模版

高级数据结构：

RMQ,存的数组为从 0 开始：

```
#include<iostream>
using namespace std;
int big[50010][20],small[50100][20],arr[50010],n;
void mb();
void ms();
int qb(int a,int b);
int qs(int a,int b);
int main(){
    int m,a,b,i;
    scanf("%d %d",&n,&m);
    for(i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    mb();
    ms();
    for(i=0;i<m;i++){
        scanf("%d %d",&a,&b);
        a--;b--;
        printf("%d\n",qb(a,b)-qs(a,b));
    }
    return 0;
}
void mb(){
    int i,j;
    for(i=0;i<n;i++)big[i][0]=i;
    for(j=1;(1<<j)<=n;j++){
        for(i=0;i+(1<<j)<=n;i++){
            int p=big[i][j-1],q=big[i+(1<<(j-1))][j-1];
            big[i][j]=(arr[p]<arr[q]?q:p);
        }
    }
}
int qb(int i,int j){
    int k;
    for(k=0;(1<<k)<=(j-i+1);k++);k--;
    i=big[i][k],j=big[j-(1<<k)+1][k];
    return (arr[i]<arr[j]?arr[j]:arr[i]);
}
void ms(){
    int i,j;
```

```
    for(i=0;i<n;i++)small[i][0]=i;
    for(j=1;(1<<j)<=n;j++){
        for(i=0;i+(1<<j)<=n;i++){
            int p=small[i][j-1],q=small[i+(1<<(j-1))][j-1];
            small[i][j]=(arr[p]<arr[q]?p:q);
        }
    }
}
int qs(int i,int j){
    int k;
    for(k=0;(1<<k)<=(j-i+1);k++);k--;
    i=small[i][k],j=small[j-(1<<k)+1][k];
    return (arr[i]<arr[j]?arr[i]:arr[j]);
}
////////////////////////////////////////////////////////////////////
/**//*
*二维 RMQ ST 算法
*构造 RMQ 数组 makermq(int n,int m,int b[][]) O(n*m*log(n)*log(m))算法复杂度
*dp2[row][col][i][j] 表示 行从 row ->row +2^i-1 列从 col ->col +2^j-1 二维区间
里最大值
*dp2[row][col][i][j] = 下行
*max{dp2[row][col][i][j-1],dp2[row][col][i-1][j],dp2[row][col+2^(j-1)][i][j-1],dp2[ro
w+2^(i-1)][col][i-1][j]}
*查询 RMQ rmq(int sx,int ex,int sy,int ey)
*同一维的将 sx->ex 分为两个 2^kx 区间 将 sy->ey 分为两个 2^ky 的区间
*kx=(int)log2(ex-sx+1) ky=(int)log2(ey-sy+1)
*查询结果为
*max{dp2[sx][sy][kx][ky],dp2[sx][ey-2^ky+1][kx][ky],dp2[ex-2^kx+1][sy][kx][ky],
dp2[ex-2^kx+1][ey-2^ky+1][kx][ky]}
*/

void makermq(int n,int m,int b[][MAXM]){
    int row,col,i,j;
    for(row=1;row<=n;row++)
        for(col=1;col<=m;col++)
            dp2[row][col][0][0]=b[row][col];
    for(i=0;(1<<i)<=n;i++)
        for(j=0;(1<<j)<=m;j++)
        {
            if(i==0&&j==0) continue;
            for(row=1;row+(1<<i)-1<=n;row++)
                for(col=1;col+(1<<j)-1<=m;col++)
                {
                if(i==0)
```

```
dp2[row][col][i][j]=max(dp2[row][col][i][j-1],dp2[row][col+(1<<(j-1))][i][j-1]);
                        else

dp2[row][col][i][j]=max(dp2[row][col][i-1][j],dp2[row+(1<<(i-1))][col][i-1][j]);
                        }
                }
}
int rmq(int  sx,int  ex,int  sy,int  ey)
{
int kx=(int)(log((ex-sx+1)*1.0)/log(2.0)),ky=(int)(log((ey-sy+1)*1.0)/log(2.0));
return
max(max(dp2[sx][sy][kx][ky],dp2[sx][ey-(1<<ky)+1][kx][ky]),max(dp2[ex-(1<<kx)+
1][sy][kx][ky],dp2[ex-(1<<kx)+1][ey-(1<<ky)+1][kx][ky]));
}
```

//

树状数组

【一维树状数组】支持元素动态改变的求 a[0..n]的和。运算符必须满足减法原则。

```
typedef int elem_t;
inline  int lowbit(int  t){ return  t&(t^(t-1));}
struct Sum{
    elem_t a[M],c[M],ret;
    int n,i;
    void init(int  x){                  //坐标范围 1..n
        n=x;      memset(a,0,sizeof(a[0])*(n+1));    memset(c,0,sizeof(c[0])*(n+1));
    }
    void update(int  x,elem_t v){
        a[x]+=v;                  //加上 v，可改成更新为 v，即 v-=a[x], a[x]+=v;
        for(i=x;i<=n;i+=lowbit(i))  c[i]+=v;
    }
    elem_t query(int  x){          //a[0]+..+a[i];
        for(ret=0,i=x;i>0;i^=lowbit(i))  ret+=c[i];
        return  ret;
    }
}
```

二维树状数组

```
//求 sum{a[1..m][1..n]}
//维护和查询复杂度均为 O(logm*logn)
//用于动态求子阵和,数组内容保存在 sum.a[][]中
//可以改成其他数据类型
struct Square{
    elem_t a[M][M],c[M][M],ret;
    int m,n,i,j;
    void init(int  x,int y){
```

```
    m=x,n=y;
memset(a,0,sizeof(a[0][0])*(m+1)*M);memset(c,0,sizeof(c[0][0])*(m+1)*M);
    }
    void update(int  x,int y,elem_t v){
        a[x][y]+=v;          //加上 v，可改成更新为 v，即 v-=a[x][y], a[x][y]+=v;
        for (i=x;i<=m;i+=lowbit(i))
            for (j=y;j<=n;j+=lowbit(j))  c[i][j]+=v;
    }
    elem_t query(int  x,int y){
        for (ret=0,i=x;i>0;i^=lowbit(i))
            for (j=y;j>0;j^=lowbit(j))  ret+=c[i][j];
        return ret;
    }
}
```

线段树：

矩形并得周长线段树：

```
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;
#define MAXN 1000000
const __int64 mininf=1000000;
const __int64 maxinf=-1000000;
struct Line
{
    __int64 count,len;//count 记录线段覆盖次数,len 该段线段长度
    __int64 l,r;//线段左右两端点
    int lbc,rbc;//线段左右两端点被覆盖的次数,0 表示未被覆盖
    __int64 nseg;//该线段中连续线段个数.
};
struct node
{
    __int64 x,down,up;//扫描线,down 线的下端点,up 线的上端点,x 扫描线的 x 位置
    bool flag;//表示是矩形的左边线还是右边线
```

```
    node(__int64 tx,__int64 td,__int64 tu,bool
tf):x(tx),down(td),up(tu),flag(tf)
    {
    }
 };
 __int64 miny,maxy;
 vector< node > vec;
 Line Ltree[MAXN];
 bool operator< (const node& a,const node& b)
 {
    return a.x<b.x;
 }
 void build(__int64 l,__int64 r,int step)
 {
    Ltree[step].count=0;Ltree[step].len=0;
    Ltree[step].lbc=false;Ltree[step].rbc=false;
    Ltree[step].nseg=0;
    Ltree[step].r=r;Ltree[step].l=l;
    if(r-l>1)
    {
       build(l,(l+r)/2,2*step);
       build((l+r)/2,r,2*step+1);
    }
 }
 void update(int step)
 {
    if(Ltree[step].count>0)
    {
       Ltree[step].len=Ltree[step].r-Ltree[step].l;
       Ltree[step].nseg=1;
    }
    else
       if(Ltree[step].r-Ltree[step].l>1)
       {

Ltree[step].len=Ltree[2*step].len+Ltree[2*step+1].le
n;

Ltree[step].nseg=Ltree[2*step].nseg+Ltree[2*step+1].
nseg;

if(Ltree[step].nseg&&Ltree[2*step].rbc&&Ltree[2*step
+1].lbc)
              Ltree[step].nseg--;
```

```
        }
        else
        {
            Ltree[step].len=0;
            Ltree[step].nseg=0;
        }
    }
    void insert(__int64 l,__int64 r,int step)
    {
        if(Ltree[step].l==l) Ltree[step].lbc++;
        if(Ltree[step].r==r) Ltree[step].rbc++;
        if(l==Ltree[step].l&&Ltree[step].r==r)
            Ltree[step].count++;
        else
        {
            __int64 mid=(Ltree[step].l+Ltree[step].r)/2;
            if(r<=mid)
                insert(l,r,2*step);
            else
                if(l>=mid)
                    insert(l,r,2*step+1);
                else
                {
                    insert(l,mid,2*step);
                    insert(mid,r,2*step+1);
                }
        }
        update(step);
    }
    void del(__int64 l,__int64 r,int step)
    {
        if(Ltree[step].l==l) Ltree[step].lbc--;
        if(Ltree[step].r==r) Ltree[step].rbc--;
        if(l==Ltree[step].l&&Ltree[step].r==r)
            Ltree[step].count--;
        else
        {
            __int64 mid=(Ltree[step].l+Ltree[step].r)/2;
            if(r<=mid)
                del(l,r,2*step);
            else
                if(l>=mid)
                    del(l,r,2*step+1);
                else
```

```
            {
                del(l,mid,2*step);
                del(mid,r,2*step+1);
            }
        }
        update(step);
    }

    int main()
    {
        int n;
        __int64 x1,x2,y1,y2;
        while(scanf("%d",&n)==1)
        {
            miny=mininf;
            maxy=maxinf;
            for(int i=0;i<n;i++)
            {

scanf("%I64d%I64d%I64d%I64d",&x1,&y1,&x2,&y2);
                vec.push_back(node(x1,y1,y2,true));
                vec.push_back(node(x2,y1,y2,false));
                miny=min(miny,y1);
                maxy=max(maxy,y2);
            }
            sort(vec.begin(),vec.end());
            //cout<<miny<<" "<<maxy<<endl;
            build(miny,maxy,1);
            __int64 peri=0;
            int m=vec.size();
            __int64 lastlen=0,lastseg=0;
            for(int i=0;i<m;i++)
            {
                if(vec[i].flag)
                    insert(vec[i].down,vec[i].up,1);
                else
                    del(vec[i].down,vec[i].up,1);
                peri+=abs(Ltree[1].len-lastlen);
                //cout<<"len:"<<Ltree[1].len<<endl;
                lastlen=Ltree[1].len;
                if(i)

peri+=2*(vec[i].x-vec[i-1].x)*lastseg;
                lastseg=Ltree[1].nseg;
```

```
            //cout<<"seg:"<<Ltree[1].nseg<<endl;
        }
        printf("%I64d\n",peri);
    }
    return 0;
}
```

矩形并得面积：

矩形的并面积 离散化：

```
#include <iostream>
#include <algorithm>
using namespace std;
#define MAXN 100001
struct {
    int c, l, r;
    long long cnt, lf, rf;
}nod[MAXN*6];
struct Line {
    long long x, y1, y2;
    int f;
}line[MAXN];
bool cmp(Line a, Line b) {
    return a.x < b.x;
}
long long y[MAXN];
void creat(int t, int l, int r) {
    nod[t].c = nod[t].cnt = 0;
    nod[t].l = l, nod[t].r = r;
    nod[t].lf = y[l], nod[t].rf = y[r];
    if(l + 1 == r) return;
    int m = (l+r) / 2;
    creat(t*2, l, m), creat(t*2+1, m, r);
}
void calen(int t) {
    if(nod[t].c > 0) {
        nod[t].cnt = nod[t].rf - nod[t].lf;
        return;
    }
    if(nod[t].l + 1 == nod[t].r) nod[t].cnt = 0;
    else nod[t].cnt = nod[t*2].cnt + nod[t*2+1].cnt;
}
void update(int t, Line e) {
    if(e.y1 == nod[t].lf && e.y2 == nod[t].rf) {
        nod[t].c += e.f;
        calen(t);
```

```
        return;
    }
    if(e.y2 <= nod[t*2].rf) update(t*2, e);
    else if(e.y1 >= nod[t*2+1].lf) update(t*2+1, e);
    else {
        Line tmp = e;
        tmp.y2 = nod[t*2].rf;
        update(t*2, tmp);
        tmp = e;
        tmp.y1 = nod[t*2+1].lf;
        update(t*2+1, tmp);
    }
    calen(t);
}
long long xx1[MAXN],xx2[MAXN],yy1[MAXN],yy2[MAXN];
int main() {
    int n, i, t, cases = 1;
    long long d,x1, x2, y1, y2, ans,a,b,c;
    scanf("%d", &n);
    t = 1;
    ans = 0;
    for(i = 1; i <= n; i++) {
        scanf("%lld %lld %lld %lld",&x1,&y1,&x2,&y2);
        line[t].x = x1, line[t+1].x = x2;
        line[t].y1 = line[t+1].y1 = y[t] = y1, line[t].y2 = line[t+1].y2
= y[t+1] = y2;
        line[t].f = 1, line[t+1].f = -1;
        t += 2;
    }
    sort(line+1, line+t, cmp), sort(y+1, y+t);
    creat(1, 1, t-1);
    update(1, line[1]);
    for(i = 2; i < t; i++) {
        ans += nod[1].cnt * (line[i].x - line[i-1].x);
        update(1, line[i]);
    }
    printf("%lld\n", ans);
    return 0;
}
```

从 1 开始。

随机第 k 小元素

```
int select(int *a,int b,int e,int k){
    if(b==e) return a[b];
```

```
    int x=a[b+rand()%(e-b+1)],i=b-1,j=e+1,tmp;
    while (i<j){
        while(a[++i]<x);
        while(a[--j]>x);
        if (i<j) tmp=a[i],a[i]=a[j],a[j]=tmp;
    }
    if (j==e) j--;
    i=j-b+1;
    if (k<=i) return select(a,b,j,k);
    else return select(a,j+1,e,k-i);
}
```

//KMP

```
int fail[maxlen];
void makefail( char *t, int lt ){
    --t;
    for(int i=1,j=0;i<=lt;i++,j++){
        fail[i]=j;
        while(j>0 && t[i]!=t[j]) j=fail[j];
    }
}
// start matching pattern T in S[i..)
// return match pos or longest match length with corresponding pos
int kmp(char *s, int ls, char *t, int lt, int i,int &longest,int &lp){
    longest = lp = 0; --s; --t;
    for(int j=1; i<=ls; i++,j++) {
        while( j>0 && s[i]!=t[j] ) j=fail[j];
        if( j>longest ) { longest = j; lp = i-j; }
        if( j==lt ) return i-lt;
    }
    return -1;
}
```

AC 自动机：

```
struct stu
{
    int flag,num,now;
    struct stu *next[26],*fail;
}root,arr[700000],*que[700000],*pp;
void bulit(char name[10],int y)
{
    int i,a;
```

```
    struct stu *p;
    p=&root;
    for(i=0;name[i];i++)
    {
        a=name[i]-'a';
        if(p->next[a]!=0)p=p->next[a];
        else
        {
            p->next[a]=&arr[wei];
            p=p->next [a];
            wei++;
        }
    }
    if(p->flag!=0)
    {
        dui[y]=p->num ;
        return ;
    }
    if(t==0)
    {
        p->flag=1;
        p->num =y;
    }
    else
    {
        p->flag=strlen(name);
        p->num =y;
    }
    p->now=-1;
}
void fail()
{
    struct stu *q;
    int head=0,tal=1,i;
    que[head]=&root;
    while(head!=tal)
    {
        if(que[head]==&root)
        {
            for(i=0;i<26;i++)
            {
                if(que[head]->next[i]==0)continue;

                que[head]->next[i]->fail=&root;
```

```
                que[tal]=que[head]->next[i];
                tal++;
            }
        }
        else
        {
            for(i=0;i<26;i++)
            {
                if(que[head]->next[i]==0)continue;
                que[tal]=que[head]->next [i];
                tal++;
                q=que[head];
                q=q->fail ;
                while(q)
                {
                    if(q->next[i]!=NULL)break;
                    else q=q->fail ;
                }
                if(q==NULL)que[head]->next [i]->fail=&root;
                else que[head]->next[i]->fail=q->next[i];
            }
        }
        if(head==tal)break;
        head++;
    }
}
```

图论：

//////////////////////

二部图匹配：

```
//邻接矩阵的
#include<iostream>
using namespace std;
bool map[501][501];
int mt[501];
bool s[501];
bool fine(int i);
int n;
int main(){
    int i,a,b,j,m;
    while(scanf("%d",&n)!=EOF){
        memset(map,0,sizeof(map));
        for(i=0;i<n;i++){
```

```
            scanf("%d: (%d)",&a,&b);
            for(j=0;j<b;j++){
                scanf(" %d",&m);
                map[a][m]=1;
            }
        }
        memset(mt,-1,sizeof(mt));
        int ans=0;
        for(i=0;i<n;i++){
            memset(s,0,sizeof(s));
            if(fine(i))ans++;
        }
        printf("%d\n",n-ans/2);
    }
    return 0;
}
bool fine(int i){
    int j;
    for(j=0;j<n;j++){
        if(map[i][j]&&!s[j]){
            s[j]=1;
            if(mt[j]==-1||fine(mt[j])){
                mt[j]=i;
                return 1;
            }
        }
    }
    return 0;
}
//////////////////////////////////////////
```

KM 算法（网络流算法）：

```
//邻接矩阵 KM 算法
#include <string.h>
#include<stdio.h>
#define MAXN 310
#define inf 1000000000
#define _clr(x) memset(x,0xff,sizeof(int)*n)

int kuhn_munkras(int m,int n,int mat[][MAXN],int* match1,int* match2){
    int s[MAXN],t[MAXN],l1[MAXN],l2[MAXN],p,q,ret=0,i,j,k;
    for (i=0;i<m;i++){
        for (l1[i]=-inf,j=0;j<n;j++){
            l1[i]=mat[i][j]>l1[i]?mat[i][j]:l1[i];
        }
```

```
    }
    for (i=0;i<n;l2[i++]=0);
    for (_clr(match1),_clr(match2),i=0;i<m;i++){
        for (_clr(t),s[p=q=0]=i;p<=q&&match1[i]<0;p++){
            for (k=s[p],j=0;j<n&&match1[i]<0;j++){
                if (l1[k]+l2[j]==mat[k][j]&&t[j]<0){
                    s[++q]=match2[j],t[j]=k;
                    if (s[q]<0){
                        for (p=j;p>=0;j=p){
                            match2[j]=k=t[j],p=match1[k],match1[k]=j;
                        }
                    }
                }
            }
        }
        if (match1[i]<0){
            for (i--,p=inf,k=0;k<=q;k++){
                for (j=0;j<n;j++){
                    if (t[j]<0&&l1[s[k]]+l2[j]-mat[s[k]][j]<p)
                        p=l1[s[k]]+l2[j]-mat[s[k]][j];
                }
            }
            for (j=0;j<n;l2[j]+=t[j]<0?0:p,j++);
            for (k=0;k<=q;l1[s[k++]]-=p);
        }
    }
    for (i=0;i<m;i++)
        ret+=mat[i][match1[i]];
    return ret;
}
int main(){
    int n,m,i,j;
    int map[MAXN][MAXN],mt[MAXN],nt[MAXN];
    scanf("%d %d",&n,&m);
    for(i=0;i<n;i++){
        for(j=0;j<m;j++){
            scanf("%d",&map[i][j]);
        }
    }
    printf("%d\n",kuhn_munkras(n,m,map,mt,nt));//mt 是 n 个点的匹配对象，
nt 是 m 个点的匹配对象。
    for(i=0;i<3;i++){
        printf("%d ",mt[i]);
    }
```

```
    printf("\n");
    for(i=0;i<3;i++){
        printf("%d  ",nt[i]);
    }
    printf("\n");
    return 0;
}
////////////////////////////////////
```
最优比例树
```
#include<iostream>
#include<cmath>
#include<algorithm>
#define N 0.000001
#define INF 100000000
using namespace std;
struct stu{
    int from,to,h;
    double bi,len;
    bool operator <(const stu &t)const{
        return bi<t.bi    ;
    }
}arr[1000001],map[1001];
int wei,pe[1001],n;
double dis(int i,int j){
    double sum;
    sum=(map[i].from  -map[j].from  )*(map[i].from  -map[j].from  )+(map[i].to
-map[j].to  )*(map[i].to  -map[j].to  );
    return sqrt(sum);
}
int fine(int x){
    int f,ff=x;
    while(x!=pe[x])x=pe[x];
    while(ff!=pe[ff]){
        f=pe[ff];
        pe[ff]=x;
        ff=f;
    }
    return x;
}
double krus(double  m);
double mm[1001][1001];
int main(){
    int i,j;
    while(scanf("%d",&n)!=EOF){
```

```
        if(n==0)break;
        for(i=0;i<n;i++){
            scanf("%d %d %d",&map[i].from ,&map[i].to ,&map[i].h );
        }
        wei=0;
        for(i=0;i<n;i++){
            for(j=i+1;j<n;j++){
                arr[wei].from =i;
                arr[wei].to =j;
                if(map[i].h >map[j].h ){
                    arr[wei].h =map[i].h -map[j].h ;
                }
                else{
                    arr[wei].h =map[j].h -map[i].h ;
                }
                arr[wei].len =dis(i,j);
                wei++;
            }
        }
        double p,q,m;
        double a;
        p=0;q=32;
        while(1){
            m=(p+q)/2;
            a=krus(m);
            if(a==0){
                p=m;
                break;
            }
            if(a>0) {
                p=m+N;
            }
            else{
                q=m-N;
            }
            if(p>q)break;
        }
        printf("%.3lf\n",m);
    }
    return 0;
}
double krus(double m) {
    int i,j;
    for(i=0;i<n;i++){pe[i]=i;}
```

```
    for(i=0;i<wei;i++){
        mm[arr[i].from ][arr[i].to ]=double(arr[i].h -m*arr[i].len);
        mm[arr[i].to ][arr[i].from ]=mm[arr[i].from ][arr[i].to ];
    }
    for(i=0;i<n;i++)mm[i][i]=INF;
    int mark;
    bool flag[1001]={0};
    double ans=0,ss[1001],mi;
    flag[0]=1;
    for(i=0;i<n;i++){
        ss[i]=mm[0][i];
    }
    for(i=0;i<n-1;i++){
        mi=INF;
        for(j=0;j<n;j++){
            if(!flag[j]&&mi>ss[j]){
                mi=ss[j];
                mark=j;
            }
        }
        flag[mark]=1;
        ans+=ss[mark];
        for(j=0;j<n;j++){
            if(!flag[j]&&mm[mark][j]<ss[j]){
                ss[j]=mm[mark][j];
            }
        }
    }
    return ans;
}
//////////////////////////////////////////
```

割边割点：

```
+#include<iostream>
#include<algorithm>
using namespace std;
struct stu{
    int num;
    struct stu *next;
}arr[200000],*head[1001];
int root,bi[1001],dep,de[1001],low[1001],bo[1010];
bool flag[1001],ssii,size[1010];
void DFS(int s,int a);
int main(){
    int a,b,n,wei,i,g=0;
```

```
while(scanf("%d",&a)&&a){
    g++;
    if(g>1)printf("\n");
    bool size[1010]={0};
    memset(flag,0,sizeof(flag));
    memset(head,0,sizeof(head));
    wei=0;
    scanf("%d",&b);
    n=max(a,b);
    arr[wei].num  =b;
    arr[wei].next  =head[a];
    head[a]=&arr[wei];
    wei++;
    arr[wei].num=a;
    arr[wei].next=head[b];
    head[b]=&arr[wei];
    wei++;
    size[a]=1;size[b]=1;
    while(scanf("%d",&a)&&a)
    {
        scanf("%d",&b);
        n=max(a,n);n=max(b,n);
        arr[wei].num  =b;
        arr[wei].next  =head[a];
        head[a]=&arr[wei];
        wei++;
        arr[wei].num=a;
        arr[wei].next=head[b];
        head[b]=&arr[wei];
        wei++;
        size[a]=1;
        size[b]=1;
    }
    memset(bi,0,sizeof(bi));
    memset(de,0,sizeof(de));
    memset(bo,0,sizeof(bo));
    dep=1;
    root=1;
    while(!size[root])root++;
    memset(flag,0,sizeof(flag));
//  for(i=1;i<=n;i++)//有连通分支时用
    {
        //if(de[i]==0&&size[i])
            DFS(-1,root);
```

```
            }
            bi[root]--;
            printf("Network  #%d\n",g);
            bool ss=0;
            for(i=0;i<=n;i++)
            {
                if(flag[i])
                {
                    ss=1;
                    printf("     SPF node  %d leaves %d subnets\n",i,bi[i]+1);
                }
            }
            if(ss==0)
            {
                printf("     No SPF nodes\n");
            }

        }
        return  0;
}
void DFS(int  s,int a)
{
        de[a]=dep;low[a]=dep;
        dep++;
        struct stu *p;
        p=head[a];
        while(p)
        {
            //bo[a]++;
            if(de[p->num  ]==0)
            {
                DFS(a,p->num  );
                bo[a]++;
                low[a]=min(low[a],low[p->num  ]);
                if(a==root&&bo[a]>1)
                {
                    flag[a]=1;
                    bi[a]=bo[a];
                }
                else
                {
                    if(root!=a&&de[a]<=low[p->num  ])
                    {
                        flag[a]=1;
```

```
                    bi[a]++;
                }
            }
        }
        else
        {
            if(p->num!=s)
            {
                low[a]=min(low[a],de[p->num ]);
            }
        }
        p=p->next;
    }
}
////////////////////////////////////////////////////////
```
最小树形图：
```
#include<iostream>
using namespace std;
#define inf 2100000000
int map[101][101],mi,n;
bool vi[101];
void DFS(int i);
void lc();
int main(){
 int m,a,b,i,j,len;
 while(scanf("%d %d",&n,&m)!=EOF){
    if(n==0&&m==0)break;
    for(i=1;i<=n;i++)
    {
     for(j=1;j<=n;j++)
     {
       map[i][j]=inf;
     }
    }
    for(i=0;i<m;i++)
    {
     scanf("%d%d%d",&a,&b,&len);
     if(map[a][b]>len)
     {
       map[a][b]=len;
     }
    }
    memset(vi,0,sizeof(vi));
    DFS(1);
```

```
    for(i=1;i<=n;i++)
    {
      if(vi[i]==0)break;
    }
    if(i!=n+1)
    {
      printf("impossible\n");
      continue;
    }
    mi=0;
    lc();
    printf("%d\n",mi);
  }
  return 0;
}
void DFS(int i)
{
  vi[i]=1;
  for(int j=1;j<=n;j++)
  {
    if(vi[j]||map[i][j]==inf)continue;
    DFS(j);
  }
}
void lc()
{
  int pre[101],i,j,k;
  bool del[101]={0};
  while(1)
  {
    for(i=2;i<=n;i++)//缩点后更新 pre
    {
      if(del[i])continue;
      map[i][i]=inf;
      pre[i]=i;
      for(j=1;j<=n;j++)
      {
        if(del[j])continue;
        if(map[j][i]<map[pre[i]][i])
        {
          pre[i]=j;
        }
      }
    }
```

```
for(i=2;i<=n;i++)//找是否有环，
{
 if(del[i])continue;
 memset(vi,0,sizeof(vi));
 int biao=i;
 while(!vi[biao]&&biao!=1)
 {
  vi[biao]=1;
  biao=pre[biao];
 }
 if(biao==1)continue;
 i=biao;//找到有环
 mi+=map[pre[i]][i];//改变 mi 值
 for(j=pre[i];j!=i;j=pre[j])
 {
  mi+=map[pre[j]][j];
  del[j]=1;
 }
 for(j=1;j<=n;j++)//更新 map 值
 {
  if(del[j])continue;
  if(map[j][i]!=inf)
  {
   map[j][i]=map[j][i]-map[pre[i]][i];
  }
 }
 for(j=pre[i];j!=i;j=pre[j])
 {
  for(k=1;k<=n;k++)
  {
   if(del[k])continue;
   map[i][k]=min(map[i][k],map[j][k]);
   if(map[k][j]!=inf)
   {
    map[k][i]=min(map[k][i],map[k][j]-map[pre[j]][j]);
   }
  }
 }
 break;//只找一个
}
if(i>n)
{
 for(i=2;i<=n;i++)
 {
```

```
        if(del[i])continue;
        mi+=map[pre[i]][i];
      }
      break;
    }
  }
}
//////////////////////////////////////////
```

2-sat 问题：

```
#include<iostream>
using namespace std;
struct stu
{
    int num;
    struct stu *next;
}arr[4000004],*zhead[2002],*fhead[2002];
int num[2002],w,biao[2002],ans;
bool flag[2002];
void DFS1(int i);
void DFS2(int i);
int main()
{
    int n,m,a,b,c,d,wei,i;
    while(scanf("%d",&n)!=EOF)
    {
        memset(zhead,0,sizeof(zhead));
        memset(fhead,0,sizeof(fhead));
        memset(num,0,sizeof(num));
        memset(biao,0,sizeof(biao));
        n=2*n;
        wei=0;
        scanf("%d",&m);
        for(i=0;i<m;i++)
        {
            scanf("%d %d %d %d",&a,&b,&c,&d);
            a=2*a+c;
            b=2*b+d;
            if(a%2==0){c=a+1;}
            else c=a-1;
            if(b%2==0){d=b+1;}
            else d=b-1;
            arr[wei].num  =d;
            arr[wei].next  =zhead[a];
            zhead[a]=&arr[wei];
```

```
            wei++;
            arr[wei].num=c;
            arr[wei].next  =zhead[b];
            zhead[b]=&arr[wei];
            wei++;
            arr[wei].num  =a;
            arr[wei].next  =fhead[d];
            fhead[d]=&arr[wei];
            wei++;
            arr[wei].num=b;
            arr[wei].next=fhead[c];
            fhead[c]=&arr[wei];
            wei++;
        }
        w=0;
        memset(flag,0,sizeof(flag));
        for(i=0;i<n;i++)
        {
            DFS1(i);
        }
        memset(flag,0,sizeof(flag));
        ans=0;
        for(i=n-1;i>=0;i--)
        {
            if(flag[num[i]])continue;
            ans++;
            DFS2(num[i]);
        }
        bool e=0;
        for(i=0;i<n;i+=2)
        {
            if(biao[i]==biao[i+1])
            {
                printf("NO\n");
                e=1;
                break;
            }
        }
        if(e==0)
        {
            printf("YES\n");
        }
}
return 0;
```

```
}
void DFS1(int i)
{
	if(flag[i])return;
	flag[i]=1;
	struct stu *p;
	p=zhead[i];
	while(p)
	{
		DFS1(p->num );
		p=p->next;
	}
	num[w]=i;w++;
}
void DFS2(int i)
{
	if(flag[i])return;
	flag[i]=1;
	struct stu *p;
	p=fhead[i];
	while(p)
	{
		DFS2(p->num);
		p=p->next;
	}
	biao[i]=ans;
}
//////////////////////////////////////////
```

LCA(最近公共祖先 O（n）):

```
#include<iostream>
using namespace std;
struct stu
{
    int num,dep;
    struct stu *next;
}arr[100010],*head[50010];
struct List_node
{
	int index;
	int node;
	int next;
}list_node[100010];
struct ss
{
```

```
    int num,li;
    bool flag;
}brr[1000];
int ans[50010],pe[50010],header[50010];;
bool visit[50010];
void DFS(int a,int b);
int fine(int x)
{
    int f=x,ff=x;
    while(x!=pe[x])
    {
        x=pe[x];
    }
    while(f!=pe[f])
    {
        ff=pe[f];
        pe[f]=x;
        f=ff;
    }
    return x;
}
int main()
{
    int n,a,b,m,i,j;
    while(scanf("%d",&n)!=EOF)
    {
        memset(brr,0,sizeof(brr[0])*(n+5));
        for(i=1;i<=n;i++)
        {
            head[i]=0;
            pe[i]=-1;
            visit[i]=0;
            header[i]=-1;
            scanf("%d",&a);
            for(j=0;j<a;j++)
            {
                scanf("%d",&b);
                arr[b].num  =b;
                arr[b].dep=0;
                arr[b].next  =head[i];
                head[i]=&arr[b];
            }
        }
        scanf("%d",&m);
```

```
        int top = 0;
         for (i = 1; i <= m; ++i)
         {
              scanf("%d %d", &a, &b);

              list_node[top].index  = i;
              list_node[top].node  = b;
              list_node[top].next  = header[a];
              header[a] = top;
              ++top;

              list_node[top].index  = i;
              list_node[top].node  = a;
              list_node[top].next  = header[b];
              header[b] = top;
              ++top;
         }

        /*for(i=0;i<m;i++)
        {
             scanf("%d %d",&a,&b);
             brr[a].li=i;
             brr[b].li=i;
             brr[a].flag  =1;
             brr[b].flag  =1;
             brr[a].num  =b;
             brr[b].num  =a;
        }*/
        DFS(1,1);
        for(i=1;i<=m;i++)
        {
             printf("%d\n",ans[i]);
        }
    }
    return 0;
}
void DFS(int root,int dep)
{
    int i,j,x,y;
    pe[root]=root;
    arr[root].dep=dep;
    struct stu *q=head[root];
    while(q)
    {
```

```
            DFS(q->num,dep+1);
            x=fine(root);
            y=fine(q->num);
            pe[y]=x;
            q=q->next ;
        }
        visit[root]=1;
/*      if(brr[root].flag  ==0)return;
        if(visit[brr[root].num]==1)
        {
            ans[brr[root].li]=arr[fine(brr[root].num)].dep ;
        }*/
        int list = header[root];
         while  (list != -1)
         {
              if (visit[list_node[list].node]  == 1)
                   ans[list_node[list].index]  = arr[fine(list_node[list].node)].dep;
              list = list_node[list].next;
         }

}
//////////////////////////////////////
```

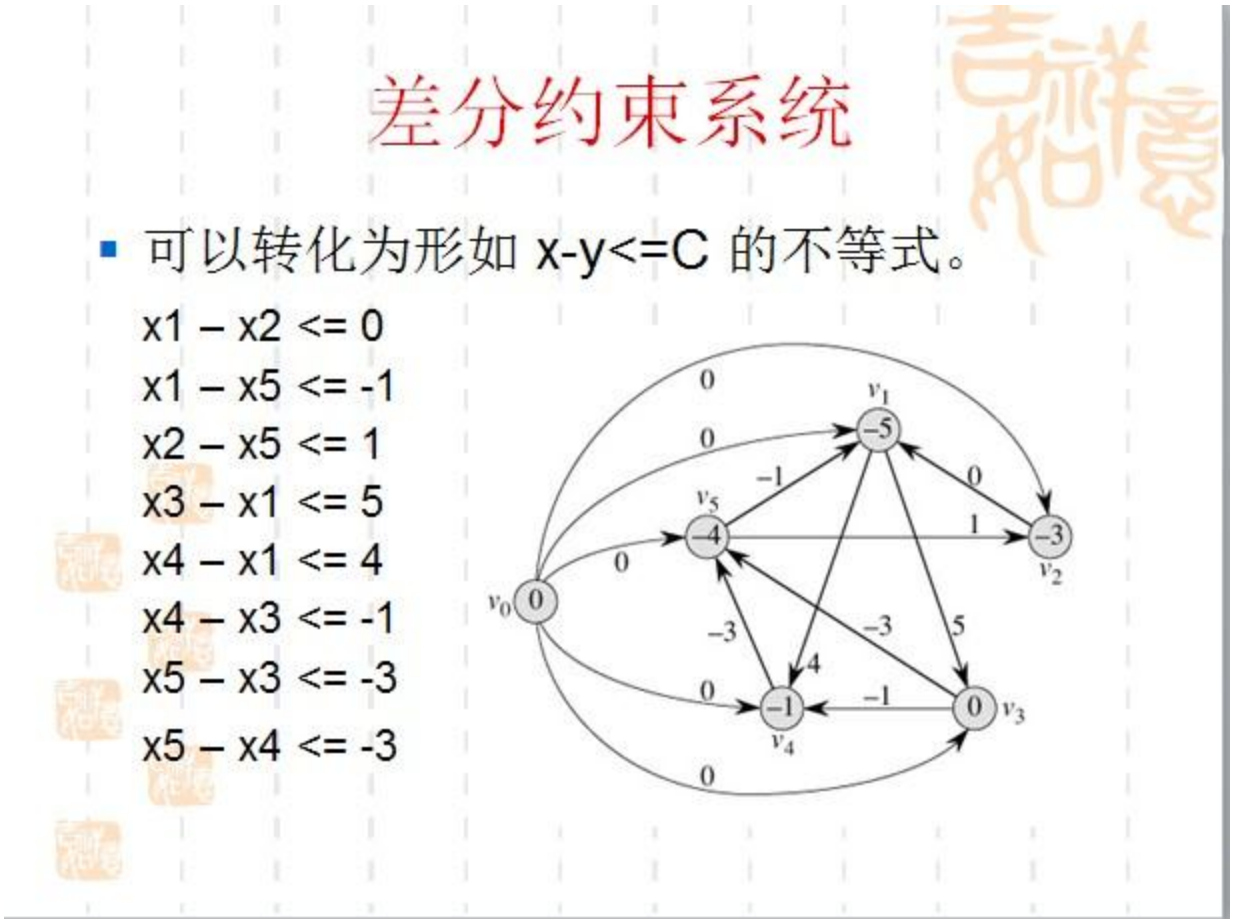
第K短路径：

```
#include <cstdio>
#include <vector>
#include <queue>
#define INF 1<<30
#define MAXN 1010
using namespace std;
struct Node{
    int w,id;
    bool operator<(const Node &cp)const{
        return w>cp.w;
    }
    Node & operator=(const Node &cp){
        id=cp.id;
        w=cp.w;
        return *this;
    }
};
vector<Node> g[MAXN],ng[MAXN];
int h[MAXN],c[MAXN];
priority_queue<Node> pq;
int n,m,s,t,times;
```

```
void djk(){
    int i,j,k;
    for(i=0;i<=n;i++)
        h[i]=INF;
    Node tmp,p;
    tmp.id=t;
    tmp.w=0;
    pq.push(tmp);
    while(!pq.empty()){
        p=pq.top();
        pq.pop();
        if (h[p.id]>p.w)
        {
            h[p.id]=p.w;
            k=ng[p.id].size();
            for (i=0;i<k;i++)
            {
                tmp.id = ng[p.id][i].id;
                tmp.w = p.w+ng[p.id][i].w;
                pq.push(tmp);
            }
        }
    }
}
int A_star(){
    Node tmp,p;
    int i,j,k;
    tmp.id=s;
    tmp.w=h[s];
    pq.push(tmp);
    while(!pq.empty()){
        p=pq.top();
        pq.pop();
        c[p.id]++;
        if(c[t]==times)return p.w;
        if(c[t]>times)continue;
        k=g[p.id].size();
        for (i=0;i<k;i++){
            tmp.id=g[p.id][i].id;
            tmp.w =p.w-h[p.id]+g[p.id][i].w+h[g[p.id][i].id];
            pq.push(tmp);
        }
    }
    return -1;
```

```
}
int main(){
    int i,j,k;
    Node tmp;
    scanf("%d%d",&n,&m);
    while(m--){
        scanf("%d%d%d",&j,&k,&tmp.w);
        tmp.id=k;
        g[j].push_back(tmp);
        tmp.id=j;
        ng[k].push_back(tmp);
    }
    scanf("%d%d%d",&s,&t,&times);
    if(s==t)times++;
    djk();
    printf("%d\n",A_star());
    return 0;
}
```

差分约束：



计算几何：

海伦公式
三角形面积S=sqrt(p*(p-a)*(p-b)*(p-c))
其中p=(a+b+c)/2
•三角形内切圆半径r = 2S/(a+b+c)

•三角形外接圆半径r = a*b*c/(4S)

【点P到直线AB距离】

$$\frac{|(P-A)\times(B-A)|}{|B-A|}$$

【多边形面积公式】$Area=\frac{1}{2}\sum\begin{vmatrix} x_i & x_{i+1} \\ y_i & y_{i+1} \end{vmatrix}$

【多边形重心公式】多边形重心(C_x,C_y)

$$C_x=\frac{1}{6Area}\sum(x_i+x_{i+1})\begin{vmatrix} x_i & x_{i+1} \\ y_i & y_{i+1} \end{vmatrix} \quad C_y=\frac{1}{6Area}\sum(y_i+y_{i+1})\begin{vmatrix} x_i & x_{i+1} \\ y_i & y_{i+1} \end{vmatrix}$$

【Graham-Scan算法】
[输入]输入点集p[n], 点数n
[输出]返回凸包点数, hull中为凸包逆时针顺序
[注意]特殊情况：*n* = 1; 2; *top* = 1; 2

```
struct TPoint { double x, y; };
double cross(const TPoint & a, const TPoint & b)
{ return (a.x * b.y - a.y * b.x); }
double cross(const TPoint & a, const TPoint & b, const TPoint & c)
{ return (b.x - a.x) * (c.y - a.y) - (b.y - a.y) * (c.x - a.x); }

TPoint p[N];
bool graham_cmp(const TPoint &b, const TPoint &c)
{
double tmp = cross(b, c, p[0]);
if(tmp > EPS) return true;
if(fabs(tmp) < EPS && (dis(b, p[0]) < dis(c, p[0]))) return true;
return false;
}
int graham_scan(TPoint hull[], int n)
{
int top, i, k = 0;
for(i = 1;i < n;++ i)
if((p[i].y < p[k].y) || (p[i].y == p[k].y && p[i].x < p[k].x))
k = i;
swap(p[0], p[k]);
sort(p+1, p + n, graham_cmp);
hull[0] = p[0], hull[1] = p[1], hull[2] = p[2];
```

```
if(n < 3) return  n; else top = 3;
for(i = 3;i < n;++ i) {
while(top  >= 2 && cross(hull[top-2],  hull[top-1],  p[i]) < EPS) -- top;
hull[top++]  = p[i];
}
return  top;}
```

【Pick定理(网格中)】

$Area = \frac{1}{2}EdgeDot + InnerDot - 1$

[输入]多边形顶点集p(按逆时针或逆时针存放),多边形顶点数n, 面积A，边界点E，多边形内点I

[输出]多边形面积A, 边界点数E, 内点数I

```
void Pick(TPoint p[], int n, double& A, int& E, int &I)
{
    A = area(p, n), E = 0, p[n] = p[0];
    for(int i = 0;i < n;++ i) {
        int dx = p[i].x - p[i+1].x, dy = p[i].y - p[i+1].y;
        if(dx == 0 || dy == 0) E += abs(dx + dy);
        else E += gcd(abs(dx), abs(dy));
    }
    I = (int)(A + 1 - E / 2.0);
}
```

数论：

欧拉函数：

```
typedef long long llong;
llong Eular(llong  n)
{
 int i;
 llong ret=n;
 for(i=2;i*i<=n;++i)
 {
     if(n%i==0)
     {
         ret-=ret/i;
         while(n%i==0)n/=i;
         if(n==1)break;
     }
 }
 if(n!=1)ret-=ret/n;
 return ret;
}
```

求区间内的欧拉函数：

```
//求区间内Eular,1..N
```

```
//顺便还产生了1..N内素数-_-if(phi[i]==i-1) so i is a prime...
#define N 1500001
int phi[N];
void mkphilist()
{
  int i,j;
  phi[1]=1;
  for(i=2;i<N;++i)
   if(!phi[i])
     for(j=i;j<N;j+=i)
     {
          if(!phi[j])
               phi[j]=j;
          phi[j]-=phi[j]/i;
     }
}
```

```
ll ext_gcd(ll a, ll b, ll &x, ll &y)
{
Ll t, d;
if (b == 0) { x = 1; y = 0; return a; }
d = ext_gcd(b, a % b, x, y);
t = x;
x = y;
y = t - a / b*y;
return d;
}
```

5.3 中国剩余定理

$$\begin{cases} a \equiv b_1 \pmod{w_1} \\ a \equiv b_2 \pmod{w_2} \\ \vdots \\ a \equiv b_n \pmod{w_n} \end{cases}$$

其中w, b已知,$w[i] > 0$且$w[i]$与$w[j]$互质,求a.解的范围$1..n, n = w[0] * w[1] * ... * w[k -$

```
int China(int b[], int w[], int k) {
    int i;
    int d, x, y, a = 0, m, n = 1;
    for (i = 0; i < k; i++) n *= w[i];
    for (i = 0; i < k; i++) {
        m = n / w[i];
        d = ext_gcd(w[i], m, x, y);
        a = (a + y * m * b[i]) % n;
    }
    if (a > 0) return a;
    else return (a + n);
}
```

4. $\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$
5. $\sum_{k=1}^{n} 2k-1 = n^2$
6. $\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$
7. $\sum_{k=1}^{n} (2k-1)^2 = \frac{n(4n^2-1)}{3}$
8. $\sum_{k=1}^{n} k^3 = \left(\frac{n(n+1)}{2}\right)^2$
9. $\sum_{k=1}^{n} (2k-1)^3 = n^2(2n^2-1)$
10. $\sum_{k=1}^{n} k^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$
11. $\sum_{k=1}^{n} k^5 = \frac{n^2(n+1)^2(2n^2+2n-1)}{12}$
12. $\sum_{k=1}^{n} k(k+1) = \frac{n(n+1)(n+2)}{3}$
13. $\sum_{k=1}^{n} k(k+1)(k+2) = \frac{n(n+1)(n+2)(n+3)}{4}$
14. $\sum_{k=1}^{n} k(k+1)(k+2)(k+3) = \frac{n(n+1)(n+2)(n+3)(n+4)}{5}$

头文件：

```
#include <vector>
#include <list>
#include <map>
#include <set>
#include <deque>
#include <queue>
#include <stack>
#include <bitset>
#include <algorithm>
#include <functional>
#include <numeric>
#include <utility>
#include <sstream>
#include <iostream>
#include <iomanip>
#include <cstdio>
```

```
#include <cmath>
#include <cstdlib>
#include <cctype>
#include <string>
#include <cstring>
#include <cstdio>
#include <cmath>
#include <cstdlib>
#include <ctime>
#include <string.h>

using namespace std;

#define ALL(X)          (X).begin(),(X).end()
#define SIZE(X)         ((int)(X).size())
#define FORI(x,a,b)     for(x=(a);x<=(b);x++)
#define FORD(x,a,b)     for(x=(a);x>=(b);x--)
#define min(a,b)        ((a)<(b)?(a):(b))
#define max(a,b)        ((a)<(b)?(b):(a))
#define MEM(X,with)     memset((X),(with),sizeof(X))
#define Contains(X,item)      ((X).find(item)!=(X).end())
#define Contains_n(X,item)
    (find((X).begin(),(X).end(),(item))!=(X).end())
#define rep(i,m,n) for((i)=m;(i)<(int)(n);(i)++)
#define PQ priority_queue
#define IT iterator
#define N 100005

//bool prime[N];
const int inf =(1<<30)-1;
const long long linf = (111<<62)-1;
const int dirx[]={-1,0,0,1,-1,-1,1,1}, diry[]={0,-1,1,0,-1,1,-1,1};
const double ERR = 1e-11, PI=(2*acos(0.0));


template<class T> bool cmpmin(T a, T b){return a<b;}
template<class T> bool cmpmax(T a, T b){return a>b;}
template<class T> inline T ABS(T a) {return ((a<0)?(-a):(a));}
template<class T> T GCD(T a,T b) {return ((!b)?(a):GCD(b,a%b));}
template<class T> T fastPow(T Base,T Power) {T Result=1;
while(Power>0){if(Power&((T)1))Result*=Base; Power>>=1; Base*=Base;}
return Result;}
template<class T> T fastModPow(T Base,T Power,T Mod) {T
Result=1;while(Power>0){if(Power&((T)1))Result=(Result*Base)%Mod;
```

```
Power>>=1; Base=(Base*Base)%Mod;} return (Result%Mod);}
inline int compDouble(double x,double y) {double d=x-y; if(d-ERR>0.0)
return 1; else if(d+ERR<0.0) return -1; else return 0;}
inline void
file(){freopen("rail.txt","r",stdin);/*freopen("rle-size.out","w",std
out);*/}
//void prim(){prime[0]=1;prime[1]=1;int i;long long
j;for(i=2;i<N;i++){if(prime[i]==0){for(j=i,j=j*i;j<N;j+=i){prime[j]=1
;}}}}

typedef long long ll;
typedef unsigned long long ull;
typedef stringstream   SS;
typedef vector<string> VS;
typedef vector<double> VD;
typedef vector<ll>     VL;
typedef vector<int>    VI;
```