ACM-ICPC Template

tokitsukaze

2021 年 10 月 16 日

目录

1 字符串

1.1 KMP

1.1.1 KMP

```
//length of min loop len-nex[len]
struct KMP
{
    int nex[MAX],len;
    char s[MAX];
    void get_next()
    {
        int i,j;
        i=0;
        j=nex[0]=-1;
        while(i<len)
        {
            if(j==-1||s[i]==s[j]) nex[++i]=++j;
            else j=nex[j];
        }
    }
    void init(char *_s)
    {
        len=strlen(_s);
        for(int i=0;i<len;i++) s[i]=_s[i];
        s[len]='\0';
        get_next();
    }
    int match(char *a)//s is a substring of a
    {
        int n,i,j;
        n=strlen(a);
        for(i=j=0;i<n;i++)
        {
            if(j==-1||a[i]==s[j]) j++;
            else j=nex[j];
            if(j==len) return 1;
        }
        return 0;
    }
}kmp;// kmp.init(s); s[0..len-1]
```

1.1.2 exKMP

```
struct exKMP
{
    int next[MAX];
    void getnext(char *s)
    {
        int i,j,pos,len;
        next[i=0]=len=strlen(s);
        while(s[i]==s[i+1]&&i+1<len) i++;
        next[1]=i;
        pos=1;
        for(i=2;i<len;i++)
        {
            if(next[i-pos]+i<next[pos]+pos) next[i]=
                next[i-pos];
            else
            {
                j=max(0,next[pos]+pos-i);
                while(i+j<len&&s[j]==s[j+i]) j++;
                next[i]=j;
                pos=i;
            }
        }
    }
    void work(char *a,char *b,int *ex)// a,b 0~len-1
    {
        int i=0,j,pos,lena,lenb;
        getnext(b);
        lena=strlen(a);
        lenb=strlen(b);
        i=0;
        while(a[i]==b[i]&&i<lenb&&i<lena) i++;
        ex[0]=i;
        pos=0;
        for(i=1;i<lena;i++)
        {
            if(next[i-pos]+i<ex[pos]+pos) ex[i]=next[
                i-pos];
            else
            {
                j=max(0,ex[pos]+pos-i);
                while(i+j<lena&&j<lenb&&a[j+i]==b[j])
                    j++;
                ex[i]=j;
                pos=i;
            }
        }
    }
}exkmp;
```

1.2 Hash

1.2.1 hash

```
struct hash_table
{
    ll seed,p;
    ll Hash[MAX],tmp[MAX];
    void set(ll _seed,ll _p)
    {
        seed=_seed;
        p=_p;
    }
    void work(char *s,int n)
```

```
    {
        tmp[0]=1;
        Hash[0]=0;
        for(int i=1;i<=n;i++)
        {
            tmp[i]=tmp[i-1]*seed%p;
            Hash[i]=(Hash[i-1]*seed+s[i])%p;//may
                need change
        }
    }
    ll get(int l,int r)
    {
        return ((Hash[r]-Hash[l-1]*tmp[r-l+1])%p+p)%p
            ;
    }
};
```

1.2.2 hash_map

```
struct hash_map
{
    static const int p=999917;
    ll val[MAX],w[MAX];
    int tot,head[p],nex[MAX];
    int top,st[MAX];
    void clear(){tot=0;while(top) head[st[top
        --]]=0;}
    void add(int x,ll y){val[++tot]=y;nex[tot]=head[
        x];head[x]=tot;w[tot]=0;}
    bool count(ll y)
    {
        int x=y%p;
        for(int i=head[x];i;i=nex[i])
        {
            if(y==val[i]) return 1;
        }
        return 0;
    }
    ll& operator [](ll y)
    {
        int x=y%p;
        for(int i=head[x];i;i=nex[i])
        {
            if(y==val[i]) return w[i];
        }
        add(x,y);
        st[++top]=x;
        return w[tot];
    }
}mp;
```

1.2.3 BKDRHash

```
struct BKDRHash
{
    static const ull seed=1313131;//
        31,131,1313,13131,131313
    static const int p=2000007;
    ull Hash[MAX],tmp[MAX];
    ull val[MAX];
    int last[p+10],nex[MAX],cnt;
    void init()//clear hash table
    {
        mem(last,0);
        cnt=0;
    }
    bool insert(ull x)
    {
        int u=x%p;
        for(int i=last[u];i;i=nex[i])
        {
            if(val[i]==x) return 1;
        }
        nex[++cnt]=last[u];
        last[u]=cnt;
        val[cnt]=x;
        return 0;
    }
    void work(char *s,int n)
    {
        tmp[0]=1;
        Hash[0]=0;
        for(int i=1;i<=n;i++)
        {
            tmp[i]=tmp[i-1]*seed;
            Hash[i]=Hash[i-1]*seed+s[i];
        }
    }
    ull get(int l,int r)
    {
        return Hash[r]-Hash[l-1]*tmp[r-l+1];
    }
}bkdr; //bkdr.init();
```

1.3 Manacher

1.3.1 插字符

```
struct Manacher
{
    int p[MAX<<1];
    char s[MAX<<1];
    int work(char *a)
    {
        int len,i,mid,r,res=0;
        len=strlen(a+1);
```

```
        for(i=1;i<=len;i++)
        {
            p[i]=0;
            s[2*i-1]='%';
            s[2*i]=a[i];
        }
        s[len=len*2+1]='%';
        mid=r=0;
        for(i=1;i<=len;i++)
        {
            if(i<r) p[i]=min(p[2*mid-i],r-i);
            else p[i]=1;
            while(i-p[i]>=1&&i+p[i]<=len&&s[i-p[i]]==
                s[i+p[i]]) p[i]++;
            if(i+p[i]>r)
            {
                r=i+p[i];
                mid=i;
            }
            res=max(res,p[i]-1);
        }
        return res;
    }
}la;
```

1.3.2 不插字符

```
struct Manacher
{
    int p[MAX];
    int work(char *s)//return max length of
        palindrome
    {
        int r,mid,i,len,res=0;
        len=strlen(s+1);
        //odd
        r=mid=0;
        mem(p,0);
        for(i=1;i<=len;i++)
        {
            //palindrome substring s[i,i]
            if(r>i) p[i]=min(p[2*mid-i],r-i);
            while(i+p[i]+1<=len&&s[i+p[i]+1]==s[i-p[i
                ]-1])
            {
                //palindrome substring s[i-p[i]-1,i+p[
                    i]+1]
                p[i]++;
            }
            if(i+p[i]>r)
            {
                r=i+p[i];
                mid=i;
            }
            res=max(res,p[i]*2+1);
        }
        //even
        r=mid=0;
        mem(p,0);
        for(i=2;i<=len;i++)
        {
            if(r>i) p[i]=min(p[2*mid-i],r-i+1);
            while(i+p[i]<=len&&s[i+p[i]]==s[i-p[i
                ]-1])
            {
                //palindrome substring s[i-p[i]-1,i+p[
                    i]]
                p[i]++;
            }
            if(i+p[i]-1>r)
            {
                r=i+p[i]-1;
                mid=i;
            }
            res=max(res,p[i]*2);
        }
        return res;
    }
}la;
```

1.4 后缀数组

1.4.1 倍增 sa

```
struct SA
{
    int s[MAX],n,SZ;
    int c[MAX],rk[MAX],tp[MAX];
    void init(char *ss) //s[1..n]
    {
        SZ=0;
        n=strlen(ss+1);
        for(int i=1;i<=n;i++)
        {
            s[i]=ss[i];
            SZ=max(SZ,s[i]);
        }
    }
    void get_sa(int *sa,int *height)
    {
        int m,i,j,k,now;
        m=SZ;
        for(i=1;i<=m;i++) c[i]=0;
        for(i=1;i<=n;i++) c[rk[i]=s[i]]++;
        for(i=2;i<=m;i++) c[i]+=c[i-1];
        for(i=n;i;i--) sa[c[rk[i]]--]=i;
        for(k=1;k<=n;k<<=1)
        {
```

```
            now=0;
            for(i=n-k+1;i<=n;i++) tp[++now]=i;
            for(i=1;i<=n;i++)
            {
                if(sa[i]>k) tp[++now]=sa[i]-k;
            }
            for(i=1;i<=m;i++) c[i]=0;
            for(i=1;i<=n;i++) c[rk[i]]++;
            for(i=2;i<=m;i++) c[i]+=c[i-1];
            for(i=n;i;i--)
            {
                sa[c[rk[tp[i]]]--]=tp[i];
                tp[i]=0;
            }
            swap(rk,tp);
            rk[sa[1]]=1;
            now=1;
            for(i=2;i<=n;i++)
            {
                if(!(tp[sa[i]]==tp[sa[i-1]]&&tp[sa[i]+
                    k]==tp[sa[i-1]+k])) now++;
                rk[sa[i]]=now;
            }
            if(now==n) break;
            m=now;
        }
        height[1]=0;
        k=0;
        for(i=1;i<=n;i++)
        {
            if(rk[i]==1) continue;
            if(k) k--;
            j=sa[rk[i]-1];
            while(j+k<=n&&i+k<=n&&s[i+k]==s[j+k]) k
                ++;
            height[rk[i]]=k;
        }
    }
}da;
```

1.4.2 SA-IS

```
struct SA
{
    char S[MAX]; int n,m;
    int s[MAX<<1],t[MAX<<1],H[MAX],sa[MAX],r[MAX],p[
        MAX],c[MAX],w[MAX];
    inline int trans(int n,const char* S){
        int m=*max_element(S+1,S+1+n);
        for(int i=1;i<=n;++i) r[S[i]]=1;
        for(int i=1;i<=m;++i) r[i]+=r[i-1];
        for(int i=1;i<=n;++i) s[i]=r[S[i]];
        return r[m];
    }
    #define ps(x) sa[w[s[x]]--]=x
    #define pl(x) sa[w[s[x]]++]=x
    inline void radix(int* v,int* s,int* t,int n,int
         m,int n1){
        memset(sa,0,n+1<<2); memset(c,0,m+1<<2);
        for(int i=1;i<=n;++i) ++c[s[i]];
        for(int i=1;i<=m;++i) w[i]=c[i]+=c[i-1];
        for(int i=n1;i;--i) ps(v[i]);
        for(int i=1;i<=m;++i) w[i]=c[i-1]+1;
        for(int i=1;i<=n;++i) if(sa[i]>1 && t[sa[i
            ]-1]) pl(sa[i]-1);
        for(int i=1;i<=m;++i) w[i]=c[i];
        for(int i=n;i;--i) if(sa[i]>1 && !t[sa[i]-1])
             ps(sa[i]-1);
    }
    inline void SAIS(int n,int m,int* s,int* t,int*
        p){
        int n1=0,ch=r[1]=0,*s1=s+n; t[n]=0;
        for(int i=n-1;i;--i) t[i]=s[i]==s[i+1]?t[i
            +1]:s[i]>s[i+1];
        for(int i=2;i<=n;++i) r[i]=t[i-1]&&!t[i]?(p
            [++n1]=i,n1):0;
        radix(p,s,t,n,m,n1);
        for(int i=1,x,y;i<=n;++i) if(x=r[sa[i]]){
            if(ch<=1 || p[x+1]-p[x]!=p[y+1]-p[y]) ++
                ch;
            else for(int j=p[x],k=p[y];j<=p[x+1];++j
                ,++k)
                if((s[j]<<1|t[j])^(s[k]<<1|t[k])){ ++
                    ch; break; }
            s1[y=x]=ch;
        }
        if(ch<n1) SAIS(n1,ch,s1,t+n,p+n1);
        else for(int i=1;i<=n1;++i) sa[s1[i]]=i;
        for(int i=1;i<=n1;++i) s1[i]=p[sa[i]];
        radix(s1,s,t,n,m,n1);
    }
    inline void get_sa(int n,const char* S,int *ssa,
        int *h){
        int m=trans(++n,S); SAIS(n,m,s,t,p);
        for(int i=1;i<n;++i) r[sa[i]=sa[i+1]]=i;
        for(int i=1,j,k=0;i<n;++i) if(r[i]>1){
            for(j=sa[r[i]-1];S[i+k]==S[j+k];++k);
            if(H[r[i]]=k) --k;
        }
        for(int i=1;i<=n;i++)
        {
            ssa[i]=sa[i];
            h[i]=H[i];
        }
    }
}sais;
```

1.5 自动机

1.5.1 AC 自动机

```
struct AC_Automaton
{
    static const int K=26;//may need change
    int nex[MAX][K],fail[MAX],cnt[MAX],last[MAX];
    int root,tot;
    inline int getid(char c){return c-'a';}//may need change
    int newnode()
    {
        mem(nex[tot],0);
        fail[tot]=0;
        cnt[tot]=0;
        return tot++;
    }
    void init()
    {
        tot=0;
        root=newnode();
    }
    void insert(char *s)
    {
        int len,now,i;
        len=strlen(s);
        now=root;
        for(i=0;i<len;i++)
        {
            int t=getid(s[i]);
            if(!nex[now][t]) nex[now][t]=newnode();
            now=nex[now][t];
        }
        cnt[now]++;
    }
    void work()
    {
        int i,now;
        queue<int>q;
        for(i=0;i<K;i++)
        {
            if(nex[root][i]) q.push(nex[root][i]);
        }
        while(!q.empty())
        {
            now=q.front();
            q.pop();
            //suffix link
            if(cnt[fail[now]]) last[now]=fail[now];
            else last[now]=last[fail[now]];
            /*
            may need add something here:
            cnt[now]+=cnt[fail[now]];
            */
            for(i=0;i<K;i++)
            {
                if(nex[now][i])
                {
                    fail[nex[now][i]]=nex[fail[now]][i];
                    q.push(nex[now][i]);
                }
                else nex[now][i]=nex[fail[now]][i];
            }
        }
    }
    int query(char *s)
    {
        int len,now,i,res;
        len=strlen(s);
        now=root;
        res=0;
        for(i=0;i<len;i++)
        {
            int t=getid(s[i]);
            now=nex[now][t];
            int tmp=now;
            while(tmp&&cnt[tmp]!=-1)
            {
                res+=cnt[tmp];
                cnt[tmp]=-1;
                tmp=last[tmp];
            }
        }
        return res;
    }
    //build fail tree
    vector<int> mp[MAX];
    void build_tree()
    {
        for(int i=0;i<=tot;i++) mp[i].clear();
        for(int i=1;i<tot;i++) mp[fail[i]].pb(i);
    }
}ac;
```

1.5.2 大字符集 AC 自动机

```
struct AC_Automaton
{
    map<int,int> nex[MAX];
    VI toplist;
    int fail[MAX],last[MAX],cnt[MAX];
    int root,tot;
    int newnode()
    {
        tot++;
        nex[tot].clear();
        return tot;
```

```
    }
    void init()
    {
        toplist.clear();
        tot=0;
        root=newnode();
    }
    void insert(VI &s)
    {
        int len,now,i;
        len=sz(s);
        now=root;
        for(i=0;i<len;i++)
        {
            int t=s[i];
            if(!nex[now].count(t)) nex[now][t]=
                newnode();
            now=nex[now][t];
        }
        cnt[now]=1;
    }
    void work()
    {
        int i,now;
        queue<int>q;
        for(auto it:nex[root])
        {
            fail[it.se]=root;
            q.push(it.se);
        }
        fail[root]=1;
        while(!q.empty())
        {
            now=q.front();
            q.pop();
            toplist.pb(now);
            //suffix link
        /* if(cnt[fail[now]]) last[now]=fail[now];
            else last[now]=last[fail[now]];*/
            cnt[now]+=cnt[fail[now]];
            for(auto it:nex[now])
            {
                int fail_now=fail[now];
                while(fail_now>1&&!nex[fail_now].count
                    (it.fi)) fail_now=fail[fail_now];
                if(nex[fail_now].count(it.fi)) fail[it
                    .se]=nex[fail_now][it.fi];
                else fail[it.se]=root;
                q.push(it.se);
            }
        }
    }
    int query(VI& s,int x)
    {
        int len,now,i,res;
        len=sz(s);
        now=root;
        res=0;
        for(i=0;i<len;i++)
        {
            int t=s[i];
            while(now>1&&!nex[now].count(t)) now=fail
                [now];
            if(nex[now].count(t)) now=nex[now][t];
            else now=root;
            //do something
        }
        return res;
    }
    void toptrans()
    {
        for(int i=sz(toplist)-1;~i;i--)/*do something
            */;
    }
}ac;
```

1.5.3 后缀自动机

```
struct Suffix_Automaton
{
    static const int N=MAX<<1;
    static const int K=26;// char size: [0,25]
    int tot,last,nex[N][K],fa[N],len[N],cnt[N],
        maxlen;
    int newnode()
    {
        tot++;
        fa[tot]=len[tot]=cnt[tot]=0;
        mem(nex[tot],0);
        return tot;
    }
    void init()
    {
        fa[0]=len[0]=cnt[0]=0;
        mem(nex[0],0);
        tot=0;
        maxlen=0;
        last=newnode();
    }
    void add(int x)
    {
        int p,q,np,nq;
        p=last;
        np=last=newnode();
        len[np]=len[p]+1;
        maxlen=max(maxlen,len[np]);
        cnt[last]=1;
        while(p&&!nex[p][x])
```

```
        {
            nex[p][x]=np;
            p=fa[p];
        }
        if(p==0) fa[np]=1;
        else
        {
            q=nex[p][x];
            if(len[q]==len[p]+1) fa[np]=q;
            else
            {
                nq=newnode();
                memcpy(nex[nq],nex[q],sizeof(nex[q]));
                len[nq]=len[p]+1;
                maxlen=max(maxlen,len[nq]);
                fa[nq]=fa[q];
                fa[q]=fa[np]=nq;
                while(p&&nex[p][x]==q)
                {
                    nex[p][x]=nq;
                    p=fa[p];
                }
            }
        }
    }
    int sum[N],tp[N];
    void topsort()
    {
        int i;
        for(i=1;i<=maxlen;i++) sum[i]=0;
        for(i=1;i<=tot;i++) sum[len[i]]++;
        for(i=1;i<=maxlen;i++) sum[i]+=sum[i-1];
        for(i=1;i<=tot;i++) tp[sum[len[i]]--]=i;
        for(i=tot;i;i--) cnt[fa[tp[i]]]+=cnt[tp[i]];
    }
    void build_tree(VI mp[])
    {
        for(int i=1;i<=tot;i++) mp[i].clear();
        for(int i=1;i<=tot;i++) mp[fa[i]].pb(i);
    }
    int pos[N],id[N];
    void init_pos(char *s,int n)//s[1..n]
    {
        int now=1;
        for(int i=1;i<=tot;i++) id[i]=-1;
        for(int i=1;i<=n;i++)
        {
            now=nex[now][s[i]-'a'];
            pos[i]=now;
            id[now]=i;
        }
    }
    int st[N][21];
    void init_ST()
    {
        int i,j,x;
        for(i=1;i<=tot;i++)
        {
            x=tp[i];
            st[x][0]=fa[x];
            for(j=1;j<20;j++)
            {
                st[x][j]=st[st[x][j-1]][j-1];
            }
        }
    }
    int get_substr(int l,int r)//init_pos init_ST
    {
        int now,tmp,i;
        now=pos[r];
        for(i=19;~i;i--)
        {
            tmp=st[now][i];
            if(tmp&&len[tmp]>=r-l+1) now=tmp;
        }
        return now;
    }
}sam;// sam.init();
```

1.5.4 回文自动机

```
struct Palindrome_Tree
{
    int len[MAX],nex[MAX][26],fail[MAX],last,s[MAX],
        tot,n;
    int cnt[MAX],deep[MAX];
    int newnode(int l)
    {
        mem(nex[tot],0);
        fail[tot]=0;
        deep[tot]=cnt[tot]=0;
        len[tot]=l;
        return tot++;
    }
    void init()
    {
        tot=n=last=0;
        newnode(0);
        newnode(-1);
        s[0]=-1;
        fail[0]=1;
    }
    int get_fail(int x)
    {
        while(s[n-len[x]-1]!=s[n]) x=fail[x];
        return x;
    }
    void add(int t)//attention the type of t is int
```

```
    {
        int id,now;
        s[++n]=t;
        now=get_fail(last);
        if(!nex[now][t])
        {
            id=newnode(len[now]+2);
            fail[id]=nex[get_fail(fail[now])][t];
            deep[id]=deep[fail[id]]+1;
            nex[now][t]=id;
        }
        last=nex[now][t];
        cnt[last]++;
    }
    void count()
    {
        for(int i=tot-1;i;i--) cnt[fail[i]]+=cnt[i];
    }
    void build_tree(VI mp[])// root is 0
    {
        for(int i=0;i<=tot+1;i++) mp[i].clear();
        for(int i=1;i<tot;i++) mp[fail[i]].pb(i);
    }
}pam; //pam.init();
```

1.5.5 序列自动机

```
int nex[MAX][26];
void work(char *s,int len)
{
    mem(nex[len],0);
    for(int i=len;i;i--)
    {
        for(int j=0;j<26;j++)
        {
            nex[i-1][j]=nex[i][j];
        }
        nex[i-1][s[i]-'a']=i;
    }
}
```

1.6 最小表示法

1.6.1 最小表示法

```
int min_representation(char *s)
{
    int i=0,j=1,k=0;
    int len=strlen(s);
    while(i<len&&j<len&&k<len)
    {
        if(s[(i+k)%len]==s[(j+k)%len])k++;
        else
        {
            if(s[(i+k)%len]>s[(j+k)%len])i=i+k+1;
            else j=j+k+1;
            if(i==j)j++;
            k=0;
        }
    }
    return i<j?i:j;
}
```

1.6.2 最大表示法

```
int max_representation(char *s)
{
    int i,j,k,len,t;
    len=strlen(s);
    i=k=0;
    j=1;
    while(i<len&&j<len&&k<len)
    {
        t=s[(i+k)%len]-s[(j+k)%len];
        if(!t) k++;
        else
        {
            if(t>0)
            {
                if(j+k+1>i) j=j+k+1;
                else j=i+1;
            }
            else if(i+k+1>j) i=i+k+1;
            else i=j+1;
            k=0;
        }
    }
    return i<j?i:j;
}
```

1.7 shift_and

```
void shift_and(char *s,char *t)//s[1..n],t[1..m] (n<=m)
{
    static const int SZ=26;
    int n,m,i;
    bitset<MAX> b[SZ],d;
    for(i=0;i<SZ;i++) b[i].reset();
    d.reset();
    n=strlen(s+1);
    m=strlen(t+1);
    for(i=1;i<=n;i++)
    {
        b[s[i]-'a'].set(i-1);//change
        //other matching character sets
```

```
    }
    for(i=1;i<=m;i++)
    {
        d<<=1;
        d.set(0);
        d&=(b[t[i]-'a']);//change
        if(d[n-1]==1)//successful match
        {

        }
    }
}
```

2 数据结构

2.1 RMQ

2.1.1 一维 RMQ

```
struct RMQ
{
    #define type int
    type v[MAX];
    int pmax(int a,int b){return v[a]>v[b]?a:b;}
    int pmin(int a,int b){return v[a]<v[b]?a:b;}
    int lg[MAX],bin[22];
    int pmx[MAX][22],pmn[MAX][22];
    type mx[MAX][22],mn[MAX][22];
    void work(int n,type *a)
    {
        int i,j;
        for(i=bin[0]=1;1<<(i-1)<=n;i++) bin[i]=(bin[i
            -1]<<1);
        for(i=2,lg[1]=0;i<=n;i++) lg[i]=lg[i>>1]+1;
        for(i=1;i<=n;i++)
        {
            v[i]=a[i];
            mx[i][0]=mn[i][0]=v[i];
            pmx[i][0]=pmn[i][0]=i;
        }
        for(j=1;1<<(j-1)<=n;j++)
        {
            for(i=1;i+bin[j]-1<=n;i++)
            {
                mx[i][j]=max(mx[i][j-1],mx[i+bin[j
                    -1]][j-1]);
                mn[i][j]=min(mn[i][j-1],mn[i+bin[j
                    -1]][j-1]);
                pmx[i][j]=pmax(pmx[i][j-1],pmx[i+bin[j
                    -1]][j-1]);
                pmn[i][j]=pmin(pmn[i][j-1],pmn[i+bin[j
                    -1]][j-1]);
            }
        }
    }
    type ask_max(int l,int r)
    {
        int t=lg[r-l+1];
        return max(mx[l][t],mx[r-bin[t]+1][t]);
    }
    type ask_min(int l,int r)
    {
        int t=lg[r-l+1];
        return min(mn[l][t],mn[r-bin[t]+1][t]);
    }
    int ask_pmax(int l,int r)
    {
        int t=lg[r-l+1];
        return pmax(pmx[l][t],pmx[r-bin[t]+1][t]);
    }
    int ask_pmin(int l,int r)
    {
        int t=lg[r-l+1];
        return pmin(pmn[l][t],pmn[r-bin[t]+1][t]);
    }
    #undef type
}rmq;
```

2.1.2 二维 RMQ

```
int v[302][302];
int maxx[302][302][9][9],minn[302][302][9][9];
void RMQ(int n,int m)
{
    int i,j,ii,jj;
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=m;j++)
        {
            maxx[i][j][0][0]=minn[i][j][0][0]=v[i][j
                ];
        }
    }
    for(ii=0;(1<<ii)<=n;ii++)
    {
        for(jj=0;(1<<jj)<=m;jj++)
        {
            if(!(ii+jj)) continue;
            for(i=1;i+(1<<ii)-1<=n;i++)
            {
                for(j=1;j+(1<<jj)-1<=m;j++)
                {
                    if(ii)
                    {
                        minn[i][j][ii][jj]=min(minn[i][
                            j][ii-1][jj],minn[i+(1<<(ii
                            -1))][j][ii-1][jj]);
```

```
                    maxx[i][j][ii][jj]=max(maxx[i][
                        j][ii-1][jj],maxx[i+(1<<(ii
                        -1))][j][ii-1][jj]);
                }
                else
                {
                    minn[i][j][ii][jj]=min(minn[i][
                        j][ii][jj-1],minn[i][j
                        +(1<<(jj-1))][ii][jj-1]);
                    maxx[i][j][ii][jj]=max(maxx[i][
                        j][ii][jj-1],maxx[i][j
                        +(1<<(jj-1))][ii][jj-1]);
                }
            }
        }
    }
}
int ask_max(int x1,int y1,int x2,int y2)
{
    int k1=0;
    while((1<<(k1+1))<=x2-x1+1) k1++;
    int k2=0;
    while((1<<(k2+1))<=y2-y1+1) k2++;
    x2=x2-(1<<k1)+1;
    y2=y2-(1<<k2)+1;
    return max(max(maxx[x1][y1][k1][k2],maxx[x1][y2
        ][k1][k2]),max(maxx[x2][y1][k1][k2],maxx[x2
        ][y2][k1][k2]))
}
int ask_min(int x1,int y1,int x2,int y2)
{
    int k1=0;
    while((1<<(k1+1))<=x2-x1+1) k1++;
    int k2=0;
    while((1<<(k2+1))<=y2-y1+1) k2++;
    x2=x2-(1<<k1)+1;
    y2=y2-(1<<k2)+1;
    return min(min(minn[x1][y1][k1][k2],minn[x1][y2
        ][k1][k2]),min(minn[x2][y1][k1][k2],minn[x2
        ][y2][k1][k2]));
}
```

2.2 单调队列

```
struct Monotone_queue
{
    #define type int
    type v[MAX][2];//0 is min, 1 is max
    int p[MAX][2];
    int l[2],r[2];
    void clear()
    {
        l[0]=r[0]=0;
        l[1]=r[1]=0;
    }
    void insert(type x,int pos)
    {
        while(r[0]-l[0]&&v[r[0]-1][0]>=x) r[0]--;
        v[r[0]][0]=x;
        p[r[0]++][0]=pos;
        while(r[1]-l[1]&&v[r[1]-1][1]<=x) r[1]--;
        v[r[1]][1]=x;
        p[r[1]++][1]=pos;
    }
    void erase(int pos)
    {
        while(r[0]-l[0]&&p[l[0]][0]<=pos) l[0]++;
        while(r[1]-l[1]&&p[l[1]][1]<=pos) l[1]++;
    }
    type get_min(){return v[l[0]][0];}
    type get_max(){return v[l[1]][1];}
    #undef type
}dq;
```

2.3 LCA

2.3.1 倍增 LCA

```
//O(nlogn)-O(logn)
struct LCA
{
    int fa[MAX][22],dep[MAX],n,limt,bin[22];
    VI mp[MAX];
    void init(int _n)
    {
        n=_n;
        for(limt=1;1<<(limt-1)<=n;limt++);
        for(int i=bin[0]=1;1<<(i-1)<=n;i++) bin[i]=(
            bin[i-1]<<1);
        for(int i=0;i<=n;i++)
        {
            mp[i].clear();
            mem(fa[i],0);
        }
    }
    void add_edge(int a,int b)
    {
        mp[a].pb(b);
        mp[b].pb(a);
    }
    void dfs(int x,int pre)
    {
        for(int i=1;bin[i]<=dep[x];i++) fa[x][i]=fa[
            fa[x][i-1]][i-1];
        for(int i=0;i<sz(mp[x]);i++)
        {
```

```
            int to=mp[x][i];
            if(to==pre) continue;
            dep[to]=dep[x]+1;
            fa[to][0]=x;
            dfs(to,x);
        }
    }
    void work(int rt)
    {
        dep[rt]=0;
        dfs(rt,0);
    }
    int go(int x,int d)
    {
        for(int i=0;i<=limt&&d;i++)
        {
            if(bin[i]&d)
            {
                d^=bin[i];
                x=fa[x][i];
            }
        }
        return x;
    }
    int lca(int a,int b)
    {
        if(dep[a]<dep[b]) swap(a,b);
        a=go(a,dep[a]-dep[b]);
        if(a==b) return a;
        for(int i=limt;~i;i--)
        {
            if(fa[a][i]!=fa[b][i])
            {
                a=fa[a][i];
                b=fa[b][i];
            }
        }
        return fa[a][0];
    }
}lca;
```

2.3.2 RMQ 维护欧拉序求 LCA

```
// O(nlogn)-O(1)
struct LCA
{
    #define type int
    struct node{int to;type w;node(){}node(int _to,
        type _w):to(_to),w(_w){}};
    type dis[MAX];
    int path[2*MAX],deep[2*MAX],first[MAX],len[MAX],
        tot,n;
    int dp[2*MAX][22];
    vector<node> mp[MAX];
    void dfs(int x,int pre,int h)
    {
        int i;
        path[++tot]=x;
        first[x]=tot;
        deep[tot]=h;
        for(i=0;i<mp[x].size();i++)
        {
            int to=mp[x][i].to;
            if(to==pre) continue;
            dis[to]=dis[x]+mp[x][i].w;
            len[to]=len[x]+1;
            dfs(to,x,h+1);
            path[++tot]=x;
            deep[tot]=h;
        }
    }
    void ST(int n)
    {
        int i,j,x,y;
        for(i=1;i<=n;i++) dp[i][0]=i;
        for(j=1;(1<<j)<=n;j++)
        {
            for(i=1;i+(1<<j)-1<=n;i++)
            {
                x=dp[i][j-1];
                y=dp[i+(1<<(j-1))][j-1];
                dp[i][j]=deep[x]<deep[y]?x:y;
            }
        }
    }
    int query(int l,int r)
    {
        int len,x,y;
        len=(int)log2(r-l+1);
        x=dp[l][len];
        y=dp[r-(1<<len)+1][len];
        return deep[x]<deep[y]?x:y;
    }
    int lca(int x,int y)
    {
        int l,r,pos;
        l=first[x];
        r=first[y];
        if(l>r) swap(l,r);
        pos=query(l,r);
        return path[pos];
    }
    type get_dis(int a,int b){return dis[a]+dis[b
        ]-2*dis[lca(a,b)];}
    int get_len(int a,int b){return len[a]+len[b]-2*
        len[lca(a,b)];}
    void init(int _n)
    {
```

```
        n=_n;
        for(int i=0;i<=n;i++)
        {
            dis[i]=0;
            len[i]=0;
            mp[i].clear();
        }
    }
    void add_edge(int a,int b,type w=1)
    {
        mp[a].pb(node(b,w));
        mp[b].pb(node(a,w));
    }
    void work(int rt)
    {
        tot=0;
        dfs(rt,0,0);
        ST(2*n-1);
    }
    int lca_root(int rt,int a,int b)
    {
        int fa,fb;
        fa=lca(a,rt);
        fb=lca(b,rt);
        if(fa==fb) return lca(a,b);
        else
        {
            if(get_dis(fa,rt)<get_dis(fb,rt)) return
                fa;
            else return fb;
        }
    }
    #undef type
}lca;
/*
lca.init(n);
lca.add_edge(a,b,w) undirected edge.
lca.work(rt);
*/
```

2.4 轻重链剖分

size[] 数组，以 x 为根的子树节点个数。

top[] 数组，当前节点的所在链的顶端节点。

son[] 数组，重儿子。

deep[] 数组，当前节点的深度。

fa[] 数组，当前节点的父亲。

idx[] 数组，树中每个节点剖分后的新编号。

rnk[] 数组，idx 的逆，表示线段上中当前位置表示哪个节点。

```
struct HLD
{
    #define type int
    struct edge{int a,b;type v;edge(int _a,int _b,
        type _v=0):a(_a),b(_b),v(_v){}};
    struct node{int to;type w;node(){}node(int _to,
        type _w):to(_to),w(_w){}};
    vector<int> mp[MAX];
    vector<edge> e;
    int deep[MAX],fa[MAX],size[MAX],son[MAX];
    int rnk[MAX],top[MAX],idx[MAX],tot;
    int n,rt;
    void init(int _n)
    {
        n=_n;
        for(int i=0;i<=n;i++) mp[i].clear();
        e.clear();
        e.pb(edge(0,0));
    }
    void add_edge(int a,int b,type v=0)
    {
        e.pb(edge(a,b,v));
        mp[a].pb(b);
        mp[b].pb(a);
    }
    void dfs1(int x,int pre,int h)
    {
        int i,to;
        deep[x]=h;
        fa[x]=pre;
        size[x]=1;
        for(i=0;i<sz(mp[x]);i++)
        {
            to=mp[x][i];
            if(to==pre) continue;
            dfs1(to,x,h+1);
            size[x]+=size[to];
            if(son[x]==-1||size[to]>size[son[x]]) son
                [x]=to;
        }
    }
    void dfs2(int x,int tp)
    {
        int i,to;
        top[x]=tp;
        idx[x]=++tot;
        rnk[idx[x]]=x;
        if(son[x]==-1) return;
        dfs2(son[x],tp);
        for(i=0;i<sz(mp[x]);i++)
        {
            to=mp[x][i];
            if(to!=son[x]&&to!=fa[x]) dfs2(to,to);
```

```
    }
}
void work(int _rt)
{
    int i;
    rt=_rt;
    mem(son,-1);
    tot=0;
    dfs1(rt,0,0);
    dfs2(rt,rt);
}
int LCA(int x,int y)
{
    while(top[x]!=top[y])
    {
        if(deep[top[x]]<deep[top[y]]) swap(x,y);
        x=fa[top[x]];
    }
    if(deep[x]>deep[y]) swap(x,y);
    return x;
}
//node
void init_node()
{
    build(n);
}
void modify_node(int x,int y,type val)
{
    while(top[x]!=top[y])
    {
        if(deep[top[x]]<deep[top[y]]) swap(x,y);
        update(idx[top[x]],idx[x],val);
        x=fa[top[x]];
    }
    if(deep[x]>deep[y]) swap(x,y);
    update(idx[x],idx[y],val);
}
type query_node(int x,int y)
{
    type res=0;
    while(top[x]!=top[y])
    {
        if(deep[top[x]]<deep[top[y]]) swap(x,y);
        res+=query(idx[top[x]],idx[x]);
        x=fa[top[x]];
    }
    if(deep[x]>deep[y]) swap(x,y);
    res+=query(idx[x],idx[y]);
    return res;
}
//path
void init_path()
{
    v[idx[rt]]=0;
    for(int i=1;i<n;i++)
    {
        if(deep[e[i].a]<deep[e[i].b]) swap(e[i].a
            ,e[i].b);
        v[idx[e[i].a]]=e[i].v;
    }
    build(n);
}
void modify_edge(int id,type val)
{
    if(deep[e[id].a]>deep[e[id].b]) update(idx[e[
        id].a],idx[e[id].a],val);
    else update(idx[e[id].b],idx[e[id].b],val);
}
void modify_path(int x,int y,type val)
{
    while(top[x]!=top[y])
    {
        if(deep[top[x]]<deep[top[y]]) swap(x,y);
        update(idx[top[x]],idx[x],val);
        x=fa[top[x]];
    }
    if(deep[x]>deep[y]) swap(x,y);
    if(x!=y) update(idx[x]+1,idx[y],val);
}
type query_path(int x,int y)
{
    type res=0;
    while(top[x]!=top[y])
    {
        if(deep[top[x]]<deep[top[y]]) swap(x,y);
        res+=query(idx[top[x]],idx[x]);
        x=fa[top[x]];
    }
    if(deep[x]>deep[y]) swap(x,y);
    if(x!=y) res+=query(idx[x]+1,idx[y]);
    return res;
}
#undef type
}hld;
/***********attention!************/
//hld.init(n)
//hld.add_edge(): undirected edge.
/*********************************/
```

2.5 并查集

2.5.1 并查集

```
struct dsu
{
    int pre[MAX];
    void init(int n)
    {
```

```
        int i;
        for(i=1;i<=n;i++) pre[i]=i;
    }
    int find(int x)
    {
        if(pre[x]!=x) pre[x]=find(pre[x]);
        return pre[x];
    }
    bool merge(int a,int b)
    {
        int ra,rb;
        ra=find(a);
        rb=find(b);
        if(ra!=rb)
        {
            pre[ra]=rb;
            return 1;
        }
        return 0;
    }
}dsu;
```

2.5.2 map 实现并查集

```
struct dsu
{
    unordered_map<int,int> pre;
    void init(){pre.clear();}
    int find(int x)
    {
        if(pre.count(x)) pre[x]=find(pre[x]);
        else return x;
        return pre[x];
    }
    bool merge(int a,int b)
    {
        int ra,rb;
        ra=find(a);
        rb=find(b);
        if(ra!=rb)
        {
            pre[ra]=rb;
            return 1;
        }
        return 0;
    }
}dsu;
```

2.5.3 可撤销并查集

```
struct dsu
{
    PII st[MAX];
    int pre[MAX],top,sz[MAX];
    void init(int n)
    {
        int i;
        for(i=1;i<=n;i++)
        {
            pre[i]=i;
            sz[i]=1;
        }
        top=0;
    }
    int find(int x)
    {
        while(x!=pre[x]) x=pre[x];
        return x;
    }
    bool merge(int a,int b)
    {
        int ra,rb;
        ra=find(a);
        rb=find(b);
        if(ra==rb) return 0;
        if(sz[ra]>sz[rb]) swap(ra,rb);
        pre[ra]=rb;
        sz[rb]+=sz[ra];
        st[top++]=MP(ra,rb);
        return 1;
    }
    void roll_back()
    {
        PII now=st[--top];
        pre[now.fi]=now.fi;
        sz[now.se]-=sz[now.fi];
    }
}dsu;
```

2.6 树状数组

2.6.1 一维单点 BIT

```
struct Fenwick_Tree
{
    #define type int
    type bit[MAX];
    int n;
    void init(int _n)
    {
        n=_n;
        for(int i=0;i<=n;i++) bit[i]=0;
    }
    int lowbit(int x){return x&(-x);}
    void insert(int x,type v)
    {
        while(x<=n)
```

```
        {
            bit[x]+=v;
            x+=lowbit(x);
        }
    }
    type get(int x)
    {
        type res=0;
        while(x)
        {
            res+=bit[x];
            x-=lowbit(x);
        }
        return res;
    }
    type ask(int l,int r)
    {
        if(l-1<=0) return get(r);
        return get(r)-get(l-1);
    }
    #undef type
}tr;
```

2.6.2 一维区间 BIT

```
struct Fenwick_Tree
{
    #define type int
    type bit[MAX][2];
    int n;
    void init(int _n)
    {
        n=_n;
        for(int i=0;i<=n;i++)
        {
            bit[i][0]=bit[i][1]=0;
        }
    }
    int lowbit(int x){return x&(-x);}
    //□□□□□□□²»_insert
    void _insert(int x,type v)
    {
        for(int i=x;i<=n;i+=lowbit(i))
        {
            bit[i][0]+=v;
            bit[i][1]+=v*(x-1);
        }
    }
    void upd(int l,int r,type v)
    {
        _insert(l,v);
        _insert(r+1,-v);
    }
    type get(int x)
    {
        type res=0;
        for(int i=x;i;i-=lowbit(i))
        {
            res+=x*bit[i][0]-bit[i][1];
        }
        return res;
    }
    type ask(int l,int r)
    {
        if(l-1<=0) return get(r);
        return get(r)-get(l-1);
    }
    #undef type
}tr;
```

2.6.3 二维单点 BIT

```
struct Fenwick_Tree
{
    #define type int
    type bit[MAX][MAX];
    int n,m;
    void init(int _n,int _m){n=_n;m=_m;mem(bit,0);}
    int lowbit(int x){return x&(-x);}
    void update(int x,int y,type v)
    {
        int i,j;
        for(i=x;i<=n;i+=lowbit(i))
        {
            for(j=y;j<=m;j+=lowbit(j))
            {
                bit[i][j]+=v;
            }
        }
    }
    type get(int x,int y)
    {
        type i,j,res=0;
        for(i=x;i>0;i-=lowbit(i))
        {
            for(j=y;j>0;j-=lowbit(j))
            {
                res+=bit[i][j];
            }
        }
        return res;
    }
    type ask(int x1,int x2,int y1,int y2)
    {
        x1--;
        y1--;
        return get(x2,y2)-get(x1,y2)-get(x2,y1)+get(
            x1,y1);
```

```
    }
    #undef type
}tr;
```

2.7 线段树

2.7.1 线段树

```
struct Segment_Tree
{
    #define type int
    #define ls (id<<1)
    #define rs (id<<1|1)
    int n,ql,qr;
    type a[MAX],v[MAX<<2],tag[MAX<<2],qv;
    void pushup(int id)
    {

    }
    void pushdown(int id)
    {
        if(!tag[id]) return;

    }
    void build(int l,int r,int id)
    {
        tag[id]=0;
        if(l==r)
        {
            //init
            return;
        }
        int mid=(l+r)>>1;
        build(l,mid,ls);
        build(mid+1,r,rs);
        pushup(id);
    }
    void update(int l,int r,int id)
    {
        if(l>=ql&&r<=qr)
        {
            //do something
            return;
        }
        pushdown(id);
        int mid=(l+r)>>1;
        if(ql<=mid) update(l,mid,ls);
        if(qr>mid) update(mid+1,r,rs);
        pushup(id);
    }
    type res;
    void query(int l,int r,int id)
    {
        if(l>=ql&&r<=qr)
        {
            //do something
            return;
        }
        pushdown(id);
        int mid=(l+r)>>1;
        if(ql<=mid) query(l,mid,ls);
        if(qr>mid) query(mid+1,r,rs);
    }
    void build(int _n){n=_n;build(1,n,1);}
    void upd(int l,int r,type v)
    {
        ql=l;
        qr=r;
        qv=v;
        update(1,n,1);
    }
    type ask(int l,int r)//init res
    {
        ql=l;
        qr=r;
        res=0;
        query(1,n,1);
        return res;
    }
    #undef type
    #undef ls
    #undef rs
}tr;
```

2.7.2 动态开点线段树

```
//空间大小是nlogm,为插入的节点总数n,为区间长度m
struct Segment_Tree
{
    #define type int
    int root,tot,ls[MAX*20],rs[MAX*20],ql,qr,n;
    type v[MAX*20],tag[MAX*20],qv;
    int newnode()
    {
        ls[tot]=rs[tot]=0;
        v[tot]=0;
        tag[tot]=-1;
        return tot++;
    }
    void init(int _n)
    {
        n=_n;
        tot=0;
        root=newnode();
    }
    void pushup(int id)
    {

```

```
    }
    void pushdown(int id)
    {
        if(tag[id]==-1) return;
        if(!ls[id]) ls[id]=newnode();
        if(!rs[id]) rs[id]=newnode();


        tag[id]=-1;
    }
    void update(int l,int r,int &id)
    {
        if(!id) id=newnode();
        if(l>=ql&&r<=qr)
        {
            //do something
            return;
        }
        pushdown(id);
        int mid=(l+r)>>1;
        if(ql<=mid) update(l,mid,ls[id]);
        if(qr>mid) update(mid+1,r,rs[id]);
        pushup(id);
    }
    type res;
    void query(int l,int r,int &id)
    {
        if(!id) return;
        if(l>=ql&&r<=qr)
        {
            //do something
            return;
        }
        pushdown(id);
        int mid=(l+r)>>1;
        if(ql<=mid) query(l,mid,ls[id]);
        if(qr>mid) query(mid+1,r,rs[id]);
    }
    void upd(int l,int r,type v)
    {
        ql=l;
        qr=r;
        qv=v;
        update(1,n,root);
    }
    type ask(int l,int r)//init res
    {
        ql=l;
        qr=r;
        res=0;
        query(1,n,root);
        return res;
    }
    #undef type
};
```

2.7.3 线段树套线段树

```
struct Segment_Tree_2D
{
    #define type int
    static const int insert_num=;
    static const int N=insert_num*20*20; //
        insert_num*20*log(m)
    int root[MAX<<2],tot,ls[N],rs[N],n,m;
    int ql_in,qr_in,ql_out,qr_out;
    type v[N],qv,tag[N];
    void init(int _n,int _m)
    {
        n=_n;
        m=_m;
        mem(root,0);
        ls[0]=rs[0]=0;
        tag[0]=0;
        v[0]=0;
        tot=1;
    }
    int newnode()
    {
        ls[tot]=rs[tot]=0;
        v[tot]=0;
        tag[tot]=0;
        return tot++;
    }
    void pushup(int id)
    {

    }
    void pushdown(int id)
    {
        if(!tag[id]) return;
        if(!ls[id]) ls[id]=newnode();
        if(!rs[id]) rs[id]=newnode();


        tag[id]=0;
    }
    void update_in(int l,int r,int &id)
    {
        if(!id) id=newnode();
        if(l>=ql_in&&r<=qr_in)
        {
            v[id]+=qv; //must update not =
            tag[id]+=qv;
            return;
        }
        pushdown(id);
        int mid=(l+r)>>1;
```

```
    if(ql_in<=mid) update_in(l,mid,ls[id]);
    if(qr_in>mid) update_in(mid+1,r,rs[id]);
    pushup(id);
}
type res_in;
void query_in(int l,int r,int &id)
{
    if(!id) return;
    if(l>=ql_in&&r<=qr_in)
    {
        res_in+=v[id];
        return;
    }
    pushdown(id);
    int mid=(l+r)>>1;
    type res=0;
    if(ql_in<=mid) query_in(l,mid,ls[id]);
    if(qr_in>mid) query_in(mid+1,r,rs[id]);
}
/*
 ********************************************
 */
#define ls (id<<1)
#define rs (id<<1|1)
void update_out(int l,int r,int id)
{
    update_in(1,m,root[id]);
    if(l>=ql_out&&r<=qr_out) return;
    int mid=(l+r)>>1;
    if(ql_out<=mid) update_out(l,mid,ls);
    if(qr_out>mid) update_out(mid+1,r,rs);
}
type res_out;
void query_out(int l,int r,int id)
{
    if(l>=ql_out&&r<=qr_out)
    {
        res_in=0;
        query_in(1,m,root[id]);
        res_out+=res_in;
        return;
    }
    int mid=(l+r)>>1;
    if(ql_out<=mid) query_out(l,mid,ls);
    if(qr_out>mid) query_out(mid+1,r,rs);
}
#undef ls
#undef rs
void upd(int x1,int y1,int x2,int y2,type val)
{
    ql_out=x1;
    qr_out=x2;
    ql_in=y1;
    qr_in=y2;
    qv=val;
    update_out(1,n,1);
}
type ask(int x1,int y1,int x2,int y2)
{
    ql_out=x1;
    qr_out=x2;
    ql_in=y1;
    qr_in=y2;
    res_out=0;
    query_out(1,n,1);
    return res_out;
}
#undef type
}tr2d;
```

2.7.4 线段树分裂合并

```
struct Segment_Tree
{
    #define type int
    int s[MAX*20],top;
    int root[MAX],tot,ls[MAX*20],rs[MAX*20],ql,qr,n;
    type v[MAX*20],tag[MAX*20],qv;
    void init()
    {
        top=0;
        mem(root,0);
        ls[0]=rs[0]=0;
        v[0]=0;
        tot=1;
    }
    int newnode()
    {
        int t;
        if(top) t=s[--top];
        else t=tot++;
        ls[t]=rs[t]=0;
        v[t]=0;
        return t;
    }
    void delnode(int x)
    {
        s[top++]=x;
    }
    void pushup(int id)
    {
        v[id]=v[ls[id]]+v[rs[id]];
    }
    void pushdown(int id)
    {
        if(tag[id]==-1) return;
        if(!ls[id]) ls[id]=newnode();
        if(!rs[id]) rs[id]=newnode();
```

```

    tag[id]=-1;
}
int split(int l,int r,int &id)
{
    if(!id) return 0;
    if(ql<=l&&r<=qr)
    {
        int temp=id;
        id=0;
        return temp;
    }
    int t=newnode();
    int mid=(l+r)>>1;
    if(ql<=mid) ls[t]=split(l,mid,ls[id]);
    if(qr>mid) rs[t]=split(mid+1,r,rs[id]);
    pushup(t);
    pushup(id);
    return t;
}
int merge(int a,int b)
{
    if(!a||!b) return a+b;
    ls[a]=merge(ls[a],ls[b]);
    rs[a]=merge(rs[a],rs[b]);
    if(!ls[a]&&!rs[a])
    {
        v[a]+=v[b];//merge a,b to b
    }
    else
    {
        pushup(a);
        //do something
    }
    delnode(b);
    return a;
}
void update(int l,int r,int &id)
{
    if(!id) id=newnode();
    if(l>=ql&&r<=qr)
    {
        v[id]=(r-l+1)*qv;
        tag[id]=qv;
        return;
    }
    pushdown(id);
    int mid=(l+r)>>1;
    if(ql<=mid) update(l,mid,ls[id]);
    if(qr>mid) update(mid+1,r,rs[id]);
    pushup(id);
}
type query(int l,int r,int &id)
{
    if(!id) return 0;
    if(l>=ql&&r<=qr) return v[id];
    int mid=(l+r)>>1;
    type res=0;
    if(ql<=mid) res+=query(l,mid,ls[id]);
    if(qr>mid) res+=query(mid+1,r,rs[id]);
    return res;
}
#undef type
}tr;
```

2.8 主席树

```
struct president_tree
{
    #define type int
    int root[MAX],ls[40*MAX],rs[40*MAX],tot,ql,qr;
    type sum[40*MAX],qv;
    void init()
    {
        mem(root,0);
        tot=1;
        ls[0]=rs[0]=sum[0]=0;
    }
    int newnode(int x)
    {
        ls[tot]=ls[x];
        rs[tot]=rs[x];
        sum[tot]=sum[x];
        return tot++;
    }
    void insert(int l,int r,int &id,int pre) //set(ql,ql,v)
    {
        id=newnode(pre);
        sum[id]+=qv;
        if(l==r) return;
        int mid=(l+r)>>1;
        if(ql<=mid) insert(l,mid,ls[id],ls[pre]);
        else insert(mid+1,r,rs[id],rs[pre]);
    }
    int kindcnt(int l,int r,int id) //set(ql,qr)
    {
        if(ql<=l&&r<=qr) return sum[id];
        int mid=(l+r)>>1;
        int res=0;
        if(ql<=mid) res+=kindcnt(l,mid,ls[id]);
        if(qr>=mid+1) res+=kindcnt(mid+1,r,rs[id]);
        return res;
    }
    int kthsmall(int l,int r,int id,int pre,int k)
    {
```

```
        if(l==r) return l;
        int mid=(l+r)>>1;
        int temp=sum[ls[id]]-sum[ls[pre]];
        if(temp>=k) return kthsmall(l,mid,ls[id],ls[
            pre],k);
        else return kthsmall(mid+1,r,rs[id],rs[pre],k
            -temp);
    }
    int kthbig(int l,int r,int id,int pre,int k)
    {
        if(l==r) return l;
        int mid=(l+r)>>1;
        int temp=sum[rs[id]]-sum[rs[pre]];
        if(temp>=k) return kthbig(mid+1,r,rs[id],rs[
            pre],k);
        else return kthbig(l,mid,ls[id],ls[pre],k-
            temp);
    }
    void set(int l,int r,type v=0){ql=l;qr=r;qv=v;}
}pt;
```

2.9 李超树

```
struct LiChao_Segment_Tree
{
    #define type ll
    #define inf -LLINF
    #define ls (id<<1)
    #define rs (id<<1|1)
    #define cmp max
    struct line
    {
        type k,b;
        void init(type _k=0,type _b=0){k=_k;b=_b;}
    }sg[MAX<<2];
    type v[MAX<<2];
    bool ext[MAX<<2];
    int ql,qr,n;
    type cal(line l,int x){return l.k*x+l.b;}
    void pushup(int id)
    {
        v[id]=cmp(v[id],v[ls]);
        v[id]=cmp(v[id],v[rs]);
    }
    void build(int l,int r,int id)
    {
        ext[id]=0;
        sg[id].init();
        v[id]=inf;
        if(l==r) return;
        int mid=(l+r)>>1;
        build(l,mid,ls);
        build(mid+1,r,rs);
    }
    void update(int l,int r,int id,line qv)
    {
        if(l>=ql&&r<=qr)
        {
            if(!ext[id])
            {
                ext[id]=1;
                sg[id]=qv;
            }
            else if(cal(qv,l)>=cal(sg[id],l)&&cal(qv,
                r)>=cal(sg[id],r)) sg[id]=qv;
            else if(cal(qv,l)>cal(sg[id],l)||cal(qv,r
                )>cal(sg[id],r))
            {
                int mid=(l+r)>>1;
                if(cal(qv,mid)>cal(sg[id],mid)) swap(
                    qv,sg[id]);
                if(cal(qv,l)>cal(sg[id],l)) update(l,
                    mid,ls,qv);
                else update(mid+1,r,rs,qv);
            }
            v[id]=cmp(cal(sg[id],l),cal(sg[id],r));
            if(l!=r) pushup(id);
            //if ask min, change '>' to '<'
        }
        else
        {
            int mid=(l+r)>>1;
            if(ql<=mid) update(l,mid,ls,qv);
            if(qr>mid) update(mid+1,r,rs,qv);
            v[id]=cmp(cal(sg[id],l),cal(sg[id],r));
            if(l!=r) pushup(id);
        }
    }
    type res;
    void query(int l,int r,int id)
    {
        if(l>=ql&&r<=qr)
        {
            res=cmp(res,v[id]);
            return;
        }
        res=cmp(res,cal(sg[id],max(l,ql)));
        res=cmp(res,cal(sg[id],min(r,qr)));
        int mid=(l+r)>>1;
        if(ql<=mid) query(l,mid,ls);
        if(qr>mid) query(mid+1,r,rs);
    }
    void build(int _n){n=_n;build(1,n,1);}
    void upd(int l,int r,type k,type b)
    {
        ql=l;
        qr=r;
```

```
        line qv;
        qv.init(k,b);
        update(1,n,1,qv);
    }
    type ask(int l,int r)
    {
        ql=l;
        qr=r;
        res=inf;
        query(1,n,1);
        return res;
    }
    #undef type
    #undef ls
    #undef rs
    #undef cmp
    #undef inf
}tr;
```

2.10 平衡树

2.10.1 Treap

```
struct Treap
{
    #define type ll
    struct node
    {
        int ch[2],fix,sz,w;
        type v;
        node(){}
        node(type x)
        {
            v=x;
            fix=rand();
            sz=w=1;
            ch[0]=ch[1]=0;
        }
    }t[MAX];
    int tot,root,tmp;
    void init()
    {
        srand(unsigned(new char));
        root=tot=0;
        t[0].sz=t[0].w=0;
        mem(t[0].ch,0);
    }
    inline void maintain(int k)
    {
        t[k].sz=t[t[k].ch[0]].sz+t[t[k].ch[1]].sz+t[k
            ].w ;
    }
    inline void rotate(int &id,int k)
    {
        int y=t[id].ch[k^1];
        t[id].ch[k^1]=t[y].ch[k];
        t[y].ch[k]=id;
        maintain(id);
        maintain(y);
        id=y;
    }
    void insert(int &id,type v)
    {
        if(!id) t[id=++tot]=node(v);
        else
        {
            if(t[id].sz++,t[id].v==v)t[id].w++;
            else if(insert(t[id].ch[tmp=v>t[id].v],v)
                ,t[t[id].ch[tmp]].fix>t[id].fix)
                rotate(id,tmp^1);
        }
    }
    void erase(int &id,type v)
    {
        if(!id)return;
        if(t[id].v==v)
        {
            if(t[id].w>1) t[id].w--,t[id].sz--;
            else
            {
                if(!(t[id].ch[0]&&t[id].ch[1])) id=t[
                    id].ch[0]|t[id].ch[1];
                else
                {
                    rotate(id,tmp=t[t[id].ch[0]].fix>t[
                        t[id].ch[1]].fix);
                    t[id].sz--;
                    erase(t[id].ch[tmp],v);
                }
            }
        }
        else
        {
            t[id].sz--;
            erase(t[id].ch[v>t[id].v],v);
        }
    }
    type kth(int k)//k small
    {
        int id=root;
        if(id==0) return 0;
        while(id)
        {
            if(t[t[id].ch[0]].sz>=k) id=t[id].ch[0];
            else if(t[t[id].ch[0]].sz+t[id].w>=k)
                return t[id].v;
            else
            {
```

```
            k-=t[t[id].ch[0]].sz+t[id].w;
            id=t[id].ch[1];
        }
    }
}
int find(type key,int f)
{
    int id=root,res=0;
    while(id)
    {
        if(t[id].v<=key)
        {
            res+=t[t[id].ch[0]].sz+t[id].w;
            if(f&&key==t[id].v) res-=t[id].w;
            id=t[id].ch[1];
        }
        else id=t[id].ch[0];
    }
    return res;
}
type find_pre(type key)
{
    type res=-LLINF;
    int id=root;
    while(id)
    {
        if(t[id].v<key)
        {
            res=max(res,t[id].v);
            id=t[id].ch[1];
        }
        else id=t[id].ch[0];
    }
    return res;
}
type find_suc(type key)
{
    type res=LLINF;
    int id=root;
    while(id)
    {
        if(t[id].v>key)
        {
            res=min(res,t[id].v);
            id=t[id].ch[0];
        }
        else id=t[id].ch[1];
    }
    return res;
}
void insert(type v){insert(root,v);}
void erase(type v){erase(root,v);}
int upper_bound_count(type key){return find(key
    ,0);}//the count >=key
int lower_bound_count(type key){return find(key
    ,1);}//the count >key
int rank(type key){return lower_bound_count(key)
    +1;}
#undef type
}t; //t.init();
```

2.11 字典树

2.11.1 trie

```
struct Trie
{
    #define type int
    struct trie
    {
        int v;
        trie *next[26];
        trie()
        {
            v=0;
            for(int i=0;i<26;i++) next[i]=NULL;
        }
    }*root;
    void insert(trie *p,char *s)
    {
        int i=0,t;
        while(s[i])
        {
            t=s[i]-'a';
            if(p->next[t]==NULL) p->next[t]=new trie;
            p=p->next[t];
            p->v++; //may need change
            i++;
        }
    }
    int find(trie *p,char *s)
    {
        int i=0,t;
        while(s[i])
        {
            t=s[i]-'a';
            if(p->next[t]==NULL) return 0;
            p=p->next[t];
            i++;
        }
        return p->v; //may need change
    }
    //r□□□□³sμ´®
    void del(char *s)
    {
        int i=0,t,temp;
        trie *p,*pre;
        pre=p=root;
```

```
        while(s[i])
        {
            t=s[i]-'a';
            if(p->next[t]==NULL) return;
            if(!s[i+1])
            {
                temp=p->next[t]->v;
                p->next[t]=NULL;
                break;
            }
            pre=p;
            p=p->next[t];
            i++;
        }
        i=0;
        p=root;
        while(s[i])
        {
            t=s[i]-'a';
            if(p->next[t]==NULL) return;
            p=p->next[t];
            p->v-=temp;
            i++;
        }
    }
    #undef type
}tr;
```

2.11.2 01trie

```
struct Trie
{
    #define type int
    static const int mx=30;
    int root,tot,nex[MAX*mx][2];
    type cnt[MAX*mx];
    int newnode()
    {
        mem(nex[tot],0);
        cnt[tot]=0;
        return tot++;
    }
    void init()
    {
        mem(nex[0],0);
        cnt[0]=0;
        tot=1;
        root=newnode();
    }
    void upd(type x,type v)
    {
        int id,t,i;
        id=root;
        for(i=mx;~i;i--)
        {
            t=(x>>i)&1;
            if(!nex[id][t]) nex[id][t]=newnode();
            id=nex[id][t];
            cnt[id]+=v;
        }
    }
    type count(int x)
    {
        int id,t,i;
        id=root;
        for(i=mx;~i;i--)
        {
            t=(x>>i)&1;
            if(!nex[id][t]) return 0;
            id=nex[id][t];
        }
        return cnt[id];
    }
    type ask_max(type x)
    {
        int id,t,i;
        type res;
        id=root;
        res=0;
        for(i=mx;~i;i--)
        {
            t=(x>>i)&1;
            if(nex[id][t^1]&&cnt[nex[id][t^1]]) t^=1;
            res|=(t<<i);
            id=nex[id][t];
        }
        return res;
    }
    type ask_min(type x)
    {
        int id,t,i;
        type res;
        id=root;
        res=0;
        for(i=mx;~i;i--)
        {
            t=(x>>i)&1;
            if(!nex[id][t]||!cnt[nex[id][t]]) t^=1;
            res|=(t<<i);
            id=nex[id][t];
        }
        return res;
    }
    #undef type
}tr;
```

2.12 kd-tree

```
namespace kd_tree
{
    const double alpha=0.75;
    const int dim=2;
    #define type int
    const type NONE=INF; //初始值
    struct kdtnode
    {
        bool exist;
        int l,r,sz,fa,dep,x[dim],mx[dim],mn[dim];
        type v,tag;
        kdtnode(){}
        void initval()
        {
            sz=exist;tag=v;
            if(exist) for(int i=0;i<dim;i++) mn[i]=mx[i]=x[i];
        }
        void null()
        {
            exist=sz=0;
            v=tag=NONE;
            for(int i=0;i<dim;i++)
            {
                mx[i]=-INF;
                mn[i]=INF;
            }
        }
        void newnode(int x0,int x1,type val=NONE)
        {
            x[0]=x0;
            x[1]=x1;
            l=r=fa=0;
            exist=1;
            v=val;
            initval();
        }
        kdtnode(int a,int b,type d=NONE){newnode(a,b,d);}
    };
    struct KDT
    {
        #define ls t[id].l
        #define rs t[id].r
        kdtnode t[MAX];
        int tot,idx,root;
        inline void pushup(int id)
        {
            t[id].initval();
            t[id].sz+=t[ls].sz+t[rs].sz;
            t[id].tag=min({t[ls].tag,t[rs].tag,t[id].tag});
            for(int i=0;i<dim;i++)
            {
                if(ls)
                {
                    t[id].mx[i]=max(t[id].mx[i],t[ls].mx[i]);
                    t[id].mn[i]=min(t[id].mn[i],t[ls].mn[i]);
                }
                if(rs)
                {
                    t[id].mx[i]=max(t[id].mx[i],t[rs].mx[i]);
                    t[id].mn[i]=min(t[id].mn[i],t[rs].mn[i]);
                }
            }
        }
        bool isbad(int id){return t[id].sz*alpha+3<max(t[ls].sz,t[rs].sz);}
        int st[MAX],top;
        void build(int &id,int l,int r,int fa,int dep=0)
        {
            id=0;if(l>r) return;
            int m=(l+r)>>1; idx=dep;
            nth_element(st+l,st+m,st+r+1,[&](int x,int y){return t[x].x[idx]<t[y].x[idx];});
            id=st[m];
            build(ls,l,m-1,id,(dep+1)%dim);
            build(rs,m+1,r,id,(dep+1)%dim);
            pushup(id);
            t[id].dep=dep;
            t[id].fa=fa;
        }
        inline void init(int n=0)
        {
            root=0;
            t[0].null();
            for(int i=1;i<=n;i++) st[i]=i;
            if(n) build(root,1,n,0);
            tot=n;
        }
        void travel(int id)
        {
            if(!id) return;
            if(t[id].exist) st[++top]=id;
            travel(ls);
            travel(rs);
        }
        void rebuild(int &id,int dep)
        {
            top=0;travel(id);
            build(id,1,top,t[id].fa,dep);
```

```
    }
    void insert(int &id,int now,int fa,int dep=0)
    {
        if(!id)
        {
            id=now;
            t[id].dep=dep;
            t[id].fa=fa;
            return;
        }
        if(t[now].x[dep]<t[id].x[dep]) insert(ls,
            now,id,(dep+1)%dim);
        else insert(rs,now,id,(dep+1)%dim);
        pushup(id);
        if(isbad(id)) rebuild(id,t[id].dep);
        t[id].dep=dep;
        t[id].fa=fa;
    }
    inline void insert(kdtnode x){t[++tot]=x;
        insert(root,tot,0,0);}
    inline void del(int id)
    {
        if(!id) return;
        t[id].null();
        int x=id;
        while(x)
        {
            pushup(x);
            x=t[x].fa;
        }
        if(isbad(id))
        {
            x=t[id].fa;
            rebuild(root==id?root:(t[x].l==id?t[x
                ].l:t[x].r),t[id].dep);
        }
    }
    kdtnode q;
    ll dist(ll x,ll y){return x*x+y*y;}
    ll getdist(int id)//点离区域qt[id最短距离]
    {
        if(!id) return LLINF;
        ll res=0;
        if(q.x[0]<t[id].mn[0]) res+=dist(q.x[0]-t
            [id].mn[0],0);
        if(q.x[1]<t[id].mn[1]) res+=dist(q.x[1]-t
            [id].mn[1],0);
        if(q.x[0]>t[id].mx[0]) res+=dist(q.x[0]-t
            [id].mx[0],0);
        if(q.x[1]>t[id].mx[1]) res+=dist(q.x[1]-t
            [id].mx[1],0);
        return res;
    }
    kdtnode a,b;
    inline int check(kdtnode &x)//在矩形x(a,b)内
    {
        int ok=1;
        for(int i=0;i<dim;i++)
        {
            ok&=(x.x[i]>=a.x[i]);
            ok&=(x.x[i]<=b.x[i]);
        }
        return ok;
    }
    inline int allin(kdtnode &x)//的子树全在矩
        形x(a,b)内
    {
        int ok=1;
        for(int i=0;i<dim;i++)
        {
            ok&=(x.mn[i]>=a.x[i]);
            ok&=(x.mx[i]<=b.x[i]);
        }
        return ok;
    }
    inline int allout(kdtnode &x)//的子树全不在矩
        形x(a,b)内
    {
        int ok=0;
        for(int i=0;i<dim;i++)
        {
            ok|=(x.mx[i]<a.x[i]);
            ok|=(x.mn[i]>b.x[i]);
        }
        return ok;
    }
    type res;
    void query(int id)
    {
        if(!id) return;
        if(allout(t[id])||t[id].sz==0) return;
        if(allin(t[id]))
        {
            res=min(res,t[id].tag);
            return;
        }
        if(check(t[id])&&t[id].exist) res=min(res
            ,t[id].v);
        int l,r;
        l=ls;
        r=rs;
        if(t[l].tag>t[r].tag) swap(l,r);
        if(t[l].tag<res) query(l);
        if(t[r].tag<res) query(r);
    }
    inline type query(kdtnode _a,kdtnode _b)
    {
        a=_a;b=_b;
        res=INF;
```

```
        query(root);
        return res;
    }
};
#undef type
#undef ls
#undef rs
}
using namespace kd_tree;
```

2.13 虚树

```
VI mp[MAX];
void add_edge(int a,int b)
{
    mp[a].pb(b);
    mp[b].pb(a);
}
int st[MAX],top,dfn[MAX];
bool cmp(int x,int y){return dfn[x]<dfn[y];}
int build_vtree(vector<int> &a)// return root
{
    int lca;
    sort(all(a),cmp);
    a.erase(unique(all(a)),a.end());
    assert(a.size()>0);
    top=0;
    st[top++]=a[0];
    VI tmp;
    for(int i=1;i<sz(a);i++)
    {
        if(top==0){st[top++]=a[i]; continue;}
        lca=lca(a[i],st[top-1]);
        while(top>1&&dfn[st[top-2]]>=dfn[lca])
            add_edge(st[top-2],st[top-1]),top--;
        if(lca!=st[top-1]) {add_edge(lca,st[top-1]),
            st[top-1]=lca; tmp.push_back(lca);}
        st[top++]=a[i];
    }
    while(top>1) add_edge(st[top-2],st[top-1]),top
        --;
    for(auto it:tmp) a.push_back(it);
    return st[0];
}
```

2.14 pbds 可并堆

```
#include <ext/pb_ds/priority_queue.hpp>
using namespace __gnu_pbds;
__gnu_pbds::priority_queue<int,greater<int>,
    pairing_heap_tag> q[MAX];
//q[i].join(q[j]) -> q[j合并到]q[i],q[j清空]
```

2.15 k 叉哈夫曼树

用两个队列代替优先队列复杂度 O(n)

注意：小的先进原数组有序

```
int a[MAX];
int Huffman(int k)
{
    int i,res,s;
    queue<int> q,d;
    s=((n-1)%(k-1)?k-1-(n-1)%(k-1):0);//计算要补多少
        个0
    while(s--) q.push(0);
    for(i=1;i<=n;i++) q.push(a[i]);
    res=0;
    while(sz(q)+sz(d)>1)
    {
        s=0;
        for(i=0;i<k;i++)
        {
            if(sz(q)&&sz(d))
            {
                if(q.front()<d.front())
                {
                    s+=q.front();
                    q.pop();
                }
                else
                {
                    s+=d.front();
                    d.pop();
                }
            }
            else if(sz(q))
            {
                s+=q.front();
                q.pop();
            }
            else if(sz(d))
            {
                s+=d.front();
                d.pop();
            }
        }
        res+=s;
        d.push(s);
    }
    return res;
}
```

2.16 笛卡尔树

O(n) 构造笛卡尔树返回根

性质:

1. 树中的元素满足二叉搜索树性质，要求按照中序遍历得到的序列为原数组序列

2. 树中节点满足堆性质，节点的 key 值要大于其左右子节点的 key 值

```
namespace Cartesian_Tree
{
    int l[MAX],r[MAX],vis[MAX],stk[MAX];
    int build(int *a,int n)
    {
        int i,top=0;
        for(i=1;i<=n;i++) l[i]=0,r[i]=0,vis[i]=0;
        for(i=1;i<=n;i++)
        {
            int k=top;
            while(k>0&&a[stk[k-1]]>a[i]) k--;
            if(k) r[stk[k-1]]=i;
            if(k<top) l[i]=stk[k];
            stk[k++]=i;
            top=k;
        }
        for(i=1;i<=n;i++) vis[l[i]]=vis[r[i]]=1;
        for(i=1;i<=n;i++)
        {
            if(!vis[i]) return i;
        }
    }
}
```

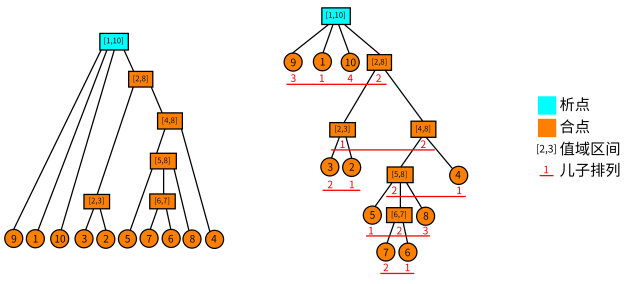
2.17 析合树

id[x]：序列中第 x 个数在析合树上的编号。

l[x],r[x]：节点 x 的作用域。

type[x]：节点 x 的类型，0 表示析点，1 表示合点，默认叶子节点为析点。

注意：若一个节点为合点，这个节点的儿子序列有序。析合树举例：




```
struct permutation_tree
{
    struct RMQ
    {
        #define type int
        int lg[MAX],bin[22];
        type mx[MAX][22],mn[MAX][22];
        void work(int n,type *v)
        {
            int i,j;
            for(i=bin[0]=1;1<<(i-1)<=n;i++) bin[i]=(
                bin[i-1]<<1);
            for(i=2,lg[1]=0;i<=n;i++) lg[i]=lg[i
                >>1]+1;
            for(i=1;i<=n;i++) mx[i][0]=mn[i][0]=v[i];
            for(j=1;1<<(j-1)<=n;j++)
            {
                for(i=1;i+bin[j]-1<=n;i++)
                {
                    mx[i][j]=max(mx[i][j-1],mx[i+bin[j
                        -1]][j-1]);
                    mn[i][j]=min(mn[i][j-1],mn[i+bin[j
                        -1]][j-1]);
                }
            }
        }
        type ask_max(int l,int r)
        {
            int t=lg[r-l+1];
            return max(mx[l][t],mx[r-bin[t]+1][t]);
        }
        type ask_min(int l,int r)
        {
            int t=lg[r-l+1];
            return min(mn[l][t],mn[r-bin[t]+1][t]);
        }
        #undef type
    }rmq;
    struct Segment_Tree
    {
        #define type int
        #define ls (id<<1)
        #define rs (id<<1|1)
        int n,ql,qr;
        type mn[MAX<<2],tag[MAX<<2],qv;
        void mdf(int id,type v){mn[id]+=v,tag[id]+=v
            ;}
        void pushup(int id){mn[id]=min(mn[ls],mn[rs])
            ;}
        void pushdown(int id)
        {
            if(!tag[id]) return;
```

```
        mdf(ls,tag[id]);
        mdf(rs,tag[id]);
        tag[id]=0;
    }
    void build(int l,int r,int id)
    {
        tag[id]=mn[id]=0;
        if(l==r) return;
        int mid=(l+r)>>1;
        build(l,mid,ls);
        build(mid+1,r,rs);
        pushup(id);
    }
    void update(int l,int r,int id)
    {
        if(l>=ql&&r<=qr){mdf(id,qv);return;}
        pushdown(id);
        int mid=(l+r)>>1;
        if(ql<=mid) update(l,mid,ls);
        if(qr>mid) update(mid+1,r,rs);
        pushup(id);
    }
    int query(int l,int r,int id)
    {
        if(l==r) return l;
        pushdown(id);
        int mid=(l+r)>>1;
        if(!mn[ls]) return query(l,mid,ls);
        else query(mid+1,r,rs);
    }
    void build(int _n){n=_n;build(1,n,1);}
    void upd(int l,int r,type v){ql=l;qr=r;qv=v;
        update(1,n,1);}
    type ask(int l,int r){ql=l;qr=r;return query
        (1,n,1);}
    #undef type
    #undef ls
    #undef rs
}tr;
bool check(int l,int r){return rmq.ask_max(l,r)-
    rmq.ask_min(l,r)==r-l;}
int st[MAX],st1[MAX],st2[MAX],top,top1,top2,m[
    MAX];
int tot,id[MAX],l[MAX],r[MAX],type[MAX];
VI mp[MAX];
void add_edge(int a,int b){mp[a].pb(b);}
int build(int n,int *a)
{
    int now,i,tmp;
    tr.build(n);
    rmq.work(n,a);
    for(i=0;i<=2*n;i++)
    {
        mp[i].clear();
        type[i]=0;
    }
    top=top1=top2=0;
    tot=0;
    for(i=1;i<=n;i++)
    {
        while(top1&&a[i]<=a[st1[top1]])
        {
            tr.upd(st1[top1-1]+1,st1[top1],a[st1[
                top1]]);
            top1--;
        }
        while(top2&&a[i]>=a[st2[top2]])
        {
            tr.upd(st2[top2-1]+1,st2[top2],-a[st2[
                top2]]);
            top2--;
        }
        tr.upd(st1[top1]+1,i,-a[i]);
        st1[++top1]=i;
        tr.upd(st2[top2]+1,i,a[i]);
        st2[++top2]=i;
        id[i]=++tot;
        l[tot]=r[tot]=i;
        tmp=tr.ask(1,n);
        now=tot;
        while(top&&l[st[top]]>=tmp)
        {
            if(type[st[top]]&&check(m[st[top]],i))
            {
                r[st[top]]=i;
                add_edge(st[top],now);
                now=st[top--];
            }
            else if(check(l[st[top]],i))
            {
                type[++tot]=1;
                l[tot]=l[st[top]];
                r[tot]=i;
                m[tot]=l[now];
                add_edge(tot,st[top--]);
                add_edge(tot,now);
                now=tot;
            }
            else
            {
                add_edge(++tot,now);
                do
                {
                    add_edge(tot,st[top--]);
                }while(top&&!check(l[st[top]],i));
                l[tot]=l[st[top]];
                r[tot]=i;
                add_edge(tot,st[top--]);
```

```
                now=tot;
            }
        }
        st[++top]=now;
        tr.upd(1,i,-1);
    }
    return st[1];
}
void work(int n,int *a)
{
    int rt=build(n,a);

}
}pt;// MAX must *2
```

3 图论

3.1 链式前向星

```
//这里指的是边数MAX 双向边要*2
int head[MAX],tot;
struct node
{
    int to,v,next;
}mp[MAX<<1];
void init()
{
    mem(head,-1);
    tot=0;
}
void add(int x,int y,int v)
{
    mp[tot].v=v;
    mp[tot].to=y;
    mp[tot].next=head[x];
    head[x]=tot++;
}
```

3.2 最短路

3.2.1 dijkstra

```
struct node
{
    int id;
    int v;
    node(){}
    node(int a,int b) :id(a),v(b){}
    friend bool operator <(node a,node b){return a.v
        >b.v;}
};
vector<node> mp[MAX];
bool flag[MAX];
int dis[MAX];
void dij(int s)
{
    priority_queue<node> q;
    node t,to;
    mem(dis,0x3f);
    mem(flag,0);
    dis[s]=0;
    q.push(node(s,0));
    while(!q.empty())
    {
        t=q.top();
        q.pop();
        if(flag[t.id]) continue;
        flag[t.id]=1;
        for(int i=0;i<sz(mp[t.id]);i++)
        {
            to=mp[t.id][i];
            if(dis[to.id]>dis[t.id]+to.v)
            {
                dis[to.id]=dis[t.id]+to.v;
                q.push(node(to.id,dis[to.id]));
            }
        }
    }
}
```

3.2.2 spfa

```
//最长路 变为dis-INF 松弛改成<
struct node
{
    int id;
    int v;
    node(){}
    node(int a,int b) :id(a),v(b){}
};
vector<node> mp[MAX];
int dis[MAX];
bool flag[MAX];
void spfa(int s)
{
    queue<node> q;
    node t,to;
    mem(dis,0x3f);
    mem(flag,0);
    dis[s]=0;
    flag[s]=1;
    q.push(node(s,dis[s]));
    while(!q.empty())
    {
        t=q.front();
        q.pop();
        flag[t.id]=0;
```

```
        for(int i=0;i<sz(mp[t.id]);i++)
        {
            to=mp[t.id][i];
            if(dis[to.id]>dis[t.id]+to.v)
            {
                dis[to.id]=dis[t.id]+to.v;
                if(!flag[to.id])
                {
                    q.push(node(to.id,dis[to.id]));
                    flag[to.id]=1;
                }
            }
        }
    }
}
```

3.2.3 floyd 求最小环

```
int mp[111][111],dis[111][111],ans;
void floyd(int n)
{
    int i,j,k;
    for(k=1;k<=n;k++)
    {
        for(i=1;i<k;i++)
        {
            if(mp[k][i]==INF) continue;
            for(j=i+1;j<k;j++)
            {
                if(mp[k][j]==INF) continue;
                ans=min(ans,mp[k][i]+mp[k][j]+dis[i][j
                    ]);
            }
        }
        for(i=1;i<=n;i++)
        {
            if(dis[i][k]==INF) continue;
            for(j=1;j<=n;j++)
            {
                if(dis[k][j]==INF) continue;
                dis[i][j]=min(dis[i][j],dis[i][k]+dis[
                    k][j]);
            }
        }
    }
}
int main()
{
    int m,i,a,b,w,n;
    while(~scanf("%d%d",&n,&m))
    {
        mem(mp,0x3f);
        mem(dis,0x3f);
        ans=INF;
        while(m--)
        {
            scanf("%d%d%d",&a,&b,&w);
            mp[a][b]=mp[b][a]=dis[a][b]=dis[b][a]=min
                (mp[a][b],w);
        }
        floyd(n);
        if(ans==INF) puts("It's impossible.");
        else printf("%d\n",ans);
    }
    return 0;
}
```

3.3 最小生成树

3.3.1 kruskal

```
struct dsu
{
    int pre[MAX];
    void init(int n)
    {
        int i;
        for(i=1;i<=n;i++) pre[i]=i;
    }
    int find(int x)
    {
        if(pre[x]!=x) pre[x]=find(pre[x]);
        return pre[x];
    }
    bool merge(int a,int b)
    {
        int ra,rb;
        ra=find(a);
        rb=find(b);
        if(ra!=rb)
        {
            pre[ra]=rb;
            return 1;
        }
        return 0;
    }
}dsu;
struct node
{
    int x,y,w;
    node(){}
    node(int a,int b,int c):x(a),y(b),w(c){}
    friend bool operator<(node a,node b) {return a.w
        <b.w;}
};
vector<node> edge;
int kruskal(int n)
{
```

```
    int res=0;
    dsu.init(n);
    sort(all(edge));
    for(int i=0;i<sz(edge);i++)
    {
        if(dsu.merge(edge[i].x,edge[i].y)) res+=edge[
            i].w;
    }
    return res;
}
```

3.3.2 prim

```
struct node
{
    int id;
    int v;
    node(){}
    node(int a,int b) :id(a),v(b){}
    friend bool operator <(node a,node b){return a.v
        >b.v;}
};
vector<node> mp[MAX];
bool flag[MAX];
int dis[MAX];
int prim()
{
    int res=0;
    node t,to;
    priority_queue<node> q;
    mem(dis,0x3f);
    mem(flag,0);
    dis[1]=0;
    q.push(node(1,dis[1]));
    while(!q.empty())
    {
        t=q.top();
        q.pop();
        if(flag[t.id]) continue;
        flag[t.id]=1;
        res+=dis[t.id];
        for(int i=0;i<sz(mp[t.id]);i++)
        {
            to=mp[t.id][i];
            if(!flag[to.id]&&dis[to.id]>to.v)
            {
                dis[to.id]=to.v;
                q.push(node(to.id,dis[to.id]));
            }
        }
    }
    return res;
}
```

3.4 二分图匹配

3.4.1 匈牙利算法

```
//二分图匹配
/*最小点覆盖的点数最大匹配数
=最小路径覆盖的边数顶点数
=n最大匹配数-最大独立集最小路径覆盖顶点数
==n最大匹配数-
*/
//匈牙利算法 O(n*m)
struct Bipartite_Matching
{
    static const int N=;
    int n,m;
    VI mp[N];
    int link[N],s[N];
    bool used[N],flag[N];
    void init(int _n,int _m)
    {
        n=_n;
        m=_m;
        for(int i=0;i<=n;i++) mp[i].clear();
    }
    void add_edge(int a,int b){mp[a].pb(b);}
    bool dfs(int x)
    {
        int i,to;
        flag[x]=1;
        for(i=0;i<sz(mp[x]);i++)
        {
            to=mp[x][i];
            if(used[to]) continue;
            used[to]=1;
            if(link[to]==-1||dfs(link[to]))
            {
                link[to]=x;
                s[x]=to;
                return 1;
            }
        }
        return 0;
    }
    int max_match()
    {
        int i,res;
        mem(link,-1);
        mem(s,-1);
        res=0;
        for(i=1;i<=n;i++)
        {
            if(!sz(mp[i])) continue;
            mem(used,0);
            if(dfs(i)) res++;
        }
```

```
        return res;
    }
    int min_cover(VI &x,VI &y)
    {
        int i,res;
        res=max_match();
        mem(flag,0);
        mem(used,0);
        x.clear();
        y.clear();
        for(i=1;i<=n;i++)
        {
            if(s[i]==-1) dfs(i);
        }
        for(i=1;i<=n;i++)
        {
            if(!flag[i]) x.pb(i);
        }
        for(i=1;i<=m;i++)
        {
            if(used[i]) y.pb(i);
        }
        return res;
    }
}bpm;
```

3.4.2 二分图带权匹配

```
struct KM
{
    #define type int
    #define inf INF
    static const int N=;
    int n,mx[N],my[N],prv[N];
    type slk[N],lx[N],ly[N],w[N][N];
    bool vx[N],vy[N];
    void init(int _n)
    {
        n=_n;
        for(int i=1;i<=n;i++)
        {
            for(int j=1;j<=n;j++)
            {
                w[i][j]=0;
            }
        }
    }
    void add_edge(int x,int y,type val){w[x][y]=val
        ;}
    void match(int y){while(y) swap(y,mx[my[y]=prv[y
        ]]);}
    void bfs(int x)
    {
        int i,y;
        type d;
        for(i=1;i<=n;i++)
        {
            vx[i]=vy[i]=0;
            slk[i]=inf;
        }
        queue<int> q;
        q.push(x);
        vx[x]=1;
        while(1)
        {
            while(!q.empty())
            {
                x=q.front();
                q.pop();
                for(y=1;y<=n;y++)
                {
                    d=lx[x]+ly[y]-w[x][y];
                    if(!vy[y]&&d<=slk[y])
                    {
                        prv[y]=x;
                        if(!d)
                        {
                            if(!my[y]) return match(y);
                            q.push(my[y]);
                            vx[my[y]]=1;
                            vy[y]=1;
                        }
                        else slk[y]=d;
                    }
                }
            }
            d=inf+1;
            for(i=1;i<=n;i++)
            {
                if(!vy[i]&&slk[i]<d)
                {
                    d=slk[i];
                    y=i;
                }
            }
            for(i=1;i<=n;i++)
            {
                if(vx[i]) lx[i]-=d;
                if(vy[i]) ly[i]+=d;
                else slk[i]-=d;
            }
            if(!my[y]) return match(y);
            q.push(my[y]);
            vx[my[y]]=1;
            vy[y]=1;
        }
    }
    type max_match()
```

```
    {
        int i;
        type res;
        for(i=1;i<=n;i++)
        {
            mx[i]=my[i]=ly[i]=0;
            lx[i]=*max_element(w[i]+1,w[i]+n+1);
        }
        for(i=1;i<=n;i++) bfs(i);
        res=0;
        for(i=1;i<=n;i++) res+=lx[i]+ly[i];
        return res;
    }
    #undef type
    #undef inf
}km;
/*
O(n^3)
km.init(n);
km.add_edge(a,b,val); a,b: 1~n
*/
```

3.5 最大流

3.5.1 dinic

```
struct Dinic
{
    #define type int
    #define inf INF
    static const int N=;
    struct node
    {
        int from,to;
        type cap,flow;
        node(int u,int v,type c,type f):from(u),to(v)
            ,cap(c),flow(f){}
    };
    int n,s,t;
    vector<node> edge;
    vector<int> mp[N];
    int vis[N],dist[N],id[N];
    void init(int _n)
    {
        n=_n;
        edge.clear();
        for(int i=0;i<=n;i++)
        {
            mp[i].clear();
            id[i]=dist[i]=vis[i]=0;
        }
    }
    void add_edge(int from,int to,type cap)
    {
        edge.pb(node(from,to,cap,0));
        edge.pb(node(to,from,0,0));
        int m=edge.size();
        mp[from].pb(m-2);
        mp[to].pb(m-1);
    }
    bool bfs()
    {
        int i,x;
        mem(vis,0);
        queue<int>q;
        q.push(s);
        dist[s]=0;
        vis[s]=1;
        while(!q.empty())
        {
            x=q.front();
            q.pop();
            for(i=0;i<mp[x].size();i++)
            {
                node &e=edge[mp[x][i]];
                if(!vis[e.to]&&e.cap>e.flow)
                {
                    vis[e.to]=1;
                    dist[e.to]=dist[x]+1;
                    q.push(e.to);
                }
            }
        }
        return vis[t];
    }
    type dfs(int x,type a)
    {
        if(x==t||!a) return a;
        type flow=0,f;
        for(int &i=id[x];i<mp[x].size();i++)
        {
            node &e=edge[mp[x][i]];
            if(dist[x]+1==dist[e.to]&&(f=dfs(e.to,min
                (a,e.cap-e.flow)))>0)
            {
                e.flow+=f;
                edge[mp[x][i]^1].flow-=f;
                flow+=f;
                a-=f;
                if(!a) break;
            }
        }
        return flow;
    }
    type max_flow(int _s,int _t)
    {
        s=_s;
        t=_t;
```

```
        type res=0;
        while(bfs())
        {
            for(int i=0;i<=n;i++) id[i]=0;
            res+=dfs(s,inf);
        }
        return res;
    }
    #undef type
    #undef inf
}dc;
/*
dc.init(n);
dc.add_edge(a,b,cap); a,b: 1~n
*/
```

3.5.2 ISAP

```
struct ISAP
{
    #define type int
    #define inf INF
    static const int N=;
    struct node
    {
        int from,to;
        type cap,flow;
        node(){}
        node(int u,int v,type c,type f):from(u),to(v)
            ,cap(c),flow(f){}
    };
    int p[N],num[N],cur[N];
    type d[N];
    int t,s,n;
    bool vis[N];
    vector<int> mp[N];
    vector<node> edge;
    void init(int _n)
    {
        n=_n;
        for(int i=0;i<=n;i++)
        {
            mp[i].clear();
            d[i]=inf;
            vis[i]=num[i]=cur[i]=0;
        }
        edge.clear();
    }
    void add(int from,int to,type cap)
    {
        edge.pb(node(from,to,cap,0));
        edge.pb(node(to,from,0,0));
        int m=edge.size();
        mp[from].pb(m-2);
        mp[to].pb(m-1);
    }
    bool bfs()
    {
        queue<int> q;
        d[t]=0;
        vis[t]=1;
        q.push(t);
        while(!q.empty())
        {
            int u=q.front();
            q.pop();
            for(int i=0;i<mp[u].size();i++)
            {
                node &e=edge[mp[u][i]^1];
                if(!vis[e.from]&&e.cap>e.flow)
                {
                    vis[e.from]=true;
                    d[e.from]=d[u]+1;
                    q.push(e.from);
                }
            }
        }
        return vis[s];
    }
    type augment()
    {
        int u=t;
        type flow=inf;
        while(u!=s)
        {
            node &e=edge[p[u]];
            flow=min(flow,e.cap-e.flow);
            u=edge[p[u]].from;
        }
        u=t;
        while(u!=s)
        {
            edge[p[u]].flow+=flow;
            edge[p[u]^1].flow-=flow;
            u=edge[p[u]].from;
        }
        return flow;
    }
    type maxflow(int _s,int _t)
    {
        s=_s;
        t=_t;
        type flow=0;
        bfs();
        if(d[s]>=n) return 0;
        for(int i=0;i<n;i++)
        {
            if(d[i]<inf) num[d[i]]++;
```

```
        }
        int u=s;
        while(d[s]<n)
        {
            if(u==t)
            {
                flow+=augment();
                u=s;
            }
            bool ok=false;
            for(int i=cur[u];i<mp[u].size();i++)
            {
                node &e=edge[mp[u][i]];
                if(e.cap>e.flow&&d[u]==d[e.to]+1)
                {
                    ok=true;
                    p[e.to]=mp[u][i];
                    cur[u]=i;
                    u=e.to;
                    break;
                }
            }
            if(!ok)
            {
                type mn=n-1;
                for(int i=0;i<mp[u].size();i++)
                {
                    node &e=edge[mp[u][i]];
                    if(e.cap>e.flow) mn=min(mn,d[e.to])
                        ;
                }
                if(--num[d[u]]==0) break;
                num[d[u]=mn+1]++;
                cur[u]=0;
                if(u!=s) u=edge[p[u]].from;
            }
        }
        return flow;
    }
    #undef type
    #undef inf
}isap;
```

3.5.3 high-level-preflow-push

```
struct High_Level_Preflow_Push
{
    static const int N=10010;
    struct node{int v,cap,index;};
    vector<node> edge[N];
    vector<int> List[N];
    vector<list<int>::iterator> listit;
    list<int> dlist[N];
    int highest,highestActive,vis[N],excess[N],
        height[N],n;
    void init(int _n)
    {
        n=_n;
        for(int i=0;i<=n;i++) edge[i].clear();
    }
    void add_edge(int u,int v,int cap)
    {
        edge[u].push_back(node{v,cap,edge[v].size()})
            ;
        edge[v].push_back(node{u,0,edge[u].size()-1})
            ;
    }
    void globalRelabel(int t)
    {
        int u,i,hp,v,index;
        queue<int> q;
        for(i=0;i<=n;i++)
        {
            height[i]=n;
            List[i].clear();
            dlist[i].clear();
            vis[i]=0;
        }
        height[t]=0;
        q.push(t);
        while(!q.empty())
        {
            u=q.front();
            q.pop();
            for(i=0;i<edge[u].size();i++)
            {
                v=edge[u][i].v;
                index=edge[u][i].index;
                if(height[v]==n&&edge[v][index].cap>0)
                {
                    height[v]=height[u]+1;
                    vis[height[v]]++;
                    q.push(hp=v);
                }
            }
        }
        for(i=0;i<n;i++)
        {
            if(height[i]<n)
            {
                listit[i]=dlist[height[i]].insert(
                    dlist[height[i]].begin(),i);
                if(excess[i]>0) List[height[i]].
                    push_back(i);
            }
        }
        highest=height[hp];
```

```
    highestActive=height[hp];
}
void push(int u,node &e)
{
    int v,df;
    v=e.v;
    df=min(excess[u],e.cap);
    e.cap=e.cap-df;
    edge[v][e.index].cap=edge[v][e.index].cap+df;
    excess[u]=excess[u]-df;
    excess[v]=excess[v]+df;
    if(excess[v]>0&&excess[v]<=df) List[height[v
        ]].push_back(v);
}
void discharge(int u)
{
    int i,nh,v,cap,h;
    nh=n;
    for(i=0;i<edge[u].size();i++)
    {
        v=edge[u][i].v;
        cap=edge[u][i].cap;
        if(cap>0)
        {
            if(height[u]==height[v]+1)
            {
                push(u,edge[u][i]);
                if(excess[u]==0) return;
            }
            else nh=min(nh,height[v]+1);
        }
    }
    h=height[u];
    if(vis[h]==1)
    {
        for(i=h;i<=highest;i++)
        {
            for(list<int>::iterator it=dlist[i].
                begin();it!=dlist[i].end();it++)
            {
                vis[height[*it]]--;
                height[*it]=n;
            }
            dlist[i].clear();
        }
        highest=h-1;
    }
    else
    {
        vis[h]--;
        listit[u]=dlist[h].erase(listit[u]);
        height[u]=nh;
        if(nh==n) return;
        vis[nh]++;
        listit[u]=dlist[nh].insert(dlist[nh].
            begin(),u);
        highestActive=nh;
        highest=max(highest,highestActive);
        List[nh].push_back(u);
    }
}
int maxflow(int s,int e)
{
    int i,u;
    if(s==e) return 0;
    highestActive=0;
    highest=0;
    for(i=0;i<=n;i++) height[i]=vis[i]=excess[i
        ]=0;
    height[s]=n;
    listit.resize(n);
    for(i=0;i<n;i++)
    {
        if(i!=s)
        {
            listit[i]=dlist[height[i]].insert(
                dlist[height[i]].begin(),i);
        }
    }
    vis[0]=n-1;
    excess[s]=INF;
    excess[e]=-INF;
    for(i=0;i<edge[s].size();i++) push(s,edge[s][
        i]);
    globalRelabel(e);
    while(highestActive>=0)
    {
        if(List[highestActive].empty()==1)
        {
            highestActive--;
            continue;
        }
        u=List[highestActive].back();
        List[highestActive].pop_back();
        discharge(u);
    }
    return excess[e]+INF;
}
}hlpp;
```

3.6 最小费用最大流

3.6.1 spfa 费用流

```
struct MCMF
{
    #define type int
    #define inf INF
```

```
    static const int N=;
    struct node
    {
        int from,to;
        type cap,flow,cost;
        node(){}
        node(int u,int v,type c,type f,type co):from(u),to(v),cap(c),flow(f),cost(co){}
    };
    int n,s,t;
    vector<node> edge;
    vector<int> mp[N];
    int vis[N],id[N];
    type d[N],a[N];
    void init(int _n)
    {
        n=_n;
        for(int i=0;i<=n;i++) mp[i].clear();
        edge.clear();
    }
    void add_edge(int from,int to,type cap,type cost=0)
    {
        edge.pb(node(from,to,cap,0,cost));
        edge.pb(node(to,from,0,0,-cost));
        int m=edge.size();
        mp[from].pb(m-2);
        mp[to].pb(m-1);
    }
    bool spfa(type& flow,type& cost)
    {
        for(int i=0;i<=n;i++)
        {
            d[i]=inf;
            vis[i]=0;
        }
        d[s]=0;vis[s]=1;id[s]=0;a[s]=inf;
        queue<int> q;
        q.push(s);
        while(!q.empty())
        {
            int x=q.front();
            q.pop();
            vis[x]=0;
            for(int i=0;i<mp[x].size();i++)
            {
                node& e=edge[mp[x][i]];
                int to=e.to;
                if(e.cap>e.flow&&d[to]>d[x]+e.cost)
                {
                    d[to]=d[x]+e.cost;
                    a[to]=min(a[x],e.cap-e.flow);
                    id[to]=mp[x][i];
                    if(!vis[to])
                    {
                        vis[to]=1;
                        q.push(to);
                    }
                }
            }
        }
        if(d[t]==inf) return false;
        flow+=a[t];
        cost+=a[t]*d[t];
        int x=t;
        while(x!=s)
        {
            edge[id[x]].flow+=a[t];
            edge[id[x]^1].flow-=a[t];
            x=edge[id[x]].from;
        }
        return true;
    }
    pair<type,type> mincost_maxflow(int _s,int _t)
    {
        type flow=0,cost=0;
        s=_s;
        t=_t;
        while(spfa(flow,cost));
        return MP(cost,flow);
    }
    #undef type
    #undef inf
}mcmf;
/*
mcmf.init(n);
mcmf.add_edge(a,b,cap,cost); a,b: 1~n
*/
```

3.6.2 dijkstra 费用流

```
struct MCMF_dij
{
    #define type int
    #define inf INF
    #define PTI pair<type,int>
    static const int N=;
    struct node
    {
        int from,to;
        type flow,cost;
        node(){}
        node(int u,int v,type f,type co):from(u),to(v),flow(f),cost(co){}
    };
    int n,s,t,id[N];
    vector<node> edge;
    vector<int> mp[N];
```

```
    type dis[N],h[N];
    void init(int _n)
    {
        n=_n;
        for(int i=0;i<=n;i++) mp[i].clear();
        edge.clear();
    }
    void add_edge(int from,int to,type cap,type cost
        =0)
    {
        edge.pb(node(from,to,cap,cost));
        edge.pb(node(to,from,0,-cost));
        int m=edge.size();
        mp[from].pb(m-2);
        mp[to].pb(m-1);
    }
    bool dij(type& flow,type& cost)
    {
        for(int i=0;i<=n;i++) dis[i]=inf;
        dis[s]=0;id[s]=0;
        priority_queue<PTI ,vector<PTI>,greater<PTI>
            > q;
        q.push(MP(type(0),s));
        while(!q.empty())
        {
            PTI x=q.top();
            q.pop();
            if(x.fi!=dis[x.se]) continue;
            if(x.se==t) break;
            for(int i=0;i<mp[x.se].size();i++)
            {
                node& e=edge[mp[x.se][i]];
                int to=e.to;
                type now_cost=e.cost+h[x.se]-h[to];
                if(e.flow>0&&dis[to]>dis[x.se]+
                    now_cost)
                {
                    dis[to]=dis[x.se]+now_cost;
                    q.push(MP(dis[to],to));
                    e.from=x.se;
                    id[to]=mp[x.se][i];
                }
            }
        }
        if(dis[t]==inf) return false;
        for(int i=0;i<=n;i++) h[i]=min(h[i]+dis[i],
            inf);
        type new_flow=inf;
        int x=t;
        while(x!=s)
        {
            new_flow=min(new_flow,edge[id[x]].flow);
            x=edge[id[x]].from;
        }
        flow+=new_flow;
        cost+=new_flow*h[t];
        x=t;
        while(x!=s)
        {
            edge[id[x]].flow-=new_flow;
            edge[id[x]^1].flow+=new_flow;
            x=edge[id[x]].from;
        }
        return true;
    }
    pair<type,type> mincost_maxflow(int _s,int _t)
    {
        type flow=0,cost=0;
        for(int i=0;i<=n;i++) h[i]=0;
        s=_s;
        t=_t;
        while(dij(flow,cost));
        return MP(cost,flow);
    }
    #undef type
    #undef inf
    #undef PTI
}mcmf;
```

3.7 强连通分量

```
int scc,top,tot;
vector<int> mp[MAX];
int low[MAX],dfn[MAX],belong[MAX];
int stk[MAX],flag[MAX];
void init(int n)
{
    int i;
    for(i=1;i<=n;i++)
    {
        mp[i].clear();
        low[i]=0;
        dfn[i]=0;
        stk[i]=0;
        flag[i]=0;
    }
    scc=top=tot=0;
}
void tarjan(int x)
{
    int to,i,temp;
    stk[top++]=x;
    flag[x]=1;
    low[x]=dfn[x]=++tot;
    for(i=0;i<mp[x].size();i++)
    {
        to=mp[x][i];
```

```
        if(!dfn[to])
        {
            tarjan(to);
            low[x]=min(low[x],low[to]);
        }
        else if(flag[to]) low[x]=min(low[x],dfn[to]);
    }
    if(low[x]==dfn[x])
    {
        scc++;
        do
        {
            temp=stk[--top];
            flag[temp]=0;
            belong[temp]=scc;
        }while(temp!=x);
    }
}
```

3.8 双联通分量

3.8.1 边双连通

```
namespace Tarjan
{
    int bcc,top,tot,n;
    vector<int> mp[MAX];
    vector<PII > bridge;
    int low[MAX],dfn[MAX],belong[MAX],fa[MAX];
    int stk[MAX];
    int cut[MAX],add_block[MAX];
    void dfs(int x,int pre)
    {
        int to,i,tmp,k,son;
        stk[top++]=x;
        low[x]=dfn[x]=++tot;
        fa[x]=pre;
        son=k=0;
        for(auto to:mp[x])
        {
            if(to==pre&&!k)
            {
                k++;
                continue;
            }
            if(!dfn[to])
            {
                son++;
                dfs(to,x);
                low[x]=min(low[x],low[to]);
                if(x!=pre&&low[to]>=dfn[x])
                {
                    cut[x]=1;
                    add_block[x]++;
                }
                if(low[to]>dfn[x]) bridge.pb(MP(x,to))
                    ;
            }
            else low[x]=min(low[x],dfn[to]);
        }
        if(x==pre&&son>1)
        {
            cut[x]=1;
            add_block[x]=son-1;
        }
        if(low[x]==dfn[x])
        {
            bcc++;
            do
            {
                tmp=stk[--top];
                belong[tmp]=bcc;
            }while(tmp!=x);
        }
    }
    void work(int _n,vector<int> e[])
    {
        n=_n;
        for(int i=1;i<=n;i++)
        {
            mp[i]=e[i];
            low[i]=dfn[i]=fa[i]=stk[i]=0;
            cut[i]=add_block[i]=0;
        }
        bcc=top=tot=0;
        bridge.clear();
        for(int i=1;i<=n;i++)
        {
            if(!dfn[i]) dfs(i,i);
        }
    }
    void rebuild(vector<int> e[])
    {
        int i,t;
        for(i=1;i<=n;i++) e[i].clear();
        for(i=1;i<=n;i++)
        {
            t=fa[i];
            if(belong[i]!=belong[t])
            {
                e[belong[i]].pb(belong[t]);
                e[belong[t]].pb(belong[i]);
            }
        }
    }
}
```

3.9 团

3.9.1 最大团

```
struct Maximum_Clique
{
    static const int N=;
    vector<int> sol; // vertex of maximum clique
    int mp[N][N/30+1],s[N][N/30+1];
    int n,ans,dp[N];
    void init(int _n)
    {
        n=_n;
        for(int i=0;i<=n;i++)
        {
            dp[i]=0;
            mem(mp[i],0);
        }
    }
    void add_edge(int a,int b) //0~n-1
    {
        if(a>b) swap(a,b);
        if(a==b) return;
        mp[a][b/32]|=(1<<(b%32));
    }
    bool dfs(int x,int k)
    {
        int c=0,d=0;
        for(int i=0;i<(n+31)/32;i++)
        {
            s[k][i]=mp[x][i];
            if(k!=1) s[k][i]&=s[k-1][i];
            c+=__builtin_popcount(s[k][i]);
        }
        if(c==0)
        {
            if(k>ans)
            {
                ans=k;
                sol.clear();
                sol.pb(x);
                return 1;
            }
            return 0;
        }
        for(int i=0;i<(n+31)/32;i++)
        {
            for(int a=s[k][i];a;d++)
            {
                if(k+(c-d)<=ans) return 0;
                int lb=a&(-a),lg=0;
                a^=lb;
                while(lb!=1)
                {
                    lb=(unsigned int)(lb)>>1;
                    lg++;
                }
                int u=i*32+lg;
                if(k+dp[u]<=ans) return 0;
                if(dfs(u,k+1))
                {
                    sol.pb(x);
                    return 1;
                }
            }
        }
        return 0;
    }
    int maximum_clique()
    {
        ans=0;
        for(int i=n-1;i>=0;i--)
        {
            dfs(i,1);
            dp[i]=ans;
        }
        return ans;
    }
}mcp;
/*
undirected graph
mcp.init(n);
mcp.add_edge(a,b); a,b: 0~n-1
*/
```

3.9.2 极大团计数

```
struct Bron_Kerbosch
{
    static const int N=;
    bitset<N> MASK,ZERO,mp[N];
    int n,cnt_clique;
    void init(int _n)
    {
        n=_n;
        for(int i=0;i<=n;i++) mp[i].reset();
        ZERO.reset();
        MASK=ZERO;
        MASK.flip();
    }
    void add_edge(int a,int b) //0~n-1 , undir
    {
        if(a==b) return;
        mp[a][b]=mp[b][a]=1;
    }
    void dfs(bitset<N> now,bitset<N> some,bitset<N>
        none)
    {
```

```
        if(some.none()&&none.none())//one maximal clique
        {
            cnt_clique++;
            return;
        }
        bitset<N> r=some;
        bool fi=1;
        for(int i=0;i<n;i++)
        {
            if(!r[i]) continue;
            if(fi)
            {
                fi=0;
                r&=mp[i]^MASK;
            }
            now[i]=1;
            dfs(now,some&mp[i],none&mp[i]);
            now[i]=0;
            some[i]=0;
            none[i]=1;
        }
    }
    int count_maximal_clique()
    {
        cnt_clique=0;
        bitset<N> now;
        dfs(now,MASK,ZERO);
        return cnt_clique;
    }
}bk;
/*
undirected graph
bk.init(n);
bk.add_edge(a,b); a,b: 0~n-1
*/
```

3.10 拓扑排序

```
VI mp[MAX];
int degree[MAX];
VI toplist;
bool topsort(int n)
{
    int i,x;
    queue<int> q;
    for(i=1;i<=n;i++)
    {
        if(!degree[i]) q.push(i);
    }
    toplist.clear();
    while(!q.empty())
    {
        x=q.front();
        q.pop();
        toplist.pb(x);
        for(auto to:mp[x])
        {
            degree[to]--;
            if(!degree[to]) q.push(to);
        }
    }
    return !(sz(toplist)==n);
}
```

3.11 2-sat

3.11.1 2-sat 输出任意解

```
//判断是否有解 输出任意一组解O(n+m)
int scc,top,tot;
vector<int> mp[MAX];
int low[MAX],dfn[MAX],belong[MAX];
int stk[MAX],flag[MAX];
int pos[MAX],degree[MAX],ans[MAX],outflag[MAX],cnt;
vector<int> dag[MAX];
void init(int n)
{
    int i;
    for(i=0;i<2*n;i++)
    {
        mp[i].clear();
        dag[i].clear();
        low[i]=0;
        dfn[i]=0;
        stk[i]=0;
        flag[i]=0;
        degree[i]=0;
        outflag[i]=0;
    }
    scc=top=tot=0;
}
void tarjan(int x)
{
    int to,i,temp;
    stk[top++]=x;
    flag[x]=1;
    low[x]=dfn[x]=++tot;
    for(i=0;i<mp[x].size();i++)
    {
        to=mp[x][i];
        if(!dfn[to])
        {
            tarjan(to);
            low[x]=min(low[x],low[to]);
        }
        else if(flag[to]) low[x]=min(low[x],dfn[to]);
```

```
    }
    if(low[x]==dfn[x])
    {
        scc++;
        do
        {
            temp=stk[--top];
            flag[temp]=0;
            belong[temp]=scc;
        }while(temp!=x);
    }
}
void add(int x,int y)
{
    mp[x].pb(y);
}
void topsort(int n)
{
    int i,t;
    queue<int> q;
    cnt=0;
    for(i=1;i<=scc;i++)
    {
        if(degree[i]==0) q.push(i);
        outflag[i]=0;
    }
    while(!q.empty())
    {
        t=q.front();
        q.pop();
        if(outflag[t]==0)
        {
            outflag[t]=1;
            outflag[pos[t]]=2;
        }
        for(i=0;i<sz(dag[t]);i++)
        {
            int to=dag[t][i];
            degree[to]--;
            if(degree[to]==0) q.push(to);
        }
    }
}
void builddag(int n)
{
    int i,j,to;
    for(i=0;i<2*n;i++)
    {
        for(j=0;j<sz(mp[i]);j++)
        {
            to=mp[i][j];
            if(belong[i]!=belong[to])
            {
                degree[belong[i]]++;
                dag[belong[to]].pb(belong[i]);
            }
        }
    }
}
void twosat(int n)
{
    int i;
    for(i=0;i<2*n;i++)
    {
        if(!dfn[i]) tarjan(i);
    }
    for(i=0;i<n;i++)
    {
        if(belong[2*i]==belong[2*i+1])//无解
        {
            puts("NO");
            return;
        }
        pos[belong[2*i]]=belong[2*i+1];
        pos[belong[2*i+1]]=belong[2*i];
    }
    builddag(n);
    topsort(n);
    cnt=0;
    for(i=0;i<2*n;i++)
    {
        if(outflag[belong[i]]==1) ans[cnt++]=i+1;
    }
    for(i=0;i<cnt;i++)
    {
        printf("%d\n",ans[i]);
    }
}
```

3.11.2 2-sat 字典序最小解

```
//判断是否有解 输出字典序最小的解O(n*m)
vector<int> mp[MAX];
bool flag[MAX];
int cnt,s[MAX];
void init(int n)
{
    int i;
    for(i=0;i<2*n;i++)
    {
        mp[i].clear();
    }
    mem(flag,0);
}
bool dfs(int x)
{
    int i;
    if(flag[x^1]) return 0;
```

```
    if(flag[x]) return 1;
    s[cnt++]=x;
    flag[x]=1;
    for(i=0;i<sz(mp[x]);i++)
    {
        if(!dfs(mp[x][i])) return 0;
    }
    return 1;
}
void twosat(int n)
{
    int i;
    for(i=0;i<2*n;i++)
    {
        if(!flag[i]&&!flag[i^1])
        {
            cnt=0;
            if(!dfs(i))
            {
                while(cnt) flag[s[--cnt]]=0;
                if(!dfs(i^1))//无解
                {
                    puts("NO");
                    return;
                }
            }
        }
    }
    for(i=0;i<2*n;i+=2)
    {
        if(flag[i]) printf("%d\n",i+1);
        else printf("%d\n",i+2);
    }
}
```

3.12 支配树

```
struct dominator_tree
{
    int n,tot,dfn[MAX],best[MAX],semi[MAX],idom[MAX
        ],id[MAX],fa[MAX];
    VI nex[MAX],pre[MAX],tmp[MAX],son[MAX];
    void init(int _n)
    {
        n=_n;
        for(int i=0;i<=n;i++)
        {
            nex[i].clear();
            pre[i].clear();
            tmp[i].clear();
            son[i].clear();
            dfn[i]=0;
            idom[i]=semi[i]=best[i]=fa[i]=i;
        }
    }
    void add_edge(int x,int y)
    {
        nex[x].pb(y);
        pre[y].pb(x);
    }
    int ckmin(int x,int y){return dfn[semi[x]]<dfn[
        semi[y]]?x:y;}
    int getfa(int k)
    {
        if(k==fa[k]) return k;
        int ret=getfa(fa[k]);
        best[k]=ckmin(best[fa[k]],best[k]);
        return fa[k]=ret;
    }
    void dfs(int x)
    {
        dfn[x]=++tot;
        id[tot]=x;
        for(auto to:nex[x])
        {
            if(dfn[to]) continue;
            dfs(to);
            son[x].pb(to);
        }
    }
    void tarjan(VI mp[])
    {
        int i,j,k;
        for(i=tot;i;i--)
        {
            k=id[i];
            for(auto to:pre[k])
            {
                if(!dfn[to]) continue;
                if(dfn[to]<dfn[k])
                {
                    if(dfn[to]<dfn[semi[k]]) semi[k]=to
                        ;
                }
                else
                {
                    getfa(to);
                    semi[k]=semi[ckmin(best[to],k)];
                }
            }
            if(k!=semi[k]) tmp[semi[k]].pb(k);
            for(auto to:tmp[k])
            {
                getfa(to);
                if(semi[best[to]]==k) idom[to]=k;
                else idom[to]=best[to];
            }
```

```
            for(auto to:son[k]) fa[to]=k;
        }
        for(i=2;i<=tot;i++)
        {
            k=id[i];
            if(idom[k]!=semi[k]) idom[k]=idom[idom[k
                ]];
            if (k!=idom[k])
            {
                mp[idom[k]].pb(k); //add edge

            }
        }
    }
    void work(int rt,VI mp[])
    {
        for(int i=0;i<=n;i++) mp[i].clear();
        tot=0;
        dfs(rt);
        tarjan(mp);
    }
}dt;
/*
dt.init(n);
dt.add_edge(a,b); // DAG
dt.work(rt,mp);
*/
```

4 数论

4.1 素数筛

4.1.1 埃筛

```
//x is a prime if prime[x]==x(x>=2)
int p[MAX],tot,prime[MAX];
void init(int n)
{
    int i,j;
    tot=0;
    mem(prime,0);
    prime[1]=1;
    for(i=2;i<=n;i++)
    {
        if(prime[i]) continue;
        p[tot++]=i;
        for(j=i;j<=n;j+=i)
        {
            if(!prime[j]) prime[j]=i;
        }
    }
}
```

4.1.2 线性筛

```
//x is a prime if prime[x]==x(x>=2)
int p[MAX],tot,prime[MAX];
void init(int n)
{
    int i,j;
    tot=0;
    mem(prime,0);
    prime[1]=1;
    for(i=2;i<=n;i++)
    {
        if(!prime[i]) prime[i]=p[tot++]=i;
        for(j=0;j<tot&&p[j]*i<=n;j++)
        {
            prime[i*p[j]]=p[j];
            if(i%p[j]==0) break;
        }
    }
}
```

4.1.3 区间筛

```
//O(r-l+1)
ll p[MAX],tot;
bool flag[MAX],prime[MAX];
void init(ll l,ll r)
{
    ll i,j,sq=sqrt(r+0.5);
    tot=0;
    for(i=0;i<=sq;i++) flag[i]=1;
    for(i=l;i<=r;i++) prime[i-l]=1;
    if(l==0) prime[0]=prime[1]=0;
    if(l==1) prime[0]=0;
    for(i=2;i<=sq;i++)
    {
        if(!flag[i]) continue;
        for(j=i+i;j<=sq;j+=i) flag[j]=0;
        for(j=max(2LL,(l+i-1)/i)*i;j<=r;j+=i) prime[j
            -l]=0;
    }
    for(i=l;i<=r;i++)
    {
        if(prime[i-l]) p[tot++]=i;
    }
}
```

4.2 扩展欧几里得

4.2.1 exgcd

```
/*解
xa+yb=gcd(a,b)返回值为
```

```
gcd(a,b)其中一组解为
x y通解
:
    x1=x+b/gcd(a,b)*t
    y1=y-a/gcd(a,b)*t
    (为任意整数t)
*/
ll exgcd(ll a,ll b,ll &x,ll &y)
{
    if(b==0)
    {
        x=1;
        y=0;
        return a;
    }
    ll g,t;
    g=exgcd(b,a%b,x,y);
    t=x;
    x=y;
    y=t-a/b*y;
    return g;
}
```

4.2.2 ax+by=c

```
/*
xa+yb=c求正整数最小的一组解
x有解条件
 c%gcd(a,b)==0
*/
ll linear_equation(ll a,ll b,ll c,ll &x,ll &y)
{
    ll g,t;
    g=exgcd(a,b,x,y);
    if(!c) x=y=0;
    else if((!a&&!b&&c)||c%g) return -1;//no
        solution
    else if(!a&&b) x=1,y=c/b;
    else if(a&&!b) x=c/a,y=-c/a;
    else
    {
        a/=g,b/=g,c/=g;
        x*=c,y*=c;
        t=x;
        x%=b;
        if(x<=0) x+=b;//or x<0
        ll k=(t-x)/b;
        y+=k*a;
    }
    return g;
}
```

4.2.3 exgcd 求逆元

```
/*扩展欧几里得求逆元条件

:gcd(a,mod)==1如果
gcd(a,mod)!=1 返回-1
*/
ll inv(ll a,ll p)
{
    ll g,x,y;
    g=exgcd(a,p,x,y);
    return g==1?(x+p)%p:-1;
}
```

4.3 中国剩余定理

4.3.1 CRT

```
//是除数m 是余数r 是除数的pLCM也就是答案的循环节()
int CRT(int *m,int *r,int n)
{
    int p=m[0],res=r[0],x,y,g;
    for(int i=1;i<n;i++)
    {
        g=exgcd(p,m[i],x,y);
        if((r[i]-res)%g) return -1;//无解
        x=(r[i]-res)/g*x%(m[i]/g);
        res+=x*p;
        p=p/g*m[i];
        res%=p;
    }
    return res>0?res:res+p;
}
```

4.3.2 exCRT

```
namespace exCRT
{
    ll excrt(VL a,VL b)//res=a_i(mod b_i)
    {
        ll x,y,k,g,c,p,res,bg;
        assert(sz(a)==sz(b));
        assert(sz(a)>0);
        p=b[0];
        res=a[0];
        for(int i=1;i<sz(a);i++)
        {
            c=(a[i]-res%b[i]+b[i])%b[i];
            g=exgcd(p,b[i],x,y);
            bg=b[i]/g;
            if(c%g!=0) return -1;
            x=(x*(c/g))%bg;
            res+=x*p;
```

```
            p*=bg;
            res=(res%p+p)%p;
        }
        return (res%p+p)%p;
    }
};
```

4.4 组合数

4.4.1 打表

```
ll C[1010][1010];
void init(int n)
{
    int i,j;
    for(i=(C[0][0]=1);i<=n;i++)
    {
        for(j=(C[i][0]=1);j<=n;j++)
        {
            C[i][j]=(C[i-1][j]+C[i-1][j-1])%mod;
        }
    }
}
```

4.4.2 预处理

```
ll pow2(ll a,ll b)
{
    ll res=1;
    while(b)
    {
        if(b&1) res=res*a%mod;
        a=a*a%mod;
        b>>=1;
    }
    return res;
}
ll inv(ll x){return pow2(x,mod-2);}
ll fac[MAX],invfac[MAX];
void init(int n)
{
    fac[0]=1;
    for(int i=1;i<=n;i++) fac[i]=fac[i-1]*i%mod;
    invfac[n]=inv(fac[n]);
    for(int i=n-1;~i;i--) invfac[i]=invfac[i+1]*(i
        +1)%mod;
}
ll C(ll n,ll m)
{
    if(m>n||m<0||n<0) return 0;
    return fac[n]*invfac[m]%mod*invfac[n-m]%mod;
}
```

4.4.3 Lucas 定理

```
//C(n,m) n,m<=1e18 p<=1e5
//p must be a prime number
ll Lucas(ll n,ll m,ll p)
{
    if(m==0) return 1;
    return C(n%p,m%p)*Lucas(n/p,m/p,p)%p;
}
```

4.4.4 exLucas

```
namespace exLucas
{
    ll pow2(ll a,ll b,ll p)
    {
        ll res=1;
        while(b>0)
        {
            if(b&1) res=res*a%p;
            a=a*a%p;
            b>>=1;
        }
        return res;
    }
    ll inv(ll a,ll p)
    {
        ll g,x,y,res;
        g=exgcd(a,p,x,y);
        res=(g==1?(x+p)%p:-1);
        assert(res!=-1);
        return res;
    }
    map<ll,pair<VL,VL> > mp;
    map<PLL,VL > fac;
    void init(VL mod_list)
    {
        ll i,j,p;
        mp.clear();
        fac.clear();
        for(auto mod_i:mod_list)
        {
            p=mod_i;
            VL a,b;
            for(i=2;i*i<=p;i++)
            {
                if(p%i) continue;
                b.pb(1LL);
                while(p%i==0) b[sz(b)-1]*=i,p/=i;
                a.pb(i);
            }
            if(p>1) a.pb(p),b.pb(p);
            mp[mod_i]=MP(a,b);
            for(i=0;i<sz(a);i++)
```

```
        {
            if(fac.count(MP(a[i],b[i]))) continue;
            VL fac_tmp=VL(b[i]+1);
            fac_tmp[0]=1;
            for(j=1;j<=b[i];j++)
            {
                if(j%a[i]) fac_tmp[j]=fac_tmp[j-1]*
                    j%b[i];
                else fac_tmp[j]=fac_tmp[j-1];
            }
            fac[MP(a[i],b[i])]=fac_tmp;
        }
    }
}
ll cal_fac(ll n,ll x,ll p)
{
    if(!n) return 1LL;
    ll res=1;
    assert(fac.count(MP(x,p)));
    res=res*fac[MP(x,p)][p-1]%p;
    res=pow2(res,n/p,p);
    res=res*fac[MP(x,p)][n%p]%p;
    return res*cal_fac(n/x,x,p)%p;
}
ll multilucas(ll n,ll m,ll x,ll p)
{
    if(m>n) return 0;
    ll i,cnt;
    cnt=0;
    for(i=n;i;i/=x) cnt+=i/x;
    for(i=m;i;i/=x) cnt-=i/x;
    for(i=n-m;i;i/=x) cnt-=i/x;
    return pow2(x,cnt,p)* \
          cal_fac(n,x,p)%p* \
          inv(cal_fac(m,x,p),p)%p* \
          inv(cal_fac(n-m,x,p),p)%p;
}
ll C(ll n,ll m,ll p)
{
    if(m>n||m<0||n<0) return 0;
    ll i,res;
    VL a,b,resa;
    assert(mp.count(p));
    a=mp[p].fi;
    b=mp[p].se;
    for(i=0;i<sz(a);i++) resa.pb(multilucas(n,m,a
        [i],b[i]));
    res=exCRT::excrt(resa,b);
    assert(res!=-1);
    return res;
}
};//exLucas::init(VL{});
```

4.5 欧拉函数

<=n 且与 n 互质的数的和：n*phi[n]/2

4.5.1 直接求

```
//O(sqrt(n))
int euler(int n)
{
    int ans,i;
    ans=n;
    for(i=2;i*i<=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;
}
```

4.5.2 线性筛

```
int prime[MAX],phi[MAX],tot;
bool flag[MAX];
void init(int n)
{
    int i,j,k;
    tot=0;
    mem(flag,0);
    phi[0]=0;
    phi[1]=1;
    for(i=2;i<=n;i++)
    {
        if(!flag[i])
        {
            prime[tot++]=i;
            phi[i]=i-1;
        }
        for(j=0;j<tot&&i*prime[j]<=n;j++)
        {
            k=i*prime[j];
            flag[k]=1;
            if(i%prime[j]==0)
            {
                phi[k]=phi[i]*prime[j];
                break;
            }
            else phi[k]=phi[i]*(prime[j]-1);
        }
    }
```

```
}
```

4.6 莫比乌斯函数

```
int mo[MAX],prime[MAX],tot;
bool flag[MAX];
void initmo(int n)
{
    int i,j;
    mem(flag,0);
    mem(mo,0);
    tot=0;
    mo[1]=1;
    for(i=2;i<=n;i++)
    {
        if(!flag[i])
        {
            prime[tot++]=i;
            mo[i]=-1;
        }
        for(j=0;j<tot&&prime[j]*i<=n;j++)
        {
            flag[i*prime[j]]=1;
            if(i%prime[j]==0)
            {
                mo[prime[j]*i]=0;
                break;
            }
            mo[prime[j]*i]=-mo[i];
        }
    }
}
```

4.7 Berlekamp-Massey

```
//Berlekamp-Massey
typedef vector<int> VI;
namespace linear_seq
{
    #define rep(i,a,n) for (int i=a;i<n;i++)
    #define SZ(x) ((int)(x).size())
    const ll mod=1e9+7;
    ll powmod(ll a,ll b){ll res=1;a%=mod; assert(b>=0); for(;b;b>>=1){if(b&1)res=res*a%mod;a=a*a%mod;}return res;}
    const int N=10010;
    ll res[N],base[N],_c[N],_md[N];
    vector<int> Md;
    void mul(ll *a,ll *b,int k)
    {
        rep(i,0,k+k) _c[i]=0;
        rep(i,0,k) if (a[i]) rep(j,0,k) _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
        for (int i=k+k-1;i>=k;i--) if (_c[i])
            rep(j,0,SZ(Md)) _c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%mod;
        rep(i,0,k) a[i]=_c[i];
    }
    int solve(ll n,VI a,VI b){
        ll ans=0,pnt=0;
        int k=SZ(a);
        assert(SZ(a)==SZ(b));
        rep(i,0,k) _md[k-1-i]=-a[i];_md[k]=1;
        Md.clear();
        rep(i,0,k) if (_md[i]!=0) Md.push_back(i);
        rep(i,0,k) res[i]=base[i]=0;
        res[0]=1;
        while ((1ll<<pnt)<=n) pnt++;
        for (int p=pnt;p>=0;p--) {
            mul(res,res,k);
            if ((n>>p)&1) {
                for (int i=k-1;i>=0;i--) res[i+1]=res[i];res[0]=0;
                rep(j,0,SZ(Md)) res[Md[j]]=(res[Md[j]]-res[k]*_md[Md[j]])%mod;
            }
        }
        rep(i,0,k) ans=(ans+res[i]*b[i])%mod;
        if (ans<0) ans+=mod;
        return ans;
    }
    VI BM(VI s){
        VI C(1,1),B(1,1);
        int L=0,m=1,b=1;
        rep(n,0,SZ(s)){
            ll d=0;
            rep(i,0,L+1) d=(d+(ll)C[i]*s[n-i])%mod;
            if(d==0) ++m;
            else if(2*L<=n){
                VI T=C;
                ll c=mod-d*powmod(b,mod-2)%mod;//□□□
                while (SZ(C)<SZ(B)+m) C.pb(0);
                rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
                L=n+1-L; B=T; b=d; m=1;
            } else {
                ll c=mod-d*powmod(b,mod-2)%mod;//□□□
                while (SZ(C)<SZ(B)+m) C.pb(0);
                rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
                ++m;
            }
        }
        return C;
    }
    int gao(VI a,ll n)
    {
```

```
        VI c=BM(a);
        c.erase(c.begin());
        rep(i,0,SZ(c)) c[i]=(mod-c[i])%mod;
        return solve(n,c,VI(a.begin(),a.begin()+SZ(c)
            ));
    }
};//linear_seq::gao(VI{},n-1)
```

4.8 exBSGS

```
//a^x≡b (mod c)
ll exBSGS(ll a,ll b,ll c)
{
    ll i,g,d,num,now,sq,t,x,y;
    if(c==1) return b?-1:(a!=1);
    if(b==1) return a?0:-1;
    if(a%c==0) return b?-1:1;
    num=0;
    d=1;
    while((g=__gcd(a,c))>1)
    {
        if(b%g) return -1;
        num++;
        b/=g;
        c/=g;
        d=(d*a/g)%c;
        if(d==b) return num;
    }
    mp.clear();
    sq=ceil(sqrt(c));
    t=1;
    for(i=0;i<sq;i++)
    {
        if(!mp.count(t)) mp[t]=i;
        else mp[t]=min(mp[t],i);
        t=t*a%c;
    }
    for(i=0;i<sq;i++)
    {
        exgcd(d,c,x,y);
        x=(x*b%c+c)%c;
        if(mp.count(x)) return i*sq+mp[x]+num;
        d=d*t%c;
    }
    return -1;
}
```

4.9 Miller_Rabin+Pollard_rho

```
const int S=20;
mt19937 rd(time(0));
ll mul2(ll a,ll b,ll p)
{
    ll res=0;
    while(b)
    {
        if(b&1) res=(res+a)%p;
        a=(a+a)%p;
        b>>=1;
    }
    return res;
}
ll pow2(ll a,ll b,ll p)
{
    ll res=1;
    while(b)
    {
        if(b&1) res=mul2(res,a,p);
        a=mul2(a,a,p);
        b>>=1;
    }
    return res;
}
int check(ll a,ll n,ll x,ll t)//一定是合数返回不一定返回1,0
{
    ll now,nex,i;
    now=nex=pow2(a,x,n);
    for(i=1;i<=t;i++)
    {
        now=mul2(now,now,n);
        if(now==1&&nex!=1&&nex!=n-1) return 1;
        nex=now;
    }
    if(now!=1) return 1;
    return 0;
}
int Miller_Rabin(ll n)
{
    if(n<2) return 0;
    if(n==2) return 1;
    if((n&1)==0) return 0;
    ll x,t,i;
    x=n-1;
    t=0;
    while((x&1)==0) x>>=1,t++;
    for(i=0;i<S;i++)
    {
        if(check(rd()%(n-1)+1,n,x,t)) return 0;
    }
    return 1;
}
ll Pollard_rho(ll x,ll c)
{
    ll i,k,g,t,y;
    i=1;
    k=2;
```

```
    y=t=rd()%x;
    while(1)
    {
        i++;
        t=(mul2(t,t,x)+c)%x;
        g=__gcd(y-t+x,x);
        if(g!=1&&g!=x) return g;
        if(y==t) return x;
        if(i==k)
        {
            y=t;
            k+=k;
        }
    }
}
vector<ll> fac;
void findfac(ll n)
{
    if(Miller_Rabin(n))
    {
        fac.pb(n);
        return;
    }
    ll t=n;
    while(t>=n) t=Pollard_rho(t,rd()%(n-1)+1);
    findfac(t);
    findfac(n/t);
}
void work(ll x)
{
    fac.clear();
    findfac(x);
}
```

4.10 第二类 Stirling 数

```
//dp[i][j表示]个元素划分到个不可区分的非空盒子里的方案数。ik
ll dp[MAX][MAX];
void init()
{
    ll i,j;
    mem(dp,0);
    dp[1][1]=1;
    for(i=2;i<MAX;i++)
    {
        for(j=1;j<=i;j++)
        {
            dp[i][j]=(dp[i-1][j-1]+j*dp[i-1][j])%mod;
        }
    }
}
```

4.11 原根

原根性质

1. 一个数 m 如果有原根，则其原根个数为 $phi[phi[m]]$。若 m 为素数，则其原根个数为 $phi[phi[m]] = phi[m-1]$。
2. 有原根的数只有 $2, 4, p^n, 2*p^n$ (p 为质数,n 为正整数)
3. 一个数的最小原根的大小是 $O(n^{0.25})$ 的
4. 如果 g 为 n 的原根，则 g^d 为 n 的原根的充要条件是 $gcd(d, phi[n]) = 1$

指标法则

1. $I(a*b)\ I(a)+I(b)(modp-1)$
2. $I(a^k)\ k*I(a)(modp-1)$

```
int p[MAX],tot,prime[MAX];
void init(int n)
{
    int i,j;
    tot=0;
    mem(prime,0);
    prime[1]=1;
    for(i=2;i<=n;i++)
    {
        if(!prime[i]) prime[i]=p[tot++]=i;
        for(j=0;j<tot&&p[j]*i<=n;j++)
        {
            prime[i*p[j]]=p[j];
            if(i%p[j]==0) break;
        }
    }
}
ll pow2(ll a,ll b,ll p)
{
    ll res=1;
    while(b)
    {
        if(b&1) res=res*a%p;
        a=a*a%p;
        b>>=1;
    }
    return res;
}
int tp[MAX];
int find_root(int x)//求素数原根
{
    if(x==2) return 1;
    int f,phi=x-1;
    tp[0]=0;
    for(int i=0;phi&&i<tot;i++)
    {
        if(phi%p[i]==0)
```

```
        {
            tp[++tp[0]]=p[i];
            while(phi%p[i]==0) phi/=p[i];
        }
    }
    if(phi!=1) tp[++tp[0]]=phi;
    phi=x-1;
    for(int g=2;g<=x-1;g++)
    {
        f=1;
        for(int i=1;i<=tp[0];i++)
        {
            if(pow2(g,phi/tp[i],x)==1)
            {
                f=0;
                break;
            }
        }
        if(f) return g;
    }
    return 0;
}
int I[MAX];
void get_I(int p)//求指标表
{
    int g,now;
    g=find_root(p);
    now=1;
    for(int i=1;i<p;i++)
    {
        now=now*g%p;
        I[now]=i;
    }
}
```

4.12 二次剩余

```
//O(log^2)
struct Tonelli_Shanks
{
    ll mul2(ll a,ll b,ll p)
    {
        ll res=0;
        a%=p;
        while(b)
        {
            if(b&1)
            {
                res+=a;
                if(res>=p) res-=p;
            }
            a+=a;
            if(a>=p) a-=p;
            b>>=1;
        }
        return res;
    }
    ll pow2(ll a,ll b,ll p)
    {
        ll res=1;
        while(b)
        {
            if(b&1) res=mul2(res,a,p);
            a=mul2(a,a,p);
            b>>=1;
        }
        return res;
    }
    ll sqrt(ll n,ll p)
    {
        if(p==2) return (n&1)?1:-1;
        if(pow2(n,p>>1,p)!=1) return -1;
        if(p&2) return pow2(n,(p+1)>>2,p);
        ll q,z,c,r,t,tmp,s,i,m;
        s=__builtin_ctzll(p^1);
        q=p>>s;
        z=2;
        for(;pow2(z,p>>1,p)==1;z++);
        c=pow2(z,q,p);
        r=pow2(n,(q+1)>>1,p);
        t=pow2(n,q,p);
        for(m=s;t!=1;)
        {
            for(i=0,tmp=t;tmp!=1;i++) tmp=tmp*tmp%p;
            for(;i<--m;) c=c*c%p;
            r=r*c%p;
            c=c*c%p;
            t=t*c%p;
        }
        return r;
    }
}ts;
```

5 多项式

5.1 FFT

```
namespace FFT
{
    #define rep(i,a,b) for(int i=(a);i<=(b);i++)
    const double pi=acos(-1);
    const int maxn=(1<<19)+10;
    struct cp
    {
        double a,b;
        cp(){}
```

```
    cp(double _x,double _y){a=_x,b=_y;}
    cp operator +(const cp &o)const{return (cp){a+o.a,b+o.b};}
    cp operator -(const cp &o)const{return (cp){a-o.a,b-o.b};}
    cp operator *(const cp &o)const{return (cp){a*o.a-b*o.b,b*o.a+a*o.b};}
    cp operator *(const double &o)const{return (cp){a*o,b*o};}
    cp operator !()const{return (cp){a,-b};}
}x[maxn],y[maxn],z[maxn],w[maxn];
void fft(cp x[],int k,int v)
{
    int i,j,l;
    for(i=0,j=0;i<k;i++)
    {
        if(i>j)swap(x[i],x[j]);
        for(l=k>>1;(j^=l)<l;l>>=1);
    }
    w[0]=(cp){1,0};
    for(i=2;i<=k;i<<=1)
    {
        cp g=(cp){cos(2*pi/i),(v?-1:1)*sin(2*pi/i)};
        for(j=(i>>1);j>=0;j-=2)w[j]=w[j>>1];
        for(j=1;j<i>>1;j+=2)w[j]=w[j-1]*g;
        for(j=0;j<k;j+=i)
        {
            cp *a=x+j,*b=a+(i>>1);
            for(l=0;l<i>>1;l++)
            {
                cp o=b[l]*w[l];
                b[l]=a[l]-o;
                a[l]=a[l]+o;
            }
        }
    }
    if(v)for(i=0;i<k;i++)x[i]=(cp){x[i].a/k,x[i].b/k};
}

//多项式与多项式相乘bc 结果存在a
//为的长度l1b-1 为的长度l2c-1
void mul(int *a,int *b,int *c,int l1,int l2)
{
    if(l1<128&&l2<128)
    {
        rep(i,0,l1+l2)a[i]=0;
        rep(i,0,l1)rep(j,0,l2)a[i+j]+=b[i]*c[j];
        return;
    }
    int K;
    for(K=1;K<=l1+l2;K<<=1);
    rep(i,0,l1)x[i]=cp(b[i],0);
    rep(i,0,l2)y[i]=cp(c[i],0);
    rep(i,l1+1,K)x[i]=cp(0,0);
    rep(i,l2+1,K)y[i]=cp(0,0);
    fft(x,K,0);fft(y,K,0);
    rep(i,0,K)z[i]=x[i]*y[i];
    fft(z,K,1);
    rep(i,0,l1+l2)a[i]=(ll)(z[i].a+0.5);
}
};
```

5.2 NTT

```
namespace NTT
{
    const int g=3;
    const int p=998244353;
    int wn[35];
    int pow2(int a,int b)
    {
        int res=1;
        while(b)
        {
            if(b&1) res=1ll*res*a%p;
            a=1ll*a*a%p;
            b>>=1;
        }
        return res;
    }
    void getwn()
    {
        for(int i=0;i<25;i++) wn[i]=pow2(g,(p-1)/(1LL<<i));
    }
    void ntt(VI &a,int len,int f)
    {
        int i,j=0,t,k,w,id;
        for(i=1;i<len-1;i++)
        {
            for(t=len;j^=t>>=1,~j&t;);
            if(i<j) swap(a[i],a[j]);
        }
        for(i=1,id=1;i<len;i<<=1,id++)
        {
            t=i<<1;
            for(j=0;j<len;j+=t)
            {
                for(k=0,w=1;k<i;k++,w=1ll*w*wn[id]%p)
                {
                    int x=a[j+k],y=1ll*w*a[j+k+i]%p;
                    a[j+k]=x+y;
                    if(a[j+k]>=p) a[j+k]-=p;
                    a[j+k+i]=x-y;
                    if(a[j+k+i]<0) a[j+k+i]+=p;
```

```
            }
        }
    }
    if(f)
    {
        for(i=1,j=len-1;i<j;i++,j--) swap(a[i],a[j]);
        int inv=pow2(len,p-2);
        for(i=0;i<len;i++) a[i]=1ll*a[i]*inv%p;
    }
}
void qpow(VI &a,int b)//limt: sz(a)*b is small
{
    int len,i,l1;
    l1=sz(a);
    for(len=1;len<=(l1+1)*b-1;len<<=1);
    a.resize(len+1);
    for(i=l1;i<=len;i++) a[i]=0;
    ntt(a,len,0);
    for(i=0;i<len;i++) a[i]=pow2(a[i],b);
    ntt(a,len,1);
    a.resize((l1+1)*b-1);
}
void mul(VI &res,VI a,VI b)
{
    int len,i,l1,l2;
    l1=sz(a);
    l2=sz(b);
    for(len=1;len<=l1+l2;len<<=1);
    a.resize(len+1);
    b.resize(len+1);
    for(i=l1+1;i<=len;i++) a[i]=0;
    for(i=l2+1;i<=len;i++) b[i]=0;
    ntt(a,len,0);ntt(b,len,0);
    res.resize(len);
    for(i=0;i<len;i++) res[i]=1ll*a[i]*b[i]%p;
    ntt(res,len,1);
    res.resize(l1+l2-1);
}
};//NTT::getwn();
```

5.3 FWT

```
namespace FWT
{
    ll inv2;//对2的逆元p
    const ll p=1e9+7;
    ll pow2(ll a,ll b)
    {
        ll res=1;
        while(b)
        {
            if(b&1) res=res*a%p;
            a=a*a%p;
            b>>=1;
        }
        return res;
    }
    void fwt(ll *a,int n,int f,int v)
    {
        for(int d=1;d<n;d<<=1)
        {
            for(int m=d<<1,i=0;i<n;i+=m)
            {
                for(int j=0;j<d;j++)
                {
                    ll x=a[i+j],y=a[i+j+d];
                    if(!v)
                    {
                        if(f==1) a[i+j]=(x+y)%p,a[i+j+d]=(x-y+p)%p;//xor
                        else if(f==2) a[i+j]=(x+y)%p;//and
                        else if(f==3) a[i+j+d]=(x+y)%p;//or
                    }
                    else
                    {
                        if(f==1) a[i+j]=(x+y)*inv2%p,a[i+j+d]=(x-y+p)%p*inv2%p;//xor
                        else if(f==2) a[i+j]=(x-y+p)%p;//and
                        else if(f==3) a[i+j+d]=(y-x+p)%p;//or
                    }
                }
            }
        }
    }

    //结果存在a
    void XOR(ll *a,ll *b,int n)
    {
        int len;
        for(len=1;len<=n;len<<=1);
        fwt(a,len,1,0);
        fwt(b,len,1,0);
        for(int i=0;i<len;i++) a[i]=a[i]*b[i]%p;
        inv2=pow2(2,p-2);
        fwt(a,len,1,1);
    }
    void AND(ll *a,ll *b,int n)
    {
        int len;
        for(len=1;len<=n;len<<=1);
        fwt(a,len,2,0);
```

```
        fwt(b,len,2,0);
        for(int i=0;i<len;i++) a[i]=a[i]*b[i]%p;
        fwt(a,len,2,1);
    }
    void OR(ll *a,ll *b,int n)
    {
        int len;
        for(len=1;len<=n;len<<=1);
        fwt(a,len,3,0);
        fwt(b,len,3,0);
        for(int i=0;i<len;i++) a[i]=a[i]*b[i]%p;
        fwt(a,len,3,1);
    }
};
```

5.4 拉格朗日插值

```
namespace polysum {
    #define rep(i,a,n) for (int i=a;i<n;i++)
    #define per(i,a,n) for (int i=n-1;i>=a;i--)
    const int D=101000;
    ll a[D],tmp[D],f[D],g[D],p[D],p1[D],p2[D],b[D],h[D][2],C[D];
    ll powmod(ll a,ll b){ll res=1;a%=mod;assert(b>=0);for(;b;b>>=1){if(b&1)res=res*a%mod;a=a*a%mod;}return res;}
    ll calcn(int d,ll *a,ll n) { // a[0].. a[d] a[n]
        if (n<=d) return a[n];
        p1[0]=p2[0]=1;
        rep(i,0,d+1) {
            ll t=(n-i+mod)%mod;
            p1[i+1]=p1[i]*t%mod;
        }
        rep(i,0,d+1) {
            ll t=(n-d+i+mod)%mod;
            p2[i+1]=p2[i]*t%mod;
        }
        ll ans=0;
        rep(i,0,d+1) {
            ll t=g[i]*g[d-i]%mod*p1[i]%mod*p2[d-i]%mod*a[i]%mod;
            if ((d-i)&1) ans=(ans-t+mod)%mod;
            else ans=(ans+t)%mod;
        }
        return ans;
    }
    void init(int M) {
        f[0]=f[1]=g[0]=g[1]=1;
        rep(i,2,M+5) f[i]=f[i-1]*i%mod;
        g[M+4]=powmod(f[M+4],mod-2);
        per(i,1,M+4) g[i]=g[i+1]*(i+1)%mod;
    }
    ll polysum(ll n,ll *a,ll m) { // a[0].. a[m] \sum_{i=0}^{n-1} a[i]
        rep(i,0,m+1) tmp[i]=a[i];
        tmp[m+1]=calcn(m,tmp,m+1);
        rep(i,1,m+2) tmp[i]=(tmp[i-1]+tmp[i])%mod;
        return calcn(m+1,tmp,n-1);
    }
    ll qpolysum(ll R,ll n,ll *a,ll m) { // a[0].. a[m] \sum_{i=0}^{n-1} a[i]*R^i
        if (R==1) return polysum(n,a,m);
        a[m+1]=calcn(m,a,m+1);
        ll r=powmod(R,mod-2),p3=0,p4=0,c,ans;
        h[0][0]=0;h[0][1]=1;
        rep(i,1,m+2) {
            h[i][0]=(h[i-1][0]+a[i-1])*r%mod;
            h[i][1]=h[i-1][1]*r%mod;
        }
        rep(i,0,m+2) {
            ll t=g[i]*g[m+1-i]%mod;
            if (i&1) p3=((p3-h[i][0]*t)%mod+mod)%mod,p4=((p4-h[i][1]*t)%mod+mod)%mod;
            else p3=(p3+h[i][0]*t)%mod,p4=(p4+h[i][1]*t)%mod;
        }
        c=powmod(p4,mod-2)*(mod-p3)%mod;
        rep(i,0,m+2) h[i][0]=(h[i][0]+h[i][1]*c)%mod;
        rep(i,0,m+2) C[i]=h[i][0];
        ans=(calcn(m,C,n)*powmod(R,n)-c)%mod;
        if (ans<0) ans+=mod;
        return ans;
    }
} // polysum::init();
```

6 矩阵

6.1 矩阵类

```
const ll mod=1e9+7;
struct Matrix
{
    ll c[6][6],n;
    Matrix(){}
    Matrix(ll a,ll v=0)
    {
        int i,j;
        n=a;
        for(i=0;i<n;i++)
        {
            for(j=0;j<n;j++)
            {
                c[i][j]=v;
            }
        }
```

```
    }
    Matrix operator *(const Matrix &b)const
    {
        int i,j,k;
        Matrix res(n);
        for(k=0;k<n;k++)
        {
            for(i=0;i<n;i++)
            {
                if(!c[i][k]) continue;
                for(j=0;j<n;j++)
                {
                    res.c[i][j]+=c[i][k]*b.c[k][j];
                    if(res.c[i][j]>=mod) res.c[i][j]%=
                        mod;
                }
            }
        }
        return res;
    }
};
Matrix matpow2(Matrix a,ll b)
{
    Matrix res(a.n);
    for(int i=0;i<a.n;i++)
    {
        res.c[i][i]=1;
    }
    while(b)
    {
        if(b&1) res=res*a;
        a=a*a;
        b>>=1;
    }
    return res;
}
```

6.2 高斯消元

6.2.1 同余方程

```
namespace Gauss
{
    int p;
    ll mp[905][905],sol[905];
    void set_mod(int _p)
    {
        p=_p;
        mem(mp,0);
        mem(sol,0);
    }
    ll pow2(ll a,ll b)
    {
        ll res=1;
        while(b)
        {
            if(b&1) res=res*a%p;
            a=a*a%p;
            b>>=1;
        }
        return res;
    }
    ll inv(ll x){return pow2(x,p-2);}
    ll lcm(ll a,ll b){return a/__gcd(a,b)*b;}
    int gauss(int n,int m)
    {
        int r,c,id,i,j;
        ll tmp,ta,tb;
        r=c=0;
        while(r<n&&c<m)
        {
            id=r;
            for(i=r+1;i<n;i++)
            {
                if(abs(mp[i][c])>abs(mp[id][c])) id=i;
            }
            if(id!=r)
            {
                for(i=0;i<=m;i++) swap(mp[r][i],mp[id
                    ][i]);
            }
            if(abs(mp[r][c])!=0)
            {
                for(i=r+1;i<n;i++)
                {
                    if(abs(mp[i][c])==0) continue;
                    tmp=lcm(abs(mp[i][c]),abs(mp[r][c])
                        );
                    ta=tmp/abs(mp[i][c]);
                    tb=tmp/abs(mp[r][c]);
                    if(mp[i][c]*mp[r][c]<0) tb=-tb;
                    for(j=c;j<=m;j++)
                    {
                        mp[i][j]=(mp[i][j]*ta-mp[r][j]*
                            tb)%p;
                        if(mp[i][j]<0) mp[i][j]+=p;
                    }
                }
                r++;
            }
            c++;
        }
        for(i=r;i<n;i++)
        {
            if(mp[i][m]!=0) return -1;//no solution
        }
    // if(r<m) return m-r;//multi solution
        for(i=m-1;~i;i--)
```

```
        {
            tmp=mp[i][m];
            for(j=i+1;j<m;j++)
            {
                if(mp[i][j]==0) continue;
                tmp=(tmp-mp[i][j]*sol[j])%p;
                if(tmp<0) tmp+=p;
            }
            sol[i]=tmp*inv(mp[i][i])%p;
        }
        return 0;
    }
}
using namespace Gauss;
//set_mod();
```

6.2.2 同余方程 mod=2

```
//同余方程 mod时=2 异或加速O(n*n*m)
int mat[22][MAX];
int Gauss(int n,int m)//是未知数个数n 是方程个数m
{
    int i,j;
    for(i=1,j=1;i<=n&&j<=m;j++)
    {
        int k=i;
        while(k<=n&&!mat[k][j]) k++;
        if(mat[k][j])
        {
            for(int r=1;r<=m+1;r++) swap(mat[i][r],
                mat[k][r]);
            for(int r=1;r<=n;r++)
            {
                if(r!=i&&mat[r][j])
                {
                    for(k=1;k<=m+1;k++)
                    {
                        mat[r][k]^=mat[i][k];
                    }
                }
            }
            i++;
        }
    }
    //第m列是等号右边+1
    for(j=i;j<=n;j++)
    {
        if(mat[j][m+1]) return -1;//无解
    }
    return m-i+1;//返回解的个数
}
```

6.3 单纯形

```
typedef double db;
typedef vector<db> VD;
typedef vector<VD> VVD;
typedef vector<int> VI;
struct Simplex
{
    int m,n;
    VI B,N;
    VVD D;
    Simplex(){}
    Simplex(const VVD &A,const VD &b,const VD &c):m(
        sz(b)),n(sz(c)),N(n+1),B(m),D(m+2,VD(n+2))
    {
        int i,j;
        for(i=0;i<m;i++)
        {
            for(j=0;j<n;j++)
            {
                D[i][j]=A[i][j];
            }
        }
        for(i=0;i<m;i++)
        {
            B[i]=n+i;
            D[i][n]=-1;
            D[i][n+1]=b[i];
        }
        for(j=0;j<n;j++)
        {
            N[j]=j;
            D[m][j]=-c[j];
        }
        N[n]=-1;
        D[m+1][n]=1;
    }
    void Pivot(int r,int s)
    {
        int i,j;
        for(i=0;i<m+2;i++)
        {
            if(i==r) continue;
            for(j=0;j<n+2;j++)
            {
                if(j==s) continue;
                D[i][j]-=D[r][j]*D[i][s]/D[r][s];
            }
        }
        for(j=0;j<n+2;j++)
        {
            if (j!=s) D[r][j]/=D[r][s];
        }
        for(i=0;i<m+2;i++)
```

```
    {
        if(i!=r) D[i][s]/=-D[r][s];
    }
    D[r][s]=1.0/D[r][s];
    swap(B[r],N[s]);
}
bool simplex(int phase)
{
    int i,j,s,r;
    int x=phase==1?m+1:m;
    while(1)
    {
        s=-1;
        for(j=0;j<=n;j++)
        {
            if(phase==2&&N[j]==-1) continue;
            if(s==-1||D[x][j]<D[x][s]||D[x][j]==D[x][s]&&N[j]<N[s]) s=j;
        }
        if(D[x][s]>-eps) return 1;
        r=-1;
        for(i=0;i<m;i++)
        {
            if(D[i][s]<eps) continue;
            if(r==-1||D[i][n+1]/D[i][s]<D[r][n+1]/D[r][s]) r=i;
            if(D[i][n+1]/D[i][s]==D[r][n+1]/D[r][s]&&B[i]<B[r]) r=i;
        }
        if(r==-1) return 0;
        Pivot(r,s);
    }
}
db work(VD &res)
{
    int i,j,k,r,s;
    r=0;
    for(i=1;i<m;i++)
    {
        if(D[i][n+1]<D[r][n+1]) r=i;
    }
    if(D[r][n+1]<-eps)
    {
        Pivot(r,n);
        if(!simplex(1)||D[m+1][n+1]<-eps) return -numeric_limits<db>::infinity();//no solution
        for(i=0;i<m;i++)
        {
            if(B[i]!=-1) continue;
            s=-1;
            for(j=0;j<=n;j++)
            {
                if(s==-1||D[i][j]<D[i][s]||D[i][j]==D[i][s]&&N[j]<N[s]) s=j;
            }
            Pivot(i,s);
        }
    }
    if(!simplex(2)) return numeric_limits<db>::infinity();//solution is INF
    res=VD(n);
    for(i=0;i<m;i++)
    {
        if(B[i]<n) res[B[i]]=D[i][n+1];
    }
    return D[m][n+1];
}
};
/*
sum(A[i]*res[i])<=B,res[i]>=0
MAX(sum(C[i]*res[i]))
*/
```

6.4 线性基

```
struct Base
{
    #define type ll
    #define mx 60
    type d[mx+3];
    int p[mx+3],cnt;
    void init()
    {
        memset(d,0,sizeof(d));
        cnt=0;
    }
    bool insert(type x,int pos=0)
    {
        int i;
        for(i=mx;~i;i--)
        {
            if(!(x&(1LL<<i))) continue;
            if(!d[i])
            {
                cnt++;
                d[i]=x;
                p[i]=pos;
                break;
            }
            if(p[i]<pos)
            {
                swap(d[i],x);
                swap(p[i],pos);
            }
            x^=d[i];
```

```
    }
    return x>0;
}
type query_max(int pos=-1)
{
    int i;
    type res=0;
    for(i=mx;~i;i--)
    {
        if(p[i]>=pos)
        {
            if((res^d[i])>res) res^=d[i];
        }
    }
    return res;
}
type query_min(int pos=-1)
{
    for(int i=0;i<=mx;i++)
    {
        if(d[i]&&p[i]>=pos) return d[i];
    }
    return 0;
}
void merge(Base x)
{
    if(cnt<x.cnt)
    {
        swap(cnt,x.cnt);
        swap(d,x.d);
        swap(p,x.p);
    }
    for(int i=mx;~i;i--)
    {
        if(x.d[i]) insert(x.d[i]);
    }
}
//kth min
//first use rebuild()
type tp[mx+3];
void rebuild()
{
    int i,j;
    cnt=0;
    for(i=mx;~i;i--)
    {
        for(j=i-1;~j;j--)
        {
            if(d[i]&(1LL<<j)) d[i]^=d[j];
        }
    }
    for(i=0;i<=mx;i++)
    {
        if(d[i]) tp[cnt++]=d[i];
    }
    for(int i=mx;~i;i--)
    {
        if(d[i])
        {
            for(int j=i-1;~j;j--)
            {
                if(d[j]&(1LL<<j)) d[i]^=d[j];
            }
        }
    }
    type kth(type k)
    {
        type res=0;
        if(k>=(1LL<<cnt)) return -1;
        for(int i=mx;~i;i--)
        {
            if(k&(1LL<<i)) res^=tp[i];
        }
        return res;
    }
};
```

7 博弈

7.1 SG 函数

f[m]: 可改变当前状态的方式，N 为方式的种类，要先从小到大 sort

sg[]:sg 表

flag[m]: 为 x 后继状态的集合

7.1.1 sg 表

```
int f[111],sg[MAX];
void SG(int n,int m)
{
    int i,j,flag[111];
    mem(sg,0);
    for(i=1;i<=n;i++)
    {
        mem(flag,0);
        for(j=0;f[j]<=i&&j<m;j++)
        {
            flag[sg[i-f[j]]]=1;
        }
        for(j=0;;j++)
        {
            if(!flag[j])
            {
                sg[i]=j;
                break;
            }
        }
    }
}
```

7.1.2 记忆化搜索求 sg 函数

```
int f[105],sg[MAX],m;
int dfs(int x)
{
    int i,j,flag[105];
    if(sg[x]!=-1) return sg[x];
    mem(flag,0);
    for(i=1;i<=m;i++)
    {
        if(x>=f[i])
        {
            dfs(x-f[i]);
            flag[sg[x-f[i]]]=1;
        }
    }
    for(i=0;;i++)
    {
        if(!flag[i])
        {
            j=i;
            break;
        }
    }
    return sg[x]=j;
}
```

7.2 结论

1. 阶梯博弈

0 层为终点的阶梯博弈，等价于奇数层的 nim，偶数层的移动不影响结果

2.SJ 定理

对于任意一个 Anti-SG 游戏，如果我们规定当局面中所有的单一游戏的 SG 值为 0 时，游戏结束。

先手必胜当且仅当：

(1) 游戏的 SG 函数不为 0 且游戏中某个单一游戏的 SG 函数大于 1；

(2) 游戏的 SG 函数为 0 且游戏中没有单一游戏的 SG 函数大于 1。

8 dp

8.1 LIS

```
//最长上升子序列(>)nlogn 返回长度
//最长下降子序列(<) 把原数组取负数
int a[MAX],b[MAX];
int LIS(int n)
{
    int i;
    mem(b,0x3f);
    for(i=0;i<n;i++)
    {
        *lower_bound(b,b+n,a[i])=a[i];//最长不下降子序列(>=)改为upper_bound
    }
    return lower_bound(b,b+n,INF)-b;
}
```

8.2 LPS

```
//最长回文子序列
//dp[i][j] 表示s[i..j] 最长回文子序列的长度
int dp[2222][2222];
void LPS(char *s,int n)
{
    int i,j,len;
    for(i=1;i<=n;i++) dp[i][i]=1;
    for(len=2;len<=n;len++)
    {
        for(i=1;i<=n-len+1;i++)
        {
            j=i+len-1;
            if(s[i]==s[j]) dp[i][j]=dp[i+1][j-1]+2;
            else dp[i][j]=max({dp[i+1][j],dp[i][j-1],
                dp[i+1][j-1]});
        }
    }
}
```

8.3 数位 dp

```
const int DIG=20+2;
ll dp[DIG][2];
ll gao(ll x)
{
    const int base=10;
    int p[DIG],tot=0;
    if(x==-1) return 0;
    while(1)
    {
        p[tot++]=x%base;
        x/=base;
        if(!x) break;
    }
    function<ll(int,int,int,int)> dfs=[&](int pos,
        int lead,int sta,int limt)->ll
    {
        if(pos==-1) return ;
```

```
        if(!limt&&!lead&&dp[pos][sta]!=-1) return dp[pos][sta];
        ll res=0;
        for(int i=(limt?p[pos]:base-1);~i;i--)
        {

            res+=dfs(pos-1,lead&&i==0&&pos,,limt&&i==p[pos]);
        }
        if(!limt&&!lead) dp[pos][sta]=res;
        return res;
    };
    return dfs(tot-1,1,0,1);
}
```

9 杂项

9.1 FastIO

```
namespace fastIO{
    #define BUF_SIZE 100000
    #define OUT_SIZE 100000
    #define ll long long
    //fread->read
    bool IOerror=0;
// inline char nc(){char ch=getchar();if(ch==-1)IOerror=1;return ch;}
    inline char nc(){
        static char buf[BUF_SIZE],*p1=buf+BUF_SIZE,*pend=buf+BUF_SIZE;
        if(p1==pend){
            p1=buf;pend=buf+fread(buf,1,BUF_SIZE,stdin);
            if(pend==p1){IOerror=1;return -1;}
        }
        return *p1++;
    }
    inline bool blank(char ch){return ch==' '||ch=='\n'||ch=='\r'||ch=='\t';}
    template<class T> inline bool read(T &x){
        bool sign=0;char ch=nc();x=0;
        for(;blank(ch);ch=nc());
        if(IOerror)return false;
        if(ch=='-')sign=1,ch=nc();
        for(;ch>='0'&&ch<='9';ch=nc())x=x*10+ch-'0';
        if(sign)x=-x;
        return true;
    }
    inline bool read(double &x){
        bool sign=0;char ch=nc();x=0;
        for(;blank(ch);ch=nc());
        if(IOerror)return false;
        if(ch=='-')sign=1,ch=nc();
        for(;ch>='0'&&ch<='9';ch=nc())x=x*10+ch-'0';
        if(ch=='.'){
            double tmp=1; ch=nc();
            for(;ch>='0'&&ch<='9';ch=nc())tmp/=10.0,x+=tmp*(ch-'0');
        }
        if(sign)x=-x;
        return true;
    }
    inline bool read(char *s){
        char ch=nc();
        for(;blank(ch);ch=nc());
        if(IOerror)return false;
        for(;!blank(ch)&&!IOerror;ch=nc())*s++=ch;
        *s=0;
        return true;
    }
    inline bool read(char &c){
        for(c=nc();blank(c);c=nc());
        if(IOerror){c=-1;return false;}
        return true;
    }
    template<class T,class... U>bool read(T& h,U&...t){return read(h)&&read(t...);}
    //fwrite->print
    struct Ostream_fwrite{
        char *buf,*p1,*pend;
        Ostream_fwrite(){buf=new char[BUF_SIZE];p1=buf;pend=buf+BUF_SIZE;}
//  void out(char ch){putchar(ch);}
        void out(char ch){if(p1==pend){fwrite(buf,1,BUF_SIZE,stdout);p1=buf;}*p1++=ch;}
        template<class T>void print(T x){
            static char s[33],*s1;s1=s;
            if(!x)*s1++='0';if(x<0)out('-'),x=-x;
            while(x)*s1++=x%10+'0',x/=10;
            while(s1--!=s)out(*s1);
        }
        void print(double x,int y){
            static ll mul[]={
            {1,10,100,1000,10000,100000,1000000,10000000,100
            10000000000LL,100000000000LL
                ,1000000000000LL,10000000000000LL,
            100000000000000LL,1000000000000000LL
                ,10000000000000000LL
                ,100000000000000000LL};
            if(x<-1e-12)out('-'),x=-x;
            ll x2=(ll)floor(x);if(!y&&x-x2>=0.5)++x2;
                x-=x2;x*=mul[y];
            ll x3=(ll)floor(x);if(y&&x-x3>=0.5)++x3;
                print(x2);
            if(y>0){out('.');for(size_t i=1;i<y&&x3*
                mul[i]<mul[y];out('0'),++i);print(x3)
```

```
            ;}
74      }
75      void print(char *s){while(*s)out(*s++);}
76      void print(const char *s){while(*s)out(*s++)
            ;}
77      void flush(){if(p1!=buf){fwrite(buf,1,p1-buf,
            stdout);p1=buf;}}
78      ~Ostream_fwrite(){flush();}
79  }Ostream;
80  template<class T>void print(T x){Ostream.print(x
        );}
81  inline void print(char x){Ostream.out(x);}
82  inline void print(char *s){Ostream.print(s);}
83  inline void print(string s){Ostream.print(s.
        c_str());}
84  inline void print(const char *s){Ostream.print(s
        );}
85  inline void print(double x,int y){Ostream.print(
        x,y);}
86  template<class T,class... U>void print(const T&
        h,const U&... t){print(h);print(t...);}
87  void println(){print('\n');}
88  template<class T,class... U>void println(const T
        & h,const U&... t){print(h);println(t...);}
89  inline void flush(){Ostream.flush();}
90  #undef ll
91  #undef OUT_SIZE
92  #undef BUF_SIZE
93 };
94 using namespace fastIO;
```

9.2 O(1) 快速乘

```
1  ll mul2(ll x,ll y,ll p)
2  {
3      ll res=(x*y-ll((long double)x/p*y+1.0e-8)*p);
4      return res<0?res+p:res;
5  }
```

9.3 快速模

```
1  typedef long long i64;
2  typedef unsigned long long u64;
3  typedef __uint128_t u128;
4  const int word_bits=sizeof(u64)*8;
5  struct FastMod
6  {
7      static u64 mod,inv,r2;
8      u64 x;
9      FastMod():x(0){}
10     FastMod(u64 n):x(init(n)){}
11     static u64 modulus(){return mod;}
12     static u64 init(u64 w){return reduce(u128(w)*r2)
           ;}
13     static void set_mod(u64 m)
14     {
15         mod=m;
16         assert(mod&1);
17         inv=m;
18         for(int i=0;i<5;i++) inv*=2-inv*m;
19         r2=-u128(m)%m;
20     }
21     static u64 reduce(u128 x)
22     {
23         u64 y=u64(x>>word_bits)-u64((u128(u64(x)*inv)
               *mod)>>word_bits);
24         return i64(y)<0?y+mod:y;
25     }
26     FastMod& operator+=(FastMod rhs)
27     {
28         x+=rhs.x-mod;
29         if(i64(x)<0) x+=mod;
30         return *this;
31     }
32     FastMod operator+(FastMod rhs)const {return
           FastMod(*this)+=rhs;}
33     FastMod& operator*=(FastMod rhs)
34     {
35         x=reduce(u128(x)*rhs.x);
36         return *this;
37     }
38     FastMod operator*(FastMod rhs)const {return
           FastMod(*this)*=rhs;}
39     u64 get()const {return reduce(x);}
40 };
41 u64 FastMod::mod,FastMod::inv,FastMod::r2;
42 // FastMod::set_mod(p);
```

9.4 xor_sum(1,n)

```
1  ll xor_sum(ll n)
2  {
3      ll t=n&3;
4      if (t&1) return t/2ull^1;
5      return t/2ull^n;
6  }
```

9.5 约瑟夫环 kth

```
1  ll kth(ll n,ll m,ll k)
2  {
3      if(m==1) return k;
4      ll res=(m-1)%(n-k+1);
5      for(ll i=n-k+2,stp=0;i<=n;i+=stp,res+=stp*m)
6      {
```

```
        if(res+m>=i)
        {
            res=(res+m)%i;
            i++;
            stp=0;
        }
        else
        {
            stp=(i-res-2)/(m-1);
            if(i+stp>n)
            {
                res+=(n-(i-1))*m;
                break;
            }
        }
    }
    return res+1;
}
```

9.6 判断星期几

```
int judge(int y,int m,int d)
{
    int res;
    if(m==1||m==2) m+=12,y--;
    if((y<1752)||(y==1752&&m<9)||(y==1752&&m==9&&d
        <3)) res=(d+2*m+3*(m+1)/5+y+y/4+5)%7;
    else res=(d+2*m+3*(m+1)/5+y+y/4-y/100+y/400)%7;
    return res+1;
}
```

9.7 离散化

```
struct Discretization
{
    #define type ll
    vector<type> a;
    void init(){a.clear();}
    void add(type x){a.pb(x);}
    void work(){sort(all(a));a.resize(unique(all(a))
        -a.begin());}
    int get_pos(type x){return lower_bound(all(a),x)
        -a.begin()+1;}
    type get_val(int pos){return a[pos-1];}
    int size(){return a.size();}
    #undef type
}d;
```

9.8 网格整数点正方形个数

```
struct node
{
    int x,y;
    void input(){scanf("%d%d",&x,&y);}
}p[511];
int main()
{
    int n,i,j,ans;
    while(~scanf("%d",&n))
    {
        map<pair<int,int>,int> mp;
        for(i=0;i<n;i++)
        {
            p[i].input();
            mp[MP(p[i].x,p[i].y)]=1;
        }
        ans=0;
        for(i=0;i<n;i++)
        {
            for(j=i+1;j<n;j++)
            {
                int a,b,c,d,e,f,g,h;
                a=p[i].x;
                b=p[i].y;
                c=p[j].x;
                d=p[j].y;
                e=a+b+c-d;
                f=-a+b+c+d;
                g=a-b+c+d;
                h=a+b-c+d;
                if(abs(e%2)+abs(f%2)+abs(g%2)+abs(h%2)
                    ==0)
                {
                    if(mp[MP(e/2,f/2)]&&mp[MP(g/2,h/2)
                        ]) ans++;
                }
            }
        }
        printf("%d\n",ans/2);
    }
    return 0;
}
```

9.9 模拟退火

```
/*简单版
1. 模拟退火求费马点->复杂版

2.求矩形区域内一点到各点距离之和最短时间复杂度

 cnt*c1*c2*n

*/
int sgn(double x)
{
```

```
    if(fabs(x)<eps) return 0;
    else return x>0?1:-1;
}
struct Point
{
    double x,y;
    Point(){}
    Point(double a,double b)
    {
        x=a;
        y=b;
    }
    void input()
    {
        scanf("%lf%lf",&x,&y);
    }
};
typedef Point Vector;
Vector operator -(Vector a,Vector b){return Vector(
    a.x-b.x,a.y-b.y);}
double dot(Vector a,Vector b){return a.x*b.x+a.y*b.
    y;}
double dist(Point a,Point b){return sqrt(dot(a-b,a-
    b));}
double lx,ly;//矩形区域(0,0)-(lx,ly)
int check(double x,double y)
{
    if(sgn(x)<0||sgn(y)<0||sgn(x-lx)>0||sgn(y-ly)>0)
        return 1;
    return 0;
}
double Rand(double r,double l)
{
    return(rand()%((int)(l-r)*1000))/(1000.0+r);
}
double getres(Point t,Point *p,int n)//求距离之和
{
    double res=0;
    for(int i=0;i<n;i++)
    {
        res+=dist(t,p[i]);
    }
    return res;
}
pair<Point,double> SA(Point *p,int n)//模拟退火
{
    srand(time(0));//重置随机种子
    const double k=0.85;//退火常数
    const int c1=30;//随机取点的个数
    const int c2=50;//退火次数
    Point q[c1+10];//随机取点
    double dis[c1+10];//每个点的计算结果
    int i,j;
    for(i=1;i<=c1;i++)
    {
        q[i]=Point(Rand(0,lx),Rand(0,ly));
        dis[i]=getres(q[i],p,n);
    }
    double tmax=max(lx,ly);
    double tmin=1e-3;
// int cnt计算外层循环次数=0;//
    while(tmax>tmin)
    {
        for(i=1;i<=c1;i++)
        {
            for(j=1;j<=c2;j++)
            {
                double ang=Rand(0,2*PI);
                Point z;
                z.x=q[i].x+cos(ang)*tmax;
                z.y=q[i].y+sin(ang)*tmax;
                if(check(z.x,z.y)) continue;
                double temp=getres(z,p,n);
                if(temp<dis[i])
                {
                    dis[i]=temp;
                    q[i]=z;
                }
            }
        }
//  cnt++;
        tmax*=k;
    }
// cout<<cnt*c1*c2*n<<endl时间复杂度;//
    int pos=1;
    for(i=2;i<=c1;i++)
    {
        if(dis[i]<dis[pos])
        {
            pos=i;
        }
    }
    pair<Point,double> res;
    res=make_pair(q[pos],dis[pos]);
    return res;
}
```

9.10 矩形面积并

```
struct node
{
    ll l,r,h;
    int tag;
    friend bool operator <(node a,node b)
    {
        return a.h<b.h;
    }
```

```
}seg[MAX<<1];//线段
ll x[MAX<<1];//横坐标离散化
struct Segment_Tree
{
    #define ls (id<<1)
    #define rs (id<<1|1)
    ll n,ql,qr,qv;
    ll cover[MAX<<3],len[MAX<<3];//注意这里要开倍8
    void build(ll _n)
    {
        mem(cover,0);
        mem(len,0);
        n=_n;
    }
    void callen(int id,int l,int r)
    {
        if(cover[id]) len[id]=x[r+1]-x[l];//被整段覆盖
        else if(l==r) len[id]=0;//不是一条线段
        else len[id]=len[ls]+len[rs];//是一条线段但又没有被整段覆盖
    }
    void update(int l,int r,int id)
    {
        if(l>=ql&&r<=qr)
        {
            cover[id]+=qv;//覆盖情况
            callen(id,l,r);
            return;
        }
        int mid=(l+r)>>1;
        if(ql<=mid) update(l,mid,ls);
        if(qr>mid) update(mid+1,r,rs);
        callen(id,l,r);
    }
}tree;
int main()
{
    int n,i,tot,l,r,cnt;
    ll x1,y1,x2,y2,ans;
    while(~scanf("%d",&n)&&n)
    {
        tot=0;
        mem(x,0);
        for(i=0;i<n;i++)
        {
            scanf("%lld%lld%lld%lld",&x1,&y1,&x2,&y2);
            //矩形的左下和右上坐标
            x[tot]=x1;
            seg[tot].tag=-1;
            seg[tot].l=x1;
            seg[tot].r=x2;
            seg[tot++].h=y1;
            //上边界
            x[tot]=x2;
            seg[tot].tag=1;
            seg[tot].l=x1;
            seg[tot].r=x2;
            seg[tot++].h=y2;
            //下边界
        }
        sort(seg,seg+tot);//线段按纵坐标升序
        sort(x,x+tot);//横坐标升序
        cnt=unique(x,x+tot)-x;
        tree.build(cnt-1);
        ans=0;
        for(i=0;i<tot;i++)
        {
            if(i) ans+=(seg[i].h-seg[i-1].h)*tree.len[1];
            tree.ql=lower_bound(x,x+cnt-1,seg[i].l)-x;
            tree.qr=lower_bound(x,x+cnt-1,seg[i].r)-x-1;
            tree.qv=seg[i].tag;
            tree.update(0,cnt-1,1);
        }
        printf("%lld\n",ans);
    }
    return 0;
}
```

9.11 维护不同颜色最值和次值

```
struct Segment_Tree
{
    #define type int //may need change
    #define NONE INF //may need change
    #define ls (id<<1)
    #define rs (id<<1|1)
    int n,ql,qr;
    pair<type,type> a[MAX],mx[2][MAX<<2],mn[2][MAX<<2];
    pair<type,type> rmx[2],rmn[2];
    void upmx(int id)
    {
        type tx[4]={mx[0][id].fi,mx[1][id].fi,rmx[0].fi,rmx[1].fi};
        type tc[4]={mx[0][id].se,mx[1][id].se,rmx[0].se,rmx[1].se};
        int idx[4]={0,1,2,3},cnt=0;
        sort(idx,idx+4,[&](int a,int b){return tx[a]>tx[b];});
        for(int i=0;i<4&&cnt<2;i++)
        {
            if(i==0||tc[idx[i]]!=tc[idx[i-1]])
            {
```

```
            rmx[cnt].fi=tx[idx[i]];
            rmx[cnt++].se=tc[idx[i]];
        }
    }
    for(int i=cnt;i<2;i++) rmx[i]=MP(-NONE,-NONE)
        ;
}
void upmn(int id)
{
    type tx[4]={mn[0][id].fi,mn[1][id].fi,rmn[0].
        fi,rmn[1].fi};
    type tc[4]={mn[0][id].se,mn[1][id].se,rmn[0].
        se,rmn[1].se};
    int idx[4]={0,1,2,3},cnt=0;
    sort(idx,idx+4,[&](int a,int b){return tx[a]<
        tx[b];});
    for(int i=0;i<4&&cnt<2;i++)
    {
        if(i==0||tc[idx[i]]!=tc[idx[i-1]])
        {
            rmn[cnt].fi=tx[idx[i]];
            rmn[cnt++].se=tc[idx[i]];
        }
    }
    for(int i=cnt;i<2;i++) rmn[i]=MP(NONE,-NONE);
}
void pushup(int id)
{
    mx[0][id]=mx[0][ls];
    rmx[0]=mx[0][rs];
    mx[1][id]=mx[1][ls];
    rmx[1]=mx[1][rs];
    upmx(id);
    mx[0][id]=rmx[0];
    mx[1][id]=rmx[1];

    mn[0][id]=mn[0][ls];
    rmn[0]=mn[0][rs];
    mn[1][id]=mn[1][ls];
    rmn[1]=mn[1][rs];
    upmn(id);
    mn[0][id]=rmn[0];
    mn[1][id]=rmn[1];
}
void build(int l,int r,int id)
{
    mx[1][id]=MP(-NONE,-NONE);
    mn[1][id]=MP(NONE,-NONE);
    if(l==r)
    {
        mx[0][id]=mn[0][id]=a[l];
        return;
    }
    int mid=(l+r)>>1;
    build(l,mid,ls);
    build(mid+1,r,rs);
    pushup(id);
}
void update(int l,int r,int id)//only l==r
{
    mx[1][id]=MP(-NONE,-NONE);
    mn[1][id]=MP(NONE,-NONE);
    if(l>=ql&&r<=qr)
    {
        mx[0][id]=mn[0][id]=a[l];
        return;
    }
    int mid=(l+r)>>1;
    if(ql<=mid) update(l,mid,ls);
    if(qr>mid) update(mid+1,r,rs);
    pushup(id);
}
void query(int l,int r,int id)
{
    if(l>=ql&&r<=qr)
    {
        upmx(id);
        upmn(id);
        return ;
    }
    int mid=(l+r)>>1;
    if(ql<=mid) query(l,mid,ls);
    if(qr>mid) query(mid+1,r,rs);
}
void build(int _n){n=_n;build(1,n,1);}
void upd(int l,int r)
{
    ql=l;
    qr=r;
    update(1,n,1);
}
void ask(int l,int r)
{
    rmx[0]=rmx[1]=MP(-NONE,-NONE);
    rmn[0]=rmn[1]=MP(NONE,-NONE);
    ql=l;
    qr=r;
    query(1,n,1);

    // something
}
#undef type
#undef NONE
#undef ls
#undef rs
}tr;
```

10 附录

10.1 NTT 常用模数

$r * 2^k + 1$,r,k,g

3,1,1,2

5,1,2,2

17,1,4,3

97,3,5,5

193,3,6,5

257,1,8,3

7681,15,9,17

12289,3,12,11

40961,5,13,3

65537,1,16,3

786433,3,18,10

5767169,11,19,3

7340033,7,20,3

23068673,11,21,3

104857601,25,22,3

167772161,5,25,3

469762049,7,26,3

998244353,119,23,3

1004535809,479,21,3

2013265921,15,27,31

2281701377,17,27,3

3221225473,3,30,5

75161927681,35,31,3

77309411329,9,33,7

206158430209,3,36,22

2061584302081,15,37,7

2748779069441,5,39,3

6597069766657,3,41,5

39582418599937,9,42,5

79164837199873,9,43,5

263882790666241,15,44,7

1231453023109121,35,45,3

1337006139375617,19,46,3

3799912185593857,27,47,5

4222124650659841,15,48,19

7881299347898369,7,50,6

31525197391593473,7,52,3

180143985094819841,5,55,6

1945555039024054273,27,56,5

4179340454199820289,29,57,3

10.2 树 hash

Method I

Formula

$$f_{now} = size_{now} \times \sum f_{son_{now,i}} \times seed^{i-1}$$

Notes

其中 f_x 为以节点 x 为根的子树对应的哈希值。特殊地，我们令叶子节点的哈希值为 1。

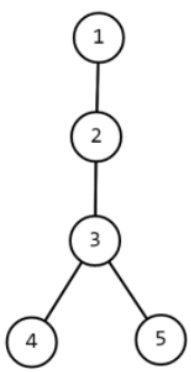
$size_x$ 表示以节点 x 为根的子树大小。

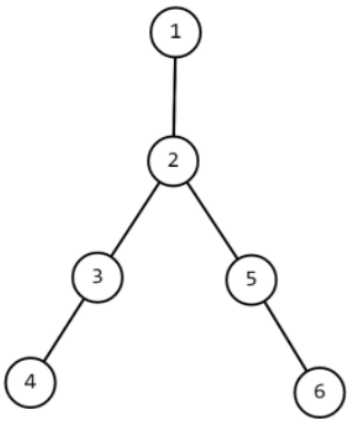
$son_{x,i}$ 表示 x 所有子节点以 f 作为关键字排序后排名第 i 的儿子。

$seed$ 为选定的一个合适的种子（最好是质数，对字符串 hash 有了解的人一定不陌生）

上述哈希过程中，可以适当取模避免溢出或加快运行速度。

Hack





上图中，可以计算出两棵树的哈希值均为 $60(1 + seed)$。

Method II

Formula

$$f_{now} = \bigoplus f_{son_{now,i}} \times seed + size_{son_{now,i}}$$

Notes

其中 f_x 为以节点 x 为根的子树对应的哈希值。特殊地，我们令叶子节点的哈希值为 1。

$size_x$ 表示以节点 x 为根的子树大小。

$son_{x,i}$ 表示 x 所有子节点之一（不用排序）。

$seed$ 为选定的一个合适的质数。

$\bigoplus$ 表示异或和。

Hack

由于异或的性质，如果一个节点下有多棵本质相同的子树，这种哈希值将无法分辨该种子树出现 $1, 3, 5, \ldots$ 次的情况。

Method III

Formula

$$f_{now} = 1 + \sum f_{son_{now,i}} \times prime(size_{son_{now,i}})$$

Notes

其中 f_x 为以节点 x 为根的子树对应的哈希值。

$size_x$ 表示以节点 x 为根的子树大小。

$son_{x,i}$ 表示 x 所有子节点之一（不用排序）。

$prime(i)$ 表示第 i 个质数。

10.3 线性基求交

```
LBasis intersection(const LBasis &a, const LBasis &
    b){
   LBasis ans, c = b, d = b;
   ans.init();
   for (int i = 0; i <= 32; i++){
      ll x = a.d[i];
      if(!x)continue;
      int j = i;
      ll T = 0;
      for(; j >= 0; --j){
         if((x >> j) & 1)
            if(c.d[j]) {x ^= c.d[j]; T ^= d.d[j];}
            else break;
      }
      if(!x) ans.d[i] = T;
      else {c.d[j] = x; d.d[j] = T;}
   }
   return ans;
}
```