Template v2.0 for ACM-CCPC in JLU

zig_zag

September 23, 2016

目录

1 Math

1.1 BigInter

```
//高精度
struct high
{
  int e[100];
  void clean()
  {
    memset(e,0,sizeof(e));
  }
}a;

high operator +(high a,high b)
{
  high c; c.clean();
  c.e[0]=max(a.e[0],b.e[0]);
  for (int i=1;i<=c.e[0];i++)
  {
    c.e[i]+=a.e[i]+b.e[i];
    c.e[i+1]+=c.e[i]/p;
    c.e[i]=c.e[i]%p;
  }
  if (c.e[c.e[0]+1]) c.e[0]++;
  return c;
}

high operator -(high a, high b)
{
  high c; c.clean();
  int add=0;
  c.e[0]=max(a.e[0],b.e[0]);
  for (int i=1;i<=c.e[0];i++)
  {
    c.e[i]=a.e[i]-add-b.e[i];
    if (c.e[i]<0) c.e[i]+=p, add=1;
    else add=0;
  }
  while (c.e[c.e[0]]==0) c.e[0]--;
  return c;
}

high operator *(high a,high b)
{
  high c; c.clean();
  c.e[0]=a.e[0]+b.e[0]-1;
  for (int i=1;i<=a.e[0];i++)
    for (int j=1;j<=b.e[0];j++)
      c.e[i+j-1]+=a.e[i]*b.e[j];
  for (int i=1;i<=c.e[0];i++)
  {
    c.e[i+1]+=c.e[i]/p;
    c.e[i]=c.e[i]%p;
  }
  if (c.e[c.e[0]+1]) c.e[0]++;;
  return c;
}

high operator /(high a,int b)
{
  int down=0;
  high c; c.clean();
  c.e[0]=a.e[0];
  for (int i=a.e[0];i;i--)
  {
    c.e[i]=(a.e[i]+down*p)/b;
    down=(a.e[i]+down*p)-c.e[i]*b;
  }
  while ((c.e[c.e[0]]==0)&&(c.e[0]>0)) c.e[0]--;
  return c;
}

int operator %(high a,int b)
{
  int mod;
  for (int i=a.e[0];i;i--)
    mod=((mod*p%b)+a.e[i])%b;
  return mod;
}

high max(high a,high b)
{
  if (a.e[0]>b.e[0]) return a;
  if (a.e[0]<b.e[0]) return b;
  for (int i=a.e[0];i;i--)
  {
    if (a.e[i]>b.e[i]) return a;
    if (a.e[i]<b.e[i]) return b;
  }
  return a;
}

void read(high &a)
{
  char ch=getchar(); high x; x.clean();
  for (;ch<'0'||ch>'9';ch=getchar());
  for (;ch>='0'&&ch<='9';ch=getchar()) x.e[++x.e[0]]=ch-'0';
  for (int i=1;i<=x.e[0];i++) a.e[x.e[0]-i+1]=x.e[i];
  a.e[0]=x.e[0];
}

void write(high a)
{
  printf("%d",a.e[a.e[0]]);
  for (int i=a.e[0]-1;i>0;i--) printf("%d",a.e[i]);
  printf("\n");
}
```

1.2 BinarySearch

```
//整数二分
int l=1,r=n, ans;
while (l<=r)
{
  int mid=(l+r)>>1;
  if (judge(mid)) l=mid+1,ans=mid;
  else r=mid-1;
}
//浮点数二分
hile (l-r<=1e-6)
{
  double mid=(l+r)/2.0;
  if (judge(mid)<=0.0) r=mid-0.000001,ans=mid;//所有>0的解均不合法
  else l=mid+0.000001;
}
```

1.3 CRT

```
//中国剩余定理(互质)
LL crt(int n, LL* a, LL* p)
{
    LL pp=1, tmp=0;
    for(int i=0;i<n;i++) pp=pp*p[i];
    for(int i=0;i<n;i++)
    {
        LL m=pp/p[i], x, y;
        ex_gcd(m,p[i],x,y);
        x=(x%p[i]+p[i])%p[i];
        tmp=(tmp+(a[i]*m%pp*x)%pp)%pp;//注意overflow
    }
    return tmp;
}
```

1.4 EulerFunction

```
//欧拉函数
long long phi(long long x)
{
  long long cnt=x;
  for (long long i=2;i*i<=x;i++)
    if (x%i==0)
    {
      cnt=cnt/i*(i-1);
      while (x%i==0) x=x/i;
    }
  if (x>1) cnt=cnt/x*(x-1);
  return cnt;
}
```

1.5 EulerSieve

```
//欧拉筛法
void get_prime(int n)
{
  memset(np,0,sizeof(np));
  phi[1]=1;
  for (int i=2;i<=n;i++)
  {
    if (!np[i])
    {
      prime.push_back(i);
      phi[i]=i-1;
    }
    for (int j=0;j<prime.size()&&prime[j]*i<=n;j++)
    {
      np[prime[j]*i]=1;
      if (i%prime[j]==0)
      {
        phi[i*prime[j]]=phi[i]*prime[j];
        break;
      }
      else
        phi[i*prime[j]]=phi[i]*(prime[j]-1);
    }
  }
}
```

1.6 ExtendedEuclidean

```
//扩展欧几里得
LL ex_gcd(LL a, LL b, LL &x, LL &y)
{
    if (!b)
    {
        x=1, y=0;
        return a;
    }
    else
    {
        LL d=ex_gcd(b, a % b, y, x);
        y-=a/b*x;
        return d;
    }
}
//求乘法逆元
LL inv(LL a, LL p)
{
    LL gcd, x, y;
    gcd=ex_gcd(a, p, x, y);
    if (gcd==1) return (x+p)%p;
    else return -1;
}
```

1.7 FFT

```
//FFT
#define PI acos(-1.0)//acosl(-1.0)

typedef double LD;//long double
typedef long long LL;
//typedef complex<LD> cpx;
struct cpx
{
  LD x, y;
  cpx(){}
  cpx(LD x, LD y):x(x),y(y){}
};
cpx operator +(cpx a, cpx b)
{
  return cpx(a.x+b.x, a.y+b.y);
}
cpx operator -(cpx a, cpx b)
{
  return cpx(a.x-b.x, a.y-b.y);
}
cpx operator *(cpx a, cpx b)
{
  return cpx(a.x*b.x-a.y*b.y, a.x*b.y+a.y*b.x);
}

int rev(int x, int n)
{
    int tmp=0;
    for (int i=n>>1;i;i>>=1,x>>=1)
        tmp=tmp<<1|x&1;
    return tmp;
}

void fft(cpx *a, int n, int flag)
{
  for (int i=0,j=i;i<n;i++,j=rev(i, n))
    if (i<j) swap(a[i], a[j]);
  for (int k=1;k<n;k<<=1)
  {
    cpx wn(cos(PI/k), flag*sin(PI/k));
    //cpx wn(cosl(PI/i), flag*sinl(PI/i));
    cpx w(1, 0);
    for (int i=0;i<k;i++,w=w*wn)
      for (int j=i;j<n;j+=(k<<1))
      {
        cpx x=a[j], y=w*a[j|k];
        a[j]=x+y;
        a[j|k]=x-y;
      }
  }
  if (flag==-1)
    for (int i=0;i<n;i++)
      a[i].x/=n, a[i].y/=n;
}

cpx A[maxn], B[maxn];
int a[maxn];
int n;
void roll(int *a, int *b, int *c, int n, int m)
{
  int num=1;
  while (num<n+m) num<<=1;//move out if slow
  for (int i=0;i<num;i++) A[i]=(i<n)?cpx(a[i],0):cpx(0,0);
  for (int i=0;i<num;i++) B[i]=(i<m)?cpx(b[i],0):cpx(0,0);
  fft(A, num, 1);
  fft(B, num, 1);
  for (int i=0;i<num;i++) A[i]=A[i]*B[i];
  fft(A, num, -1);
  for (int i=0;i<num;i++) c[i]=(LL)(A[i].x+0.5)%mod;
}
```

1.8 FFT-2

```
void fft(cpx *a, int n, int flag)
{
    if (flag==-1)
    {
      for (int i=1;i<n/2;i++) swap(a[i], a[n-i]);
        for (int i=0;i<n;i++) a[i].x/=n, a[i].y/=n;
    }
    cpx wn(cos(2*PI/n), sin(2*PI/n));
    for (int s=n>>1;s;s>>=1, wn=wn*wn)
    {
      cpx w(1,0);
        for (int k=0; k<s; k++, w=w*wn)
      for (int i=k; i<n; i+=s<<1)
            {
                cpx x=a[i], y=a[i|s];
        a[i]=x+y;
        a[i|s]=(x-y)*w;
            }
    }
    for (int i=1, j=0;i<n;i++,j=0)
  {
    for (int s=i, k=n>>1; k; s>>=1, k>>=1)
      j=j<<1|s&1;
    if (i<j) swap(a[i], a[j]);
  }
}
inline void ntt(LL a[], int n, int flag)
{
  LL wn=exp(G, (mod-1)/n, mod), w=1;
  if (flag==-1)
  {
    for (int i=1;i<n/2;i++) swap(a[i], a[n-i]);
    LL inv=exp(n, mod-2, mod);
    for (int i=0;i<n;i++) a[i]=a[i]*inv%mod;
  }
  for (int s=n>>1; s; s=s>>1, wn=wn*wn%mod, w=1)
```

```
    for (int k=0; k<s; k++, w=w*wn%mod)
      for (int i=k; i<n; i+=s<<1)
      {
        LL x=a[i], y=a[i|s]%mod;
        a[i]=(x+y)%mod;
        a[i|s]=(x-y+mod)%mod*w%mod;
      }
  for (int i=1, j=0;i<n;i++,j=0)
  {
    for (int s=i, k=n>>1; k; s>>=1, k>>=1)
      j=j<<1|s&1;
    if (i<j) swap(a[i], a[j]);
  }
}
```

1.9 GCD

```
//GCD
long long gcd(long long a,long long b)
{
    return (b==0?a:gcd(b, a%b));
}
```

1.10 GaussElimination

```
//高斯消元
void gauss(int n)
{
    int k=1;
    for (int i=1;i<=n;i++)
    {
        int p=0;
        for (int j=k;j<=n;j++)
            if (fab(a[j][i])>eps) {    p=j; break;    }
        if (!p) continue;
        for (int l=1;l<=n+1;l++) swap(a[p][l],a[k][l]);
        for (int j=k+1;j<=n;j++)
        {
            lb rate=a[j][i]/a[k][i];
            for (int l=1;l<=n+1;l++)
                a[j][l]-=a[k][l]*rate;
        }
        k++;
    }
    for (int i=n;i;i--)
    {
        v[i]=a[i][n+1];
        for (int j=i+1;j<=n;j++)
            v[i]-=v[j]*a[i][j];
        v[i]/=a[i][i];
    }
}
```

1.11 LucasTheory

```
//Lucas定理(求组合数取模,注意p<=10^5)
LL com(LL n,LL m,LL p)
{
  if (m>n) return 0;
  LL ans=1;
  for (int i=1;i<=m;i++)
  {
    ans=ans*(n-i+1)%p;
    ans=ans*exp(i,p-2,p)%p;
  }
  return ans;
}

LL lucas(LL n,LL m,LL p)
{
  if (m==0) return 1;
  return (com(n%p,m%p,p)*lucas(n/p,m/p,p))%p;
}

LL get_ans(LL n, LL m)
{
  return lucas(n+m,n,oo);
}
```

1.12 MatrixQuickPower

```
//矩阵乘法&快速幂(ms<=500)
const int ms=5;
struct matrix
{
  LL e[ms][ms];
  matrix()
  {
    memset(e,0,sizeof(e));
  }
  matrix(int x)
  {
    memset(e,0,sizeof(e));
    for (int i=0;i<ms;i++) e[i][i]=x;
  }
  void clear()
  {
    memset(e,0,sizeof(e));
  }
  friend matrix operator *(matrix a, matrix b)
  {
    matrix c;
    for (int k=0;k<ms;k++)
      for (int i=0;i<ms;i++)
        for (int j=0;j<ms;j++)
          (c.e[i][j]+=a.e[i][k]*b.e[k][j]%oo)%=oo;
    return c;
  }
  friend matrix operator ^(matrix e, LL k)
  {
    matrix tmp=matrix(1);
    while (k)
    {
      if (k&1) tmp=tmp*e;
      k=k>>1;
      e=e*e;
    }
    return tmp;
  }
};
```

1.13 MillerRabin_PollardRho

```
//miller_rabin+pollard_rho分解质因数
const int S=20;//随机算法判定次数，S越大，判错概率越小
//计算 (a*b)%c.   a,b都是long long的数，直接相乘可能溢出的
//  a,b,c <2^63
long long mult_mod(long long a,long long b,long long c)
{
    a%=c;
    b%=c;
    long long ret=0;
    while(b)
    {
        if(b&1){ret+=a;ret%=c;}
        a<<=1;
        if(a>=c)a%=c;
        b>>=1;
    }
    return ret;
}

//计算  x^n %c
long long pow_mod(long long x,long long n,long long mod)//
x^n%c
{
    if(n==1)return x%mod;
    x%=mod;
    long long tmp=x;
    long long ret=1;
    while(n)
    {
        if(n&1) ret=mult_mod(ret,tmp,mod);
        tmp=mult_mod(tmp,tmp,mod);
        n>>=1;
    }
    return ret;
}

//以a为基,n-1=x*2^t      a^(n-1)=1(mod n)  验证n是不是合数
//一定是合数返回true,不一定返回false
bool check(long long a,long long n,long long x,long long t
)
```

```
{
    long long ret=pow_mod(a,x,n);
    long long last=ret;
    for(int i=1;i<=t;i++)
    {
        ret=mult_mod(ret,ret,n);
        if(ret==1&&last!=1&&last!=n-1) return true;//合数
        last=ret;
    }
    if(ret!=1) return true;
    return false;
}

// Miller_Rabin()算法素数判定
//是素数返回true.(可能是伪素数，但概率极小)
//合数返回false;

bool Miller_Rabin(long long n)
{
    if(n<2)return false;
    if(n==2)return true;
    if((n&1)==0) return false;//偶数
    long long x=n-1;
    long long t=0;
    while((x&1)==0){x>>=1;t++;}
    for(int i=0;i<S;i++)
    {
        long long a=rand()%(n-1)+1;//rand()需要stdlib.h头文件
        if(check(a,n,x,t))
            return false;//合数
    }
    return true;
}

//pollard_rho 算法进行质因数分解
long long factor[100];//质因数分解结果（刚返回时是无序的）
int tol;//质因数的个数。数组小标从0开始

long long gcd(long long a,long long b)
{
    return (b==1?a:gcd(b, a%b));
}

long long Pollard_rho(long long x,long long c)
{
    long long i=1,k=2;
    long long x0=rand()%x;
    long long y=x0;
    while(1)
    {
        i++;
        x0=(mult_mod(x0,x0,x)+c)%x;
        long long d=gcd(y-x0,x);
        if(d!=1&&d!=x) return d;
        if(y==x0) return x;
        if(i==k){y=x0;k+=k;}
    }
}
//对n进行素因子分解
void findfac(long long n)
{
    if(Miller_Rabin(n))//素数
    {
        factor[tol++]=n;
        return;
    }
    long long p=n;
    while(p>=n)p=Pollard_rho(p,rand()%(n-1)+1);
    findfac(p);
    findfac(n/p);
}

int main()
{
    //srand(time(NULL));//需要time.h头文件//POJ上G++不能加这句话
    long long n;
    while(scanf("%I64d",&n)!=EOF)
    {
        tol=0;
        findfac(n);
        for(int i=0;i<tol;i++)printf("%I64d ",factor[i]);
        printf("\n");
        if(Miller_Rabin(n))printf("Yes\n");
        else printf("No\n");
    }
    return 0;
}
```

1.14 MobiusInversion

```
//莫比乌斯反演
void get_prime(int n)
{
    memset(np,0,sizeof(np));
    tot=0;
    mu[1]=1;
    for (int i=2;i<=n;i++)
    {
        if (!np[i])
        {
            prime[++tot]=i;
            mu[i]=-1;
        }
        for (int j=1;j<=tot && i*prime[j]<=n;j++)
        {
            np[i*prime[j]]=1;
            if (i%prime[j]) mu[i*prime[j]]=-mu[i];
            else
            {
                mu[i*prime[j]]=0;
                break;
            }
        }
    }
}
//求与序列中第j个数互质数的个数
int main()
{
    for (int i=1;i<=100000;i++)
        for (int j=i;j<=100000;j+=i)
            sum[i]+=v[j];
    for (int i=1;i<=100000;i++)
    {
        for (int j=i;j<=100000;j+=i) f[j]+=sum[i]*mu[i];
        if (i==1) f[i]--;
    }
}


//已知f[n]=(求和d|n)g(d),求g nlogn
for (int i = 1; i <= n; ++i)
  for (int j = i + i; j <= n; j += i)
    f[j] -= f[i];

//f[n]=(求和d|n)g(d),已知g,求f
for (int i = n; i >= 1; --i)
  for (int j = i + i; j <= n; j += i)
    f[j] += f[i];

//已知f[n]=(求和n|d)g(d),求g nlogn
for (int i = n; i >= 1; --i)
  for (int j = i + i; j <= n; j += i)
    f[i] -= f[j];

//f[n]=(求和n|d)g(d),已知g,求f
for (int i = 1; i <= n; ++i)
  for (int j = i + i; j <= n; j += i)
    f[i] += f[j];

//f[s]存原来的元素，之后f[s]存子集所有元素和
for (int i = 0; i < n; i++)
    for (int s = 0; s < (1 << n); s++)
        if (s >> i & 1)
            f[s] += f[s ^ 1 << i];


**************************************
f[s]存子集所有元素和，之后f[s]存原来的元素

for (int i = 0; i < n; i++)
    for (int s = 0; s < (1 << n); s++)
        if (s >> i & 1)
            f[s] -= f[s ^ 1 << i];
**************************************
```

```
数论卷积nlogn  算1—n的h[x]=(求和d|x)(f[d]*g[x/d])  已知f,g的1—n

int f[MAXN],g[MAXN],h[MAXN]={0};
void calc(int n)
{
    for (int i=1;i*i<=n;i++)
        for (int j=i;i*j<=n;j++)
            if(j==i)h[i*j]+=f[i]*g[i];
            else h[i*j]+=f[i]*g[j]+f[j]*g[i];
}
```

1.15 MulInverse

```
//递归求逆元
LL rev(LL x)
{
  if (x==1) return 1;
  else return (oo-oo/x)*rev(oo%x)%oo;
}
```

1.16 NTT

```
//NTT
#define G 3
typedef long long LL;
const LL mod=998244353;
inline LL exp(LL a, LL b, LL p)
{
  a%=p;
  LL tmp=1;
  while (b)
  {
    if (b&1) tmp=(tmp*a)%p;
    a=a*a%p;
    b=b/2;
  }
  return tmp;
}
inline int rev(int x, int n)
{
    int tmp=0;
    for (int i=n>>1;i;i>>=1,x>>=1)
        tmp=tmp<<1|x&1;
    return tmp;
}
inline void ntt(LL *a, int n, int flag)
{
  for (int i=0,j=i;i<n;i++,j=rev(i, n))
    if (i<j) swap(a[i], a[j]);
  for (int k=1;k<n;k<<=1)
  {
    LL wn=exp(G, (mod-1)/(k<<1), mod), w=1;
    if (flag==-1) wn=exp(wn, mod-2, mod);
    for (int i=0;i<k;i++,w=w*wn%mod)
      for (int j=i;j<n;j+=(k<<1))
      {
        LL x=a[j], y=w*a[j|k]%mod;
        a[j]=(x+y)%mod;
        a[j|k]=(x-y+mod)%mod;
      }
  }
  if (flag==-1)
  {
    LL inv=exp(n, mod-2, mod);
    for (int i=0;i<n;i++) a[i]=a[i]*inv%mod;
  }
}

inline void roll(LL *a, LL *b, LL *c, int n, int m)
{
  int num=1;
  while (num<n+m) num<<=1;
  for (int i=0;i<num;i++) a[i]=(i<n)?a[i]:0;
  for (int i=0;i<num;i++) b[i]=(i<m)?b[i]:0;
  ntt(a, num, 1);
  ntt(b, num, 1);
  for (int i=0;i<num;i++) c[i]=a[i]*b[i]%mod;
  ntt(c, num, -1);
}
merge
inline void solve(int l, int k)
{
  if (k==0)
  {
    f[l]=f[l]*exp(l, mod-2, mod)%mod;
    return ;
  }
  int r=l+(1<<k)-1, mid=(l+r)>>1;
  solve(l, k-1);
  memset(a, 0, sizeof(*a)<<k);
  //cout<<k<<endl;
  for (int i=l;i<=mid;i++) a[i-l]=f[i];
  ntt(a, 1<<k, 1);
  for (int i=0;i<1<<k;i++) (a[i]*=b[k][i])%=mod;
  ntt(a, 1<<k, -1);
  for (int i=mid+1;i<=r;i++)
    (f[i]+=a[i-l-1])%=mod;
  solve(mid+1, k-1);
}
//求原根(MOD=P * 2^k + 1)
int g = 2;
while (exp(g, (MOD - 1) / 2, MOD) == 1 || exp(g, (MOD - 1) / 479, MOD) == 1) {
    g ++;
}
```

1.17 QuadraticField

```
//二次域运算
typedef long long LL;
const LL w=5;
const LL oo=1e9+7;
struct PP
{
  LL x, y;
  PP(){}
  PP(LL x, LL y):x(x%oo), y(y%oo){}
};
PP operator +(PP a, PP b){
  return PP((a.x+b.x)%oo, (a.y+b.y)%oo);
}
PP operator -(PP a, PP b){
  return PP((a.x-b.x+oo)%oo, (a.y-b.y+oo)%oo);
}
PP operator *(PP a, PP b){
  return PP((a.x*b.x%oo+a.y*b.y%oo*w%oo)%oo, (a.x*b.y%oo+a.y*b.x%oo)%oo);
}
void write(PP a){
  cout<<a.x<<' '<<a.y<<endl;
}
//快速幂
PP exp(PP a, LL b, LL p){
  PP tmp=PP(1, 0);
  while (b){
    if (b&1) tmp=tmp*a;
    a=a*a;
    b=b>>1;
  }
  return tmp;
}
//二次域逆元
PP inv(PP a)
{
  PP r=exp(PP((a.x*a.x%oo-a.y*a.y%oo*w%oo+oo)%oo, 0), oo-2, oo);
  return r*PP(a.x, oo-a.y);
}
```

1.18 QuickPower

```
//快速幂
LL exp(LL a, LL b, LL p)
{
  a%=p;
  LL tmp=1;
  while (b)
  {
    if (b&1) tmp=(tmp*a)%p;
    a=a*a%p;
    b=b/2;
  }
  return tmp;
```

```
}
```

1.19 SlowMul

```
//慢速乘
llg mul(llg a,llg b,llg p)
{
  a%=p,b%=p;
  llg tmp=0;
  while (b)
  {
    if (b&1) tmp=(tmp+a)%p;
    a=(a+a)%p;
    b/=2;
  }
  return tmp;
}
```

1.20 Trisection

```
//三分求最小值
double l=mi,r=mx,ans;
while (dcmp(l-r)<=0)
{
  double lmid=l+(r-l)/3.0;
  double rmid=r-(r-l)/3.0;
  if (dcmp(get_ans(lmid)-get_ans(rmid))<=0)
  {
    r=rmid-eps;
    ans=get_ans(lmid);
  }
  else
  {
    l=lmid+eps;
  }
}
```

1.21 XorGauss

```
//高斯消元XOR
int gauss(int n, int m)
{
    int k=1;
    for (int i=1;i<=m;i++)
    {
        int p=0;
        for (int j=k;j<=n;j++) if (a[j][i]) {    p=j;
        break;    }
        if (!p) continue;
        for (int l=1;l<=m;l++) swap(a[p][l],a[k][l]);//有常数的话m+1
        for (int j=k+1;j<=n;j++)
          if (a[j][i]==1)
             for (int l=1;l<=m;l++)//有常数的话m+1
                  a[j][l]=a[j][l]^a[k][l];
        k++;
    }
    return k-1;//返回线性无关方程个数
}
```

2 DP

2.1 DigitDP

```
//数位动归
int get_ans(int x)
{
  if (!x) return 1;
  int len=0,tmp=0;
  while (x)
  {
    b[++len]=x%10;
    x=x/10;
  }
  b[1]++;
  b[len+1]=0;
  b[len+2]=0;
  for (int i=len;i>=1;i--)//固定第i+1位之前的位
  {
    if (b[i+1]==4) break;
    if (b[i+2]==6&&b[i+1]==2) break;
    for (int j=0;j<b[i];j++)//枚举第i位
    {
      if ((b[i+1]==6&&j==2)||(j==4)) continue ;
      tmp+=f[i-1][0]+f[i-1][1]*(j!=6);
    }
  }
  return tmp;
}
```

2.2 QuadrilateralInequality

```
//四边形不等式
for (int len=2;len<=n;len++)
{
  for (int i=1;i<=n-len+1;i++)
  {
    int j=i+len-1;
    f[i][j]=inf;
    for (int k=K[i][j-1];k<=min(j-1,K[i+1][j]);k++)
    {
      if (f[i][j]>f[i][k]+f[k+1][j]+x[k+1]-x[i]+y[k]-y[j])
      {
        f[i][j]=f[i][k]+f[k+1][j]+x[k+1]-x[i]+y[k]-y[j];
        K[i][j]=k;
      }
    }
  }
}
```

2.3 StateCompressionDP

```
//状压DP
#define inf 1000000007
#define maxn 11
#define maxm 1050

using namespace std;

int n,m,mb;
long long f[maxn][maxn][88][maxm];
int bit[maxm][maxn];

bool legal(int x, int y,int mask)
{
    if (y==1)
    {
        if (bit[mask][n]&bit[mask][n+1]) return 0;
        if (bit[mask][n-1]&bit[mask][n+1]) return 0;
    }
    else
    {
        if (bit[mask][y-1]&bit[mask][n+1]) return 0;
        if (bit[mask][y-2]&bit[mask][n+1]) return 0;
    }
    for (int i=1;i<n;i++) if (bit[mask][i]&bit[mask][i+1])
     return 0;
    return 1;
}

int main()
{
    scanf("%d%d",&n,&m);
    mb=(1<<(n+1))-1;
    memset(bit,0,sizeof(bit));
    for (int mask=0;mask<=mb;mask++)
    {
        int tmp=mask;
        for (int i=1;i<=n+1;i++)
        {
            bit[mask][i]=tmp%2;
            tmp/=2;
        }
    }
    memset(f,0,sizeof(f));
    f[0][n][0][0]=1;
    for (int i=1;i<=n;i++)
        for (int j=1;j<=n;j++)
            for (int k=0;k<=m;k++)
                for (int mask=0;mask<=mb;mask++)
                    if (legal(i,j,mask))
                    {
```

```
                        int tmp=1<<(j-1);
                        int ost=mask&(~tmp),nst=mask|tmp;
                        if (((mask>>(j-1))&1)!=((mask>>n)&1)) ost^=(1<<n),nst^=(1<<n);
                        if (j==1)
                        {
                            f[i][j][k][ost]+=f[i-1][n][k][mask];
                            if (k && !bit[mask][j] && !bit[mask][j+1] && !bit[mask][j-1])
                                f[i][j][k][nst]+=f[i-1][n][k-1][mask];
                        }
                        else
                        {
                            f[i][j][k][ost]+=f[i][j-1][k][mask];
                            if (k && !bit[mask][j] && !bit[mask][j+1] && !bit[mask][j-1] && !bit[mask][n+1])
                                f[i][j][k][nst]+=f[i][j-1][k-1][mask];
                        }
                    }
    long long ans=0;
    for (int mask=0;mask<=mb;mask++)
        if (legal(n+1,1,mask))
        {
            ans+=f[n][n][m][mask];
        }
    printf("%lld\n",ans);
    return 0;
}

Kings
```

2.4 Tree_Divide_Point

```
//Poj1471 Tree
//点分治
//每次分治先选择中心点为根，然后删去根
//分解为若干子树，递归处理。
struct edge
{
  int s,t,val;
  edge(){}
  edge(int s, int t, int val):s(s),t(t),val(val){}
};

vector<edge> e[maxn];
vector<int> q;
int vis[maxn], size[maxn], mx_size[maxn], dep[maxn];
int n, m, rot, ans;

void chose(int x, int fa, int num)//递归寻找重心
{
  size[x]=1; mx_size[x]=0;
  int tot=e[x].size();
  for (int i=0;i<tot;i++)
  {
    int y=e[x][i].t;
    if (!vis[y] && y!=fa)
    {
      chose(y, x, num);
      size[x]+=size[y];
      mx_size[x]=max(mx_size[x], size[y]);
    }
  }
  mx_size[x]=max(mx_size[x], num-size[x]);
  if (mx_size[x]<mx_size[rot]) rot=x;
}

int chose_rot(int x, int num)
{
  rot=0;
  chose(x, 0, num);

  return rot;
}

void get_deep(int x, int fa)//递归求每个点的深度
{
  q.push_back(dep[x]);
  int tot=e[x].size();
  for (int i=0;i<tot;i++)
  {
    int y=e[x][i].t;
    if (!vis[y] && y!=fa)
    {
      dep[y]=dep[x]+e[x][i].val;
      get_deep(y, x);
    }
  }
}

int get_ans(int x, int h)//计算当前结点为根的答案
{
  q.clear();
  dep[x]=h;
  get_deep(x, 0);
  sort(q.begin(), q.end());
  int l=0, r=q.size()-1, tmp=0;
  while (l<r)
  {
    if (q[l]+q[r]<=m) tmp+=(r-l), l++;
    else r--;
  }
  return tmp;
}

void solve(int x)//分治处理x为根的子树
{
  vis[x]=1;
  ans+=get_ans(x, 0);
  int tot=e[x].size();
  for (int i=0;i<tot;i++)
  {
    int y=e[x][i].t;
    if (!vis[y])
    {
      ans-=get_ans(y, dep[y]);
      solve(chose_rot(y, size[y]));
    }
  }
}

void init()
{
  for (int i=1;i<=n;i++) e[i].clear();
  memset(vis,0,sizeof(vis));
  ans=0;
  mx_size[0]=inf;
}

int main()
{
  while (scanf("%d%d",&n,&m) && (n+m)!=0)
  {
    int x,y,z;
    init();
    for (int i=1;i<n;i++)
    {
      scanf("%d%d%d",&x,&y,&z);
      e[x].push_back(edge(x,y,z));
      e[y].push_back(edge(y,x,z));
    }
    solve(chose_rot(1, n));
    printf("%d\n",ans);
  }
  return 0;
}
```

3 String

3.1 AC_Automaton

```
//AC自动机
int ch[maxn][27], v[maxn];
int fail[maxn];
int rot, num, now, n;
char s[maxn];

void ins(int w)
{
```

```
    if (ch[now][w]==-1) ch[now][w]=++num;
    now=ch[now][w];
}

void build()
{
    queue<int> q;
    memset(fail,0,sizeof(fail));
    for (int w=0;w<26;w++)
        if (ch[rot][w]==-1)
            ch[0][w]=rot;
        else
            q.push(ch[0][w]);
    while (!q.empty())
    {
        int x=q.front(); q.pop();
        for (int w=0;w<26;w++)
            if (ch[x][w]==-1)
                ch[x][w]=ch[fail[x]][w];
            else
            {
                int y=ch[x][w];
                fail[y]=ch[fail[x]][w];
                q.push(y);
            }
    }
}
```

3.2 KMP

```
//KMP
int kmp(char *t)
{
  int n=strlen(t);
  memset(nxt,-1,sizeof(nxt));
  int j=-1;
  for (int i=1;i<n;i++)
  {
    while (j!=-1&&t[i]!=t[j+1]) j=nxt[j];
    if (t[i]==t[j+1]) j++;
    nxt[i]=j;
  }
  return n-1-nxt[n-1];//n-next[n-1]是最小循环节
}
void match(char *s, char *t)
{
  int ls=strlen(s), lt=strlen(t);
  int j=-1;
  for (int i=0;i<ls;i++)
  {
    while (j!=-1 && s[i]!=t[j+1]) j=nxt[j];
    if (s[i]==t[j+1]) j++;
  }
}
```

3.3 StringHash

```
//字符串哈希
unsigned int BKDRHash(char *str)
{
    unsigned int seed=131; // 31 131 1313 13131 131313 etc..
    unsigned int hash=0;
    while (*str) hash=hash*seed+(*str++);
    return (hash&0x7FFFFFFF);
}
```

3.4 Suffix_Automaton

```
//后缀自动机
struct node
{
    node *pre,*ch[27];
    int mx;
    node(){
      mx=0;
      pre=NULL;
      for (int i=0;i<=26;i++) ch[i]=0;
    }
}*rot,*now,*que[maxn];

inline void ins(int w)
{
  node *p=now,*np=&rot[++num];
  np->mx=p->mx+1;
  while (p!=NULL && p->ch[w]==NULL) p->ch[w]=np,p=p->pre;
  if (p==NULL) np->pre=rot;
  else
  {
    node *q=p->ch[w];
    if (q->mx==p->mx+1) np->pre=q;
    else
    {
      node *nq=&rot[++num];
      *nq=*q;
      nq->mx=p->mx+1;
      np->pre=q->pre=nq;
      while (p!=NULL && p->ch[w]==q) p->ch[w]=nq,p=p->pre;
    }
  }
  now=np;
}

inline void build(char *s)
{
  rot=new node[maxn];
  now=&rot[num=0];
  int n=strlen(s);
  for (int i=0;i<n;i++) ins(s[i]-'a');
}

void clear()
{
  delete [] rot;
}

inline void tong_sort()
{
  for (int i=1;i<n;i++) tong[i]=0;
  for (int i=0;i<=num;i++) tong[rot[i].mx]++;
  for (int i=1;i<=n;i++) tong[i]+=tong[i-1];
  for (int i=num;i>=0;i--) que[--tong[rot[i].mx]]=&rot[i];
}
```

4 DataStruct

4.1 BIT_2Dimensions

```
struct bit
{
    int b[maxn][maxn];
    void add(int x,int y,int z)
    {
        for (int i=x;i<=n;i+=(i&-i))
            for (int j=y;j<=m;j+=(j&-j))
                b[i][j]+=z;
    }
    int ask(int x,int y)
    {
        int tmp=0;
        for (int i=x;i>0;i-=(i&-i))
            for (int j=y;j>0;j-=(j&-j))
                tmp+=b[i][j];
        return tmp;
    }
}s,t,l,r;

void ins(int x1,int y1,int x2,int y2,int z)
{
    s.add(x1,y1,z); s.add(x2+1,y1,-z);
    s.add(x1,y2+1,-z); s.add(x2+1,y2+1,z);

    l.add(x1,y1,z*x1); l.add(x2+1,y1,-z*(x2+1));
    l.add(x1,y2+1,-z*x1); l.add(x2+1,y2+1,z*(x2+1));

    r.add(x1,y1,z*y1); r.add(x2+1,y1,-z*y1);
    r.add(x1,y2+1,-z*(y2+1)); r.add(x2+1,y2+1,z*(y2+1));

    t.add(x1,y1,z*x1*y1); t.add(x2+1,y1,-z*(x2+1)*y1);
    t.add(x1,y2+1,-z*x1*(y2+1)); t.add(x2+1,y2+1,z*(x2+1)*(y2+1));
}

int query(int x1,int y1,int x2,int y2)
```

```
{
    int a1=s.ask(x2,y2)*(x2+1)*(y2+1)-s.ask(x1-1,y2)*x1*(y2+1)
          -s.ask(x2,y1-1)*(x2+1)*y1+s.ask(x1-1,y1-1)*x1*y1;

    int a2=l.ask(x2,y2)*(y2+1)-l.ask(x1-1,y2)*(y2+1)
          -l.ask(x2,y1-1)*y1+l.ask(x1-1,y1-1)*y1;

    int a3=r.ask(x2,y2)*(x2+1)-r.ask(x1-1,y2)*x1
          -r.ask(x2,y1-1)*(x2+1)+r.ask(x1-1,y1-1)*x1;

    int a4=t.ask(x2,y2)-t.ask(x1-1,y2)
          -t.ask(x2,y1-1)+t.ask(x1-1,y1-1);
    return a1-a2-a3+a4;
}
```

4.2 BIT_Segment_Add

```
//BIT改段求段
struct bit
{
  int b[maxn];
  int num;
  void add(int x,int z)
  {
    for (int i=x;i<=num;i+=(i&-i)) b[i]+=z;
  }
  int ask(int x)
  {
    int tmp=0;
    for (int i=x;i;i-=(i&-i)) tmp+=b[i];
    return tmp;
  }
  void init(int n)
  {
    num=n;
    memset(b,0,sizeof(b));
  }
}s,t;

void ins(int x,int y,int z)
{
  s.add(x,z); s.add(y+1,-z);
  t.add(x,z*x); t.add(y+1,-z*(y+1));
}

int query(int x,int y)
{
  int tmp=s.ask(y)*(y+1)-s.ask(x-1)*x
       -(t.ask(y)-t.ask(x-1));
  return tmp;
}
```

4.3 BIT_Segment_Max

```
//BIT求最值
struct mx_bit
{
  int mx[maxn],key[maxn];
  int num;
  void cov(int x,int z)
  {
    key[x]=z;
    for (int i=x;i<=num;i+=(i&-i))
      mx[i]=max(mx[i],z);
  }
  int ask(int l,int r)
  {
    int tmp=-inf;
    while (l<=r)
    {
      tmp=max(tmp,key[r]);
      for (r-=1;r-(r&-r)>=l;r-=(r&-r))
        tmp=max(tmp,mx[r]);
    }
    return tmp;
  }
  void init(int x)
  {
    num=x;
    memset(mx,0,sizeof(mx));
    memset(key,0,sizeof(key));
  }
}tree;
```

4.4 BIT_single

```
struct bit
{
  int sum[maxn];
  int num;
  bit(){
    memset(sum,0,sizeof(sum));}
  bit(int n):num(n){
    memset(sum,0,sizeof(sum));}
  void add(int x,int z)
  {
    for (int i=x;i<=num;i+=(i&-i))
      sum[i]+=z;
  }
  int ask(int x)
  {
    int tmp=0;
    for (int i=x;i>=1;i-=(i&-i))
      tmp+=sum[i];
    return tmp;
  }
  void clear()
  {
    memset(sum,0,sizeof(sum));
  }
};
```

4.5 BitSet

```
struct score
{
    int x,y;
    friend bool operator <(score a, score b)
    {
        return a.x<b.x;
    };
}s[6][maxn];

int n,m,p,num,block;
int bel[maxn],l[maxm],r[maxm];
int a[6][maxn];
bitset<50005> b[6][maxm];
bitset<50005> now[6];

int main()
{
    int Case;
    scanf("%d",&Case);
    for (int i=1;i<=5;i++) b[i][0].reset();
    for (int o=1;o<=Case;o++)
    {
        scanf("%d%d",&n,&m);
        for (int j=1;j<=n;j++)
            for (int i=1;i<=5;i++)
            {
                scanf("%d",&a[i][j]);
                s[i][j].x=a[i][j];
                s[i][j].y=j;
            }
        for (int i=1;i<=5;i++)
        {
            sort(s[i]+1,s[i]+n+1);
            sort(a[i]+1,a[i]+n+1);
        }
        int block=sqrt(n);
        int num=n/block+(n%block!=0);
        for (int i=1;i<=n;i++) bel[i]=(i-1)/block+1;
        for (int i=1;i<=num;i++) l[i]=(i-1)*block+1;
        for (int i=1;i<=num;i++) r[i]=i*block;
        r[num]=n;
        for (int i=1;i<=5;i++)
            for (int j=1;j<=num;j++)
            {
                b[i][j]=b[i][j-1];
                for (int k=l[j];k<=r[j];k++)
                    b[i][j][s[i][k].y]=1;
            }
```

```
        scanf("%d",&p);
        int x[6],ans=0;
        for (int q=1;q<=p;q++)
        {
            for (int i=1;i<=5;i++) scanf("%d",&x[i]);
            for (int i=1;i<=5;i++) x[i]^=ans;
            for (int i=1;i<=5;i++) now[i].reset();
            for (int i=1;i<=5;i++)
            {
                int tmp=upper_bound(a[i]+1,a[i]+n+1,x[i])-a[i]-1;
                if (tmp==0) continue;
                now[i]=b[i][bel[tmp]-1];
                for (int j=l[bel[tmp]];j<=tmp;j++)
                    now[i][s[i][j].y]=1;
            }
            now[0]=now[1]&now[2]&now[3]&now[4]&now[5];
            ans=now[0].count();
            printf("%d\n",ans);
        }
    }
    return 0;
}
```

4.6 CDQ_Divide

```
//CDQ分治(带修改区间k大)
int n,m,tot,cnt;
int a[maxn], ans[maxm];

struct query
{
  int id,x,y,k,cur;
  query(){}
  query(int id,int x,int y,int k,int cur):
    id(id),x(x),y(y),k(k),cur(cur){}
}q[maxm],q1[maxm],q2[maxm];

struct bit
{
  int num;
  int s[maxn];
  void add(int x,int z)
  {
    for (int i=x;i<=num;i+=(i&-i)) s[i]+=z;
  }
  int ask(int x)
  {
    int tmp=0;
    for (int i=x;i>=1;i-=(i&-i)) tmp+=s[i];
    return tmp;
  }
  void init(int n)
  {
    num=n;
    memset(s,0,sizeof(s));
  }
}tree;

void solve(int st,int ed,int l,int r)
{
  if (st>ed) return ;
  if (l==r)
  {
    for (int i=st;i<=ed;i++) ans[q[i].id]=l;
    return ;
  }
  int mid=(l+r)>>1, n1=0, n2=0;;
  for (int i=st;i<=ed;i++)
  {
    if (q[i].id==0)
    {
      if (q[i].y<=mid)
      {
        tree.add(q[i].x, q[i].k);
        q1[n1++]=q[i];
      }
      else
        q2[n2++]=q[i];
    }
    else
    {
      int tmp=tree.ask(q[i].y)-tree.ask(q[i].x-1);
      if (q[i].cur+tmp>=q[i].k)
        q1[n1++]=q[i];
      else
      {
        q[i].cur+=tmp;
        q2[n2++]=q[i];
      }
    }
  }
  for (int i=st;i<=ed;i++)
    if (q[i].id==0 && q[i].y<=mid) tree.add(q[i].x, -q[i].k);
  for (int i=0;i<n1;i++) q[st+i]=q1[i];
  for (int i=0;i<n2;i++) q[st+n1+i]=q2[i];
  solve(st,st+n1-1,l,mid);
  solve(st+n1,ed,mid+1,r);
}

int main()
{
  int Case;
  scanf("%d",&Case);
  while (Case--)
  {
    scanf("%d%d",&n,&m);
    tot=cnt=0;
    for (int i=1;i<=n;i++)
    {
      scanf("%d",&a[i]);
      q[tot++]=query(0,i,a[i],1,0);
    }
    int x,y,z;
    char sign;
    for (int i=1;i<=m;i++)
    {
      scanf(" %c",&sign);
      if (sign=='C')
      {
        scanf("%d%d",&x,&y);
        q[tot++]=query(0,x,a[x],-1,0);
        a[x]=y;
        q[tot++]=query(0,x,a[x],1,0);
      }
      else
      {
        scanf("%d%d%d",&x,&y,&z);
        q[tot++]=query(++cnt,x,y,z,0);
      }

    }
    tree.init(n);
    solve(0,tot-1,0,inf);
    for (int i=1;i<=cnt;i++) printf("%d\n",ans[i]);
  }
  return 0;
}
```

4.7 ChainPartitionTree

```
//树链剖分
vector<int> t[maxn];
int dep[maxn], pos[maxn], top[maxn], size[maxn], fa[maxn];
int sum[6*maxn], mx[6*maxn];
int n,m,num;

void dfs_one(int x)
{
  size[x]=1;
  int ss=t[x].size();
  for (int j=0;j<ss;j++)
  {
    int y=t[x][j];
    if (y==fa[x]) continue;
    dep[y]=dep[x]+1;
    fa[y]=x;
    dfs_one(y);
    size[x]+=size[y];
  }
}
void dfs_two(int x, int anc)
{
  pos[x]=++num;
  top[x]=anc;
```

```
  int son=0, ss=t[x].size();
  for (int j=0;j<ss;j++)
  {
    int y=t[x][j];
    if (y!=fa[x] && size[y]>size[son]) son=y;
  }
  if (son) dfs_two(son, anc);
  for (int j=0;j<ss;j++)
  {
    int y=t[x][j];
    if (y!=fa[x] && y!=son) dfs_two(y, y);
  }
}
void update(int x)
{
  sum[x]=sum[x<<1]+sum[x<<1|1];
  mx[x]=max(mx[x<<1], mx[x<<1|1]);
}
void build(int x, int l, int r)
{
  if (l==r)
  {
    sum[x]=0;
    mx[x]=-inf;
    return ;
  }
  int mid=(l+r)>>1;
  build(x<<1, l, mid);
  build(x<<1|1, mid+1, r);
}
void change(int x, int l, int r, int p, int z)
{
  if (l==r)
  {
    sum[x]=mx[x]=z;
    return ;
  }
  int mid=(l+r)>>1;
  if (p<=mid) change(x<<1, l, mid, p, z);
  else change(x<<1|1, mid+1, r, p, z);
  update(x);
}
int ask_sum(int x, int l, int r, int ll, int rr)
{
  if (ll<=l && r<=rr) return sum[x];
  int mid=(l+r)>>1, tmp=0;
  if (ll<=mid) tmp+=ask_sum(x<<1, l, mid, ll, rr);
  if (rr>mid) tmp+=ask_sum(x<<1|1, mid+1, r, ll, rr);
  return tmp;
}
int ask_max(int x, int l, int r, int ll, int rr)
{
  if (ll<=l && r<=rr) return mx[x];
  int mid=(l+r)>>1, tmp=-inf;
  if (ll<=mid) tmp=max(tmp, ask_max(x<<1, l, mid, ll, rr))
  ;
  if (rr>mid) tmp=max(tmp, ask_max(x<<1|1, mid+1, r, ll,
  rr));
  return tmp;
}
int get_sum(int x, int y)
{
  int tmp=0;
  while (top[x]!=top[y])
  {
    if (dep[top[x]]<dep[top[y]]) swap(x, y);
    tmp+=ask_sum(1, 1, n, pos[top[x]], pos[x]);
    x=fa[top[x]];
  }
  if (pos[x]>pos[y]) swap(x, y);
  tmp+=ask_sum(1, 1, n, pos[x], pos[y]);
  return tmp;
}
int get_max(int x, int y)
{
  int tmp=-inf;
  while (top[x]!=top[y])
  {
    if (dep[top[x]]<dep[top[y]]) swap(x, y);
    tmp=max(tmp, ask_max(1, 1, n, pos[top[x]], pos[x]));
    x=fa[top[x]];
  }
  if (pos[x]>pos[y]) swap(x, y);
  tmp=max(tmp, ask_max(1, 1, n, pos[x], pos[y]));
  return tmp;
}
int change_value(int x, int z)
{
  change(1, 1, n, pos[x], z);
}

int main()
{
  scanf("%d",&n);
  int x, y;
  for (int i=1;i<=n;i++) t[i].clear();
  for (int i=1;i<n;i++)
  {
    scanf("%d%d",&x,&y);
    t[x].push_back(y);
    t[y].push_back(x);
  }
  num=0;
  dfs_one(1);
  dfs_two(1, 1);
  build(1,1,n);
  for (int i=1;i<=n;i++)
  {
    scanf("%d",&x);
    change_value(i, x);
  }
  char sign[10];
  int z;
  scanf("%d",&m);
  for (int i=1;i<=m;i++)
  {
    scanf(" %s%d%d",sign,&x,&y);
    if (sign[0]=='C')
      change_value(x, y);
    else if (sign[1]=='M')
      printf("%d\n",get_max(x, y));
    else
      printf("%d\n",get_sum(x, y));
  }
  return 0;
}
```

4.8 ChairManTree

```
//主席树
struct node
{
  node *lc,*rc;
  int s;
  node () {s=0;}
}*rot[maxn],*sum[maxn],*p;
vector <node*> st[4];
struct et
{
  int s,t,next;
}e[maxn*2];
int ll[maxn],rr[maxn],fir[maxn],key[maxn];
int w[maxn*2],dep[maxn],fa[maxn][20];
int x[maxn],y[maxn],z[maxn];
int n,m,tot,low,cnt,num;
int v[5];

node *build(int l,int r)
{
  node *now=new node();
  if (l==r) return now;
  int mid=(l+r)>>1;
  now->lc=build(l,mid);
  now->rc=build(mid+1,r);
  return now;
}

node *change(int l,int r,int x,int z)
{
  node *now=new node();
  now->lc=p->lc; now->rc=p->rc;
  now->s=p->s+z;
  if (l==r) return now;
  int mid=(l+r)>>1;
  if (x<=mid) p=p->lc,now->lc=change(l,mid,x,z);
  else p=p->rc,now->rc=change(mid+1,r,x,z);
```

```
  return now;
}

void adj(int x,int y,int z)
{
  if (x==0) return ;
  for (int i=x;i<=n;i+=(i&-i)) p=rot[i],rot[i]=change(0,num,y,z);
}

void fill(int k)
{
  st[k].clear();
  st[k].push_back(sum[v[k]]);
  for (int i=ll[v[k]];i>0;i-=(i&-i)) st[k].push_back(rot[i]);
}

int getsum(int k)
{
  int ans=0, n=st[k].size();
  for (int i=0;i<n;i++) ans+=st[k][i]->s;
  return ans;
}

int getsize(int k)
{
  int ans=0, n=st[k].size();
  for (int i=0;i<n;i++) ans+=st[k][i]->rc->s;
  return ans;
}

void goleft(int k)
{
  int n=st[k].size();
  for (int i=0;i<n;i++) st[k][i]=st[k][i]->lc;
}

void goright(int k)
{
  int n=st[k].size();
  for (int i=0;i<n;i++) st[k][i]=st[k][i]->rc;
}

int query(int l,int r,int k)
{
  if (k>getsum(1)+getsum(2)-getsum(3)-getsum(4)) return 0;
  if (l==r) return l;
  int tmp=getsize(1)+getsize(2)-getsize(3)-getsize(4);
  //cout<<l<<' '<<r<<"tmp:"<<tmp<<endl;
  int mid=(l+r)>>1;
  if (k<=tmp)
  {
    //cout<<"right\n";
    goright(1),goright(2),goright(3),goright(4);
    return query(mid+1,r,k);
  }
  else
  {
    //cout<<"left\n";
    goleft(1),goleft(2),goleft(3),goleft(4);
    return query(l,mid,k-tmp);
  }
}

int lca(int x,int y)
{
  if (dep[x]>dep[y]) swap(x,y);
  for (int i=low;i>=0;i--)
    if (dep[fa[y][i]]>=dep[x]) y=fa[y][i];
  if (x==y) return x;
  for (int i=low;i>=0;i--)
    if (fa[x][i]!=fa[y][i]) x=fa[x][i],y=fa[y][i];
  return fa[x][0];
}

void write(node *now,int l,int r)
{
  if (l==r)
  {
    cout<<now->s;
    return ;
  }
  int mid=(l+r)>>1;
  write(now->lc,l,mid);
  write(now->rc,mid+1,r);
}

void dfs(int now)
{
  ll[now]=rr[now]=++cnt;
  p=sum[fa[now][0]];
  sum[now]=change(0,num,key[now],1);

  for (int j=fir[now];j;j=e[j].next)
  {
    int k=e[j].t;
    if (fa[now][0]!=k)
    {
      fa[k][0]=now; dep[k]=dep[now]+1;
      for (int i=1;i<=low;i++) fa[k][i]=fa[fa[k][i-1]][i-1];
      dfs(k);
      rr[now]=rr[k];
    }
  }
}

void add(int x,int y)
{
  e[++tot].s=x; e[tot].t=y; e[tot].next=fir[x]; fir[x]=tot;
  e[++tot].s=y; e[tot].t=x; e[tot].next=fir[y]; fir[y]=tot;
}

int main()
{
  //freopen("network10.in","r",stdin);
  //freopen("network.out","w",stdout);
  scanf("%d%d",&n,&m);
  low=log(n)/log(2.0);
  for (int i=1;i<=n;i++) scanf("%d",&key[i]),w[++num]=key[i];
  int st,ed;
  for (int i=1;i<n;i++) scanf("%d%d",&st,&ed),add(st,ed);
  num=n;
  for (int i=1;i<=m;i++)
  {
    scanf("%d%d%d",&z[i],&x[i],&y[i]);
    if (!z[i]) w[++num]=y[i];
  }
  sort(w+1,w+num+1);
  num=unique(w+1,w+num+1)-w;
  for (int i=1;i<=n;i++) key[i]=lower_bound(w+1, w+num+1, key[i])-w;
  for (int i=1;i<=m;i++) if (!z[i]) y[i]=lower_bound(w+1, w+num+1, y[i])-w;

  sum[0]=rot[0]=build(0,num);
  for (int i=1;i<=n;i++) rot[i]=rot[0];
  int root=rand()%n+1; dep[root]=1;
  dfs(root);

  for (int i=1;i<=m;i++)
  {
    if (z[i])
    {
      int tmp;
      v[1]=x[i],v[2]=y[i],v[3]=tmp=lca(x[i],y[i]),v[4]=fa[tmp][0];
      fill(1); fill(2); fill(3); fill(4);
      int ans=query(0,num,z[i]);
      if (ans)
        printf("%d\n",w[ans]);
      else
        printf("invalid request!\n");
    }
    else
    {
      adj(ll[x[i]],key[x[i]],-1),adj(rr[x[i]]+1,key[x[i]],1);
      key[x[i]]=y[i];
      adj(ll[x[i]],key[x[i]],1),adj(rr[x[i]]+1,key[x[i]],-1);
    }
```

```
  }

  return 0;
}
```

4.9 DSJ

```
//并查集
int f[maxn];
int find(int x)
{
  return f[x]==-1?x:f[x]=find(f[x]);
}

bool merge(int x, int y)
{
  int fx=find(x), fy=find(y);
  if (fx!=fy)
  {
    f[fx]=fy;
    return 1;
  }
  return 0;
}
```

4.10 KD_Tree

```
//KD-Tree
//维护K维空间上离某点最近的点序
//HDU4347
#include <bits/stdc++.h>
#define maxn 52000
#define square(x) ((x) * (x))
using namespace std;

int idx;
struct Node
{
  int x[12];
  Node *lc, *rc;
  node(){}
  bool operator < (const Node &u) const
  {
    return x[idx]<u.x[idx];
  }
}data[maxn];
typedef pair<double, Node*> pdn;
struct kdt
{
  Node *rot;
  int k;
  priority_queue<pdn>que;
  kdt(){}
  kdt(int k):k(k){rot=NULL;}
  Node* build(int l, int r, int dep)
  {
    if (l>r) return NULL;
    Node *now=new Node();
    idx=dep%k;
    int mid=(l+r)>>1;
    nth_element(data+l, data+mid, data+r+1);
    memcpy(now->x, data[mid].x, sizeof(now->x));
    now->lc=build(l, mid-1, dep+1);
    now->rc=build(mid+1, r, dep+1);
    return now;
  }
  void build(int n, int nk)
  {
    k=nk;
    rot=build(0, n-1, 0);
  }
  void query(Node* now, Node *p, int m, int dep)
  {
    if (now==NULL) return ;
    double sum=0;
    for (int i=0;i<k;i++)
      sum+=square(now->x[i]-p->x[i]);
    int dim=dep%k;
    Node *ll=now->lc, *rr=now->rc;
    if (p->x[dim]>=now->x[dim]) swap(ll, rr);
    query(ll, p, m, dep+1);
    if (que.size()<m)
    {
      que.push(pdn(sum, now));
      query(rr, p, m, dep+1);
    }
    else
    {
      if (sum<que.top().first)
      {
        que.pop();
        que.push(pdn(sum, now));
      }
      if (square(p->x[dim]-now->x[dim])<que.top().first)
        query(rr, p, m, dep+1);
    }
  }
  void query(Node *p, int m)
  {
    while (!que.empty()) que.pop();
    query(rot, p, m, 0);
  }
  void print()
  {
    vector<Node*> t;
    t.clear();
    while (!que.empty())
    {
      t.push_back(que.top().second);
      que.pop();
    }
    int num=t.size();
    for (int i=num-1;i>=0;i--)
      for (int j=0;j<k;j++)
        printf(j==k-1?"%d\n":"%d ",t[i]->x[j]);
  }
}s;
int n,k,m,q;

int main()
{
  while (scanf("%d%d",&n,&k)!=EOF)
  {
    for (int i=0;i<n;i++)
      for (int j=0;j<k;j++)
        scanf("%d",&data[i].x[j]);
    s.build(n, k);
    scanf("%d",&q);
    for (int i=1;i<=q;i++)
    {
      Node *p=new Node();
      for (int j=0;j<k;j++)
        scanf("%d",&p->x[j]);
      scanf("%d",&m);
      s.query(p, m);
      printf("the closest %d points are:\n",m);
      s.print();
      delete p;
    }
  }
  return 0;
}
```

4.11 LCT

```
//LCT
void update(int x)
{
  if (!x) return ;
  sum[x]=(sum[c[x][0]]+sum[c[x][1]]+key[x])%oo;
  siz[x]=(siz[c[x][0]]+siz[c[x][1]]+1)%oo;
}

void change(int x, int mu, int ad)
{
  if (!x) return ;
  key[x]=(key[x]*mu%oo+ad)%oo;
  mul[x]=mul[x]*mu%oo;
  add[x]=(add[x]*mu%oo+ad)%oo;
  sum[x]=(sum[x]*mu%oo+ad*siz[x]%oo)%oo;
}

void reverse(int x)
{
  if (!x) return ;
```

```
  swap(c[x][0], c[x][1]);
  rev[x]^=1;
}

void down(int x)
{
  if (rev[x])
  {
    reverse(c[x][0]);
    reverse(c[x][1]);
    rev[x]=0;
  }
  change(c[x][0], mul[x], add[x]);
  change(c[x][1], mul[x], add[x]);
  mul[x]=1;
  add[x]=0;
}

bool is_root(int x)
{
  return c[fa[x]][0]!=x && c[fa[x]][1]!=x;
}

void relax(int x)
{
  if (!is_root(x)) relax(fa[x]);
  down(x);
}

void zigzag(int x)
{
  int y=fa[x], z=fa[y];
  int p=(c[y][1]==x), q=p^1;
  if (!is_root(y)) c[z][c[z][0]!=y]=x;
  fa[x]=z; fa[y]=x; fa[c[x][q]]=y;
  c[y][p]=c[x][q]; c[x][q]=y;
  update(y);
}

void splay(int x)
{
  relax(x);
  while (!is_root(x))
  {
    int y=fa[x], z=fa[y];
    if (!is_root(y))
      ((c[y][0]==x) ^ (c[z][0]==y))?zigzag(x):zigzag(y);
    zigzag(x);
  }
  update(x);
}

int access(int x)
{
  int y=0;
  for (; x; y=x, x=fa[x])
  {
    splay(x);
    c[x][1]=y;
    update(x);
  }
  return y;
}

int root(int x)
{
  while (fa[x]) x=fa[x];
  return x;
}

void make_root(int x)
{
  access(x);
  splay(x);
  reverse(x);
}

void split(int x, int y)//x is root
{
  make_root(y);
  access(x);
  splay(x);
}

void link(int x, int y)
{
  make_root(x);
  fa[x]=y;
}

void cut(int x, int y)
{
  split(y, x);
  c[y][0]=fa[x]=0;
}

void solve(int x, int y, int z)
{
  access(x);
  int lca=access(y);
  splay(x);
  change(c[lca][1], z);
  key[lca]=z;
  if (x!=lca) change(x, z);
}
```

4.12 LeftistTree

```
//左偏树
int v[maxn], dis[maxn], sz[maxn];
int l[maxn], r[maxn];
int rot[maxn], st[maxn], ed[maxn];
int num;
int build(int z){
  v[++num]=z;
  sz[num]=1;
  l[num]=r[num]=dis[num]=0;
  return num;
}
int merge(int x, int y){
  if (!x || !y) return x+y;
  if (v[x]<v[y]) swap(x, y);//bit rot
  r[x]=merge(r[x], y);
  if (dis[r[x]]>dis[l[x]]) swap(l[x], r[x]);
  sz[x]=sz[l[x]]+sz[r[x]]+1;
  dis[x]=dis[r[x]]+1;
  return x;
}
int top(int x){
  return v[x];
}
int size(int x){
  return sz[x];
}
void pop(int &x){
  x=merge(l[x], r[x]);
}
```

4.13 Merge

```
//分治求逆序对
void solve(int l, int r)
{
  if (l==r) return ;
  int mid=(l+r)>>1;
  solve(l, mid);
  solve(mid+1, r);
  int i=l, j=mid+1, k=0;
  while (i<=mid || j<=r)
  {
    if (j>r || i<=mid && q[i]<=q[j]) nq[k++]=q[i++], sum
    +=(j-mid-1);
    else
    if (i>mid || j<=r && q[i]>q[j]) nq[k++]=q[j++];
  }
  for (int i=0;i<k;i++) q[l+i]=nq[i];
}
```

4.14 SegTree

```
//线段树最大连续1
struct segtree
{
  int mx[maxn*8], lmx[maxn*8], rmx[maxn*8], size[maxn*8];
  void update(int x)
```

```
  {
    mx[x]=max(max(mx[x*2],mx[x*2+1]),rmx[x*2]+lmx[x*2+1]);
    if (lmx[x*2]==size[x*2]) lmx[x]=size[x*2]+lmx[x*2+1];
    else lmx[x]=lmx[x*2];
    if (rmx[x*2+1]==size[x*2+1]) rmx[x]=size[x*2+1]+rmx[x
    *2];
    else rmx[x]=rmx[x*2+1];
  }
  void build(int x,int l,int r)
  {
    if (l==r)
    {
      size[x]=1;
      return ;
    }
    int mid=(l+r)>>1;
    build(x*2,l,mid);
    build(x*2+1,mid+1,r);
    size[x]=size[x*2]+size[x*2+1];
  }
  void add(int x, int l, int r, int pos, int z)
  {
    if (l==r)
    {
      mx[x]=lmx[x]=rmx[x]=z;
      return ;
    }
    int mid=(l+r)>>1;
    if (pos<=mid) add(x*2, l, mid, pos, z);
    else add(x*2+1, mid+1, r, pos, z);
    update(x);
  }
  int ask(int x, int l, int r, int ll, int rr, int &res)
  {
    if (ll<=l && r<=rr)
    {
      int tmp=max(res+lmx[x], mx[x]);
      if (rmx[x]==size[x]) res+=size[x];
      else res=rmx[x];
      return tmp;
    }
    int mid=(l+r)>>1, tmp=0;
    if (ll<=mid) tmp=max(tmp,ask(x*2, l, mid, ll, rr, res)
    );
    if (rr>mid) tmp=max(tmp,ask(x*2+1, mid+1, r, ll, rr,
    res));
    return tmp;
  }
  void init()
  {
    memset(mx,0,sizeof(mx));
    memset(lmx,0,sizeof(lmx));
    memset(rmx,0,sizeof(rmx));
  }
}tree;
```

4.15 SetUnion

```
//set启发式合并
int merge(int x,int y)
{
  int tmp=inf;
  SIT prev,succ;
  if (s[x].size()<s[y].size()) swap(s[x],s[y]);
  for (SIT i=s[y].begin();i!=s[y].end();i++)
  {
    prev=succ=s[x].lower_bound(*i);
    if (prev!=s[x].begin()) prev--;
    if (prev!=s[x].end()) tmp=min(tmp,abs(*prev-*i));
    if (succ!=s[x].end())   tmp=min(tmp,abs(*succ-*i));
    s[x].insert(*i);
  }
  s[y].clear();
  return tmp;
}
```

4.16 Splay_Segment

```
//Splay区间维护
//支持区间求和，最大连续子段，区间反转，区间覆盖操作
int c[maxn][2],fa[maxn];
int a[maxn],key[maxn],sum[maxn],la[maxn],ra[maxn],ma[maxn
],cov[maxn],size[maxn];
int q[maxn];
bool rev[maxn];
int n,m,tot,num,rot,st,ed,tail;

void update(int x)
{
    if (!x) return;
    la[x]=max(la[c[x][0]],sum[c[x][0]]+key[x]+max(0,la[c[x
    ][1]]));
    ra[x]=max(ra[c[x][1]],sum[c[x][1]]+key[x]+max(0,ra[c[x
    ][0]]));
    ma[x]=max(max(ma[c[x][0]],ma[c[x][1]]),key[x]+max(0,ra
    [c[x][0]])+max(0,la[c[x][1]]));
    sum[x]=sum[c[x][0]]+sum[c[x][1]]+key[x];
    size[x]=size[c[x][0]]+size[c[x][1]]+1;
}

void reverse(int x)
{
    if (!x) return;
    swap(c[x][0],c[x][1]);
    swap(la[x],ra[x]);
    rev[x]^=1;
}

void recover(int x,int z)
{
    if (!x) return ;
    key[x]=cov[x]=z;
    sum[x]=size[x]*z;
    la[x]=ra[x]=ma[x]=max(z,sum[x]);
}

void down(int x)
{
    if (!x) return;
    if (rev[x])
    {
        reverse(c[x][0]);
        reverse(c[x][1]);
        rev[x]=0;
    }
    if (cov[x]!=-inf)
    {
        recover(c[x][0],cov[x]);
        recover(c[x][1],cov[x]);
        cov[x]=-inf;
    }
}

void relax(int x,int rot)
{
    if (x!=rot) relax(fa[x],rot);
    down(x);
}

void rotate(int x,int &rot)
{
    int y=fa[x],z=fa[y];
    int p=(c[y][1]==x),q=p^1;
    if (y==rot) rot=x;
    c[z][c[z][1]==y]=x;
    fa[x]=z; fa[y]=x; fa[c[x][q]]=y;
    c[y][p]=c[x][q]; c[x][q]=y;
    update(y);
}

void splay(int x,int &rot)
{
    relax(x,rot);
    while (x!=rot)
    {
        int y=fa[x], z=fa[y];
        if (y!=rot)
        {
            if ((c[y][0]==x)xor(c[z][0]==y)) rotate(x,rot)
            ;
            else rotate(y,rot);
        }
        rotate(x,rot);
    }
    update(x);
}
```

```
int pick()
{
    if (tail) return q[tail--];
    else return ++num;
}

int setup(int x)
{
    int t=pick();
    key[t]=a[x];
    cov[t]=-inf;
    rev[t]=0;
    la[t]=ra[t]=ma[t]=-inf;
    return t;
}

int build(int l,int r)
{
    int mid=(l+r)>>1,left=0,right=0;
    if (l<mid)      left=build(l,mid-1);
    int t=setup(mid);
    if (r>mid)      right=build(mid+1,r);
    if (left)       c[t][0]=left,fa[left]=t;
    if (right)c[t][1]=right,fa[right]=t;
    update(t);
    return t;
}

int find(int t,int k)
{
    down(t);
    if (k==size[c[t][0]]+1) return t;
    if (k<size[c[t][0]]+1) return find(c[t][0],k);
    if (k>size[c[t][0]]+1) return find(c[t][1],k-size[c[t
    ][0]]-1);
}

void del(int &x)
{
    if (!x) return;
    q[++tail]=x;
    fa[x]=0;
    del(c[x][0]);
    del(c[x][1]);
    la[x]=ra[x]=ma[x]=-inf;
    x=0;
}

int main()
{
    //freopen("build.in","r",stdin);
    //freopen("build.out","w",stdout);
    scanf("%d %d",&n,&m);
    for (int i=2;i<=n+1;i++)
            scanf("%d",&a[i]);
    a[st=1]=0; a[ed=n+2]=0;
    ra[0]=la[0]=ma[0]=-inf;
    rot=build(1,n+2);
    char sign[20];
    int x,y,l,r,z,ans;
    for (int i=1;i<=m;i++)
    {
        scanf(" %s",sign);
        if (sign[0]=='I')
        {
            scanf("%d %d",&x,&y);
            //处理[x..y]区间,
            //l=find(rot, x), r=find(rot, y+2)
            l=find(rot,x+1); r=find(rot,x+2);
            splay(r,rot); splay(l,c[rot][0]);
            for (int j=1;j<=y;j++)
                scanf("%d",&a[j]);
            int tmp=build(1,y);
            fa[tmp]=l; c[l][1]=tmp;
            update(l); update(r);
        }
        if (sign[0]=='D')
        {
            scanf("%d %d",&x,&y);
            l=find(rot,x);  r=find(rot,x+y+1);
            splay(r,rot); splay(l,c[rot][0]);
            del(c[l][1]);
            update(l); update(r);
        }
        if (sign[0]=='M'&&sign[2]=='K')
        {
            scanf("%d %d %d",&x,&y,&z);
            l=find(rot,x); r=find(rot,x+y+1);
            splay(r,rot); splay(l,c[rot][0]);
            recover(c[l][1],z);
        }
        if (sign[0]=='R')
        {
            scanf("%d %d",&x,&y);
            l=find(rot,x); r=find(rot,x+y+1);
            splay(r,rot); splay(l,c[rot][0]);
            reverse(c[l][1]);
        }
        if (sign[0]=='G')
        {
            scanf("%d %d",&x,&y);
            l=find(rot,x); r=find(rot,x+y+1);
            splay(r,rot); splay(l,c[rot][0]);
            ans=sum[c[l][1]];
            printf("%d\n",ans);
        }
        if (sign[0]=='M'&&sign[2]=='X')
        {
            splay(ed,rot); splay(st,c[rot][0]);
            ans=ma[c[st][1]];
            printf("%d\n",ans);
        }
    }
    return 0;
}
```

4.17 TriPartialOrder

```
//三维偏序
struct point{
    int x,y,z,s;
}a[maxn],q1[maxn],q2[maxn];
int f[maxn];
int n,m;
struct bit
{
    int b[maxn];
    void clear(int x){
        for (int i=x;i;i-=(i&-i)) b[i]=0;
    }
    void add(int x,int z){
        for (int i=x;i;i-=(i&-i)) b[i]=max(b[i],z);
    }
    int ask(int x){
        int tmp=0;
        for (int i=x;i<=n;i+=(i&-i)) tmp=max(b[i],tmp);
        return tmp;
    }
}s;

bool cmp1(point a,point b)
{    return a.x>b.x;}

bool cmp2(point a,point b)
{    return a.y>b.y;}

void solve(int l,int r){
    if (l==r) return ;
    int mid=(l+r)>>1;
    solve(l,mid);
    int t1=0,t2=0;
    for (int i=l;i<=mid;i++) q1[++t1]=a[i];
    for (int i=mid+1;i<=r;i++) q2[++t2]=a[i];
    sort(q1+1,q1+t1+1,cmp2);
    sort(q2+1,q2+t2+1,cmp2);
    int i=1;
    for (int j=1;j<=t2;j++){
        for (;q1[i].y>q2[j].y&&i<=t1;i++)
            s.add(q1[i].z,f[q1[i].s]);
        f[q2[j].s]=max(f[q2[j].s],s.ask(q2[j].z)+1);
    }
    for (int i=1;i<=t1;i++)
        s.clear(q1[i].z);
    solve(mid+1,r);
}
```

```

int main()
{
    //freopen("3sort.in","r",stdin);
    scanf("%d",&n);
    for (int i=1;i<=n;i++)
        scanf("%d%d%d",&a[i].x,&a[i].y,&a[i].z),a[i].s=i;
    sort(a+1,a+n+1,cmp1);
    for (int i=1;i<=n;i++) f[i]=1;
    solve(1,n);
    int ans=0;
    for (int i=1;i<=n;i++) ans=max(ans,f[i]);
    printf("%d\n",ans);
    return 0;
}
```

5 Graph

5.1 Floyd

```
//floyd最小环
for (int i=1;i<=n;i++)
  for (int j=1;j<=n;j++)
    a[i][j]=d[i][j]=inf;
for (int i=1;i<=m;i++)
{
  scanf("%d%d%d",&x,&y,&z);
  if (z<a[x][y])
  {
    a[x][y]=a[y][x]=z;
    d[x][y]=d[y][x]=z;
  }
}
ans=inf;
for (int k=1;k<=n;k++)
{
  for (int i=1;i<k;i++)
    for (int j=i+1;j<k;j++)
      ans=min(ans,d[i][j]+a[j][k]+a[k][i]);
  for (int i=1;i<=n;i++)
    for (int j=1;j<=n;j++)
      d[i][j]=min(d[i][j],dis[i][k]+dis[k][j]);
}
```

5.2 HeapDij

```
//堆DJ
struct node
{
  int id,dis;
  node(){}
  node(int x,int y)
  {
    id=x; dis=y;
  }
  friend bool operator <(node a,node b)
  {
    return a.dis>b.dis;
  }
};

priority_queue<node> q;

void dij(int st)
{
  for (int i=1;i<=n;i++) dis[i]=inf;
  memset(vis,0,sizeof(vis));
  while (!q.empty()) q.pop();
  q.push(node(st,0));
  dis[st]=0;
  while (!q.empty())
  {
    int i=q.top().id;
    q.pop();
    if (vis[i]) continue;
    vis[i]=1;
    for (int j=fir[i];j;j=e[j].next)
    {
      int k=e[j].t;
      if (!vis[k]&&dis[i]+e[j].val<dis[k])
      {
        dis[k]=dis[i]+e[j].val;
        q.push(node(k,dis[k]));
      }
    }
  }
}
```

5.3 Hungary

```
//匈牙利算法(二分图匹配)
bool match(int i)
{
  for (int j=fir[i];j;j=e[j].next)
  {
    int k=e[j].t;
    if (!vis[k])
    {
      vis[k]=1;
      if ((!mch[k])||(match(mch[k])))
      {
        mch[k]=i;
        return 1;
      }
    }
  }
  return 0;
}

int hungary()
{
  int tmp=0;
  memset(mch,0,sizeof(mch));
  for (int i=1;i<=n;i++)
  {
    memset(vis,0,sizeof(vis));
    if (match(i)) tmp++;
  }
  return tmp;
}
```

5.4 LCA

```
//倍增LCA
void set_father(int n)
{
  for (int k=1;k<=low;k++)
    for (int i=1;i<=n;i++)
      fa[i][k]=fa[fa[i][k-1]][k-1];
}

int get_father(int x,int y)
{
  if (dep[x]>dep[y]) swap(x,y);
  for (int i=low;i>=0;i--)
    if (dep[fa[y][i]]>=dep[x]) y=fa[y][i];
  if (x==y) return x;
  for (int i=low;i>=0;i--)
    if (fa[x][i]!=fa[y][i]) x=fa[x][i],y=fa[y][i];
  return fa[x][0];
}
```

5.5 Prim

```
//prim算法
void prim()
{
    int ans=0;
    for (int i=1;i<=n;i++) d[i]=e[0][i],v[i]=0;
    v[0]=1;
    for (int i=1;i<=n;i++)
    {
        int min=inf,k;
        for (int j=1;j<=n;j++)
            if (!v[j]&&d[j]<min)
            {
                min=d[j];
                k=j;
            }
        v[k]=1;
        ans+=min;
        for (int j=1;j<=n;j++)
        if (!v[j]&&e[k][j]<d[j])
```

```
      d[j]=e[k][j];
    }
    printf("%d\n",ans);
}

//堆优化prim

priority_queue<node> q;

int prim(int n)
{
    int ans=0;
    memset(vis,0,sizeof(vis));
    for (int i=0;i<=n;i++) dis[i]=inf;
    while (!q.empty())q.pop();
  q.push(node(0,0));
    dis[0]=0;
    while (!q.empty())
    {
      int i=q.top().id;
      q.pop();
      if (vis[i]) continue;
      vis[i]=1;
      ans+=dis[i];
        for (int j=fir[i];j;j=e[j].next)
        {
          int k=e[j].t;
          if (!vis[k] && e[j].val<dis[k])
          {
        dis[k]=e[j].val;
        q.push(node(k,dis[k]));
          }
    }
    }
    return ans;
}
```

5.6 SAP_MaxFlow

```
//最大流sap
struct edge
{
  int s,t,val,next;
}e[maxm];

void add_edge(int x, int y, int z)
{
  e[++tot].s=x; e[tot].t=y; e[tot].val=z; e[tot].next=fir[x]; fir[x]=tot;
  e[++tot].s=y; e[tot].t=x; e[tot].val=0; e[tot].next=fir[y]; fir[y]=tot;
}

int dfs(int x, int flow)
{
  if (x==ed) return flow;
  int sap=0;
  for (int j=last[x];j;j=e[j].next)
  {
    int y=e[j].t;
    if (e[j].val&&dis[x]==dis[y]+1)
    {
      last[x]=j;
      int tmp=dfs(y, min(e[j].val, flow-sap));
      e[j].val-=tmp;
      e[j^1].val+=tmp;
      sap+=tmp;
      if (sap==flow) return sap;
    }
  }
  if (dis[st]>=num) return sap;
  if (!(--gap[dis[x]])) dis[st]=num;
  ++gap[++dis[x]];
  last[x]=fir[x];
  return sap;
}

int max_flow()
{
  gap[0]=num;
  memcpy(last,fir,sizeof(fir));
  int tmp=0;
  while (dis[st]<num) tmp+=dfs(st, inf);
  return tmp;
}

void init()
{
  st=0; ed=2*n+m+p+1; num=ed+1; tot=1;
  memset(fir,0,sizeof(fir));
  memset(last,0,sizeof(last));
  memset(dis,0,sizeof(dis));
  memset(gap,0,sizeof(gap));
}
```

5.7 SPFA

```
//负环spfa
queue<int> q;

bool neg_loop(int st)
{
  while (!q.empty()) q.pop();
  for (int i=0;i<=n;i++) dis[i]=inf;
  memset(times,0,sizeof(times));
  memset(vis,0,sizeof(vis));
  q.push(st);
  vis[st]=1;
  times[st]=1;
  dis[st]=0;
  while (!q.empty())
  {
    int i=q.front();
    q.pop();
    for (int j=fir[i];j;j=e[j].next)
    {
      int k=e[j].t;
      if (dis[k]>dis[i]+e[j].val)
      {
        dis[k]=dis[i]+e[j].val;
        if (!vis[k])
        {
          q.push(k);
          vis[k]=1;
          times[k]++;
          if (times[k]>=n) return 1;
        }
      }
    }
    vis[i]=0;
  }
  return 0;
}
```

5.8 SPFA_CostFlow

```
//spfa费用流(min)
struct edge
{
  int s, t, val, cost, next;
}e[maxm];

void add_edge(int x, int y, int z, int w)
{
  e[++tot].s=x; e[tot].t=y; e[tot].val=z; e[tot].cost=w; e[tot].next=fir[x], fir[x]=tot;
  e[++tot].s=y; e[tot].t=x; e[tot].val=0; e[tot].cost=-w; e[tot].next=fir[y]; fir[y]=tot;
}

void init()
{
  st=0, ed=n*m+1; tot=1;
  memset(fir,0,sizeof(fir));
  memset(dis,0,sizeof(dis));
}

bool find_path()
{
  for (int i=st;i<=ed;i++) dis[i]=inf;//max
  queue<int> q;
  q.push(st);
  dis[st]=0;
  inque[st]=1;
  while (!q.empty())
```

```
  {
    int x=q.front();
    q.pop();
    inque[x]=0;
    for (int j=fir[x];j;j=e[j].next)
    {
      int y=e[j].t;
      if (e[j].val && dis[y]>dis[x]+e[j].cost)//max
      {
        dis[y]=dis[x]+e[j].cost;
        pre[y]=j;
        if (!inque[y]) q.push(y), inque[y]=1;
      }
    }
  }
  return dis[ed]<inf;//max
}

int fare_flow()
{
  int fare=0, flow=0;
  while (find_path())
  {
    int tmp=inf;
    for (int j=pre[ed];j;j=pre[e[j].s]) tmp=min(tmp, e[j].val);
    for (int j=pre[ed];j;j=pre[e[j].s]) e[j].val-=tmp, e[j^1].val+=tmp;
    flow+=tmp;
    fare+=tmp*dis[ed];
  }
  return fare;
}
```

5.9 Simplex

```
//单纯形(复杂度不确定，慎用!!!)
int a[maxn][maxm],next[maxm];
int n,m;

void pivot(int l,int e)
{
    int last=-1;
    for (int i=0;i<=m;i++)
        if (a[l][i])
        {
            next[i]=last;
            last=i;
        }
    for (int i=0;i<=n;i++)
    {
        if (a[i][e]==0||i==l) continue;
        for (int j=last;j!=-1;j=next[j])
        {
            //cout<<j<<' ';
            if (j==e) continue;
            a[i][j]-=a[i][e]*a[l][j];
        }
        //cout<<endl;
        a[i][e]=-a[i][e];
    }
}

int simplex()
{
    while (1)
    {
        int now=0;
        for (int i=1;i<=m;i++)
            if (a[0][i]>0) { now=i; break; }
        if (now==0) return -a[0][0];
        int tmp,mi=inf;
        for (int i=1;i<=n;i++)
        {
            if (a[i][now]>0&&a[i][0]<mi)
            {
                tmp=i;
                mi=a[i][0];
            }
        }
        pivot(tmp,now);
    }
}

int main()
{
    //freopen("defend.in","r",stdin);
    scanf("%d%d",&n,&m);
    for (int i=1;i<=n;i++) scanf("%d",&a[i][0]);
    int x,y;
    for (int i=1;i<=m;i++)
    {
        scanf("%d%d%d",&x,&y,&a[0][i]);
        for (int j=x;j<=y;j++)
            a[j][i]=1;
    }
    int ans=simplex();
    printf("%d\n",ans);
    return 0;
}
```

5.10 Tarjan_BCC

```
//tarjan割点
void tarjan(int now)
{
    dfn[now]=low[now]=++tim;
    v[now]=1;
    for (int j=fir[now];j;j=e[j].next)
  {
    int k=e[j].t;
    if (!v[k])
    {
      tarjan(k);
      low[now]=min(low[now],low[k]);
      if (dfn[now]<=low[k]) v[now]++;
    }
    else
      low[now]=min(low[now],dfn[k]);
  }
  if ((dfn[now]==1&&v[now]>2)||(dfn[now]>1&&v[now]>1))
    cut[now]=1;
}
//tarjan割边
void tarjan(int i)
{
  vis[i]=1;
  dfn[i]=low[i]=++cnt;
  for (int j=fir[i];j;j=e[j].next)
  {
    int k=e[j].t;
    if (!vis[k])
    {
      pre[k]=j^1;
      tarjan(k);
      low[i]=min(low[i],low[k]);
    }
    else if (j!=pre[i])
      low[i]=min(low[i],dfn[k]);
  }
  if (pre[i] && dfn[i]==low[i]) bridge[pre[i]]=bridge[pre[i]^1]=1;
}
```

5.11 Tarjan_SCC

```
//tarjan强连通分量
void tarjan(int now)
{
    dfn[now]=low[now]=++tim;
  stack[++top]=now;
    v[now]=1;
  for (int j=fir[now];j;j=e[j].next)
  {
    int k=e[j].t;
    if (!v[k]) tarjan(k);
    if (v[k]<2) low[now]=min(low[now],low[k]);
  }
  if (dfn[now]==low[now])
  {
    num++;
    while (stack[top+1]!=now)
    {
      web[stack[top]]=num;
      v[stack[top--]]=2;
```

```
    }
    }
}
```

5.12 ZKW_CostFlow

```
//zkw最大费用流
struct et
{
    int s,t,val,cost,next;
}e[maxm];
int fir[maxn],dis[maxn],v[maxn],q[maxm],pre[maxn];
bool inque[maxn];
int n,m,st,ed,tot,ans;

void prepare()
{
    for (int i=st;i<=ed;i++) dis[i]=-inf;//min
    int head=0,tail=1;
    q[1]=ed; dis[ed]=0; inque[ed]=1;
    while (head<tail)
    {
        int now=q[++head];
        for (int j=fir[now];j;j=e[j].next)
        {
            int k=e[j].t;
            if (e[j^1].val&&dis[k]<dis[now]+e[j^1].cost)//min
            {
                dis[k]=dis[now]+e[j^1].cost;
                if (!inque[k]) q[++tail]=k,inque[k]=1;
            }
        }
        inque[now]=0;
    }
}

int dfs(int now,int flow)
{
    if (now==ed)
    {
        ans+=dis[st]*flow;
        return flow;
    }
    int sap=0;    v[now]=1;
    for (int j=fir[now];j;j=e[j].next)
    {
        int k=e[j].t;
        if (!v[k]&&e[j].val&&dis[now]==dis[k]+e[j].cost)
        {
            int tmp=dfs(k,min(e[j].val,flow-sap));
            e[j].val-=tmp;
            e[j^1].val+=tmp;
            sap+=tmp;
            if (sap==flow) return sap;
        }
    }
    return sap;
}

bool adjust()
{
    int tmp=-inf;
    for (int i=st;i<=ed;i++) if (v[i])
        for (int j=fir[i];j;j=e[j].next)
        {
            int k=e[j].t;
            if (!v[k]&&e[j].val) tmp=max(tmp,dis[k]+e[j].cost-dis[i]);//min
        }
    if (tmp==-inf) return 0;
    for (int i=st;i<=ed;i++) if (v[i])
        dis[i]+=tmp;
    return 1;
}

void add_edge(int x,int y,int z,int w)
{
    e[++tot].s=x; e[tot].t=y; e[tot].val=z; e[tot].cost=w; e[tot].next=fir[x]; fir[x]=tot;
    e[++tot].s=y; e[tot].t=x; e[tot].val=0; e[tot].cost=-w; e[tot].next=fir[y]; fir[y]=tot;
}

int zkw_flow()
{
  ans=0;
  prepare();
  do{
    do memset(v,0,sizeof(v));
    while (dfs(st,inf));
  }while (adjust());
}

int init()
{
  memset(fir,0,sizeof(fir));
  tot=1; st=0; ed=n+2;
}
```

6 Geometry

6.1 Geometry

```
//=======计算几何=======
const double EPS=1e-10;
const double PI=acos(-1.0);

int dcmp(double x){ if (fabs(x)<EPS) return 0; else return x>0 ?1 :-1;}

//=======点与向量=======

struct Point
{
  double x, y;
  Point(){}
  Point(double xx, double yy):x(xx), y(yy){}
  friend bool operator <(Point A, Point B)
  {
    return A.x<B.x || (A.x==B.x && A.y<B.y);
  }
  friend bool operator ==(Point A, Point B)
  {
    return dcmp(A.x-B.x)==0 && dcmp(A.y-B.y)==0;
  }
};
typedef Point Vec;
typedef vector<Point> Points;
typedef vector<Point> Poly;
double AngleOnEarth(Point A,Point B)
{
  double x1=PI*A.x/180.0;
  double y1=PI*A.y/180.0;
  double x2=PI*B.x/180.0;
  double y2=PI*B.y/180.0;
  return acos( cos(x1-x2)*cos(y1)*cos(y2)+sin(y1)*sin(y2) );
}

//向量加法
Vec operator +(Vec a, Vec b){ return Vec(a.x+b.x, a.y+b.y);}

//向量减法
Vec operator -(Point A, Point B){ return Vec(A.x-B.x, A.y-B.y);}

//向量数乘
Vec operator *(Vec a, double k){ return Vec(a.x*k, a.y*k);}

//向量数除
Vec operator /(Vec a, double k){ return Vec(a.x/k, a.y/k);}

//向量点积
double operator ^(Vec a, Vec b){ return a.x*b.x+a.y*b.y;}

//向量叉积
double operator *(Vec a, Vec b){ return a.x*b.y-a.y*b.x;}

//向量长度
double Len(Vec a) { return sqrt(a^a);}
```

```
//向量极角
double Ang(Vec a) { return atan2(a.y, a.x);}

//向量与向量夹角
double Ang(Vec a, Vec b) { return acos((a^b)/(Len(a)*Len(b)));}

//三角形有向面积的两倍
double Area2(Point A, Point B, Point C) { return (B-A)*(C-A);}

//向量逆时针旋转rad度
Vec Rotate(Vec a, double rad)
{
  return Vec(a.x*cos(rad)-a.y*sin(rad), a.x*sin(rad)+a.y*cos(rad));
}

//向量的单位方向向量
Vec Orient(Vec a)
{
  double l=Len(a);
  return a/l;
}

//向量的单位法向量
Vec Normal(Vec a)
{
  double l=Len(a);
  return Vec(-a.y/l, a.x/l);
}

//凸包
Poly ConvexHull(Points P)
{
  sort(P.begin(),P.end());
  P.erase(unique(P.begin(),P.end()),P.end());
  int n=P.size();
  Poly St(n+1);
  int m=0;
  for (int i=0;i<n;i++)
  {
    while (m>1 && dcmp((St[m-1]-St[m-2])*(P[i]-St[m-2]))<=0)
      m--;
    St[m++]=P[i];
  }
  int k=m;
  for (int i=n-2;i>=0;i--)
  {
    while (m>k && dcmp((St[m-1]-St[m-2])*(P[i]-St[m-2]))<=0)
      m--;
    St[m++]=P[i];
  }
  if (n>1) St.resize(--m);
  return St;
}

//旋转卡壳
//t为卡壳上顶点，l,r分别为左右顶点
//D为最大直径，S为外接矩形的最小面积，C为最小周长
void RotatingCalipers(Points &P,double &D, double &S, double &C)
{
  P=ConvexHull(P);
  int n=P.size();
  S=C=1e15,D=0;
  int t=1,l=1,r=1;
  for (int i=0;i<n;i++)
  {
    while (dcmp( (P[(i+1)%n]-P[i]) * (P[(t+1)%n]-P[t]) ) > 0) t=(t+1)%n;
    while (dcmp( (P[(i+1)%n]-P[i]) ^ (P[(r+1)%n]-P[r]) ) > 0) r=(r+1)%n;
    if (i==0) l=t;
    while (dcmp( (P[(i+1)%n]-P[i]) ^ (P[(l+1)%n]-P[l]) ) < 0) l=(l+1)%n;
    double h=(P[(i+1)%n]-P[i]) * (P[t]-P[i]) / Len(P[(i+1)%n]-P[i]);
    double w=(P[(i+1)%n]-P[i]) ^ (P[r]-P[l]) / Len(P[(i+1)%n]-P[i]);
    D=max(D,h);
    S=min(S,w*h);
    C=min(C,(w+h)*2);
  }
}

//多边形有向面积
double PolygonArea(Poly P)
{
  double area=0;
  int n=P.size();
  for (int i=1;i<n-1;i++)
    area+=(P[i]-P[0])*(P[i+1]-P[0]);
  return area/2;
}

//多边形周长
double PolygonPerimeter(Poly P)
{
  double len=0;
  int n=P.size();
  for (int i=0;i<n-1;i++)
    len+=Len(P[i+1]-P[i]);
  return len+Len(P[0]-P[n-1]);
}

//=======直线与线段=======

struct Line
{
  Point P;
  Vec v;
  double ang;
  Line(){}
  Line(Point pp, Vec vv):P(pp), v(Orient(vv)), ang(Ang(vv)){}
  friend bool operator <(Line a, Line b)
  {
    return a.ang<b.ang;
  }
  Point point(double k)
  {
    return P+v*k;
  }
};

typedef vector<Line> Lines;

//直线交点(确保唯一)
Point GetIntersection(Line a, Line b)
{
  Point P=a.P, Q=b.P;
  Vec v=a.v, w=b.v;
  Vec u=P-Q;
  double t=(w*u)/(v*w);
  return a.point(t);
}

//点到直线距离
double DistanceToLine(Point P, Line l)
{
  Point A=l.P, B=l.P+l.v;
  Vec u=B-A, v=P-A;
  //不取绝对值得到有向距离
  return fabs((u*v)/Len(u));
}

//点到线段距离
double DistanceToSeg(Point P, Point A, Point B)
{
  if (A==B) return Len(P-A);
  Vec u=B-A, v=P-A, w=P-B;
  if (dcmp(u^v)<0) return Len(v);
  else if (dcmp(u^w)>0) return Len(w);
  else return fabs((u*v)/Len(u));
}

//点在直线上投影
Point GetProjection(Point P, Line l)
{
  Point A=l.P, B=l.P+l.v;
  Vec v=B-A;
  return A+v*((v^(P-A))/(v^v));
}
```

```
210
211 //判断是否规范相交
212 bool ProIntersection(Point A1, Point A2, Point B1, Point
    B2)
213 {
214   double c1=(A2-A1)*(B1-A1);
215   double c2=(A2-A1)*(B2-A1);
216   double c3=(B2-B1)*(A1-B1);
217   double c4=(B2-B1)*(A2-B1);
218   return dcmp(c1)*dcmp(c2)<0 && dcmp(c3)*dcmp(c4)<0;
219 }
220
221 //判断点在线段上
222 bool OnSegment(Point P,Point A1,Point A2)
223 {
224   return dcmp((A1-P)*(A2-P))==0 && dcmp((A1-P)^(A2-P))<0;
225 }
226
227 //点与多边形的位置关系
228 int InPolygon(Point P,const Poly &Poly)
229 {
230   int cnt=0, n=Poly.size();
231   for (int i=0;i<n;i++)
232   {
233     if (OnSegment(P,Poly[i], Poly[(i+1)%n])) return -1;
234     int k=dcmp((Poly[(i+1)%n]-Poly[i])*(P-Poly[i]));
235     int d1=dcmp(Poly[i].y-P.y);
236     int d2=dcmp(Poly[(i+1)%n].y-P.y);
237     if (k>0 && d1<=0 && d2>0) cnt++;
238     if (k<0 && d2<=0 && d1>0) cnt--;
239   }
240   if (cnt!=0)  return 1;
241   else return 0;
242 }
243
244 //半平面交
245 bool OnLeft(Point P,Line l)
246 {
247   return dcmp(l.v*(P-l.P))>=0;
248 }
249
250 Poly HalfPlaneIntersection(vector<Line> l)
251 {
252   Poly Sol;
253   Sol.clear();
254   int n=l.size();
255   sort(l.begin(),l.end());
256   int first,last;
257   Point *p=new Point[n];
258   Line *q=new Line[n];
259   q[first=last=0]=l[0];
260   for (int i=1;i<n;i++)
261   {
262     while (first<last && !OnLeft(p[last-1],l[i])) last--;
263     while (first<last && !OnLeft(p[first],l[i])) first++;
264     q[++last]=l[i];
265     if (dcmp(q[last].v*q[last-1].v)==0)
266     {
267       last--;
268       if (OnLeft(l[i].P,q[last])) q[last]=l[i];
269     }
270     if (first<last) p[last-1]=GetIntersection(q[last-1], q
      [last]);
271   }
272   while (first<last && !OnLeft(p[last-1],q[first])) last
    --;
273   if (last-first<=1) return Sol;
274   p[last]=GetIntersection(q[last], q[first]);
275   for (int i=first;i<=last;i++) Sol.push_back(p[i]);
276   return Sol;
277 }
278
279 //=======圆相关=======
280
281 struct Circle
282 {
283   Point O;
284   double r;
285   Circle(Point O,double r):O(O),r(r){}
286   Point point(double rad)
287   {
288     return Point(O.x+r*cos(rad), O.y+r*sin(rad));
289   }
290 };
291
292 //圆和直线交点(方程法)
293 Points GetIntersection(Line L, Circle C, double &t1,
    double &t2)
294 {
295   Points Sol; Sol.clear();
296   double a=L.v.x, b=L.P.x-C.O.x, c=L.v.y, d=L.P.y-C.O.y;
297   double e=a*a+c*c, f= 2*(a*b+c*d), g=b*b-C.r*C.r;
298   double delta=f*f-4*e*g;
299   if (dcmp(delta)<0) return Sol;
300   if (dcmp(delta)==0)
301   {
302     t1=t2=-f/(2*e);
303     Sol.push_back(L.point(t1));
304   }
305   else//相交
306   {
307     t1=(-f-sqrt(delta))/(2*e);
308     Sol.push_back(L.point(t1));
309     t2=(-f+sqrt(delta))/(2*e);
310     Sol.push_back(L.point(t2));
311   }
312   return Sol;
313 }
314
315 //圆和直线交点(几何法)
316 Points GetIntersection(Line L, Circle C)
317 {
318   Points Sol; Sol.clear();
319   double d=DistanceToLine(C.O, L);
320   if (dcmp(d-C.r)>0) return Sol;//相离
321   Point ans=GetIntersection(L,Line(C.O,Normal(L.v)));
322   if (dcmp(d-C.r)==0)
323     Sol.push_back(ans);
324   else
325   {
326     double len=sqrt(C.r*C.r-d*d);
327     Vec v=Orient(L.v);
328     Sol.push_back(ans+v*len);
329     Sol.push_back(ans-v*len);
330   }
331   return Sol;
332 }
333
334 //圆与圆的交点
335 Points GetIntersection(Circle C, Circle D)
336 {
337   Vec v=D.O-C.O;
338   Points Sol; Sol.clear();
339   double d=Len(v);
340   if (dcmp(C.r-D.r)>0) swap(C,D);
341   if (dcmp(d)==0 || dcmp(C.r+D.r-d) <0 || dcmp(D.r-C.r-d)
    >0) return Sol;//同心 | 外离 | 内含
342   if (dcmp(C.r+D.r-d)==0 || dcmp(D.r-C.r-d)==0)//相切
343     Sol.push_back(C.O+Orient(v)*C.r);
344   else//相交
345   {
346     double rad=Ang(v);
347     double th=acos((C.r*C.r+d*d-D.r*D.r)/(2*C.r*d));
348     Point P=C.point(rad+th);
349     Point Q=C.point(rad-th);
350     Sol.push_back(P);
351     Sol.push_back(Q);
352   }
353   return Sol;
354 }
355
356 //过点P到圆C的切线
357 Lines GetTangents(Point P,Circle C)
358 {
359   Vec v=C.O-P,tmp;
360   Lines Sol; Sol.clear();
361   double d=Len(v);
362   if (d<C.r) return Sol;
363   if (dcmp(d-C.r)==0)
364     Sol.push_back(Line(P,Normal(v)));
365   else
366   {
367     double rad=asin(C.r/d);
368     Sol.push_back(Line(P,Rotate(v,rad)));
369     Sol.push_back(Line(P,Rotate(v,-rad)));
370   }
```

```
  return Sol;
}

//两圆公切线
Lines GetTangents(Circle C, Circle D)
{
  Point A,B;
  Lines Sol; Sol.clear();
  if (dcmp(C.r-D.r)<0) swap(C, D);
  double dd=(C.O.x-D.O.x)*(C.O.x-D.O.x)+(C.O.y-D.O.y)*(C.O.y-D.O.y);
  double rdif=C.r-D.r;
  double rsum=C.r+D.r;
  if (dcmp(dd-rdif*rdif)<0) return Sol;//内含
  if (dcmp(dd)==0 && dcmp(C.r-D.r)==0) return Sol;//重合
  double rad=atan2(D.O.y-C.O.y,D.O.x-C.O.x);
  if (dcmp(dd-rdif*rdif)==0)//内切
  {
    A=C.point(rad), B=D.point(rad);
    Sol.push_back(Line(A,B-A));
    //Sol.push_back(Line(A,Normal(C.O-D.O)));
    return Sol;
  }
  double th=acos((C.r-D.r)/sqrt(dd));
  A=C.point(rad+th), B=D.point(rad+th);
  Sol.push_back(Line(A,B-A));
  A=C.point(rad-th), B=D.point(rad-th);
  Sol.push_back(Line(A,B-A));
  if (dcmp(dd-rsum*rsum)==0)//外切
  {
    A=C.point(rad), B=D.point(rad+PI);
    Sol.push_back(Line(A,B-A));
    //Sol.push_back(Line(A,Normal(D.O-C.O)));
  }
  else if (dcmp(dd-rsum*rsum)>0)//相离
  {
    double ro=acos((C.r+D.r)/sqrt(dd));
    A=C.point(rad+ro), B=D.point(PI+rad+ro);
    Sol.push_back(Line(A,B-A));
    A=C.point(rad-ro), B=D.point(PI+rad-ro);
    Sol.push_back(Line(A,B-A));
  }
  return Sol;
}
```

7 Others

7.1 Bitwise

```
//位运算
while(mask<(1<<n))
{//枚举大小为i的集合
  int x=mask&-mask,y=mask+x;
  int p=mask;
  do
  {//枚举包含最低位的mask的子集p
    if ((p & x) >0)//保证该子集不被任何一个子集包含
    /*
    ...
    */
    p=(p-1)&(mask);
  }while (p!=mask);
  mask=((mask & ~y)/x >>1)|y;
}
```

7.2 MoDui

```
//莫队算法
#define inf 2147483647
struct query
{
    int l,r,s,w;
}a[maxn];
int c[maxn];
long long col[maxn],size[maxn],ans[maxn];
int n,m,cnt,len;

long long gcd(long long x,long long  y)
{
    return (!x)?y:gcd(y%x,x);
}

bool cmp(query a,query b)
{
    return (a.w==b.w)?a.r<b.r:a.w<b.w;
}

int main()
{
    //freopen("hose.in","r",stdin);
    scanf("%d%d",&n,&m);
    for (int i=1;i<=n;i++) scanf("%d",&c[i]);
    len=(int)sqrt(m);
    cnt=(len*len==m)?len:len+1;
    for (int i=1;i<=m;i++)
    {
        scanf("%d%d",&a[i].l,&a[i].r);
        if (a[i].l>a[i].r) swap(a[i].l,a[i].r);
        size[i]=a[i].r-a[i].l+1;
        a[i].w=a[i].l/len+1;
        a[i].s=i;
    }
    sort(a+1,a+m+1,cmp);
    int i=1;
    while (i<=m)
    {
        int now=a[i].w;
        memset(col,0,sizeof(col));
        for (int j=a[i].l;j<=a[i].r;j++) ans[a[i].s]+=2*(col[c[j]]++);
        i++;
        for (;a[i].w==now;i++)
        {
            ans[a[i].s]=ans[a[i-1].s];
            for (int j=a[i-1].r+1;j<=a[i].r;j++)
                ans[a[i].s]+=2*(col[c[j]]++);
            if (a[i-1].l<a[i].l)
                for (int j=a[i-1].l;j<a[i].l;j++)
                    ans[a[i].s]-=2*(--col[c[j]]);
            else
                for (int j=a[i].l;j<a[i-1].l;j++)
                    ans[a[i].s]+=2*(col[c[j]]++);
        }
    }
    long long all,num;
    for (int i=1;i<=m;i++)
    {
        if (size[i]==1) all=1; else all=size[i]*(size[i]-1);
        num=gcd(ans[i],all);
        printf("%lld/%lld\n",ans[i]/num,all/num);
    }
    return 0;
}
```

7.3 OpenStack

```
//手动开栈
VS(C++):
#pragma comment(linker, "/STACK:102400000,102400000")

G++:
int size = 256 << 20; // 256MB
char *p = (char*)malloc(size) + size;
__asm__("movl %0, %%esp\n" :: "r"(p));
//__asm__("movl %0, %%rsp\n" :: "r"(p));
```

7.4 PK

```
//对拍
:loop
datamaker.exe
a1.exe a2.exe
fc a1.out a2.out
if %errorlevel%==1 pause
goto loop
```

7.5 QuickRead

```
//快速读(慎用!!!)
const int BufferSize=1<<16;
char buffer[BufferSize],*head,*tail;
inline char Getchar() {
```

```
    if(head==tail) {
        int l=fread(buffer,1,BufferSize,stdin);
        tail=(head=buffer)+l;
    }
    return *head++;
}
inline int read() {
    int x=0,f=1;char c=Getchar();
    for(;!isdigit(c);c=Getchar()) if(c=='-') f=-1;
    for(;isdigit(c);c=Getchar()) x=x*10+c-'0';
    return x*f;
}


inline LL read() {
    LL x=0,f=1;char c=getchar();
    for(;!isdigit(c);c=getchar()) if(c=='-') f=-1;
    for(;isdigit(c);c=getchar()) x=x*10+c-'0';
    return x*f;
}
```