) return;
    if(now.d[f]<t[k].d[f]) insert(t[k].l,f^1,now);
    else insert(t[k].r,f^1,now);
    updata(k);
}
int dis(Point &Q)
{
    int d=0;
    for(int i=0;i<2;++i) d+=abs(Q.d[i]-P.d[i]);
    return d;
}
int getmn(Point &Q)//估计的最小哈密尔顿距离
{
    int mn=0;
    for(int i=0;i<2;++i)
    {
        mn+=max(0,P.d[i]-Q.mx[i]);
        mn+=max(0,Q.mn[i]-P.d[i]);
    }
    return mn;
}
int getmx(Point &Q)//估计的最大哈密尔顿距离
{
    int mx=0;
    for(int i=0;i<2;++i)
        mx+=max(abs(P.d[i]-Q.mn[i]),abs(P.d[i]-Q.mx[i]));
```

```
    return mx;
}
void querymn(int k)//查找最小的哈密尔顿距离
{
    //如果查找的点集中包括自己，则不算自己
    //int tmp=dis(t[k]);
    //if(tmp) ans=min(ans,tmp);
    ans=min(ans,dis(t[k]));
    int l=t[k].l,r=t[k].r,dl=inf,dr=inf;
    if(l) dl=getmn(t[l]);
    if(r) dr=getmn(t[r]);
    if(dl<dr)
    {
        if(dl<ans) querymn(l);
        if(dr<ans) querymn(r);
    }
    else
    {
        if(dr<ans) querymn(r);
        if(dl<ans) querymn(l);
    }
}
void querymx(int k)//查找最大的哈密尔顿距离
{
    ans=max(ans,dis(t[k]));
    int l=t[k].l,r=t[k].r,dl=-inf,dr=-inf;
    if(l) dl=getmx(t[l]);
    if(r) dr=getmx(t[r]);
    if(dl>dr)
    {
        if(dl>ans) querymx(l);
        if(dr>ans) querymx(r);
    }
    else
    {
        if(dr>ans) querymx(r);
        if(dl>ans) querymx(l);
    }
}
/*
f=0 为查找最大,f=1 为查找最小
*/
int query(int f,int x,int y)
{
```

```
        P.d[0]=x;P.d[1]=y;
        if(f==0) {ans=inf;querymn(rt);}
        else {ans=-inf;querymx(rt);}
        return ans;
    }
    /*
    当插入的节点太多时，可能导致树不平衡，此时重建
    */
    int rebuild(int l,int r,int f)
    {
        if(l>r) return 0;
        int mid=l+r>>1;
        D=f;nth_element(t+l,t+mid,t+r+1);
        t[mid].l=rebuild(l,mid-1,f^1);
        t[mid].r=rebuild(mid+1,r,f^1);
        updata(mid);
        return mid;
    }
}T;
int n,m;
int read()
{
    int x=0;char ch=getchar();
    while(ch<'0'||ch>'9') ch=getchar();
    while('0'<=ch&&ch<='9'){x=x*10+ch-'0';ch=getchar();}
    return x;
}
int main()
{
    //freopen("BZOJ2716.in","r",stdin);
    n=read();m=read();
    for(int i=1;i<=n;++i){p[i].d[0]=read();p[i].d[1]=read();}
    T.rt=T.build(1,n,0);T.cnt=n;
    int t,z=n+50000;Point tmp;
    for(int i=1;i<=m;++i)
    {
        t=read();tmp.d[0]=read();tmp.d[1]=read();
        if(t==1)
        {
            T.insert(T.rt,0,tmp);
            if(T.cnt==z){T.rt=T.rebuild(1,T.cnt,0);n+=50000;}
        }
        else printf("%d\n",T.query(0,tmp.d[0],tmp.d[1]));
    }
```

```
    return 0;
}
```

9.树分治 treedp 计数

```
/*
给定一棵 n 个节点的树，树上的权值要么为 0，要么为 1，
如果一条路径上的 0 和 1 数量相同且路径上存在一个点（称为休息点）（非路径的端点），
在这个点的左边路径上 0 和 1 的数量相等，右边路径上的 0 和 1 的数量也相等，
则称这条路径符合条件，问符合条件的路有多少条。
解：先把树上的权值 0 变成-1，
在计数的时候，运用 treedp 的思想，
在当前根中，
la[x][0/1]表示当前根中以前访问的子树到根的路径权值为 x 的没有/有休息点的节点数量
g[x][0/1]表示当前根中现在访问的节点的子树到根的路径权值为 x 的没有/有休息点的节点数量
点的编号为 1~n
*/
#include<iostream>
#include<stdio.h>
#include<algorithm>
#include<string.h>
#include<vector>
#define N 100010
typedef long long ll;
using namespace std;
struct node{
    int x,y;
    node(int xx,int yy)
    {
        x=xx,y=yy;
    }
};
vector<node> lin[N];
vector<int> d;
int sz[N],f[N];//sz[x]-->x 的树大小，f[x]-->x 最大子树的节点数；
int n;
int rt;
int vis[N],dis[N],g[N*2][2],la[N*2][2],s[N*2];
ll ans,sum;
int size;
int aa,bb,cc,md;
void getrt(int x,int fa)//利用*sz,*f 求重心
{
    sz[x]=1;
    f[x]=0;
```

```
    for(int i=0;i<lin[x].size();i++)
    {
        int u=lin[x][i].x;
        if(vis[u]||u==fa) continue;
        getrt(u,x);
        sz[x]+=sz[u];
        f[x]=max(sz[u],f[x]);
    }
    f[x]=max(f[x],size-sz[x]);//!  x 最大子树的节点数=max(与此子树大小－f[x],f[x])
    if(f[x]<f[rt]) rt=x;
}
void getdis(int x,int fa)//一遍 dfs 求距离＋求重心的一点预处理 sz[x],求子树大小 size
{
    sz[x]=1;
    for(int i=0;i<lin[x].size();i++)
    {
        int u=lin[x][i].x;
        if(vis[u]||u==fa) continue;
        getdis(u,x);
        sz[x]+=sz[u];
    }
}
void dfs(int x,int fa,int w)//求出 g[][]
{
    //printf("%d %d ",x,w);
    md=max(md,max(w,-w));
    if(s[w+N]) ++g[w+N][1];
    else ++g[w+N][0];
    //printf("%d %d !!\n",g[w+N][0],g[w+N][1]);
    ++s[w+N];
    int y;
    for(int i=0;i<lin[x].size();++i)
    {
        y=lin[x][i].x;
        if(vis[y]||y==fa) continue;
        dfs(y,x,w+lin[x][i].y);
    }
    --s[w+N];
}
ll cal(int x)
{
    //printf("%d ----------\n",x);
    int y;
    ll tmp=0;
```

```
    int mdd=0;
    for(int i=0;i<lin[x].size();++i)
    {
        y=lin[x][i].x;
        if(vis[y]) continue;
        md=0;
        dfs(y,x,lin[x][i].y);
        mdd=max(mdd,md);
        tmp+=g[N][1]+(ll)g[N][0]*la[N][0];
        for(int j=-md;j<=md;++j)
        {

tmp+=(ll)g[N-j][0]*la[N+j][1]+(ll)g[N-j][1]*la[N+j][0]+(ll)g[N-j][1]*la[N+j][1];

//printf("%d %d %d %d %d %d ??\n",y,j,g[N+j][0],g[N+j][1],la[N+j][0],la[N+j][1]);
        }
        for(int j=N-md;j<=N+md;++j)
        {
            la[j][1]+=g[j][1];
            la[j][0]+=g[j][0];
            g[j][1]=g[j][0]=0;
        }
    }
    for(int i=N-mdd;i<=N+mdd;++i)
        la[i][0]=la[i][1]=0;
    //printf("%d ?\n",tmp);
    return tmp;
}
void solve(int x)
{
    ans+=cal(x);
    vis[x]=1;
    for(int i=0;i<lin[x].size();i++)
    {
        int u=lin[x][i].x;
        if(vis[u]) continue;
        getdis(u,0);
        size=sz[u];//!!! getdis 中已经处理子树的全大小
        getrt(u,rt=0);
        solve(rt);
    }
}
int main()
{
```

```
    while(scanf("%d",&n)!=EOF)
    {
        for(int i=1;i<=n;i++)
        {
            vis[i]=0;
            lin[i].clear();
        }
        for(int i=1;i<n;i++)
        {
            scanf("%d%d%d",&aa,&bb,&cc);
            if(cc==0) cc=-1;
            lin[aa].push_back(node(bb,cc));
            lin[bb].push_back(node(aa,cc));
        }
        ans=0;
        f[0]=n+1;size=n;
        getrt(1,rt=0);
        solve(rt);
        printf("%lld\n",ans);
    }
}
```

10.树分治计数

```
/*
求树上距离小于 K 的点对的数量，
对于一个节点，先算出以它为根的子树的符合条件的点对的数量
再减去以它儿子为根的子树的符合条件的点对的数量，
求点对数量时用双指针
点的编号为 1~n
*/
#include<iostream>
#include<stdio.h>
#include<algorithm>
#include<string.h>
#include<vector>
#define N 10005
using namespace std;
struct node{
    int x,y;
    node(int xx,int yy)
    {
        x=xx,y=yy;
    }
```

```
};
vector<node> lin[N];
vector<int> d;
int sz[N],f[N];//sz[x]-->x 的树大小，f[x]-->x 最大子树的节点数；
int n;
int rt;
int vis[N],dis[N];
int K;
int ans,size;
int aa,bb,cc;
void getrt(int x,int fa)//利用*sz,*f 求重心
{
    sz[x]=1;
    f[x]=0;
    for(int i=0;i<lin[x].size();i++)
    {
        int u=lin[x][i].x;
        if(vis[u]||u==fa) continue;
        getrt(u,x);
        sz[x]+=sz[u];
        f[x]=max(sz[u],f[x]);
    }
    f[x]=max(f[x],size-sz[x]);//!  x 最大子树的节点数=max(与此子树大小－f[x],f[x])
    if(f[x]<f[rt]) rt=x;
}
void getdis(int x,int fa)//一遍 dfs 求距离＋求重心的一点预处理 sz[x],求子树大小 size
{
    sz[x]=1;
    d.push_back(dis[x]);
    for(int i=0;i<lin[x].size();i++)
    {
        int u=lin[x][i].x;
        if(vis[u]||u==fa) continue;
        dis[u]=dis[x]+lin[x][i].y;
        getdis(u,x);
        sz[x]+=sz[u];
    }
}
int cal(int x,int y)
{
    int ret=0;
    d.clear();
    dis[x]=y;
    getdis(x,0);
```

```
    sort(d.begin(),d.end());
    int l=0;
    int r=d.size()-1;
    while(l<r)
    {
        while(d[l]+d[r]>K&l<r) r--;
        ret+=r-l;
        l++;
    }
    return ret;
}
void solve(int x)
{
    //cout<<x<<endl;
    ans+=cal(x,0);
    vis[x]=1;
    for(int i=0;i<lin[x].size();i++)
    {
        int u=lin[x][i].x;
        if(vis[u]) continue;
        ans-=cal(u,lin[x][i].y);
        f[0]=size=sz[u];//!!! getdis 中已经处理子树的全大小
        getrt(u,rt=0);
        solve(rt);
    }

}
int main()
{
    while(scanf("%d%d",&n,&K))
    {
        if(!n&&!K) return 0;
        for(int i=1;i<=n;i++)
        {
            vis[i]=0;
            lin[i].clear();
        }
        for(int i=1;i<n;i++)
        {
            scanf("%d%d%d",&aa,&bb,&cc);
            lin[aa].push_back(node(bb,cc));
            lin[bb].push_back(node(aa,cc));
        }
```

```
        ans=0;
        f[0]=size=n;
        getrt(1,rt=0);
        solve(rt);
        printf("%d\n",ans);
    }
}
```

1.基础

```
//平面几何模板整理
//在原来的内容上进行了一些修改
#include <cstdio>
#include <algorithm>
#include <queue>
#include <cstring>
#include <string>
#include <cmath>
#include <iostream>
using namespace std;
const int maxn=10050;
const double eps=1e-8;//精度范围

//*********平面点，线，简单图形的表示*********
struct point{
    //平面点的表示，坐标值为实数，适用于对时间要求不太高的情况
    //优点是可以进行精确计算，在大部分情况下都是使用这种表示法表示平面上的点
    //缺点是运算起来可能会比较慢
    double x,y;
    point(double x0=0,double y0=0){
        x=x0;y=y0;
    }
};

struct point2{
    int x,y;
```

```
    //如果仅仅是进行定性判断或者是不涉及实数计算的算法（如求凸包），可以使用以下平面点的表示法
    //注意如果涉及到计算的话，最好还是使用坐标值为实数的点表示法
    point2(int x0=0,int y0=0){
        x=x0;y=y0;
    }
};
struct line{
    //待定系数法表示直线，形如 ax+by+c=0 可以表示平面上的一条直线
    //该表示法可以方便利用解析几何的知识解决问题
    //用这种表示法需要记住一些公式，例如两直线交点坐标公式，下面有写
    double a,b,c;
    line(double a0=0,double b0=0,double c0=0){
        a=a0;b=b0;c=c0;
    }
};

struct line2{
    //两点式表示一条直线或线段
    //实际上很多时候不用刻意使用结构体，使用两个 point 数组亦可表示
    point p1,p2;
    line2(point _p1,point _p2){
        p1=_p1;p2=_p2;
    }
};

struct circle{
    //点径式表示一个圆
    //其实有时也不用刻意使用结构体表示圆
    point o;
    double r;
};

typedef point vctor;//将 point 另命名为向量，这样一个点坐标就可表示为一条由坐标原点指向该点坐标的向量

//任意多边形可以使用一组按照顺时针或逆时针给出的点序列表示

//*********点，向量的运算符的重载*********
vctor operator + (vctor a,vctor b){
    //平面两向量相加
    return vctor(a.x+b.x,a.y+b.y);
}
vctor operator - (vctor a,vctor b){
```

```
    //平面两向量相减
    return vctor(a.x-b.x,a.y-b.y);
}
vctor operator * (vctor a,double k){
    //向量的数乘，乘以一个实数
    return vctor(a.x*k,a.y*k);
}
vctor operator / (vctor a,double k){
    //向量的数乘，除以一个实数
    return vctor(a.x/k,a.y/k);
}
bool operator == (point a,point b){
    //判断两点是否为同一点
    return a.x==b.x&&a.y==b.y;
}

point p[maxn],res[maxn],d[maxn],tmp[maxn],q;
int cnt,n,m,r;

//*********点，线的基本操作*********
int sig(double x){
    //返回实数 x 的符号，如果 x 非常接近精确度的范围，那么 x=0
    //否则 x>0 即为正数，返回 1，x<0 返回-1
    return fabs(x)<eps?0:(x>0)?1:-1;
}

double dot(point a,point b,point c){
    //求向量 ab 与向量 ac 的点积
    //就是高中数学的点积，所有高中学过的点积知识均适用
    //以下情况可能会用到点积的运算
    //1.判断夹角是锐角，直角还是钝角，判垂直
    //2.求两向量的夹角
    //3.求点到直线的最短距离
    //具体应用下面会写到
    return (b.x-a.x)*(c.x-a.x)+(b.y-a.y)*(c.y-a.y);
}

double cross(point a,point b,point c){
    //求向量 ab 与向量 ac 的叉积
    //平面向量 ab 与向量 ac 的叉积定义为(sig)*|ab|*|ac|*sin(theta)
    //sig 表示符号，由 ab 与 ac 的位置关系决定
    //如果 ab 在 ac 的顺时针方向，那么 sig=1，ab 在 ac 逆时针方向时 sig=-1，ab 与 ac 共线时 sig=0，
整个叉积为 0
    //sin(theta)表示两向量夹角的正弦
```

```
    //叉积的几何意义为以两向量为邻边组成的平行四边形的有向面积
    //注意是有向面积，意味着可能会出现值为负的面积，除了定性判断或者计算多边形的面积，一般加上绝对值
    //以下情况可能会用到叉积的运算
    //1.判断一个点在一个向量的顺时针方向还是逆时针方向
    //2.判断折线的折向
    //3.求三角形面积，求多边形面积，求点到直线的最短距离
    //4.几乎所有几何算法，如极角排序，凸包，半平面交，旋转卡壳
    //具体应用下面会写到
    return (b.x-a.x)*(c.y-a.y)-(b.y-a.y)*(c.x-a.x);
}

point rotate(point p,double theta){
    //将一个点向逆时针旋转 theta 弧度，返回这个点
    double x=p.x*cos(theta)-p.y*sin(theta);
    double y=p.x*sin(theta)+p.y*cos(theta);
    return point(x,y);
}

point midpoint(point a,point b){
    //中点坐标公式，其实不需要写函数，因为已经重载了运算符
    return (a+b)/2;
}

double dist(point a,point b){
    //求两点距离
    return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
}

double dist2(point a,point b){
    //返回两点距离的平方，有时可以方便计算用
    return (a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y);
}

double dist_point_to_line(point p1,point p2,point p3){
    //求点 p3 到线段 p1p2 的最短距离，解释如下
    if(dot(p1,p2,p3)<0)  return dist(p1,p3);
    //如果向量 p1p3 和向量 p1p2 形成钝角，那么最短距离当然是向量 p1p3 的模
    if(dot(p2,p1,p3)<0)  return dist(p2,p3);
    //如果向量 p2p3 和向量 p2p1 形成钝角，那么最短距离当然是向量 p2p3 的模
    return fabs(cross(p1,p2,p3))/dist(p1,p2);
    //其余情况，我们让 p1,p2,p3 组成一个三角形
    //那么三角形面积除以线段 p1p2 的长就是三角形的高，也就是 p3 到线段 p1p2 的（最短）距离了
}
```

```
double area(point p[],int n){
    //求多边形的面积，转化为求三角形的面积求解
    //不断以原点(0,0)，p[i]和 p[i+1](i=1,2,...,n)作三角形，求出这些三角形的（有向）面积加起来
    //这样求出来的一系列三角形（有向）面积有正有负，但是没关系
    //把所有正的面积减去负的面积，剩下的就是多边形的面积了
    //多边形的顶点必须以顺时针或逆时针给出
    p[n+1]=p[1];
    double s=0;
    for(int i=1;i<=n;i++)s+=cross(point(0,0),p[i],p[i+1]);//s 为面积
    return fabs(s)/2.0;
}

double angle(point a,point b,point c){
    //求向量 ab 和向量 ac 的夹角
    //定性判断只需判断点积符号即可
    return acos(dot(a,b,c)/(dist(a,b)*(dist(a,c))));
}

bool iscross(point p1,point p2,point p3,point p4){
    if(min(p1.x,p2.x)>max(p3.x,p4.x)||
       min(p3.x,p4.x)>max(p1.x,p2.x)||
       min(p1.y,p2.y)>max(p3.y,p4.y)||
       min(p3.y,p4.y)>max(p1.y,p2.y))return 0;
    double r1=cross(p1,p2,p3);
    double r2=cross(p1,p2,p4);
    double r3=cross(p3,p4,p1);
    double r4=cross(p3,p4,p2);
    if(sig(r1*r2)<=0&&sig(r3*r4)<=0)return 1;
    return 0;
}

point cross_point(line l1,line l2){
    //求两直线 l1,l2 交点坐标，前提条件为两直线相交，使用解方程法
    //x=(b1*c2-b2*c1)/(a1*b2-a2*b1)
    //y=(c1*a2-c2*a1)/(a1*b2-a2*b1)
    double a1=l1.a,b1=l1.b,c1=l1.c;
    double a2=l2.a,b2=l2.b,c2=l2.c;
    double x=(b1*c2-b2*c1)/(a1*b2-a2*b1);
    double y=(c1*a2-c2*a1)/(a1*b2-a2*b1);
    return point(x,y);
}
```

```
point cross_point2(point p1,point p2,point p3,point p4){
    //求线段 p1p2 和 p3p4 的交点坐标，前提条件为两线段相交
    //方法同求直线交点坐标，只是直线方程的系数需要自己去求
    double a1=p1.y-p2.y;
    double b1=p2.x-p1.x;
    double c1=p1.x*p2.y-p2.x*p1.y;
    double a2=p3.y-p4.y;
    double b2=p4.x-p3.x;
    double c2=p3.x*p4.y-p4.x*p3.y;
    double x=(b1*c2-b2*c1)/(a1*b2-a2*b1);
    double y=(c1*a2-c2*a1)/(a1*b2-a2*b1);
    return point(x,y);
}

//*********极角排序*********
//对于平面上的所有点，我们取一个点作为原点，以平行于 x 轴坐标轴且向右的射线为极轴
//在这个极坐标平面上，任一点的角度定义为极轴向逆时针旋转的角度
//将所有点的坐标按照角度从小到大排序，就将所有点进行了极角排序
//也可以使用 atan2(y,x)函数，该函数返回一个弧度，取值范围为(-pi,pi]
//它返回的是向量(x,y)与坐标轴形成的夹角。
int cmp(point a,point b){
    //极角排序
    return atan2(a.y,a.x)<atan2(b.y,b.x);
}

//*********凸包算法*********
//给定平面上的一些点，要求一个多边形，使得平面上的所有点或者在这个多边形上，或者在多边形内
//简单地说就是求一个多边形把所有平面上所有点围起来
//有四种算法：卷包裹法，Greham 扫描法，Jarvis 步进法，完美凸包算法
//这里仅给出时间复杂度较好的 Greham 扫描法，算法复杂度为 O(n*logn)
int cmp2(point a,point b){
    //求凸包前必须先将所有点进行坐标排序，
    //排序之后，所有点按照纵坐标从下到上的顺序排序，如果纵坐标相同则按横坐标从左到右的顺序排
    return a.y<b.y||a.y==b.y&&a.x<b.x;
}
int convex(int n,point p[]){
     //该函数传入平面上点的个数和所有点的坐标（已排序）
     //返回凸包上点的个数，同时求出的凸包按逆时针顺序存在数组 res[]内
     int now=1;
     for(int i=1;i<=n;i++){
        while(now>2&&sig(cross(res[now-2],p[i],res[now-1]))>0) now--;
        //res[]数组相当于一个栈
        //首先凸包是一个多边形，至少要有三个点，故 now>2
        //当 res[now-2],p[i],res[now-1]三点按顺序组成的折线的折向向右时，说明 res[now-1]
```

不满足条件

```
            //因为凸包的点要按逆时针给出，而此时三点是顺时针顺序
            //故将 res[now-1]弹出，继续循环直到 res[now-2],p[i],res[now-1]组成的折线折向向左
            //此时三点是逆时针顺序，故 res[]中压入 p[i]
            res[now++]=p[i];
            //循环到满足条件的点时，压入栈中
        }
        //此时已求出半个凸包（从最左下方的点到最右上方的点的一个逆时针凸包）
        //但凸包应该是一个闭合多边形，故还要从右上方到左下方来一遍
        int k=now;
        for(int i=n-1;i>=1;i--){
            while(now>k&&sig(cross(res[now-2],p[i],res[now-1]))>0) now--;
            res[now++]=p[i];
        }
        //此时求出的凸包会多出一个左下角的点，删掉这个点
        if(n>1) now--;
        return now-1;
}

//*********半平面交*********
//如果我们规定直线的一侧（例如右侧）为有效区域（半平面）
//那么平面上的多条直线组合起来就可以划出一块区域
//这块区域称为平面交，求平面交的算法叫做半平面交算法
//直白的说，半平面交就是将一块区域不断地用一条直线进行切割然后扔掉不要的部分
//当用直线不断切割平面时，被切割的区域可能会不断的产生新的点，也可能会有原来在平面交上的点被删掉
//因此这个算法就是对一个点序列进行不断的删除和添加操作
//需要用两个数组 d[]和 tmp[]，先将 d[]求平面交的结果放到 tmp[]里，再将 tmp[]复制给 d[]
//由于篇幅所限，可能有些难理解，如果不好理解就不必深入，直接 CV 代码也可以
void init(int n){
    //初始化平面，向 d[]中放入按顺时针给出的点序列
    //点序列可以是给定的多边形，如该函数就传入了多边形的顶点数
    //也可以是无限大平面（此时 d[]中放入的是四个无穷大的点(inf,inf),(inf,-inf),(-inf,-inf),(-inf,inf)）
    for(int i=1;i<=n;i++)
        d[i]=p[i];
    d[n+1]=d[1];d[0]=d[n];
    //由于切割时可能涉及到当前点的前一个点或后一个点的操作
    //故在 d[0]和 d[n+1]补充上 d[1]的前一个点和 d[n]的后一个点
    m=n;//m 为全局变量，表示当前平面交区域的顶点数
}
void cut(point a,point b){
    //用向量 ab 切割平面以得到一个新的半平面交，规定有效区域为 ab 右边
    //由于这种规定，最后得到的平面交总是顺时针的顶点序列
```

```
    //m 为平面交的顶点的数量
    //tmp[]为临时数组，存放平面交的，最后再复制给 d[]
    int cnt=1;//cnt 为新的半平面交的顶点个数，最后再复制给 m
    for(int i=1;i<=m;i++){
        //判断所有点是否在新的半平面交内，然后对这些点进行一些操作
        double r=cross(a,b,d[i]);
        if(sig(r)<=0)    tmp[cnt++]=d[i];
        //如果 r<=0 说明该点在平面交内，直接放进 tmp[]即可
        else{
            double r1=cross(a,b,d[i-1]);
            if(sig(r1)<0)    tmp[cnt++]=cross_point2(a,b,d[i],d[i-1]);
            //由于 r>=0，故 d[i]在新的平面交外，而这个点的前一个点有可能在新的平面交内
            //这里我们需要判断一下这个点的前一个点是否在平面交内
            //如果是，把这个点和这个点的前一个点作为一条线段
            //然后求这条线段和 ab 的交点，这个交点一定在新的半平面交上，放入 tmp[]
            double r2=cross(a,b,d[i+1]);
            if(sig(r2)<0)    tmp[cnt++]=cross_point2(a,b,d[i],d[i+1]);
            //同样 d[i]的后一个点有可能在新的半平面交内，像上述一样操作
        }
    }
    //接下来就是对 cnt 和 tmp[]的复制操作
    m=cnt-1;
    for(int i=1;i<=m;i++)
        d[i]=tmp[i];
    d[m+1]=d[1];d[0]=d[m];
}
int getcut(int n){
    //p[]中存入了用于求交的直线，用相邻的两个点坐标表示一条直线
    init(n);//先初始化，将所有点放入 d[]中
    for(int i=1;i<=n;i+=2){
        cut(p[i],p[i+1]);//用给出的直线不断的切割
    }
    return m;//返回半平面交的顶点数
}

//*********旋转卡壳*********
//求凸多边形上距离最远的一对顶点，复杂度为 O(n)，一般结合凸包算法计算
//首先选取凸多边形的一条边，一般是 p1p2
//然后从 p3 开始，查找 p3,p4,...,pn
//以 p1p2 为底的三角形 p1p2pi(i=3,4,...,n)的面积是一个单峰函数，也就是说一定有一个最大值
//假设查找到 pk 时三角形面积最大
//那么三角形 p1p2pk 的两条边 p1pk 和 p2pk 中，(p1,pk)和(p2,pk)中就可能有一对点是最远的一对顶点
//用一个 ans 记录一下两对点距离较大的那一对点的距离
```

```
//然后再以 p2p3 为底，从原来的 pk 接着往下找
//由于面积是单峰函数，故一定又可以找到可能为最远顶点对的两对点，用这两对点中距离较大的维护 ans
//然后再以 p3p4 为底……一直到以 pnp1 为底时结束，查找了一圈
//得到的 ans 就是凸多边形距离最远的一对顶点的距离
//由于篇幅所限，不好理解的话，直接照着 CV
//如果多边形顶点个数比较少，暴力 O(n^2)也可以
double caliper(int n){
    //p[]中逆时针给出了凸多边形的顶点，n 为凸多边形的顶点数
    //返回距离最远的一对顶点的距离
    int now=2;
    //now 表示当前查找的点的下标
    double ans=0;
    for(int i=1;i<=n;i++){
        while(sig(cross(p[i],p[i+1],p[now+1])-cross(p[i],p[i+1],p[now]))>0)
    now=now%n+1;
        //如果当前面积大于上一个面积，说明单峰函数还没达到极值，我们让 now=now+1
        //直到当前面积小于等于上一个面积为止，说明三角形的面积已达最大
        //注意取余，查找到最后一个点的下一个时，即回到第一个点
        double d1=dist(p[i],p[now]);
        double d2=dist(p[i+1],p[now]);
        ans=max(ans,max(d1,d2));//ans 维护最大距离
    }
    return ans;
}

//*********pick 定理*********
//皮克定理是一个计算点阵中顶点在格点上的多边形面积公式，该公式可以表示为 2S=2a+b-2
//a 表示多边形内部的点数，b 表示多边形边上的点数，S 表示多边形的面积
//计算多边形内的点的个数，只需将公式变形为 a=(2S-b+2)/2
//设两点 p1(x1,y1),p2(x2,y2)
//则线段 p1p2 的点数=gcd(abs(x1-x2),abs(y1-y2))（注意只计算了线段两个端点中的一个）
//以下是一个计算格点三角形内部点的算法
int gcd(int x,int y){
    //求最大公约数
    return y==0?x:gcd(y,x%y);
}
//这里需要使用坐标为整数的向量的叉积
int cross(point2 a,point2 b,point2 c){
    return (b.x-a.x)*(c.y-a.y)-(c.x-a.x)*(b.y-a.y);
}
int pick(point2 p1,point2 p2,point2 p3){
    //返回格点三角形内部的点数
    //由于是格点三角形，故坐标均为整数
    int s=abs(cross(p1,p2,p3));//求出三角形面积的 2 倍
```

```
    int b1=gcd(abs(p1.x-p2.x),abs(p1.y-p2.y));
    int b2=gcd(abs(p2.x-p3.x),abs(p2.y-p3.y));
    int b3=gcd(abs(p3.x-p1.x),abs(p3.y-p1.y));
    //求出三边上的格点数
    int b=b1+b2+b3;
    return (s-b+2)/2;
}

//*********最小圆覆盖*********
//最小圆覆盖指给定平面上的一些点，求作一个圆使得平面上的所有点均在这个圆内，并且圆的面积最小
//可以发现，这个最小圆至少应该过平面上的两个点
//我我们先用 p[1]和 p[i]做圆，再从 2 到 i-1 判断是否有点不在这个圆上
//如果都在，则说明已经找到覆盖 1 到 i-1 的圆
//如果没有都在，假设我们找到第一个不在这个圆上的点为 p[j]
//于是我们用两个已知点 p[j]与 p[i]去找覆盖 1 到 j-1 的最小覆盖圆
//而对于两个已知点 p[j]与 p[i]求最小覆盖圆
//只要从 1 到 j-1 中，第 k 个点求过 p[k],p[j],p[i]三个点的圆
//再判断 k+1 到 j-1 是否都在圆上，若都在，说明找到圆
//若有不在的，则再用新的点 p[k]更新圆即可
//由于篇幅所限，不必深入，直接 CV 即可
point func(point p1,point p2,point p3){
    //这个函数返回的是三角形的外心，也就是对三角形其中两条边作中垂线得到的交点
    line l1=(p1.x-p2.x,p1.y-p2.y,(dist2(point(0,0),p1)-dist2(point(0,0),p2))/2);
    line l2=(p2.x-p3.x,p2.y-p3.y,(dist2(point(0,0),p2)-dist2(point(0,0),p3))/2);
    return cross_point(l1,l2);
}

void circlecover(){
    q=p[1];r=0;
    //初始化圆心为 p[1]
    for(int i=2;i<=n;i++){
        //先确定过 p[i]的圆
        if(dist(q,p[i])-r>eps){
            q=p[i];r=0;
            for(int j=1;j<=i-1;j++){
                //寻找不在当前圆覆盖上的点 p[j]，确定过两点的圆
                if(dist(q,p[j])-r>eps){
                    q=(p[i]+p[j])/2;r=dist(q,p[j]);
                    for(int k=1;k<=j-1;k++){
                        //寻找不在当前圆覆盖上的点 p[k]，确定过两三点的圆
                        if(dist(q,p[k])-r>eps){
                            q=func(p[i],p[j],p[k]);//求外心，外心即为三角形外接圆的圆
心

                            r=dist(q,p[k]);
```

```
                        }
                    }
                }
            }
        }
    }
}

int main(){

}
```

2.求凸包内核

```
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>
using namespace std;
#define N 1510
const double inf = (1<<18)+10;
struct Point{
    double x,y;
    Point(double a=0,double b=0):x(a),y(b){}
    void f(double &a,double &b){
        a=x;b=y;
    }
    void input(){
        scanf("%lf%lf",&x,&y);
    }
    void output(){
        printf("%.1f %.1f\n",x,y);
    }
}P[N],pp[N];
struct Line{
    Point s,t;
    double angle;
    Line(){}
    Line(Point a,Point b):s(a),t(b){}
    void get_angle(){
        angle=atan2(t.y-s.y,t.x-s.x);
    }
}L[N];
int n,m;
```

```
int que[N];
const double eps=1e-8;
int sig(double x){
    if(x>eps) return 1;
    if(x<-eps) return -1;
    return 0;
}
double cross(Point p0,Point p1,Point p2){
    return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
}
bool cmp(Line l1,Line l2){
    int t=sig(l1.angle-l2.angle);
    if(t>0) return true;
    else if(t<0) return false;
    else if(sig(cross(l1.s,l1.t,l2.s))>0) return true;
    else return false;
}
Point intersect(Line l1,Line l2){
    double u=cross(l2.s,l2.t,l1.s),v=cross(l2.t,l2.s,l1.t);
    Point p;
    p.x=(u*l1.t.x+v*l1.s.x)/(u+v);
    p.y=(u*l1.t.y+v*l1.s.y)/(u+v);
    return p;
}
Line Move(Line l,double r){
    double k,dx,dy,tx,ty;
    dx=l.t.x-l.s.x;
    dy=l.t.y-l.s.y;
    k=r/sqrt(dx*dx+dy*dy);
    tx=k*dy;ty=-k*dx;
    l.s.x+=tx;l.s.y+=ty;
    l.t.x+=tx;l.t.y+=ty;
    return l;
}
bool judge(Line l,Line l1,Line l2){
    Point p=intersect(l1,l2);
    if(sig(cross(l.s,l.t,p))>0) return true;
    else return false;
}
void solve(Line L[],int n){
    sort(L+1,L+n+1,cmp);
    int cnt=1;
    for(int i=2;i<=n;++i)
        if(sig(L[i].angle-L[i-1].angle)!=0)
```

```
            L[++cnt]=L[i];
    n=cnt;
    int head=1,tail=2;
    que[1]=1;que[2]=2;
    for(int i=3;i<=n;++i)
    {
        while(head<tail&&judge(L[i],L[que[tail]],L[que[tail-1]])) --tail;
        while(head<tail&&judge(L[i],L[que[head]],L[que[head+1]])) ++head;
        que[++tail]=i;
    }
    while(head<tail&&judge(L[que[head]],L[que[tail]],L[que[tail-1]])) --tail;
    while(head<tail&&judge(L[que[tail]],L[que[head]],L[que[head+1]])) ++head;
    m=0;
    if(head==tail) return;
    que[tail+1]=que[head];
    for(int i=head;i<=tail;++i)
        P[++m]=intersect(L[que[i]],L[que[i+1]]);
}
double area(Point P[],int n){
    P[n+1]=P[1];
    double ans=0;
    for(int i=1;i<=n;++i)
        ans+=cross(P[1],P[i],P[i+1]);
    return fabs(-ans/2);
}

void reverse(Point P[],int n){
    for(int i=1;i<=n/2;i++){
        swap(P[i],P[n-i+1]);
    }
    P[n+1]=P[1];
}

int main(){
    int t;scanf("%d",&t);
    while(t--){
        scanf("%d",&n);
        for(int i=1;i<=n;i++)pp[i].input();
        if(area(pp,n)<0){
            reverse(pp,n);
        }
        pp[n+1]=pp[1];
        for(int i=1;i<=n;i++){
            L[i]=Line(pp[i],pp[i+1]);L[i].get_angle();
```

```
        }
        solve(L,n);
        if(m<3)puts("0.00");
        else printf("%.2f\n",area(P,m)+eps);
    }
    return 0;
}
```

3.卷包裹算法

```
bool cmp2(point a,point b){
    point c=tmp;
    if(cross(c,a,b)==0)
        return a.x<b.x;
    else return cross(c,a,b)>0;
}

int main(){
    int t;read(t);
    point p[100];
    while(t--){
        int n;read(n);
        double miny=105;
        int id=1;
        for(int i=1;i<=n;i++){
            read(p[i].id,p[i].x,p[i].y);
            if(miny>p[i].y){
                miny=p[i].y;
                id=i;
            }
        }
        swap(p[id],p[1]);
        for(int i=2;i<=n;i++){
            tmp=p[i-1];
            sort(p+i,p+1+n,cmp2);
        }
        print(n);
        for(int i=1;i<=n;i++){
            print(' ',p[i].id);
        }print('\n');
    }
    return 0;
}
```

4.安德鲁算法求凸包

```
#include<cstdio>
#include<cmath>
#include<algorithm>
using namespace std;
const double eps=1e-7;
const int maxn=105;
int n;
struct point{
    double x,y;
    point() {}
    point(double a,double b):x(a),y(b) {}
    bool operator<(const point&b)const{
        if (x<b.x) return 1;
        if (x>b.x) return 0;
        return y<b.y;
    }
    point operator-(const point&b) {return point(x-b.x,y-b.y);}
}a[maxn],stk[maxn];
typedef point vec;
int dcmp(double x){
    if (fabs(x)<=eps) return 0;
    return x>0?1:-1;
}
double getdst(point a,point b){
    return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
}
double cross(vec a,vec b){
    return a.x*b.y-a.y*b.x;
}
int Andrew(){
    sort(a+1,a+1+n);
    int len=0;
    for (int i=1;i<=n;i++){
        while (len>1&&dcmp(cross(stk[len]-stk[len-1],a[i]-stk[len-1]))==-1) len--;
        stk[++len]=a[i];
    }
    int k=len;
    for (int i=n-1;i>=1;i--){
        while (len>k&&dcmp(cross(stk[len]-stk[len-1],a[i]-stk[len-1]))==-1) len--;
        stk[++len]=a[i];
    }
    return len;
}
int main(){
```

```
    for (scanf("%d",&n);n;scanf("%d",&n)){
        for (int i=1;i<=n;i++) scanf("%lf%lf",&a[i].x,&a[i].y);
        int t=Andrew();
        double ans=0;
        for (int i=1;i<t;i++) ans+=getdst(stk[i],stk[i+1]);
        printf("%.2lf\n",n==2?ans/2:ans);
    }
    return 0;
}
```

5.求凸包宽度

```
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <iostream>
#include <cmath>
using namespace std;
#define N 550010
struct Point{
    double x,y;
    Point(){}
    Point(double _x,double _y):x(_x),y(_y){}
    bool operator<(const Point &a)const
    {
        return x<a.x||x==a.x&&y<a.y;
    }
    bool operator==(const Point &a)const
    {
        return x==a.x&&y==a.y;
    }
    void input()
    {
        scanf("%lf%lf",&x,&y);
    }
}P[N],sta[N];
int n;
const double eps=1e-10;
double cross(Point p0,Point p1,Point p2)
{
    return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
}
Point pp[10];
double area(Point P[],int n)
{
```

```
    double ans=0;
    P[n]=P[0];
    for(int i=0;i<n;++i)
        ans+=cross(P[0],P[i],P[i+1]);
    return -ans/2;
}
void Reverse(Point P[],int n)
{
    for(int i=0;i<(n+1)/2;++i)
        swap(P[i],P[n-i-1]);
}
double dis2(Point p1,Point p2)
{
    return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y));
}
double solve(Point P[],int n)
{
    int i1=0,i2=0;
    P[n]=P[0];
    for(int i=1;i<n;++i)
    {
        if(P[i].y>P[i1].y) i1=i;
        if(P[i].y<P[i2].y) i2=i;
    }
    double ans=1e18,tmp;
    for(int i=0;i<n;++i)
    {
        while(tmp=cross(P[i1],P[i2],P[i1+1])-cross(P[i1],P[i2+1],P[i1+1])<-eps)
            i2=(i2+1)%n;
        pp[0]=P[i1];pp[1]=P[i1+1];pp[2]=P[i2];
        if(i1==i2||i1+1==i2)continue;
        double q1 = area(pp,3)*2/dis2(pp[0],pp[1]);
        ans=min(ans,q1);
        i1=(i1+1)%n;
    }
    return ans;
}
void Graham()
{
    sort(P,P+n);
    int top=1;
    sta[0]=P[0];sta[1]=P[1];
    for(int i=2;i<n;++i)
    {
```

```
        while(top&&cross(sta[top-1],sta[top],P[i])>-eps) --top;
        sta[++top]=P[i];
    }
    int k=top;
    for(int i=n-2;i>=0;--i)
    {
        while(top>k&&cross(sta[top-1],sta[top],P[i])>-eps) --top;
        sta[++top]=P[i];
    }
    if(top) top--;
    for(int i=0;i<=top;++i) P[i]=sta[i];
    n=top+1;
}
double R;
int main()
{
    while(scanf("%d%lf",&n,&R)!=EOF)
    {
        for(int i=0;i<n;++i) P[i].input();
        if(area(P,n)<0) Reverse(P,n);
        Graham();
        printf("%.10f\n",solve(P,n)+eps);
    }
    return 0;
}
```

6.Simpson 多圆面积并

```
#include<stdio.h>
#include<math.h>
#include<algorithm>
using namespace std;
typedef double du;
const du pi2=M_PI*2;
struct point{
    du x,y;
    point(du a=0,du b=0){x=a;y=b;}
    du len(){return sqrt(x*x+y*y);}
};
point operator-(point a,point b){return point(a.x-b.x,a.y-b.y);}
bool operator<(point a,point b){return a.x==b.x?a.y<b.y:a.x<b.x;}
bool operator==(point a,point b){return a.x==b.x&&a.y==b.y;}
struct circle{
    point o;
```

```
    du r;
    du                    oint(du                    t1,du                    t2){return
r*(r*(t2-t1)+o.x*(sin(t2)-sin(t1))-o.y*((cos(t2)-cos(t1))));}
}c[1010];
bool operator<(circle a,circle b){return a.o==b.o?a.r<b.r:a.o<b.o;}
bool operator==(circle a,circle b){return a.o==b.o&&a.r==b.r;}
struct angle{
    du a;
    int d;
    angle(du x=0,int y=0){a=x;d=y;}
}p[2010];
bool operator<(angle a,angle b){return a.a<b.a;}
int n;
du solve(int u){
    int i,M,cnt;
    du len,an,f,l,r,s;
    point d;
    cnt=0;
    M=0;
    for(i=1;i<=n;i++){
        if(i!=u){
            d=c[i].o-c[u].o;
            len=d.len();
            if(c[u].r+len<=c[i].r)return 0;
            if(c[u].r>=c[i].r+len||len>=c[i].r+c[u].r)continue;
            an=atan2(d.y,d.x);
            f=acos((c[u].r*c[u].r+len*len-c[i].r*c[i].r)/(2*c[u].r*len));
            l=an-f;
            r=an+f;
            if(l<-M_PI)l+=pi2;
            if(r>M_PI)r-=pi2;
            if(l>r)cnt++;
            p[++M]=angle(l,1);
            p[++M]=angle(r,-1);
        }
    }
    p[0]=-M_PI;
    p[++M]=M_PI;
    sort(p+1,p+M+1);
    s=0;
    for(i=1;i<=M;cnt+=p[i++].d){
        if(cnt==0)s+=c[u].oint(p[i-1].a,p[i].a);
    }
    return s*.5;
```

```
}
int main(){
    int i;
    du s;
    scanf("%d",&n);
    for(i=1;i<=n;i++)scanf("%lf%lf%lf",&c[i].o.x,&c[i].o.y,&c[i].r);
    sort(c+1,c+n+1);
    n=unique(c+1,c+n+1)-c-1;
    for(i=1;i<=n;i++)s+=solve(i);
    printf("%.3lf",s);
}
```

7.多边形重心

```
#include <cstdiO>
#include <cstring>
#include <algorithm>
#include <cmath>
using namespace std;
#define N 1000010
struct Point{
    double x,y;
    Point(){}
    Point(double _x,double _y):x(_x),y(_y){}
    void input()
    {
        scanf("%lf%lf",&x,&y);
    }
}P[N];
const double eps=1e-8;
int n;
double cross(Point p0,Point p1,Point p2)
{
    return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
}
//求多边形重心的思路把多边形划分为三角形，算出三角形的重心
//三角形的质量都集中在重心，可以把三角形看成一个点
//然后就变成了求一些离散的质点的重心，套用重心公式即可
Point center(Point P[],int n)
{
    P[n]=P[0];
    double area,sum_area=0,sum_x=0,sum_y=0;
    for(int i=0;i<n;++i)
    {
        area=cross(P[0],P[i],P[i+1])/2;
```

```
        sum_area+=area;
        sum_x+=(P[0].x+P[i].x+P[i+1].x)/3*area;
        sum_y+=(P[0].y+P[i].y+P[i+1].y)/3*area;
    }
    return Point(sum_x/sum_area,sum_y/sum_area);
}
int main()
{
    int ca;
    scanf("%d",&ca);
    while(ca--)
    {
        scanf("%d",&n);
        for(int i=0;i<n;++i) P[i].input();
        Point p;
        p=center(P,n);
        printf("%.2f %.2f\n",p.x+eps,p.y+eps);
    }
    return 0;
}
```

8.最近点对

```
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>
using namespace std;
#define N 100010
const double inf=1e20;
struct Point{
    double x,y;
}a[N];
int tmp[N];
int n;
bool cmpxy(Point &p1,Point &p2)
{
    return (p1.x<p2.x)||(p1.x==p2.x&&p1.y<p2.y);
}
bool cmpy(int i,int j)
{
    return a[i].y<a[j].y;
}
double dis(Point &p1,Point &p2)
{
```

```
    return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y));
}
double closest_pair(int l,int r)
{
    double d=inf;
    if(l==r) return d;
    if(l+1==r) return dis(a[l],a[r]);
    int mid=l+r>>1;
    double d1=closest_pair(l,mid);
    double d2=closest_pair(mid+1,r);
    d=min(d1,d2);
    int k=0;
    for(int i=l;i<=r;++i)
        if(fabs(a[i].x-a[mid].x)<=d)
            tmp[++k]=i;
    sort(tmp+1,tmp+k+1,cmpy);
    for(int i=1;i<k;++i)
        for(int j=i+1;j<=k&&a[tmp[j]].y-a[tmp[i]].y<d;++j)
            d=min(d,dis(a[tmp[i]],a[tmp[j]]));
    return d;
}
int main()
{
    while(scanf("%d",&n),n)
    {
        for(int i=1;i<=n;++i)
            scanf("%lf%lf",&a[i].x,&a[i].y);
        sort(a+1,a+n+1,cmpxy);
        double ans=closest_pair(1,n);
        if(ans<=10000) printf("%.4f\n",ans);
        else printf("INFINITY\n");
    }
    return 0;
}
```

9.最远点对

```
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>
using namespace std;
#define N 50010
struct Point{
    double x,y;
```

```
    Point(){}
    Point(double _x,double _y):x(_x),y(_y){}
    bool operator<(const Point &a)const
    {
        return x<a.x||x==a.x&&y<a.y;
    }
    void input()
    {
        scanf("%lf%lf",&x,&y);
    }
}P[N],sta[N];
int n;
const double eps=1e-8;
double cross(Point p0,Point p1,Point p2)
{
    return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
}
double area(Point P[],int n)
{
    double ans=0;
    P[n]=P[0];
    for(int i=0;i<n;++i)
        ans+=cross(P[0],P[i],P[i+1]);
    return -ans/2;
}
void Reverse(Point P[],int n)
{
    for(int i=0;i<(n+1)/2;++i)
        swap(P[i],P[n-i-1]);
}
double dis2(Point p1,Point p2)
{
    return (p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y);
}
double solve(Point P[],int n)
{
    int i1=0,i2=0;
    P[n]=P[0];
    for(int i=1;i<n;++i)
    {
        if(P[i].y>P[i1].y) i1=i;
        if(P[i].y<P[i2].y) i2=i;
    }
    double ans=0,tmp;
```

```
    for(int i=0;i<n;++i)
    {
        while(tmp=cross(P[i1],P[i2],P[i1+1])-cross(P[i1],P[i2+1],P[i1+1])<-eps)
            i2=(i2+1)%n;
        ans=max(ans,max(dis2(P[i1],P[i2]),dis2(P[i1+1],P[i2+1])));
        i1=(i1+1)%n;
    }
    return ans;
}
void Graham()
{
    sort(P,P+n);
    int top=1;
    sta[0]=P[0];sta[1]=P[1];
    for(int i=2;i<n;++i)
    {
        while(top&&cross(sta[top-1],sta[top],P[i])>-eps) --top;
        sta[++top]=P[i];
    }
    int k=top;
    for(int i=n-2;i>=0;--i)
    {
        while(top>k&&cross(sta[top-1],sta[top],P[i])>-eps) --top;
        sta[++top]=P[i];
    }
    if(top) top--;
    for(int i=0;i<=top;++i) P[i]=sta[i];
    n=top+1;
}
int main()
{
    while(scanf("%d",&n)!=EOF)
    {
        for(int i=0;i<n;++i) P[i].input();
        if(area(P,n)<0) Reverse(P,n);
        Graham();
        printf("%.0f\n",solve(P,n));
    }
    return 0;
}
```

10.凸包最大三角形

```
#include <cstdio>
#include <cstring>
#include <algorithm>
```

```
#include <cmath>
#include <set>
#include <utility>
using namespace std;
#define N 50010
struct Point{
    double x,y;
    Point(){}
    Point(double _x,double _y):x(_x),y(_y){}
    bool operator<(const Point &a)const
    {
        return x<a.x||x==a.x&&y<a.y;
    }
    void input()
    {
        scanf("%lf%lf",&x,&y);
    }
}P[N],sta[N];
int n;
bool f[N];
const double eps=1e-8;
double cross(Point p0,Point p1,Point p2)
{
    return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
}
double solve()
{
    if(n<3) return 0;
    double ans=0;
    int i=0,j,k;
    P[n]=P[0];
    for(int i=0;i<n;++i)
    {
        j=(i+1)%n;k=(j+1)%n;
        while(i!=k)
        {
            while(j!=k&&cross(P[i],P[k],P[j])-cross(P[i],P[k],P[j+1])<-eps)
                j=(j+1)%n;
            ans=max(ans,cross(P[i],P[k],P[j])/2);
            k=(k+1)%n;
        }
    }
    return ans;
}
```

```
void Graham()
{
    sort(P,P+n);
    int top=1;
    sta[0]=P[0];sta[1]=P[1];
    for(int i=2;i<n;++i)
    {
        while(top&&cross(sta[top-1],sta[top],P[i])>-eps) --top;
        sta[++top]=P[i];
    }
    int k=top;
    for(int i=n-2;i>=0;--i)
    {
        while(top>k&&cross(sta[top-1],sta[top],P[i])>-eps) --top;
        sta[++top]=P[i];
    }
    if(top) --top;
    for(int i=0;i<=top;++i) P[i]=sta[i];
    n=top+1;
}
int main()
{
    //freopen("in.txt","r",stdin);
    //freopen("out.txt","w",stdout);
    while(scanf("%d",&n),n!=-1)
    {
        for(int i=0;i<n;++i) P[i].input();
        Graham();
        printf("%.2f\n",solve());
    }
    return 0;
}
```

11.最小矩形覆盖

```
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>
using namespace std;
#define N 50010
const double eps=1e-12;
struct Point{
    double x,y,angle;
```

```
    Point(){}
    Point(double _x,double _y):x(_x),y(_y){}
    void input()
    {
        scanf("%lf%lf",&x,&y);
    }
    void output()
    {
        printf("%.5f %.5f\n",x+eps,y+eps);
    }
    void f(double &a,double &b)
    {
        a=x;b=y;
    }
    bool operator<(const Point &a)const
    {
        return y<a.y||y==a.y&&x<a.x;
    }
    Point operator-(const Point &a)const
    {
        return Point(x-a.x,y-a.y);
    }
    double operator*(const Point &a)const
    {
        return x*a.x+y*a.y;
    }
    bool operator!=(const Point &a)const
    {
        return x!=a.x||y!=a.y;
    }
    Point operator+(const Point &a)const
    {
        return Point(x+a.x,y+a.y);
    }
}P[N],sta[N],Q[5];
//Q 数组记录最小矩形的四个点
int n;
const double inf=1e20;
double cross(Point p0,Point p1,Point p2)
{
    return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
}
void Graham()
{
```

```
    sort(P,P+n);
    int cnt=0;
    for(int i=1;i<n;++i)
        if(P[i]!=P[i-1]) P[++cnt]=P[i];
    int top=1;
    sta[0]=P[0];sta[1]=P[1];
    for(int i=2;i<=cnt;++i)
    {
        while(top&&cross(sta[top-1],sta[top],P[i])>-eps) --top;
        sta[++top]=P[i];
    }
    int k=top;
    for(int i=cnt-1;i>=0;--i)
    {
        while(top>k&&cross(sta[top-1],sta[top],P[i])>-eps) --top;
        sta[++top]=P[i];
    }
    if(top) --top;
    for(int i=0;i<=top;++i) P[i]=sta[i];
    n=top+1;
}
double dis(Point p1,Point p2)
{
    return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y));
}
double dis_lp(Point p1,Point p2,Point p3)
{
    return fabs(cross(p1,p2,p3))/dis(p1,p2);
}
Point intersect(Point p1,Point p2,Point p3)//过点 p3 与向量 p1p2 垂直的向量与直线 p1p2 的
交点
{
    double x1,x2,x3,y1,y2,y3,d,dx,dy,a1,b1,c1,a2,b2,c2;
    p1.f(x1,y1);p2.f(x2,y2);p3.f(x3,y3);
    a1=x2-x1;b1=y2-y1;c1=x3*(x2-x1)+y3*(y2-y1);
    a2=y2-y1;b2=x1-x2;c2=x1*y2-x2*y1;
    d=a1*b2-a2*b1;dx=c1*b2-c2*b1;dy=a1*c2-a2*c1;
    return Point(dx/d,dy/d);
}
double solve()
{
    P[n]=P[0];
    int i1=0,i2=0;
    for(int i=1;i<n;++i)
```

```
    {
        if(P[i].y<P[i1].y) i1=i;
        if(P[i].y>P[i2].y) i2=i;
    }
    int i3=i1+1,i4=i2+1;
    double ans=inf,a,h;
    for(int i=0;i<n;++i)
    {
        while(cross(P[i1],P[i2],P[i1+1])-cross(P[i1],P[i2+1],P[i1+1])<-eps)
            i2=(i2+1)%n;
        h=dis_lp(P[i1],P[i1+1],P[i2]);
        while((P[i1+1]-P[i1])*(P[i3+1]-P[i3])>eps) i3=(i3+1)%n;
        while((P[i1+1]-P[i1])*(P[i4+1]-P[i4])<-eps) i4=(i4+1)%n;
        a=fabs((P[i1+1]-P[i1])*(P[i3]-P[i4]))/dis(P[i1+1],P[i1]);
        if(ans>a*h)
        {
            ans=a*h;
            Q[0]=intersect(P[i1],P[i1+1],P[i3]);
            Q[1]=intersect(P[i1],P[i1+1],P[i4]);
            Point p=P[i2]+(P[i1]-P[i1+1]);
            Q[2]=intersect(P[i2],p,P[i3]);
            Q[3]=intersect(P[i2],p,P[i4]);
        }
        i1=(i1+1)%n;
    }
    return ans;
}
bool cmp(Point p1,Point p2)
{
    return p1.angle<p2.angle;
}
int main()
{
    scanf("%d",&n);
    for(int i=0;i<n;++i) P[i].input();
    Graham();
    if(n==1)
    {
        printf("%.5f\n",0);
        for(int i=1;i<=4;++i)
            P[0].output();
    }
    else if(n==2)
    {
```

```
        printf("%.5f\n",0);
        sort(P,P+1);
        P[0].output();P[0].output();
        P[1].output();P[1].output();
    }
    else
    {
        printf("%.5f\n",solve()+eps);
        sort(Q,Q+4);
        for(int i=1;i<4;++i)
            Q[i].angle=atan2(Q[i].y-Q[0].y,Q[i].x-Q[0].x);
        sort(Q+1,Q+4,cmp);
        for(int i=0;i<4;++i)
            Q[i].output();
    }
}
```