回文自动机

```
#include <bits/stdc++.h>
#define N 300100          //字符串长度
#define M 27              //字符集大小
#define LL long long
using namespace std;

struct Palindromic_Tree  //从0开始
{
    int next[N][M];   //next 指针，指向的串为当前串两端加上同一个字符构成
    int fail[N];      //fail 指针
    int cnt[N];       //节点表示的回文串个数
    int num[N];       //以“节点表示的回文串的右端点 ”为结尾的回文串个数
    int len[N];       //节点表示的回文串长度
    int s[N];         //字符串
    int last;         //添加最后一个字母后形成的最长回文串表示的节点
    int n;            //字符串长度
    int p;            //节点数
    int root0;        //偶数根
    int root1;        //奇数根

    int newnode(int l)
    {
        for (int i=0;i<M;i++) next[p][i]=0;
        cnt[p]=num[p]=0; len[p]=l;
        return p++;
    }

    void init()
    {
        p=0; root0=newnode(0); root1=newnode(-1);
        last=root0; n=0; s[n]=-1;
        fail[root0]=root1;
    }

    int get_fail(int x)
    {
        while (s[n-len[x]-1]!=s[n]) x=fail[x];
        return x;
    }

    void add(int c)
    {
```

```
        s[++n]=c; int cur=get_fail(last);
        if (!next[cur][c])
        {
            int now=newnode(len[cur]+2);
            fail[now]=next[get_fail(fail[cur])][c];
            next[cur][c]=now; num[now]=num[fail[now]]+1;
        }
        last=next[cur][c]; cnt[last]++;
    }

    void count() {for (int i=p-1;i>=0;--i) cnt[fail[i]]+=cnt[i];}   //建树后要加 count()才是真正的 cnt
}tree;

char s[N];
int n;

int main()
{
    scanf("%s",s); n=strlen(s);
    tree.init();
    for (int i=0;i<n;i++) tree.add(s[i]-'a');
    tree.count();
}
```

FFT&NTT

```
#include <bits/stdc++.h>
#define N 50100
#define P 998244353
#define G 3
#define LL long long
using namespace std;

int n,m,len;

namespace FFT
{
    struct cplx
    {
        double r,i;
        cplx(double _r=0,double _i=0):r(_r),i(_i){}
        friend cplx operator+(cplx x,cplx y){return cplx(x.r+y.r,x.i+y.i);};
        friend cplx operator-(cplx x,cplx y){return cplx(x.r-y.r,x.i-y.i);};
        friend cplx operator*(cplx x,cplx y){ return
cplx(x.r*y.r-x.i*y.i,x.r*y.i+x.i*y.r); };
    };

    const double PI=acos(-1.0);

    void FFT(cplx *a,int len,int flag)
    {
        static int rev[N*4];
        rev[0]=0;
        for (int i=1;i<len;i++) rev[i]=rev[i>>1]>>1|((i&1)?(len>>1):0);
        for (int i=0;i<len;i++) if(i<rev[i]) swap(a[i],a[rev[i]]);
        for (int l=2;l<=len;l<<=1)
        {
            cplx wn(cos(2*PI/l),flag*sin(2*PI/l));
            for (int i=0;i<len;i+=l)
            {
                cplx temp(1,0);
                for (int j=0;j<l/2;j++)
                {
                    cplx t1=a[i+j],t2=a[i+j+l/2];
                    a[i+j]=t1+temp*t2;
                    a[i+j+l/2]=t1-temp*t2;
                    temp=temp*wn;
                }
```

```
            }
        }
        if(flag==-1) for(int i=0;i<len;i++) a[i].r/=len;
    }

    void work()
    {
        int len;
        for (len=1;len<n+m;len<<=1);
        static cplx a[N*4],b[N*4];
        for (int i=0;i<n;i++) scanf("%lf",&a[i].r);
        for (int i=0;i<m;i++) scanf("%lf",&b[i].r);
        FFT(a,len,1);
        FFT(b,len,1);
        for (int i=0;i<len;i++) a[i]=a[i]*b[i];
        FFT(a,len,-1);
        for (int i=0;i<=n+m-2;i++) printf("%lld\n",(LL)(a[i].r+0.3));
    }
}

namespace NTT{
    LL getPow(LL x,LL y)
    {
        LL res=1;
        while(y)
        {
            if(y&1) res=res*x%P;
            x=x*x%P;
            y>>=1;
        }
        return res;
    }

    void FFT(LL *a,int len,int flag)
    {
        static int rev[N*4];
        rev[0]=0;
        for (int i=1;i<len;i++) rev[i]=rev[i>>1]>>1|((i&1)?(len>>1):0);
        for (int i=0;i<len;i++) if(i<rev[i]) swap(a[i],a[rev[i]]);
        for(int l=2;l<=len;l<<=1)
        {
            LL wn=getPow(G,(P-1)/l);
            for (int i=0;i<len;i+=l)
            {
```

```
                LL temp=1;
                for (int j=0;j<l/2;j++)
                {
                    LL t1=a[i+j],t2=a[i+j+l/2];
                    a[i+j]=(t1+temp*t2)%P;
                    a[i+j+l/2]=(t1-temp*t2)%P;
                    temp=temp*wn%P;
                }
            }
        }
        for(int i=0;i<len;i++) a[i]=(a[i]+P)%P;
        if(flag==-1)
        {
            for (int i=1;i<len;i++) if(i<len-i) swap(a[i],a[len-i]);
            LL invn=getPow(len,P-2);
            for(int i=0;i<len;i++) a[i]=a[i]*invn%P;
        }
    }

    void work(){
        int len;
        for (len=1;len<n+m;len<<=1);
        static LL a[N*4],b[N*4];
        for (int i=0;i<n;i++) scanf("%lld",a+i);
        for (int i=0;i<m;i++) scanf("%lld",b+i);
        FFT(a,len,1);
        FFT(b,len,1);
        for (int i=0;i<len;i++) a[i]=a[i]*b[i]%P;
        FFT(a,len,-1);
        for(int i=0;i<=n+m-2;i++) printf("%lld\n",a[i]);
    }
}

int main()
{
    freopen("1.in","r",stdin);
    int opt;
    scanf("%d %d %d",&n,&m,&opt); n++; m++;
    if (opt==0) FFT::work();
    else        NTT::work();
}
```

任意模数 FFT：

```
#include <bits/stdc++.h>
#define N 100100
#define P 998244353
#define G 3
#define LL long long
using namespace std;

 struct cplx
    {
        long double r,i;
        cplx(long double _r=0,long double _i=0):r(_r),i(_i){}
        friend cplx operator+(cplx x,cplx y){ return cplx(x.r+y.r,x.i+y.i); };
        friend cplx operator-(cplx x,cplx y){ return cplx(x.r-y.r,x.i-y.i); };
        friend cplx operator*(cplx x,cplx y){ return
cplx(x.r*y.r-x.i*y.i,x.r*y.i+x.i*y.r); };
    };

    const long double PI=acos(-1.0);

LL a[N],b[N];
cplx a1[N*4],a2[N*4],b1[N*4],b2[N*4],c1[N*4],c2[N*4],c3[N*4];
int n,m;

LL mod;

    void FFT(cplx *a,int len,int flag)
    {
        static int rev[N*4];
        rev[0]=0; for (int i=1;i<len;i++)
rev[i]=rev[i>>1]>>1|((i&1)?(len>>1):0);
        for (int i=0;i<len;i++) if(i<rev[i]) swap(a[i],a[rev[i]]);
        for (int l=2;l<=len;l<<=1)
        {
            cplx wn(cos(2*PI/l),flag*sin(2*PI/l));
            for (int i=0;i<len;i+=l)
            {
                cplx temp(1,0);
                for (int j=0;j<l/2;j++)
                {
                    cplx t1=a[i+j],t2=a[i+j+l/2];
                    a[i+j]=t1+temp*t2;
                    a[i+j+l/2]=t1-temp*t2;
```

```
                    temp=temp*wn;
                }
            }
        }
        if(flag==-1) for(int i=0;i<len;i++) a[i].r/=len;
    }

    void work()
    {
        int len; for (len=1;len<n+m;len<<=1);
        LL M=(LL)ceil(sqrt(mod));
        for (int i=0;i<n;i++) {a1[i].r=a[i]/M; a2[i].r=a[i]%M;}
        for (int i=0;i<m;i++) {b1[i].r=b[i]/M; b2[i].r=b[i]%M;}
        FFT(a1,len,1); FFT(a2,len,1);
        FFT(b1,len,1); FFT(b2,len,1);
        for (int i=0;i<len;i++)
        {
            c1[i]=a1[i]*b1[i];
            c2[i]=a1[i]*b2[i]+a2[i]*b1[i];
            c3[i]=a2[i]*b2[i];
        }
        FFT(c1,len,-1);
        FFT(c2,len,-1);
        FFT(c3,len,-1);
        for (int i=0;i<=n+m-2;i++)
        {
            LL
k1=(LL)(c1[i].r+0.3)%mod,k2=(LL)(c2[i].r+0.3)%mod,k3=(LL)(c3[i].r+0.3)%mod;
            printf("%lld ",(k1*M*M+k2*M+k3)%mod);
        }
    }

int main()
{
    scanf("%d %d %lld",&n,&m,&mod); n++; m++;
    for (int i=0;i<n;i++) scanf("%d",&a[i]);
    for (int i=0;i<m;i++) scanf("%d",&b[i]);
    work();
}
```

范德蒙德恒等式

$$\sum_{j=0}^{k}\binom{m}{j}\binom{n-m}{k-j}=\binom{n}{k}$$

高斯消元（异或方程组）

```
procedure gauss;
var
  i,j,k,l,pos:longint;
begin
  i:=1; j:=1;
  while (i<=n) and (j<=n) do
    begin
      pos:=i;
      for k:=i+1 to n do if a[k,j]>a[pos,j] then
        begin
          pos:=k; break;
        end;
      if a[pos,j]<>0 then
        begin
          for k:=j to n+1 do swap(a[i,k],a[pos,k]);
          for k:=i+1 to n do if a[k,j]=1 then
            for l:=j to n+1 do a[k,l]:=a[k,l] xor a[i,l];
          i:=i+1;
        end;
      j:=j+1;
    end;
end;
```

凸包+旋转卡壳

```
//平面最远点对距离
//凸包+旋转卡壳   注意：求凸包时，务必把极角相等的去重，只留下最远点
const
  maxn=50100;
type
  node1=
    record
      x,y:int64;
    end;
var
  p,stack:array[0..2*maxn] of node1;
  n,top:longint;
  ans:int64;

function max(a,b:int64):int64; begin if a>b then exit(a); exit(b); end;

operator -(a,b:node1)c:node1; begin c.x:=a.x-b.x; c.y:=a.y-b.y; end;

procedure swap(var a,b:node1); var t:node1; begin t:=a; a:=b; b:=t; end;

function cross(a,b:node1):int64; begin exit(a.x*b.y-a.y*b.x); end;

function cmp(a,b:node1):boolean;
var
  t1,t2:longint;
begin
  t1:=cross(a-p[1],b-p[1]);
  if t1=0 then
    begin
      if a.x=b.x then exit(a.y<b.y);
      exit(a.x=b.x);
    end;
  exit(t1>0);
end;

procedure qsort(l,r:longint);
var
  i,j:longint; mid,tmp:node1;
begin
```

```
  i:=l; j:=r; mid:=p[(i+j) shr 1];
  repeat
    while cmp(p[i],mid) do inc(i);
    while cmp(mid,p[j]) do dec(j);
    if i<=j then
      begin
        swap(p[i],p[j]); inc(i); dec(j);
      end;
  until i>j;
  if i<r then qsort(i,r); if l<j then qsort(l,j);
end;

function dis(a,b:node1):int64;
begin
  a:=a-b; exit(a.x*a.x+a.y*a.y);
end;

procedure main;
var
  i,j,cnt:longint;
begin
  read(n);
  for i:=1 to n do
    begin
      read(p[i].x,p[i].y);
      if (p[i].x<p[1].x) or ((p[i].x=p[1].x) and (p[i].y<p[1].y)) then
swap(p[1],p[i]);
    end;
  qsort(2,n);
  cnt:=2;
  for i:=3 to n do
    if cross(p[i]-p[1],p[cnt]-p[1])=0 then
      begin
        if dis(p[i],p[1])>dis(p[cnt],p[1]) then p[cnt]:=p[i];
      end
    else begin inc(cnt); p[cnt]:=p[i]; end;
//
  top:=1; stack[1]:=p[1];
  for i:=2 to n do
    begin
      while (top>1) and (cross(stack[top]-stack[top-1],p[i]-stack[top])<0) do
```

```
dec(top);
      inc(top); stack[top]:=p[i];
    end;
  for i:=1 to top do p[i]:=stack[i];
  for i:=1 to top do p[i+top]:=stack[i];
  n:=2*top; ans:=0;
  j:=1;
  for i:=1 to n-1 do
    begin
      while
abs(cross(p[i+1]-p[i],p[j+1]-p[i]))>abs(cross(p[i+1]-p[i],p[j]-p[i])) do
inc(j);
      ans:=max(ans,dis(p[i],p[j]));
      ans:=max(ans,dis(p[i+1],p[j]));
    end;
  writeln(ans);
end;

begin
  main;
end.
```

KD-tree

```
#include <bits/stdc++.h>
#define N 500100
#define inf 1<<30
#define DEG 3
using namespace std;

struct node1{int d[DEG],Max[DEG],Min[DEG],ls,rs,val,sum,f;}tree[N*2];
int ask[N][3];
int pos[N];
int n,m,tot,cnt,root,cmp_D,ans,qx,qy,qz;

bool cmp(node1 a,node1 b) {return a.d[cmp_D]<b.d[cmp_D];}

void pushup(int x)
{
    if (tree[x].ls!=0)
    {
        for (int i=0;i<DEG;i++)
        {
            tree[x].Max[i]=max(tree[x].Max[i],tree[tree[x].ls].Max[i]);
            tree[x].Min[i]=min(tree[x].Min[i],tree[tree[x].ls].Min[i]);
        }
    }
    if (tree[x].rs!=0)
    {
        for (int i=0;i<DEG;i++)
        {
            tree[x].Max[i]=max(tree[x].Max[i],tree[tree[x].rs].Max[i]);
            tree[x].Min[i]=min(tree[x].Min[i],tree[tree[x].rs].Min[i]);
        }
    }
}

int build(int l,int r,int dir,int id)
{
    if (dir>=DEG) dir-=DEG;
    int mid=(l+r)>>1;
    cmp_D=dir; std::nth_element(tree+l+1,tree+mid+1,tree+r+1,cmp);
    pos[tree[mid].f]=id;
```

```
    tree[mid].f=id;
    tree[mid].Max[0]=tree[mid].Min[0]=tree[mid].d[0];
    tree[mid].Max[1]=tree[mid].Min[1]=tree[mid].d[1];
    tree[mid].val=tree[mid].sum=0;
    if (l!=mid) tree[mid].ls=build(l,mid-1,dir+1,mid); else tree[mid].ls=0;
    if (r!=mid) tree[mid].rs=build(mid+1,r,dir+1,mid); else tree[mid].rs=0;
    pushup(mid);
    return mid;
}

/*
估价函数：
欧几里得距离下界：
sqr(max(max(X?x.Max[0],x.Min[0]?X),0))+sqr(max(max(Y?x.Max[1],x.Min[1]?Y),0))
曼哈顿距离下界：
max(x.Min[0]?X,0)+max(X?x.Max[0],0)+max(x.Min[1]?Y,0)+max(Y?x.Max[1],0)
欧几里得距离上界：
max(sqr(X?x.Min[0]),sqr(X?x.Max[0]))+max(sqr(Y?x.Min[1]),sqr(Y?x.Max[1])
曼哈顿距离上界：
max(abs(X?x.Max[0]),abs(x.Min[0]?X))+max(abs(Y?x.Max[1]),abs(x.Min[1]?Y))
*/

int dist(int now) {return abs(qx-tree[now].d[0])+abs(qy-tree[now].d[1]);}

int predict(int now)
{
    if (tree[now].Min[2]>qz) return inf;
    int res=0;
    if (qx<tree[now].Min[0]) res+=tree[now].Min[0]-qx;
    if (qx>tree[now].Max[0]) res+=qx-tree[now].Max[0];
    if (qy<tree[now].Min[1]) res+=tree[now].Min[1]-qy;
    if (qy>tree[now].Max[1]) res+=qy-tree[now].Max[1];
    return res;
}

void get(int now)
{
    int d0=dist(now),dl=inf,dr=inf;
    if (tree[now].d[2]<=qz) ans=min(ans,d0);
    if (tree[now].ls!=0) dl=predict(now);
    if (tree[now].rs!=0) dr=predict(now);
```

```
    if (dl<dr)
    {
        if (dl<ans) get(tree[now].ls);
        if (dr<ans) get(tree[now].rs);
    }
    else
    {
        if (dr<ans) get(tree[now].rs);
        if (dl<ans) get(tree[now].ls);
    }
}

void getans(int x,int y,int z)
{
    ans=inf;
    qx=x; qy=y; qz=z;
    get(root);
    printf("%d\n",ans);
}

int main()
{
    scanf("%d %d",&n,&m);
    for (int i=1;i<=n;i++)
    {
        scanf("%d %d",&tree[i].d[0],&tree[i].d[1]); tree[i].d[2]=0;
        tree[i].f=i;
    }
    tot=n;
    for (int i=1;i<=m;i++)
    {
        int opt;
        scanf("%d",&opt);
        if (opt==1)
        {
            tot++; scanf("%d %d",&tree[tot].d[0],&tree[tot].d[1]);
tree[tot].d[2]=i;
            tree[i].f=i;
        }
        else
        {
```

```
            cnt++; scanf("%d %d",&ask[cnt][0],&ask[cnt][1]); ask[cnt][2]=i;
        }
    }
    root=build(1,tot,0,0);
    for (int i=1;i<=cnt;i++) getans(ask[i][0],ask[i][1],ask[i][2]);
}
```

堆

```
priority_queue<Type,Container,Compare>

//    Type  为数据类型
//    Container  为保存数据的容器,  必须是用数组实现的容器，比如  vector,  deque  但不能用 list。STL 里面默认用的是  vector
//    Compare 为元素比较方式。.  比较方式默认用  operator<  ,  所以如果你把后面俩个参数缺省的话，优先队列就是大顶堆，队头元素最大。
```

常用的操作如下：
empty() 如果优先队列为空，则返回真
pop() 删除第一个元素
push() 加入一个元素
size() 返回优先队列中拥有的元素的个数
top() 返回优先队列中有最高优先级的元素

Treap

```
procedure lturn(var x:longint);inline; var t:longint;
begin
  t:=a[x].rs; a[x].rs:=a[t].ls; a[t].ls:=x; x:=t;
end;
procedure rturn(var x:longint);inline; var t:longint;
begin
  t:=a[x].ls; a[x].ls:=a[t].rs; a[t].rs:=x; x:=t;
end;

procedure insert(var x:longint;v:longint);
begin
  if x=0 then begin x:=newnode(v); exit; end;
  if a[x].v=v then begin inc(a[x].cnt); exit; end;
  if a[x].v>v then
    begin
      insert(a[x].ls,v);
      if a[x].rnd<a[a[x].ls].rnd then rturn(x);
    end
  else
    begin
      insert(a[x].rs,v);
      if a[x].rnd<a[a[x].rs].rnd then lturn(x);
    end;
end;

procedure delete(var x:longint;v:longint);
begin
  if x=0 then exit;
  if a[x].v=v then
    begin
      if a[x].cnt>1 then begin dec(a[x].cnt); exit; end;
      if (a[x].ls=0) or (a[x].rs=0) then x:=a[x].ls+a[x].rs else
      if a[a[x].ls].rnd>a[a[x].rs].rnd then begin rturn(x); delete(x,v); end
                                       else begin lturn(x); delete(x,v); end;
      exit;
    end;
  if a[x].v>v then delete(a[x].ls,v)
              else delete(a[x].rs,v);
end;
```

Splay

```
const
  inf=100007;
  maxq=500000;
type
  node=record
    son:array[0..1] of longint;
    same,lazy,size,v,sf,f,sum,max,lsum,rsum:longint;
  end;
var
  ch:char;
  head,tail,n,m,i,j,root,tot,x,y,z,size:longint;
  a:array[0..maxq] of node;
  v:array[1..maxq] of longint;
  que:array[1..maxq] of longint;

function newnode:longint; inline;
begin
  inc(head); if head>maxq then head:=1;
  fillchar(a[que[head]],sizeof(a[que[head]]),0);
  a[que[head]].same:=-inf; exit(que[head]);
end;

procedure recycle(t:longint); inline;
begin
  inc(tail); if tail>maxq then tail:=1;
  que[tail]:=t;
end;

function max(a,b:longint):longint; inline;
begin if a>b then exit(a); exit(b); end;

procedure swap(var a,b:longint); inline;
var
  t:longint;
begin
  t:=a; a:=b; b:=t;
end;
```

```
procedure dfs(t:longint); inline;
begin
  if t=0 then exit;
  if a[t].son[0]<>0 then dfs(a[t].son[0]);
  if a[t].son[1]<>0 then dfs(a[t].son[1]);
  recycle(t);
end;

procedure connect(x,y,d:longint); inline;
begin a[x].son[d]:=y; a[y].f:=x; a[y].sf:=d; end;

procedure pushdown(t:longint); inline;
var
  z:longint;
begin
  if t=0 then exit;
  if a[t].lazy<>0 then
  begin
    a[t].lazy:=0; swap(a[t].son[0],a[t].son[1]);
    a[a[t].son[0]].sf:=a[a[t].son[0]].sf xor 1;
    a[a[t].son[1]].sf:=a[a[t].son[1]].sf xor 1;
    swap(a[t].lsum,a[t].rsum);
    a[a[t].son[0]].lazy:=a[a[t].son[0]].lazy xor 1;
    a[a[t].son[1]].lazy:=a[a[t].son[1]].lazy xor 1;
  end;
  if a[t].same<>-inf then
  begin
    z:=a[t].same; a[t].v:=z; a[t].same:=-inf;
    a[a[t].son[0]].same:=z; a[a[t].son[1]].same:=z;
    a[t].sum:=z*a[t].size;
    a[t].max:=max(a[t].sum,z);
    a[t].lsum:=max(a[t].sum,z);
    a[t].rsum:=max(a[t].sum,z);
  end;
end;

procedure pushup(t:longint); inline;
var
  x,y:longint;
begin
```

```
  if t=0 then exit;
  pushdown(t);
  x:=a[t].son[0]; y:=a[t].son[1];
  pushdown(x); pushdown(y);
  a[t].size:=a[x].size+a[y].size+1;
  a[t].sum:=a[x].sum+a[y].sum+a[t].v;
  if x=0 then a[t].lsum:=a[t].v else a[t].lsum:=a[x].lsum;
  if y=0 then a[t].rsum:=a[t].v else a[t].rsum:=a[y].rsum;
  a[t].lsum:=max(a[t].lsum,a[x].sum+a[t].v);
  a[t].lsum:=max(a[t].lsum,a[x].sum+a[t].v+a[y].lsum);
  a[t].rsum:=max(a[t].rsum,a[y].sum+a[t].v);
  a[t].rsum:=max(a[t].rsum,a[y].sum+a[t].v+a[x].rsum);
  a[t].max:=a[t].v;
  if x<>0 then a[t].max:=max(a[t].max,a[x].max);
  if y<>0 then a[t].max:=max(a[t].max,a[y].max);
  a[t].max:=max(a[t].max,a[x].rsum+a[t].v+a[y].lsum);
  a[t].max:=max(a[t].max,a[x].rsum+a[t].v);
  a[t].max:=max(a[t].max,a[y].lsum+a[t].v);
end;

procedure rotate(t:longint);
var
  sf,rf,rs:longint;
begin
  if t=0 then exit;
  pushdown(a[t].f); pushdown(t);
  sf:=a[t].sf; rf:=a[t].f;
  rs:=a[t].son[sf xor 1];
  connect(a[rf].f,t,a[rf].sf);
  connect(rf,rs,sf); connect(t,rf,sf xor 1);
  pushup(rf); pushup(t);
end;

procedure splay(t,x:longint);
begin
  if t=0 then exit;
  pushdown(t);
  while a[t].f<>x do
  begin
    if a[a[t].f].f=x then rotate(t) else
    if a[t].sf=a[a[t].f].sf then begin rotate(a[t].f); rotate(t); end
```

```
    else begin rotate(t); rotate(t); end;
  end;
  if x=0 then root:=t;
end;

function find(t,rk:longint):longint;
var
  tmp:longint;
begin
  if rk>size then exit(0);
  pushdown(t);
  tmp:=a[a[t].son[0]].size;
  if rk=tmp+1 then exit(t);
  if rk<tmp+1 then exit(find(a[t].son[0],rk));
  exit(find(a[t].son[1],rk-tmp-1));
end;

procedure insert(pos,len:longint);
var
  i,x,y,head,tail,z:longint;
begin
  size:=size+len;
  head:=newnode; a[head].v:=v[1]; x:=head; y:=x;
  for i:=2 to len do
  begin
    y:=newnode; a[y].v:=v[i];
    connect(x,y,1); x:=y;
  end;
  tail:=y;
  if pos=0 then
  begin
    x:=root; connect(tail,x,1); root:=head;
    while tail<>head do begin pushup(tail); tail:=a[tail].f; end;
    pushup(head); splay(x,0);
  end else
  begin
    x:=find(root,pos+1); splay(x,0);
    y:=find(root,pos); splay(y,x);
    connect(y,head,1); z:=tail;
    while tail<>head do begin pushup(tail); tail:=a[tail].f; end;
    pushup(head); pushup(y); pushup(x);
```

```
    splay(z,0);
  end;
end;

procedure delete(pos,len:longint);
var
  x,y:longint;
begin
  if pos=1 then
  begin
    x:=find(root,pos+len); splay(x,0);
    dfs(a[x].son[0]);
    connect(x,0,0); pushup(x);
  end else
  begin
    x:=find(root,pos+len); splay(x,0);
    y:=find(root,pos-1); splay(y,x);
    dfs(a[y].son[1]);
    connect(y,0,1); pushup(y); pushup(x);
  end;
  size:=size-len;
end;

procedure makesame(pos,len,v:longint);
var
  x,y:longint;
begin
  if pos=1 then
  begin
    x:=find(root,pos+len); splay(x,0);
    a[a[x].son[0]].same:=v;
    pushup(a[x].son[0]); pushup(x);
  end else
  begin
    x:=find(root,pos+len); splay(x,0);
    y:=find(root,pos-1); splay(y,x);
    a[a[y].son[1]].same:=v;
    pushup(a[y].son[1]); pushup(y); pushup(x);
  end;
end;
```

```
procedure reverse(pos,len:longint);
var
  x,y:longint;
begin
  if pos=1 then
  begin
    x:=find(root,pos+len); splay(x,0);
    a[a[x].son[0]].lazy:=1;
    pushup(a[x].son[0]); pushup(x);
  end else
  begin
    x:=find(root,pos+len); splay(x,0);
    y:=find(root,pos-1);
    splay(y,x);
    a[a[y].son[1]].lazy:=1;
    pushup(a[y].son[1]);
    pushup(y);
    pushup(x);
  end;
end;

procedure asksum(pos,len:longint);
var
  x,y:longint;
begin
  if pos=1 then
  begin
    x:=find(root,pos+len); splay(x,0);
    pushdown(a[x].son[0]);
    writeln(a[a[x].son[0]].sum);
  end else
  begin
    x:=find(root,pos+len); splay(x,0);
    y:=find(root,pos-1); splay(y,x);
    pushdown(a[y].son[1]);
    writeln(a[a[y].son[1]].sum);
  end;
end;

procedure askmax; inline;
begin
```

```
  pushdown(root);
  writeln(a[root].max);
end;

begin
  readln(n,m);
  for i:=1 to n do read(v[i]); readln;
  head:=0; tail:=0;
  for i:=1 to maxq do recycle(i);
  insert(0,n);
  for i:=1 to m do
  begin
    read(ch,ch,ch);
    if ch='T' then begin readln(ch,ch,ch,ch,x,y); asksum(x,y); end else
    if ch='X' then begin readln; askmax; end else
    if ch='S' then
    begin
      read(ch,ch,ch,x,y);
      for j:=1 to y do read(v[j]); readln;
      insert(x,y);
    end else
    if ch='L' then begin readln(ch,ch,ch,x,y); delete(x,y); end else
    if ch='V' then begin readln(ch,ch,ch,ch,x,y); reverse(x,y); end else
    if ch='K' then begin readln(ch,ch,ch,ch,ch,ch,x,y,z); makesame(x,y,z); end;
  end;
end.
```

Link-cut-tree

```
const
  maxn=300100;
type
  node1=
    record
      son:array[0..1] of longint;
      form,fa,tag,v,max:longint;
      root,rev:boolean;
    end;
  node2=
    record
      x,y:longint;
    end;
var
  a:array[0..maxn] of node1;
  path:array[0..maxn] of node2;
  n,q:longint;

function max(a,b:longint):longint; begin if a>b then exit(a); exit(b); end;

procedure reverse(x:longint);
var
  t:longint;
begin
  if x=0 then exit;
  a[a[x].son[0]].form:=1;
  a[a[x].son[1]].form:=0;
  t:=a[x].son[0]; a[x].son[0]:=a[x].son[1]; a[x].son[1]:=t;
  a[x].rev:=not a[x].rev;
end;

procedure modify(x,w:longint);
begin
  if x=0 then exit;
  a[x].v:=a[x].v+w;
  a[x].tag:=a[x].tag+w;
  a[x].max:=a[x].max+w;
end;
```

```
procedure pushdown(x:longint);
begin
  if x=0 then exit;
  if a[x].rev then
    begin
      reverse(a[x].son[0]);
      reverse(a[x].son[1]);
      a[x].rev:=false;
    end;
  if a[x].tag>0 then
    begin
      modify(a[x].son[0],a[x].tag);
      modify(a[x].son[1],a[x].tag);
      a[x].tag:=0;
    end;
end;

procedure pushup(x:longint);
begin
  if x=0 then exit;
  a[x].max:=max(a[x].v,max(a[a[x].son[0]].max,a[a[x].son[1]].max));
end;

procedure rotate(x:longint);
var
  fa1,son1,form1:longint;
begin
  fa1:=a[x].fa; form1:=a[x].form; son1:=a[x].son[form1 xor 1];
  a[x].fa:=a[fa1].fa;
  if (a[x].fa<>0) and (not a[fa1].root) then
    begin
      a[a[x].fa].son[a[fa1].form]:=x; a[x].form:=a[fa1].form;
    end
  else begin a[x].root:=true; a[fa1].root:=false; end;
  a[x].son[form1 xor 1]:=fa1; a[fa1].fa:=x; a[fa1].form:=form1 xor 1;
  a[fa1].son[form1]:=son1; a[son1].fa:=fa1; a[son1].form:=form1;
  pushup(fa1);
end;

procedure dealtag(x:longint);
begin
```

```
  if x=0 then exit;
  if not a[x].root then dealtag(a[x].fa);
  pushdown(x);
end;

procedure splay(x:longint);
begin
  dealtag(x);
  while not a[x].root do
    begin
      if not a[a[x].fa].root then
        begin
          if a[x].form=a[a[x].fa].form then rotate(a[x].fa)
                                      else rotate(x);
        end;
      rotate(x);
    end;
  pushup(x);
end;

procedure access(x:longint);
var
  last:longint;
begin
  last:=0;
  while x<>0 do
    begin
      splay(x);
      a[a[x].son[1]].root:=true;
      a[x].son[1]:=last; a[last].fa:=x; a[last].form:=1; a[last].root:=false;
      last:=x; x:=a[x].fa;
    end;
  pushup(last);
end;

procedure makeroot(x:longint);
begin
  access(x); splay(x);
  reverse(x);
end;
```

```
function getroot(x:longint):longint;
begin
  access(x); splay(x);
  while a[x].son[0]<>0 do x:=a[x].son[0];
  exit(x);
end;

procedure link(x,y:longint);
begin
  makeroot(x); makeroot(y);
  a[x].fa:=y; access(x); splay(x);
end;

procedure cut(x,y:longint);
begin
  makeroot(y);
  access(x); splay(x);
  y:=a[x].son[0]; a[y].root:=true; a[y].fa:=0; a[x].son[0]:=0;
  pushup(x);
end;

procedure add(x,y,w:longint);
begin
  makeroot(y); access(x); splay(x);
  modify(x,w);
end;

procedure query(x,y:longint);
begin
  makeroot(y);
  access(x); splay(x);
  writeln(a[x].max);
end;

procedure init;
var
  i:longint;
begin
  read(n);
  for i:=1 to n-1 do read(path[i].x,path[i].y);
  for i:=1 to n do
```

```
    begin
      read(a[i].v); a[i].max:=a[i].v; a[i].root:=true;
    end;
  for i:=1 to n-1 do link(path[i].x,path[i].y);
end;

procedure solve;
var
  i,opt,w,x,y:longint;
begin
  read(q);
  for i:=1 to q do
    begin
      read(opt);
      case opt of
        1:begin
            read(x,y);
            if getroot(x)=getroot(y)  then begin writeln('-1'); continue; end;
            link(x,y);
          end;
        2:begin
            read(x,y);
            if getroot(x)<>getroot(y) then begin writeln('-1'); continue; end;
            cut(y,x);
          end;
        3:begin
            read(w,x,y);
            if getroot(x)<>getroot(y) then begin writeln('-1'); continue; end;
            add(x,y,w);
          end;
        4:begin
            read(x,y);
            if getroot(x)<>getroot(y) then begin writeln('-1'); continue; end;
            query(x,y);
          end;
      end;
    end;
end;
```

Lucas 定理： C(a,b) mod p=C(a mod p,b mod p)*C(....) mod p

扩展欧几里得：已知 a,b 求 x,y　S.t. ax+by = gcd(a, b) =d

```
function exgcd(a,b:int64;var x,y:int64):int64;
var
  tmp,ls:int64;
begin
  if b=0 then
    begin
      x:=1;
      y:=0;
      exit(a);
    end;
  ls:=exgcd(b,a mod b,x,y);
  tmp:=x;
  x:=y;
  y:=tmp-a div b*y;
  exit(ls);
end;
```

线性筛素数（含莫比乌斯函数与欧拉函数）

欧拉函数：phi[x]=（比 x 小的正整数中与 x 互质的数个数）

```
procedure getprime;
var
  i,j:longint;
begin
  mu[1]:=1; f[1]:=1;
  for i:=2 to maxn do
    begin
      if not p[i] then
        begin
          inc(prime[0]); prime[prime[0]]:=i; mu[i]:=-1; f[i]:=1-i;
          phi[i]:=i-1;
        end;
      for j:=1 to prime[0] do
        begin
          if i>maxn/prime[j] then break;
          p[i*prime[j]]:=true;
          if i mod prime[j]=0 then
            begin
              mu[i*prime[j]]:=0; phi[prime[j]*i]=phi[i]*prime[j];
              break;
            end;
          mu[i*prime[j]]:=-mu[i]; phi[prime[j]*i]:=phi[i]*(prime[j]-1);
        end;
    end;
end;
```

中国剩余定理

设正整数$m_1, m_2, ..., m_k$两两互素，则同余方程组

$$\begin{aligned} x &\equiv a_1 (mod\ m_1) \\ x &\equiv a_2 (mod\ m_2) \\ x &\equiv a_3 (mod\ m_3) \\ &\cdot \\ &\cdot \\ &\cdot \\ x &\equiv a_k (mod\ m_k) \end{aligned}$$

有整数解。并且在模$M = m_1 \cdot m_2 \cdot ... \cdot m_k$下的解是唯一的，解为

$$x \equiv (a_1 M_1 M_1^{-1} + a_2 M_2 M_2^{-1} + ... + a_k M_k M_k^{-1}) mod\ M$$

其中$M_i = M/m_i$，而M_i^{-1}为M_i模m_i的逆元。

Tarjan

```
procedure dfs(x:longint);
var
  i,k:longint;
begin
  inc(tot); dfn[x]:=tot; low[x]:=tot;
  inc(stack[0]); stack[stack[0]]:=x; instack[x]:=true;
  i:=map[x];
  while i<>-1 do
    begin
      if dfn[g[i].y]=0 then
        begin
          dfs(g[i].y); low[x]:=min(low[x],low[g[i].y]);
        end
      else if instack[g[i].y] then low[x]:=min(low[x],dfn[g[i].y]);
      i:=g[i].next;
    end;
  if low[x]=dfn[x] then
    begin
      k:=0; inc(cnt); size[cnt]:=0;
      while k<>x do
        begin
          k:=stack[stack[0]]; dec(stack[0]); instack[k]:=false;
          o[k]:=cnt; inc(size[cnt]);
        end;
    end;
end;
```

双联通分量：点双存边　边双存点

2-sat：边<x,y>：若选 x,则必选 y

差分约束系统：A-B<=C（A 为起点，向 B 连接一条权值为 C 的边）

二分图匹配（匈牙利算法） 最大匹配=最小覆盖

//dfs 求最大匹配 dfs1 求最小覆盖点集

```
int dfs(int x)
{
    for (int i=head[x];i>0;i=g[i].next) if (p[g[i].x]==0)
    {
        p[g[i].x]=1;
        if (link[g[i].x]==0||dfs(link[g[i].x]))
        {
            if (link[g[i].x]!=0)
               for (int j=head[link[g[i].x]];j>0;j=g[j].next) if (g[j].x==g[i].x)
{g[j].p=0; break;}
            g[i].p=1; link[g[i].x]=x; return 1;
        }
    }
    return 0;
}

void dfs1(int x)
{
    pd[x]=1;
    for (int i=head[x];i>0;i=g[i].next)
    {
        pd[g[i].x]=1;
        for (int j=head[g[i].x];j>0;j=g[j].next) if (pd[g[j].x]==0&&g[j].p==1)
dfs1(g[j].x);
    }
}

    for (int i=1;i<=tot;i++)// if (abs(pos[i].x)%2!=abs(pos[i].y)%2)
    {
        memset(p,0,sizeof(p));
        if (dfs(i)==1) ans++;
    }
```

基尔霍夫矩阵（无向图生成树数量）

*给定一个无向图 G，求它生成树的个数 t(G);

*

*算法思想：

*(1)G 的度数矩阵 D[G]是一个 n*n 的矩阵,并且满足:当 i≠j 时,dij=0;当 i=j 时,dij 等于 vi 的度数;

*(2)G 的邻接矩阵 A[G]是一个 n*n 的矩阵,并且满足:如果 vi,vj 之间有边直接相连,则 aij=1,否则为 0;

*定义图 G 的 Kirchhoff 矩阵 C[G]为 C[G]=D[G]-A[G];

*Matrix-Tree 定理:G 的所有不同的生成树的个数等于其 Kirchhoff 矩阵 C[G]任何一个 n-1 阶主子式的行列式的绝对值；

*所谓 n-1 阶主子式,就是对于 r(1≤r≤n),将 C[G]的第 r 行,第 r 列同时去掉后得到的新矩阵,用 Cr[G]表示;

网络流（dinic）

```
//上下界网络流：超级源 sst 超级汇 eed
(x,y,down,up)-->(sst,y,down)+(x,eed,down)+(x,y,up-down)
//有源汇可行流：+(ed,st,inf) 跑 sst 到 eed 最大流 sum1
//有源汇最大流：跑完可行流 sum1，删除(ed,st,inf)，再跑 st 到 ed 的最大流 sum2，
ans=sum1+sum2
//有源汇最小流：跑 sst 到 eed 的最大流 sum1-->+(ed,st,inf)--> 跑 sst 到 eed 最大流，
ans=(ed,st,inf)的实际流量
void addpath(int x,int y,int flow)
{
    totp++; g[totp].x=y; g[totp].flow=flow; g[totp].next=head[x]; head[x]=totp;
    totp++; g[totp].x=x; g[totp].flow=0;    g[totp].next=head[y]; head[y]=totp;
}

int bfs()
{
```

```
    for (int i=1;i<=tot;i++) {dep[i]=0; head1[i]=head[i];}
    int l=0,r=1; que[1]=st; dep[st]=1;
    while (l<r)
    {
        int x=que[++l];
        for (int i=head[x];i!=-1;i=g[i].next) if (g[i].flow>0&&dep[g[i].x]==0)
        {
            que[++r]=g[i].x; dep[g[i].x]=dep[x]+1;
            if (g[i].x==ed) return 1;
        }
    }
    return 0;
}

int dfs(int x,int maxflow)
{
    if (x==ed) return maxflow; int res=0;
    for (int i=head1[x];i!=-1;i=g[i].next,head1[x]=i) if
(g[i].flow>0&&dep[g[i].x]==dep[x]+1)
    {
        int k=dfs(g[i].x,min(maxflow,g[i].flow));
        g[i].flow-=k; g[i^1].flow+=k;
        maxflow-=k; res+=k;
        if (maxflow==0) return res;
    }
    return res;
}

int main()
{
    for (int i=1;i<=tot;i++) head[i]=-1; totp=-1;
    addpath();
    while (bfs()) ans+=dfs(st,inf);
}
```

费用流（SPFA）

```
void addpath(int x,int y,int flow,int cost)
{
    totp++; g[totp].x=y; g[totp].flow=flow; g[totp].cost=cost;
g[totp].next=head[x]; head[x]=totp;
    totp++; g[totp].x=x; g[totp].flow=0;    g[totp].cost=-cost;
g[totp].next=head[y]; head[y]=totp;
}

int spfa()
{
    memset(inque,0,sizeof(inque)); memset(vis,0,sizeof(vis));
    int l=0,r=1; que[1]=st; vis[st]=1; dis[st]=0;
    while (l!=r)
    {
        l=l%maxn+1; inque[que[l]]=0; int x=que[l];
        for (int i=head[x];i!=-1;i=g[i].next)
        {
            if (g[i].flow>0&&(vis[g[i].x]==0||dis[g[i].x]>dis[x]+g[i].cost))
            {
                vis[g[i].x]=1; dis[g[i].x]=dis[x]+g[i].cost;
                link[g[i].x]=i; fa[g[i].x]=x;
                if (inque[g[i].x]==0)
                {
                    r=r%maxn+1; que[r]=g[i].x; inque[g[i].x]=1;
                }
            }
        }
    }
    return (vis[ed]==1);
}

int main()
{
    totp=-1; for (int i=1;i<=tot;i++) head[i]=-1;
    while (spfa())
    {
        int now=ed,maxflow=inf;
        while (now!=st)
        {
```

```
            maxflow=min(maxflow,g[link[now]].flow);
            now=fa[now];
        }
        now=ed;
        while (now!=st)
        {
            g[link[now]].flow-=maxflow;
            g[link[now]^1].flow+=maxflow;
            now=fa[now];
        }
        ans+=maxflow*dis[ed];
    }
}
```

上下界网络流（无源汇　最大流）

题目大意：给 n 个点，及 m 根 pipe，每根 pipe 用来流躺液体的，单向的，每时每刻每根 pipe 流进来的物质要等于流出去的物质，要使得 m 条 pipe 组成一个循环体，里面流躺物质。并且满足每根 pipe 一定的流量限制，范围为[Li,Ri].即要满足每时刻流进来的不能超过 Ri(最大流问题)，同时最小不能低于 Li。

建图模型： 以前写的最大流默认的下界为 0，而这里的下界却不为 0，所以我们要进行再构造让每条边的下界为 0，这样做是为了方便处理。对于每根管子有一个上界容量 up 和一个下界容量 low，我们让这根管子的容量下界变为 0，上界为 up-low。可是这样做了的话流量就不守恒了，为了再次满足流量守恒，即每个节点"入流=出流"，我们增设一个超级源点 st 和一个超级终点 sd。我们开设一个数组 du[]来记录每个节点的流量情况。

du[i]=in[i]（i 节点所有入流下界之和）-out[i]（i 节点所有出流下界之和）。

当 du[i]大于 0 的时候，st 到 i 连一条流量为 du[i]的边。

当 du[i]小于 0 的时候，i 到 sd 连一条流量为-du[i]的边。

最后对（st，sd）求一次最大流即可，当所有附加边全部满流时（即 maxflow==所有 du[]>0 之和），有可行解。

上下界网络流（有源汇　最大流）

建图模型：源点 s，终点 d。超级源点 ss，超级终点 dd。首先判断是否存在满足所有边上下界的可行流，方法可以转化成无源汇有上下界的可行流问题。怎么转换呢？

增设一条从 d 到 s 没有下界容量为无穷的边，那么原图就变成了一个无源汇的循环流图。接下来的事情一样，超级源点 ss 连 i（du[i]>0），i 连超级汇点（du[i]<0），

对（ss，dd）进行一次最大流，当 maxflow 等于所有(du[]>0)之和时，有可行流，否则没有。

当有可行流时，删除超级源点 ss 和超级终点 dd，再对（s，d）进行一次最大流，此时得到的 maxflow 则为题目的解。为什么呢？因为第一次 maxflow（）只是求得所有满足下界的流量，而残留网络（s，d）路上还有许多自由流（没有和超级源点和超级汇点连接的边）没有流满，所有最终得到的 maxflow=（第一次流满下界的流+第二次能流通的自由流）。

上下界网络流（有源汇　最小流）

题目大意：有一个类似于工业加工生产的机器，起点为 1 终点为 n，中间生产环节有货物加工数量限制，输出 u v z c， 当 c 等于 1 时表示这个加工的环节必须对纽带上的货物全部加工（即上下界都为 z），c 等于 0 表示加工没有上界限制，下界为 0，求节点 1（起点）最少需要投放多少货物才能传送带正常工作。

解题思路：

1、du[i]表示 i 节点的入流之和与出流之和的差。

2、增设超级源点 st 和超级汇点 sd，连（st，du[i]（为正）），（-du[i]（为负），sd）。 ///增设超级源点和超级汇点，因为网络中规定不能有弧指向 st，也不能有流量流出 sd

3、做一次 maxflow（）。

4、源点（Sd）和起点（St）连一条容量为 oo 的边。

5、再做一次 maxflow（）。

6、当且仅当所有附加弧满载时有可行流，最后答案为 flow[（Sd->St）^1]，St 到 Sd 最大流就是 Sd 到 St 最小流

AC 自动机

```
procedure insert;
var
  i,len,now:longint;
begin
  readln(s); now:=root; len:=length(s);
  for i:=1 to len do
    begin
      if a[now].son[s[i]]=0 then a[now].son[s[i]]:=newnode;
      now:=a[now].son[s[i]];
    end;
  a[now].p:=true;
end;

procedure getpre;
var
  ch:char;
  i,head,tail,now:longint;
begin
  head:=0; tail:=1; que[1]:=root;
  while head<tail do
    begin
      inc(head); now:=que[head];
      for ch:='A' to 'Z' do
        begin
          if a[now].son[ch]>0 then
            begin
              inc(tail); que[tail]:=a[now].son[ch];
              if now=root then a[a[now].son[ch]].pre:=root
                          else a[a[now].son[ch]].pre:=a[a[now].pre].son[ch];
            end
          else
            begin
              if now=root then a[now].son[ch]:=root
                          else a[now].son[ch]:=a[a[now].pre].son[ch];
            end;
        end;
      a[now].p:=a[now].p or a[a[now].pre].p;
    end;
end;
```

kmp 算法（next[x]为 s[1..x]最长相等前后缀长度）

```
  for i:=2 to len do
    begin
      while (j>0) and (s[j+1]<>s[i]) do j:=next[j];
      if s[j+1]=s[i] then inc(j);
      next[i]:=j;
    end;
```

扩展 kmp 算法（next[x]为 s[1..len]与 s[x..len]最长相等前缀长度）

ex[x]为 ss[1..len]与 s[1..len]最长相等前缀长度

```
//get next
  k:=1;
  for i:=2 to lena do
    begin
      l:=next[i-k+1];
      if max>i+l-1 then next[i]:=l
      else
        begin
          j:=max-i+1; if j<0 then j:=0;
          while (i+j<=lena) and (sa[j+1]=sa[i+j]) do inc(j);
          next[i]:=j; k:=i; max:=k+j-1;
        end;
    end;
//get ex
  k:=0;
  for i:=1 to lenb do
    begin
      l:=next[i-k+1];
      if max>i+l-1 then ex[i]:=l
      else
        begin
          j:=max-i+1; if j<0 then j:=0;
          while (i+j<=lenb) and (j<lena) and (sa[j+1]=sb[i+j]) do inc(j);
          ex[i]:=j; k:=i; max:=k+j-1;
        end;
    end;
```

Manacher（记得补$,*）

```
procedure manacher;
var
    i,mx,id,ans:longint;
begin
    mx:=0; id:=0; ans:=0;
    for i:=1 to n do
        begin
            if mx>i then f[i]:=min(f[2*id-i],mx-i+1) else f[i]:=1;
            while (i-f[i]>0) and (i+f[i]<=n) and (s[i-f[i]]=s[i+f[i]]) do inc(f[i]);
            if i+f[i]-1>mx then
                begin
                    id:=i; mx:=i+f[i]-1;
                end;
            ans:=max(ans,f[i]);
        end;
    writeln(ans-1);
end;
```

K 短路（A*）

```
#include <bits/stdc++.h>
#define N 1010
#define M 10100
using namespace std;

struct node1{int x,next,w;}g[M];
struct node2
{
    int x,f,dis;
    node2(int x=0,int dis=0,int f=0):x(x),dis(dis),f(f) {}
    bool operator < (const node2 &A) const {return A.dis+A.f<dis+f;}
};
int head[N],cnt[N],vis[N],f[N],mp[N][N];
int n,m,st,ed,t,k,totp;
priority_queue<node2> q;

void addpath(int x,int y,int w)
{
    totp++; g[totp].x=y; g[totp].next=head[x]; g[totp].w=w; head[x]=totp;
}

void init()
{
    totp=0; memset(head,0,sizeof(head));
    memset(cnt,0,sizeof(cnt));
    memset(vis,0,sizeof(vis));
    memset(mp,0x3f,sizeof(mp));
}

void getf()
{
    memset(f,0x3f,sizeof(f)); f[ed]=0;
    for (int i=1;i<=n;i++)
    {
        int x=0;
        for (int j=1;j<=n;j++) if (!vis[j]&&f[x]>f[j]) x=j;
        vis[x]=1;
        for (int j=1;j<=n;j++) if (!vis[j]) f[j]=min(f[j],f[x]+mp[x][j]);
```

```
    }
}

void solve()
{
    while (!q.empty()) q.pop();
    q.push(node2(st,0,f[st]));
    while (!q.empty()) {
        node2 now=q.top(); q.pop();
        if (++cnt[now.x]>k) continue;
        if (cnt[ed]==k)
        {
            puts("yareyaredawa");
            return;
        }
        for (int i=head[now.x];i>0;i=g[i].next)
        {
            int y=g[i].x;
            if (cnt[y]<k&&now.dis+g[i].w<=t)
q.push(node2(y,now.dis+g[i].w,f[y]));
        }
    }
    puts("Whitesnake!");
}

void work()
{
    scanf("%d %d %d %d",&st,&ed,&k,&t);
    for (int i=1,x,y,w;i<=m;i++)
    {
        scanf("%d %d %d",&x,&y,&w);
        addpath(x,y,w); mp[y][x]=min(mp[y][x],w);
    }
    getf();
    k+=(st==ed);
    solve();
}

int main()
{
    while (scanf("%d %d",&n,&m)!=EOF)
```

```
    {
        init();
        work();
    }
}
```

后缀数组（SA）sa[i]排名 i 是第 sa[i]个后缀

rk[i]第 i 后缀的排名是 rk[i]

DC3

```
#include <bits/stdc++.h>
#define N 50100
#define F(x) ((x)/3+((x)%3==1?0:tb))
#define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
using namespace std;

char s[N];
int sa[10*N],rk[N],h[N];
int r[10*N],wa[10*N],wb[10*N],wv[10*N];
int wws[10*N];
int n;

void sort(int *r,int *a,int *b,int n,int m)
{
    int i;
    for(i=0;i<n;i++) wv[i]=r[a[i]];
    for(i=0;i<m;i++) wws[i]=0;
    for(i=0;i<n;i++) wws[wv[i]]++;
    for(i=1;i<m;i++) wws[i]+=wws[i-1];
    for(i=n-1;i>=0;i--) b[--wws[wv[i]]]=a[i];
    return;
}
int c0(int *r,int a,int b) {return r[a]==r[b]&&r[a+1]==r[b+1]&&r[a+2]==r[b+2];}

int c12(int k,int *r,int a,int b)
{
    if(k==2) return r[a]<r[b]||(r[a]==r[b]&&c12(1,r,a+1,b+1));
    else     return r[a]<r[b]||(r[a]==r[b]&&wv[a+1]<wv[b+1]);
```

```
}


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

void geth()
{
    int j=0,k; h[1]=0;
    for (int i=1;i<=n;i++) if (rk[i]>1)
    {
        k=sa[rk[i]-1];
        while (i+j<=n&&k+j<=n&&s[i+j-1]==s[k+j-1]) j++;
        h[rk[i]]=j; if (j>0) j--;
    }
}


int main()
{
    scanf("%s\n",s);
    n=strlen(s); int m=255;                    //s 从 0 开始    n 长度   m 字符集大小
```

```
    for (int i=0;i<n;i++) r[i]=(int)s[i]; r[n]=0;
    dc3(r,sa,n+1,m+1);                          //dc3 过程后 r 会被破坏
    for (int i=1;i<=n;i++) rk[sa[i]]=i;
    for (int i=1;i<=n;i++) sa[i]++;
    for (int i=n;i>0;i--) rk[i]=rk[i-1];      //sa、rk 均从下标 1 开始
    geth();
    for (int i=1;i<=n;i++) printf("%d ",rk[i]); puts("");
    for (int i=1;i<=n;i++) printf("%d ",h[i]);  puts("");
}
```

倍增

```
#include <bits/stdc++.h>
#define N 50100
using namespace std;

char s[N];
int rk[N],sa[N],trk[N],tsa[N],tmp[N],sum[N],h[N];
int n,m;

void getsa(int n,int m)
{
    memset(sum,0,sizeof(sum));
    for (int i=1;i<=n;i++) trk[i]=(int)s[i],sum[trk[i]]++;
    for (int i=1;i<=m;i++) sum[i]+=sum[i-1];
    for (int i=n;i>=1;i--) sa[sum[trk[i]]--]=i;
    int p=1; rk[sa[1]]=1;
    for (int i=2;i<=n;i++)
    {
        if (trk[sa[i]]!=trk[sa[i-1]]) p++;
        rk[sa[i]]=p;
    }
    m=p; int j=1;
    while (m<n)
    {
        memset(sum,0,sizeof(sum)); p=0;
        for (int i=n-j+1;i<=n;i++) tsa[++p]=i;
        for (int i=1;i<=n;i++) if (sa[i]>j) tsa[++p]=sa[i]-j;
        for (int i=1;i<=n;i++) tmp[i]=rk[tsa[i]],sum[tmp[i]]++;
        for (int i=1;i<=m;i++) sum[i]+=sum[i-1];
```

```
        for (int i=n;i>=1;i--) sa[sum[tmp[i]]--]=tsa[i];
        for (int i=1;i<=n;i++) trk[i]=rk[i]; p=1; rk[sa[1]]=1;
        for (int i=2;i<=n;i++)
        {
            if (trk[sa[i]]!=trk[sa[i-1]]||trk[sa[i]+j]!=trk[sa[i-1]+j]) p++;
            rk[sa[i]]=p;
        }
        m=p; j<<=1;
    }
}

void geth()
{
    h[1]=0; int j,p=0;
    for (int i=1;i<=n;i++) if (rk[i]>1)
    {
        j=sa[rk[i]-1];
        while (i+p<=n&&j+p<=n&&s[i+p]==s[j+p]) p++;
        h[rk[i]]=p; if (p>0) p--;
    }
}

int main()
{
    scanf("%s",s); n=strlen(s); m=255;
    for (int i=n;i>=1;i--) s[i]=s[i-1];
    getsa(n,m);
    geth();
    for (int i=1;i<=n;i++) printf("%d ",rk[i]); puts("");
    for (int i=1;i<=n;i++) printf("%d ",h[i]); puts("");
}
```