# 字符串

## 后缀数组

```
int n,m,cnt[N],x[N],y[N],t[N],sa[N];
int g[N][20],rank[N],height[N];
char s[N];
bool cmp(int *g,int a,int b,int l)
{return    g[a]==g[b]&&g[a+l]==g[b+l];}
void get_sa()
{
   m=128;
   for(int i=1;i<=n;i++)cnt[x[i]=s[i]]++;
   for(int i=1;i<=m;i++)cnt[i]+=cnt[i-1];
   for(int i=n;i;i--)sa[cnt[x[i]]--]=i;
   for(int j=1,tot=0;tot<=n;m=tot,j<<=1)
   {
      tot=0;for(int i=n-j+1;i<=n;i++)y[++tot]=i;
      for(int i=1;i<=n;i++)if(sa[i]>j)y[++tot]=sa[i]-j;
      for(int i=1;i<=n;i++)t[i]=x[y[i]];
     memset(cnt,0,sizeof(cnt));
      for(int i=1;i<=n;i++)cnt[t[i]]++;
      for(int i=1;i<=m;i++)cnt[i]+=cnt[i-1];
      for(int i=n;i;i--)sa[cnt[t[i]]--]=y[i];
      tot=2;swap(x,y);x[sa[1]]=1;
      for(int i=2;i<=n;i++)
         x[sa[i]]=cmp(y,sa[i],sa[i-1],j)?tot-1:tot++;
   }
}
void get_height()
{
   for(int i=1;i<=n;i++)rank[sa[i]]=i;
   for(int i=1,j,k=0;i<=n;height[rank[i++]]=k)
      for(k=k?k-1:0,j=sa[rank[i]-1];s[i+k]==s[j+k];k++);
}
void rmq()
{
   memset(g,127,sizeof(g));
   for(int i=1;i<=n;i++)g[i][0]=height[i];
   for(int j=1;(1<<j)<=n;j++)
      for(int i=1;i<=n-(1<<j)+1;i++)
        g[i][j]=min(g[i][j-1],g[i+(1<<j-1)][j-1]);
}
int get_lcp(int x,int y)
{
   x=rank[x];y=rank[y];
   if(x>y)swap(x,y);x++;
   int p=log2(y-x+1);
   return min(g[x][p],g[y-(1<<p)+1][p]);
}

int main()
{
   scanf("%s",s+1);n=strlen(s+1);
   get_sa();get_height();get_rmq();
   return 0;
}
```

## 马拉车

```
int s[1000010],f[1000010],l,k,p;
char str[500010];
int main()
{
   int i;
   scanf("%s",str);l=strlen(str);
   for(i=0;i<l;i++) s[++k]=str[i]-48,s[++k]=-1;
   s[0]=inf-1;s[k]=inf;
   for(i=1;i<k;i++)
   {
      if(i<=p+f[p]-1) f[i]=min(f[p+p-i],p+f[p]-i);
      else f[i]=1;
      while(s[i-f[i]]==s[i+f[i]]) f[i]++;
      if(i+f[i]>p+f[p]) p=i;
   }
   return 0;
}
```

## 后缀自动机

```
class SAM{
   public:
   int build(int x,int c){
      int nx=++cnt;f[nx]=f[x]+1;
      while(x&&!ch[x][c])ch[x][c]=nx,x=fa[x];
      if(!x)fa[nx]=1;
      else{
         int p=ch[x][c];
         if(f[p]==f[x]+1)fa[nx]=p;
         else{
            int np=++cnt;f[np]=f[x]+1;
            memcpy(ch[np],ch[p],sizeof(ch[p]));
            fa[np]=fa[p];fa[nx]=fa[p]=np;
            while(x&&ch[x][c]==p)ch[x][c]=np,x=fa[x];
         }
      }
      return nx;
   }
}S;
int main()
{
   int pos;
   scanf(" %s",s+1);
   for(int i=1;i<=n;i++)pos=S.build(i,pos);
   return 0;
}
```

## KMP

```
int main()
{
   scanf(" %s",s+1);n=strlen(s+1);g[0]=-1;//模式串
   for(int i=1;i<=n;i++)
   {
     int x=g[i-1];
      while(x>=0&&s[x+1]!=s[i])x=g[x];
     g[i]=x+1;
   }
   scanf(" %s",t+1);n=strlen(t+1);f[0]=0;//匹配串
   for(int i=1;i<=n;i++)
   {
      int x=f[i-1];
      while(x>=0&&s[x+1]!=t[i])x=g[x];
      f[i]=x+1;
   }
   return 0;
}
```

## AC 自动机

```
class AC_machine
{
   public:
   void build(char *s,int len,int pos)
   {
      int x=0,num;
      for(int i=1;i<=len;i++)
      {
         if(!ch[x][s[i]-‘a’])ch[x][s[i]-‘a’]=++cnt;
         x=ch[x][s[i]-‘a’];
      }
   }
   void get_fail()
   {
      int l=1,r=1;
      for(int i=1;i<=52;i++)if(ch[0][i])q[r++]=ch[0][i];
      while(l<r)
      {
         int x=q[l++];
         for(int i=0;i<52;i++)
         {
            if(!ch[x][i]){ch[x][i]=ch[fail[x]][i];continue;}
            q[r]=ch[x][i];fail[q[r++]]=ch[fail[x]][i];
         }
      }
   }
}T;
```

# 数据结构

## 左偏树

```
int n,top,cnt,s[N],l[N],r[N],rt[N]; ll ans;
struct node{int lc,rc,size,num,dist;}t[N];

int merge(int a,int b){
   if(!a)return b;if(!b)return a;
   if(t[a].num<t[b].num)swap(a,b);
   t[a].size+=t[b].size;
   t[a].rc=merge(t[a].rc,b);
   if(t[t[a].lc].dist<t[t[a].rc].dist)
      swap(t[a].lc,t[a].rc);
   t[a].dist=t[t[a].rc].dist+1;
   return a;
}

int main(){
   scanf("%d",&n);
   for(int i=1;i<=n;i++)scanf("%d",&s[i]),s[i]-=i;
   for(int i=1;i<=n;i++)
   {
      rt[++top]=++cnt;t[cnt].num=s[i];t[cnt].size=1;l[top]=i;
      while(top>1&&t[rt[top]].num<t[rt[top-1]].num)
      {
         top--;
         rt[top]=merge(rt[top],rt[top+1]);
         while(t[rt[top]].size>(i-l[top]+2>>1))
            rt[top]=merge(t[rt[top]].lc,t[rt[top]].rc);
      }
   }
   l[top+1]=n+1;
   for(int i=1;i<=top;i++)
      for(int j=l[i];j<l[i+1];j++)
         ans+=abs(t[rt[i]].num-s[j]);
   printf("%lld\n",ans);
   return 0;
}
```

## Splay

```
#define inf 2100000000
using namespace std;
int n,m,root,sum,null=4000005,s[4000010];
struct node{
int num,fa,c[2],maxl,maxr,maxn,sum,change,turn,size;}t[4000010];

class splay_tree
{
   public:
   void up(int x)
   {
      int lc=t[x].c[0],rc=t[x].c[1],maxx;
      t[x].maxl=max(t[lc].maxl,t[lc].sum+t[x].num+max(0,t[rc].maxl));
      t[x].maxr=max(t[rc].maxr,t[rc].sum+t[x].num+max(0,t[lc].maxr));
      maxx=max(0,t[lc].maxr)+max(0,t[rc].maxl)+t[x].num;
      t[x].maxn=max(maxx,max(t[lc].maxn,t[rc].maxn));
      t[x].size=t[lc].size+t[rc].size+1;t[x].sum=t[lc].sum+t[rc].sum+t[x].num;
   }
   void rotate(int x,int k)
   {
      int ff=t[t[t[x].fa].fa].c[0]==t[x].fa?0:1;
      t[t[x].fa].c[k]=t[x].c[k^1];
      t[t[x].c[k^1]].fa=t[x].fa;
      t[x].c[k^1]=t[x].fa;
      int temp=t[t[x].fa].fa;
      t[t[x].fa].fa=x;t[x].fa=temp;
      t[t[x].fa].c[ff]=x;
      up(t[x].c[k^1]);
   }
   void turn(int pos)
   {
      t[pos].turn^=1;swap(t[pos].c[0],t[pos].c[1]);
      swap(t[pos].maxl,t[pos].maxr);
   }
   void change(int pos,int x)
   {
      t[pos].change=1;t[pos].num=x;t[pos].sum=t[pos].size*x;
```

```
    if(x>=0)t[pos].maxl=t[pos].maxr=t[pos].maxn=t[pos].sum;
    else t[pos].maxl=t[pos].maxr=t[pos].maxn=x;
}
void down(int x)
{
    int lc=t[x].c[0],rc=t[x].c[1];
    if(t[x].change)change(lc,t[x].num),change(rc,t[x].num),t[x].change=0;
    if(t[x].turn)turn(lc),turn(rc),t[x].turn=0;
}
void build(int x,int l,int r)
{
    t[x].num=t[x].sum=t[x].maxl=t[x].maxr=t[x].maxn=s[x];
    int mid;t[x].c[0]=t[x].c[1]=null;
    if(l<x)mid=(l+x-1)>>1,t[x].c[0]=mid,t[mid].fa=x,build(mid,l,x-1);
    if(r>x)mid=(x+r+1)>>1,t[x].c[1]=mid,t[mid].fa=x,build(mid,x+1,r);
    up(x);
}
void splay(int x,int pos)
{
    int flag=0;
    while(!flag)
    {
        down(x);int fa=t[x].fa,fb=t[fa].fa;
        if(fa==pos||fb==pos)flag=1;
        if(fa==pos){rotate(x,t[fa].c[0]==x?0:1);continue;}
        rotate(x,t[fa].c[0]==x?0:1);rotate(x,t[fb].c[0]==x?0:1);
    }
    up(x);if(pos==root)root=x;
}
void select(int x,int des)
{
    int p=root,cnt=x+1;
    while(cnt)
    {
        int temp=t[t[p].c[0]].size;down(p);
        if(cnt==temp+1)break;
        if(cnt<=temp)p=t[p].c[0];
        else p=t[p].c[1],cnt-=temp+1;
    }
    if(p!=des)splay(p,des);
  }
}T;

int main()
{
    char str[10];int rt,temp,a,b,c,pos;
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++)scanf("%d",&s[i]);
    sum=++n;root=n>>1;t[root].fa=null;
    t[null].maxn=-inf;
    t[0].maxn=t[n].maxn=t[0].num=s[0]=t[n].num=s[n]=-inf;
    T.build(root,0,n);
    while(m--)
    {
        scanf(" %s",str+1);
        if(str[3]=='S')//pos 后插入 k 个数
        {
            scanf("%d%d",&a,&b);n+=b;
            for(int i=sum+1;i<=n;i++)scanf("%d",&s[i]);
            T.select(a,root);T.select(a+1,t[root].c[1]);
            T.build(rt=sum+((b+1)>>1),sum+1,n);sum=n;
            temp=t[root].c[1];t[rt].fa=temp;t[temp].c[0]=rt;
            T.up(t[root].c[1]);T.up(root);
        }
        if(str[3]=='L')//pos 后删除 k 个数
        {
            scanf("%d%d",&a,&b);
            T.select(a-1,root);T.select(a+b,t[root].c[1]);
            t[t[root].c[1]].c[0]=null;
            T.up(t[root].c[1]);T.up(root);
        }
        if(str[3]=='K')//pos 后 k 个数改为 c
        {
            scanf("%d%d%d",&a,&b,&c);
            T.select(a-1,root);T.select(a+b,t[root].c[1]);
            pos=t[t[root].c[1]].c[0];T.change(pos,c);
            T.up(t[root].c[1]);T.up(root);
```

```
    }
    if(str[3]=='V')//区间翻转
    {
        scanf("%d%d",&a,&b);
        T.select(a-1,root);T.select(a+b,t[root].c[1]);
        T.turn(t[t[root].c[1]].c[0]);
        T.up(t[root].c[1]);T.up(root);
    }
    if(str[3]=='T')//区间和
    {
        scanf("%d%d",&a,&b);
        T.select(a-1,root);T.select(a+b,t[root].c[1]);
        int temp=t[t[root].c[1]].c[0];
        printf("%d\n",t[temp].sum);
    }
    if(str[3]=='X')printf("%d\n",t[root].maxn);//最大子段和
  }
  return 0;
}
```

## LCT

```
int n,q,fa[N],k,la[N],ff[N*2];
struct node{int c[2],fa,sum,val,size,rev,add,mul;}t[N];
stuct edge{int a,b;}map[N*2];

void add(int a,int b)
{
  map[++k]=(edge){a,b};ff[k]=la[a];la[a]=k;
  map[++k]=(edge){b,a};ff[k]=la[b];la[b]=k;
}

class link_cut_tree
{
  void pushdown(int x)
  {
    int lc=t[x].c[0],rc=t[x].c[1];
    if(t[x].rev)
    {
      if(lc)t[lc].rev^=1;swap(t[lc].c[0],t[lc].c[1]);
      if(rc)t[rc].rev^=1;swap(t[rc].c[0],t[rc].c[1]);
      t[x].rev=0;
    }
    if(t[x].mul!=1)
    {
      if(lc)
      {
        t[lc].sum=((ll)t[lc].sum*t[x].mul)%mod;
        t[lc].mul=((ll)t[lc].mul*t[x].mul)%mod;
        t[lc].add=((ll)t[lc].add*t[x].mul)%mod;
        t[lc].val=((ll)t[lc].val*t[x].mul)%mod;
      }
      if(rc)
      {
        t[rc].sum=((ll)t[rc].sum*t[x].mul)%mod;
        t[rc].mul=((ll)t[rc].mul*t[x].mul)%mod;
        t[rc].add=((ll)t[rc].add*t[x].mul)%mod;
        t[rc].val=((ll)t[rc].val*t[x].mul)%mod;
      }
      t[x].mul=1;
    }
    if(t[x].add)
    {
      if(lc)
      {
```

```
            t[lc].sum=(t[lc].sum+(ll)t[lc].size*t[x].add)%mod;
            t[lc].add=(t[lc].add+t[x].add)%mod;
            t[lc].val=(t[lc].val+t[x].add)%mod;
        }
        if(rc)
        {
            t[rc].sum=(t[rc].sum+(ll)t[rc].size*t[x].add)%mod;
            t[rc].add=(t[rc].add+t[x].add)%mod;
            t[rc].val=(t[rc].val+t[x].add)%mod;
        }
        t[x].add=0;
    }
}
void update(int x)
{
    int lc=t[x].c[0],rc=t[x].c[1];
    t[x].sum=(t[lc].sum+t[rc].sum+t[x].val)%mod;
    t[x].size=t[lc].size+t[rc].size+1;
}
void rotate(int x,int k)
{
    int y=t[x].fa,f=(t[t[y].fa].c[1]==y);
    pushdown(y);pushdown(x);
    if(!t[y].fa)fa[x]=fa[y];
    t[x].fa=t[y].fa;t[t[y].fa].c[f]=x;
    t[y].fa=x;t[y].c[k]=t[x].c[k^1];
    t[t[y].c[k]].fa=y;t[x].c[k^1]=y;
    update(y);
}
void splay(int x){
    while(t[x].fa)
    {
        int y=t[x].fa,f=t[y].c[1]==x;
        if(!t[y].fa)rotate(x,f);
        else
        {
            if(f==(t[t[y].fa].c[1]==y))rotate(y,f),rotate(x,f);
            else rotate(x,f),rotate(x,f^1);
        }
    }
    pushdown(x);//!!!!!
    update(x);
}
void access(int x)
{
    for(int y=0;x;y=x,x=fa[x])
    {
        splay(x);
        t[t[x].c[1]].fa=0;fa[t[x].c[1]]=x;
        t[x].c[1]=y;t[y].fa=x;fa[y]=0;
        update(x);
    }
}
int lca(int x,int y)
{
    access(x);
    for(splay(y);fa[y];splay(y))y=fa[y];
    return y;
}
public:
void add(int a,int b,int c){
    int p=lca(a,b),pos;
    access(a);splay(p);pos=t[p].c[1];
    t[pos].sum=(t[pos].sum+(ll)t[pos].size*c)%mod;
    t[pos].add=(t[pos].add+c)%mod;
    t[pos].val=(t[pos].val+c)%mod;
    access(b);splay(p);pos=t[p].c[1];
    t[pos].sum=(t[pos].sum+(ll)t[pos].size*c)%mod;
    t[pos].add=(t[pos].add+c)%mod;
    t[pos].val=(t[pos].val+c)%mod;
    t[p].val=(t[p].val+c)%mod;update(p);
}
void mul(int a,int b,int c){
    int p=lca(a,b),pos;
    access(a);splay(p);pos=t[p].c[1];
    t[pos].sum=((ll)t[pos].sum*c)%mod;
    t[pos].mul=((ll)t[pos].mul*c)%mod;
    t[pos].add=((ll)t[pos].add*c)%mod;
    t[pos].val=((ll)t[pos].val*c)%mod;
    access(b);splay(p);pos=t[p].c[1];
    t[pos].sum=((ll)t[pos].sum*c)%mod;
    t[pos].mul=((ll)t[pos].mul*c)%mod;
    t[pos].add=((ll)t[pos].add*c)%mod;
    t[pos].val=((ll)t[pos].val*c)%mod;
    t[p].val=((ll)t[p].val*c)%mod;update(p);
}
void link(int x,int y)
{
```

```
        access(x);splay(x);fa[x]=y;
        t[x].rev^=1;swap(t[x].c[0],t[x].c[1]);
    }
    void cut(int x,int y){
        access(x);splay(y);
        if(fa[y]==x)swap(x,y);splay(x);
        t[t[x].c[0]].fa=0;fa[t[x].c[0]]=1;
        t[x].c[0]=fa[x]=0;update(x);
    }
    int qry(int x,int y)
    {
        int p=lca(x,y),pos,res=t[p].val;
        access(x);splay(p);pos=t[p].c[1];
        res=(res+t[pos].sum)%mod;
        access(y);splay(p);pos=t[p].c[1];
        res=(res+t[pos].sum)%mod;
        return res;
    }
}T;

void dfs(int x)
{
    t[x].size=1;t[x].mul=1;t[x].val=1;t[x].sum=1;
    for(int a=la[x];a;a=ff[a])
        if(fa[x]!=map[a].b)
            fa[map[a].b]=x,dfs(map[a].b);
}

int main()
{
    int a,b,c,d;char ch;
    scanf("%d%d",&n,&q);
    for(int i=1;i<n;i++)scanf("%d%d",&a,&b),add(a,b);
    dfs(1);
    while(q--)
    {
        scanf(" %c",&ch);
        if(ch=='+')scanf("%d%d%d",&a,&b,&c),T.add(a,b,c);
        if(ch=='-')scanf("%d%d%d%d",&a,&b,&c,&d),T.cut(a,b),T.link(c,d);
        if(ch=='*')scanf("%d%d%d",&a,&b,&c),T.mul(a,b,c);
        if(ch=='/')scanf("%d%d",&a,&b),printf("%d\n",T.qry(a,b));
    }
    return 0;
}
```

## 可持久化 treap

```
int n,cnt,rt,L=2147483647;
struct node{
    char x;int lc,rc,tag,size;
    bool operator <(const node &p)const{
        int k=2000000207;
        return (ll)((k+=(k<<2)+1)&L)*(size+p.size)<(ll)(size)*L;
    }
}t[N*200];

class functional_trape
{
    void pushdown(int x)
    {
        if(!t[x].tag)return;
        int p;
        if(t[x].lc)
        {
            p=t[x].lc;t[x].lc=++cnt;t[cnt]=t[p];
            swap(t[cnt].lc,t[cnt].rc);t[cnt].tag^=1;
        }
        if(t[x].rc)
        {
            p=t[x].rc;t[x].rc=++cnt;t[cnt]=t[p];
            swap(t[cnt].lc,t[cnt].rc);t[cnt].tag^=1;
        }
        t[x].tag=0;
    }
    int merge(int a,int b)
    {
        if(!a)return b;
        if(!b)return a;
        int x=++cnt;
        if(t[a]<t[b])
        {
            pushdown(a);t[x]=t[a];
            t[x].rc=merge(t[a].rc,b);
            t[x].size+=t[b].size;
        }
        else
        {
            pushdown(b);t[x]=t[b];
            t[x].lc=merge(a,t[b].lc);
            t[x].size+=t[a].size;
```

```
    }
    return x;
}
void split(int x,int &a,int &b,int k)
{
    if(!x){a=0;b=0;return;}
    pushdown(x);
    if(t[t[x].lc].size>=k)
    {
        b=++cnt;t[b]=t[x];
        split(t[x].lc,a,t[b].lc,k);
        t[b].size-=t[a].size;
    }
    else
    {
        a=++cnt;t[a]=t[x];
        split(t[x].rc,t[a].rc,b,k-t[t[x].lc].size-1);
        t[a].size-=t[b].size;
    }
}
public:
void insert(int x,int y)
{
    int a=0,b=0;
    split(rt,a,b,x);
    rt=merge(merge(a,y),b);
}
void erase(int l,int r)
{
    int a=0,b=0,c=0,d=0;
    split(rt,a,b,r);
    split(a,c,d,l-1);
    rt=merge(c,b);
}
void copy(int l,int r,int x)
{
    int a=0,b=0,c=0,d=0;
    split(rt,a,b,r);
    split(a,c,d,l-1);
    insert(x,d);
}
void reverse(int l,int r)
{
    int a=0,b=0,c=0,d=0;
        split(rt,a,b,r);
        split(a,c,d,l-1);
        int x=++cnt;t[x]=t[d];
        swap(t[x].lc,t[x].rc);t[x].tag^=1;
        rt=merge(merge(c,x),b);
    }
    char qry(int k)
    {
        int x=rt;
        while(1)
        {
            pushdown(x);
            if(t[t[x].lc].size+1==k)break;
            if(t[t[x].lc].size>=k)x=t[x].lc;
            else k-=t[t[x].lc].size+1,x=t[x].rc;
        }
        return t[x].x;
    }
}T;

int main()
{
    int x,l,r;char ch,c;
    scanf("%d",&n);
    while(n--)
    {
        scanf(" %c",&ch);
        if(ch=='I')
        {
            scanf("%d %c",&x,&c);
            t[++cnt].x=c;t[cnt].size=1;
            T.insert(x,cnt);
        }
        if(ch=='D')scanf("%d%d",&l,&r),T.erase(l,r);
        if(ch=='C')scanf("%d%d%d",&l,&r,&x),T.copy(l,r,x);
        if(ch=='R')scanf("%d%d",&l,&r),T.reverse(l,r);
        if(ch=='Q')scanf("%d",&x),printf("%c",T.qry(x));
    }
    printf("\n");
    return 0;
}
```

# 动态点分治

```
int n,m,val[N],fa[N],w[N],flag[N],check[N];
int k,la[N],ff[N*2],q[N],size[N],h[N],change[N];
ll f1[N],f2[N],pre[N],d[N][20];
struct node{int a,b,c;}map[N*2];
void add(int a,int b,int c)
{
    map[++k]=(node){a,b,c};ff[k]=la[a];la[a]=k;
    map[++k]=(node){b,a,c};ff[k]=la[b];la[b]=k;
}
int find_root(int S)
{
    int l=1,r=2,num=inf,res=0;q[1]=S;flag[S]=1;fa[S]=0;
    while(l<r)
    {
        int x=q[l];l++;w[x]=0;size[x]=1;
        for(int a=la[x];a;a=ff[a])
            if(!flag[map[a].b]&&!check[map[a].b])
                q[r]=map[a].b,flag[q[r]]=1,fa[q[r]]=x,r++;
    }
    for(int i=r-1;i;i--)
    {
        int x=q[i],val=max(w[x],r-size[x]-1); flag[x]=0;
        if(fa[x])size[fa[x]]+=size[x],w[fa[x]]=max(w[fa[x]],size[x]);
        if(val<num)num=val,res=x;
    }
    return res;
}
void bfs(int S,int dep)
{
    int l=1,r=2;q[1]=S;check[S]=1;
    while(l<r)
    {
        int x=q[l++];
        for(int a=la[x];a;a=ff[a])
            if(!flag[map[a].b]&&!check[map[a].b])
            {
                q[r]=map[a].b;flag[map[a].b]=1;
                d[q[r]][dep]=d[x][dep]+map[a].c;r++;
            }
    }
    for(int i=r-1;i;i--)flag[q[i]]=0;
}
int build(int x,int dep)
{
    x=find_root(x);check[x]=1;
    h[x]=dep;bfs(x,dep);
    for(int a=la[x];a;a=ff[a])
    {
        if(check[map[a].b])continue;
        pre[build(map[a].b,dep+1)]=x;
    }
    return x;
}
void modify(int x)
{
    if(change[x])return;
    change[x]=1;size[x]++;
    for(int i=x;pre[i];i=pre[i])
    {
        int num=d[x][h[pre[i]]];
        f1[pre[i]]+=num;f2[i]+=num;size[pre[i]]++;
    }
}
ll qry(int x)
{
    ll res=f1[x];
    for(int i=x;pre[i];i=pre[i])
        res+=f1[pre[i]]-f2[i]+(ll)(size[pre[i]]-size[i])*d[x][h[pre[i]]];
    return res;
}
int main()
{
    int tp,x;
    scanf("%d%d",&n,&m);
    for(int i=2;i<=n;i++)scanf("%d",&fa[i]);
    for(int i=2;i<=n;i++)scanf("%d",&val[i]);
    for(int i=2;i<=n;i++)add(fa[i]+1,i,val[i]);
    memset(fa,0,sizeof(fa));build(1,1);
    memset(size,0,sizeof(size));
    while(m--)
    {
        scanf("%d%d",&tp,&x);x++;
        if(tp==1)modify(x);
        else printf("%lld\n",qry(x));
    }
    return 0;
}
```

## KD 树求关于（x，y）点的 k 远点对

```
#define inf 2100000000
#define N 100005
using namespace std;
int n,m,cnt,rt,D;
struct point{
    int c[2],id;
    bool operator<(const point &x)const{
        return c[D]<x.c[D]||(c[D]==x.c[D]&&c[D^1]<x.c[D^1]);}
}s[N];
struct node{int lc,rc,minx,maxx,miny,maxy;point poi;}t[N];
struct info{
    ll dis;int id;
    bool operator<(const info &x)const{
        return dis>x.dis||(dis==x.dis&&id<x.id);
    }
};
priority_queue<info>q;

class kd_tree{
    inline ll pf(int x){return (ll)x*x;}
    inline ll get(int X1,int Y1,int X2,int Y2){
        return pf(X1-X2)+pf(Y1-Y2);
    }
    inline ll cal(int x,int X,int Y){
        ll res=0;
        res=max(res,get(t[x].minx,t[x].miny,X,Y));
        res=max(res,get(t[x].minx,t[x].maxy,X,Y));
        res=max(res,get(t[x].maxx,t[x].miny,X,Y));
        res=max(res,get(t[x].maxx,t[x].maxy,X,Y));
        return res;
    }
    inline void update(int x){
        int lc=t[x].lc,rc=t[x].rc;
        t[x].minx=min(t[x].poi.c[0],min(t[lc].minx,t[rc].minx));
        t[x].maxx=max(t[x].poi.c[0],max(t[lc].maxx,t[rc].maxx));
        t[x].miny=min(t[x].poi.c[1],min(t[lc].miny,t[rc].miny));
        t[x].maxy=max(t[x].poi.c[1],max(t[lc].maxy,t[rc].maxy));
    }
    public:
    inline void prepare(){
        t[0].minx=inf;t[0].maxx=-inf;
        t[0].miny=inf;t[0].maxy=-inf;
    }//!!!!!!!!!!!!!!!!!记得把 0 位置清空
    inline void build(int &x,int l,int r,int inv){
        if(l>r)return;x=++cnt;
        int mid=l+r>>1;
        D=inv;nth_element(s+l,s+mid,s+r+1);
        t[x].minx=t[x].maxx=s[mid].c[0];
        t[x].miny=t[x].maxy=s[mid].c[1];
        t[x].poi=s[mid];
        build(t[x].lc,l,mid-1,inv^1);
        build(t[x].rc,mid+1,r,inv^1);
        update(x);
    }
    inline void qry(int x,int X,int Y){
        if(!x)return;
        int lc=t[x].lc,rc=t[x].rc;
        ll ls=cal(lc,X,Y),rs=cal(rc,X,Y);
        ll tmp=get(t[x].poi.c[0],t[x].poi.c[1],X,Y);
        if((info){tmp,t[x].poi.id}<q.top())
            q.pop(),q.push((info){tmp,t[x].poi.id});
        if(ls>rs){
            if(ls>=q.top().dis)qry(lc,X,Y);
            if(rs>=q.top().dis)qry(rc,X,Y);
        }
        else{
            if(rs>=q.top().dis)qry(rc,X,Y);
            if(ls>=q.top().dis)qry(lc,X,Y);
        }
    }
}T;
int main(){
    int x,y,k;
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
        scanf("%d%d",&s[i].c[0],&s[i].c[1]),s[i].id=i;
    T.prepare();T.build(rt,1,n,0);
    scanf("%d",&m);
    while(m--){
        scanf("%d%d%d",&x,&y,&k);
        while(!q.empty())q.pop();
        for(int i=1;i<=k;i++)q.push((info){-1,n+1});
        T.qry(rt,x,y);printf("%d\n",q.top().id);
    }
    return 0;
}
```

## 吉司机线段树

区间取 min（a[i]，x）+区间取 max（a[i]，x）+求和+单点修改
对线段树上的每一个区间维护区间最大值 mx，这个区间中最大值出现的次数 t，区间次大值 se，区间和 sum
现在考虑打上区间取 min 标记 x：
如果 mx<=x，那么对 sum 就没有修改。
如果 se<x<mx，那么 sum=sum-(mx-x)×t。
如果 x<=se<mx，那么我们分别 DFS 这个节点的两个孩子，如果当前 DFS 的过程中遇到了前两种情况，就直接修改打上标记然后退出，否则就继续 DFS。
时间复杂度 O（mlogn）

## 线性基

```
class linear_basis
{
   public:
   void insert(ll num)
   {
      for(int i=D;i>=0;i--)
      {
         if(num>>i)
         {
            if(!s[i]){s[i]=num;return;}
            num=(num^s[i]);
         }
      }
   }//插入
   ll qry(ll num)
   {
      ll res=num;
      for(int i=D;i>=0;i--)
         if((res^s[i])>res)res=(res^s[i]);
      return res;
   }//num 与在线性基中的数的异或最大值
}S;
//线性基合并：把其中一个取出来一个个塞到另一个里面
```

## 哈希表

```
#define N 1000005
#define cyc 19990213
int cnt,ff[N],la[N],h[N],tot,tong[N];
struct node{
   int a,b;
   bool operator==(const node &x)
   const{return a==x.a&&b==x.b;}
}s[N];
class hash_it
{
   public:
   inline void add(int x,int y,int val)
   {
      int num=((ll)x*cyc+y)%N;//hash
      node tmp=(node){x,y};
      for(int a=la[num],lx=0;a;lx=a,a=ff[a])
         if(s[a]==tmp)
         {
            h[a]+=val;
            if(!h[a])
            {
               if(!lx)la[num]=ff[a];
               ff[lx]=ff[a];ff[a]=0;h[a]=0;
               tong[++tot]=a;//空间回收
            }
            return;
         }
      int p;
      if(!tot)p=++cnt;
      else p=tong[tot],tot--;//空间释放
      ff[p]=la[num];la[num]=p;
      s[p]=(node){x,y};h[p]=val;
   }
   inline int qry(int x,int y)
   {
      int num=((ll)x*cyc+y)%N;//hash
      node tmp=(node){x,y};
      for(int a=la[num];a;a=ff[a])
         if(s[a]==tmp)return h[a];
      return 0;
   }
}hash;
```

## 可持久化并查集

```
#define N 200005
using namespace std;
int n,m,pre,cnt,rt[N],size[N];
struct info{int lc,rc,val,size;}t[N*40];
class seg_tree{
  public:
  void build(int &x,int l,int r){
    x=++cnt;
    if(l==r){t[x].val=l;t[x].size=1;return;}
    int mid=l+r>>1;
    build(t[x].lc,l,mid);build(t[x].rc,mid+1,r);
  }//初始化
  void modify1(int &x,int pre,int l,int r,int pos,int val){
    x=++cnt;t[x]=t[pre];
    if(l==r){t[x].val=val;return;}
    int mid=l+r>>1;
    if(pos<=mid)modify1(t[x].lc,t[pre].lc,l,mid,pos,val);
    else modify1(t[x].rc,t[pre].rc,mid+1,r,pos,val);
  }//修改父亲
  void modify2(int &x,int pre,int l,int r,int pos,int val){
    x=++cnt;t[x]=t[pre];
    if(l==r){t[x].size+=val;return;}
    int mid=l+r>>1;
    if(pos<=mid)modify2(t[x].lc,t[pre].lc,l,mid,pos,val);
    else modify2(t[x].rc,t[pre].rc,mid+1,r,pos,val);
  }//修改 size
  int qry1(int x,int l,int r,int pos)
  {
    if(l==r)return t[x].val;
    int mid=l+r>>1;
    if(pos<=mid)return qry1(t[x].lc,l,mid,pos);
    return qry1(t[x].rc,mid+1,r,pos);
  }//查询父亲
  int qry2(int x,int l,int r,int pos)
  {
    if(l==r)return t[x].size;
    int mid=l+r>>1;
    if(pos<=mid)return qry2(t[x].lc,l,mid,pos);
    return qry2(t[x].rc,mid+1,r,pos);
  }//查询 size
}T;
```

```
int find(int p,int x)
{
  int pos=T.qry1(p,1,n,x);
  if(pos==x)return x;
  return find(p,pos);
}//并查集的查找函数
```

## 线段树合并

```
class seg_tree
{
  public:
  void orzcyc(){t[0].maxn=-N;}
  void modify(int &x,int l,int r,int des,int val)
  {
    if(!x)x=++cnt;
    if(l==r){t[x].maxn+=val;t[x].pos=l;return;}
    int mid=l+r>>1;
    if(des<=mid)modify(t[x].lc,l,mid,des,val);
    else modify(t[x].rc,mid+1,r,des,val);
    update(x);
  }
  int merge(int x,int y,int l,int r)
  {
    if(!x)return y;
    if(!y)return x;
    if(l==r){t[x].maxn+=t[y].maxn;return x;}
    int mid=l+r>>1;
    t[x].lc=merge(t[x].lc,t[y].lc,l,mid);
    t[x].rc=merge(t[x].rc,t[y].rc,mid+1,r);
    update(x);
    return x;
  }
}T;
```

## Kruscal 重构树

将边排序按照一定顺序依次加入，每次一条边，把两个点连向一个新的点，构建一颗重构树。

这棵树满足从上往下权值依次减小，且原始点为叶子结点。

对于一个点出发在一定权值条件下能到达的点的相关询问，等价于重构树的一个子树的询问，可以方便解决。

# 计算几何

## 凸包

```
int n,top=0;
struct point{
   int a,b;
   bool operator<(point p)const{return a<p.a||(a==p.a&&b<p.b);}
   int operator*(point p)const{return a*p.b-b*p.a;}
   point operator-(point p)const{return (point){a-p.a,b-p.b};}
}s[50010],q[50010];

int main()
{
   scanf("%d",&n);
   for(int i=1;i<=n;i++) scanf("%d%d",&s[i].a,&s[i].b);
   sort(s+1,s+n+1);top=0;
   for(int i=1;i<=n;i++)
   {
      while(top>1 && (q[top]-q[top-1])*(s[i]-q[top])<=0) top--;
      q[++top]=s[i];
   }int sum=top;
   for(int i=n-1;i>0;i--)
   {
      while(top>sum && (q[top]-q[top-1])*(s[i]-q[top])<=0) top--;
      q[++top]=s[i];
   }
   printf("%d\n",top-1);
   return 0;
}
```

## 半平面交

```
int n,tot;
double ans1,ans2;
struct point{
   double x,y;
   point operator+(const point &p)
   const{return (point){x+p.x,y+p.y};}
   point operator-(const point &p)
   const{return (point){x-p.x,y-p.y};}
   point operator*(const double &num)
   const{return (point){x*num,y*num};}
   double operator^(const point &p)
   const{return x*p.y-p.x*y;}
}t[N],p[N*2];
struct node{
   point p,v;double ang;
   bool operator<(const node &x)
   const{return ang<x.ang||(ang==x.ang&&(v^(x.p-p))<eps);}
}s[N*2],q[N*2];
bool pd(node a,point b){return (a.v^(b-a.p))<-eps;}
double get_ang(point p){return atan2(p.y,p.x);}
point get_jd(node a,node b){
   point x=a.p-b.p;
   double t=(b.v^x)/(a.v^b.v);
   return a.p+a.v*t;
}
void solve()
{
   sort(s+1,s+n+1);tot=1;
   for(int i=2;i<=n;i++)
      if(fabs(s[i].ang-s[i-1].ang)>eps)s[++tot]=s[i];
   int l=1,r=2;q[1]=s[1];q[2]=s[2];n=tot;
   for(int i=3;i<=n;i++)
   {
      while(l<r&&pd(s[i],get_jd(q[r],q[r-1])))r--;
      while(l<r&&pd(s[i],get_jd(q[l],q[l+1])))l++;
      q[++r]=s[i];
   }
```

```
    while(l<r&&pd(q[l],get_jd(q[r],q[r-1])))r--;
    while(l<r&&pd(q[r],get_jd(q[l],q[l+1])))l++;
    if(r-l<=1)return;
    q[r+1]=q[l];tot=0;
    for(int i=l;i<=r;i++)p[++tot]=get_jd(q[i],q[i+1]);
    p[tot+1]=p[1];
    for(int i=1;i<=tot;i++)ans+=fabs((p[i]-p[1])^(p[i+1]-p[1]));
}
int main()
{
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    scanf("%lf%lf",&s[i].p.x,&s[i].p.y,&s[i].v.x,&s[i].v.y);
    solve();printf("%.4lf",ans);
    return 0;
}
```

## 辛普森积分

```
int n;double minn=inf,maxn;
struct node{double x,y,r;}t[N];
struct info{
    double l,r;
    bool operator<(const info &x)
    const{return l<x.l||(l==x.l&&r<x.r);}
}s[N];
double p(double x){return x*x;}
double f(double x)
{
    int tot=0;
    for(int i=1;i<=n;i++)
    {
        if(t[i].x-t[i].r>x||t[i].x+t[i].r<x)continue;
        double tmp=sqrt(p(t[i].r)-p(x-t[i].x));
        s[++tot]=(info){-tmp+t[i].y,tmp+t[i].y};
    }
    if(!tot)return 0.0;
    sort(s+1,s+tot+1);
    double res=0.0,L=s[1].l,R=s[1].r;
    for(int i=2;i<=tot;i++)
    {
        if(s[i].l<=R)R=fmax(R,s[i].r);
        else res+=R-L,L=s[i].l,R=s[i].r;
    }
    return res+R-L;
}
double simpson(double l,double r)
{
    double mid=(l+r)*0.5;
    return (f(l)+f(r)+f(mid)*4.0)*(r-l)/6.0;
}
double solve(double l,double r,double S,int dep)
{
    if(dep>21)return simpson(l,r);
    double mid=(l+r)*0.5;
    double s1=simpson(l,mid),s2=simpson(mid,r);
    if(fabs(s1+s2-S)<eps&&dep>5)return simpson(l,r);
    return solve(l,mid,s1,dep+1)+solve(mid,r,s2,dep+1);
}

int main()
{
    double x,y,r;
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        scanf("%lf%lf%lf",&x,&y,&r);
        t[i]=(node){x,y,r};
        minn=fmin(minn,x-r);
        maxn=fmax(maxn,x+r);
    }
    printf("%.4lf\n",solve(minn,maxn,0,1));
    return 0;
}
```

## 旋转卡壳

```
int n,top=0,ans,temp;
struct point{
    int a,b;
    bool operator<(point p)const{return a<p.a||(a==p.a&&b<p.b);}
    int operator*(point p)const{return a*p.b-b*p.a;}
    point operator-(point p)const{return (point){a-p.a,b-p.b};}
}s[50010],q[50010];
int cal(point a,point b,point c){return a*b+b*c+c*a;}
int get_jl(point a){return a.a*a.a+a.b*a.b;}
int next(int k){return (k>=top)?1:k+1;}
int main()
{
    scanf("%d",&n);
    for(int i=1;i<=n;i++) scanf("%d%d",&s[i].a,&s[i].b);
    sort(s+1,s+n+1);top=0;
    for(int i=1;i<=n;i++)
    {
        while(top>1 && (q[top]-q[top-1])*(s[i]-q[top])<=0) top--;
        q[++top]=s[i];
    }int sum=top;
    for(int i=n-1;i>0;i--)
    {
        while(top>sum && (q[top]-q[top-1])*(s[i]-q[top])<=0) top--;
        q[++top]=s[i];
    }
    for(int i=1,j=2;i<top;i++)
    {
      while(i!=next(j)&&cal(q[i],q[i+1],q[j])<=cal(q[i],q[i+1],q[j+1]))j=next(j);
        ans=max(ans,max(get_jl(q[i]-q[j]),get_jl(q[i+1]-q[j])));
    }
    printf("%d\n",ans);
    return 0;
}
```

## 最小圆覆盖

```
long double esp=1e-10,r;
int n;
struct point
{
    long double a,b;
    bool operator<(point p)const{return a<p.a||(a==p.a&&b<p.b);}
    int operator*(point p)const{return a*p.b-b*p.a;}
    point operator-(point p)const{return (point){a-p.a,b-p.b};}
}s[100010],c;
long double get_jl(point x){return sqrt((x.a*x.a)+(x.b*x.b));}

void get_zd(point a,point b)
{c.a=(a.a+b.a)/2;c.b=(a.b+b.b)/2;}

void get_wx(point x,point y,point z)
{
    point temp;long double a,b,d,t1,t2,t3,p,q;
    a=get_jl(y-z);b=get_jl(x-z);d=get_jl(x-y);
    p=(a*a+b*b+d*d)/2;q=1/(1/(p-a*a)+1/(p-b*b)+1/(p-d*d));
    t1=q/(p-a*a);t2=q/(p-b*b);t3=q/(p-d*d);
    c.a=((1-t1)*x.a+(1-t2)*y.a+(1-t3)*z.a)/2;
    c.b=((1-t1)*x.b+(1-t2)*y.b+(1-t3)*z.b)/2;
    r=sqrt((p-q)/4);
}//三点求外心（ r 为半径，c 为外心坐标）

void get_circle()
{
    r=0;c=s[1];
    for(int i=2;i<=n;i++)
    {
        if(get_jl(c-s[i])+esp<=r) continue;
        c=s[i];r=0;
        for(int j=1;j<i;j++)
        {
            if(get_jl(s[j]-c)+esp<=r) continue;
            get_zd(s[i],s[j]);r=get_jl(s[i]-c);
            for(int k=1;k<j;k++)
```

```
            if(get_jl(s[k]-c)>r+esp) get_wx(s[i],s[j],s[k]);
        }
    }
    printf("%.10lf\n%.10lf %.10lf\n",double(r),double(c.a),double(c.b));
}

int main()
{
    long double x,y;
    scanf("%d",&n);
    for(int i=1;i<=n;i++) cin>>s[i].a>>s[i].b;
    get_circle();
    return 0;
}
```

## 动态凸包

```
#define iter multiset<node>::iterator
int n;ll S;
struct node{
    int a,b;double ang;
    bool operator<(const node &x)
    const{return ang<x.ang||(ang==x.ang&&b<x.b);}
    node operator-(const node &x){return (node){a-x.a,b-x.b,ang};}
    ll operator*(const node &x){return (ll)a*x.b-(ll)b*x.a;}
}t[4],o,p;
multiset<node>q;
double get(node x){return (double)atan2(x.b,x.a);}
iter getpre(iter x){if(x==q.begin())x=q.end();return (--x);}
iter getnxt(iter x){return (++x)==q.end()?q.begin():x;}
node operator-(iter x,iter y){return (node)*x-(node)*y;}
ll operator*(iter x,iter y){return ((node)*x)*((node)*y);}
int main()
{
    for(int i=1;i<=3;i++)
    {
        scanf("%d%d",&t[i].a,&t[i].b);
        o.a+=t[i].a;o.b+=t[i].b;
        t[i].a*=3;t[i].b*=3;
    }
    for(int i=1;i<=3;i++)t[i].ang=(double)get(t[i]-o),q.insert(t[i]-o);
    sort(t+1,t+4);
    S=t[1]*t[2]+t[2]*t[3]+t[3]*t[1];
    scanf("%d",&n);
    while(n--)
    {
        scanf("%d%d",&p.a,&p.b);
        p.a*=3;p.b*=3;p=p-o;p.ang=get(p);
        iter nxt=q.lower_bound(p);
        if(nxt==q.end())nxt=q.begin();
        iter pre=getpre(nxt);
        if((nxt-pre)*(p-*pre)<0)
        {
            S-=pre*nxt;
            iter pos=getpre(pre);
            while((pre-pos)*(p-*pos)<=0)
            {
                S-=pos*pre;q.erase(pre);
                pre=pos;pos=getpre(pos);
            }
            pos=getnxt(nxt);
            while((nxt-pos)*(p-*pos)>=0)
            {
                S-=nxt*pos;q.erase(nxt);
                nxt=pos;pos=getnxt(pos);
            }
            S+=p*(*nxt)-(p*(*pre));q.insert(p);
        }
        printf("%lld\n",S/9);
    }
    return 0;
}
```

# 图论

## 费用流

```
inline bool spfa()
{
   for(int i=S;i<=T;i++)w[i]=-inf;
   int l=1,r=2;q[1]=S;w[S]=0;
   while(l!=r)
   {
      int x=q[l];l=next(l);flag[x]=0;
      for(int a=la[x];a;a=ff[a])
         if(map[a].c&&w[map[a].b]<w[x]+map[a].v)
         {
            w[map[a].b]=w[x]+map[a].v;pre[map[a].b]=a;
            if(!flag[map[a].b])
               q[r]=map[a].b,flag[q[r]]=1,r=next(r);
         }
   }
   return w[T]>0;
}

inline void add()
{
   int flow=inf;
   for(int i=pre[T];i;i=pre[map[i].a])flow=min(flow,map[i].c);
   for(int i=pre[T];i;i=pre[map[i].a])
      ans+=flow*map[i].v,map[i].c-=flow,map[i^1].c+=flow;
}

int main()
{
   while(spfa())add();
   printf("%d\n",ans+1);
   return 0;
}
```

## 网络流(dinic)

```
ll dfs(int x,ll flow){
   if(x==T)return flow;
   ll res=0,tmp;
   for(int a=cur[x];a&&flow;a=ff[a])
   {
     cur[x]=a;
      if(dep[e[a].b]==dep[x]+1&&e[a].c)
      {
         tmp=dfs(map[a].b,min(flow,e[a].c));
         e[a].c-=tmp;e[a^1].c+=tmp;res+=tmp;flow-=tmp;
      }
   }
   if(!res)dep[x]=-1;
   return res;
}
bool bfs(){
   memset(dep,0,sizeof(dep));
   for(int i=S;i<=T;i++)cur[i]=la[i];
   int l=1,r=2;q[1]=S;dep[S]=1;
   while(l<r)
   {
      int x=q[l];l++;
      for(int a=la[x];a;a=ff[a])
         if(!dep[e[a].b]&&e[a].c)
            q[r]=e[a].b,dep[q[r]]=dep[x]+1,r++;
   }
   return dep[T];
}
int main()
{
   int a,b;
   scanf("%d%d",&n,&m);
   for(int i=1;i<=m;i++)scanf("%d%d%d",&a,&b,&c),add(a,b,c);
   while(bfs(S,0))ans+=dfs(S,0);
   printf("%d\n",ans);
   return 0;
}
```

## KM 算法（二分图最大权匹配）

```
#include<bits/stdc++.h>
#define ll long long
#define inf 2100000000
#define N 1005
#define M 1000005
using namespace std;
int n,n1,n2,m,labl[N],labr[N],pl[N],pr[N],pre[N];
int e[N][N],q[N],flag[M],slack[N];ll ans;
void find(int x){
   int p;
   while(x){
      p=pl[pre[x]];pr[x]=pre[x];
     pl[pre[x]]=x;x=p;
   }
}
void KM(int S){
   memset(flag,0,sizeof(flag));
   memset(slack,63,sizeof(slack));
   int l=1,r=2;q[1]=S;flag[S]=1;
   while(1)
   {
      while(l<r)
      {
         int x=q[l++];
         for(int y=1;y<=n;y++)
         {
           int tmp=labl[x]+labr[y]-e[x][y];
            if(flag[pr[y]]||slack[y]<tmp)continue;
           pre[y]=x;
            if(!tmp)
            {
               if(!pr[y])return find(y);
               q[r]=pr[y];flag[q[r]]=1;r++;
            }
            else slack[y]=tmp;
        }
      }
      int delta=inf;
      for(int i=1;i<=n;i++)
         if(!flag[pr[i]])delta=min(delta,slack[i]);
     for(int i=1;i<=n;i++)
     {
        if(flag[i])labl[i]-=delta;
        if(flag[pr[i]])labr[i]+=delta;
        else slack[i]-=delta;
     }
     for(int i=1;i<=n;i++)
        if(!flag[pr[i]]&&!slack[i])
        {
          if(!pr[i])return find(i);
          q[r]=pr[i];flag[q[r]]=1;r++;
        }
   }
}
void KM(){
   for(int i=1;i<=n;i++)
      for(int j=1;j<=n;j++)
        labl[i]=max(labl[i],e[i][j]);
   for(int i=1;i<=n;i++)KM(i);
}

int main()
{
   int a,b,c;
   scanf("%d%d%d",&n1,&n2,&m);n=max(n1,n2);
   for(int i=1;i<=m;i++)
      scanf("%d%d%d",&a,&b,&c),e[a][b]=c;
   KM();
   for(int i=1;i<=n;i++)ans+=labl[i]+labr[i];
   printf("%lld\n",ans);
   for(int i=1;i<=n1;i++)printf("%d ",e[i][pl[i]]?pl[i]:0);
   printf("\n");
   return 0;
}
```

## 带花树（一般图最大匹配）

```
int n,m,ans,cnt,check[N],fa[N],parent[N],mate[N];
int k,la[N],ff[M],l,r,q[M],flag[N];
struct node{int a,b;}map[M];
//加双向边，略
int find(int x){
   if(fa[x]==x)return x;
   return fa[x]=find(fa[x]);
}

int lca(int x,int y){
   x=find(x);y=find(y);cnt++;
   while(1)
   {
      if(check[x]==cnt)return x;
     if(x)check[x]=cnt,x=find(parent[mate[x]]);
     if(check[y]==cnt)return y;
     if(y)check[y]=cnt,y=find(parent[mate[y]]);
   }
}

void merge(int x,int y,int t){
   while(find(x)!=t)
   {
     parent[x]=y;
     if(flag[mate[x]]==2)
        q[r]=mate[x],flag[q[r]]=1,r++;
      if(find(x)==x)fa[x]=t;
     if(find(mate[x])==mate[x])fa[mate[x]]=t;
     y=mate[x];x=parent[y];
   }
}

int bfs(int S){
   for(int i=1;i<=n;i++)fa[i]=i,flag[i]=0;
   l=1;r=2;q[1]=S;flag[S]=1;parent[S]=0;
   while(l<r)
   {
      int x=q[l];l++;
     for(int a=la[x];a;a=ff[a])
     {
        int y=map[a].b;
        if(!flag[y])
        {
          parent[y]=x;flag[y]=2;
          if(!mate[y])
          {
             while(y)
             {
                int t=mate[parent[y]];
                mate[y]=parent[y];
                mate[parent[y]]=y;y=t;
             }
             return 1;
          }
          q[r]=mate[y];flag[q[r]]=1;r++;continue;
        }
        if(flag[y]==1&&find(x)!=find(y))
        {
          int t=lca(x,y);
          merge(x,y,t);merge(y,x,t);
        }
      }
   }
   return 0;
}

int main(){
   int a,b;
   scanf("%d%d",&n,&m);
   for(int i=1;i<=m;i++)scanf("%d%d",&a,&b),add(a,b);
   for(int i=1;i<=n;i++)if(!mate[i])ans+=bfs(i);
   printf("%d\n",ans);
   for(int i=1;i<=n;i++)printf("%d ",mate[i]);
   printf("\n");
   return 0;
}
```

## 匈牙利算法（二分图最大匹配）

```
#include<bits/stdc++.h>
#define N 1005
using namespace std;
int n1,n2,m,ans,e[N][N],mate[N],flag[N];

int dfs(int x)
{
   for(int i=1;i<=n1;i++)
      if(!flag[i]&&e[x][i])
      {
        flag[i]=1;
        if(!mate[i]||dfs(mate[i]))
           {mate[i]=x;return 1;}
      }
   return false;
}

int main()
{
   int a,b;
   scanf("%d%d%d",&n1,&n2,&m);
   for(int i=1;i<=m;i++)
      scanf("%d%d",&a,&b),e[b][a]=1;
   for(int i=1;i<=n2;i++)
   {
      memset(flag,0,sizeof(flag));
      ans+=dfs(i);
   }
   printf("%d\n",ans);
   for(int i=1;i<=n1;i++)printf("%d ",mate[i]);
   printf("\n");
   return 0;
}
```

## 单纯形

```
int n,m,pre[N],s[M][N];
void pivot(int x,int y){
   int pos=0;
   for(int i=1;i<=n;i++)
      if(s[x][i])pre[i]=pos,pos=i;
   for(int i=0;i<=m;i++)
      if(i!=x&&s[i][y]){
         for(int j=pos;j>=0;j=pre[j])
            if(j!=y)s[i][j]+=s[i][y]*s[x][j];
         s[i][y]*=s[x][y];
      }
}
int simplex(){
   pre[0]=-1;
   while(1){
      int pos=0,po;double minn=inf;
      for(int i=1;i<=n;i++)
         if(s[0][i]>0){pos=i;break;}
      if(!pos)return s[0][0];
      for(int i=1;i<=m;i++)
         if(s[i][pos]<0&&-s[i][0]/s[i][pos]<minn)
            po=i,minn=-s[i][0]/s[i][pos];
      if(minn==inf)return inf;
      pivot(po,pos);
   }
}
int main(){
   int a,b,c;
   scanf("%d%d",&n,&m);
   for(int i=1;i<=n;i++)scanf("%d",&s[0][i]);
   for(int i=1;i<=m;i++){
      scanf("%d%d%d",&a,&b,&c);s[i][0]=c;
      for(int j=a;j<=b;j++)s[i][j]=-1;
   }
   printf("%d\n",simplex());
   return 0;
}
```

## 有向图欧拉回路与无向图欧拉回路的判定和求解

```
int n,m,cd[N],rd[N],d[N],tot,ans[M];
int Tx,k=1,la[N],ff[M],now[N],used[M];
struct node{int a,b;}e[M];
//单向边加边，略

int get(int e){
   return (e&1)?-(e>>1):(e>>1);
}
//无向边正向走返回+，反向走返回-

void dfs1(int x){
   for(;now[x];now[x]=ff[now[x]]){
      int a=now[x];
      if(used[a])continue;
       used[a]=1;used[a^1]=1;
      dfs1(e[a].b);ans[++tot]=get(a);
   }
}

bool solve1(){
   int a,b;
   scanf("%d%d",&n,&m);
   for(int i=1;i<=m;i++){
      scanf("%d%d",&a,&b);
     add(a,b);add(b,a);d[a]++;d[b]++;
   }
   for(int i=1;i<=n;i++)
      if(d[i]&1)return false;
   for(int i=1;i<=n;i++){
      if(!la[i])continue;
     dfs1(i);
      if(tot<m)return false;
      return true;
   }
}
//无向图欧拉回路

void dfs2(int x){
   for(;now[x];now[x]=ff[now[x]]){
     int a=now[x];
      if(used[a])continue;
      used[a]=1;dfs2(e[a].b);
      ans[++tot]=a-1;
   }
}

bool solve2(){
   int a,b;
   scanf("%d%d",&n,&m);
   for(int i=1;i<=m;i++){
      scanf("%d%d",&a,&b);
     add(a,b);cd[a]++;rd[b]++;
   }
   for(int i=1;i<=n;i++)
      if(rd[i]!=cd[i])return false;
   for(int i=1;i<=n;i++){
      if(!la[i])continue;
     dfs2(i);
      if(tot<m)return false;
      return true;
   }
}
//有向图欧拉回路

int main(){
   scanf("%d",&Tx);
   bool tmp=(Tx==1)?solve1():solve2();
   if(!tmp)printf("NO\n");//没有欧拉回路
   else{
      printf("YES\n");
      for(int i=m;i;i--)printf("%d ",ans[i]);//第 i 条边为哪条边
      printf("\n");
   }
   return 0;
}
```

## 最小割树

```
int n,m,S,T,po[N],lx[N],rx[N],ans[N][N];
int k=1,la[N],ff[M],q[N],dep[N],flag[N],out[N*N];
struct node{int a,b,c;}map[M];

void add(int a,int b,int c)
{
   map[++k]=(node){a,b,c};ff[k]=la[a];la[a]=k;
   map[++k]=(node){b,a,c};ff[k]=la[b];la[b]=k;
}

//接网络流的 bfs 和 dfs

void dfs(int x)
{
   flag[x]=1;
   for(int a=la[x];a;a=ff[a])
      if(map[a].c&&!flag[map[a].b])dfs(map[a].b);
}

void solve(int l,int r)
{
   if(l==r)return;
   int res=0;S=po[l];T=po[r];
   for(int i=2;i<=k;i+=2)
      map[i].c=map[i^1].c=(map[i].c+map[i^1].c>>1);
   while(bfs())res+=dfs(S,inf);
   memset(flag,0,sizeof(flag));dfs(S);
   for(int i=1;i<=n;i++)if(flag[i])
      for(int j=1;j<=n;j++)if(!flag[j])
         ans[i][j]=ans[j][i]=min(ans[i][j],res);
   int l1=0,l2=0;
   for(int i=l;i<=r;i++)
   {
      if(flag[po[i]])lx[++l1]=po[i];
      else rx[++l2]=po[i];
   }
   for(int i=l,j=1;j<=l1;i++,j++)po[i]=lx[j];
   for(int i=l+l1,j=1;j<=l2;i++,j++)po[i]=rx[j];
   solve(l,l+l1-1);solve(l+l1,r);
}

int main()
{
   int a,b,c,x,res;
   scanf("%d%d",&n,&m);
   for(int i=1;i<=n;i++)po[i]=i;
   for(int i=1;i<=m;i++)
      scanf("%d%d%d",&a,&b,&c),add(a,b,c);
   memset(ans,127,sizeof(ans));
   solve(1,n);
   for(int i=1;i<=n;i++)
   {
      for(int j=i+1;j<=n;j++)printf("%d ",ans[i][j]);
      printf("\n");
   }
   return 0;
}
```

## tarjen 求点双，边双，割点，桥边

```
#include<bits/stdc++.h>
#define N 100005
using namespace std;
int n,m,cnt,q[N],top,dfn[N],low[N],flag[N];
int tot1,blo1[N],tot2,blo2[N];
int k=1,la[N],ff[N],check[N],iscut[N],isbridge[N];
struct node{int a,b;}e[N];

void add(int a,int b)
{
   e[++k]=(node){a,b};ff[k]=la[a];la[a]=k;
   e[++k]=(node){b,a};ff[k]=la[b];la[b]=k;
}

void dfs(int x,int pre)
{
   dfn[x]=++cnt;low[x]=cnt;
   for(int a=la[x];a;a=ff[a])
   {
      int ed=(a>>1);
       if(!flag[ed])flag[ed]=1,q[++top]=ed;//把边入栈
      if(!dfn[e[a].b])
      {
         dfs(e[a].b,ed);
         low[x]=min(low[x],low[e[a].b]);
         if(low[e[a].b]>=dfn[x])
         {
            iscut[x]++;
            //计算点割掉后多产生的联通块个数
            tot1++;
             while(top&&q[top]!=ed)blo1[q[top]]=tot1,top--;
             blo1[ed]=tot1;top--;
         }
         //点双联通分量
         if(low[e[a].b]>dfn[x])isbridge[ed]=1;
         //判断割边
      }
      else if(ed!=pre)low[x]=min(low[x],dfn[e[a].b]);
   }
}

void solve(int x,int now)
{
   check[x]=1;blo2[x]=now;
   for(int a=la[x];a;a=ff[a])
      if(!check[e[a].b]&&!isbridge[a>>1])
         solve(e[a].b,now);
}
//求边双联通分量

int main()
{
   int a,b;
   scanf("%d%d",&n,&m);
   for(int i=1;i<=m;i++)
      scanf("%d%d",&a,&b),add(a,b);
   for(int i=1;i<=n;i++)if(!dfn[i]){
      dfs(i,0);//求割点桥边点双
      if(iscut[i]==1)iscut[i]=0;
   }
   for(int i=1;i<=n;i++)if(!check[i])solve(i,++tot2);
   //求边双
   for(int i=1;i<=m;i++)cout<<blo1[i]<<' ';cout<<endl;
   //blo1  每条边属于的点双联通分量
   for(int i=1;i<=n;i++)cout<<blo2[i]<<' ';cout<<endl;
   //blo2  每个点属于的边双联通分量
   for(int i=1;i<=n;i++)cout<<iscut[i]<<' ';cout<<endl;
   //iscut  去掉这个点后分成的联通块数量
   for(int i=1;i<=m;i++)cout<<isbridge[i]<<' ';cout<<endl;
   //isbridge  是否桥边
   return 0;
}
```

# 有向图 Tar jen 缩点

```
int n,m,dfn[N],low[N],q[N],blo[N],flag[N];
int k,ff[N],la[N],cnt,tot,top,tag[N];
struct node{int a,b;}map[N];
void add(int a,int b){map[++k]=(node){a,b};ff[k]=la[a];la[a]=k;}
void dfs(int x)
{
    dfn[x]=++cnt;low[x]=cnt;
    flag[x]=1;q[++top]=x;tag[x]=1;
    for(int a=la[x];a;a=ff[a])
    {
        if(!flag[map[a].b])
            dfs(map[a].b),low[x]=min(low[x],low[map[a].b]);
        else
            if(tag[map[a].b])low[x]=min(low[x],dfn[map[a].b]);
    }
    if(low[x]==dfn[x])
    {
        tot++;
        while(q[top]!=x)blo[q[top]]=tot,tag[q[top]]=0,top--;
        blo[x]=tot;tag[x]=0;top--;
    }
}
int main()
{
    int a,b;
    scanf("%d%d",&n,&m);
    for(int i=1;i<=m;i++)scanf("%d%d",&a,&b),add(a,b);
    for(int i=1;i<=n;i++)if(!flag[i])dfs(i);
    for(int i=1;i<=n;i++)cout<<blo[i]<<' ';cout<<endl;
    return 0;
}
```

# 数论

## 线性筛求欧拉函数(phi)、莫比乌斯(u)、因数个数（sum)、最大因子重数（e）

```
const int N=50000;
int tot,n,ans,check[50010],f[50010],p[50010];
int main()
{
   int i,j,a,b,d,pl,k;f[1]=1;
   u[1]=1;sum[1]=1;
   for(int i=2;i<=N;i++)
   {
      if(!check[i])
     {
        prime[++tot]=i;
        phi[i]=i-1;u[i]=-1;
        e[i]=1;sum[i]=2;
      }
      for(int j=1;j<=tot;j++)
      {
        if(i*prime[j]>N)break;
        check[i*prime[j]]=1;
        if(i%prime[j]==0)
        {
          phi[i*prime[j]]=phi[i]*prime[j];
           u[i*prime[j]]=0;
           sum[i*prime[j]]=sum[i]/(e[i]+1)*(e[i]+2);
          e[i*prime[j]]=e[i]+1;
          break;
        }
        phi[i*prime[j]]=phi[i]*(prime[j]-1);
        u[i*prime[j]]=-u[i];
        e[i*prime[j]]=1;
        sum[i*prime[j]]=sum[i]*sum[prime[j]];
     }
   }
   return 0;
}
```

## 欧拉函数（直接求）

```
int n;ll ans;
ll phi(int x)
{
   ll res=x;
   for(int i=2;i*i<=x;i++)
      if(!(x%i))
      {
         res=res/i*(i-1);
         while(!(x%i))x/=i;
      }
   if(x>1)res=res/x*(x-1);
   return res;
}

int main()
{
   return 0;
}
```

## Exgcd//ax+by==gcd(a,b),求 x,y

```
int a,b;
void exgcd(int a,int b,int &x,int &y)
{
   if(b==0){x=1;y=0;return;}
   gcd(b,a%b,x,y);
   int t=x;x=y;y=t-a/b*y;
}
int main()
{
   scanf("%d%d",&a,&b);
   exgcd(a,b,x,y);//x=a%b 意义下逆元
   return 0;
}
```

a%p 意义下逆元：ax==1(mod p)->ax+py==1,x 为逆元(gcd(a,p)==1 时存在，否则不存在逆元)

## Pollard_rho 求一个数的因子+miller_rabin 素性测试=大数质因数分解

```
int cnt;ll s[100];
inline ll gcd(ll a,ll b)
{
   if(!b)return a;
   return gcd(b,a%b);
}

inline ll Pow(ll a,ll b,ll mod)
{
   ll res=1;
   while(b)
   {
      if(b&1)res=res*a%mod;
      a=a*a%mod;b>>=1;
   }
   return res;
}//快速幂

inline ll add(ll a,ll b,ll mod)
{return (a+=b)>=mod?a-mod:a;}

inline ll mul(ll a,ll b,ll n)
{
   ll res=0;a%=n;
   while(b)
   {
      if(b&1)res=add(res,a,n);
      a=(a<<1)%n;b/=2;
   }
   return res;
}//快速乘

inline ll Pollard_rho(ll n,ll c)
{
   ll i=1,k=2,x,y,d,p;
   x=rand()%(n-1)+1;y=x;
   while(1)
   {
      i++;x=add(mul(x,x,n),c,n);
      if(y==x)return n;
      p=abs(x-y);d=gcd(p,n);
      if(d!=1&&d!=n)return d;
      if(i==k)y=x,k+=k;
   }
}//求一个数的任意因子

const int T=10;
inline bool miller_rabin(ll n)
{
   if(n==2)return 1;
   int i,j,k=0;ll x,pre,u=n-1;
   while(!(u&1))k++,u>>=1;
   srand(9999);
   for(i=1;i<=T;i++)
   {
      x=rand()%(n-2)+2;
      if(!(x%n))continue;
      x=Pow(x,u,n);pre=x;
      for(j=0;j<k;++j)
     {
         x=mul(x,x,n);
         if(x==1&&pre!=1&&pre!=n-1)return 0;
         pre=x;
      }
      if(x!=1)return 0;
   }
   return 1;
}//素性测试

inline void solve(ll x,int c)
{
   if(x==1)return;
   if(miller_rabin(x)){s[++cnt]=x;return;}
   ll p=x,k=c;
```

```
    while(p>=x)p=Pollard_rho(p,c--);
    solve(p,k);
    solve(x/p,k);
}

int main()
{
    ll n;
    cin>>n;
    cnt=0;solve(n,5);
    for(int i=1;i<=cnt;i++)cout<<s[i]<<' ';cout<<endl;
    return 0;
}
```

## 卢卡斯定理 c(a,b)%p

```
int c(int a,int b,int p)
{
    if(b>a) return 0;
    return fac[a]*Pow((ll)fac[b]*fac[a-b],pp-2,p)%p;
}

int lucas(int x,int y,int p)
{
    int k=0,ans=1;
    while(x)a[++k]=x%p,x/=p;
    for(ll i=1;i<=k;i++) b[i]=y%p,y/=p;
    for(ll i=1;i<=k;i++) ans=((ll)ans*c(a[i],b[i],p))%p;
    return ans;
}
```

## 矩阵乘法快速幂

```
int N;
struct matrix{int s[12][12];}t,c,ans;

matrix mul(matrix a,matirx b)
{
    int tmp;
    for(int i=1;i<=N;i++)
        for(int j=1;j<=N;j++)
        {
            tmp=0;
            for(int k=1;k<=N;k++)
                tmp=(tmp+a.s[i][k]*b.s[k][j])%mod;
            c.s[i][j]=tmp;
        }
    return c;
}

void pow(matirx t,ll b)
{
    while(b)
    {
        if(b&1)ans=mul(ans,t);
        t=mul(t,t);b>>=1;
    }
}
```

## 高斯消元

```
//浮点数版
void gauss()
{
   for(int i=1;i<=n;i++)
   {
      int pos=i;
      for(int j=i+1;j<=n;j++)
         if(fabs(s[j][i])>fabs(s[pos][i]))pos=j;
      if(pos!=i)
         for(int j=i;j<=n+1;j++)swap(s[i][j],s[pos][j]);
      for(int j=i+1;j<=n;j++)
      {
         double num=s[j][i]/s[i][i];
         for(int k=i;k<=n+1;k++)s[j][k]-=s[i][k]*num;
      }
   }
   for(int i=n;i;i--)
   {
      for(int j=i+1;j<=n;j++)s[i][n+1]-=s[i][j]*ans[j];
      ans[i]=s[i][n+1]/s[i][i];
   }
}
//模意义下，模数是质数用逆元
//模数不是质数，第 i 行*lcm(s[i][i],s[j][i])/s[i][i]，第 j 行*lcm(s[i][i],s[j][i])/s[j][i]
//模意义下，模数不是质数，求行列式（不能除）→:辗转相除
void gauss()
{
   int inv=0;
   for(int i=1;i<=cnt;i++)
   {
      int pos=i;
      for(int j=i+1;j<=cnt;j++)
         if(s[j][i]>s[pos][i])pos=j;
      if(i!=pos)swap(s[i],s[pos]);
      for(int j=i+1;j<=cnt;j++)
      {
         while(s[j][i])
         {
            int tmp=s[j][i]/s[i][i];
            for(int k=i;k<=cnt;k++)
               s[j][k]=(s[j][k]-tmp*s[i][k]%mod+mod)%mod;
            if(!s[j][i])break;
            swap(s[i],s[j]);inv^=1;
         }
         if(!s[i][i]){ans=0;return;}
      }
   }
   for(int i=1;i<=cnt;i++)ans=(ll)ans*s[i][i]%mod;
   if(inv)ans=mod-ans;
}
```

## 中国剩余定理

```
int T,tot,s[N],m[N],t[N],a[N],M=1,ans;
void exgcd(int a,int b,int &x,int &y)
{
   if(b==0){x=1;y=0;return;}
   exgcd(b,a%b,x,y);
   int t=x;x=y;y=t-a/b*y;
}
//x===s[i](mod p[i])，求 x
//求模 mod 意义下的解（题目没给则 mod= π p[i]）
//前提模数互质
ll CRT()
{
   ll w,x,y,res=0;
   for(int i=1;i<=k;i++)
   {
      w=mod/p[i];exgcd(w,p[i],x,y);
      res=(res+w*x*s[i])%mod;
   }
   return (res+mod)%mod;
}
```

## 同余方程组求解（模数不互质）

```
void exgcd(ll a,ll b,ll &d,ll &x,ll &y)
{
   if(b==0){d=a;x=1;y=0;return;}
   exgcd(b,a%b,d,y,x);y-=x*(a/b);
}
int main()
{
   ll a1,a2,r1,r2;bool o=true;
   for(int i=1;i<=k;i++)
   {
      if(i==1)
      {
         scanf("%lld%lld",&a1,&r1);//a 为模数，r 为余数
         continue;
      }
      scanf("%lld%lld",&a2,&r2);//a 为模数，r 为余数
      ll x,y,d;
      exgcd(a1,a2,d,x,y);
      if ((r2-r1)%d) o=false;//如果不符合要求，标记
      x=x*((r2-r1)/d);
      ll m=a2/d;
      x=(x%m+m)%m;
      r1=r1+a1*x;//更新余数
      a1=a1/d*a2;//更新模数
      r1=(r1%a1+a1)%a1;
   }
   if(o)printf("%lld\n",r1);
   else printf("-1\n");//无解
   return 0;
}
```

## 大步小步法（BSGS）//求 y^x==z(mod p)的 x

```
void solve2(int y,int z,int p)
{
   if(!y&&!z){printf("1\n");return;}
   if(!y){printf("Orz, I cannot find x!\n");return;}//无解
   ll m=ceil(sqrt(p)),sum=1,res,tmp;
   w.clear();w[1]=m+1;
   for(int i=1;i<m;i++)
   {
      sum=(ll)sum*y%p;
      if(!w[sum])w[sum]=i;
   }
   res=Pow(y,p-m-1,p);sum=1;
   for(int i=0;i<m;i++)
   {
      tmp=w[z*sum%p];
      if(tmp)
      {
         if(tmp==m+1)tmp=0;
         printf("%lld\n",m*i+tmp);
         return;
      }
      sum=sum*res%p;
   }
   printf("Orz, I cannot find x!\n");//无解
}
```

## 二次剩余（x^2==n(mod p)，输入 n,p 求 x）

```
int n,p;ll w;
struct number{ll r,i;};//实数
number mul(number a,number b,int mod){
    ll x=(a.r*b.r%mod+a.i*b.i%mod*w%mod)%mod;
    ll y=(a.r*b.i%mod+a.i*b.r%mod+mod)%mod;
    return (number){x,y};
}//实数乘法
number r_Pow(number a,int b,int mod){
    number res=(number){1,0};
    while(b){
        if(b&1)res=mul(res,a,mod);
        a=mul(a,a,mod);b>>=1;
    }
    return res;
}//实数快速幕
int legendre(int a,int p){return Pow(a,(p-1)>>1,p);}
//勒让德符号 返回 1 是二次剩余，-1（mod-1）不是二次剩余
int solve(int n,int p){
    if(p==2)return 1;
    if(legendre(n,p)==p-1)return -1;
    int a;
    while(1){
        a=rand()%p;w=(a*a%p-n+p)%p;
        if(legendre(w,p)==p-1)break;
    }
    return r_Pow((number){a,1},(p+1)>>1,p).r;
}
int main(){
    scanf("%d%d",&n,&p);
    printf("%d\n",solve(n,p));    //有 ans 和 p-ans 两个根    -1 无解
    return 0;
}
```

## 组合数模非素数

```
int n,mod,l,r,tot,ans,p[M],P[M],s[M];
struct info{ll num;int cnt;}fac[M][N];
//接快速幂(Pow)

ll inv(ll a,ll b){return Pow(a,b-2,b);}

void exgcd(ll a,ll b,ll &x,ll &y)
{
    if(b==0){x=1;y=0;return;}
    exgcd(b,a%b,x,y);ll t=x;x=y;y=t-a/b*y;
}

ll cal(int k,ll n,ll m)
{
    info a=fac[k][n],b=fac[k][m],c=fac[k][n-m];
    ll res=Pow(p[k],a.cnt-b.cnt-c.cnt,P[k])*a.num%P[k];
    res=res*inv(b.num,P[k])%P[k];
    res=res*inv(c.num,P[k])%P[k];
    return res;
}



ll CRT()
{
    ll w,x,y,res=0;
    for(int i=1;i<=tot;i++)
    {
        w=mod/P[i];exgcd(w,P[i],x,y);
        res=(res+w*x*s[i])%mod;
    }
    return (res+mod)%mod;
}//中国剩余合并
```

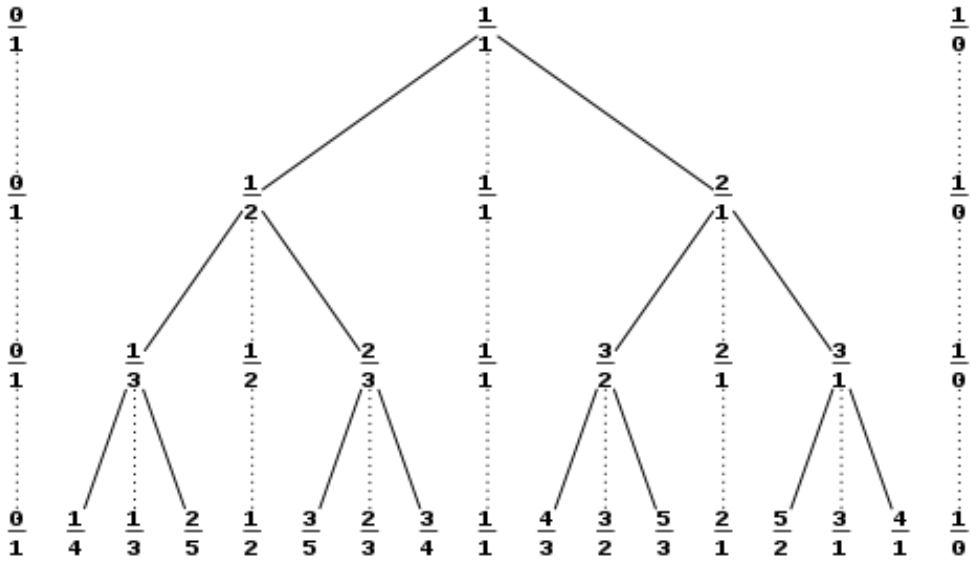
```
ll C(ll n,ll m)
{
    if(n<m||m<0)return 0;
    for(int i=1;i<=tot;i++)s[i]=cal(i,n,m);
    return CRT();
}//求值部分

void prepare()
{
    int x=mod;
    for(int i=2;i*i<x;i++)
    {
        if(x%i)continue;
        p[++tot]=i;P[tot]=1;
        while(!(x%i))x/=i,P[tot]*=i;
    }
    if(x>1)p[++tot]=x,P[tot]=x;//分解模数

    for(int i=1;i<=tot;i++)
    {
        fac[i][0]=(info){1,0};
        for(int j=1;j<=n;j++)
        {
            fac[i][j]=fac[i][j-1];
            int x=j,cnt=0;
            while(!(x%p[i]))x/=p[i],cnt++;
            fac[i][j].num=(fac[i][j].num*x)%p[i];
            fac[i][j].cnt+=cnt;
        }//预处理阶乘
    }
}
```

## stern brocot tree（已知 x 求最接近 x 的分数）

```
void solve(ld num)
{
    ld a1=0,b1=1,a2=1,b2=1;
    while(1)
    {
        ld A=a1+a2,B=b1+b2;
        if(B>100000)break;
        if(A>num*B)a2=A,b2=B;
        else a1=A,b1=B;
    }
    if(fabs(a1/b1-num)<fabs(a2/b2-num))ans1=a1,ans2=b1;
    else ans1=a2,ans2=b2;
}
```



## FFT（实数意义下）（ans[i]=Σa[x]*b[y](x+y=i)）

```
#define Pi atan2(0,-1)
int n,m,L,len,rev[N];
struct comp{
   double r,i;
   comp operator+(const comp &x)
   const{return (comp){r+x.r,i+x.i};}
   comp operator-(const comp &x)
   const{return (comp){r-x.r,i-x.i};}
   comp operator*(const comp &x)
   const{return (comp){r*x.r-i*x.i,r*x.i+i*x.r};}
}a[N],b[N],ans[N],t[N];

void DFT(comp *x,int n,int inv)
{
   for(int i=0;i<n;i++)t[rev[i]]=x[i];
   for(int i=0;i<n;i++)x[i]=t[i];
   for(int i=1,d=2;i<=L;i++,d<<=1)
   {
      comp w0=(comp){cos(2*Pi*inv/d),sin(2*Pi*inv/d)},w,u,v;
      for(int j=0,k;j<n;j+=d)
         for(k=j,w=( comp){1,0};k<j+(d>>1);k++,w=w*w0)
            u=x[k],v=x[k+(d>>1)]*w,x[k]=u+v,x[k+(d>>1)]=u-v;
   }
   if(inv==-1)for(int i=0;i<n;i++)x[i].r/=n;
}

int main()
{
   int x;
   scanf("%d%d",&n,&m);
   for(int i=0;i<=n;i++)
      scanf("%d",&x),a[i]=( comp){x,0};
   for(int i=0;i<=m;i++)
      scanf("%d",&x),b[i]=( comp){x,0};
   for(len=1,L=0;len<=n+m;len<<=1,L++);
   for(int i=0;i<len;i++)rev[i]=(rev[i>>1]>>1)|(i&1)<<(L-1);
   DFT(a,len,1);DFT(b,len,1);
   for(int i=0;i<len;i++)ans[i]=a[i]*b[i];
   DFT(ans,len,-1);
   for(int i=0;i<=n+m;i++)printf("%d\n",int(ans[i].r+0.5));
   return 0;
}
```

## NTT（模意义下）

```
#define ll long long
#define mod 998244353
void NTT(ll *x,int n,int inv)
{
   for(int i=0;i<n;i++)t[rev[i]]=x[i];
   for(int i=0;i<n;i++)x[i]=t[i];
   for(int i=1,d=2;i<=L;i++,d<<=1)
   {
     ll w0=Pow(3,(mod-1)/d),u,v,w;//原根是 3
      for(int j=0,k;j<n;j+=d)
         for(k=j,w=1;k<j+(d>>1);k++,w=w*w0%mod)
         {
            u=x[k];v=x[k+(d>>1)]*w%mod;
           x[k]=(u+v)%mod;x[k+(d>>1)]=(u-v+mod)%mod;
         }
   }
   if(inv==-1)
   {
     ll y=Pow(n,mod-2);reverse(x+1,x+n);
      for(int i=0;i<len;i++)x[i]=x[i]*y%mod;
   }
}

int main()
{
   NTT(a,len,1);NTT(b,len,1);
   for(int i=0;i<len;i++)a[i]=a[i]*b[i]%mod;
   NTT(a,len,-1);
}
```

## FWT（ans[i]=Σa[x](^、&、|) b[y](x+y=i)）

```
void FWT(ll *a,int n)
{
   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];
           a[i+j]=x+y;a[i+j+d]=x-y;
           //xor:a[i+j]=x+y,a[i+j+d]=x-y;
            //and:a[i+j]=x+y;
            //or:a[i+j+d]=x+y;
         }
}

void UFWT(ll *a,int n)
{
   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];
           a[i+j]=x+y>>1,a[i+j+d]=x-y>>1;
           //xor:a[i+j]=(x+y)/2,a[i+j+d]=(x-y)/2;
            //and:a[i+j]=x-y;
            //or:a[i+j+d]=y-x;
         }
}

int main()
{
   FWT(a,len);FWT(b,len);
   for(int i=0;i<=len;i++)a[i]*=b[i];
   UFWT(a,len);
}
```

## 求原根（m is prime）

```
bool check(int x)
{
   int num=m-1;
   for(int i=2;i*i<=num;i++)
   {
      if(num%i)continue;
      while(!(num%i))num/=i;
      if(Pow(x,(m-1)/i)==1)return false;
   }
   if(num>1&&Pow(x,(m-1)/num)==1)return false;
   return true;
}
int main()
{
   scanf("%d",&m);//mod m  意义下的原根
   for(int i=2;;i++)
     if(check(i)){printf("%d\n",i);break;}
   return 0;
}
```

## 类欧几里得

```
ll f(ll a,ll b,ll c,ll n)
{
   if(!c)return 0;
   ll m=(a*n+b)/c;
   if(a<c&&b<c)return n*m-f(c,c-b-1,a,m-1);
   return (a/c)*n*(n+1)/2+(b/c)*(n+1)+f(a%c,b%c,c,n);
}
//(ax+b)/c 在 0 到 n 上的正整数点个数
//记得加上  c/a+1
int main()
{
   scanf("%lld%lld%lld",&a,&b,&c);
   printf("%lld\n",f(a,c%a,b,c/a)+c/a+1);
   return 0;
}
```

## 分治 fft 多项式求逆

```
inline void inverse(int n,int *a,int *b,int *c)
{
   if(n==1)
   {
      b[0]=Pow(a[0],mod-2);
       return;
   }
   inverse((n+1)>>1,a,b,c);
   int len=0;
   for(len=1,L=0;len<=(n<<1);len<<=1,L++);
   for(int i=0;i<len;i++)rev[i]=(rev[i>>1]>>1)|(i&1)<<(L-1);
   for(int i=0;i<n;i++)c[i]=a[i];
   for(int i=n;i<len;i++)c[i]=0;
   NTT(b,len,1);NTT(c,len,1);
   for(int i=0;i<len;i++)
      b[i]=(ll)(2-(ll)c[i]*b[i]%mod+mod)%mod*b[i]%mod;
   NTT(b,len,-1);
   for(int i=n;i<len;i++)b[i]=0;
}
int main()
{
   读入 B 数组。
   for(len=1,L=0;len<=(n<<1);len<<=1,L++);
   inverse(len,B,b,s);//b 数组存 B 数组的逆，s 为临时数组
   return 0;
}
```

## 分治 fft 多项式开根号

```
void Sqrt(int n,ll *a,ll *b,ll *c)
{
   if(n==1)
   {
      b[0]=solve(a[0],mod);
      return;
   }
   Sqrt(n>>1,a,b,c);
   memset(invb,0,sizeof(invb));
   inverse(n>>1,b,invb,c);
   int len=0;
   for(len=1,L=0;len<=(n<<1);len<<=1,L++);
   for(int i=0;i<len;i++)rev[i]=(rev[i>>1]>>1)|(i&1)<<(L-1);
   for(int i=0;i<n;i++)c[i]=a[i];
   for(int i=n;i<len;i++)c[i]=0;
   NTT(c,len,1);NTT(invb,len,1);
   for(int i=0;i<len;i++)c[i]=c[i]*invb[i]%mod*inv2%mod;
   NTT(c,len,-1);
   for(int i=0;i<n;i++)b[i]=(c[i]+b[i]*inv2%mod)%mod;
   for(int i=0;i<len;i++)cout<<b[i]<<' ';cout<<endl;
}

int main()
{
   scanf("%d",&n);
   inv2=Pow(2,mod-2,mod);
   for(int i=0;i<=n;i++)scanf("%d",&a[i]);
   for(len=1,L=0;len<=(n<<1);len<<=1,L++);
   Sqrt(len,a,b,s);//a 是原多项式，b 是答案  s 是临时数组
   for(int i=0;i<=n;i++)printf("%d ",b[i]);
   printf("\n");
   return 0;
}
```

# 搜索

## IDA*搜索

ID：迭代加深。一层一层扩大深度上限。

A*：启发式搜索，增加估价函数对当前局面估计最少步数，启发搜索。

## 状压 dp 枚举子集

for(int s=S;s;s=(s-1)&S);(S 为全集，即枚举 S 的子集)

优化:子集卷积→fwt

## 乱搞版最大团/最大独立集（random_shuffle N 次，每次按点编号从前往后贪心选）

```
void check(){
   int tot=0;
   memset(flag,0,sizeof(flag));
   for(int i=1;i<=n;i++){
      int tag=0;
      for(int j=1;j<i;j++)
         if(flag[j]&&!e[i][j]){tag=1;break;}//j 在里面且 ij 没边
      if(!tag)p[++tot]=po[i],flag[i]=1;
   }
   if(tot>ans){
      ans=tot;//最大团大小
       for(int i=1;i<=tot;i++)s[i]=p[i];
   }
}
int main(){
   for(int i=1;i<=n;i++)po[i]=i;
   for(int i=1;i<=1000;i++) {//随 1000 次
       for(int i=1;i<=n;i++){
          x=rand()%n+1;y=rand()%n+1;
          swap(po[x],po[y]);
      }//随机
      check();
   }
}
```

## 模拟退火求费马点（到所有点距离和最小）

```
#include<bits/stdc++.h>
#define PI 3.1415926536
#define eps 1e-4
#define N 100005
using namespace std;
int n;double X[N],Y[N],sum1,sum2,s[N];
double get_rand(){return rand()%10000/10000.0;}
double pf(double x){return x*x;}
double get_dis(double a,double b,double c,double d){
   return sqrt(pf(a-c)+pf(b-d));
}

double cal(double x,double y)
{
   double res=0;
   for(int i=1;i<=n;i++)res+=get_dis(x,y,X[i],Y[i])*s[i];
   return res;
}

int main()
{
   scanf("%d",&n);
   for(int i=1;i<=n;i++)
      scanf("%lf%lf%lf",&X[i],&Y[i],&s[i]),sum1+=X[i],sum2+=Y[i];
   double T=100000.0,x=sum1/n,y=sum2/n;
   while(T>eps)
   {
      double ang=2.0*PI*get_rand();
      double xx=x+T*cos(ang),yy=y+T*sin(ang);
      double dE=cal(xx,yy)-cal(x,y);
      if(dE<0||exp(dE/T)<get_rand())x=xx,y=yy;//一定概率接受不优的解
      T*=0.98;//退火
   }
   printf("%.3lf %.3lf\n",x,y);
   return 0;
}
```

# 其他

## 部分高精模板（+，-，*）

```
#include<bits/stdc++.h>
#define ll long long
#define N 100005
#define M 27
#define mod 1000000000
using namespace std;
char c[1000];

struct big
{
   int len;ll s[50];
   big operator+(const big &x)const{
      big res;int l=max(len,x.len);
      memset(res.s,0,sizeof(res.s));
       for(int i=1;i<=l;i++)
      {
         res.s[i]+=s[i]+x.s[i];
         if(res.s[i]>=mod)res.s[i+1]++,res.s[i]-=mod;
      }
      while(res.s[l+1]>=mod)
         l++,res.s[l+1]+=res.s[l]/mod,res.s[l]%=mod;
      if(res.s[l+1])res.len=l+1;
      else res.len=l;
      return res;
   }
   big operator-(const big &x)const{
       big res;int l=max(len,x.len);
       memset(res.s,0,sizeof(res.s));
       for(int i=1;i<=l;i++)
       {
         res.s[i]+=s[i]-x.s[i];
         if(res.s[i]<0)res.s[i+1]--,res.s[i]+=mod;
      }
      while(!res.s[l])l--;
      res.len=l;return res;
   }
   big operator*(const big &x)const{
      big res;memset(res.s,0,sizeof(res.s));res.len=0;
      for(int i=1;i<=len;i++)
         for(int j=1,w=i;j<=x.len;j++,w++)
         {
            res.s[w]+=s[i]*x.s[j];
            if(res.s[w]>mod)

res.s[w+1]+=res.s[w]/mod,res.s[w]%=mod,res.len=max(res.len,w+1);
            res.len=max(res.len,w);
         }
      while(res.s[res.len]>mod)
         res.s[res.len+1]+=res.s[res.len]/mod,res.s[res.len]%=mod,res.len++;
      return res;
   }
   void get(){
      scanf(" %s",c+1);
      int l=strlen(c+1);ll tmp;len=0;
      while(l>=9)
      {
         tmp=0;
        for(int i=l-8;i<=l;i++)tmp=tmp*10+c[i]-48;
        l-=9;s[++len]=tmp;
      }
      tmp=0;
      for(int i=1;i<=l;i++)tmp=tmp*10+c[i]-48;
      s[++len]=tmp;
   }
   void put(){
     printf("%lld",s[len]);
      for(int i=len-1;i;i--)printf("%.9lld",s[i]);
      printf("\n");
   }
   void clear()
   {
      len=1;memset(s,0,sizeof(s));
   }
};
```

```
big make(int x)
{
   big res;res.clear();
   res.s[res.len=1]=x;
   return res;
}

int main()
{
   big A,B;
   A.get();B.get();
   big res=A*B;
   res.put();
   return 0;
}
```

## 快速读入

```
#define _(d) while(d((ch=getchar()-48)>=0))
inline int get()
{
   char ch;_(!);int x=ch;
   _() x=(x<<3)+(x<<1)+ch;return x;
}
```

## 动态规划决策单调性优化（栈+二分）

```
l=1;top=1;A[1]=0;B[1]=1;
for(int i=j;i<=n;i++)
{
   while(top>1&&F(A[top],B[top],j)>=F(i,B[top],j))top--;
   while(B[l+1]<x&&l<top)l++;
   f[i]=F(A[l],i,j);//求 f[i]; F()为转移计算函数
   int l=B[top]+1,r=n+1;//n+1
   while(r-l>1)
   {
      int mid=l+r>>1;
      if(F(i,mid,j)<F(A[top],mid,j))r=mid;
      else l=mid+1;
   }
   po=F(i,l,j)<F(A[top],l,j)?l:r;
   if(po<n+1)A[++top]=i,B[top]=po;
}、
//2D1D 可用分治，每次对 mid 找决策点，左孩子决策点<=Kmid，右孩子>=Kmid
//1D1D 只能这样写
```

## 五边形数求整数划分

```
#include<bits/stdc++.h>
#define ll long long
#define mod 1000000007
#define N 100005
using namespace std;
int T,tot,n,g[N],f[N],flag[N],inv[N];
//f[x] x 的整数划分数
//g[x]  五边形数

int solve(int x)
{
    if(x<=1)return 1;
    if(flag[x])return f[x];flag[x]=1;
    for(int i=1;g[i]<=x;i++)
    {
        f[x]+=inv[i]*solve(x-g[i]);
        if(f[x]>=mod)f[x]-=mod;
       if(f[x]<0)f[x]+=mod;
    }
    return f[x];
}

int main()
{
    scanf("%d",&T);
    for(int i=1,j=1;g[tot]<100000;i++,j=-j)
    {
        g[++tot]=i*(3*i-1)/2;inv[tot]=j;
       g[++tot]=i*(3*i+1)/2;inv[tot]=j;
    }
    while(T--)
    {
        scanf("%d",&n);
       printf("%d\n",solve(n));
    }
    return 0;
}
```