Contents

Chapter 1

data structure

1.1 binary indexed tree.cpp

```
int tree[MAXX];

inline int lowbit(const int &a)
{
    return a&-a;
}

inline void update(int pos,const int &val)
{
    while(pos<MAXX)
    {
        tree[pos]+=val;
        pos+=lowbit(pos);
    }
}

inline int read(int pos)
{
    int re(0);
    while(pos>0)
    {
        re+=tree[pos];
        pos-=lowbit(pos);
    }
    return re;
}

int find_Kth(int k)
```

```
{
    int now=0;
    for (char i=20;i>=0;--i)
    {
        now|=(1<<i);
        if (now>MAXX || tree[now]>=k)
            now^=(1<<i);
        else k-=tree[now];
    }
    return now+1;
}
```

1.2 COT.cpp

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

#define MAXX 100111
#define MAX (MAXX*23)
#define N 18

int sz[MAX],lson[MAX],rson[MAX],cnt;
int head[MAXX];
int pre[MAXX][N];
int map[MAXX],m;

int edge[MAXX],nxt[MAXX<<1],to[MAXX<<1];
int n,i,j,k,q,l,r,mid;
int num[MAXX],dg[MAXX];

int make(int l,int r)
{
    if(l==r)
        return ++cnt;
    int id(++cnt),mid((l+r)>>1);
    lson[id]=make(l,mid);
    rson[id]=make(mid+1,r);
    return id;
}

inline int update(int id,int pos)
{
    int re(++cnt);
    l=1;
    r=m;
    int nid(re);
    sz[nid]=sz[id]+1;
    while(l<r)
    {
        mid=(l+r)>>1;
        if(pos<=mid)
        {
            lson[nid]=++cnt;
            rson[nid]=rson[id];
            nid=lson[nid];
            id=lson[id];
            r=mid;
        }
        else
        {
            lson[nid]=lson[id];
            rson[nid]=++cnt;
            nid=rson[nid];
            id=rson[id];
            l=mid+1;
        }
        sz[nid]=sz[id]+1;
    }
    return re;
}

void rr(int now,int fa)
{
    dg[now]=dg[fa]+1;
    head[now]=update(head[fa],num[now]);
    for(int i(edge[now]);i;i=nxt[i])
        if(to[i]!=fa)
        {
            j=1;
            for(pre[to[i]][0]=now;j<N;++j)
                pre[to[i]][j]=pre[pre[to[i]][j-1]][j-1];
            rr(to[i],now);
        }
}

inline int query(int a,int b,int n,int k)
{
    static int tmp,t;
    l=1;
    r=m;
    a=head[a];
    b=head[b];
    t=num[n];
    n=head[n];
    while(l<r)
    {
        mid=(l+r)>>1;
        tmp=sz[lson[a]]+sz[lson[b]]-2*sz[lson[n]]+(l<=t && t<=mid);
        if(tmp>=k)
        {
            a=lson[a];
            b=lson[b];
            n=lson[n];
            r=mid;
        }
        else
        {
            k-=tmp;
            a=rson[a];
            b=rson[b];
            n=rson[n];
            l=mid+1;
        }
    }
    return l;
}

inline int lca(int a,int b)
{
    static int i,j;
    j=0;
    if(dg[a]<dg[b])
        std::swap(a,b);
    for(i=dg[a]-dg[b];i;i>>=1,++j)
        if(i&1)
            a=pre[a][j];
    if(a==b)
        return a;
    for(i=N-1;i>=0;--i)
        if(pre[a][i]!=pre[b][i])
        {
            a=pre[a][i];
            b=pre[b][i];
        }
    return pre[a][0];
}
```

```
int main()
{
    scanf("%d %d",&n,&q);
    for(i=1;i<=n;++i)
    {
        scanf("%d",num+i);
        map[i]=num[i];
    }
    std::sort(map+1,map+n+1);
    m=std::unique(map+1,map+n+1)-map-1;
    for(i=1;i<=n;++i)
        num[i]=std::lower_bound(map+1,map+m+1,num[i])-map;
    for(i=1;i<n;++i)
    {
        scanf("%d %d",&j,&k);
        nxt[++cnt]=edge[j];
        edge[j]=cnt;
        to[cnt]=k;

        nxt[++cnt]=edge[k];
        edge[k]=cnt;
        to[cnt]=j;
    }
    cnt=0;
    head[0]=make(1,m);
    rr(1,0);
    while(q--)
    {
        scanf("%d %d %d",&i,&j,&k);
        printf("%d\n",map[query(i,j,lca(i,j),k)]);
    }
    return 0;
}
```

1.3 divide tree.cpp

```
//                    val[0][1],val[0][n]

template<class Tp>class DT
{
    public:
        int n;
        Tp val[20][MAXX],sorted[MAXX];
        inline void make()
        {
            std::sort(sorted+1,sorted+1+n);
            make(1,1,n,0);
        }
        inline int query(const int &l,const int &r,const int &k)
        {
            return query(1,1,n,l,r,k,0);
        }
    private:
        int toleft[20][MAXX],mid[MAXX<<2];
        // toleft:
        void make(const int &id,const int &l,const int &r,const int &d)
        {
            if(l!=r)
            {
                mid[id]=(l+r)>>1;
                int lsame(mid[id]-l+1),i;
                for(i=l;i<=r;++i)
                    if(val[d][i]<sorted[mid[id]])
                        --lsame;
                int lpos(l),rpos(mid[id]+1),same(0);
                for(i=l;i<=r;++i)
                {
                    if(i==l)
                        toleft[d][i]=0;
                    else
                        toleft[d][i]=toleft[d][i-1];
                    if(val[d][i]<sorted[mid[id]])
                    {
                        ++toleft[d][i];
                        val[d+1][lpos++]=val[d][i];
                    }
                    else
                        if(val[d][i]>sorted[mid[id]])
                            val[d+1][rpos++]=val[d][i];
                        else
                            if(same<lsame)
                            {
                                ++same;
                                ++toleft[d][i];
                                val[d+1][lpos++]=val[d][i];
                            }
                            else
                                val[d+1][rpos++]=val[d][i
```

```
];
            }
            make(id<<1,l,mid[id],d+1);
            make(id<<1|1,mid[id]+1,r,d+1);
        }
    }
    int query(const int &id,const int &ll,const int &rr,const int &l,const int &r,const int &k,const int &d)
    {
        if(l==r)
            return val[d][l];
        int s,ss;
        if(l==ll)
        {
            s=toleft[d][r];
            ss=0;
        }
        else
        {
            s=toleft[d][r]-toleft[d][l-1];
            ss=toleft[d][l-1];
        }
        if(s>=k)
        {
            int newl(ll+ss),newr(ll+ss+s-1);
            return query(id<<1,ll,mid[id],newl,newr,k,d+1);
        }
        int bb(l-ll-ss),b(r-l+1-s),newl(mid[id]+bb+1),newr(mid[id]+bb+b);
        return query(id<<1|1,mid[id]+1,rr,newl,newr,k-s,d+1);
    }
};
```

1.4 GSS7.cxx

```
#include<cstdio>
#include<algorithm>
#include<queue>

#define MAXX 100111
#define MAX (MAXX<<1)

struct node
```

```
{
    bool set,rev;
    node *pre,*nxt[2],*fa;
    int lmax,max,rmax,sum,val,sz;
    node();
    node(int a);
}*tree[MAXX],*nil,*a,*b;

node::node()
{
    rev=set=false;
    fa=pre=nil;
    nxt[0]=nxt[1]=nil;
    sz=lmax=max=rmax=sum=val=0;
}

node::node(int a)
{
    set=rev=false;
    sum=val=a;
    sz=1;
    lmax=max=rmax=std::max(0,a);
    fa=pre=nxt[0]=nxt[1]=nil;
}

inline void add(node &x,const node &l,const node &r)
{
    x.max=std::max(l.rmax+r.lmax,std::max(l.max,r.max));
    x.lmax=std::max(l.lmax,l.sum+r.lmax);
    x.rmax=std::max(r.rmax,r.sum+l.rmax);
    x.sum=l.sum+r.sum;
}

inline void up(node *id)
{
    id->sz=id->nxt[0]->sz+id->nxt[1]->sz+1;
    id->sum=id->val+id->nxt[0]->sum+id->nxt[1]->sum;
    id->lmax=std::max(id->nxt[0]->lmax,id->nxt[0]->sum+id->val+id->nxt[1]->lmax);
    id->rmax=std::max(id->nxt[1]->rmax,id->nxt[1]->sum+id->val+id->nxt[0]->rmax);
    id->max=std::max(id->nxt[0]->rmax+id->val+id->nxt[1]->lmax,std::max(id->nxt[0]->max,id->nxt[1]->max));
}

```

```
inline void set(node *id,int val)
{
    if(id==nil)
        return;
    id->set=true;
    id->val=val;
    id->sum=val*id->sz;
    id->max=id->lmax=id->rmax=std::max(0,id->sum);
}

inline void down(node *id)
{
    if(id==nil)
        return;
    if(id->rev)
    {
        id->rev=false;
        for(int i(0);i<2;++i)
            if(id->nxt[i]!=nil)
            {
                id->nxt[i]->rev^=true;
                std::swap(id->nxt[i]->nxt[0],id->nxt[i]->nxt[1]);
                std::swap(id->nxt[i]->lmax,id->nxt[i]->rmax);
            }
    }
    if(id->set)
    {
        for(int i(0);i<2;++i)
            if(id->nxt[i]!=nil)
                set(id->nxt[i],id->val);
        id->set=false;
    }
}

inline void rot(node *id,int tp)
{
    node *k(id->pre);
    k->nxt[tp^1]=id->nxt[tp];
    if(id->nxt[tp]!=nil)
        id->nxt[tp]->pre=k;
    if(k->pre!=nil)
        k->pre->nxt[k==k->pre->nxt[1]]=id;
    id->pre=k->pre;
    id->nxt[tp]=k;
    k->pre=id;
    up(k);
    up(id);
}

node *fresh(node* id)
{
    node *re(id);
    if(id->pre!=nil)
        re=fresh(id->pre);
    down(id);
    return re;
}

inline void splay(node *id)
{
    node *rt(fresh(id));
    if(id!=rt)
        for(std::swap(rt->fa,id->fa);id->pre!=nil;rot(id,id==id->pre->nxt[0]));
}

inline void access(node *id)
{
    for(node *to(nil);id!=nil;id=id->fa)
    {
        splay(id);
        id->nxt[1]->pre=nil;
        if(id->nxt[1]!=nil)
            id->nxt[1]->fa=id;
        id->nxt[1]=to;
        if(to!=nil)
            to->pre=id;
        to->fa=nil;
        up(to=id);
    }
}

inline void lca(node *&to,node *&id)
{
    access(to);
    splay(id);
    for(to=nil;id->fa!=nil;splay(id=id->fa))
    {
        id->nxt[1]->pre=nil;
        if(id->nxt[1]!=nil)
```

```
                id->nxt[1]->fa=id;
            id->nxt[1]=to;
            if(to!=nil)
                to->pre=id;
            to->fa=nil;
            up(to=id);
        }
}

int n,i,j,k;
int nxt[MAX],to[MAX],edge[MAXX],cnt;
std::queue<int>q;

inline void add(int a,int b)
{
    nxt[++cnt]=edge[a];
    edge[a]=cnt;
    to[cnt]=b;
}

void rr(int now,int fa)
{
    for(int i(edge[now]);i;i=nxt[i])
        if(to[i]!=fa)
        {
            tree[to[i]]->fa=tree[now];
            rr(to[i],now);
        }
}

/*
void print(node *id)
{
    if(id!=nil)
    {
        print(id->nxt[0]);
        printf("%2d %2d %2d %2d %2d %2d %c %2d\n",id->val,id->sum,id->sz,id->lmax,id->max,id->rmax,id->rev?'r':'n',id->pre->val);
        print(id->nxt[1]);
    }
}
*/

int main()
{
    nil=new node();
    scanf("%d",&n);
    for(i=1;i<=n;++i)
    {
        scanf("%d",&j);
        tree[i]=new node(j);
    }
    for(i=1;i<n;++i)
    {
        scanf("%d %d",&j,&k);
        add(j,k);
        add(k,j);
    }
    tree[0]=nil;
    rr(1,0);
    scanf("%d",&n);
    while(n--)
    {
        scanf("%d %d %d",&k,&i,&j);
        a=tree[i];
        b=tree[j];
        access(a);
        splay(a);
        a->rev^=true;
        std::swap(a->nxt[0],a->nxt[1]);
        std::swap(a->lmax,a->rmax);
        access(b);
        splay(b);
        /*
        print(b);
        puts("");
        printf("%d %d %d %d\n",b->sum,b->nxt[0]->sum,b->val,b->nxt[1]->sum);
        */
        if(k==1)
            printf("%d\n",b->max);
        else
        {
            scanf("%d",&k);
            set(b,k);
        }
    }
    return 0;
}
```

1.5 OTOCI.cpp

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

#define MAXX 30111

int nxt[MAXX][2],fa[MAXX],pre[MAXX],val[MAXX],sum[MAXX];
bool rev[MAXX];

inline void up(int id)
{
    static int i;
    sum[id]=val[id];
    for(i=0;i<2;++i)
        if(nxt[id][i])
            sum[id]+=sum[nxt[id][i]];
}

inline void rot(int id,int tp)
{
    static int k;
    k=pre[id];
    nxt[k][tp^1]=nxt[id][tp];
    if(nxt[id][tp])
        pre[nxt[id][tp]]=k;
    if(pre[k])
        nxt[pre[k]][k==nxt[pre[k]][1]]=id;
    pre[id]=pre[k];
    nxt[id][tp]=k;
    pre[k]=id;
    up(k);
    up(id);
}

inline void down(int id)
{
    static int i;
    if(rev[id])
    {
        rev[id]=false;
        std::swap(nxt[id][0],nxt[id][1]);
        for(i=0;i<2;++i)
            if(nxt[id][i])
                rev[nxt[id][i]]^=true;
    }
}

int freshen(int id)
{
    int re(id);
    if(pre[id])
        re=freshen(pre[id]);
    down(id);
    return re;
}

inline void splay(int id)
{
    static int rt;
    if(id!=(rt=freshen(id)))
        for(std::swap(fa[id],fa[rt]);pre[id];rot(id,id==
            nxt[pre[id]][0]));
}

inline void access(int id)
{
    static int to;
    for(to=0;id;id=fa[id])
    {
        splay(id);
        if(nxt[id][1])
        {
            pre[nxt[id][1]]=0;
            fa[nxt[id][1]]=id;
        }
        nxt[id][1]=to;
        if(to)
        {
            pre[to]=id;
            fa[to]=0;
        }
        up(to=id);
    }
}

inline int getrt(int id)
{
    access(id);
    splay(id);
    while(nxt[id][0])
    {
        id=nxt[id][0];
        down(id);
    }
    return id;
}

inline void makert(int id)
{
    access(id);
    splay(id);
    if(nxt[id][0])
        rev[id]^=true;
}

int n,i,j,k,q;
char buf[11];

int main()
{
    scanf("%d",&n);
    for(i=1;i<=n;++i)
        scanf("%d",val+i);
    scanf("%d",&q);
    while(q--)
    {
        scanf("%s␣%d␣%d",buf,&i,&j);
        switch(buf[0])
        {
            case 'b':
                if(getrt(i)==getrt(j))
                    puts("no");
                else
                {
                    puts("yes");
                    makert(i);
                    fa[i]=j;
                }
                break;
            case 'p':
                access(i);
                splay(i);
                val[i]=j;
                up(i);
                break;
            case 'e':
                if(getrt(i)!=getrt(j))
                    puts("impossible");
                else
                {
                    makert(i);
                    access(j);
                    splay(j);
                    printf("%d\n",sum[j]);
                }
                break;
        }
    }
    return 0;
}
```

1.6 segment tree - discretization.cpp

```
std::map<double,short>map;  //     hash
std::map<double,short>::iterator it;
double rmap[inf];  //     hash

short mid[MAX],cnt[MAX];
double len[MAX];

void make(const short &id,const short &l,const short &r)
{
    mid[id]=(l+r)>>1;
    if(l!=r)
    {
        make(id<<1,l,mid[id]);
        make(id<<1|1,mid[id]+1,r);
    }
}

void update(const short &id,const short &ll,const short &rr,const short &l,const short &r,const char &val)
{
    if(ll==l && rr==r)
    {
        cnt[id]+=val;
        if(cnt[id])
            len[id]=rmap[r]-rmap[l-1];
        else
            if(l!=r)
                len[id]=len[id<<1]+len[id<<1|1];
            else
```

```
                len[id]=0;
        return;
    }
    if(mid[id]>=r)
        update(id<<1,ll,mid[id],l,r,val);
    else
        if(mid[id]<l)
            update(id<<1|1,mid[id]+1,rr,l,r,val);
        else
        {
            update(id<<1,ll,mid[id],l,mid[id],val);
            update(id<<1|1,mid[id]+1,rr,mid[id]+1,r,val);
        }
    if(!cnt[id])
        len[id]=len[id<<1]+len[id<<1|1];
}

int main()
{
    n<<=1;
    map.clear();
    for(i=0;i<n;++i)
    {
        scanf(”%lf%lf%lf%lf%lf”,&x1,&y1,&x2,&y2,&d);
        if(x1>x2)
            std::swap(x1,x2);
        if(y1>y2)
            std::swap(y1,y2);
        sum+=(x2-x1)*(y2-y1)*d;
        ln[i].l=x1;
        ln[i].r=x2;
        ln[i].h=y1;
        ln[i].up=1;
        ln[++i].l=x1;
        ln[i].r=x2;
        ln[i].h=y2;
        ln[i].up=-1;
        map[x1]=1;
        map[x2]=1;
    }
    k=1;
    for(it=map.begin();it!=map.end();++it,++k)  //

    {
        it->second=k;
        rmap[k]=it->first;
    }
    std::sort(ln,ln+n);
    update(1,1,inf,map[ln[0].l]+1,map[ln[0].r],ln[0].up);
    for(i=1;i<n;++i)
    {
        //            len[1]
        //ln[i].h-ln[i-1].h

        update(1,1,inf,map[ln[i].l]+1,map[ln[i].r],ln[i].up);
    }
}
```

1.7 size-blanced binary search tree.cpp

```
template<class Tp>class sbt
{
    public:
        inline void init()
        {
            rt=cnt=l[0]=r[0]=sz[0]=0;
        }
        inline void ins(const Tp &a)
        {
            ins(rt,a);
        }
        inline void del(const Tp &a)
        {
            del(rt,a);
        }
        inline bool find(const Tp &a)
        {
            return find(rt,a);
        }
        inline Tp pred(const Tp &a)
        {
            return pred(rt,a);
        }
        inline Tp succ(const Tp &a)
        {
            return succ(rt,a);
        }
        inline bool empty()
        {
            return !sz[rt];
        }
        inline Tp min()
        {
            return min(rt);
        }
        inline Tp max()
        {
            return max(rt);
        }
        inline void delsmall(const Tp &a)
        {
            dels(rt,a);
        }
        inline int rank(const Tp &a)
        {
            return rank(rt,a);
        }
        inline Tp sel(const int &a)
        {
            return sel(rt,a);
        }
        inline Tp delsel(int a)
        {
            return delsel(rt,a);
        }
    private:
        int cnt,rt,l[MAXX],r[MAXX],sz[MAXX];
        Tp val[MAXX];
        inline void rro(int &pos)
        {
            int k(l[pos]);
            l[pos]=r[k];
            r[k]=pos;
            sz[k]=sz[pos];
            sz[pos]=sz[l[pos]]+sz[r[pos]]+1;
            pos=k;
        }
        inline void lro(int &pos)
        {
            int k(r[pos]);
            r[pos]=l[k];
            l[k]=pos;
            sz[k]=sz[pos];
            sz[pos]=sz[l[pos]]+sz[r[pos]]+1;
            pos=k;
        }
        inline void mt(int &pos,bool flag)
    {
        if(!pos)
            return;
        if(flag)
            if(sz[r[r[pos]]]>sz[l[pos]])
                lro(pos);
            else
                if(sz[l[r[pos]]]>sz[l[pos]])
                {
                    rro(r[pos]);
                    lro(pos);
                }
                else
                    return;
        else
            if(sz[l[l[pos]]]>sz[r[pos]])
                rro(pos);
            else
                if(sz[r[l[pos]]]>sz[r[pos]])
                {
                    lro(l[pos]);
                    rro(pos);
                }
                else
                    return;
        mt(l[pos],false);
        mt(r[pos],true);
        mt(pos,false);
        mt(pos,true);
    }
    void ins(int &pos,const Tp &a)
    {
        if(pos)
        {
            ++sz[pos];
            if(a<val[pos])
                ins(l[pos],a);
            else
                ins(r[pos],a);
            mt(pos,a>=val[pos]);
            return;
        }
        pos=++cnt;
        l[pos]=r[pos]=0;
        val[pos]=a;
        sz[pos]=1;
        }
        Tp del(int &pos,const Tp &a)
        {
            --sz[pos];
            if(val[pos]==a || (a<val[pos] && !l[pos]) || (a>val[pos] && !r[pos]))
            {
                Tp ret(val[pos]);
                if(!l[pos] || !r[pos])
                    pos=l[pos]+r[pos];
                else
                    val[pos]=del(l[pos],val[pos]+1);
                return ret;
            }
            else
                if(a<val[pos])
                    return del(l[pos],a);
                else
                    return del(r[pos],a);
        }
        bool find(int &pos,const Tp &a)
        {
            if(!pos)
                return false;
            if(a<val[pos])
                return find(l[pos],a);
            else
                return (val[pos]==a || find(r[pos],a));
        }
        Tp pred(int &pos,const Tp &a)
        {
            if(!pos)
                return a;
            if(a>val[pos])
            {
                Tp ret(pred(r[pos],a));
                if(ret==a)
                    return val[pos];
                else
                    return ret;
            }
            return pred(l[pos],a);
        }
        Tp succ(int &pos,const Tp &a)
        {
            if(!pos)
```

```
                return a;
            if(a<val[pos])
            {
                Tp ret(succ(l[pos],a));
                if(ret==a)
                    return val[pos];
                else
                    return ret;
            }
            return succ(r[pos],a);
        }
        Tp min(int &pos)
        {
            if(l[pos])
                return min(l[pos]);
            else
                return val[pos];
        }
        Tp max(int &pos)
        {
            if(r[pos])
                return max(r[pos]);
            else
                return val[pos];
        }
        void dels(int &pos,const Tp &v)
        {
            if(!pos)
                return;
            if(val[pos]<v)
            {
                pos=r[pos];
                dels(pos,v);
                return;
            }
            dels(l[pos],v);
            sz[pos]=1+sz[l[pos]]+sz[r[pos]];
        }
        int rank(const int &pos,const Tp &v)
        {
            if(val[pos]==v)
                return sz[l[pos]]+1;
            if(v<val[pos])
                return rank(l[pos],v);
            return rank(r[pos],v)+sz[l[pos]]+1;
        }
        Tp sel(const int &pos,const int &v)
        {
            if(sz[l[pos]]+1==v)
                return val[pos];
            if(v>sz[l[pos]])
                return sel(r[pos],v-sz[l[pos]]-1);
            return sel(l[pos],v);
        }
        Tp delsel(int &pos,int k)
        {
            --sz[pos];
            if(sz[l[pos]]+1==k)
            {
                Tp re(val[pos]);
                if(!l[pos] || !r[pos])
                    pos=l[pos]+r[pos];
                else
                    val[pos]=del(l[pos],val[pos]+1);
                return re;
            }
            if(k>sz[l[pos]])
                return delsel(r[pos],k-1-sz[l[pos]]);
            return delsel(l[pos],k);
        }
};
```

1.8 sparse table - rectangle.cpp

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

#define MAXX 310

int mat[MAXX][MAXX];
int table[9][9][MAXX][MAXX];
int n;
short lg[MAXX];

int main()
{
    for(int i(2);i<MAXX;++i)
        lg[i]=lg[i>>1]+1;
    int T;
    std::cin >> T;
    while (T--)
    {
        std::cin >> n;
        for (int i = 0; i < n; ++i)
            for (int j = 0; j < n; ++j)
            {
                std::cin >> mat[i][j];
                table[0][0][i][j] = mat[i][j];
            }

        //


        for(int i=0;i<=lg[n];++i) // width
        {
            for(int j=0;j<=lg[n];++j) //height
            {
                if(i==0 && j==0)
                    continue;
                for(int ii=0;ii+(1<<j)<=n;++ii)
                    for(int jj=0;jj+(1<<i)<=n;++jj)
                        if(i==0)
                            table[i][j][ii][jj]=std::min(
                                table[i][j-1][ii][jj],
                                table[i][j-1][ii+(1<<(j-1)
                                )][jj]);
                        else
                            table[i][j][ii][jj]=std::min(
                                table[i-1][j][ii][jj],
                                table[i-1][j][ii][jj+(1<<(
                                i-1))]);
            }
        }
        long long N;
        std::cin >> N;
        int r1, c1, r2, c2;
        for (int i = 0; i < N; ++i)
        {
            scanf("%d%d%d%d",&r1,&c1,&r2,&c2);
            --r1;
            --c1;
            --r2;
            --c2;
            int w=lg[c2-c1+1];
            int h=lg[r2-r1+1];
            printf("%d\n",std::min(table[w][h][r1][c1],
               std::min(table[w][h][r1][c2-(1<<w)+1],std
                    ::min(table[w][h][r2-(1<<h)+1][c1],table[w
                    ][h][r2-(1<<h)+1][c2-(1<<w)+1]))));
        }
    }
    return 0;
}
```

1.9 sparse table - square.cpp

```
int num[MAXX][MAXX],max[MAXX][MAXX][10];
short lg[MAXX];

int main()
{
    for(i=2;i<MAXX;++i)
        lg[i]=lg[i>>1]+1;
    scanf("%hd␣%d",&n,&q);
    for(i=0;i<n;++i)
        for(j=0;j<n;++j)
        {
            scanf("%d",num[i]+j);
            max[i][j][0]=num[i][j];
        }
    for(k=1;k<=lg[n];++k)
    {
        l=n+1-(1<<k);
        for(i=0;i<l;++i)
            for(j=0;j<l;++j)
                max[i][j][k]=std::max(std::max(max[i][j][
                    k-1],max[i+(1<<(k-1))][j][k-1]),std::
                    max(max[i][j+(1<<(k-1))][k-1],max[i
                    +(1<<(k-1))][j+(1<<(k-1))][k-1]));
    }
    printf("Case␣%hd:\n",t);
    while(q--)
    {
        scanf("%hd␣%hd␣%hd",&i,&j,&l);
        --i;
        --j;
        k=lg[l];
        printf("%d\n",std::max(std::max(max[i][j][k],max[
            i][j+l-(1<<k)][k]),std::max(max[i+l-(1<<k)][j
            ][k],max[i+l-(1<<k)][j+l-(1<<k)][k])));
    }
}
```

1.10 sparse table.cpp

```
int num[MAXX],min[MAXX][20];
int lg[MAXX];


int main()
{
    for(i=2;i<MAXX;++i)
        lg[i]=lg[i>>1]+1;
    scanf("%d␣%d",&n,&q);
    for(i=1;i<=n;++i)
    {
        scanf("%d",num+i);
        min[i][0]=num[i];
    }
    for(j=1;j<=lg[n];++j)
    {
        l=n+1-(1<<j);
        j_=j-1;
        j__=(1<<j_);
        for(i=1;i<=l;++i)
            min[i][j]=std::min(min[i][j_],min[i+j__][j_]);
    }
    printf("Case␣%hd:\n",t);
    while(q--)
    {
        scanf("%d␣%d",&i,&j);
        k=lg[j-i+1];
        printf("%d\n",std::min(min[i][k],min[j-(1<<k)+1][k]));
    }
}
```

1.11 treap.cpp

```
#include<cstdlib>
#include<ctime>
#include<cstring>

struct node
{
    node *ch[2];
    int sz,val,key;
    node(){memset(this,0,sizeof(node));}
    node(int a);
}*null;

node::node(int a):sz(1),val(a),key(rand()-1){ch[0]=ch[1]=null;}

class Treap
{
    inline void up(node *pos)
    {
        pos->sz=pos->ch[0]->sz+pos->ch[1]->sz+1;
    }
    inline void rot(node *&pos,int tp)
    {
        node *k(pos->ch[tp]);
        pos->ch[tp]=k->ch[tp^1];
        k->ch[tp^1]=pos;
        up(pos);
        up(k);
        pos=k;
    }

    void insert(node *&pos,int val)
    {
        if(pos!=null)
        {
            int t(val>=pos->val);
            insert(pos->ch[t],val);
            if(pos->ch[t]->key<pos->key)
                rot(pos,t);
            else
                up(pos);
            return;
        }
        pos=new node(val);
    }
    void rec(node *pos)
    {
        if(pos!=null)
        {
            rec(pos->ch[0]);
            rec(pos->ch[1]);
            delete pos;
        }
    }
    inline int sel(node *pos,int k)
    {
        while(pos->ch[0]->sz+1!=k)
            if(pos->ch[0]->sz>=k)
                pos=pos->ch[0];
            else
            {
                k-=pos->ch[0]->sz+1;
                pos=pos->ch[1];
            }
        return pos->val;
    }
    void del(node *&pos,int val)
    {
        if(pos!=null)
        {
            if(pos->val==val)
            {
                int t(pos->ch[1]->key<pos->ch[0]->key);
                if(pos->ch[t]==null)
                {
                    delete pos;
                    pos=null;
                    return;
                }
                rot(pos,t);
                del(pos->ch[t^1],val);
            }
            else
                del(pos->ch[val>pos->val],val);
            up(pos);
        }
    }
    public:
    node *rt;

    Treap():rt(null){}
    inline void insert(int val)
    {
        insert(rt,val);
    }
    inline void reset()
    {
        rec(rt);
        rt=null;
    }
    inline int sel(int k)
    {
        if(k<1 || k>rt->sz)
            return 0;
        return sel(rt,rt->sz+1-k);
    }
    inline void del(int val)
    {
        del(rt,val);
    }
    inline int size()
    {
        return rt->sz;
    }
}treap[MAXX];

init:
{
    srand(time(0));
    null=new node();
    null->val=0xc0c0c0c0;
    null->sz=0;
    null->key=RAND_MAX;
    null->ch[0]=null->ch[1]=null;
    for(i=0;i<MAXX;++i)
        treap[i].rt=null;
}
```

Chapter 2

geometry

2.1 3D.cpp

```
struct pv
{
        double x,y,z;
        pv() {}
        pv(double xx,double yy,double zz):x(xx),y(yy),z(zz)     {}
        pv operator -(const pv& b)const
        {
                return pv(x-b.x,y-b.y,z-b.z);
        }
        pv operator *(const pv& b)const
        {
                return pv(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);
        }
        double operator &(const pv& b)const
        {
                return x*b.x+y*b.y+z*b.z;
        }
};

//
double Norm(pv p)
{
        return sqrt(p&p);
}

//‘              V      theta        ‘
```

```
pv Trans(pv pa,pv V,double theta)
{
    double s = sin(theta);
    double c = cos(theta);
    double x,y,z;
    x = V.x;
    y = V.y;
    z = V.z;
    pv pp =
        pv(
                (x*x*(1-c)+c)*pa.x+(x*y*(1-c)-z*s)*pa.y+(
                    x*z*(1-c)+y*s)*pa.z,
                (y*x*(1-c)+z*s)*pa.x+(y*y*(1-c)+c)*pa.y+(
                    y*z*(1-c)-x*s)*pa.z,
                (x*z*(1-c)-y*s)*pa.x+(y*z*(1-c)+x*s)*pa.y
                    +(z*z*(1-c)+c)*pa.z
          );
    return pp;
}

//

x=r*sin(  )*cos(  );
y=r*sin(  )*sin(  );
z=r*cos(  );

r=sqrt(x*2+y*2+z*2);//??
r=sqrt(x^2+y^2+z^2);//??

  =atan(y/x);
  =acos(z/r);

 r   [0,    ]
      [0,2   ]
      [0,   ]

 lat1  [-  /2,  /2]
 lng1  [-   ,   ]

pv getpv(double lat,double lng,double r)
{
        lat += pi/2;
        lng += pi;
        return
    pv(r*sin(lat)*cos(lng),r*sin(lat)*sin(lng),r*cos(lat)
        );
```

```
}

//

#include<cstdio>
#include<cmath>

#define MAXX 1111

char buf[MAXX];
const double r=6875.0/2,pi=acos(-1.0);
double a,b,c,x1,x2,y2,ans;

int main()
{
    double y1;
    while(gets(buf)!=NULL)
    {
        gets(buf);
        gets(buf);

        scanf("%lf^%lf'%lf\"␣%s\n",&a,&b,&c,buf);
        x1=a+b/60+c/3600;
        x1=x1*pi/180;
        if(buf[0]=='S')
            x1=-x1;

        scanf("%s",buf);
        scanf("%lf^%lf'%lf\"␣%s\n",&a,&b,&c,buf);
        y1=a+b/60+c/3600;
        y1=y1*pi/180;
        if(buf[0]=='W')
            y1=-y1;

        gets(buf);

        scanf("%lf^%lf'%lf\"␣%s\n",&a,&b,&c,buf);
        x2=a+b/60+c/3600;
        x2=x2*pi/180;
        if(buf[0]=='S')
            x2=-x2;

        scanf("%s",buf);
        scanf("%lf^%lf'%lf\"␣%s\n",&a,&b,&c,buf);
        y2=a+b/60+c/3600;
        y2=y2*pi/180;
        if(buf[0]=='W')
            y2=-y2;

        ans=acos(cos(x1)*cos(x2)*cos(y1-y2)+sin(x1)*sin(x2))*r;
        printf("The distance to the iceberg: %.2lf miles.\n",ans);
        if(ans+0.005<100)
            puts("DANGER!");

        gets(buf);
    }
    return 0;
}

inline bool ZERO(const double &a)
{
    return fabs(a)<eps;
}

//
inline bool ZERO(pv p)
{
    return (ZERO(p.x) && ZERO(p.y) && ZERO(p.z));
}

//
bool LineIntersect(Line3D L1, Line3D L2)
{
    pv s = L1.s-L1.e;
    pv e = L2.s-L2.e;
    pv p  = s*e;
    if (ZERO(p))
        return false;    //
    p = (L2.s-L1.e)*(L1.s-L1.e);
    return ZERO(p&L2.e);         //
}

//
bool inter(pv a,pv b,pv c,pv d)
{
    pv ret = (a-b)*(c-d);
    pv t1 = (b-a)*(c-a);
    pv t2 = (b-a)*(d-a);
    pv t3 = (d-c)*(a-c);
    pv t4 = (d-c)*(b-c);
    return sgn(t1&ret)*sgn(t2&ret) < 0 && sgn(t3&ret)*sgn(t4&ret) < 0;
}

//
bool OnLine(pv p, Line3D L)
{
    return ZERO((p-L.s)*(L.e-L.s));
}

//
bool OnSeg(pv p, Line3D L)
{
    return (ZERO((L.s-p)*(L.e-p)) && EQ(Norm(p-L.s)+Norm(p-L.e),Norm(L.e-L.s)));
}

//
double Distance(pv p, Line3D L)
{
    return (Norm((p-L.s)*(L.e-L.s))/Norm(L.e-L.s));
}

//
//              [0,   ]
double Inclination(Line3D L1, Line3D L2)
{
    pv u = L1.e - L1.s;
    pv v = L2.e - L2.s;
    return acos( (u & v) / (Norm(u)*Norm(v)) );
}
```

2.2 3DCH.cpp

```
#include<cstdio>
#include<cmath>
#include<vector>
#include<algorithm>

#define MAXX 1111
#define eps 1e-8
#define inf 1e20

struct pv
{
    double x,y,z;
    pv(){}
    pv(const double &xx,const double &yy,const double &zz):x(xx),y(yy),z(zz){}
    inline pv operator-(const pv &i)const
    {
        return pv(x-i.x,y-i.y,z-i.z);
    }
    inline pv operator*(const pv &i)const //
    {
        return pv(y*i.z-z*i.y,z*i.x-x*i.z,x*i.y-y*i.x);
    }
    inline double operator^(const pv &i)const //
    {
        return x*i.x+y*i.y+z*i.z;
    }
    inline double len()
    {
        return sqrt(x*x+y*y+z*z);
    }
};

struct pla
{
    short a,b,c;
    bool ok;
    pla(){}
    pla(const short &aa,const short &bb,const short &cc):a(aa),b(bb),c(cc),ok(true){}
    inline void set();
    inline void print()
    {
        printf("%hd %hd %hd\n",a,b,c);
    }
};

pv pnt[MAXX];
std::vector<pla>fac;
short to[MAXX][MAXX];

inline void pla::set()
{
    to[a][b]=to[b][c]=to[c][a]=fac.size();
}

inline double ptof(const pv &p,const pla &f) //
```

```
{
    return (pnt[f.b]-pnt[f.a])*(pnt[f.c]-pnt[f.a])^(p-pnt[f.a]);
}

inline double vol(const pv &a,const pv &b,const pv &c,const pv &d)//          *6
{
    return (b-a)*(c-a)^(d-a);
}

inline double ptof(const pv &p,const short &f) // p     f
{
    return fabs(vol(pnt[fac[f].a],pnt[fac[f].b],pnt[fac[f].c],p)/((pnt[fac[f].b]-pnt[fac[f].a])*(pnt[fac[f].c]-pnt[fac[f].a])).len());
}

void dfs(const short&,const short&);

void deal(const short &p,const short &a,const short &b)
{
    if(fac[to[a][b]].ok)
        if(ptof(pnt[p],fac[to[a][b]])>eps)
            dfs(p,to[a][b]);
        else
        {
            pla add(b,a,p);
            add.set();
            fac.push_back(add);
        }
}

void dfs(const short &p,const short &now)
{
    fac[now].ok=false;
    deal(p,fac[now].b,fac[now].a);
    deal(p,fac[now].c,fac[now].b);
    deal(p,fac[now].a,fac[now].c);
}

inline void make()
{
    fac.resize(0);
    if(n<4)
```

```
        return;

    for(i=1;i<n;++i)
        if((pnt[0]-pnt[i]).len()>eps)
        {
            std::swap(pnt[i],pnt[1]);
            break;
        }
    if(i==n)
        return;

    for(i=2;i<n;++i)
        if(((pnt[0]-pnt[1])*(pnt[1]-pnt[i])).len()>eps)
        {
            std::swap(pnt[i],pnt[2]);
            break;
        }
    if(i==n)
        return;

    for(i=3;i<n;++i)
        if(fabs((pnt[0]-pnt[1])*(pnt[1]-pnt[2])^(pnt[2]-pnt[i]))>eps)
        {
            std::swap(pnt[3],pnt[i]);
            break;
        }
    if(i==n)
        return;

    for(i=0;i<4;++i)
    {
        pla add((i+1)%4,(i+2)%4,(i+3)%4);
        if(ptof(pnt[i],add)>0)
            std::swap(add.c,add.b);
        add.set();
        fac.push_back(add);
    }
    for(;i<n;++i)
        for(j=0;j<fac.size();++j)
            if(fac[j].ok && ptof(pnt[i],fac[j])>eps)
            {
                dfs(i,j);
                break;
            }

    short tmp(fac.size());
    fac.resize(0);
    for(i=0;i<tmp;++i)
        if(fac[i].ok)
            fac.push_back(fac[i]);
}

inline pv gc() //
{
    pv re(0,0,0),o(0,0,0);
    double all(0),v;
    for(i=0;i<fac.size();++i)
    {
        v=vol(o,pnt[fac[i].a],pnt[fac[i].b],pnt[fac[i].c]);
        re+=(pnt[fac[i].a]+pnt[fac[i].b]+pnt[fac[i].c])*0.25*v;
        all+=v;
    }
    return re*(1/all);
}

inline bool same(const short &s,const short &t) //
{
    pv &a=pnt[fac[s].a],&b=pnt[fac[s].b],&c=pnt[fac[s].c];
    return fabs(vol(a,b,c,pnt[fac[t].a]))<eps && fabs(vol(a,b,c,pnt[fac[t].b]))<eps && fabs(vol(a,b,c,pnt[fac[t].c]))<eps;
}

//
inline short facetcnt()
{
    short ans=0;
    for(short i=0;i<fac.size();++i)
    {
        for(j=0;j<i;++j)
            if(same(i,j))
                break;
        if(j==i)
            ++ans;
    }
    return ans;
}
```

```

//
inline short trianglecnt()
{
    return fac.size();
}

//                              *2
inline double area(const pv &a,const pv &b,const pv &c)
{
        return (b-a)*(c-a).len();
}

//
inline double area()
{
    double ret(0);
    for(i=0;i<fac.size();++i)
        ret+=area(pnt[fac[i].a],pnt[fac[i].b],pnt[fac[i].
            c]);
    return ret/2;
}

//
inline double volume()
{
    pv o(0,0,0);
    double ret(0);
    for(short i(0);i<fac.size();++i)
        ret+=vol(o,pnt[fac[i].a],pnt[fac[i].b],pnt[fac[i
            ].c]);
    return fabs(ret/6);
}
```

2.3 circle & ploy's area.cpp

```
bool InCircle(Point a,double r)
{
        return cmp(a.x*a.x+a.y*a.y,r*r) <= 0;
        //‘
                                   EPS
            ‘
}

double CalcArea(Point a,Point b,double r)
{
    Point p[4];
    int tot = 0;
    p[tot++] = a;

    Point tv = Point(a,b);
    Line tmp = Line(Point(0,0),Point(tv.y,-tv.x));
    Point near = LineToLine(Line(a,b),tmp);
    if (cmp(near.x*near.x+near.y*near.y,r*r) <= 0)
    {
        double A,B,C;
        A = near.x*near.x+near.y*near.y;
        C = r;
        B = C*C-A;
        double tvl = tv.x*tv.x+tv.y*tv.y;
        double tmp = sqrt(B/tvl); //

        p[tot] = Point(near.x+tmp*tv.x,near.y+tmp*tv.y);
        if (OnSeg(Line(a,b),p[tot]) == true) tot++;
        p[tot] = Point(near.x-tmp*tv.x,near.y-tmp*tv.y);
        if (OnSeg(Line(a,b),p[tot]) == true) tot++;
    }
    if (tot == 3)
    {
        if (cmp(Point(p[0],p[1]).Length(),Point(p[0],p[2]).Length()) > 0)
            swap(p[1],p[2]);
    }
    p[tot++] = b;

    double res = 0.0,theta,a0,a1,sgn;
    for (int i = 0;i < tot-1;i++)
    {
        if (InCircle(p[i],r) == true && InCircle(p[i+1],r) == true)
        {
            res += 0.5*xmult(p[i],p[i+1]);
        }
        else
        {
            a0 = atan2(p[i+1].y,p[i+1].x);
            a1 = atan2(p[i].y,p[i].x);
            if (a0 < a1)    a0 += 2*pi;
```

```
                        theta = a0-a1;
                        if (cmp(theta,pi) >= 0) theta = 2*pi-theta;
                        sgn = xmult(p[i],p[i+1])/2.0;
                        if (cmp(sgn,0) < 0) theta = -theta;
                        res += 0.5*r*r*theta;
                }
        }
        return res;
}

//

area2 = 0.0;
for (int i = 0;i < resn;i++) //

    area2 += CalcArea(p[i],p[(i+1)%resn],r);
```

2.4 circle's area.cpp

```
//
{
    for (int i = 0; i < n; i++)
    {
        scanf("%lf%lf%lf",&c[i].c.x,&c[i].c.y,&c[i].r);
        del[i] = false;
    }
    for (int i = 0; i < n; i++)
        if (del[i] == false)
        {
            if (c[i].r == 0.0)
                del[i] = true;
            for (int j = 0; j < n; j++)
                if (i != j)
                    if (del[j] == false)
                        if (cmp(Point(c[i].c,c[j].c).Len()+c[i].r,c[j].r) <= 0)
                            del[i] = true;
        }
    tn = n;
    n = 0;
    for (int i = 0; i < tn; i++)
        if (del[i] == false)
            c[n++] = c[i];
}
```

```

//ans[i]                    i
const double pi = acos(-1.0);
const double eps = 1e-8;
struct Point
{
    double x,y;
    Point(){}
    Point(double _x,double _y)
    {
        x = _x;
        y = _y;
    }
    double Length()
    {
        return sqrt(x*x+y*y);
    }
};
struct Circle
{
    Point c;
    double r;
};
struct Event
{
    double tim;
    int typ;
    Event(){}
    Event(double _tim,int _typ)
    {
        tim = _tim;
        typ = _typ;
    }
};

int cmp(const double& a,const double& b)
{
    if (fabs(a-b) < eps)     return 0;
    if (a < b)   return -1;
    return 1;
}

bool Eventcmp(const Event& a,const Event& b)
{
    return cmp(a.tim,b.tim) < 0;
}
```

```

double Area(double theta,double r)
{
    return 0.5*r*r*(theta-sin(theta));
}

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

int n,cur,tote;
Circle c[1000];
double ans[1001],pre[1001],AB,AC,BC,theta,fai,a0,a1;
Event e[4000];
Point lab;

int main()
{
    while (scanf("%d",&n) != EOF)
    {
        for (int i = 0;i < n;i++)
            scanf("%lf%lf%lf",&c[i].c.x,&c[i].c.y,&c[i].r);
        for (int i = 1;i <= n;i++)
            ans[i] = 0.0;
        for (int i = 0;i < n;i++)
        {
            tote = 0;
            e[tote++] = Event(-pi,1);
            e[tote++] = Event(pi,-1);
            for (int j = 0;j < n;j++)
                if (j != i)
                {
                    lab = Point(c[j].c.x-c[i].c.x,c[j].c.y-c[i].c.y);
                    AB = lab.Length();
                    AC = c[i].r;
                    BC = c[j].r;
                    if (cmp(AB+AC,BC) <= 0)
                    {
                        e[tote++] = Event(-pi,1);
                        e[tote++] = Event(pi,-1);
                        continue;
                    }
                    if (cmp(AB+BC,AC) <= 0) continue;
                if (cmp(AB,AC+BC) > 0)  continue;
                theta = atan2(lab.y,lab.x);
                fai = acos((AC*AC+AB*AB-BC*BC)/(2.0*AC*AB));
                a0 = theta-fai;
                if (cmp(a0,-pi) < 0)     a0 += 2*pi;
                a1 = theta+fai;
                if (cmp(a1,pi) > 0)  a1 -= 2*pi;
                if (cmp(a0,a1) > 0)
                {
                    e[tote++] = Event(a0,1);
                    e[tote++] = Event(pi,-1);
                    e[tote++] = Event(-pi,1);
                    e[tote++] = Event(a1,-1);
                }
                else
                {
                    e[tote++] = Event(a0,1);
                    e[tote++] = Event(a1,-1);
                }
            }
        sort(e,e+tote,Eventcmp);
        cur = 0;
        for (int j = 0;j < tote;j++)
        {
            if (cur != 0 && cmp(e[j].tim,pre[cur]) != 0)
            {
                ans[cur] += Area(e[j].tim-pre[cur],c[i].r);
                ans[cur] += xmult(Point(c[i].c.x+c[i].r*cos(pre[cur]),c[i].c.y+c[i].r*sin(pre[cur])),
                        Point(c[i].c.x+c[i].r*cos(e[j].tim),c[i].c.y+c[i].r*sin(e[j].tim)))/2.0;
            }
            cur += e[j].typ;
            pre[cur] = e[j].tim;
        }
    }
    for (int i = 1;i < n;i++)
        ans[i] -= ans[i+1];
    for (int i = 1;i <= n;i++)
        printf("[%d]␣=␣%.3f\n",i,ans[i]);
}
```

```
    return 0;
}
```

2.5 circle.cpp

```
//
#include<cstdio>
#include<cmath>
#include<vector>
#include<algorithm>

#define MAXX 333
#define  eps 1e-8

struct pv
{
    double x,y;
    pv(){}
    pv(const double &xx,const double &yy):x(xx),y(yy){}
    inline pv operator-(const pv &i)const
    {
        return pv(x-i.x,y-i.y);
    }
    inline double cross(const pv &i)const
    {
        return x*i.y-y*i.x;
    }
    inline void print()
    {
        printf("%lf %lf\n",x,y);
    }
    inline double len()
    {
        return sqrt(x*x+y*y);
    }
}pnt[MAXX];

struct node
{
    double k;
    bool flag;
    node(){}
    node(const double &kk,const bool &ff):k(kk),flag(ff){}
    inline bool operator<(const node &i)const
    {
        return k<i.k;
    }
};

std::vector<node>alpha;

short n,i,j,k,l;
short ans,sum;
double R=2;
double theta,phi,d;
const double pi(acos(-1.0));

int main()
{
    alpha.reserve(MAXX<<1);
    while(scanf(”%hd”,&n),n)
    {
        for(i=0;i<n;++i)
            scanf(”%lf %lf”,&pnt[i].x,&pnt[i].y);
        ans=0;
        for(i=0;i<n;++i)
        {
            alpha.resize(0);
            for(j=0;j<n;++j)
                if(i!=j)
                {
                    if((d=(pnt[i]-pnt[j]).len())>R)
                        continue;
                    if((theta=atan2(pnt[j].y-pnt[i].y,pnt[j].x-pnt[i].x))<0)
                        theta+=2*pi;
                    phi=acos(d/R);
                    alpha.push_back(node(theta-phi,true));
                    alpha.push_back(node(theta+phi,false));
                }
            std::sort(alpha.begin(),alpha.end());
            for(j=0;j<alpha.size();++j)
            {
                if(alpha[j].flag)
                    ++sum;
                else
                    --sum;
                ans=std::max(ans,sum);
            }
        }
        printf(”%hd\n”,ans+1);
    }
    return 0;
}

//

#include<cstdio>
#include<cmath>

#define MAXX 511
#define eps 1e-8

struct pv
{
    double x,y;
    pv(){}
    pv(const double &xx,const double &yy):x(xx),y(yy){}
    inline pv operator-(const pv &i)const
    {
        return pv(x-i.x,y-i.y);
    }
    inline pv operator+(const pv &i)const
    {
        return pv(x+i.x,y+i.y);
    }
    inline double cross(const pv &i)const
    {
        return x*i.y-y*i.x;
    }
    inline double len()
    {
        return sqrt(x*x+y*y);
    }
    inline pv operator/(const double &a)const
    {
        return pv(x/a,y/a);
    }
    inline pv operator*(const double &a)const
    {
        return pv(x*a,y*a);
    }
}pnt[MAXX],o,tl,lt,aa,bb,cc,dd;

short n,i,j,k,l;
```

```
double r,u;

inline pv ins(const pv &a1,const pv &a2,const pv &b1,const pv &b2)
{
    tl=a2-a1;
    lt=b2-b1;
    u=(b1-a1).cross(lt)/(tl).cross(lt);
    return a1+tl*u;
}

inline pv get(const pv &a,const pv &b,const pv &c)
{
    aa=(a+b)/2;
    bb.x=aa.x-a.y+b.y;
    bb.y=aa.y+a.x-b.x;
    cc=(a+c)/2;
    dd.x=cc.x-a.y+c.y;
    dd.y=cc.y+a.x-c.x;
    return ins(aa,bb,cc,dd);
}

int main()
{
    while(scanf(”%hd”,&n),n)
    {
        for(i=0;i<n;++i)
            scanf(”%lf %lf”,&pnt[i].x,&pnt[i].y);
        o=pnt[0];
        r=0;
        for(i=1;i<n;++i)
            if((pnt[i]-o).len()>r+eps)
            {
                o=pnt[i];
                r=0;
                for(j=0;j<i;++j)
                    if((pnt[j]-o).len()>r+eps)
                    {
                        o=(pnt[i]+pnt[j])/2;
                        r=(o-pnt[j]).len();
                        for(k=0;k<j;++k)
                            if((o-pnt[k]).len()>r+eps)
                            {
                                o=get(pnt[i],pnt[j],pnt[k]);
                                r=(o-pnt[i]).len();
                            }
                    }
            }
        printf("%.2lf␣%.2lf␣%.2lf\n",o.x,o.y,r);
    }
    return 0;
}

//
double dis(int x,int y)
{
    return sqrt((double)(x*x+y*y));
}

double area(int x1,int y1,int x2,int y2,double r1,double r2)
{
    double s=dis(x2-x1,y2-y1);
    if(r1+r2<s) return 0;
    else if(r2-r1>s) return PI*r1*r1;
    else if(r1-r2>s) return PI*r2*r2;
    double q1=acos((r1*r1+s*s-r2*r2)/(2*r1*s));
    double q2=acos((r2*r2+s*s-r1*r1)/(2*r2*s));
    return (r1*r1*q1+r2*r2*q2-r1*s*sin(q1));
}

//
{
    for (int i = 0; i < 3; i++)
        scanf("%lf%lf",&p[i].x,&p[i].y);
    tp = pv((p[0].x+p[1].x)/2,(p[0].y+p[1].y)/2);
    l[0] = Line(tp,pv(tp.x-(p[1].y-p[0].y),tp.y+(p[1].x-p[0].x)));
    tp = pv((p[0].x+p[2].x)/2,(p[0].y+p[2].y)/2);
    l[1] = Line(tp,pv(tp.x-(p[2].y-p[0].y),tp.y+(p[2].x-p[0].x)));
    tp = LineToLine(l[0],l[1]);
    r = pv(tp,p[0]).Length();
    printf("(%.6f,%.6f,%.6f)\n",tp.x,tp.y,r);
}

//
{
    for (int i = 0; i < 3; i++)
        scanf("%lf%lf",&p[i].x,&p[i].y);
    if (xmult(pv(p[0],p[1]),pv(p[0],p[2])) < 0)
```

```
        swap(p[1],p[2]);
    for (int i = 0; i < 3; i++)
        len[i] = pv(p[i],p[(i+1)%3]).Length();
    tr = (len[0]+len[1]+len[2])/2;
    r = sqrt((tr-len[0])*(tr-len[1])*(tr-len[2])/tr);
    for (int i = 0; i < 2; i++)
    {
        v = pv(p[i],p[i+1]);
        tv = pv(-v.y,v.x);
        tr = tv.Length();
        tv = pv(tv.x*r/tr,tv.y*r/tr);
        tp = pv(p[i].x+tv.x,p[i].y+tv.y);
        l[i].s = tp;
        tp = pv(p[i+1].x+tv.x,p[i+1].y+tv.y);
        l[i].e = tp;
    }
    tp = LineToLine(l[0],l[1]);
    printf("(%.6f,%.6f,%.6f)\n",tp.x,tp.y,r);
}
```

2.6 closest point pair.cpp

```
//                  1

struct Point {double x, y;} p[10], t[10];
bool cmpx(const Point& i, const Point& j) {return i.x < j.x;}
bool cmpy(const Point& i, const Point& j) {return i.y < j.y;}

double DnC(int L, int R)
{
    if (L >= R) return 1e9; //


    /*
        Divide
        */

    int M = (L + R) / 2;

    /*  Conquer                                        */

    double d = min(DnC(L,M), DnC(M+1,R));
    //  if (d == 0.0) return d; //

    /* Merge                      Y          O(NlogN) */

    int N = 0;  //
    for (int i=M;   i>=L && p[M].x - p[i].x < d; --i) t[N++] = p[i];
    for (int i=M+1; i<=R && p[i].x - p[M].x < d; ++i) t[N++] = p[i];
    sort(t, t+N, cmpy); // Quicksort O(NlogN)

    /* Merge                              O(N) */

    for (int i=0; i<N-1; ++i)
        for (int j=1; j<=2 && i+j<N; ++j)
            d = min(d, distance(t[i], t[i+j]));

    return d;
}

double closest_pair()
{
    sort(p, p+10, cmpx);
    return DnC(0, N-1);
}


//          2

struct Point {double x, y;} p[10], t[10];
bool cmpx(const Point& i, const Point& j) {return i.x < j.x;}
bool cmpy(const Point& i, const Point& j) {return i.y < j.y;}

double DnC(int L, int R)
{
    if (L >= R) return 1e9; //


    /* Divide */

    int M = (L + R) / 2;

    //
                        X

    double x = p[M].x;

    /*  Conquer                                             */

    //                                Y
    double d = min(DnC(L,M), DnC(M+1,R));
    //  if (d == 0.0) return d; //

    /*
        Merge                                      Y                O
        (N)     */

    //
                                                                   Y

    int N = 0;  //
    for (int i=0; i<=M; ++i)
        if (x - p[i].x < d)
            t[N++] = p[i];

    //
                                                                   Y

    int P = N;  //  P
    for (int i=M+1; i<=R; ++i)
        if (p[i].x - x < d)
            t[N++] = p[i];

    //    Y                  Merge
        Sort
    inplace_merge(t, t+P, t+N, cmpy);

    /*  Merge                                       O (N)
          */

    for (int i=0; i<N; ++i)
        for (int j=1; j<=2 && i+j<N; ++j)
            d = min(d, distance(t[i], t[i+j]));

    /*  Merge            Y                        O (N)
        */

    //                                                    Merge Sort
    inplace_merge(p+L, p+M+1, p+R+1, cmpy);

    return d;
}

double closest_pair()
{
    sort(p, p+10, cmpx);
    return DnC(0, N-1);
}

//mzry
//
double calc_dis(Point &a ,Point &b) {
        return sqrt((a.x-b.x)*(a.x-b.x) + (a.y-b.y)*(a.y-b.y));
}
//
bool operator<(const Point &a ,const Point &b) {
        if(a.y != b.y) return a.x < b.x;
        return a.x < b.x;
}
double Gao(int l ,int r ,Point pnts[]) {
        double ret = inf;
        if(l == r) return ret;
        if(l+1 ==r) {
                ret = min(calc_dis(pnts[l],pnts[l+1]) , ret);
                return ret;
        }
        if(l+2 ==r) {
                ret = min(calc_dis(pnts[l],pnts[l+1]) , ret);
                ret = min(calc_dis(pnts[l],pnts[l+2]) , ret);
                ret = min(calc_dis(pnts[l+1],pnts[l+2]) , ret);
                return ret;
        }

        int mid = l+r>>1;
        ret = min (ret ,Gao(l ,mid,pnts));
        ret = min (ret , Gao(mid+1, r,pnts));

        for(int c = l ; c<=r; c++)
                for(int d = c+1; d <=c+7 && d<=r; d++) {
                        ret = min(ret , calc_dis(pnts[c], pnts[d]));
                }
        return ret;
}

//
#include <iostream>
#include <cstdio>
#include <cstring>
#include <map>
#include <vector>
#include <cmath>
#include <algorithm>
#define Point pair<double,double>
using namespace std;

const int step[9][2] = {{-1,-1},{-1,0},{-1,1},{0,-1},{0,0},{0,1},{1,-1},{1,0},{1,1}};
int n,x,y,nx,ny;
map<pair<int,int>,vector<Point > > g;
vector<Point > tmp;
Point p[20000];
double tx,ty,ans,nowans;
vector<Point >::iterator it,op,ed;
pair<int,int> gird;
bool flag;

double Dis(Point p0,Point p1)
{
        return sqrt((p0.first-p1.first)*(p0.first-p1.first)+
                                        (p0.second-p1.second)*(p0.second-p1.second));
}

double CalcDis(Point p0,Point p1,Point p2)
{
        return Dis(p0,p1)+Dis(p0,p2)+Dis(p1,p2);
}

void build(int n,double w)
```

```
{
    g.clear();
    for (int i = 0;i < n;i++)
        g[make_pair((int)floor(p[i].first/w),(int)floor(p[i].second/w))].push_back(p[i]);
}

int main()
{
    int t;
    scanf(”%d”,&t);
    for (int ft = 1;ft <= t;ft++)
    {
        scanf(”%d”,&n);
        for (int i = 0;i < n;i++)
        {
            scanf(”%lf%lf”,&tx,&ty);
            p[i] = make_pair(tx,ty);
        }
        random_shuffle(p,p+n);
        ans = CalcDis(p[0],p[1],p[2]);
        build(3,ans/2.0);
        for (int i = 3;i < n;i++)
        {
            x = (int)floor(2.0*p[i].first/ans);
            y = (int)floor(2.0*p[i].second/ans);
            tmp.clear();
            for (int k = 0;k < 9;k++)
            {
                nx = x+step[k][0];
                ny = y+step[k][1];
                gird = make_pair(nx,ny);
                if (g.find(gird) != g.end())
                {
                    op = g[gird].begin();
                    ed = g[gird].end();
                    for (it = op;it != ed;it++)
                        tmp.push_back
(*it);
                }
            }
            flag = false;
            for (int j = 0;j < tmp.size();j++)
                for (int k = j+1;k < tmp.size();k++)
                {
                    nowans = CalcDis(p[i],tmp[j],tmp[k]);
                    if (nowans < ans)
                    {
                        ans = nowans;
                        flag = true;
                    }
                }
            if (flag == true)
                build(i+1,ans/2.0);
            else
                g[make_pair((int)floor(2.0*p[i].first/ans),(int)floor(2.0*p[i].second/ans))].push_back(p[i]);
        }
        printf("%.3f\n",ans);
    }
}
```

2.7 half-plane intersection.cpp

```
// a b c
inline pv ins(const pv &p1,const pv &p2)
{
    u=fabs(a*p1.x+b*p1.y+c);
    v=fabs(a*p2.x+b*p2.y+c);
    return pv((p1.x*v+p2.x*u)/(u+v),(p1.y*v+p2.y*u)/(u+v));
}

```

```
inline void get(const pv& p1,const pv& p2,double & a,double & b,double & c)
{
    a=p2.y-p1.y;
    b=p1.x-p2.x;
    c=p2.x*p1.y-p2.y*p1.x;
}

inline pv ins(const pv &x,const pv &y)
{
    get(x,y,d,e,f);
    return pv((b*f-c*e)/(a*e-b*d),(a*f-c*d)/(b*d-a*e));
}

std::vector<pv>p[2];
int main()
{
    k=0;
    p[k].resize(0);
    p[k].push_back(pv(-inf,inf));
    p[k].push_back(pv(-inf,-inf));
    p[k].push_back(pv(inf,-inf));
    p[k].push_back(pv(inf,inf));
    for(i=0;i<n;++i)
    {
        get(pnt[i],pnt[(i+1)%n],a,b,c);
        c+=the*sqrt(a*a+b*b);
        p[!k].resize(0);
        for(l=0;l<p[k].size();++l)
            if(a*p[k][l].x+b*p[k][l].y+c<eps)
                p[!k].push_back(p[k][l]);
            else
            {
                m=(l+p[k].size()-1)%p[k].size();
                if(a*p[k][m].x+b*p[k][m].y+c<-eps)
                    p[!k].push_back(ins(p[k][m],p[k][l]));
                m=(l+1)%p[k].size();
                if(a*p[k][m].x+b*p[k][m].y+c<-eps)
                    p[!k].push_back(ins(p[k][m],p[k][l]));
            }
        k=!k;
        if(p[k].empty())
            break;
    }
    //          p [k]
    return p[k].empty();
}

//
//

inline pv ins(const pv &a,const pv &b)
{
    u=fabs(ln.cross(a-pnt[i]));
    v=fabs(ln.cross(b-pnt[i]))+u;
    tl=b-a;
    return pv(u*tl.x/v+a.x,u*tl.y/v+a.y);
}

int main()
{
    j=0;
    for(i=0;i<n;++i)
    {
        ln=pnt[(i+1)%n]-pnt[i];
        p[!j].resize(0);
        for(k=0;k<p[j].size();++k)
            if(ln.cross(p[j][k]-pnt[i])<=0)
                p[!j].push_back(p[j][k]);
            else
            {
                l=(k-1+p[j].size())%p[j].size();
                if(ln.cross(p[j][l]-pnt[i])<0)
                    p[!j].push_back(ins(p[j][k],p[j][l]));
                l=(k+1)%p[j].size();
                if(ln.cross(p[j][l]-pnt[i])<0)
                    p[!j].push_back(ins(p[j][k],p[j][l]));
            }
        j=!j;
    }
    //          p [j]
}

//mrzy

bool HPIcmp(Line a, Line b)
{
    if (fabs(a.k - b.k) > eps)
```

```
        return a.k < b.k;
    return ((a.s - b.s) * (b.e-b.s)) < 0;
}

Line Q[100];

void HPI(Line line[], int n, Point res[], int &resn)
{
    int tot = n;
    std::sort(line, line + n, HPIcmp);
    tot = 1;
    for (int i = 1; i < n; i++)
        if (fabs(line[i].k - line[i - 1].k) > eps)
            line[tot++] = line[i];
    int head = 0, tail = 1;
    Q[0] = line[0];
    Q[1] = line[1];
    resn = 0;
    for (int i = 2; i < tot; i++)
    {
        if (fabs((Q[tail].e-Q[tail].s)*(Q[tail - 1].e-Q[tail - 1].s)) < eps || fabs((Q[head].e-Q[head].s)*(Q[head + 1].e-Q[head + 1].s)) < eps)
            return;
        while (head < tail && (((Q[tail]&Q[tail - 1]) - line[i].s) * (line[i].e-line[i].s)) > eps)
            --tail;
        while (head < tail && (((Q[head]&Q[head + 1]) - line[i].s) * (line[i].e-line[i].s)) > eps)
            ++head;
        Q[++tail] = line[i];
    }
    while (head < tail && (((Q[tail]&Q[tail - 1]) - Q[head].s) * (Q[head].e-Q[head].s)) > eps)
        tail--;
    while (head < tail && (((Q[head]&Q[head + 1]) - Q[tail].s) * (Q[tail].e-Q[tail].s)) > eps)
        head++;
    if (tail <= head + 1)
        return;
    for (int i = head; i < tail; i++)
        res[resn++] = Q[i] & Q[i + 1];
    if (head < tail + 1)
        res[resn++] = Q[head] & Q[tail];
}
```

2.8 kdtree.cpp

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <stack>
#include <algorithm>
using namespace std;
#define MAXN 100010
typedef long long ll;
struct Point{
    ll x,y;
    void operator =(const Point &p){
        x=p.x; y=p.y;
    }
    ll dis(const Point &a){
        return (x-a.x)*(x-a.x)+(y-a.y)*(y-a.y);
    }
}point[MAXN],pp[MAXN];

struct Node{
    int split;//{0,1}   0   x   1   y
    Point p;//
}tree[MAXN*4];

bool cmpx(const Point &a,const Point &b)
{
    return a.x<b.x;
}

bool cmpy(const Point &a,const Point &b)
{
    return a.y<b.y;
}

void initTree(int x,int y,int split,int pos)
{
    if(y<x) return ;
    int mid=(x+y)>>1;
    random_shuffle(point+x,point+y);
    if(split==0) nth_element(point+x,point+mid,point+y+1,cmpx);
    else nth_element(point+x,point+mid,point+y+1,cmpy);
    tree[pos].split=split;
    tree[pos].p=point[mid];
    initTree(x,mid-1,(split^1),2*pos);
    initTree(mid+1,y,(split^1),2*pos+1);
}

ll ans;
void insert(int x,int y,Point &p,int pos)
{
    if(y<x) return ;
    int mid=(x+y)>>1;
    ll temp=p.dis(tree[pos].p);
    if(temp!=0) ans=min(ans,temp);
    if(tree[pos].split==0){
        if(p.x<=tree[pos].p.x){
            insert(x,mid-1,p,2*pos);
            if(ans>=(p.x-tree[pos].p.x)*(p.x-tree[pos].p.x))
                insert(mid+1,y,p,2*pos+1);
        }
        else{
            insert(mid+1,y,p,2*pos+1);
            if(ans>=(p.x-tree[pos].p.x)*(p.x-tree[pos].p.x))
                insert(x,mid-1,p,2*pos);
        }
    }
    else
    {
        if(p.y<=tree[pos].p.y){
            insert(x,mid-1,p,2*pos);
            if(ans>=(p.y-tree[pos].p.y)*(p.y-tree[pos].p.y))
                insert(mid+1,y,p,2*pos+1);
        }
        else{
            insert(mid+1,y,p,2*pos+1);
            if(ans>=(p.y-tree[pos].p.y)*(p.y-tree[pos].p.y))
                insert(x,mid-1,p,2*pos);
        }
    }
}

int main()
```

```
{
    int cases,n;
    scanf("%d",&cases);
    while(cases--)
    {
        scanf("%d",&n);
        for(int i=1;i<=n;i++){
            scanf("%I64d%I64d",&pp[i].x,&pp[i].y);
            point[i]=pp[i];
        }
        initTree(1,n,0,1);
        for(int i=1;i<=n;i++){
            ans=1LL<<62;
            insert(1,n,pp[i],1);
            printf("%I64d\n",ans);
        }
    }
    return 0;
}
```

2.9 others

```
eps

    sqrt(a), asin(a), acos(a)
          a                                                        a
   -1e-12         sqrt(a)
          0                 a                                    a
   ,  asin(a)  acos(a)
                                                                a


                                                                 case
   :005,                     0 :01
                                 0 :005000000001(      )
                          0 :004999999999(      )
                   printf("%.2lf", a)

     a                  a + eps,            a - eps

                -0.000

     double

a==b  fabs(a-b)<eps
a!=b  fabs(a-b)>eps
```

```
a<b     a+eps<b
a<=b    a<b+eps
a>b     a>b+eps
a>=b    a+eps>b



cos/sin/tan
acos          [-1,+1]              [0,    ]
asin          [-1,+1]             [-    /2,+    /2]
atan          [-    /2,+    /2]
atan2           (y,x)(                    ),         tan(y/x),[-    ,+    ]
         x y

other

log                   (ln)
log10
ceil
floor

round

cpp:
java: add 0.5,then floor
cpp:
                                                4

                                                6

                            5                                            0

                            5                    5                                     0
```

2.10 Pick's theorem

```
A:
i:
b:
A = i + b/2 - 1



```

```
A = 2i + b - 2
```

2.11 PointInPoly.cpp

```
/*
          ,
 poly                              3

0 --         poly
1 --          poly
2 --         poly
*/

int inPoly(pv p,pv poly[], int n)
{
        int i, count;
        Line ray, side;

        count = 0;
        ray.s   = p;
        ray.e.y  = p.y;
        ray.e.x  = -1;  //-INF

        for (i = 0; i < n; i++)
        {
                side.s = poly[i];
                side.e = poly[(i+1)%n];

                if(OnSeg(p, side))
                        return 1;

                //         side        x
                if (side.s.y == side.e.y)
                        continue;

        if (OnSeg(side.s, ray))
        {
            if (side.s.y > side.e.y)
                count++;
        }
        else
            if (OnSeg(side.e, ray))
            {
                if (side.e.y > side.s.y)
                    count++;
            }
            else
                if (inter(ray, side))
                    count++;
        }
        return ((count % 2 == 1) ? 0 : 2);
}
```

2.12 rotating caliper.cpp

```
//

l=ans=0;
for(i=0;i<n;++i)
{
    tl=pnt[(i+1)%n]-pnt[i];
    while(abs(tl.cross(pnt[(l+1)%n]-pnt[i]))>abs(tl.cross(pnt[l]-pnt[i])))
        l=(l+1)%n;
    ans=std::max(ans,std::max(dist(pnt[l],pnt[i]),dist(pnt[l],pnt[(i+1)%n])));
}
return ans;

//
int main()
{
    sq=sp=0;
    for(i=1;i<ch[1].size();++i)
        if(ch[1][sq]<ch[1][i])
            sq=i;
    tp=sp;
    tq=sq;
    ans=(ch[0][sp]-ch[1][sq]).len();
    do
    {
        a1=ch[0][sp];
        a2=ch[0][(sp+1)%ch[0].size()];
        b1=ch[1][sq];
        b2=ch[1][(sq+1)%ch[1].size()];
        tpv=b1-(b2-a1);
        tpv.x = b1.x - (b2.x - a1.x);
        tpv.y = b1.y - (b2.y - a1.y);
        len=(tpv-a1).cross(a2-a1);
        if(fabs(len)<eps)
        {
                ans=std::min(ans,p2l(a1,b1,b2));
                ans=std::min(ans,p2l(a2,b1,b2));
                ans=std::min(ans,p2l(b1,a1,a2));
                ans=std::min(ans,p2l(b2,a1,a2));
                sp=(sp+1)%ch[0].size();
                sq=(sq+1)%ch[1].size();
            }
            else
                if(len<-eps)
                {
                    ans=std::min(ans,p2l(b1,a1,a2));
                    sp=(sp+1)%ch[0].size();
                }
                else
                {
                    ans=std::min(ans,p2l(a1,b1,b2));
                    sq=(sq+1)%ch[1].size();
                }
    }while(tp!=sp || tq!=sq);
    return ans;
}

//              by mzry
inline void solve()
{
    resa = resb = 1e100;
    double dis1,dis2;
    Point xp[4];
    Line l[4];
    int a,b,c,d;
    int sa,sb,sc,sd;
    a = b = c = d = 0;
    sa = sb = sc = sd = 0;
    Point va,vb,vc,vd;
    for (a = 0; a < n; a++)
    {
        va = Point(p[a],p[(a+1)%n]);
        vc = Point(-va.x,-va.y);
        vb = Point(-va.y,va.x);
        vd = Point(-vb.x,-vb.y);
        if (sb < sa)
        {
            b = a;
            sb = sa;
        }
        while (xmult(vb,Point(p[b],p[(b+1)%n])) < 0)
        {
            b = (b+1)%n;
            sb++;
        }
        if (sc < sb)
        {
            c = b;
            sc = sb;
        }
        while (xmult(vc,Point(p[c],p[(c+1)%n])) < 0)
        {
            c = (c+1)%n;
            sc++;
        }
        if (sd < sc)
        {
            d = c;
            sd = sc;
        }
        while (xmult(vd,Point(p[d],p[(d+1)%n])) < 0)
        {
            d = (d+1)%n;
            sd++;
        }

        //‘      p[a],p[b],p[c],p[d]     ‘
        sa++;
    }
}

//
                    P = { p(1) , ... , p(m) }     Q = { q
    (1) , ... , q(n) }               (p(i), q(j))
   P    Q

(p(i), q(j))
p(i-1), p(i+1), q(j-1), q(j+1)              (p(i), q(j))



1              P     Q              y

                                           x
```

```
2

    x
3

                                                  (p(i), q(j))


4                                                     (p(i), q(j))
                 p(i-1), p(i+1), q(j-1), q(j+1)
                                            (p(i), q(j))




5
                                        3                 4

6




                                       1      5      6                                    O
   (N)                                N


                                                                                 
                                                                                 



//
1                  P           y
   yminP                  Q           y
                                                   ymaxQ
2                                  yminP          ymaxQ
     LP          LQ

                  LP          LQ
   yminP          ymaxQ

3            p(i)= yminP    q(j)= ymaxQ    (p(i), q(j))

                                 p(i-1),p(i+1)             (p(i), q(j))
                                         q(j-1),q(j+1)
                                 (p(i), q(j))                          C S
4
```

```
5




6                             4              5
                                    (yminP ,ymaxQ)
7           C S

//        /                  /
1
     xminP    xmaxP    yminP    ymaxP
2                                  P

3



4


5                                        /



6                    4             5
                                                 90
7
```

2.13 sort - polar angle.cpp

```
inline bool cmp(const Point& a,const Point& b)
{
    if (a.y*b.y <= 0)
    {
        if (a.y > 0 || b.y > 0)
            return a.y < b.y;
        if (a.y == 0 && b.y == 0)
            return a.x < b.x;
    }
    return a.cross(b) > 0;
}
```

2.14 triangle's fermat point

1 120

2

3 120

4

5

6 ABC
' BCA' CAB'

7 CC' BB' AA'

Chapter 3

graph

3.1 2-sat.cpp

```
#define maxn 2008
struct Twosat
{
    int n;
    std::vector<int>G[maxn*2];
    bool mark[maxn*2];
    int s[maxn*2],c;

    bool dfs(int x)
    {
        if(mark[x^1])return false;
        if(mark[x])return true;
        mark[x]=true;
        s[c++]=x;
        for(int i=0;i<G[x].size();++i)
            if(!dfs(G[x][i]))return false;
        return true;
    }

    void init(int n)
    {
        this->n=n;
        for(int i=0;i<n*2;++i)
            G[i].clear();
        memset(mark,0,sizeof(mark));
    }
    void add_clause(int x,int xval,int y,int yval)//
    {
        x=x*2+xval;
        y=y*2+yval;
        G[x^1].push_back(y);
        G[y^1].push_back(x);
    }

    bool solve()
    {
        for(int i=0;i<n*2;i+=2)
            if(!mark[i]&&!mark[i+1])
            {
                c=0;
                if(!dfs(i))
                {
                    while(c>0)
                        mark[s[--c]]=false;
                    if(!dfs(i+1))
                        return false;
                }
            }
        return true;
    }
};
```

3.2 Articulation.cpp

```
void dfs(int now,int fa)  // now  1
{
    int p(0);
    dfn[now]=low[now]=cnt++;
    for(std::list<int>::const_iterator it(edge[now].begin());it!=edge[now].end();++it)
        if(dfn[*it]==-1)
        {
            dfs(*it,now);
            ++p;
            low[now]=std::min(low[now],low[*it]);
            if((now==1 && p>1) || (now!=1 && low[*it]>=dfn[now])) //
                ans.insert(now);
        }
        else
            if(*it!=fa)
                low[now]=std::min(low[now],dfn[*it]);
}
```

3.3 Augmenting Path Algorithm for Maximum Cardinality Bipartite Matching.cpp

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

#define MAXX 111

bool Map[MAXX][MAXX],visit[MAXX];
int link[MAXX],n,m;
bool dfs(int t)
{
    for (int i=0; i<m; i++)
        if (!visit[i] && Map[t][i]){
            visit[i] = true;
            if (link[i]==−1 || dfs(link[i])){
                link[i] = t;
                return true;
            }
        }
    return false;
}
int main()
{
    int k,a,b,c;
    while (scanf(”%d”,&n),n){
        memset(Map,false,sizeof(Map));
        scanf(”%d%d”,&m,&k);
        while (k−−){
            scanf(”%d%d%d”,&a,&b,&c);
            if (b && c)
                Map[b][c] = true;
        }
        memset(link,−1,sizeof(link));
        int ans = 0;
        for (int i=0; i<n; i++){
            memset(visit,false,sizeof(visit));
            if (dfs(i))
                ans++;
        }
        printf("%d\n",ans);
    }
}
```

3.4 best spanning tree.cpp

```
#include<cstdio>
#include<cstring>
#include<cmath>

#define MAXX 1111

struct
{
    int x,y;
    double z;
} node[MAXX];

struct
{
    double l,c;
} map[MAXX][MAXX];

int n,l,f[MAXX],pre[MAXX];
double dis[MAXX];

double mst(double x)
{
    int i,j,tmp;
    double min,s=0,t=0;
    memset(f,0,sizeof(f));
    f[1]=1;
    for (i=2; i<=n; i++)
    {
        dis[i]=map[1][i].c-map[1][i].l*x;
        pre[i]=1;
    }
    for (i=1; i<n; i++)
    {
        min=1e10;
        for (j=1; j<=n; j++)
            if (!f[j] && min>dis[j])
            {
                min=dis[j];
                tmp=j;
            }
        f[tmp]=1;
        t+=map[pre[tmp]][tmp].l;
        s+=map[pre[tmp]][tmp].c;
        for (j=1; j<=n; j++)
            if (!f[j] && map[tmp][j].c-map[tmp][j].l*x<dis[j])
            {
                dis[j]=map[tmp][j].c-map[tmp][j].l*x;
                pre[j]=tmp;
            }
    }
    return s/t;
}

int main()
{
    int i,j;
    double a,b;
    while (scanf("%d",&n),n);
    {
        for (i=1; i<=n; i++)
            scanf("%d%d%lf",&node[i].x,&node[i].y,&node[i].z);
        for (i=1; i<=n; i++)
            for (j=i+1; j<=n; j++)
            {
                map[j][i].l=map[i][j].l=sqrt(1.0*(node[i].x-node[j].x)*(node[i].x-node[j].x)+(node[i].y-node[j].y)*(node[i].y-node[j].y));
                map[j][i].c=map[i][j].c=fabs(node[i].z-node[j].z);
            }
        a=0,b=mst(a);
        while (fabs(b-a)>1e-8)
        {
            a=b;
            b=mst(a);
        }
        printf("%.3lf\n",b);
    }
    return 0;

}
```

3.5 Biconnected Component.cpp

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

const int MAXN=100000*2;
const int MAXM=200000;

//0-based

struct edges
{
    int to,next;
    bool cut,visit;
} edge[MAXM<<1];

int head[MAXN],low[MAXN],dpt[MAXN],L;
bool visit[MAXN],cut[MAXN];
int idx;
std::stack<int> st;
int bcc[MAXM];

void init(int n)
{
    L=0;
    memset(head,-1,4*n);
    memset(visit,0,n);
}

void add_edge(int u,int v)
{
    edge[L].cut=edge[L].visit=false;
    edge[L].to=v;
    edge[L].next=head[u];
    head[u]=L++;
}

void dfs(int u,int fu,int deg)
{
    cut[u]=false;
    visit[u]=true;
    low[u]=dpt[u]=deg;
    int tot=0;
    for (int i=head[u]; i!=-1; i=edge[i].next)
    {
        int v=edge[i].to;
        if (edge[i].visit)
            continue;
        st.push(i/2);
        edge[i].visit=edge[i^1].visit=true;
        if (visit[v])
        {
            low[u]=dpt[v]>low[u]?low[u]:dpt[v];
            continue;
        }
        dfs(v,u,deg+1);
        edge[i].cut=edge[i^1].cut=(low[v]>dpt[u] || edge[i].cut);
        if (u!=fu) cut[u]=low[v]>=dpt[u]?1:cut[u];
        if (low[v]>=dpt[u] || u==fu)
        {
            while (st.top()!=i/2)
            {
                int x=st.top()*2,y=st.top()*2+1;
                bcc[st.top()]=idx;
                st.pop();
            }
            bcc[i/2]=idx++;
            st.pop();
        }
        low[u]=low[v]>low[u]?low[u]:low[v];
        tot++;
    }
    if (u==fu && tot>1)
        cut[u]=true;
}

int main()
{
    int n,m;
    while (scanf("%d%d",&n,&m)!=EOF)
    {
        init(n);
        for (int i=0; i<m; i++)
        {
            int u,v;
            scanf("%d%d",&u,&v);
            add_edge(u,v);
            add_edge(v,u);
        }
        idx=0;
        for (int i=0; i<n; i++)
            if (!visit[i])
                dfs(i,i,0);
    }
    return 0;
}
```

3.6 Bridge.cpp

```
void dfs(const short &now,const short &fa)
{
    dfn[now]=low[now]=cnt++;
    for(int i(0);i<edge[now].size();++i)
        if(dfn[edge[now][i]]==-1)
        {
            dfs(edge[now][i],now);
            low[now]=std::min(low[now],low[edge[now][i]])
                ;
            if(low[edge[now][i]]>dfn[now]) //

                ,
            {
                if(edge[now][i]<now)
                {
                    j=edge[now][i];
                    k=now;
                }
                else
                {
                    j=now;
                    k=edge[now][i];
                }
                ans.push_back(node(j,k));
            }
        }
        else
            if(edge[now][i]!=fa)
                low[now]=std::min(low[now],low[edge[now][
                    i]]);
}
```

3.7 chu-liu algorithm.cpp

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

const int inf = 0x5fffffff;

int n,m,u,v,cost,dis[1001][1001],L;
int pre[1001],id[1001],visit[1001],in[1001];

void init(int n)
{
    L = 0;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            dis[i][j] = inf;
}

struct Edge
{
    int u,v,cost;
};

Edge e[1001*1001];


int zhuliu(int root,int n,int m,Edge e[])
{
    int res = 0,u,v;
    while (true)
    {
        for (int i = 0; i < n; i++)
            in[i] = inf;
        for (int i = 0; i < m; i++)
            if (e[i].u != e[i].v && e[i].cost < in[e[i].v])
            {
                pre[e[i].v] = e[i].u;
                in[e[i].v] = e[i].cost;
            }
        for (int i = 0; i < n; i++)
            if (i != root)
                if (in[i] == inf)
                    return -1;
        int tn = 0;
        memset(id,-1,sizeof(id));
        memset(visit,-1,sizeof(visit));
        in[root] = 0;
        for (int i = 0; i < n; i++)
        {
            res += in[i];
            v = i;
            while (visit[v] != i && id[v] == -1 && v != root)
            {
                visit[v] = i;
                v = pre[v];
            }
            if(v != root && id[v] == -1)
            {
                for(int u = pre[v] ; u != v ; u = pre[u])
                    id[u] = tn;
                id[v] = tn++;
            }
        }
        if(tn == 0) break;
        for (int i = 0; i < n; i++)
            if (id[i] == -1)
                id[i] = tn++;
        for (int i = 0; i < m;)
        {
            int v = e[i].v;
            e[i].u = id[e[i].u];
            e[i].v = id[e[i].v];
            if (e[i].u != e[i].v)
                e[i++].cost -= in[v];
            else
                std::swap(e[i],e[--m]);
        }
        n = tn;
        root = id[root];
    }
    return res;
}

int main()
{
    freopen("asdf","r",stdin);
    while (scanf("%d%d",&n,&m) != EOF)
    {
        init(n);
        for (int i = 0; i < m; i++)
        {
            scanf("%d%d%d",&u,&v,&cost);
            if (u == v) continue;
            dis[u][v] = std::min(dis[u][v],cost);
        }
        L = 0;
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                if (dis[i][j] != inf)
                {
                    e[L].u = i;
                    e[L].v = j;
                    e[L++].cost = dis[i][j];
                }
        printf("%d\n",zhuliu(0,n,L,e));
    }
    return 0;
}
```

3.8 k-th shortest path.cpp

```
#include<cstdio>
#include<cstring>
#include<queue>
#include<vector>

int K;

class states
{
    public:
        int cost,id;
};

int dist[1000];

class cmp
{
    public:
        bool operator ()(const states &i,const states &j)
        {
            return i.cost>j.cost;
        }
};

class cmp2
{
    public:
        bool operator()(const states &i,const states &j)
        {
            return i.cost+dist[i.id]>j.cost+dist[j.id];
        }
};

struct edges
{
    int to,next,cost;
} edger[100000],edge[100000];

int headr[1000],head[1000],Lr,L;

void dijkstra(int s)
{
    states u;
    u.id=s;
    u.cost=0;
    dist[s]=0;
    std::priority_queue<states,std::vector<states>,cmp> q;
    q.push(u);
    while (!q.empty())
    {
        u=q.top();
        q.pop();
        if (u.cost!=dist[u.id])
            continue;
        for (int i=headr[u.id]; i!=-1; i=edger[i].next)
        {
            states v=u;
            v.id=edger[i].to;
            if (dist[v.id]>dist[u.id]+edger[i].cost)
            {
                v.cost=dist[v.id]=dist[u.id]+edger[i].cost;
                q.push(v);
            }
        }
    }
}

int num[1000];

inline void init(int n)
```

```
{
    Lr=L=0;
    memset(head,-1,4*n);
    memset(headr,-1,4*n);
    memset(dist,63,4*n);
    memset(num,0,4*n);
}

void add_edge(int u,int v,int x)
{
    edge[L].to=v;
    edge[L].cost=x;
    edge[L].next=head[u];
    head[u]=L++;
    edger[Lr].to=u;
    edger[Lr].cost=x;
    edger[Lr].next=headr[v];
    headr[v]=Lr++;
}

inline int a_star(int s,int t)
{
    if (dist[s]==0x3f3f3f3f)
        return -1;
    std::priority_queue<states,std::vector<states>,cmp2>
        q;
    states tmp;
    tmp.id=s;
    tmp.cost=0;
    q.push(tmp);
    while (!q.empty())
    {
        states u=q.top();
        q.pop();
        num[u.id]++;
        if (num[t]==K)
            return u.cost;
        for (int i=head[u.id]; i!=-1; i=edge[i].next)
        {
            int v=edge[i].to;
            tmp.id=v;
            tmp.cost=u.cost+edge[i].cost;
            q.push(tmp);
        }
    }
    return -1;
}

int main()
{
    int n,m;
    scanf(”%d%d”,&n,&m);
    init(n);
    for (int i=0; i<m; i++)
    {
        int u,v,x;
        scanf(”%d%d%d”,&u,&v,&x);
        add_edge(u-1,v-1,x);
    }
    int s,t;
    scanf(”%d%d%d”,&s,&t,&K);
    if (s==t)
        ++K;
    dijkstra(t-1);
    printf(”%d\n”, a_star(s-1,t-1));
}
```

3.9 Kuhn-Munkres algorithm.cpp

```
bool match(int u)//
{
    vx[u]=true;
    for(int i=1;i<=n;++i)
        if(lx[u]+ly[i]==g[u][i]&&!vy[i])
        {
            vy[i]=true;
            if(!d[i]||match(d[i]))
            {
                d[i]=u;
                return true;
            }
        }
    return false;
}
inline void update()//
{
    int i,j;
    int a=1<<30;
    for(i=1;i<=n;++i)if(vx[i])
        for(j=1;j<=n;++j)if(!vy[j])
            a=min(a,lx[i]+ly[j]-g[i][j]);
    for(i=1;i<=n;++i)
    {
        if(vx[i]) lx[i]-=a;
        if(vy[i]) ly[i]+=a;
    }
}
void km()
{
    int i,j;
    for(i=1;i<=n;++i)
    {
        lx[i]=ly[i]=d[i]=0;
        for(j=1;j<=n;++j)
            lx[i]=max(lx[i],g[i][j]);
    }
    for(i=1;i<=n;++i)
    {
        while(true)
        {
            memset(vx,0,sizeof(vx));
            memset(vy,0,sizeof(vy));
            if(match(i))
                break;
            update();
        }
    }
    int ans=0;
    for(i=1;i<=n;++i)
        if(d[i]!=0)
            ans+=g[d[i]][i];
    printf("%d\n",ans);
}
int main()
{
    while(scanf("%d\n",&n)!=EOF)
    {
        for(int i=1;i<=n;++i)gets(s[i]);
        memset(g,0,sizeof(g));
        for(int i=1;i<=n;++i)
            for(int j=1;j<=n;++j)
                if(i!=j) g[i][j]=cal(s[i],s[j]);
        km();
    }
    return 0;
}
```

```
//bupt

// k m                                        n^3
int dfs(int u)//
{
    int v;
    sx[u]=1;
    for ( v=1; v<=n; v++)
        if (!sy[v] && lx[u]+ly[v]==map[u][v])
        {
            sy[v]=1;
            if (match[v]==-1 || dfs(match[v]))
            {
                match[v]=u;
                return 1;
            }
        }
    return 0;
}

int bestmatch(void)// k m
{
    int i,j,u;
    for (i=1; i<=n; i++)//
    {
        lx[i]=-1;
        ly[i]=0;
        for (j=1; j<=n; j++)
            if (lx[i]<map[i][j])
                lx[i]=map[i][j];
    }
    memset(match, -1, sizeof(match));
    for (u=1; u<=n; u++)
    {
        while (true)
        {
            memset(sx,0,sizeof(sx));
            memset(sy,0,sizeof(sy));
            if (dfs(u))
                break;
            int dx=Inf;//
                                                                  ˜˜
            for (i=1; i<=n; i++)
            {
                if (sx[i])
                    for (j=1; j<=n; j++)
```

```
                        if(!sy[j] && dx>lx[i]+ly[j]-map[i][j])
                            dx=lx[i]+ly[j]-map[i][j];
            }
            for (i=1; i<=n; i++)
            {
                if (sx[i])
                    lx[i]-=dx;
                if (sy[i])
                    ly[i]+=dx;
            }
        }
    }
    int sum=0;
    for (i=1; i<=n; i++)
        sum+=map[match[i]][i];
    return sum;
}
```

3.10 LCA - DA.cpp

```
int edge[MAXX],nxt[MAXX<<1],to[MAXX<<1],cnt;
int pre[MAXX][N],dg[MAXX];

inline void add(int j,int k)
{
    nxt[++cnt]=edge[j];
    edge[j]=cnt;
    to[cnt]=k;
}

void rr(int now,int fa)
{
    dg[now]=dg[fa]+1;
    for(int i(edge[now]);i;i=nxt[i])
        if(to[i]!=fa)
        {
            static int j;
            j=1;
            for(pre[to[i]][0]=now;j<N;++j)
                pre[to[i]][j]=pre[pre[to[i]][j-1]][j-1];
            rr(to[i],now);
        }
}

inline int lca(int a,int b)
```

```
{
    static int i,j;
    j=0;
    if(dg[a]<dg[b])
        std::swap(a,b);
    for(i=dg[a]-dg[b];i;i>>=1,++j)
        if(i&1)
            a=pre[a][j];
    if(a==b)
        return a;
    for(i=N-1;i>=0;--i)
        if(pre[a][i]!=pre[b][i])
        {
            a=pre[a][i];
            b=pre[b][i];
        }
    return pre[a][0];
}
```

3.11 LCA - tarjan - minmax.cpp

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

#define MAXX 100111
#define inf 0x5fffffff

short T,t;
int set[MAXX],min[MAXX],max[MAXX],ans[2][MAXX];
bool done[MAXX];
std::list<std::pair<int,int> >edge[MAXX];
std::list<std::pair<int,int> >q[MAXX];
int n,i,j,k,l,m;

struct node
{
    int a,b,id;
    node() {}
    node(const int &aa,const int &bb,const int &idd): a(aa),b(bb),id(idd){}
};

std::list<node>to[MAXX];

```

```
int find(const int &a)
{
    if(set[a]==a)
        return a;
    int b(set[a]);
    set[a]=find(set[a]);
    max[a]=std::max(max[a],max[b]);
    min[a]=std::min(min[a],min[b]);
    return set[a];
}

void tarjan(const int &now)
{
    done[now]=true;
    for(std::list<std::pair<int,int> >::const_iterator it(q[now].begin());it!=q[now].end();++it)
        if(done[it->first])
            if(it->second>0)
                to[find(it->first)].push_back(node(now,it->first,it->second));
            else
                to[find(it->first)].push_back(node(it->first,now,-it->second));
    for(std::list<std::pair<int,int> >::const_iterator it(edge[now].begin());it!=edge[now].end();++it)
        if(!done[it->first])
        {
            tarjan(it->first);
            set[it->first]=now;
            min[it->first]=it->second;
            max[it->first]=it->second;
        }
    for(std::list<node>::const_iterator it(to[now].begin());it!=to[now].end();++it)
    {
        find(it->a);
        find(it->b);
        ans[0][it->id]=std::min(min[it->b],min[it->a]);
        ans[1][it->id]=std::max(max[it->a],max[it->b]);
    }
}

int main()
{
    scanf("%hd",&T);
    for(t=1;t<=T;++t)
    {
        scanf("%d",&n);
        for(i=1;i<=n;++i)
        {
            edge[i].clear();
            q[i].clear();
            to[i].clear();
            done[i]=false;
            set[i]=i;
            min[i]=inf;
            max[i]=0;
        }
        for(i=1;i<n;++i)
        {
            scanf("%d%d%d",&j,&k,&l);
            edge[j].push_back(std::make_pair(k,l));
            edge[k].push_back(std::make_pair(j,l));
        }
        scanf("%d",&m);
        for(i=0;i<m;++i)
        {
            scanf("%d %d",&j,&k);
            q[j].push_back(std::make_pair(k,i));
            q[k].push_back(std::make_pair(j,-i));
        }
        tarjan(1);
        printf("Case %hd:\n",t);
        for(i=0;i<m;++i)
            printf("%d %d\n",ans[0][i],ans[1][i]);
    }
    return 0;
}
```

3.12 Minimum Cost Maximum Flow.cpp

```
struct mcmf
{
    struct Edge
    {
        int from,to,cap,flow,cost;
    };
    int n,m,s,t;
    std::vector<Edge>edges;
    std::vector<int>G[maxn];
    int inq[maxn],d[maxn],p[maxn],a[maxn];

    void init(int n)
    {
        this->n=n;
        for(int i=0;i<n;++i)
            G[i].clear();
        edges.clear();
    }

    void addedge(int from,int to,int cap,int cost)
    {
        Edge x={from,to,cap,0,cost};
        edges.push_back(x);
        Edge y={to,from,0,0,-cost};
        edges.push_back(y);
        m=edges.size();
        G[from].push_back(m-2);
        G[to].push_back(m-1);
    }
    int mincost(int s,int t)
    {
        int flow=0,cost=0;
        while(BellmanFord(s,t,flow,cost));
        if(flow!=(n-1)/2)return -1;
        return cost;
    }
private:
    bool BellmanFord(int s,int t,int& flow,int& cost)
    {
        for(int i=0;i<=n;++i)
            d[i]=INF;
        memset(inq,0,sizeof(inq));
        d[s]=0; inq[s]=1; p[s]=0; a[s]=INF;
        std::queue<int>Q;
        Q.push(s);
        while(!Q.empty())
        {
            int u=Q.front();
            Q.pop();
            inq[u]=0;
            for(int i=0;i<G[u].size();++i)
            {
                Edge& e=edges[G[u][i]];
                if(e.cap>e.flow && d[e.to]>d[u]+e.cost)
                {
                    d[e.to]=d[u]+e.cost;
                    p[e.to]=G[u][i];
                    a[e.to]=min(a[u],e.cap-e.flow);
                    if(!inq[e.to])
                    {
                        Q.push(e.to);
                        inq[e.to]=1;
                    }
                }
            }
        }
        if(d[t]==INF)
            return false;
        flow+=a[t];
        cost+=d[t]*a[t];
        int u=t;
        while(u!=s)
        {
            edges[p[u]].flow+=a[t];
            edges[p[u]^1].flow-=a[t];
            u=edges[p[u]].from;
        }
        return true;
    }

}G;
```

3.13 minimum cut.cpp

```
#include <iostream>
using namespace std;
const int maxn=510;
int map[maxn][maxn];
int n;
void contract(int x,int y)//
{
   int i,j;
    for (i=0; i<n; i++)
    if (i!=x) map[x][i]+=map[y][i],map[i][x]+=map[i][y];
    for (i=y+1; i<n; i++) for (j=0; j<n; j++)
    {
        map[i-1][j]=map[i][j];
        map[j][i-1]=map[j][i];
    }
    n--;
}
int w[maxn],c[maxn];
int sx,tx;
int mincut()
//


{
    int i,j,k,t;
    memset(c,0,sizeof(c));
    c[0]=1;
    for (i=0; i<n; i++) w[i]=map[0][i];
    for (i=1; i+1<n; i++)
    {
        t=k=-1;
        for (j=0; j<n; j++) if (c[j]==0&&w[j]>k)
        k=w[t=j];
        c[sx=t]=1;
        for (j=0; j<n; j++) w[j]+=map[t][j];
    }
    for (i=0; i<n; i++) if (c[i]==0) return w[tx=i];
}
int main()
{
    int i,j,k,m;
    while (scanf("%d%d",&n,&m)!=EOF)
    {
        memset(map,0,sizeof(map));
        while (m--)
        {
            scanf("%d%d%d",&i,&j,&k);
            map[i][j]+=k;
            map[j][i]+=k;
        }
        int mint=999999999;
        while (n>1)
        {
            k=mincut();
            if (k<mint) mint=k;
            contract(sx,tx);
        }
    printf("%d\n",mint);
    }
return 0;
}
```

3.14 others

```
                                                        ,
                                         .

 ,                                                                  .



 ,
             s                    s                    s





         G                 D              D
 , v                 u   ,
  v                      G          =
                                          G  = (V;E)
           U      V                  u  , v      U
    < u;  v >
           E         U   G             G            U    G
 ;  v        U   < u;  v >
             E         U   G            G           U    G




                 +              = V
        =
                 =
```

3.15 Shortest Augmenting Path algorithm.cpp

```
#include <cstring>
#include <cstdio>
#include <vector>
#include <queue>
#define maxn 1005
#define INF 1<<30
using namespace std;
struct Edge
```

```
{
        int from,to,cap,flow;
};
vector<Edge>edges;
vector<int>G[maxn];
int num[maxn],p[maxn],n,m;
int st[maxn],et[maxn],nt[maxn];
int d[maxn],s,t,cur[maxn];
void addedge(int from,int to,int cap)
{
    struct Edge x={from,to,cap,0};
        edges.push_back(x);
        struct Edge y={to,from,0,0};
        edges.push_back(y);
        int m=edges.size();
        G[from].push_back(m-2);
        G[to].push_back(m-1);
}
void bfs()//
         BFS
{
        queue<int>q;
        memset(d,0,sizeof(d));
        d[t]=1;
        q.push(t);
        while(!q.empty())
        {
                int u=q.front(); q.pop();
                for(int i=0;i<G[u].size();++i)
                if(G[u][i]&1)
                {
                        Edge& e=edges[G[u][i]];
                        if(!d[e.to])
                        {
                                d[e.to]=d[u]+1;
                                q.push(e.to);
                        }
                }
        }
}
int augment()//
{
        int u=t, a=INF;
        while(u!=s)
        {
                Edge& e=edges[p[u]];
        a=min(a,e.cap-e.flow);//
        u=e.from;
    }
    u=t;
    while(u!=s)
    {
        edges[p[u]].flow+=a;
        edges[p[u]^1].flow-=a;
        u=edges[p[u]].from;
    }
    return a;
}
int sap()
{
    int flow=0;
    bfs();
    memset(num,0,sizeof(num));
    for(int i=0;i<=t;++i)num[d[i]]++;
    int u=s;
    memset(cur,0,sizeof(cur));
    while(d[s]<t)
    {
        if(u==t)
        {
            flow+=augment();
            u=s;
        }
        int ok=0;
        for(int i=cur[u];i<G[u].size();++i)
        {
            Edge& e=edges[G[u][i]];
            if(e.cap>e.flow && d[u]==d[e.to]+1)
            {
                ok=1;
                p[e.to]=G[u][i];
                cur[u]=i;
                u=e.to;
                break;
            }
        }
        if(!ok)//
```

```
                {
                        int m=t-1;
                        for(int i=0;i<G[u].size();++i)
                        {
                                Edge& e=edges[G[u][i]];
                                if(e.cap>e.flow) m=min(m,d[e.to]);
                        }
                        if(--num[d[u]]==0)break;// gap
                        num[d[u]=m+1]++;
                        cur[u]=0;//
                        if(u!=s)u=edges[p[u]].from;
                }
        }
        return flow;
}
void init()
{
 edges.clear();
 for(int i=0;i<maxn;++i)G[i].clear();
}
int main()
{
    int T, i;
  //        freopen("1.txt","r",stdin);
    scanf("%d", &T);
    while (T--)
    {
        scanf("%d%d", &n, &m);
        s=0;  t=-1;
        int res=0;
        init();
        for(i=1;i<=n;++i)
        {
            scanf("%d%d%d",&st[i],&et[i],&nt[i]);
            res+=nt[i];
            if(et[i]+n+1>t)t=et[i]+n+1;
        }
        for (i=1; i<=n; ++i)
        {
            int j;
            addedge(s,i,nt[i]);
            for(j=st[i];j<=et[i];++j) addedge(i,j+n,1);
        }
        for(i=n+1;i<t;++i) addedge(i,t,m);
        int ans = sap();
        if(ans==res) printf("Yaha, Garfield, You Finish it!\n");
        else printf("Oh~~Garfield, You are just as before!\n");
    }
        return 0;
}
```

3.16 Stable Marriage.cpp

```
//


while(!g.empty())   //
{
    if(dfn[edge[g.front()].front()]==-1)
        dfn[edge[g.front()].front()]=g.front();  //


    else
    {
        for(it=edge[edge[g.front()].front()].begin();it!=edge[edge[g.front()].front()].end();++it)
            if(*it==dfn[edge[g.front()].front()] || *it==g.front())  //


                break;
        if(*it==g.front())  //
        {
            g.push_back(dfn[edge[g.front()].front()]);
            dfn[edge[g.front()].front()]=g.front();
        }
        else
            g.push_back(g.front());  //

    }
    edge[g.front()].pop_front();  //

    g.pop_front();
}
```

3.17 Strongly Connected Component.cpp

```
void dfs(const short &now)
{
    dfn[now]=low[now]=cnt++;
    st.push(now);
    for(std::list<short>::const_iterator it(edge[now].begin());it!=edge[now].end();++it)
        if(dfn[*it]==-1)
        {
            dfs(*it);
            low[now]=std::min(low[now],low[*it]);
        }
        else
            if(sc[*it]==-1)
                low[now]=std::min(low[now],dfn[*it]);
    if(dfn[now]==low[now])
    {
        while(sc[now]==-1)
        {
            sc[st.top()]=p;
            st.pop();
        }
        ++p;
    }
}
```

Chapter 4

math

4.1 cantor.cpp

```
const int PermSize = 12;
int fac[PermSize] = {1, 1, 2, 6, 24, 120, 720, 5040,
    40320, 362880, 3628800, 39916800};

inline int Cantor(int a[])
{
    int i, j, cnt;
    int res = 0;
    for (i = 0; i < PermSize; ++i)
    {
        cnt = 0;
        for (j = i + 1; j < PermSize; ++j)
            if (a[i] > a[j])
                ++cnt;
        res = res + cnt * fac[PermSize - i - 1];
    }
    return res;
}

bool h[13];

inline void UnCantor(int x, int res[])
{
    int i,j,l,t;
    for (i = 1;i <= 12;i++)
        h[i] = false;
    for (i = 1; i <= 12; i++)
    {
        t = x / fac[12 - i];
        x -= t * fac[12 - i];
        for (j = 1, l = 0; l <= t; j++)
            if (!h[j])
                l++;
        j--;
        h[j] = true;
        res[i - 1] = j;
    }
}
```

4.2 combinations.cpp

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

int mod;
long long num[100000];
int ni[100],mi[100];
int len;

void init(int p)
{
    mod=p;
    num[0]=1;
    for (int i=1; i<p; i++)
        num[i]=i*num[i-1]%p;
}

void get(int n,int ni[],int p)
{
    for (int i = 0; i < 100; i++)
        ni[i] = 0;
    int tlen = 0;
    while (n != 0)
    {
        ni[tlen++] = n%p;
        n /= p;
    }
    len = tlen;
}

long long power(long long x,long long y)
{
    long long ret=1;
    for (long long a=x%mod; y; y>>=1,a=a*a%mod)
        if (y&1)
            ret=ret*a%mod;
    return ret;
}

long long getInv(long long x)//‘mod          ‘
{
    return power(x,mod-2);
}

long long calc(int n,int m,int p)//C(n,m)%p
{
    init(p);
    long long ans=1;
    for (; n && m && ans; n/=p,m/=p)
    {
        if (n%p>=m%p)
            ans = ans*num[n%p]%p *getInv(num[m%p]%p)%p *
                getInv(num[n%p-m%p])%p;
        else
            ans=0;
    }
    return ans;
}

int main()
{
    int t;
    scanf("%d",&t);
    while (t--)
    {
        int n,m,p;
        scanf("%d%d%d",&n,&m,&p);
        printf("%lld\n",calc(n+m,m,p));
    }
    return 0;
}
```

4.3 euler's totient function.cpp

```
                n                                           n                 n
   (x)=x(1-1/p1)(1-1/p2)(1-1/p3)(1-1/p4)     ..(1-1/pn),
        p1  ,
    p2         pn     x                           x           0
   (1)=1              1                  1                    m   ,
    n            (mn)=   (m)   (n)

int Euler(int n)
{
    int ans = n;
    for (int i=2; i<=sqrt(n); i++)
    {
        if (n%i==0)
        {
            ans = ans-ans/i;
            while (n%i==0)
                n /= i;
        }
    }
    if (n>1)
        ans = ans-ans/n;
    return ans;
}

//
inline void Euler2()
{
    memset(euler,0,sizeof(euler));
    euler[1] = 1;
    for (int i = 2; i <= 3000000; i++)
    {
        if (!euler[i])
        {
            for (int j = i; j <= 3000000; j += i)
            {
                if (!euler[j])
                    euler[j] = j;
                euler[j] = euler[j]/i*(i-1);
            }
        }
    }
}
```

4.4 extended euclidean algorithm.cpp

```
//      ax+by=gcd(a,b)
long long ex_gcd(long long a,long long b,long long &x,
    long long &y)
{
    if (b)
    {
        long long ret = ex_gcd(b,a%b,x,y),tmp = x;
        x = y;
        y = tmp-(a/b)*y;
        return ret;
    }
    else
    {
        x = 1;
        y = 0;
        return a;
    }
}
```

4.5 inverse element.cpp

```
inline void getInv2(int x,int mod)
{
    inv[1]=1;
    for (int i=2; i<=x; i++)
        inv[i]=(mod-(mod/i)*inv[mod%i]%mod)%mod;
}

long long power(long long x,long long y,int mod)
{
    long long ret=1;
    for (long long a=x%mod; y; y>>=1,a=a*a%mod)
        if (y&1)
            ret=ret*a%mod;
    return ret;
}

inline int getInv(int x,int mod)//mod
{
    return power(x,mod-2);
}
```

4.6 lucas.cpp

```
#include <cstdio>
/*
Lucas            C (n,m)%p
*/
void gcd(int n,int k,int &x,int &y)
{
      if(k==0){x=1;y=0;return ;}
```

```
        else
        {
                gcd(k,n%k,x,y);
                int t=x;x=y;
                y=t-(n/k)*y;
                return;
        }
}

int CmodP(int n,int k,int p)
{
        if(k>n) return 0;
        int a,b,flag=0,x,y;
        a=b=1;
        for(int i=1;i<=k;i++)
        {
                x=n-i+1,y=i;
                while(x%p==0) x/=p,flag++;
                while(y%p==0) y/=p,flag--;
                x%=p,y%=p,a*=x,b*=y;
                b%=p,a%=p;
        }
        if(flag) return 0;
        gcd(b,p,x,y);
        if(x<0) x+=p;
        a*=x,a%=p;
        return a;
}

// Lucas                    C(n,m) % p  ,p
long long Lucas(long long n, long long m, long long p)
{
      long long ans=1;
      long long a,b;
      while(m&&n&&ans)
      {
            ans*=(CmodP(n%p,m%p,p));
            ans=ans%p;
            n=n/p;
            m=m/p;
      }
      return ans;
}
int main()
{
    long long n,k,p,ans;
    int cas=0;
//      freopen("1.txt","r",stdin);
//      freopen("out2.txt","w",stdout);
    while(scanf("%I64d%I64d%I64d",&n,&k,&p)!=EOF)
    {
        ++cas;
        if(k>n-k)k=n-k;
        ans=Lucas(n+1,k,p)+n-k;
        printf("Case␣#%d:␣%I64d\n",cas,ans%p);
    }
    return 0;
}
```

4.7 matrix.cpp

```
//
struct Matrix
{
    int a[52][52];
    Matrix operator * (const Matrix &b)const
    {
        Matrix res;
        for (int i = 0; i < 52; i++)
            for (int j = 0; j < 52; j++)
            {
                res.a[i][j] = 0;
                for (int k = 0; k < 52; k++)
                    res.a[i][j] += a[i][k] * b.a[k][j];
            }
        return res;
    }
    Matrix operator ^ (int y)const
    {
        Matrix res, x;
        for (int i = 0; i < 52; i++)
        {
            for (int j = 0; j < 52; j++)
                res.a[i][j] = 0, x.a[i][j] = a[i][j];
            res.a[i][i] = 1;
        }
        for (; y; y >>= 1, x = x * x)
            if (y & 1)
                res = res * x;
        return res;
    }
};
```

4.8 miller rabin.cpp

```cpp
inline unsigned long long multi_mod(const unsigned long long &a,unsigned long long b,const unsigned long long &n)
{
    unsigned long long exp(a%n),tmp(0);
    while(b)
    {
        if(b&1)
        {
            tmp+=exp;
            if(tmp>n)
                tmp-=n;
        }
        exp<<=1;
        if(exp>n)
            exp-=n;
        b>>=1;
    }
    return tmp;
}

inline unsigned long long exp_mod(unsigned long long a,unsigned long long b,const unsigned long long &c)
{
    unsigned long long tmp(1);
    while(b)
    {
        if(b&1)
            tmp=multi_mod(tmp,a,c);
        a=multi_mod(a,a,c);
        b>>=1;
    }
    return tmp;
}

inline bool miller_rabbin(const unsigned long long &n,short T)
{
    if(n==2)
        return true;
    if(n<2 || !(n&1))
        return false;
    unsigned long long a,u(n-1),x,y;
    short t(0),i;
    while(!(u&1))
    {
        ++t;
        u>>=1;
    }
    while(T--)
    {
        a=rand()%(n-1)+1;
        x=exp_mod(a,u,n);
        for(i=0;i<t;++i)
        {
            y=multi_mod(x,x,n);
            if(y==1 && x!=1 && x!=n-1)
                return false;
            x=y;
        }
        if(y!=1)
            return false;
    }
    return true;
}
```

4.9 mod

```
(a+b)%m = (a%m+b%m)%m
(a-b)%m = ((a%m-b%m)%m+m)%m (a>b)
(a*b)%m = (a%m*b%m)%m

(a/b)%m = (a%(b*m))/b //
```

4.10 pollard rho.cpp

```
#include<cstdio>
#include<cstdlib>
#include<list>

short T;
unsigned long long a;
std::list<unsigned long long>fac;

inline unsigned long long multi_mod(const unsigned long long &a,unsigned long long b,const unsigned long long &n)
{
    unsigned long long exp(a%n),tmp(0);
    while(b)
    {
        if(b&1)
        {
            tmp+=exp;
            if(tmp>n)
                tmp-=n;
        }
        exp<<=1;
        if(exp>n)
            exp-=n;
        b>>=1;
    }
    return tmp;
}

inline unsigned long long exp_mod(unsigned long long a,
    unsigned long long b,const unsigned long long &c)
{
    unsigned long long tmp(1);
    while(b)
    {
        if(b&1)
            tmp=multi_mod(tmp,a,c);
        a=multi_mod(a,a,c);
        b>>=1;
    }
    return tmp;
}

inline bool miller_rabbin(const unsigned long long &n,
    short T)
{
    if(n==2)
        return true;
    if(n<2 || !(n&1))
        return false;
    unsigned long long a,u(n-1),x,y;
    short t(0),i;
    while(!(u&1))
    {
        ++t;
        u>>=1;
    }
    while(T--)
    {
        a=rand()%(n-1)+1;
        x=exp_mod(a,u,n);
        for(i=0;i<t;++i)
        {
            y=multi_mod(x,x,n);
            if(y==1 && x!=1 && x!=n-1)
                return false;
            x=y;
        }
        if(y!=1)
            return false;
    }
    return true;
}

unsigned long long gcd(const unsigned long long &a,const
    unsigned long long &b)
{
    return b?gcd(b,a%b):a;
}

inline unsigned long long pollar_rho(const unsigned long
    long n,const unsigned long long &c)
{
    unsigned long long x(rand()%(n-1)+1),y,d,i(1),k(2);
    y=x;
    while(true)
    {
        ++i;
        x=(multi_mod(x,x,n)+c)%n;
        d=gcd((x-y+n)%n,n);
        if(d>1 && d<n)
            return d;
        if(x==y)
            return n;
        if(i==k)
        {
            k<<=1;
            y=x;
        }
    }
}

void find(const unsigned long long &n,short c)
{
    if(n==1)
        return;
    if(miller_rabbin(n,6))
    {
        fac.push_back(n);
        return;
    }
    unsigned long long p(n);
    short k(c);
    while(p>=n)
        p=pollar_rho(p,c--);
    find(p,k);
    find(n/p,k);
}

int main()
{
    scanf("%hd",&T);
    while(T--)
    {
        scanf("%llu",&a);
        fac.clear();
        find(a,120);
        if(fac.size()==1)
            puts("Prime");
        else
        {
            fac.sort();
            printf("%llu\n",fac.front());
        }
    }
    return 0;
}
```

4.11 prime.cpp

```
#include<vector>

std::vector<int>prm;
bool flag[MAXX];

int main()
{
    prm.reserve(MAXX); // pi(x)=x/ln(x);
    for(i=2;i<MAXX;++i)
    {
        if(!flag[i])
            prm.push_back(i);
        for(j=0;j<prm.size() && i*prm[j]<MAXX;++j)
        {
            flag[i*prm[j]]=true;
            if(i%pmr[j]==0)
                break;
        }
    }
    return 0;
}
```

Chapter 5

others

5.1 .vimrc

```
set number
set history=1000000
set autoindent
set smartindent
set tabstop=4
set shiftwidth=4
set expandtab
set showmatch

set nocp
filetype plugin indent on

filetype on
syntax on
```

5.2 bigint.cpp

```
// header files
#include <cstdio>
#include <string>
#include <algorithm>
#include <iostream>

struct Bigint
{
    // representations and structures
    std::string a; // to store the digits
    int sign; // sign = -1 for negative numbers, sign = 1 otherwise
    // constructors
    Bigint() {} // default constructor
    Bigint( std::string b ) { (*this) = b; } // constructor for std::string
    // some helpful methods
    int size() // returns number of digits
    {
        return a.size();
    }
    Bigint inverseSign() // changes the sign
    {
        sign *= -1;
        return (*this);
    }
    Bigint normalize( int newSign ) // removes leading 0, fixes sign
    {
        for( int i = a.size() - 1; i > 0 && a[i] == '0'; i-- )
            a.erase(a.begin() + i);
        sign = ( a.size() == 1 && a[0] == '0' ) ? 1 : newSign;
        return (*this);
    }
    // assignment operator
    void operator = ( std::string b ) // assigns a std::string to Bigint
    {
        a = b[0] == '-' ? b.substr(1) : b;
        reverse( a.begin(), a.end() );
        this->normalize( b[0] == '-' ? -1 : 1 );
    }
    // conditional operators
    bool operator < ( const Bigint &b ) const // less than operator
    {
        if( sign != b.sign )
            return sign < b.sign;
        if( a.size() != b.a.size() )
            return sign == 1 ? a.size() < b.a.size() : a.size() > b.a.size();
        for( int i = a.size() - 1; i >= 0; i-- )
            if( a[i] != b.a[i] )
                return sign == 1 ? a[i] < b.a[i] : a[i] >
                b.a[i];
        return false;
    }
    bool operator == ( const Bigint &b ) const // operator for equality
    {
        return a == b.a && sign == b.sign;
    }

    // mathematical operators
    Bigint operator + ( Bigint b ) // addition operator overloading
    {
        if( sign != b.sign )
            return (*this) - b.inverseSign();
        Bigint c;
        for(int i = 0, carry = 0; i<a.size() || i<b.size() || carry; i++ )
        {
            carry+=(i<a.size() ? a[i]-48 : 0)+(i<b.a.size() ? b.a[i]-48 : 0);
            c.a += (carry % 10 + 48);
            carry /= 10;
        }
        return c.normalize(sign);
    }

    Bigint operator - ( Bigint b ) // subtraction operator overloading
    {
        if( sign != b.sign )
            return (*this) + b.inverseSign();
        int s = sign; sign = b.sign = 1;
        if( (*this) < b )
            return ((b - (*this)).inverseSign()).normalize(-s);
        Bigint c;
        for( int i = 0, borrow = 0; i < a.size(); i++ )
        {
            borrow = a[i] - borrow - (i < b.size() ? b.a[i] : 48);
            c.a += borrow >= 0 ? borrow + 48 : borrow + 58;
            borrow = borrow >= 0 ? 0 : 1;
        }
        return c.normalize(s);
    }
    Bigint operator * ( Bigint b ) // multiplication operator overloading
    {
        Bigint c("0");
        for( int i = 0, k = a[i] - 48; i < a.size(); i++, k = a[i] - 48 )
        {
            while(k--)
                c = c + b; // ith digit is k, so, we add k times
            b.a.insert(b.a.begin(), '0'); // multiplied by 10
        }
        return c.normalize(sign * b.sign);
    }
    Bigint operator / ( Bigint b ) // division operator overloading
    {
        if( b.size() == 1 && b.a[0] == '0' )
            b.a[0] /= ( b.a[0] - 48 );
        Bigint c("0"), d;
        for( int j = 0; j < a.size(); j++ )
            d.a += "0";
        int dSign = sign * b.sign;
        b.sign = 1;
        for( int i = a.size() - 1; i >= 0; i-- )
        {
            c.a.insert( c.a.begin(), '0');
            c = c + a.substr( i, 1 );
            while( !( c < b ) )
            {
                c = c - b;
                d.a[i]++;
            }
        }
        return d.normalize(dSign);
    }
    Bigint operator % ( Bigint b ) // modulo operator overloading
    {
        if( b.size() == 1 && b.a[0] == '0' )
            b.a[0] /= ( b.a[0] - 48 );
        Bigint c("0");
        b.sign = 1;
        for( int i = a.size() - 1; i >= 0; i-- )
        {
            c.a.insert( c.a.begin(), '0');
            c = c + a.substr( i, 1 );
            while( !( c < b ) )
                c = c - b;
        }
        return c.normalize(sign);
    }

    // output method
    void print()
    {
        if( sign == -1 )
            putchar('-');
        for( int i = a.size() - 1; i >= 0; i-- )
            putchar(a[i]);
    }
};



int main()
{
    Bigint a, b, c; // declared some Bigint variables
    /////////////////////////
    // taking Bigint input //
    /////////////////////////

    std::string input; // std::string to take input
    std::cin >> input; // take the Big integer as std::string
    a = input; // assign the std::string to Bigint a

    std::cin >> input; // take the Big integer as std::string
    b = input; // assign the std::string to Bigint b

    ////////////////////////////////////
    // Using mathematical operators //
    ////////////////////////////////////

    c = a + b; // adding a and b
    c.print(); // printing the Bigint
    puts(""); // newline

    c = a - b; // subtracting b from a
    c.print(); // printing the Bigint
    puts(""); // newline

    c = a * b; // multiplying a and b
    c.print(); // printing the Bigint
    puts(""); // newline

    c = a / b; // dividing a by b
    c.print(); // printing the Bigint
    puts(""); // newline

    c = a % b; // a modulo b
    c.print(); // printing the Bigint
    puts(""); // newline

    ////////////////////////////////////
    // Using conditional operators //
    ////////////////////////////////////

    if( a == b )
        puts("equal"); // checking equality
    else
        puts("not␣equal");

    if( a < b )
        puts("a␣is␣smaller␣than␣b"); // checking less than operator

    return 0;
}
```

5.3 java.java

```
//Scanner

Scanner in=new Scanner(new FileReader("asdf"));
PrintWriter pw=new PrintWriter(new Filewriter("out"));
boolean         in.hasNext();
String          in.next();
BigDecimal      in.nextBigDecimal();
BigInteger      in.nextBigInteger();
BigInteger      in.nextBigInteger(int radix);
double          in.nextDouble();
int             in.nextInt();
int             in.nextInt(int radix);
String          in.nextLine();
```

```
long          in.nextLong();
long          in.nextLong(int radix);
short         in.nextShort();
short         in.nextShort(int radix);
int           in.radix(); //Returns this scanner's default radix.
Scanner       in.useRadix(int radix);// Sets this scanner's default radix to the specified radix.
void          in.close();//Closes this scanner.

//String

char          str.charAt(int index);
int           str.compareTo(String anotherString); // <0 if less. ==0 if equal. >0 if greater.
int           str.compareToIgnoreCase(String str);
String        str.concat(String str);
boolean       str.contains(CharSequence s);
boolean       str.endsWith(String suffix);
boolean       str.startsWith(String preffix);
boolean       str.startsWith(String preffix,int toffset);
int           str.hashCode();
int           str.indexOf(int ch);
int           str.indexOf(int ch,int fromIndex);
int           str.indexOf(String str);
int           str.indexOf(String str,int fromIndex);
int           str.lastIndexOf(int ch);
int           str.lastIndexOf(int ch,int fromIndex);
//(ry
int           str.length();
String        str.substring(int beginIndex);
String        str.substring(int beginIndex,int endIndex);
String        str.toLowerCase();
String        str.toUpperCase();
String        str.trim();// Returns a copy of the string, with leading and trailing whitespace omitted.

//StringBuilder
StringBuilder str.insert(int offset,...);
StringBuilder str.reverse();
void          str.setCharAt(int index,int ch);

//BigInteger
compareTo(); equals(); doubleValue(); longValue(); hashCode(); toString(); toString(int radix); max(); min(); mod(); modPow(BigInteger exp,BigInteger m);
```

```
nextProbablePrime(); pow();
andNot(); and(); xor(); not(); or(); getLowestSetBit(); bitCount(); bitLength(); setBig(int n); shiftLeft(int n); shiftRight(int n);
add(); divide(); divideAndRemainder(); remainder(); multiply(); subtract(); gcd(); abs(); signum(); negate();

//BigDecimal
movePointLeft(); movePointRight(); precision(); stripTrailingZeros(); toBigInteger(); toPlainString();
```

5.4 others

```
chmod +x [filename]

while true; do
./gen > input
./sol < input > output.sol
./bf < input > output.bf

diff output.sol output.bf
if[ $? -ne 0];then break fi
done


1


2 calm_down();calm_down();calm_down();
3
4
5     30min          TM         ,
```

Chapter 6

search

6.1 dlx

```
                                        0 1



                                        0 1

```

6.2 dlx - precise cover.cpp

```
#include<cstdio>

#define INF 0x7FFFFFFF
#define MAXN 1000010

int n, m, size;
int L[MAXN], R[MAXN], U[MAXN], D[MAXN], H[MAXN];
int S[MAXN], C[MAXN], X[MAXN], Q[MAXN];

void Init()
{
    int i;
    for (i = 0; i <= m; i++)
    {
        S[i] = 0;
        L[i + 1] = i;
        R[i] = i + 1;
        U[i] = D[i] = i;
    }
    R[m] = 0;
    size = m + 1;
}
void Remove(int c)
{
    int i, j;
    R[L[c]] = R[c];
    L[R[c]] = L[c];
    for (i = D[c]; i != c; i = D[i])
    {
        for (j = R[i]; j != i; j = R[j])
        {
            D[U[j]] = D[j];
            U[D[j]] = U[j];
            S[C[j]]--;
        }
    }
}
void Resume(int c)
{
    int i, j;
    R[L[c]] = c;
    L[R[c]] = c;
    for (i = D[c]; i != c; i = D[i])
    {
        for (j = R[i]; j != i; j = R[j])
        {
            U[D[j]] = j;
            D[U[j]] = j;
            S[C[j]]++;
        }
    }
}
void Link(int r, int c)
{
    D[size] = D[c];
    U[size] = c;
    U[D[c]] = size;
    D[c] = size;
    if (H[r] < 0)
        H[r] = L[size] = R[size] = size;
    else
    {
        L[size] = H[r];
        R[size] = R[H[r]];
        L[R[H[r]]] = size;
        R[H[r]] = size;
    }
    S[c]++;
    C[size] = c;
    X[size++] = r;
}
bool Dance(int now)
{
    int i, j, c, temp;
    if (R[0] == 0)
        return true;
    for (temp = INF, i = R[0]; i; i = R[i])
    {
        if (S[i] < temp)
        {
            c = i;
            temp = S[i];
        }
    }
    Remove(c);
    for (i = D[c]; i != c; i = D[i])
    {
        for (j = R[i]; j != i; j = R[j])
            Remove(C[j]);
        if (Dance(now + 1))
            return true;
        for (j = L[i]; j != i; j = L[j])
            Resume(C[j]);
    }
    Resume(c);
    return false;
}
```

6.3 dlx - repeat cover.cpp

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

#define MAXN 110
#define MAXM 1000000
#define INF 0x7FFFFFFF

```

```
using namespace std;

int G[MAXN][MAXN];
int L[MAXM], R[MAXM], U[MAXM], D[MAXM];
int size, ans, S[MAXM], H[MAXM], C[MAXM];
bool vis[MAXN * 100];
void Link(int r, int c)
{
    U[size] = c;
    D[size] = D[c];
    U[D[c]] = size;
    D[c] = size;
    if (H[r] < 0)
        H[r] = L[size] = R[size] = size;
    else
    {
        L[size] = H[r];
        R[size] = R[H[r]];
        L[R[H[r]]] = size;
        R[H[r]] = size;
    }
    S[c]++;
    C[size++] = c;
}
void Remove(int c)
{
    int i;
    for (i = D[c]; i != c; i = D[i])
    {
        L[R[i]] = L[i];
        R[L[i]] = R[i];
    }
}
void Resume(int c)
{
    int i;
    for (i = D[c]; i != c; i = D[i])
        L[R[i]] = R[L[i]] = i;
}
int A()
{
    int i, j, k, res;
    memset(vis, false, sizeof(vis));
    for (res = 0, i = R[0]; i; i = R[i])
    {
        if (!vis[i])
        {
            res++;
            for (j = D[i]; j != i; j = D[j])
            {
                for (k = R[j]; k != j; k = R[k])
                    vis[C[k]] = true;
            }
        }
    }
    return res;
}
void Dance(int now)
{
    if (R[0] == 0)
        ans = min(ans, now);
    else if (now + A() < ans)
    {
        int i, j, temp, c;
        for (temp = INF, i = R[0]; i; i = R[i])
        {
            if (temp > S[i])
            {
                temp = S[i];
                c = i;
            }
        }
        for (i = D[c]; i != c; i = D[i])
        {
            Remove(i);
            for (j = R[i]; j != i; j = R[j])
                Remove(j);
            Dance(now + 1);
            for (j = L[i]; j != i; j = L[j])
                Resume(j);
            Resume(i);
        }
    }
}
void Init(int m)
{
    int i;
    for (i = 0; i <= m; i++)
    {
        R[i] = i + 1;
        L[i + 1] = i;
        U[i] = D[i] = i;
```

```
            S[i] = 0;
        }
        R[m] = 0;
        size = m + 1;
}
```

6.4 fibonacci knapsack.cpp

```
#include<stdio.h>
#include<stdlib.h>
#include<algorithm>

#define MAXX 71

struct mono
{
    long long weig,cost;
}goods[MAXX];

short n,T,t,i;
long long carry,sumw,sumc;
long long ans,las[MAXX];

int com(const void *n,const void *m)
{
    struct mono *a=(struct mono *)n,*b=(struct mono *)m;
    if(a->weig!=b->weig)
        return a->weig-b->weig;
    else
        return b->cost-a->cost;
}

bool comp(const struct mono a,const struct mono b)
{
    if(a.weig!=b.weig)
        return a.weig<b.weig;
    else
        return b.cost<a.cost;
}

void dfs(short i,long long cost_n,long long carry_n,short last)
{
    if(ans<cost_n)
        ans=cost_n;
    if(i==n || goods[i].weig>carry_n || cost_n+las[i]<=ans)
        return;
    if(last || (goods[i].weig!=goods[i-1].weig && goods[i].cost>goods[i-1].cost))
        dfs(i+1,cost_n+goods[i].cost,carry_n-goods[i].weig,1);
    dfs(i+1,cost_n,carry_n,0);
}

int main()
{
    //      freopen("asdf","r",stdin);
    scanf("%hd",&T);
    for(t=1;t<=T;++t)
    {
        scanf("%hd%lld",&n,&carry);
        sumw=0;
        sumc=0;
        ans=0;
        for(i=0;i<n;++i)
        {
            scanf("%lld%lld",&goods[i].weig,&goods[i].cost);
            sumw+=goods[i].weig;
            sumc+=goods[i].cost;
        }
        if(sumw<=carry)
        {
            printf("Case %hd: %lld\n",t,sumc);
            continue;
        }
//          qsort(goods,n,sizeof(struct mono),com);
        std::sort(goods,goods+n,comp);
        for(i=0;i<n;++i)
        {
//              printf("%lld %lld\n",goods[i].weig,goods[i].cost);
            las[i]=sumc;
            sumc-=goods[i].cost;
        }
        dfs(0,0,carry,1);
        printf("Case %hd: %lld\n",t,ans);
    }
    return 0;
}
```

Chapter 7

string

7.1 aho corasick.cpp

```
#include<cstring>
#include<queue>

#define MAX 1000111
#define N 26

int nxt[MAX][N],fal[MAX],cnt;
bool ed[MAX];
char buf[MAX];

inline void init(int a)
{
    memset(nxt[a],0,sizeof(nxt[0]));
    fal[a]=0;
    ed[a]=false;
}

inline void insert()
{
    static int i,p;
    for(i=p=0;buf[i];++i)
    {
        if(!nxt[p][map[buf[i]]])
            init(nxt[p][map[buf[i]]]=++cnt);
        p=nxt[p][map[buf[i]]];
    }
    ed[p]=true;
}
```

```

inline void make()
{
    static std::queue<int>q;
    int i,now,p;
    q.push(0);
    while(!q.empty())
    {
        now=q.front();
        q.pop();
        for(i=0;i<N;++i)
            if(nxt[now][i])
            {
                q.push(p=nxt[now][i]);
                if(now)
                    fal[p]=nxt[fal[now]][i];
                ed[p]|=ed[fal[p]];
            }
            else
                nxt[now][i]=nxt[fal[now]][i];
    }
}
```

7.2 manacher.cpp

```
#include<cstdio>
#include<vector>

#define MAXX 1111

std::vector<char>str;
char buf[MAXX];
int z[MAXX<<1];
int i,j,l,r;
int ii,n,c;

inline int match(const int &a,const int &b)
{
    int i(0);
    while(a-i>=0 && b+i<str.size() && str[a-i]==str[b+i])
        ++i;
    return i;
}

int main()
{
    gets(buf);
    str.reserve(MAXX<<1);
    for(i=0;buf[i];++i)
    {
        str.push_back('$');
        str.push_back(buf[i]);
    }
    str.push_back('$');

    z[0]=1;
    c=l=r=0;
    for(i=1;i<str.size();++i)
    {
        ii=(l<<1)-i;
        n=r+1-i;

        if(i>r)
        {
            z[i]=match(i,i);
            l=i;
            r=i+z[i]-1;
        }
        else
            if(z[ii]==n)
            {
                z[i]=n+match(i-n,i+n);
                l=i;
                r=i+z[i]-1;
            }
            else
                z[i]=std::min(z[ii],n);
        if(z[i]>z[c])
            c=i;
    }

    for(i=c-z[c]+2,n=c+z[c];i<n;i+=2)
        putchar(str[i]);
    puts("");
    return 0;
}
```

7.3 morris-pratt.cpp

```
int i,j;

inline void make(char *buf,int *fal)
```

```
{
    fal[0]=-1;
    for(i=1,j=-1;buf[i];++i)
    {
        while(j>=0 && buf[j+1]!=buf[i])
            j=fal[j];
        if(buf[j+1]==buf[i])
            ++j;
        fal[i]=j;
    }

}

inline int void match(char *p,char *t,int* fal)
{
    for(i=0,j=-1;t[i];++i)
    {
        while(j>=0 && p[j+1]!=t[i])
            j=fal[j];
        if(p[j+1]==t[i])
            ++j;
        if(!p[j+1])
        {
            //
            j=fal[j];
        }
}
```

7.4 smallest representation.cpp

```
int min(char a[],int len)
{
    int i = 0,j = 1,k = 0;
    while (i < len && j < len && k < len)
    {
        int cmp = a[(j+k)%len]-a[(i+k)%len];
        if (cmp == 0)
            k++;
        else
        {
            if (cmp > 0)
                j += k+1;
            else
                i += k+1;
            if (i == j) j++;
            k = 0;
        }
    }
    return std::min(i,j);
}
```

7.5 suffix array - da.cpp

```
int wx[maxn],wy[maxn],*x,*y,wss[maxn],wv[maxn];

bool cmp(int *r,int n,int a,int b,int l)
{
    return a+l<n && b+l<n && r[a]==r[b]&&r[a+l]==r[b+l];
}
void da(int str[],int sa[],int rank[],int height[],int n, int m)
{
    int *s = str;
    int *x=wx,*y=wy,*t,p;
    int i,j;
    for(i=0; i<m; i++)
        wss[i]=0;
    for(i=0; i<n; i++)
        wss[x[i]=s[i]]++;
    for(i=1; i<m; i++)
        wss[i]+=wss[i-1];
    for(i=n-1; i>=0; i--)
        sa[--wss[x[i]]]=i;
    for(j=1,p=1; p<n && j<n; j*=2,m=p)
    {
        for(i=n-j,p=0; i<n; i++)
            y[p++]=i;
        for(i=0; i<n; i++)
            if(sa[i]-j>=0)
                y[p++]=sa[i]-j;
        for(i=0; i<n; i++)
            wv[i]=x[y[i]];
        for(i=0; i<m; i++)
            wss[i]=0;
        for(i=0; i<n; i++)
            wss[wv[i]]++;
        for(i=1; i<m; i++)
            wss[i]+=wss[i-1];
        for(i=n-1; i>=0; i--)
            sa[--wss[wv[i]]]=y[i];
        for(t=x,x=y,y=t,p=1,i=1,x[sa[0]]=0; i<n; i++)
            x[sa[i]]=cmp(y,n,sa[i-1],sa[i],j)?p-1:p++;
    }
    for(int i=0; i<n; i++)
        rank[sa[i]]=i;
    for(int i=0,j=0,k=0; i<n; height[rank[i++]]=k)
        if(rank[i]>0)
            for(k?k--:0,j=sa[rank[i]-1]; i+k < n && j+k < n && str[i+k]==str[j+k]; ++k);
}
```

7.6 suffix array.cpp

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

#define MAXX 1111
#define F(x) ((x)/3+((x)%3==1?0:tb))
#define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)

int wa[MAXX],wb[MAXX],wv[MAXX],ws[MAXX];

inline bool c0(const int *str,const int &a,const int &b)
{
    return str[a]==str[b] && str[a+1]==str[b+1] && str[a+2]==str[b+2];
}

inline bool c12(const int *str,const int &k,const int &a,const int &b)
{
    if(k==2)
        return str[a]<str[b] || str[a]==str[b] && c12(str,1,a+1,b+1);
    else
        return str[a]<str[b] || str[a]==str[b] && wv[a+1]<wv[b+1];
}

inline void sort(int *str,int *a,int *b,const int &n,const int &m)
{
    memset(ws,0,sizeof(ws));
    int i;
    for(i=0;i<n;++i)
        ++ws[wv[i]=str[a[i]]];
    for(i=1;i<m;++i)
```

```
        ws[i]+=ws[i-1];
    for(i=n-1;i>=0;--i)
        b[--ws[wv[i]]]=a[i];
}

inline void dc3(int *str,int *sa,const int &n,const int &m)
{
    int *strn(str+n);
    int *san(sa+n),tb((n+1)/3),ta(0),tbc(0),i,j,k;
    str[n]=str[n+1]=0;
    for(i=0;i<n;++i)
        if(i%3)
            wa[tbc++]=i;
    sort(str+2,wa,wb,tbc,m);
    sort(str+1,wb,wa,tbc,m);
    sort(str,wa,wb,tbc,m);
    for(i=j=1,strn[F(wb[0])]=0;i<tbc;++i)
        strn[F(wb[i])]=c0(str,wb[i-1],wb[i])?j-1:j++;
    if(j<tbc)
        dc3(strn,san,tbc,j);
    else
        for(i=0;i<tbc;++i)
            san[strn[i]]=i;
    for(i=0;i<tbc;++i)
        if(san[i]<tb)
            wb[ta++]=san[i]*3;
    if(n%3==1)
        wb[ta++]=n-1;
    sort(str,wb,wa,ta,m);
    for(i=0;i<tbc;++i)
        wv[wb[i]=G(san[i])]=i;
    for(i=j=k=0;i<ta && j<tbc;)
        sa[k++]=c12(str,wb[j]%3,wa[i],wb[j])?wa[i++]:wb[j++];
    while(i<ta)
        sa[k++]=wa[i++];
    while(j<tbc)
        sa[k++]=wb[j++];
}

int rk[MAXX],lcpa[MAXX],sa[MAXX*3];
int str[MAXX*3]; //     int

int main()
{
    scanf("%d␣%d",&n,&j);
    for(i=0;i<n;++i)
    {
        scanf("%d",&k);
        num[i]=k-j+100;
        j=k;
    }
    num[n]=0;

    dc3(num,sa,n+1,191); //191:
        str

    for(i=1;i<=n;++i) // rank
        rk[sa[i]]=i;
    for(i=k=0;i<n;++i) // lcp
        if(!rk[i])
            lcpa[0]=0;
        else
        {
            j=sa[rk[i]-1];
            if(k>0)
                --k;
            while(num[i+k]==num[j+k])
                ++k;
            lcpa[rk[i]]=k;
        }


    for(i=1;i<=n;++i)
        sptb[0][i]=i;
    for(i=1;i<=lg[n];++i)  //sparse table RMQ
    {
        k=n+1-(1<<i);
        for(j=1;j<=k;++j)
        {
            a=sptb[i-1][j];
            b=sptb[i-1][j+(1<<(i-1))];
            sptb[i][j]=lcpa[a]<lcpa[b]?a:b;
        }
    }
}

inline int ask(int l,int r)
{
    a=lg[r-l+1];
    r-=(1<<a)-1;
    l=sptb[a][l];
    r=sptb[a][r];
    return lcpa[l]<lcpa[r]?l:r;
}

inline int lcp(int l,int r) //            [l,r]
          rmq
{
    l=rk[l];
    r=rk[r];
    if(l>r)
        std::swap(l,r);
    return lcpa[ask(l+1,r)];
}
```

7.7 z algorithm.cpp

```
inline void make(int *z,char *buf)
{
    int i,j,l,r;
    l=0;
    r=1;
    z[0]=strlen(buf);
    for(i=1;i<z[0];++i)
        if(r<=i || z[i-l]>=r-i)
        {
            j=std::max(i,r);
            while(j<z[0] && buf[j]==buf[j-i])
                ++j;
            z[i]=j-i;
            if(i<j)
            {
                l=i;
                r=j;
            }
        }
        else
            z[i]=z[i-l];
}

for(i=1;i<len && i+z[i]<len;++i); //i=
```