ICPC Templates

Aether

November 7, 2020

Contents

1 数学

1.1 素数

1.1.1 唯一分解定理

```cpp
const int M = 1000000;
bool isprime[M+5];
int prime[M+5], cnt;

void solve() {
    for(int i = 1;i <= M; i++) isprime[i] = 1;
    for(int i = 3;i <= M; i++) if(!(i%2)) isprime[i] = 0;
    for(int i = 2;i <= M; i++) {
        if(isprime[i]) {
            for(int j = i*2; j<= M; j += i)
                isprime[j] = 0;
        }
    }
    isprime[1] = 0;
    for(int i = 2;i <= M; i++) {
        if(isprime[i]) prime[++cnt] = i;
    }
}

long long getfac(long long x) {
    long long ans=1;
    for(int i = 1;i <= cnt && prime[i] <= x; i++) {
        long long sum = 0;//当前质因数的幂指数
        while(x%prime[i] == 0) { //当是这个数的因子时
            sum++;
            x /= prime[i];
        }
        ans *= (sum+1);//应用定理的结论
    }
    if(x>1) ans *= 2;//当搜索不到的时候，如果这个数最后大于一，那么这个最后结果肯定是素数，并且指数是1
    return ans;
}
```

1.1.2 pollard-rho(大素数质因数分解大于 1e18)

```cpp
#include <bits/stdc++.h>

using namespace std;

#define ll long long
ll input(){
    ll x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
```

```
}

#define LL __int128

ll gcd(ll a,ll b){
    if(b==0) return a;
    return gcd(b,a%b);
}

ll powmod(ll a,ll b,ll mod){
    ll res=1;
    while(b){
        if(b&1) res=(LL)res*a%mod;
        a=(LL)a*a%mod;
        b>>=1;
    }
    return res;
}

bool Miller_Rabin(ll p){
    if(p<2) return 0;
    if(p==2) return 1;
    if(p==3) return 1;
    ll d=p-1,r=0;
    while(!(d&1)) ++r,d>>=1;
    for(ll k=0;k<10;k++){
        ll a=rand()%(p-2)+2;
        ll x=powmod(a,d,p);
        if(x==1||x==p-1) continue;
        for(int i=0;i<r-1;i++){
            x=(LL)x*x%p;
            if(x==p-1) break;
        }
        if(x!=p-1) return 0;
    }
    return 1;
}

ll f(ll x,ll c,ll n){return ((LL)x*x+c)%n;}

ll Pollard_Rho(ll x){
    ll s=0,t=0;
    ll c=(ll)rand()%(x-1)+1;
    int step=0,goal=1;
    ll val=1;
    for(goal=1;;goal<<=1,s=t,val=1){
        for(step=1;step<=goal;step++){
            t=f(t,c,x);
            val=(LL)val*abs(t-s)%x;
            if((step%127)==0){
                ll d=gcd(val,x);
                if(d>1) return d;
            }
```

```
64          }
65          ll d=gcd(val,x);
66          if(d>1) return d;
67      }
68  }
69
70  vector<ll> pri;
71
72  void fac(ll x){
73      if(x<2) return;
74      if(Miller_Rabin(x)){
75          pri.push_back(x);
76          return;
77      }
78      ll p=x;
79      while(p>=x) p=Pollard_Rho(x);
80      while(x%p==0) x/=p;
81      fac(x),fac(p);
82  }
83
84  int main(){
85      srand((unsigned)time(NULL));
86      int T=input();
87      while(T--){
88          pri.clear();
89          ll n=input();
90          ll max_factor=0;
91
92          fac(n);
93
94          for(int i=0;i<pri.size();i++){
95              max_factor=max(max_factor,pri[i]);
96          }
97
98          if(max_factor==n) printf("Prime\n");
99          else printf("%lld\n",max_factor);
100     }
101 }
```

1.1.3 素数测试

米勒拉宾算法，米勒-拉宾素性检验是一种概率算法，但是，Jim Sinclair 发现了一组数：2, 325, 9375, 28178, 450775, 9780504, 1795265022。用它们做 a ,2^{24} 以内不会出错，我们使用这组数，就不用担心运气太差了

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  #define IO ios::sync_with_stdio(false)
5  #define pb push_back
6  #define mk make_pair
7  const int N = 1e5+10;
```

```cpp
const int mod = 1e9+7;

ll a, b;

const long long S=20;

ll mult_mod(ll a,ll b,ll mod)//快速乘
{
    ll res=0;
    for(;b;b>>=1){
        if(b&1) res=(res+a)%mod;
        a=(a+a)%mod;
    }
    return res;
}
ll pow_mod(ll a,ll b,ll mod)//快速幂
{
    ll res=1;
    a%=mod;
    for(;b;b>>=1){
        if(b&1) res=mult_mod(res,a,mod);
        a=mult_mod(a,a,mod);
    }
    return res;
}
int check(ll a,ll n,ll x,ll t){
    ll ret=pow_mod(a,x,n);//费马小定理 a^(p-1)%p==1
    ll last=ret;
    for(ll i=1;i<=t;i++){//二次检测定理 如果p是一个素数，则x^2%p==1的解为，则x=1或者x=n-1。
        ret=mult_mod(ret,ret,n);
        if(ret==1&&last!=1&&last!=n-1) return 1;
        last=ret;
    }
    if(ret!=1) return 1;
    return 0;
}
int Miller_Rabin(ll n){
    if(n<2)return 0;
    if(n==2)return 1;
    if((n&1)==0) return 0;
    ll x=n-1;
    ll t=0;
    while((x&1)==0){x>>=1;t++;}
    for(ll i=0;i<S;i++){
        ll a=rand()%(n-1)+1;
        if(check(a,n,x,t)) return 0;
    }
    return 1;
}

int main(){
    for(ll i=100000000000;i<=100000000100;++i){
```

```
60        if(Miller_Rabin(i)) printf("%lld 是素数\n",i);
61        else printf("%lld 不是素数\n",i);
62    }
63 }
64
65
66 ////////////////////////////////////////////////////
67
68
69 #include<stdio.h>
70 #include<iostream>
71 #include<algorithm>
72 using namespace std;
73 typedef long long ll;
74 ll mul(ll a,ll b,ll mod){//高精度
75    a%=mod;
76    b%=mod;
77    ll c=(long double)a*b/mod;
78    ll ans=a*b-c*mod;
79    return (ans%mod+mod)%mod;
80 }
81 ll pow_mod(ll x,ll n,ll mod){//快速幂
82    ll res=1;
83    while(n){
84       if(n&1)
85       res=mul(res,x,mod);
86       x=mul(x,x,mod);
87       n>>=1;
88    }
89    return (res+mod)%mod;
90 }
91 bool Miller_Rabbin(ll a,ll n){
92
93    //把n-1 转化成 (2^r)*d
94    ll s=n-1,r=0;
95    while((s&1)==0){
96       s>>=1;r++;
97    }
98
99    //算出 2^d 存在 k 里
100   ll k=pow_mod(a,s,n);
101
102   //二次探测 看变化过程中是不是等于1 或 n-1
103   if(k==1)return true;
104   for(int i=0;i<r;i++,k=k*k%n){
105      if(k==n-1)return true;
106   }
107   return false;
108 }
109 bool isprime(ll n){
110   //这里可以随机取a值进行探测 探测次数可以自己定
111   ll times=7;
112   ll prime[100]={2,325,9375,28178,450775,9780504,1795265022};
```

```
    for(int i=0;i<times;i++){
        if(n==prime[i])return true;
        if(Miller_Rabbin(prime[i],n)==false)return false;//未通过探测 返回假
    }
    return true;//所有探测结束 返回真
}
```

1.2 线性筛

1.2.1 筛素数

```
const int N=1e5+5;
bool mark[N];
int prim[N];
int cnt;
void initial()
{
    cnt=0;
    for (int i=2 ; i<N ; ++i)
    {
        if (!mark[i])
            prim[cnt++]=i;
        for (int j=0 ; j<cnt && i*prim[j]<N ; ++j)
        {
            mark[i*prim[j]]=1;
            if (!(i%prim[j]))
                break;
        }
    }
}
```

1.2.2 筛约数个数

```
//sd(i) 表示 i 的约数和
//sp[i] 表示 i 的最小素因子的等比数列的和
//prim[i] 表示第 i 个素数
const int N=1e5+5;
bool mark[N];
int prim[N];
long long sd[N],sp[N];
int cnt;
void initial()
{
    cnt=0;
    sd[1]=1;
    for (int i=2 ; i<N ; ++i)
    {
        if (!mark[i])
        {
            prim[cnt++]=i;
            sd[i]=i+1;
            sp[i]=i+1;
```

```
        }
        for (int j=0 ; j<cnt && i*prim[j]<N ; ++j)
        {
            mark[i*prim[j]]=1;
            if (!(i%prim[j]))
            {
                sp[i*prim[j]]=sp[i]*prim[j]+1;
                sd[i*prim[j]]=sd[i]/sp[i]*sp[i*prim[j]];
                break;
            }
            sd[i*prim[j]]=sd[i]*sd[prim[j]];
            sp[i*prim[j]]=1+prim[j];
        }
    }
}
```

1.2.3 筛约数和

```
//d(i) 表示 i 的约数个数
//num[i] 表示 i 的最小素因子的个数
//prim[i] 表示 第 i 个素数
const int N=1e5+5;
bool mark[N];
int prim[N],d[N],num[N];
int cnt;
void initial()
{
    cnt=0;
    d[1]=1;
    for (int i=2 ; i<N ; ++i)
    {
        if (!mark[i])
        {
            prim[cnt++]=i;
            num[i]=1;
            d[i]=2;
        }
        for (int j=0 ; j<cnt && i*prim[j]<N ; ++j)
        {
            mark[i*prim[j]]=1;
            if (!(i%prim[j]))
            {
                num[i*prim[j]]=num[i]+1;
                d[i*prim[j]]=d[i]/(num[i]+1)*(num[i*prim[j]]+1);
                break;
            }
            d[i*prim[j]]=d[i]*d[prim[j]];
            num[i*prim[j]]=1;
        }
    }
}
```

1.3 GCD 及其扩展

1.3.1 欧几里得

```
int GCD(int a, int b){
    return b?GCD(b, a%b):a;
}
```

1.3.2 更相减损

```
int GCD(int a, int b) {
   while (a != b) {
      if (a > b)
         a = a - b;
      else
         b = b - a;
   }
   return a;
}
```

1.3.3 扩展欧几里得

```
//已知a, b求解一组x，y，使它们满足贝祖等式： ax+by = gcd(a, b) =d
int ExtendGCD(int a, int b, int *x, int *y) {
   if (b == 0) {
      *x = 1; *y = 0; // b=0
      return a;
   }

   int r = ExtendGCD(b, a%b, x, y);
   int t = *y; // temp
   *y = *x - (a / b)*(*y); // y1 = x2 - k*y2
   *x = t; // x1 = y2
   return r;
}
```

1.3.4 Stein 算法（针对存在大素数）

```
int SteinGCD(int a, int b) {
   if (a < b) { int t = a; a = b; b = t; }
   if (b == 0) return a;
   if ((a & 1) == 0 && (b & 1) == 0)
      return SteinGCD(a >> 1, b >> 1) << 1;
   else if ((a & 1) == 0 && (b & 1) != 0)
      return SteinGCD(a >> 1, b);
   else if ((a & 1) != 0 && (b & 1) == 0)
      return SteinGCD(a, b >> 1);
   else
      return SteinGCD(a - b, b);
}
```

1.4 欧拉函数

欧拉函数（Euler's totient function），即 $\varphi(n)$ ，表示的是小于等于 n 和 n 互质的数的个数。

1.4.1 欧拉函数（筛法）

```
void phi_table(int n, int* phi) {
  for (int i = 2; i <= n; i++) phi[i] = 0;
  phi[1] = 1;
  for (int i = 2; i <= n; i++)
    if (!phi[i])
      for (int j = i; j <= n; j += i) {
        if (!phi[j]) phi[j] = j;
        phi[j] = phi[j] / i * (i - 1);
      }
}
```

1.4.2 欧拉函数（直接求）

```
int euler_phi(int n) {
  int m = int(sqrt(n + 0.5));
  int ans = n;
  for (int i = 2; i <= m; i++)
    if (n % i == 0) {
      ans = ans / i * (i - 1);
      while (n % i == 0) n /= i;
    }
  if (n > 1) ans = ans / n * (n - 1);
  return ans;
}
```

1.5 同余

1.5.1 BSGS

BSGS（baby-step giant-step），即大步小步算法。常用于求解离散对数问题。形式化地说，该算法可以在 $O(\sqrt{n})$ 的时间内求解

$$a^x \equiv b(mod\ p)$$

其中 $a \perp p$。方程的解满足 $0 < x < p$。（不需要 p 为质数）

```
#include <bits/stdc++.h>

#define ll long long

using namespace std ;
unordered_map<ll, int> H ;
int N,M,P,ans; // N ^x = M (mod P)

ll gcd(ll a,ll b){
```

```
    if(!b)return a;
    return gcd(b,a%b);
}
ll expow(ll a,ll b,ll mod){
    ll res=1;
    while(b) res=((b&1)?res*a%mod:res),a=a*a%mod,b>>=1;
    return res;
}
ll exgcd(ll &x,ll &y,ll a,ll b){
    if(!b){x=1,y=0;return a;}
    ll t=exgcd(y,x,b,a%b);y-=x*(a/b);return t;
}
ll BSGS(ll a,ll b,ll mod,ll qaq){
    H.clear();ll Q,p=ceil(sqrt(mod)),x,y;
    exgcd(x,y,qaq,mod),b=(b*x%mod+mod)%mod,
    Q=expow(a,p,mod),exgcd(x,y,Q,mod),Q=(x%mod+mod)%mod;
    for(ll i=1,j=0;j<=p;++j,i=i*a%mod) if(!H.count(i))H[i]=j;
    for(ll i=b,j=0;j<=p;++j,i=i*Q%mod) if(H[i])return j*p+H[i];return -1;
}
ll exBSGS(){
    ll qaq=1;
    ll k=0,qwq=1;
    if(M==1) return 0;
    while((qwq=gcd(N,P))>1){
        if(M%qwq)return -1;
        ++k,M/=qwq,P/=qwq,qaq=qaq*(N/qwq)%P;
        if(qaq==M)return k;
    }
    return (qwq=BSGS(N,M,P,qaq))==-1? -1:qwq+k;
}
int main(){
    while(cin>>N){
        scanf("%d%d",&P,&M);if(!N&&!M&&!P) return 0;
        N%=P,M%=P,ans=exBSGS();if(ans<0) puts("No Solution");else cout<<ans<<'\n';
    }
}
```

1.5.2 CRT

中国剩余定理 (Chinese Remainder Theorem, CRT) 可求解如下形式的一元线性同余方程组（其中 $n_1, n_2, \cdots, n_k$ 两两互质）：

$$\begin{cases} x & \equiv a_1 \pmod{n_1} \\ x & \equiv a_2 \pmod{n_2} \\ & \vdots \\ x & \equiv a_k \pmod{n_k} \end{cases}$$

算法流程:

1. 计算所有模数的积 n；2. 对于第 i 个方程：1. 计算 $m_i = \frac{n}{n_i}$；2. 计算 m_i 在模 n_i 意义

下的逆元 m_i^{-1} ；3. 计算 $c_i = m_i m_i^{-1}$ （不要对 n_i 取模）。3. 方程组的唯一解为：$a = \sum_{i=1}^{k} a_i c_i \pmod{n}$ 。

应用:

某些计数问题或数论问题出于加长代码、增加难度、或者是一些其他不可告人的原因，给出的模数：不是质数！

但是对其质因数分解会发现它没有平方因子，也就是该模数是由一些不重复的质数相乘得到。

那么我们可以分别对这些模数进行计算，最后用 CRT 合并答案。

下面这道题就是一个不错的例子。

给出 G, n （$1 \le G, n \le 10^9$ ），求：

$$G^{\sum_{k|n} \binom{n}{k}} \bmod 999\ 911\ 659$$

首先，当 $G = 999\ 911\ 659$ 时，所求显然为 0 。

否则，根据欧拉定理，可知所求为：

$$G^{\sum_{k|n} \binom{n}{k} \bmod 999\ 911\ 658} \bmod 999\ 911\ 659$$

现在考虑如何计算：

$$\sum_{k|n} \binom{n}{k} \bmod 999\ 911\ 658$$

因为 999 911 658 不是质数，无法保证 $\forall x \in [1, 999\ 911\ 657]$ ，x 都有逆元存在，上面这个式子我们无法直接计算。

注意到 $999\ 911\ 658 = 2 \times 3 \times 4679 \times 35617$ ，其中每个质因子的最高次数均为一，我们可以考虑分别求出 $\sum_{k|n} \binom{n}{k}$ 在模 2 ，3 ，4679 ，35617 这几个质数下的结果，最后用中国剩余定理来合并答案。

也就是说，我们实际上要求下面一个线性方程组的解：

$$\begin{cases} x \equiv a_1 \pmod{2} \\ x \equiv a_2 \pmod{3} \\ x \equiv a_3 \pmod{4679} \\ x \equiv a_4 \pmod{35617} \end{cases}$$

而计算一个组合数对较小的质数取模后的结果，可以利用卢卡斯定理。

比较两 CRT 下整数:

考虑 CRT, 不妨假设 $n_1 \le n_2 \le ... \le n_k$

$$\begin{cases} x & \equiv a_1 \pmod{n_1} \\ x & \equiv a_2 \pmod{n_2} \\ & \vdots \\ x & \equiv a_k \pmod{n_k} \end{cases}$$

与 PMR(Primorial Mixed Radix) 表示

$x = b_1 + b_2 n_1 + b_3 n_1 n_2 ... + b_k n_1 n_2 ... n_{k-1}, b_i \in [0, n_i)$

将数字转化到 PMR 下，逐位比较即可

转化方法考虑依次对 PMR 取模

$$\begin{aligned} b_1 &= a_1 \bmod n_1 \\ b_2 &= (a_2 - b_1)c_{1,2} \bmod n_2 \\ b_3 &= ((a_3 - b_1')c_{1,3} - x_2')c_{2,3} \bmod n_3 \\ &... \\ b_k &= (...((a_k - b_1)c_{1,k} - b_2)c_{2,k}) - ...)c_{k-1,k} \bmod n_k \end{aligned}$$

其中 $c_{i,j}$ 表示 n_i 对 n_j 的逆元，$c_{i,j}n_i \equiv 1 \pmod{n_j}$

扩展：模数不互质的情况

两个方程:

设两个方程分别是 $x \equiv a_1 \pmod{m_1}$ 、$x \equiv a_2 \pmod{m_2}$ ；

将它们转化为不定方程：$x = m_1 p + a_1 = m_2 q + a_2$ ，其中 p, q 是整数，则有 $m_1 p - m_2 q = a_2 - a_1$ 。

由裴蜀定理，当 $a_2 - a_1$ 不能被 $\gcd(m_1, m_2)$ 整除时，无解；

其他情况下，可以通过扩展欧几里得算法解出来一组可行解 (p, q) ；

则原来的两方程组成的模方程组的解为 $x \equiv b \pmod{M}$ ，其中 $b = m_1 p + a_1$ ，$M = \text{lcm}(m_1, m_2)$ 。

多个方程:

用上面的方法两两合并就可以了……

```
1 → n
0 → ans
for i = 1 to k
    n * n[i] → n
for i = 1 to k
    n / n[i] → m
    inv(m, n[i]) → b // b * m mod n[i] = 1
    (ans + m * b) mod n → ans
return ans
```

1.5.3 exCRT(扩展中国剩余定理)

1.5.4 excrt (python 版本)

```python
x = []
y = []
x.append(1)
y.append(1)
ai = []
bi = []
n = 100
m = 100


def exgcd(a, b, x, y):
    if b == 0:
        x[0] = 1
        y[0] = 0
        return a
    gcd = exgcd(b, a % b, x, y);
    tp = x[0]
    x[0] = y[0]
    y[0] = tp - a // b * y[0]
    return gcd


def excrt():
    M = bi[1]
    Ans = ai[1]
    for i in range(2, n + 1):
        a = M
        b = bi[i]
        c = (ai[i] - Ans % b + b) % b
        gcd = exgcd(a, b, x, y)
        bg = b // gcd
        if c % gcd != 0:
            return -1

        x[0] = (x[0] * (c // gcd)) % bg

        Ans = Ans + x[0] * M

        M = M * bg
        Ans = (Ans % M + M) % M

    return (Ans % M + M) % M


n, m = map(int, input().split())

ai.append(-1)
bi.append(-1)

for i in range(1, n + 1):
    l,r = map(int, input().split())
    bi.append(1)
```

```
    ai.append(r)

Ans = excrt()

if Ans == -1:
    print("he was definitely lying");
elif Ans >= m:
    print("he was probably lying")
else:
    print(Ans)
```

1.5.5 二次剩余

一个数 a，如果不是 p 的倍数且模 p 同余于某个数的平方，则称 a 为模 p 的二次剩余。而一个不是 p 的倍数的数 b，不同余于任何数的平方，则称 b 为模 p 的非二次剩余。

对二次剩余求解，也就是对常数 a 解下面的这个方程：

$$x^2 \equiv a \pmod{p}$$

通俗一些，可以认为是求模意义下的开方。这里只讨论 $\boldsymbol{p}$ 为奇素数的求解方法，将会使用 Cipolla 算法。

解的数量:

对于 $x^2 \equiv n \pmod{p}$，能满足 " n 是模 p 的二次剩余 " 的 n 一共有 $\frac{p-1}{2}$ 个（0 不包括在内），非二次剩余有 $\frac{p-1}{2}$ 个。

Cipolla 算法:

找到一个数 a 满足 $a^2 - n$ 是非二次剩余，至于为什么要找满足非二次剩余的数，在下文会给出解释。这里通过生成随机数再检验的方法来实现，由于非二次剩余的数量为 $\frac{p-1}{2}$，接近 $\frac{p}{2}$，所以期望约 2 次就可以找到这个数。

建立一个 " 复数域 "，并不是实际意义上的复数域，而是根据复数域的概念建立的一个类似的域。在复数中 $i^2 = -1$，这里定义 $i^2 = a^2 - n$，于是就可以将所有的数表达为 $A + Bi$ 的形式，这里的 A 和 B 都是模意义下的数，类似复数中的实部和虚部。

在有了 i 和 a 后可以直接得到答案，$x^2 \equiv n \pmod{p}$ 的解为 $(a+i)^{\frac{p+1}{2}}$。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
ll w;
struct num{
    ll x,y;
};

num mul(num a,num b,ll p)
{
    num ans={0,0};
    ans.x=((a.x*b.x%p+a.y*b.y%p*w%p)%p+p)%p;
    ans.y=((a.x*b.y%p+a.y*b.x%p)%p+p)%p;
    return ans;
```

```
}

ll powwR(ll a,ll b,ll p){
    ll ans=1;
    while(b){
        if(b&1)ans=1ll*ans%p*a%p;
        a=a%p*a%p;
        b>>=1;
    }
    return ans%p;
}
ll powwi(num a,ll b,ll p){
    num ans={1,0};
    while(b){
        if(b&1)ans=mul(ans,a,p);
        a=mul(a,a,p);
        b>>=1;
    }
    return ans.x%p;
}

ll solve(ll n,ll p)
{
    n%=p;
    if(p==2)return n;
    if(powwR(n,(p-1)/2,p)==p-1)return -1;//不存在
    ll a;
    while(1)
    {
        a=rand()%p;
        w=((a*a%p-n)%p+p)%p;
        if(powwR(w,(p-1)/2,p)==p-1)break;
    }
    num x={a,1};
    return powwi(x,(p+1)/2,p);
}

int main()
{
    srand(time(0));
    int t;
    scanf("%d",&t);
    while(t--)
    {
        ll n,p;
        scanf("%lld%lld",&n,&p);
        if(!n){
            printf("0\n");continue;
        }
        ll ans1=solve(n,p),ans2;
        if(ans1==-1)printf("Hola!\n");//无解
        else
        {
```

```
68          ans2=p-ans1;
69          if(ans1>ans2)swap(ans1,ans2);
70          if(ans1==ans2)printf("%lld\n",ans1);
71          else printf("%lld %lld\n",ans1,ans2);
72       }
73    }
74 }
```

1.6 莫比乌斯反演

1.6.1 数论分块

```
1  long long ans = n * k;
2  for (long long l = 1, r; l <= n;
3      l = r + 1) { //此处l意同i,r意同j,下个计算区间的l应为上个区间的r+1
4    if (k / l != 0)
5      r = min(k / (k / l), n);
6    else
7      r = n; // l大于k时
8    ans -= (k / l) * (r - l + 1) * (l + r) /
9         2; //这个区间内k/i均相等,对i求和是等差数列求和
10 }
```

1.6.2 莫比乌斯反演

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  #define ll long long
6  ll input(){
7     ll x=0,f=0;char ch=getchar();
8     while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
9     while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
10    return f?-x:x;
11 }
12
13 const int N=50007;
14
15 bool vis[N];
16 int mu[N],pri[N],cnt=0;
17
18 void init(){
19    mu[1]=1;
20    for(int i=2;i<=N;++i){
21       if(!vis[i]) pri[++cnt]=i,mu[i]=-1;
22       for(int j=1;j<=cnt;++j){
23          if(i*pri[j]>N) break;
24          vis[i*pri[j]]=1;
25          if(i%pri[j]==0){mu[i*pri[j]]=0;break;}
26          else mu[i*pri[j]]=-mu[i];
27       }
```

```
    }
    for(int i=1;i<=N;i++) mu[i]+=mu[i-1];
}

int solve(int n,int m){
    int res=0;
    for(int i=1,j;i<=min(n,m);i=j+1){
        j=min(n/(n/i),m/(m/i));
        res+=(mu[j]-mu[i-1])*(n/i)*(m/i);
    }
    return res;
}

int main(){
    init();
    int t,a,b,c,d,k;
    t=input();
    for(int i=1;i<=t;i++){
        a=input()-1,b=input(),c=input()-1,d=input(),k=input();
        a/=k,b/=k,c/=k,d/=k;
        printf("%d\n",solve(b,d)-solve(a,d)-solve(c,b)+solve(a,c));
    }
}
```

1.7 多项式

1.7.1 FFT

```
#include <bits/stdc++.h>

using namespace std;

#define ll long long
#define debug printf("PASS IN LINE:%d\n",__LINE__)

ll input(){
    ll x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

#define N (int)300050
const double pi=acos(-1.0);

namespace FFT{
    int rev[N],bit,len;
    struct complex{
        double r,i;
        complex(){}
        complex(double _r,double _i){r=_r,i=_i;}
    };
```

```
25      complex operator +(const complex &x,const complex &y){return complex(x.r+y.r,x.i
            +y.i);}
26      complex operator -(const complex &x,const complex &y){return complex(x.r-y.r,x.i
            -y.i);}
27      complex operator *(const complex &x,const complex &y){return complex(x.r*y.r-x.i
            *y.i,x.r*y.i+x.i*y.r);}
28      void FFT(complex *a,int n,int f){
29          for(int i=0;i<n;i++){
30              if(i<rev[i]) swap(a[i],a[rev[i]]);
31          }
32          for(int p=1;p<n;p<<=1){
33              complex wn(cos(pi/p),sin(pi/p)*f);
34              for(int i=0;i<n;i+=(p<<1)){
35                  complex e(1,0);
36                  for(int j=0;j<p;j++,e=e*wn){
37                      complex x=a[i+j],y=a[i+j+p]*e;
38                      a[i+j]=x+y,a[i+j+p]=x-y;
39                  }
40              }
41          }
42          if(f==-1){
43              for(int i=0;i<n;i++) a[i].r/=n;
44          }
45          return;
46      }
47      void main(complex *a,complex *b,int *c,int n,int m,int &Len){
48          len=1,bit=0;
49          while(len<n+m-1) len<<=1,bit++;
50          for(int i=1;i<len;i++) rev[i]=(rev[i>>1]>>1)|((i&1)<<(bit-1));
51          FFT(a,len,1);FFT(b,len,1);
52          for(int i=0;i<len;i++) a[i]=a[i]*b[i];
53          FFT(a,len,-1);Len=n+m-1;
54          for(int i=0;i<Len;i++) c[i]=int(a[i].r+0.5);
55          while(Len-1&&!c[Len-1]) Len--;
56          return;
57      }
58  }
59
60  struct fnc{
61      int a[N],len;bool flag;
62
63      void Clear(){memset(a,0,sizeof(a));len=flag=0;}
64
65      fnc operator *(const fnc &x){
66          static fnc Kano;Kano.Clear();
67          static FFT::complex Arch[N],Bers[N];
68          Kano.flag=x.flag^flag;
69          for(int i=0;i<len;i++) Arch[i].r=a[i];
70          for(int i=0;i<x.len;i++) Bers[i].r=x.a[i];
71          FFT::main(Arch,Bers,Kano.a,len,x.len,Kano.len);
72          return Kano;
73      }
74  };
```

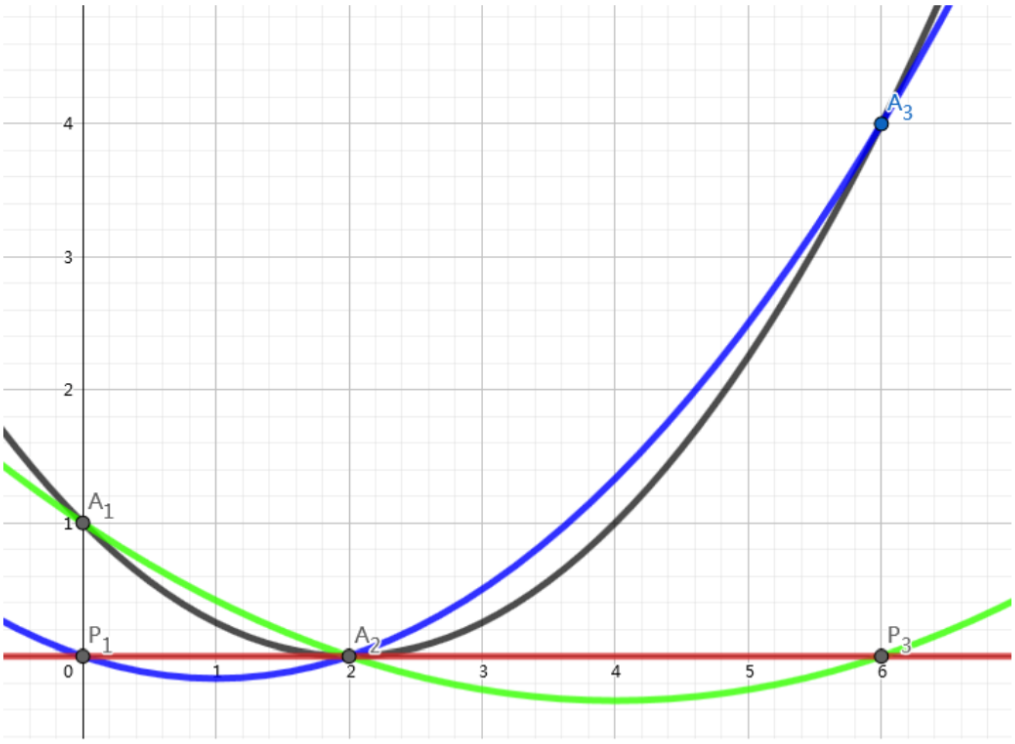
```
75
76  fnc _A,_B;
77
78  int main(){
79      _A.len=input()+1,_B.len=input()+1;
80      for(int i=0;i<_A.len;i++){
81          _A.a[i]=input();
82      }
83      for(int i=0;i<_B.len;i++){
84          _B.a[i]=input();
85      }
86      fnc _C=_A*_B;
87      for(int i=0;i<_C.len;i++){
88          printf("%d ",_C.a[i]);
89      }
90      printf("\n");
91      return 0;
92  }
```

1.7.2 拉格朗日插值法

题目大意

给出 n 个点 $P_i(x_i, y_i)$，将过这 n 个点的最多 $n-1$ 次的多项式记为 $f(x)$，求 $f(k)$ 的值。

考虑将每个点做一个对于 x 轴的垂线，设垂足为 $H_i(x_i,0)$ 。



如上图所示，黑线等于蓝线加绿线加红线。每次我们选择 1 个 P_i ，并选择其他的 $H_j[j\neq i]$ ，做一条过这些点的一条至多 $n-1$ 次的线。由于有 $n-2$ 个点都在 x 轴上，我们知道这条线的解析式一定是形如 $g_i(x)=y_i\times(\prod_{i=1}^{n}(x-x_i)[i\neq x])$ 的形式。

最后将所有的 $g(x)$ 相加，即 $f(x)=\sum_{i=1}^{n}g_i(x)$ 。因为对于每个点 P_i ，都只有一条函数经过 P_i ，其余都经过 H_i ，这一项的系数是 0 ，所以最后的和函数总是过所有 n 个点的。

公式整理得：

$$f(x)=\sum_{i=1}^{n}y_i\times(\prod_{j\neq i}\frac{x-x_j}{x_i-x_j})$$

如果要将每一项都算出来，时间复杂度仍是 $O(n^2)$ 的，但是本题中只用求出 $f(k)$ 的值，所以只需将 k 代入进式子里得：

$$\text{answer}=\sum_{i=1}^{n}y_i\times(\prod_{j\neq i}\frac{k-x_j}{x_i-x_j})$$

本题中，还需要求解逆元。如果先分别计算出分子和分母，再将分子乘进分母的逆元，累加进最后的答案，时间复杂度的瓶颈就不会在求逆元上，时间复杂度为 $O(n^2)$ 。

```
#include <algorithm>
#include <cstdio>
#include <cstring>
const int maxn = 2010;
using ll = long long;
ll mod = 998244353;
ll n, k, x[maxn], y[maxn], ans, s1, s2;
ll powmod(ll a, ll x) {
```

```
  ll ret = 1ll, nww = a;
  while (x) {
    if (x & 1) ret = ret * nww % mod;
    nww = nww * nww % mod;
    x >>= 1;
  }
  return ret;
}
ll inv(ll x) { return powmod(x, mod - 2); }
int main() {
  scanf("%lld%lld", &n, &k);
  for (int i = 1; i <= n; i++) scanf("%lld%lld", x + i, y + i);
  for (int i = 1; i <= n; i++) {
    s1 = y[i] % mod;
    s2 = 1ll;
    for (int j = 1; j <= n; j++)
      if (i != j)
        s1 = s1 * (k - x[j]) % mod, s2 = s2 * ((x[i] - x[j] % mod) % mod) % mod;
    ans += s1 * inv(s2) % mod;
    ans = (ans + mod) % mod;
  }
  printf("%lld\n", ans);
  return 0;
}
```

1.7.3 BM 线性递推

给定一个有 n 个元素的数列 a，其中第 i 个元素是 a_i 。

求一个较短/最短的数列 b，假设 b 有 m 个元素，那么要求满足

$$\forall m < i \le n, \quad a_i = \sum_{j=1}^{m} a_{i-j} b_j$$

要求在 $O(n^2)$ 的时间复杂度内解决此问题。

```
#include <bits/stdc++.h>

using namespace std;

#define ll long long
ll input(){
    ll x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

const ll mod=1e9+7;

#define VI vector<int>

ll powmod(ll a,ll b){
```

```
18      ll res=1;
19      while(b){
20          if(b&1) res=res*a%mod;
21          a=a*a%mod;
22          b>>=1;
23      }
24      return res;
25  }
26
27  int n;
28
29  #define debug printf("pass in line:%d\n",__LINE__)
30
31  namespace linear_seq{
32      const int N=10010;
33      ll res[N],base[N],_c[N],_md[N];
34
35      vector<int> Md;
36
37      void mul(ll *a,ll *b,int k){
38          for(int i=0;i<k*2;i++) _c[i]=0;
39          for(int i=0;i<k;i++)
40              if(a[i])
41                  for(int j=0;j<k;j++)
42                      _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
43          for(int i=k*2-1;i>=k;i--)
44              if(_c[i])
45                  for(int j=0;j<Md.size();j++)
46                      _c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%mod;
47          for(int i=0;i<k;i++) a[i]=_c[i];
48      }
49
50      int solve(ll n,VI a,VI b){
51          ll ans=0,pnt=0;
52          int k=a.size();
53          for(int i=0;i<k;i++) _md[k-1-i]=-a[i];_md[k]=1;
54          Md.clear();
55          for(int i=0;i<k;i++) if(_md[i]!=0) Md.push_back(i);
56          for(int i=0;i<k;i++) res[i]=base[i]=0;
57          res[0]=1;
58          while((1ll<<pnt)<=n) pnt++;
59          for(int p=pnt;p>=0;p--){
60              mul(res,res,k);
61              if((n>>p)&1){
62                  for(int i=k-1;i>=0;i--) res[i+1]=res[i];res[0]=0;
63                  for(int j=0;j<Md.size();j++) res[Md[j]]=(res[Md[j]]-res[k]*_md[Md[j]])%
                        mod;
64              }
65          }
66          for(int i=0;i<k;i++) ans=(ans+res[i]*b[i])%mod;
67          if(ans<0) ans+=mod;
68          return ans;
69      }
```

```
70
71    VI BM(VI s){
72        VI C(1,1),B(1,1);
73        int L=0,m=1,b=1;
74        for(int n=0;n<s.size();n++){
75            ll d=0;
76            for(int i=0;i<=L;i++) d=(d+(ll)C[i]*s[n-i])%mod;
77            if(d==0) ++m;
78            else if(2*L<=n){
79                VI T=C;
80                ll c=mod-d*powmod(b,mod-2)%mod;
81                while(C.size()<B.size()+m) C.push_back(0);
82                for(int i=0;i<B.size();i++) C[i+m]=(C[i+m]+c*B[i])%mod;
83                L=n+1-L,B=T,b=d,m=1;
84            }else{
85                ll c=mod-d*powmod(b,mod-2)%mod;
86                while(C.size()<B.size()+m) C.push_back(0);
87                for(int i=0;i<B.size();i++) C[i+m]=(C[i+m]+c*B[i])%mod;
88                ++m;
89            }
90        }
91        return C;
92    }
93
94    int gao(VI a,ll n){
95        VI c=BM(a);
96        c.erase(c.begin());
97        for(int i=0;i<c.size();i++) c[i]=(mod-c[i])%mod;
98        return solve(n,c,VI(a.begin(),a.begin()+c.size()));
99    }
100 }
101
102 int main(){
103     while(~scanf("%d",&n)){
104         VI v;
105         v.push_back(1);
106         v.push_back(2);
107         v.push_back(4);
108         v.push_bac(7);
109         v.push_back(13);
110         v.push_back(24);
111         printf("%d\n",linear_seq::gao(v,n-1));
112     }
113 }
```

1.8 线性代数

1.8.1 线性基

```
1 #include <bits/stdc++.h>
2
3 #define ll long long
```

```
4
5  struct LinearBasis{
6      static const int maxbase=64;
7
8      bool flag=false;
9      int cnt=0;
10     ll a[maxbase+7],b[maxbase+7];
11
12     LinearBasis(){
13         cnt=flag=0;
14         memset(a,0,sizeof a);
15     }
16
17     LinearBasis(ll *x, int n){
18         LinearBasis();
19         build(x,n);
20     }
21
22     void build(ll *x,int n){
23         for(int i=1;i<=n;++i)
24             insert(x[i]);
25     }
26
27     void clear(){
28         cnt=flag=0;
29         memset(a,0,sizeof a);
30     }
31
32     bool insert(ll t){
33         //暴力插入一个数，维护的是一个上三角型的线性基矩阵，时间复杂度低，当待插入元素能插入
               时，返回true
34         for(int i=maxbase;i>=0;--i){
35             if(t&(1ll<<i)){
36                 if(!a[i]){
37                     a[i]=t;
38                     break;
39                 }
40                 t^=a[i];
41             }
42         }
43         if(t==0)flag=true;
44         return t;
45     }
46
47     bool query(ll t){
48         // 询问t是否可以被当前线性基表示，不插入
49         if(t>queryMax()) return false;
50         if(t==0) return true;
51         for(int i=maxbase;i>=0;--i){
52             if(t&(1ll<<i)){
53                 if(!a[i]){
54                     return false;
55                 }
```

```
56              t^=a[i];
57          }
58      }
59      return true;
60  }
61
62  void Insert(ll t){
63      //插入一个线性基，利用高斯消元法维护一个对角矩阵
64      for(int i=maxbase;i>=0;--i){
65          if(t>>i&1){
66              if(a[i]) t^=a[i];
67              else{
68                  a[i] = t;
69                  for(int j=i-1;j>=0;--j)if(a[j]&&(a[i]>>j&1))a[i]^=a[j];
70                  for(int j=i+1;j<=maxbase;++j)if(a[j]>>j&1)a[j]^=a[i];
71                  break;
72              }
73          }
74      }
75  }
76
77  LinearBasis merge(const LinearBasis &l1, const LinearBasis &l2){
78      // 得到两个线性基的并
79      LinearBasis ret=l1;
80      for(int i=maxbase;i>=0;--i)
81          if(l2.a[i])
82              ret.insert(l2.a[i]);
83      return ret;
84  }
85
86  LinearBasis intersection(const LinearBasis &l1, const LinearBasis &l2){
87      //得到两个线性基的交
88      LinearBasis all,ret,full;
89      ret.clear();
90      for(int i=maxbase;i>=0;--i){
91          all.a[i]=l1.a[i];
92          full.a[i]=1ll<<i;
93      }
94      for(int i=maxbase;i>=0;--i){
95          if(l2.a[i]){
96              ll v=l2.a[i],k=0;
97              bool flag=true;
98              for(int j=maxbase;j>=0;--j){
99                  if(v & (1ll<<j)){
100                     if(all.a[j]){
101                         v^=all.a[j];
102                         k^=full.a[j];
103                     }
104                     else{
105                         // l2's basis is not in l1's;
106                         flag=false;
107                         all.a[j]=v;
108                         full.a[j]=k;
```

```
109                     break;
110                 }
111             }
112         }
113         if(flag){
114             ll v = 0; // get intersection by k;
115             for(int j=maxbase;j>= 0; --j){
116                 if(k&(1ll<<j)){
117                     v^=l1.a[j];
118                 }
119             }
120             ret.insert(v); //save ans
121         }
122     }
123 }
124     return ret;
125 }
126 //询问最值
127 ll queryMax(){
128     ll ret=0;
129     for(int i=maxbase;i>=0;--i)
130         if((ret^a[i])>ret)
131             ret^=a[i];
132     return ret;
133 }
134 ll queryMin(){
135     for(int i=0;i<=maxbase;++i)
136         if(a[i])
137             return a[i];
138     return 0;
139 }
140
141 void rebuild(){
142 //高斯消元方便查询第k大
143     for(int i=maxbase;i>=0;i--)
144         for(int j=i-1;j>=0;j--)
145             if(a[i]&(1ll<<j))
146                 a[i]^=a[j];
147     for(int i=0;i<=maxbase;i++)
148         if(a[i]) b[++cnt]=a[i];
149
150
151 }
152 ll kth(ll k){
153     //查询第K小的元素
154     ll ret=0;
155     if(flag) k--;
156     if(k>=(1ll<<cnt))//一共有2^cnt-1个元素
157         return -1;
158     for(int i=maxbase;i>=0;i--)//如果K的第i位有元素，则异或
159         if(k&(1ll<<i)) ret^=b[i];
160     return ret;
161 }
```

```
};
```

1.8.2 高斯消元

```
// 约旦-高斯消元
#include <bits/stdc++.h>

using namespace std;

#define ll long long
ll input(){
    ll x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

const int N=107;
double a[N][N];
int n;

int main(){
    n=input();
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n+1;j++)
            a[i][j]=input();

    for(int i=1;i<=n;i++){
        int mx=i;
        for(int j=i+1;j<=n;j++)
            if(fabs(a[j][i])>fabs(a[mx][i]))
                mx=j;

        for(int j=1;j<=n+1;j++)
            swap(a[i][j],a[mx][j]);

        if(!a[i][i]){
            printf("No Solution\n");
            exit(0);
        }

        for(int j=1;j<=n;j++){
            if(j!=i){
                double tmp=a[j][i]/a[i][i];
                for(int k=i+1;k<=n+1;k++){
                    a[j][k]-=a[i][k]*tmp;
                }
            }
        }
    }

    for(int i=1;i<=n;i++){
```

```
49          printf("%.2lf\n",a[i][n+1]/a[i][i]);
50      }
51  }
52
53  ////////////////////////////////////////////////
54
55
56  // //高斯消元
57  #include <bits/stdc++.h>
58
59  using namespace std;
60
61  #define ll long long
62  ll input(){
63      ll x=0,f=0;char ch=getchar();
64      while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
65      while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
66      return f? -x:x;
67  }
68
69  const ll N=107;
70  const double eps=1e-7;
71
72  double a[N][N],Ans[N];
73
74  int main(){
75      int n=input();
76      for(int i=1;i<=n;i++)
77          for(int j=1;j<=n+1;j++)
78              a[i][j]=input();
79
80      for(int i=1;i<=n;i++){
81          int mx=i;
82          for(int j=i+1;j<=n;j++)
83              if(fabs(a[mx][i])<fabs(a[j][i]))
84                  mx=j;
85
86          if(fabs(a[mx][i])<eps){
87              printf("No Solution\n");
88              return 0;
89          }
90
91          if(i!=mx) swap(a[i],a[mx]);
92
93          double tmp=a[i][i];
94          for(int j=i;j<=n+1;j++)
95              a[i][j]/=tmp;
96
97          for(int j=i+1;j<=n;j++){
98              tmp=a[j][i];
99              for(int k=i;k<=n+1;k++)
100                 a[j][k]-=a[i][k]*tmp;
101         }
```

```
    }

    Ans[n]=a[n][n+1];
    for(int i=n-1;i>=1;i--){
        Ans[i]=a[i][n+1];
        for(int j=i+1;j<=n;j++)
            Ans[i]-=(a[i][j]*Ans[j]);
    }
    for(int i=1;i<=n;i++)
        printf("%.2lf\n",Ans[i]);
}
```

1.9 计算几何

1.9.1 凸包面积

```
struct Point{
  double x,y;
}p[N];

int n;
double polygonarea()
{
    int i,j;
    double area = 0;
    for(i = 0;i < n;++i){
        j = (i+1)%n;
        area += p[i].x*p[j].y;
        area -= p[i].y*p[j].x;
    }
    //area /= 2.0;
    return area;
}
```

1.9.2 最近点对二维版本

```
// 题意：给n个二维坐标，求任意两点最近的欧几里得距离的一半。
#include<bits/stdc++.h>
using namespace std;
const int N=1e5+10;
struct node
{
    double x,y;
};
bool cmpx(node a,node b)
{
    return a.x<b.x;
}
bool cmpy(node a,node b)
{
    return a.y<b.y;
}
```

```
node p[N], a[N];
int cnt, n;
double dis(node a, node b)
{
    return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
}
double run(int l, int r)
{

    if(l + 1 == r) return dis(p[l], p[r]);
    if(l + 2 == r){
        return min({dis(p[l],p[l+1]), dis(p[l],p[l+2]), dis(p[l+1],p[l+2])});
    }
    int mid = l + r >> 1;
    double ans = min(run(l, mid), run(mid+1, r));
    cnt = 0;
    for(int i=l;i<=r;++i){
        if(p[i].x >= p[mid].x - ans && p[i].x <= p[mid].x + ans){
            a[cnt++] = p[i];
        }
    }
    sort(a, a+cnt, cmpy);
    for(int i=0;i<cnt;++i){
        for(int j=i+1;j<cnt;++j){
            if(a[j].y-a[i].y > ans) break;
            ans = min(ans, dis(a[i], a[j]));
        }
    }
    return ans;

}
int main()
{
    while(~scanf("%d", &n)&&n){
        for(int i=0;i<n;++i){
            scanf("%lf%lf", &p[i].x,&p[i].y);
        }
        sort(p, p+n, cmpx);
        printf("%.2f\n", run(0, n - 1) / 2);
    }
}
```

1.9.3 最近点对三维版本

```
#include<bits/stdc++.h>
using namespace std;
const int N=1e5+10;
struct node
{
    double x,y,z;
};
bool cmpx(node a,node b)
```

```
{
    return a.x<b.x;
}
bool cmpy(node a,node b)
{
    return a.y<b.y;
}
node p[N], a[N];
int cnt, n;
double dis(node a, node b)
{
    return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y)+(a.z-b.z)*(a.z-b.z));
}
double run(int l, int r)
{

    if(l + 1 == r) return dis(p[l], p[r]);
    if(l + 2 == r){
        return min({dis(p[l],p[l+1]), dis(p[l],p[l+2]), dis(p[l+1],p[l+2])});
    }
    int mid = l + r >> 1;
    double ans = min(run(l, mid), run(mid+1, r));
    cnt = 0;
    for(int i=l;i<=r;++i){
        if(p[i].x >= p[mid].x - ans && p[i].x <= p[mid].x + ans){
            a[cnt++] = p[i];
        }
    }
    sort(a, a+cnt, cmpy);
    for(int i=0;i<cnt;++i){
        for(int j=i+1;j<cnt;++j){
            if(a[j].y-a[i].y > ans) break;
            ans = min(ans, dis(a[i], a[j]));
        }
    }
    return ans;

}
int main()
{
    scanf("%d", &n);
    for(int i=0;i<n;++i){
        scanf("%lf%lf%lf", &p[i].x,&p[i].y,&p[i].z);
    }
    sort(p, p+n, cmpx);
    printf("%.3f\n", run(0, n - 1));
}
```

1.10 博弈论

策梅洛定理（Zermelo's theorem）指出，若一个游戏满足如下条件：

1. 双人、回合制；

2. 信息完全公开（perfect information）；

3. 无随机因素（deterministic）；

4. 必然在有限步内结束；

5. 没有平局；

则游戏中的任何一个状态，要么先手有必胜策略，要么后手有必胜策略（下文把这两种状态分别称为“胜态”、“败态”）。

常见的牌类游戏大多不满足条件 2、3；常见的棋类游戏（如井字棋、五子棋、围棋、象棋、跳棋）大多满足条件 2、3，在正式竞技中也会通过禁止循环的方式保证条件 4，但不一定满足条件 5。而第一节中提出的三种游戏，满足全部 5 个条件。

策梅洛定理的结论其实颇为显然。它的证明过程也就是必胜策略的构造过程：

对于终局状态，根据游戏规则可以判定“先手者”（即面对此状态的玩家）的胜负；对于非终局状态 A，可以考虑先手玩家走一步之后的所有可能状态（称为 A 的“次态”）：若 A 的次态全都是胜态，则 A 本身就是败态；否则，A 为胜态，且必胜策略就是在次态中选择一个败态留给对方。由于游戏会在有限步内结束，这个递归过程必然能够终止。 根据策梅洛定理，可以很容易地使用记忆化搜索算法判断一个状态是胜态还是败态：

```
mem = {}
def win(A): # 判断状态A是否为胜态
    if A not in mem:
        if is_final(A): # 若A为终局态
            mem[A] = rule(A) # 根据游戏规则判断A的胜负
        else: # 若A为非终局态，则根据策梅洛定理判断其胜负
            mem[A] = not all(win(B) for B in next_states(A))
                            # next_states(A)返回A的所有次态
    return mem[A]
```

需要注意，这里的“状态”是需要包含“下一步轮到谁”这一信息的。另外需要讨论一下游戏满足的第 4 个条件——策梅洛定理本身只要求游戏在有限步内结束，但如果要使用上面的记忆化搜索算法，则需要枚举一个状态的所有次态，这要求在游戏中每一步的可能走法数也是有限的。下文也只讨论游戏的总步数和每步的走法数都有限的情况，这种游戏称为“有限游戏”（finite game）。

游戏状态的组合

有些读者可能已经发现了，在第一节提出的三种游戏中，有许多状态是等价的。例如在 Flip Game 中，两个加号（“++”）和三个加号（“+++”）就是等价的状态，因为它们都是走一步之后就无路可走；在抢票游戏中，一个 2x3 的矩形和一个 3x2 的矩形也是等价的状态。状态的等价有许多种原因，其中一种是因为两个状态都是同样一些相互独立的子状态的组合。这里，“相互独立”的意思是指玩家的任意一步行动都只能影响一个子状态。例如，在 Flip Game 中，“++++-++-”和“-++-++++”就是等价的状态，因为它们都是“2 个加号”和“4 个加号”这两个子状态的组合。而 Nim 游戏的状态天然就是一些子状态的组合，其中每堆石子是一个子状态。如果能够通过子状态的胜负推断出它们的组合（下文称为“母状态”）的胜负，那么就可以大幅减少记忆化搜索过程中需要考虑的状态数，提高搜索效率。

在讨论由子状态胜负推断母状态胜负的方法之前，我想先指出第一节中三个游戏的另外三个共同特征。它们正是 Sprague-Grundy 定理成立的条件，也是下文所有讨论的前提。

游戏双方可以采取的行动是相同的。井字棋、五子棋、围棋、象棋、跳棋这些棋类游戏均不满足这个条件，因为游戏的双方只能下（或移动）己方的棋子。游戏双方的胜利目标是相同的。常见的胜利目标包括把棋盘清空或填满，或者把棋子排成特定的形状。注意，如果双方的目标是把棋子排成不同的形状，则游戏不满足这个条件。

满足上面两个条件的游戏称为 impartial game，反之则称为 partisan game。impartial game 的状态中只需包含棋盘信息，partisan game 的状态则还需包括“下面轮到谁”。正因为如此，partisan game 的状态无法拆分成“相互独立”的子状态，因为玩家的每一步行动会影响到所有子状态中“下面轮到谁”的信息。双方的胜利目标具体来说，是自己亲手达成终局状态，或者说走最后一步者为胜（术语称为 normal play）。第一节中的三个游戏也都可以稍微修改规则，改成走最后一步者为负（术语称为 misère play），但下文的讨论仅适用于 normal play 的情况。

下面讨论状态的组合对胜负的影响。请温习一下胜态和败态的关键性质：经过一步行动，败态只能变成胜态，胜态可以（但不一定）变成败态。

先看两个败态的组合。两个败态的组合还是败态。从后手玩家的角度来看，先手玩家的行动只能将两个败态中的一个改变为胜态，于是后手玩家可以再将这个胜态变成败态，从而将两个败态的组合抛回给先手玩家。由于终局状态为败态，最终先手玩家必将面对两个终局状态组成的败态，故后手必胜。

再看一胜一败两个状态的组合。胜态与败态的组合还是胜态——先手玩家只要把胜态变成败态，就可以把两个败态组合成的败态抛给后手玩家了。

最后看两个胜态的组合。这种组合就比较复杂了：先手玩家不应把其中一个胜态变成败态，因为这样会把一胜一败两个状态组合成的胜态留给对方。因此，先手玩家应当把其中一个胜态变成一个新的胜态。后手玩家面对新的胜态 + 胜态的组合，应当采取相同的策略。然而，由于游戏是有限的，早晚会有一个玩家只能把一个胜态变成败态，从而输掉游戏，但我们并不知道这会在哪一步发生。也就是说，仅仅知道两个子状态都是胜态，不足以推出母状态的胜负；我们需要挖掘胜态的更多性质。

Sprague-Grundy 数的提出

我们以 Flip Game 为例，研究一下胜态还有什么更深入的性质。

状态“++”是最简单的胜态，它只有一种走法，结果是败态。状态“+++”跟“++”在这一点上是一样的，因此它们其实是等价状态。状态“++++”就有两种不同的走法（对称的走法算同一种）：一是把中间两个加号变成减号，这样得到的次态“+−+”是个败态；二是把某一端的两个加号变成减号，这样得到的次态“−++”或“++−”（等价于“++”）是个胜态。

于是我们发现了两种不同的胜态。像“++”、“+++”这样，只能变成败态的胜态，我们称之为“一级胜态”。像“++++”这样，可以变成败态，也可以变成一级胜态的胜态，我们称之为“二级胜态”。类似地，如果一个胜态可以变成败态，也可以变成 1 至 n-1 级的所有胜态，则我们称之为“n 级胜态”。而败态可以称为“零级”。

我们看一下胜态的组合是否与级数有关。两个一级胜态的组合是败态，因为先手玩家的任意一步行动都会将其中一个胜态变为败态，留给后手玩家的就是胜态与败态组合成的胜态。一个一级胜态与一个二级胜态的组合是胜态，因为先手玩家可以将二级胜态变为一级胜态，留

给后手玩家的就是两个一级胜态组合成的败态。两个二级胜态组合成的胜态也是败态，因为先手玩家无论将其中一个二级胜态变成败态还是一级胜态，留给对方的组合都是胜态。

我们似乎发现了一个规律：两个同级胜态的组合是败态，两个不同级胜态的组合是胜态。没错！考察两个同级胜态的组合，无论先手玩家如何降低其中一个胜态的级数（甚至将其变成零级的败态），后手玩家总可以将另一个胜态降到同一级，最终先手玩家将面对两个败态组合成的败态。而若先手玩家面对的是两个不同级的胜态，他就总可以将其中较高级的胜态降至与较低级的胜态同级，这样留给后手玩家的就是败态。

上面对于胜态等级的定义有一个漏洞：如果一个胜态 A 可以变成败态或二级胜态，但不能变成一级胜态，那么它应该算一级还是三级呢？规律的证明过程同样也有一个漏洞：我们默认了一步行动只能让胜态的级数降低，那么能不能让胜态的级数升高呢？注意到规律证明过程的关键，在于如果要降低一个胜态的级数，则可以降低到任一级。于是我们就知道，上面的状态 A 应当定义为一级胜态。这导致胜态的级数可以升高，不过没关系，可以这样弥补规律证明的漏洞：在两个同级胜态的组合下，若先手玩家升高了其中一个胜态的级数，则后手玩家可以将它降回原级，这样两个同级胜态的组合仍是败态。

通过定义胜态的级数，我们解决了两个胜态组合而成的母状态的胜负判定问题。事实上，我们定义的“级数”，就是传说中的 Sprague-Grundy 数（简称 SG 数）。SG 数是一个从状态映射到非负整数的函数，它的形式化定义如下：

$$SG(A) = mex\{SG(B)|A \to B\}$$

式中 A、B 代表状态，$A \to B$ 代表 B 是 A 的一个次态。mex 是一个定义在集合上的函数，表示不属于集合的最小非负整数，它是 minimum excludant 的缩写。这个定义用通俗的语言表达，就是说一个状态的 SG 数，等于它的次态取不到的最小 SG 数。

状态组合时 Sprague-Grundy 数的运算规则

有了 SG 数，我们就可以判断任意两个子状态组合成的母状态的胜负了。但是，如果一个母状态是由三个子状态组成的，怎么办？我们发现，仅仅判断两个子状态组合成的母状态的胜负是不够的，我们还需要求出母状态的 SG 数。在下文中，我们用 $a \oplus b = c$ 表示 SG 数分别为 a、b 的两个子状态组合成的母状态的 SG 数为 c，我们的目标，就是弄清 $\oplus$ 运算的法则。当然，我们这么写默认了由子状态的 SG 数可以唯一确定母状态的 SG 数，这一点其实未经证明。

依然从最简单的情况开始。两个败态的组合还是败态，也就是说 $0 \oplus 0 = 0$。两个一级胜态的组合也是败态，即 $1 \oplus 1 = 0$。一个败态和一个一级胜态的组合，我们可以考虑最简单的情况：败态是终局态，不能再改变；而一级胜态只能变成败态。显然，这个组合的次态只能是两个败态组成的败态，故它本身是一级胜态，即 $0 \oplus 1 = 1$。

——$0 \oplus 0 = 0$，$1 \oplus 1 = 0$，$0 \oplus 1 = 1$，聪明的读者，你看出规律了吗？

如果你没看出规律，或者不相信你看出的规律，我们可以再算几个 SG 数稍微大一点儿的情况。考虑最简单的败态和二级胜态的组合：败态不能变化，二级胜态只能变成一级胜态或败态，于是组合的次态的 SG 数只能是 $0 \oplus 1 = 1$ 或 $0 \oplus 0 = 0$。这说明败态和二级胜态的组合是二级胜态，即 $0 \oplus 2 = 2$。同理可得 $0 \oplus 3 = 3$。再看胜态和胜态的组合。前面已经得到，两个同级胜态的组合为败态，故 $2 \oplus 2 = 3 \oplus 3 = 0$。那么两个不同级胜态的组合呢？

$1 \oplus 2$ 的次态可能是 $0 \oplus 2 = 2$、$1 \oplus 0 = 1$、$1 \oplus 1 = 0$，次态的 SG 数中 0、1、2 俱全，故 $1 \oplus 2 = 3$。

$1 \oplus 3$ 的次态可能是 $0 \oplus 3 = 3$、$1 \oplus 0 = 1$、$1 \oplus 1 = 0$、$1 \oplus 2 = 3$，次态的 SG 数缺少 2，故 $1 \oplus 3 = 2$。

$2 \oplus 3$ 的次态可能是 $0 \oplus 3 = 3$、$1 \oplus 3 = 2$、$2 \oplus 0 = 2$、$2 \oplus 1 = 3$、$2 \oplus 2 = 0$，次态的 SG 数缺少 1，故 $2 \oplus 3 = 1$。

现在看出规律了吗？相信了吗？

我们再换个角度，看看我们已经得到了 $\oplus$ 运算的哪些性质：

交换律 $a \oplus b = b \oplus a$：显然；

结合律 $(a \oplus b) \oplus c = a \oplus (b \oplus c)$：显然；

归零律 $a \oplus a = 0$：因为两个同级胜态的组合为败态；

恒等律 $0 \oplus a = 0$：本节已用最简单的情况说明。

具有这四个性质的二元运算是什么呢？是异或！

到此为止，我们通过举例的方法，发现了状态的组合对应着 SG 数的异或。不过，我们并没有证明通过子状态的 SG 数能够唯一确定母状态的 SG 数（即 $\oplus$ 运算结果的唯一性），也没有证明异或是能够达到这个目的的唯一一种运算。下面，我们就通过 SG 数和状态组合的定义，证明状态的组合对应着 SG 数的异或，即 $SG(A + B) = SG(A) \oplus SG(B)$，其中加号表示状态的组合，$\oplus$ 号表示异或。

下面给出 Sprague-Grundy 定理的完整表述：

若一个游戏满足以下条件：

1. 双人、回合制；
2. 信息完全公开（perfect information）；
3. 无随机因素（deterministic）；
4. 必然在有限步内结束，且每步的走法数有限（finite）；
5. 没有平局；
6. 双方可采取的行动及胜利目标都相同（impartial）；
7. 这个胜利目标是自己亲手达成终局状态，或者说走最后一步者为胜（normal play）；

则游戏中的每个状态可以按如下规则赋予一个非负整数，称为 Sprague-Grundy 数：

$$SG(A) = mex\{SG(B)|A \to B\}$$

（式中 A、B 代表状态，$A \to B$ 代表 A 状态经一步行动可以到达 B 状态，mex 表示一个集合所不包含的最小非负整数）。SG 数有如下性质：

1. SG 数为 0 的状态，后手必胜；SG 数为正的状态，先手必胜；
2. 若一个母状态可以拆分成多个相互独立的子状态，则母状态的 SG 数等于各个子状态的 SG 数的异或。

利用 Sprague-Grundy 定理，可以将记忆化搜索的程序优化成如下形式：

```
1 mem = {}
2 def SG(A): # 求状态A的SG数
```

```
    if A not in mem:
        S = sub_states(A) # sub_states(A)将A尽可能细致地拆分成子状态
        if len(S) > 1: # A可以拆分，用子状态的异或求其SG数
            mem[A] = reduce(operator.xor, [SG(B) for B in S])
        else: # A不可拆分，根据定义求其SG数
            mem[A] = mex(set(SG(B) for B in next_states(A)))
                        # next_states(A)返回A的所有次态
                        # 注意这条语句蕴含了“终局态的SG数为0”
    return mem[A]
```

1.10.1 T1(传统 sg 函数递推)

```
// hdu1848
/*
roblem Description
任何一个大学生对菲波那契数列(Fibonacci numbers)应该都不会陌生，它是这样定义的：

F(1)=1;

F(2)=2;

F(n)=F(n-1)+F(n-2)(n>=3);

所以，1,2,3,5,8,13……就是菲波那契数列。

在HDOJ上有不少相关的题目，比如1005 Fibonacci again就是曾经的浙江省赛题。
今天，又一个关于Fibonacci的题目出现了，它是一个小游戏，定义如下：

1、 这是一个二人游戏;

2、 一共有3堆石子，数量分别是m, n, p个；

3、 两人轮流走;

4、 每走一步可以选择任意一堆石子，然后取走f个；

5、 f只能是菲波那契数列中的元素（即每次只能取1，2，3，5，8…等数量）；

6、 最先取光所有石子的人为胜者；

假设双方都使用最优策略，请判断先手的人会赢还是后手的人会赢。
*/
#include <bits/stdc++.h>

using namespace std;

#define ll long long
ll input(){
    ll x=0,f=0;char ch=getchar();
```

```
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

const int N=1007;

int f[N],S[N],sg[N];

void init(){
    f[0]=1,f[1]=1;
    for(int i=2;f[i]<N;i++){
        f[i]=f[i-1]+f[i-2];
    }

    memset(sg,0,sizeof sg);

    for(int i=1;i<N;i++){
        memset(S,0,sizeof(S));
        for(int j=1;f[j]<=i;j++)
            S[sg[i-f[j]]]=1;

        for(int j=0;;j++)
            if(!S[j]){sg[i]=j;break;}
    }
}

int main(){
    init();

    int n,m,p;
    while(~scanf("%d%d%d",&n,&m,&p)){
        if(!(n||m||p)) break;
        if(sg[n]^sg[m]^sg[p]) printf("Fibo\n");
        else printf("Nacci\n");
    }
}
```

1.10.2 T2(利用 sg 函数在树上求 mex 知道答案)

```
#include <bits/stdc++.h>

using namespace std;

#define ll long long
ll input(){
    ll x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

```

```cpp
const int N=1e4+7;

int sg[N],vis[N];
vector<int> G[N];

void dfs(int u,int fa){
    for(auto v:G[u]){
        if(v==fa) continue;
        dfs(v,u);
    }

    memset(vis,0,sizeof vis);
    for(auto v:G[u]){
        if(v==fa) continue;
        vis[sg[v]]=1;
    }

    for(int i=0;i<N;i++){
        if(!vis[i]){sg[u]=i;break;}
    }
}

int main(){
    int T=input();
    while(T--){
        int n=input(),r=input();
        for(int i=1;i<=n;i++){
            G[i].clear();
            sg[i]=0;
        }

        for(int i=1;i<n;i++){
            int u=input(),v=input();
            G[u].push_back(v),G[v].push_back(u);
        }

        dfs(r,0);

        if(sg[r]) printf("Gen\n");
        else printf("Dui\n");
    }
}
```

1.10.3 T3(利用 sg 函数分析题目性质求出答案)

```cpp
//一棵树上，两个人进行博弈，每次操作至少删去一个叶子节点，不能删的人输。
/*
首先瞎编一个定义：一个叶子节点一直往上到分叉点为止（不包括分岔点）的这一条链称为一个枝条。

如果某棵树上存在一个长度为1的枝条，那么先手必胜。这个我们可以分两种情况进行证明：

如果删去这根长度为1的枝条，剩下的树先手必败，那么我们只需要删去这根长度为1的枝条，把必败的状
```

```
    态扔给对面就行了
8   如果删去这跟长度为1的枝条，剩下的树先手必胜，那么我们就删去剩下的树先手应该删的点，顺带删掉
    我们这个长度为1的枝条。
9   所以如果存在长度为1的枝条，先手必胜。那如果有人碰到枝条长度全是2的情况，就必败，因为它无论怎
    么删，必然制造出长度为1的枝条。
10
11  这样子的话，我们就可以把这棵树所有的枝条都拿出来，把他们的长度减去2，放进一个数组里。如果把
    数组里的数看成石子，题目转化成有若干个石子堆，每次至少拿走一颗石子，每个石子堆至多拿走一
    颗石子，不能拿的人输。
12
13  这就是个比较简单的博弈题，如果石子堆个数都是偶数，那先手必败，因为先手无论怎么取，后手都可以
    做一样的操作。如果存在奇数个石子堆，那先手必胜，他只需要把所有石子堆变成偶数即可。
14  */
15
16  #include <bits/stdc++.h>
17
18  using namespace std;
19
20  #define ll long long
21  ll input(){
22      ll x=0,f=0;char ch=getchar();
23      while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
24      while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
25      return f? -x:x;
26  }
27
28  const int N=2e6+7;
29
30  int deg[N],f[N];
31
32  int main(){
33      int T=input();
34      while(T--){
35          int n=input();
36
37          for(int i=0;i<=n;i++){
38              f[i]=deg[i]=0;
39          }
40
41          for(int i=2;i<=n;i++){
42              int x=input();
43              deg[x]++;f[i]=x;
44          }
45
46          int Ans=0;
47          for(int i=1;i<=n;i++){
48              if(deg[i]==0){
49                  int now=i,cnt=0;
50                  while(deg[now]<2&&now){
51                      now=f[now];
52                      cnt++;
53                  }
54                  if(cnt&1) Ans=1;
```

```
55          }
56      }
57
58      if(Ans) printf("Takeru\n");
59      else printf("Meiya\n");
60  }
61  }
```

1.11 其它

1.11.1 高精度

```
1   #include <bits/stdc++.h>
2
3   using namespace std;
4
5   #define ll long long
6
7   inline ll input(){
8       ll x=0,f=0;char ch=getchar();
9       while (ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
10      while (ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
11      return f? -x:x;
12  }
13
14  #define N 205
15  #define B 1000
16  const double pi=acos(-1.0);
17
18  namespace FFT{
19      int rev[N],bit,len;
20      struct complex{
21          double r,i;
22          complex(){}
23          complex(double _r,double _i){r=_r,i=_i;}
24      };
25      complex operator +(const complex &x,const complex &y){return complex(x.r+y.r,x.i
            +y.i);}
26      complex operator -(const complex &x,const complex &y){return complex(x.r-y.r,x.i
            -y.i);}
27      complex operator *(const complex &x,const complex &y){return complex(x.r*y.r-x.i
            *y.i,x.r*y.i+x.i*y.r);}
28      void FFT(complex *a,int n,int f){
29          for(int i=0;i<n;i++){
30              if(i<rev[i]) swap(a[i],a[rev[i]]);
31          }
32          for(int p=1;p<n;p<<=1){
33              complex wn(cos(pi/p),sin(pi/p)*f);
34              for(int i=0;i<n;i+=(p<<1)){
35                  complex e(1,0);
36                  for(int j=0;j<p;j++,e=e*wn){
37                      complex x=a[i+j],y=a[i+j+p]*e;
```

```
                a[i+j]=x+y,a[i+j+p]=x-y;
            }
        }
    }
    if(f==-1){
        for(int i=0;i<n;i++) a[i].r/=n;
    }
    return;
}
void main(complex *a,complex *b,int *c,int n,int m,int &Len){
    len=1,bit=0;
    while(len<n+m-1) len<<=1,bit++;
    for(int i=1;i<len;i++) rev[i]=(rev[i>>1]>>1)|((i&1)<<(bit-1));
    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);Len=n+m-1;
    for(int i=0;i<Len;i++) c[i]=int(a[i].r+0.5);
    while(Len-1&&!c[Len-1]) Len--;
    return;
}
}

struct BigNum{
    int a[N],len;bool flag;

    //Clear it,must
    void Clear(){memset(a,0x00,sizeof(a));len=flag=0;}

    //Get BigNum from Input,choice
    void GetInput(){
        static char ch[N];char* c;Clear();
        scanf("%s",ch+1);c=ch;
        if (ch[1]=='-') flag=true,c++;
        len=strlen(c+1);int k=100;
        for (int i=len;i;i--)
          k=k==100?1:k*10,a[(len-i)/3]+=(c[i]-'0')*k;
        len=(len-1)/3+1;
        return;
     }

    //Get BigNum from a number,choice
    void GetInt(int w){
        Clear();
        if (w<0) flag=true,w=-w;
        while (w) a[len++]=w%B,w/=B;
        return;
     }

    //Repair it,make it legal,must
    void Repair(){
        while (a[len]) a[len+1]=a[len]/B,a[len++]%=B;
        while (len-1&&!a[len-1]) len--;
        if (len==1&&!a[0]) flag=false;
```

```
91        return;
92     }
93
94     //Plus it with unsigned x and assign the sign of flag
95     //+,- must
96     BigNum Plus(BigNum x,bool flag){
97        static BigNum Kano;Kano.Clear();
98        Kano.len=max(len,x.len);Kano.flag=flag;
99        for (int i=0;i<Kano.len;i++)
100         Kano.a[i]+=a[i]+x.a[i],
101         Kano.a[i+1]=Kano.a[i]/B,
102         Kano.a[i]%=B;
103       Kano.Repair();
104       return Kano;
105    }
106
107    //Delete it by y from x and assign the sign of flag
108    //+,-,/,% must
109    BigNum Del(int x,BigNum y,bool flag){
110       static BigNum Kano;Kano.Clear();
111       Kano.len=len;Kano.flag=flag;
112       memcpy(Kano.a,a,sizeof a);
113       for (int i=0;i<y.len;i++){
114          Kano.a[i+x]-=y.a[i];
115          if (Kano.a[i+x]<0)
116            Kano.a[i+x]+=B,Kano.a[i+x+1]--;
117       }
118       Kano.Repair();
119       return Kano;
120    }
121
122    //Be Deleted by x
123    //+,- must
124    BigNum Deled(int x,BigNum y,bool flag){
125       static BigNum Kano;Kano.Clear();
126       Kano.len=y.len;Kano.flag=flag;
127       memcpy(Kano.a,y.a,sizeof y.a);
128       for (int i=0;i<len;i++){
129          Kano.a[i+x]-=a[i];
130          if (Kano.a[i+x]<0)
131            Kano.a[i+x]+=B,Kano.a[i+x+1]--;
132        }
133       Kano.Repair();
134       return Kano;
135    }
136
137    //Compare it without sign from the xth
138    //+,-,<, /, %,must
139    bool compare(const BigNum &y,int x=0){
140       if (len-x!=y.len) return len-x<y.len;
141       for (int i=len-1;~(i-x);i--)
142        if (a[i]!=y.a[i-x]) return a[i]<y.a[i-x];
143       return false;
```

```
144     }
145
146     //Multi it with a int
147     //<<,%,/,must
148     BigNum operator* (const int &x){
149        static BigNum Kano;Kano.Clear();
150        Kano.flag=flag^(x<0);Kano.len=len;
151        for (int i=0;i<len;i++)
152          Kano.a[i]+=a[i]*x,
153          Kano.a[i+1]=Kano.a[i]/B,
154          Kano.a[i]%=B;
155        Kano.Repair();
156        return Kano;
157     }
158
159     //Multi it with a BigNum by NTT
160     BigNum operator* (const BigNum &x){
161        static BigNum Kano;Kano.Clear();
162        static FFT::complex Arch[N],Bers[N];
163        Kano.flag=x.flag^flag;
164        for (int i=0;i<len;i++) Arch[i].r=a[i];
165        for (int i=0;i<x.len;i++) Bers[i].r=x.a[i];
166        FFT::main(Arch,Bers,Kano.a,len,x.len,Kano.len);
167        for (int i=0;i<Kano.len;i++)
168          Kano.a[i+1]+=Kano.a[i]/B,Kano.a[i]%=B;
169        Kano.Repair();
170        return Kano;
171     }
172
173     //Let it move to left for x bits
174     //%,/,must
175     BigNum operator<< (int x){
176        static BigNum Kano;
177        memcpy(Kano.a,a,sizeof a);
178        Kano.len=len;Kano.flag=flag;
179        while (x--) Kano=Kano*2;
180        return Kano;
181     }
182
183     //Compare it with sign
184     //-,must
185     bool operator< (const BigNum &x){
186        if (flag^x.flag) return flag;
187        if (len!=x.len) return (len<x.len)^flag;
188        for (int i=len-1;~i;i--)
189         if (x.a[i]!=a[i]) return (a[i]<x.a[i])^flag;
190        return false;
191     }
192
193     //Plus it to y with sign
194     BigNum operator+ (const BigNum &x){
195        if (flag==x.flag) return Plus(x,x.flag); else
196         if (compare(x)) return Deled(0,x,x.flag); else
```

```
197         return Del(0,x,flag);
198     }
199
200     //Delete y from it with sign
201     BigNum operator- (const BigNum &x){
202         bool Flag=*this<x;
203         if (flag^x.flag) return Plus(x,Flag); else
204          if (compare(x)) return Deled(0,x,Flag); else
205           return Del(0,x,Flag);
206     }
207
208     //Divide it by BigNum x
209     BigNum operator/ (BigNum &x){
210         static BigNum Kano,Hama,Kuro;Kano.Clear();
211         if (compare(x)) {Kano.len=1;return Kano;}
212         memcpy(Kuro.a,a,sizeof a);
213         Kano.flag=flag^x.flag;Kuro.flag=flag;
214         Kano.len=len-x.len+1;Kuro.len=len;
215         for (int i=len-x.len;~i;i--)
216          for (int j=9;~j;j--){
217             Hama=x << j;
218             if (!Kuro.compare(Hama,i))
219               Kuro=Kuro.Del(i,Hama,Kuro.flag),
220               Kano.a[i]+=1 << j;
221          }
222         Kano.Repair();
223         return Kano;
224     }
225
226     //Modulo it by BigNum x
227     BigNum operator% (BigNum &x){
228         static BigNum Hama,Kuro;
229         memcpy(Kuro.a,a,sizeof a);
230         Kuro.flag=flag;Kuro.len=len;
231         if (compare(x)) return Kuro;
232         for (int i=len-x.len;~i;i--)
233          for (int j=9;~j;j--){
234             Hama=x << j;
235             if (!Kuro.compare(Hama,i))
236               Kuro=Kuro.Del(i,Hama,Kuro.flag);
237          }
238         Kuro.Repair();
239         return Kuro;
240     }
241
242     //Output it
243     void Output(bool Flag = false){
244         if (flag) putchar('-');
245         printf("%d",a[len-1]);
246         for (int i=len-2;i>=0;i--){
247             if (a[i]<100) putchar('0');
248             if (a[i]<10) putchar('0');
249             printf("%d",a[i]);
```

```
250         }
251         if (Flag) puts("");
252         return;
253     }
254 };
255 
256 /*
257     define BigNum bn
258     support:
259         bn=bn+bn
260         bn=bn-bn
261         bn=bn*bn
262         bn=bn/bn
263         bn=bn%bn
264         int -> bn: bn.GetInt(int)
265         read_bn: bn.GetInput()
266 */
```

2 数据结构

2.1 链表

```
1  #include <bits/stdc++.h>
2  
3  #define DATA_SIZE 200
4  #define EXTEND_DATA_SIZE 50
5  #define NO 0
6  #define OK 1
7  #define ERROR -1
8  
9  //******基本数据类型别名******
10 typedef int Status;
11 typedef char Excelelem;
12 typedef int Numelem;
13 
14 //******链表数据结构******
15 typedef struct Node
16 {
17     Excelelem book;
18     struct Node *next;
19 }Liststruct;
20 
21 /******链表表头信息******/
22 typedef struct
23 {
24     Excelelem book[100]; //表头信息
25     Liststruct *next;
26     Numelem Length;
27 }ListHead;
28 
29 //******初始化链表******
30 Liststruct *init(int *i)
```

```
{
    Liststruct *Head,*p,*q;
    Excelelem ch;
    Head=q=NULL;
    printf("请输入顺序表Head的内容:\n");
    while((ch=getchar())!='\n')
    {
        p=(Liststruct *)malloc(sizeof(Liststruct));
        p->book=ch;
        if(!(*i)) Head=q=p,(*i)++;
        else
        {
            q->next=p;
            q=p;
            (*i)++;
        }
    }//注意*q++与(*q)++,有区别!
    if(q) q->next=NULL;
    return Head;
}

ListHead *Headinit()
{
    ListHead *Head;
    Head=(ListHead *)malloc(sizeof(ListHead));
    Head->Length=0;
    Head->next=init(&Head->Length);
    return Head;
}

/******打印表中数据内容******/
void DATA_cout(ListHead *Head)
{
    Liststruct *p=Head->next;
    while(p!=NULL)
    {
        printf("%c",p->book);
        p=p->next;
    }
    printf("\n");
    return ;
}
 /******打印表中local位置元素内容******/
void Local(ListHead* Head,int local)
{
    if(Head->Length<local || local<1)
    {
        printf("警告! 不存在%d位置的元素\n",local);
        return ;
    }
    Liststruct *p=Head->next;
    int i=1;
    while(p && i++!=local)
```

```
84          p=p->next;
85      printf("%c\n",p->book);
86      return ;
87  }
88
89  /******找到表中出现的字符ch的位置******/
90  void DATA_find(ListHead* Head,char ch)
91  {
92      int i=1,flag=0,count=1;
93      Liststruct *p=Head->next;
94      while(p)
95      {
96          if(p->book==ch)
97          {
98              flag=1;
99              printf("%d.在第%d个位置出现元素%c\n",count++,i,ch);
100         }
101         p=p->next;
102         i++;
103     }
104     if(!flag)
105         printf("未能找到%c元素！\n",ch);
106     return ;
107 }
108
109 /******在第k个元素前插入一个元素******/
110 void DATA_Insert(ListHead *Head,int k,Excelelem ch)
111 {
112     if(Head->Length<k || k<1)
113     {
114         printf("警告！不存在%d位置的元素\n",k);
115         return ;
116     }
117     Liststruct *p=Head->next,*q,*t;
118     int i=1;
119     while(p && i++!=k)
120         t=p,p=p->next;
121     q=(Liststruct *)malloc(sizeof(Liststruct));
122     q->book=ch;
123     if(k==1)
124     {
125         Head->next=q;
126         q->next=p;
127     }
128     else
129     {
130         q->next=p;
131         t->next=q;
132     }
133     Head->Length++;
134     return ;
135 }
136
```

```
137 /******删除第k个元素******/
138 void DATA_Delete(ListHead *Head,int k)
139 {
140     if(Head->Length<k || k<1)
141     {
142         printf("警告！不存在%d位置的元素\n",k);
143         return ;
144     }
145     int i=1;
146     Liststruct *p=Head->next,*q;
147     while(p && i++!=k)
148         q=p,p=p->next;
149     if(k==1)
150         Head->next=p->next;
151     else
152         q->next=p->next;
153     free(p);
154     Head->Length--;
155     return ;
156 }
157 
158 /******逆置链表******/
159 void DATA_UN(ListHead *Head)
160 {
161     Liststruct *p,*q;
162     p=Head->next;
163     Head->next=NULL;
164     while(p!=NULL)
165     {
166         q=p;
167         p=p->next;
168         q->next=Head->next;
169         Head->next=q;
170     }
171     return ;
172 }
173 
174 /******返还内存******/
175 void List_FREE(ListHead *Head)
176 {
177     Liststruct *p=Head->next,*q;
178     free(Head);
179     while(p)
180     {
181         q=p;
182         p=p->next;
183         free(q);
184     }
185     return ;
186 }
187 
188 int main()
189 {
```

```
    ListHead *Head;
    Numelem i;
    Excelelem ch;
    puts("");
    puts("******等待链表Head初始化!******");
    Head=Headinit();
    puts("******链表Head初始化完成!******");
    printf("链表中的内容为:\n");
    DATA_cout(Head);
    printf("链表Head的长度为:\n");
    printf("%d\n",Head->Length);
    printf("链表第%d个元素是:\n",i=2);
    Local(Head,i);
    printf("链表中出现%c元素的位置分别是:\n",ch='6');
    DATA_find(Head,ch);
    printf("在链表的第%d个元素之前插上%c\n",i=4,ch='9');
    DATA_Insert(Head,i,ch);
    printf("链表中的内容为:\n");
    DATA_cout(Head);
    printf("将链表中第%d个元素删除\n",i=3);
    DATA_Delete(Head,i);
    printf("链表中的内容为:\n");
    DATA_cout(Head);
    printf("将链表所有元素逆置，请稍后...\n\n"); //多种方法
    DATA_UN(Head);
    printf("链表中的内容为:\n");
    DATA_cout(Head);
    puts("******链表Head使用完毕!******\n");
    List_FREE(Head);
    return 0;
}
```

2.2 并查集

```
#include <bits/stdc++.h>

using namespace std;

#define ll long long
ll input(){
    ll x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

const int N=1e4+7;

int fa[N],rk[N];
int find(int x){ return fa[x]==x? x:(fa[x]=find(fa[x]));}
void merge(int x,int y){
    x=find(x),y=find(y);
```

```
    if(x!=y){
        if(rk[x]>rk[y]) fa[y]=x,rk[x]+=rk[y];
        else fa[x]=y,rk[y]+=rk[x];
    }
}

int main(){
    int n=input(),m=input();
    for(int i=1;i<=n;i++){
        fa[i]=i,rk[i]=1;
    }

    while(m--){
        int t=input(),x=input(),y=input();
        if(t==1) merge(x,y);
        else{
            if(find(x)==find(y)) printf("Y\n");
            else printf("N\n");
        }
    }
}
```

2.3 树状数组

```
#include <iostream>
#include <cstdio>
#define lowbit(x) x&-x
using namespace std;
int a[10001];
int n;
void change(int x,int ad){while(x<n) a[x]+=ad,x+=lowbit(x);}
int sum(int l,int r){
    int ans1=0,ans2=0;l--;
    while(l>0) ans1+=a[l],l-=lowbit(l);
    while(r>0) ans2+=a[r],r-=lowbit(r);
    return ans2-ans1;
}
int main(){
    scanf("%d",&n);
    int re;
    for(int i=1;i<=n;i++){
        scanf("%d",&re);
        change(i,re);
    }
    int m;
    int l,r;
    scanf("%d",&m);
    while(m--){
        scanf("%d%d",&l,&r);
        printf("%d\n",sum(l,r));
    }
}
```

2.4 线段树

```cpp
#include <bits/stdc++.h>

using namespace std;

#define ll long long
ll input(){
    ll f=0,x=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

const int N=50007;

ll n,m;
ll a[N];
struct node{
    ll mxl,mxr,mxs,sum;
}t[N*4];
node merge(node a,node b){
    node res;
    res.sum=a.sum+b.sum;
    res.mxs=max(a.mxr+b.mxl,max(a.mxs,b.mxs));
    res.mxl=max(a.mxl,a.sum+b.mxl);
    res.mxr=max(b.mxr,b.sum+a.mxr);
    return res;
}

void build(int rt,int l,int r){
    if(l==r){
        t[rt].sum=t[rt].mxl=t[rt].mxr=t[rt].mxs=a[l];
        return;
    }

    int mid=(l+r)>>1;
    build(rt<<1,l,mid),build(rt<<1|1,mid+1,r);
    t[rt]=merge(t[rt<<1],t[rt<<1|1]);
}

node query(int rt,int l,int r,int ql,int qr){
    if(ql<=l&&r<=qr)
        return t[rt];
    node res;
    int mid=(l+r)>>1;
    if(qr<=mid) res=query(rt<<1,l,mid,ql,qr);
    else if(ql>mid) res=query(rt<<1|1,mid+1,r,ql,qr);
    else res=merge(query(rt<<1,l,mid,ql,qr),query(rt<<1|1,mid+1,r,ql,qr));
    return res;
}

int main(){
```

```
52      n=input();
53      for(int i=1;i<=n;i++) a[i]=input();
54      build(1,1,n);
55      m=input();
56      for(int i=1;i<=m;i++){
57          int l=input(),r=input();
58          printf("%lld\n",query(1,1,n,l,r).mxs);
59      }
60  }
61
62
63  //////////////////////////////////////////////////////////////////////////
64
65  #include<iostream>
66  #include<cstdio>
67  #define INF 19260817
68  using namespace std;
69  int input(){
70      int x=0,f=1;char ch=getchar();
71      while(ch<'0'||ch>'9'){if(ch=='-')f=-1;ch=getchar();}
72      while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
73      return f*x;
74  }
75  int n,m;
76  class segment_tree{
77      public:
78          int a[100007];
79          void build(int rootr,int l,int r){
80              tree[rootr].l=l;tree[rootr].r=r;
81              if(l==r){tree[rootr].MAX=tree[rootr].MIN=tree[rootr].tot=a[l];return;}
82              int mid=(r-l>>1)+l;
83              build(rootr*2,l,mid);
84              build(rootr*2+1,mid+1,r);
85              put(rootr);
86          }
87          int MAX=0,MIN=INF,tot=0;
88          void query(int l,int r){
89              MAX=0,MIN=INF,tot=0;
90              getans(1,l,r);
91          }
92          void updata(int rootr,int l,int r,int add){
93              int L=tree[rootr].l,R=tree[rootr].r;
94              if(L>r||R<l)return;
95              if(l<=L&&R<=r){tree[rootr].tot+=add*(R-L+1);tree[rootr].MAX+=add;tree[
                    rootr].MIN+=add;tree[rootr].lazy+=add;return;}
96              putdown(rootr);
97              updata(rootr*2,l,r,add);
98              updata(rootr*2+1,l,r,add);
99              put(rootr);
100         }
101     private:
102     struct node{
103             int l,r,lazy,MAX,MIN,tot;
```

```
104         }tree[200007];
105     void put(int rootr){
106         tree[rootr].MAX=tree[rootr*2].MAX>tree[rootr*2+1].MAX? tree[rootr*2].MAX:tree
                [rootr*2+1].MAX;
107         tree[rootr].MIN=tree[rootr*2].MIN<tree[rootr*2+1].MIN? tree[rootr*2].MIN:tree
                [rootr*2+1].MIN;
108         tree[rootr].tot=tree[rootr*2].tot+tree[rootr*2+1].tot;
109     }
110     void putdown(int rootr){
111         if(tree[rootr].lazy!=0){
112             tree[rootr*2].lazy+=tree[rootr].lazy;
113             tree[rootr*2+1].lazy+=tree[rootr].lazy;
114             tree[rootr*2].MAX+=tree[rootr].lazy;tree[rootr*2].MIN+=tree[rootr].lazy;
115             tree[rootr*2+1].MAX+=tree[rootr].lazy;tree[rootr*2+1].MIN+=tree[rootr].
                    lazy;
116             tree[rootr*2].tot+=tree[rootr].lazy*(tree[rootr*2].r-tree[rootr*2].l+1);
117             tree[rootr*2+1].tot+=tree[rootr].lazy*(tree[rootr*2+1].r-tree[rootr*2+1].l
                    +1);
118             tree[rootr].lazy=0;
119         }
120     }
121     void getans(int rootr,int l,int r){
122         int L=tree[rootr].l,R=tree[rootr].r;
123         if(L>r||R<l)return;
124         if(l<=L&&R<=r){tot+=tree[rootr].tot;MAX=MAX>tree[rootr].MAX?MAX:tree[rootr].
                MAX;MIN=MIN<tree[rootr].MIN?MIN:tree[rootr].MIN;return;}
125         putdown(rootr);
126         getans(rootr*2,l,r);
127         getans(rootr*2+1,l,r);
128         put(rootr);
129     }
130 };
131 segment_tree tree1;
132 int main(){
133     n=input();
134     m=input();
135     for(int i=1;i<=n;i++){
136         tree1.a[i]=input();
137     }
138     tree1.build(1,1,n);
139     int fl,a,b,c;
140     for(int i=1;i<=m;i++){
141         fl=input();a=input();b=input();
142         if(fl==1){tree1.query(a,b);printf("%d\n%d\n%d\n",tree1.MIN,tree1.MAX,tree1.
                tot);}
143         if(fl==2){c=input();tree1.updata(1,a,b,c);}
144     }
145     return 0;
146 
147 }
148 
149 /*
150 10 10
```

```
1 2 3 4 5 6 7 8 9 10

*/
```

2.5 主席树

```cpp
#include <bits/stdc++.h>

using namespace std;

#define ll long long

int input(){
    int x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

#define N (int)(1e5+7)

int n,m;
struct node{
    int l,r,sum;
}t[N*40];
int rt[N],a[N];
int cnt,x,y,k;
vector <int> v;

int getid(ll x) {return lower_bound(v.begin(),v.end(),x)-v.begin()+1;}

void updata(int l,int r,int &x,int y,int pos) {
    t[++cnt]=t[y],t[cnt].sum++,x=cnt;
    if(l==r) return;
    int mid=(l+r)>>1;
    if(mid>=pos) updata(l,mid,t[x].l,t[y].l,pos);
    else updata(mid+1,r,t[x].r,t[y].r,pos);
}

int query(int l,int r,int x,int y,int k){
    if(l==r) return l;
    int mid=(l+r)>>1;
    int sum=t[t[y].l].sum-t[t[x].l].sum;
    if(sum>=k) return query(l,mid,t[x].l,t[y].l,k);
    else return query(mid+1,r,t[x].r,t[y].r,k-sum);
}

int main() {
    n=input(),m=input();
    for(int i=1;i<=n;i++) {
        a[i]=input();
        v.push_back(a[i]);
```

```
    }

    sort(v.begin(),v.end()),v.erase(unique(v.begin(),v.end()),v.end());

    for(int i=1;i<=n;i++) updata(1,n,rt[i],rt[i-1],getid(a[i]));

    for(int i=1;i<=m;i++) {
        x=input(),y=input(),k=input();
        printf("%d\n",v[query(1,n,rt[x-1],rt[y],k)-1]);
    }

    return 0;
}
```

2.6 segment tree beats!

2.6.1 区间取最小值

给一个数组，m 次操作:

1:l r x，将 a[i](l<=i<=r)=min(a[i],x)

2:l r，求区间最大值。

3:l r，求区间和。

维护一个最大值 mx 和次大值 se，分类讨论：

当 mx<=x 显然没有影响。

当 se<x<mx 打标记修改区间。

否则暴力递归两个儿子。

```
#include <bits/stdc++.h>

using namespace std;

#define ll long long
ll input(){
    ll x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

#define ms(a,b) memset(a,b,sizeof(a))
#define lson l, m, rt << 1
#define rson m + 1, r, rt << 1 | 1

const int maxn = 1000000 + 5;

int n, m, a[maxn], mx[maxn << 2], cnt[maxn << 2], se[maxn << 2], add[maxn << 2];
ll sum[maxn << 2];

void pushup(int rt){
    sum[rt] = sum[rt << 1] + sum[rt << 1 | 1];
```

```
    if (mx[rt << 1] == mx[rt << 1 | 1]){
        se[rt] = max(se[rt << 1], se[rt << 1 | 1]);
        mx[rt] = mx[rt << 1];
        cnt[rt] = cnt[rt << 1] + cnt[rt << 1 | 1];
    }
    else if (mx[rt << 1] < mx[rt << 1 | 1]){
        se[rt] = max(mx[rt << 1], se[rt << 1 | 1]);
        mx[rt] = mx[rt << 1 | 1];
        cnt[rt] = cnt[rt << 1 | 1];
    }
    else {
        se[rt] = max(mx[rt << 1 | 1], se[rt << 1]);
        mx[rt] = mx[rt << 1];
        cnt[rt] = cnt[rt << 1];
    }
}
void pushdown(int rt){
    if (add[rt] == -1) return;
    // 区间覆盖
    int& t = add[rt]; int ls = rt << 1, rs = rt << 1 | 1;
    if (mx[ls] > t && t > se[ls]) {
        sum[ls] -= 1ll * (mx[ls] - t) * cnt[ls];
        add[ls] = mx[ls] = t;
    }
    if (mx[rs] > t && t > se[rs]){
        sum[rs] -= 1ll * (mx[rs] - t) * cnt[rs];
        add[rs] = mx[rs] = t;
    }
    t = -1;
}
void build(int l, int r, int rt){
    add[rt] = -1;
    if (l == r){
        se[rt] = -1; cnt[rt] = 1;
        sum[rt] = mx[rt] = a[l];
        return;
    }
    int m = l + r >> 1;
    build(lson); build(rson);
    pushup(rt);
}
void update(int L, int R, int t, int l, int r, int rt){
    if (mx[rt] <= t) return;
    if (L <= l && r <= R && se[rt] < t){
        sum[rt] -= 1ll * (mx[rt] - t) * cnt[rt];
        mx[rt] = add[rt] = t;
        return;
    }
    int m = l + r >> 1;
    pushdown(rt);
    if (L <= m) update(L, R, t, lson);
    if (R > m) update(L, R, t, rson);
    pushup(rt);
```

```
}
int qmax(int L, int R, int l, int r, int rt){
    if (L <= l && r <= R) return mx[rt];
    int m = l + r >> 1; int ans = 0;
    pushdown(rt);
    if (L <= m) ans = max(ans, qmax(L, R, lson));
    if (R > m) ans = max(ans, qmax(L, R, rson));
    return ans;
}
ll qsum(int L, int R, int l, int r, int rt){
    if (L <= l && r <= R) return sum[rt];
    int m = l + r >> 1; ll ans = 0;
    pushdown(rt);
    if (L <= m) ans += qsum(L, R, lson);
    if (R > m) ans += qsum(L, R, rson);
    return ans;
}

int main(){
    int T; scanf("%d", &T);
    while (T--){
        scanf("%d%d", &n, &m);
        for (int i = 1; i <= n; i++) scanf("%d", a + i);
        build(1, n, 1); int op, x, y, t;
        while (m--){
            scanf("%d", &op);
            if (op == 0){
                scanf("%d%d%d", &x, &y, &t);
                update(x, y, t, 1, n, 1);
            }
            if (op == 1){
                scanf("%d%d", &x, &y);
                printf("%d\n", qmax(x, y, 1, n, 1));
            }
            if (op == 2){
                scanf("%d%d", &x, &y);
                printf("%lld\n", qsum(x, y, 1, n, 1));
            }
        }
    }
    return 0;
}
```

2.6.2 区间取 max 取 min

长度为 n 的序列，支持区间加 x、区间对 x 取 max、区间对 x 取 min、求区间和、求区间最大值、求区间最小值

```
#include <cstdio>
#include <iostream>
using namespace std;

```

```cpp
int inline rd() {
  register char act = 0;
  register int f = 1, x = 0;
  while (act = getchar(), act < '0' && act != '-')
    ;
  if (act == '-') f = -1, act = getchar();
  x = act - '0';
  while (act = getchar(), act >= '0') x = x * 10 + act - '0';
  return x * f;
}

const int N = 5e5 + 5, SZ = N << 2, INF = 0x7fffffff;

int n, m;
int a[N];

struct data {
  int mx, mx2, mn, mn2, cmx, cmn, tmx, tmn, tad;
  long long sum;
};
data t[SZ];

void pushup(int u) {
  const int lu = u << 1, ru = u << 1 | 1;
  t[u].sum = t[lu].sum + t[ru].sum;
  if (t[lu].mx == t[ru].mx) {
    t[u].mx = t[lu].mx, t[u].cmx = t[lu].cmx + t[ru].cmx;
    t[u].mx2 = max(t[lu].mx2, t[ru].mx2);
  } else if (t[lu].mx > t[ru].mx) {
    t[u].mx = t[lu].mx, t[u].cmx = t[lu].cmx;
    t[u].mx2 = max(t[lu].mx2, t[ru].mx);
  } else {
    t[u].mx = t[ru].mx, t[u].cmx = t[ru].cmx;
    t[u].mx2 = max(t[lu].mx, t[ru].mx2);
  }
  if (t[lu].mn == t[ru].mn) {
    t[u].mn = t[lu].mn, t[u].cmn = t[lu].cmn + t[ru].cmn;
    t[u].mn2 = min(t[lu].mn2, t[ru].mn2);
  } else if (t[lu].mn < t[ru].mn) {
    t[u].mn = t[lu].mn, t[u].cmn = t[lu].cmn;
    t[u].mn2 = min(t[lu].mn2, t[ru].mn);
  } else {
    t[u].mn = t[ru].mn, t[u].cmn = t[ru].cmn;
    t[u].mn2 = min(t[lu].mn, t[ru].mn2);
  }
}
void push_add(int u, int l, int r, int v) {
  // 更新加法标记的同时，更新 $\min$ 和 $\max$ 标记
  t[u].sum += (r - l + 1ll) * v;
  t[u].mx += v, t[u].mn += v;
  if (t[u].mx2 != -INF) t[u].mx2 += v;
  if (t[u].mn2 != INF) t[u].mn2 += v;
  if (t[u].tmx != -INF) t[u].tmx += v;
```

```
58  if (t[u].tmn != INF) t[u].tmn += v;
59  t[u].tad += v;
60 }
61 void push_min(int u, int tg) {
62  // 注意比较 $\max$ 标记
63  if (t[u].mx <= tg) return;
64  t[u].sum += (tg * 1ll - t[u].mx) * t[u].cmx;
65  if (t[u].mn2 == t[u].mx) t[u].mn2 = tg; // !!!
66  if (t[u].mn == t[u].mx) t[u].mn = tg; // !!!!!
67  if (t[u].tmx > tg) t[u].tmx = tg; // 更新取 $\max$ 标记
68  t[u].mx = tg, t[u].tmn = tg;
69 }
70 void push_max(int u, int tg) {
71  if (t[u].mn > tg) return;
72  t[u].sum += (tg * 1ll - t[u].mn) * t[u].cmn;
73  if (t[u].mx2 == t[u].mn) t[u].mx2 = tg;
74  if (t[u].mx == t[u].mn) t[u].mx = tg;
75  if (t[u].tmn < tg) t[u].tmn = tg;
76  t[u].mn = tg, t[u].tmx = tg;
77 }
78 void pushdown(int u, int l, int r) {
79  const int lu = u << 1, ru = u << 1 | 1, mid = (l + r) >> 1;
80  if (t[u].tad)
81    push_add(lu, l, mid, t[u].tad), push_add(ru, mid + 1, r, t[u].tad);
82  if (t[u].tmx != -INF) push_max(lu, t[u].tmx), push_max(ru, t[u].tmx);
83  if (t[u].tmn != INF) push_min(lu, t[u].tmn), push_min(ru, t[u].tmn);
84  t[u].tad = 0, t[u].tmx = -INF, t[u].tmn = INF;
85 }
86 void build(int u = 1, int l = 1, int r = n) {
87  t[u].tmn = INF, t[u].tmx = -INF; // 取极限
88  if (l == r) {
89    t[u].sum = t[u].mx = t[u].mn = a[l];
90    t[u].mx2 = -INF, t[u].mn2 = INF;
91    t[u].cmx = t[u].cmn = 1;
92    return;
93  }
94  int mid = (l + r) >> 1;
95  build(u << 1, l, mid), build(u << 1 | 1, mid + 1, r);
96  pushup(u);
97 }
98 void add(int L, int R, int v, int u = 1, int l = 1, int r = n) {
99  if (R < l || r < L) return;
100  if (L <= l && r <= R) return push_add(u, l, r, v); // !!! 忘 return
101  int mid = (l + r) >> 1;
102  pushdown(u, l, r);
103  add(L, R, v, u << 1, l, mid), add(L, R, v, u << 1 | 1, mid + 1, r);
104  pushup(u);
105 }
106 void tomin(int L, int R, int v, int u = 1, int l = 1, int r = n) {
107  if (R < l || r < L || t[u].mx <= v) return;
108  if (L <= l && r <= R && t[u].mx2 < v) return push_min(u, v); // BUG: 忘了返回
109  int mid = (l + r) >> 1;
110  pushdown(u, l, r);
```

```
111   tomin(L, R, v, u << 1, l, mid), tomin(L, R, v, u << 1 | 1, mid + 1, r);
112   pushup(u);
113 }
114 void tomax(int L, int R, int v, int u = 1, int l = 1, int r = n) {
115   if (R < l || r < L || t[u].mn >= v) return;
116   if (L <= l && r <= R && t[u].mn2 > v) return push_max(u, v);
117   int mid = (l + r) >> 1;
118   pushdown(u, l, r);
119   tomax(L, R, v, u << 1, l, mid), tomax(L, R, v, u << 1 | 1, mid + 1, r);
120   pushup(u);
121 }
122 long long qsum(int L, int R, int u = 1, int l = 1, int r = n) {
123   if (R < l || r < L) return 0;
124   if (L <= l && r <= R) return t[u].sum;
125   int mid = (l + r) >> 1;
126   pushdown(u, l, r);
127   return qsum(L, R, u << 1, l, mid) + qsum(L, R, u << 1 | 1, mid + 1, r);
128 }
129 long long qmax(int L, int R, int u = 1, int l = 1, int r = n) {
130   if (R < l || r < L) return -INF;
131   if (L <= l && r <= R) return t[u].mx;
132   int mid = (l + r) >> 1;
133   pushdown(u, l, r);
134   return max(qmax(L, R, u << 1, l, mid), qmax(L, R, u << 1 | 1, mid + 1, r));
135 }
136 long long qmin(int L, int R, int u = 1, int l = 1, int r = n) {
137   if (R < l || r < L) return INF;
138   if (L <= l && r <= R) return t[u].mn;
139   int mid = (l + r) >> 1;
140   pushdown(u, l, r);
141   return min(qmin(L, R, u << 1, l, mid), qmin(L, R, u << 1 | 1, mid + 1, r));
142 }
143 int main() {
144   n = rd();
145   for (int i = 1; i <= n; i++) a[i] = rd();
146   build();
147   m = rd();
148   for (int i = 1; i <= m; i++) {
149     int op, l, r, x;
150     op = rd(), l = rd(), r = rd();
151     if (op <= 3) x = rd(); // scanf("%d",&x);
152     if (op == 1)
153       add(l, r, x);
154     else if (op == 2)
155       tomax(l, r, x);
156     else if (op == 3)
157       tomin(l, r, x);
158     else if (op == 4)
159       printf("%lld\n", qsum(l, r));
160     else if (op == 5)
161       printf("%lld\n", qmax(l, r));
162     else
163       printf("%lld\n", qmin(l, r));
```

```
164   }
165   return 0;
166 }
```

2.6.3 区间历史最值

$Q\ X\ Y$：询问从 X 到 Y 的当前最大值

$A\ X\ Y$：询问从 X 到 Y 的历史最大值（出现过的最大数）

$P\ X\ Y\ Z$：将 X 到 Y 这段区间加 Z

$C\ X\ Y\ Z$：将 X 到 Y 这段区间赋值为 Z

```
1  #include<cstdio>
2  const int MAXN = 1e5 + 5;
3  const int inf = 0x7fffffff;
4
5  inline int max(int a,int b){ return a>b? a: b;}
6  inline void chk_max(int &a,int b){ if(a<b) a=b;}
7
8  int ans[MAXN<<2],max_ans[MAXN<<2];//区间最大值 区间历史最大值
9
10 bool vis[MAXN<<2];//是否进行过区间赋值操作
11 int sum[MAXN<<2], val[MAXN<<2];//上次下放之后的加和 上次下放之后的赋值操作 （赋值之后的加
       操作一并算入赋值，下同）
12 int max_sum[MAXN<<2], max_val[MAXN<<2];//上次下放之后达到的最大加和 上次下放之后赋的最大
       值
13 /*
14 注意这里实际上使用了4个lazy_tag，不过如果您理解了算法的核心，您应该知道我在干什么
15 */
16 #define ls(u) ((u)<<1)
17 #define rs(u) ((u)<<1|1)//个人习惯，左右儿子
18
19
20 inline void push_up(int u)//push_up比较简单，就是用左右儿子的答案更新本节点的答案
21 {
22    ans[u] = max(ans[ls(u)], ans[rs(u)]);
23    max_ans[u] = max(max_ans[ls(u)], max_ans[rs(u)]);
24 }
25
26
27 //我将更新lazy_tag的操作打包成函数，不然有可能在细节上出错（其实我一开始就是在这里出的错）
28 /*
29 更新加操作的tag，分为两种情况：
30  1.赋过值，算作赋值操作
31  2.没赋过值
32 */
33 inline void do_sum(int u,int k,int max_k)//max_k表示父节点在上一次push_down之后达到的
       最大加和
34 {
35    if(vis[u])
36    {
37       chk_max(max_val[u], val[u]+max_k);
```

```
38          chk_max(max_ans[u], ans[u]+max_k);
39          ans[u]+=k;
40          val[u]+=k;
41      }
42      else
43      {
44          chk_max(max_sum[u], sum[u]+max_k);
45          chk_max(max_ans[u], ans[u]+max_k);
46          ans[u]+=k;
47          sum[u]+=k;
48      }
49  }
50
51  inline void do_val(int u,int k,int max_k)//max_k表示父节点在上一次push_down之后达到的
        最大赋值
52  {
53      if(vis[u])
54      {
55          chk_max(max_val[u], max_k);
56          chk_max(max_ans[u], max_k);
57      }
58      else
59      {
60          vis[u]=1;//该点标记为赋过值
61          max_val[u] = max_k;
62          chk_max(max_ans[u], max_k);
63      }
64      ans[u] = val[u] = k;
65  }
66
67  inline void push_down(int u)
68  {
69      do_sum(ls(u),sum[u],max_sum[u]);
70      do_sum(rs(u),sum[u],max_sum[u]);//先传递和
71
72      sum[u] = max_sum[u] = 0;//下传之后清零
73
74      if(vis[u])
75      {
76          do_val(ls(u),val[u],max_val[u]);
77          do_val(rs(u),val[u],max_val[u]);
78
79          vis[u] = 0;
80          val[u] = max_val[u] = 0;//下传之后清零
81      }
82  }
83
84  void build(int u,int l,int r)//建树
85  {
86      if(l==r)
87      {
88          scanf("%d",&ans[u]);
89          max_ans[u]=ans[u];
```

```
90          return;
91      }
92
93      int mid=(l+r)>>1;
94      build(ls(u),l,mid);
95      build(rs(u),mid+1,r);
96      push_up(u);
97  }
98
99  int query(int u,int l,int r, int ql,int qr)//Q操作
100 {
101     if(ql<=l && r<=qr) return ans[u];
102
103     push_down(u);
104     int mid=(l+r)>>1, ret=-inf;
105     if(ql<=mid) ret = query(ls(u),l,mid, ql,qr);
106     if(mid<qr) chk_max(ret, query(rs(u),mid+1,r, ql,qr));
107
108     return ret;
109 }
110
111 int query_max(int u,int l,int r, int ql,int qr)//A操作，与Q类似
112 {
113     if(ql<=l && r<=qr) return max_ans[u];
114
115     push_down(u);
116     int mid=(l+r)>>1, ret=-inf;
117     if(ql<=mid) ret = query_max(ls(u),l,mid, ql,qr);
118     if(mid<qr) chk_max(ret, query_max(rs(u),mid+1,r, ql,qr));
119
120     return ret;
121 }
122
123 void add(int u,int l,int r, int ql,int qr,int k)//P操作
124 {
125     if(ql<=l && r<=qr)
126     {
127         do_sum(u,k,k);//这里max_k也填k就行了
128         return;
129     }
130
131     push_down(u);
132     int mid = (l+r)>>1;
133     if(ql<=mid) add(ls(u),l,mid, ql,qr,k);
134     if(mid<qr) add(rs(u),mid+1,r, ql,qr,k);
135     push_up(u);
136 }
137
138 void assign(int u,int l,int r, int ql,int qr,int k)//C操作
139 {
140     if(ql<=l && r<=qr)
141     {
142         do_val(u,k,k);
```

```
143         return;
144     }
145 
146     push_down(u);
147     int mid = (l+r)>>1;
148     if(ql<=mid) assign(ls(u),l,mid, ql,qr,k);
149     if(mid<qr) assign(rs(u),mid+1,r, ql,qr,k);
150     push_up(u);
151 }
152 
153 //这两个函数是调试用的，用于打印序列
154 void print(int u,int l,int r)
155 {
156     if(l==r)
157     {
158         printf("%d ",ans[u]);
159         return;
160     }
161     push_down(u);
162     int mid=(l+r)>>1;
163     print(ls(u),l,mid);
164     print(rs(u),mid+1,r);
165 }
166 inline void test(int t)
167 {
168     printf("========================================\n");
169     print(1,1,t);
170     printf("\n========================================\n");
171 }
172 
173 int main(void)
174 {
175     int t;
176     scanf("%d",&t);
177     build(1,1,t);
178 
179     int e;
180     scanf("%d",&e);
181     while(e--)
182     {
183         char c=getchar();
184         while(c!='Q' && c!='A' && c!='P' && c!='C') c=getchar();//个人习惯，感觉这么写
              比较保险
185         int x,y;
186         scanf("%d%d",&x,&y);
187 
188         if(c=='Q') printf("%d\n",query(1,1,t, x,y));
189         if(c=='A') printf("%d\n",query_max(1,1,t, x,y));
190         if(c=='P')
191         {
192             int z;
193             scanf("%d",&z);
194             add(1,1,t, x,y,z);
```

```
195             //test(t);
196         }
197         if(c=='C')
198         {
199             int z;
200             scanf("%d",&z);
201             assign(1,1,t, x,y,z);
202             //test(t);
203         }
204     }
205     return 0;
206 }
```

2.7 权值 splay

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  int input(){
6      int x=0,f=0;char ch=getchar();
7      while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
8      while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
9      return f? -x:x;
10 }
11
12 #define N 500007
13 #define debug printf("pass in line:%d\n",__LINE__)
14
15 class splay{
16     public:
17         struct node{
18             int ch[2];
19             int fa,cnt,val,siz;
20         }t[N];
21
22         int rt;
23
24         splay(){
25             tot=0;
26             rt=0;
27         }
28
29         void Splay(int x,int goal){
30             while(t[x].fa!=goal){
31                 int y=t[x].fa,z=t[y].fa;
32                 if(z!=goal) (t[y].ch[0]==x)^(t[z].ch[0]==y)? Rot(x):Rot(y);
33                 Rot(x);
34             }
35             if(!goal) rt=x;
36         }
37
```

```
38      void Ins(int x){
39         int u=rt,fa=0;
40         while(u&&t[u].val!=x) fa=u,u=t[u].ch[x>t[u].val];
41         if(u) t[u].cnt++;
42         else{
43            u=++tot;
44            if(fa) t[fa].ch[x>t[fa].val]=u;
45            t[tot].ch[0]=0,t[tot].ch[1]=0;
46            t[tot].fa=fa,t[tot].val=x;
47            t[tot].cnt=1,t[tot].siz=1;
48         }
49         Splay(u,0);
50      }
51
52      int find(int x){
53         Find(x);
54         return t[t[rt].ch[0]].siz;
55      }
56
57      int Next(int x,int f){
58         Find(x);
59         int u=rt;
60         if((t[u].val>x&&f)||(t[u].val<x&&!f)) return u;
61         u=t[u].ch[f];
62         while(t[u].ch[f^1]) u=t[u].ch[f^1];
63         return u;
64      }
65
66      void Del(int x){
67         int last=Next(x,0),next=Next(x,1);
68         Splay(last,0),Splay(next,last);
69         int del=t[next].ch[0];
70         if(t[del].cnt>1){
71            t[del].cnt--;
72            Splay(del,0);
73         }else t[next].ch[0]=0;
74      }
75
76      int K_th(int x){
77          int u=rt;
78          if(t[u].siz<x) return 0;
79          while(1){
80             int y=t[u].ch[0];
81             if(x>t[y].siz+t[u].cnt) x-=t[y].siz+t[u].cnt,u=t[u].ch[1];
82             else if(t[y].siz>=x) u=y;
83             else return t[u].val;
84          }
85      }
86   private:
87      int tot;
88
89      void Push_up(int u){
90         t[u].siz=t[t[u].ch[0]].siz+t[t[u].ch[1]].siz+t[u].cnt;
```

```
91        }
92
93        void Rot(int x){
94           int y=t[x].fa,z=t[y].fa,k=t[y].ch[1]==x;
95           t[z].ch[t[z].ch[1]==y]=x,t[x].fa=z;
96           t[y].ch[k]=t[x].ch[k^1],t[t[x].ch[k^1]].fa=y;
97           t[x].ch[k^1]=y;t[y].fa=x;
98           Push_up(y),Push_up(x);
99        }
100
101        void Find(int x){
102           int u=rt;
103           if(!u) return;
104           while(t[u].ch[x>t[u].val]&&x!=t[u].val) u=t[u].ch[x>t[u].val];
105           Splay(u,0);
106        }
107 };
108
109 splay Splay;
110 int n;
111
112 int main(){
113    n=input();
114    Splay.Ins(-2147483647);
115    Splay.Ins(2147483647);
116    while(n--){
117       int opt=input();
118       if(opt==1) Splay.Ins(input());
119       else if(opt==2) Splay.Del(input());
120       else if(opt==3) printf("%d\n",Splay.find(input()));
121       else if(opt==4) printf("%d\n",Splay.K_th(input()+1));
122       else if(opt==5) printf("%d\n",Splay.t[Splay.Next(input(),0)].val);
123       else if(opt==6) printf("%d\n",Splay.t[Splay.Next(input(),1)].val);
124    }
125 }
```

2.8 ST 表（RMQ）

```
1  #include<cstdio>
2  #include<cstring>
3  #include<algorithm>
4  #define M 25
5  #define N 100005
6  using namespace std;
7  int n,m;
8  int a[N],Log[N];
9  int f[N][M];
10 void GetLog()
11 {
12    int i;
13    Log[1]=0;
14    for(i=2;i<=n+1;++i)
```

```
    Log[i]=Log[i/2]+1;
}
void RMQ()
{
   int i,j;
   for(i=1;i<=n;++i)
     f[i][0]=a[i];
   for(j=1;(1<<j)<=n;++j)
     for(i=1;i+(1<<(j-1))<=n;++i)
       f[i][j]=max(f[i][j-1],f[i+(1<<(j-1))][j-1]);
}
int main()
{
   int l,r,i,k,ans;
   scanf("%d%d",&n,&m);
   for(i=1;i<=n;++i)
     scanf("%d",&a[i]);
   GetLog();
   RMQ();
   for(i=1;i<=m;++i)
   {
      scanf("%d%d",&l,&r);
      k=Log[r-l+1];
      ans=max(f[l][k],f[r-(1<<k)+1][k]);
      printf("%d\n",ans);
   }
   return 0;
}
```

2.9 bitset

```
//foo.size() 返回大小（位数）
//foo.count() 返回1的个数
//foo.any() 返回是否有1
//foo.none() 返回是否没有1
//foo.set() 全都变成1
//foo.set(p) 将第p + 1位变成1
//foo.set(p, x) 将第p + 1位变成x
//foo.reset() 全都变成0
//foo.reset(p) 将第p + 1位变成0
//foo.flip() 全都取反
//foo.flip(p) 将第p + 1位取反
//foo.to_ulong() 返回它转换为unsigned long的结果，如果超出范围则报错
//foo.to_ullong() 返回它转换为unsigned long long的结果，如果超出范围则报错
//foo.to_string() 返回它转换为string的结果
// #include <bits/stdc++.h>
#include <iostream>
#include <cstdio>
#include <bitset>

using namespace std;

```

```
#define ll long long
ll input(){
    ll x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

bitset<10007> bs[1007];
int n;

int main(){
    n=input();
    for(int i=1;i<=n;i++){
        int m=input();
        for(int j=1;j<=m;j++){
            bs[i][input()]=1;
        }
    }

    int Q=input();

    for(int i=1;i<=Q;i++){
        int x=input(),y=input();
        bool f=0;
        for(int j=1;j<=n;j++){
            if(bs[j][x]&&bs[j][y]){printf("Yes\n");f=1;break;}
        }
        if(!f) printf("No\n");
    }
    return 0;
}
```

2.10 hash_map

```
const ll mod=1e9+7;
const int maxsz=3e5+7;
template<typename key,typename val>
class hash_map{public:
  struct node{key u;val v;int next;};
  vector<node> e;
  int head[maxsz],nume,numk,id[maxsz];
  int geths(PII &u){
    int x=(1ll*u.fr*mod+u.sc)%maxsz;
    if(x<0) return x+maxsz;
    return x;
  }
  val& operator[](key u){
    int hs=geths(u);
    for(int i=head[hs];i;i=e[i].next)if(e[i].u==u) return e[i].v;
    if(!head[hs])id[++numk]=hs;
    if(++nume>=e.size())e.resize(nume<<1);
```

```
    return e[nume]=(node){u,0,head[hs]},head[hs]=nume,e[nume].v;
  }
  void clear(){
    for(int i=0;i<=numk;i++) head[id[i]]=0;
    numk=nume=0;
  }
};
hash_map<PII,int> G;
//pair 的 hash_map

truct hash_map { // 哈希表模板
  struct data {
    long long u;
    int v, nex;
  }; // 前向星结构
  data e[SZ << 1]; // SZ 是 const int 表示大小
  int h[SZ], cnt;
  int hash(long long u) { return u % SZ; }
  int& operator[](long long u) {
    int hu = hash(u); // 获取头指针
    for (int i = h[hu]; i; i = e[i].nex)
      if (e[i].u == u) return e[i].v;
    return e[++cnt] = (data){u, -1, h[hu]}, h[hu] = cnt, e[cnt].v;
  }
  hash_map() {
    cnt = 0;
    memset(h, 0, sizeof(h));
  }
};
```

2.11 单调栈

```
//单调递增栈
void push(int x){
   while(!s.empty()&&s.top()>x) stk.pop();
   s.push(A[i]);
}
//单调递减栈
void push(int x){
   while(!s.empty()&&s.top()<=x) stk.pop();
   s.push(A[i]);
}
```

2.12 单调队列

```
deque<int> q;

void push(int x){
   if(q.empty()) q.push_back(x);
   else if(q.back()>x){
      while((!q.empty())&&q.back()>x) q.pop_back();
```

```
        q.push_back(x);
    }else q.push_back(x);
}
```

2.13 二维区间最值

```
//二维区间最值
#include <bits/stdc++.h>

using namespace std;

#define ll long long
ll input(){
    ll x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

const int N=1007;

int h[N][N],sum[N][N],mind[N][N];
int n,m,q;

int Solve(int a,int b){
    int tmp[N];
    for(int j=1;j<=m;j++){
        int l=1,r=0;
        for(int i=1;i<a;i++){
            while(r>=l&&h[tmp[r]][j]>=h[i][j]) r--;
            tmp[++r]=i;
        }

        for(int i=1;i+a-1<=n;i++){
            while(r>=l&&h[tmp[r]][j]>=h[i+a-1][j])r--;
            tmp[++r]=i+a-1;
            while(tmp[l]<i) l++;
            mind[i][j]=h[tmp[l]][j];
        }
    }

    int Ans=1e9;
    for(int i=1;i+a-1<=n;i++){
        int l=1,r=0;

        for(int j=1;j<b;j++){
            while(r>=l&&mind[i][tmp[r]]>=mind[i][j]) r--;
            tmp[++r]=j;
        }

        for(int j=1;j+b-1<=m;j++){
            while(r>=l&&mind[i][tmp[r]]>=mind[i][j+b-1]) r--;
```

```
            tmp[++r]=j+b-1;
            while(tmp[l]<j) l++;
            int tmin=mind[i][tmp[l]];
            int ii=i+a-1,jj=j+b-1;
            int tsum=sum[ii][jj]-sum[ii][j-1]-sum[i-1][jj]+sum[i-1][j-1];
            Ans=min(Ans,tsum-tmin*a*b);
        }
    }
    return Ans;
}

int main(){
    n=input(),m=input();

    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            h[i][j]=input();
            sum[i][j]=sum[i-1][j]+sum[i][j-1]-sum[i-1][j-1]+h[i][j];
        }
    }

    q=input();
    for(int i=1;i<=q;i++){
        int a=input(),b=input();
        printf("%d\n",Solve(a,b));
    }
    return 0;
}
```

2.14 Treap

2.14.1 权值 Teeap

```
#include <bits/stdc++.h>

using namespace std;

#define ll long long
ll input(){
    ll x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

const int N=1e5+7;

#define ls(x) t[x].ch[0]
#define rs(x) t[x].ch[1]

struct node{
    int ch[2],val,key,size;
```

```
}t[N];
int tot;

void put(int rt){
   t[rt].size=t[ls(rt)].size+t[rs(rt)].size+1;
}

void split(int rt,int &x,int &y,int val){
   if(!rt){x=y=0;return;}
   if(t[rt].val<=val) x=rt,split(rs(rt),rs(x),y,val);
   else y=rt,split(ls(rt),x,ls(y),val);
   put(rt);
}

void merge(int &rt,int x,int y){
   if(!x||!y){rt=x+y;return;}
   if(t[x].key<t[y].key) rt=x,merge(rs(rt),rs(x),y);
   else rt=y,merge(ls(rt),x,ls(y));
   put(rt);
}

void Ins(int &rt,int val){
   int x=0,y=0,z=++tot;
   t[z].size=1,t[z].val=val,t[z].key=rand();
   split(rt,x,y,val);
   merge(x,x,z);
   merge(rt,x,y);
}

void erase(int &rt,int val){
   int x=0,y=0,z=0;
   split(rt,x,y,val);
   split(x,x,z,val-1);
   merge(z,ls(z),rs(z));
   merge(x,x,z);
   merge(rt,x,y);
}

int k_th(int rt,int k){
   while(t[ls(rt)].size+1!=k){
      if(t[ls(rt)].size>=k) rt=ls(rt);
      else k-=(t[ls(rt)].size+1),rt=rs(rt);
   }
   return t[rt].val;
}

int get_rank(int &rt,int val){
   int x=0,y=0,res;
   split(rt,x,y,val-1);
   res=t[x].size+1;
   merge(rt,x,y);
   return res;
}
```

```
73
74  int pre(int &rt,int val){
75      int x=0,y=0,res;
76      split(rt,x,y,val-1);
77      res=k_th(x,t[x].size);
78      merge(rt,x,y);
79      return res;
80  }
81
82  int suf(int &rt,int val){
83      int x=0,y=0,res;
84      split(rt,x,y,val);
85      res=k_th(y,1);
86      merge(rt,x,y);
87      return res;
88  }
89
90  int n,rt;
91
92  int main(){
93      srand(19260817);
94      n=input();
95      for(int i=1;i<=n;i++){
96          int opr=input(),x=input();
97          if(opr==1) Ins(rt,x);
98          if(opr==2) erase(rt,x);
99          if(opr==3) printf("%d\n",get_rank(rt,x));
100         if(opr==4) printf("%d\n",k_th(rt,x));
101         if(opr==5) printf("%d\n",pre(rt,x));
102         if(opr==6) printf("%d\n",suf(rt,x));
103     }
104 }
```

2.14.2 区间翻转 Treap

```
1  //区间翻转
2  #include <bits/stdc++.h>
3
4  using namespace std;
5
6  #define ll long long
7  ll input(){
8      ll x=0,f=0;char ch=getchar();
9      while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
10     while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
11     return f? -x:x;
12 }
13
14 const int N=1e5+7;
15
16 #define ls(x) t[x].ch[0]
17 #define rs(x) t[x].ch[1]
```

```
18
19  struct node{
20      int ch[2],val,key,size;
21      bool rev;
22  }t[N];
23  int tot;
24
25  int newnode(int val){
26      t[++tot].val=val;
27      t[tot].key=rand();
28      t[tot].size=1;
29      t[tot].rev=0;
30      return tot;
31  }
32
33  void put(int rt){
34      t[rt].size=t[ls(rt)].size+t[rs(rt)].size+1;
35  }
36
37  void putdown(int rt){
38      swap(ls(rt),rs(rt));
39      t[ls(rt)].rev^=1;
40      t[rs(rt)].rev^=1;
41      t[rt].rev=0;
42  }
43
44  void split(int rt,int &x,int &y,int siz){
45      if(!rt){x=y=0;return;}
46      if(t[rt].rev) putdown(rt);
47      if(t[ls(rt)].size<siz) x=rt,split(rs(rt),rs(x),y,siz-t[ls(rt)].size-1);
48      else y=rt,split(ls(rt),x,ls(y),siz);
49      put(rt);
50  }
51
52  void merge(int &rt,int x,int y){
53      if(!x||!y){rt=x+y;return;}
54      if(t[x].key<t[y].key){
55          if(t[x].rev) putdown(x);
56          rt=x,merge(rs(rt),rs(x),y);
57      }
58      else{
59          if(t[y].rev) putdown(y);
60          rt=y,merge(ls(rt),x,ls(y));
61      }
62      put(rt);
63  }
64
65  void reverse(int &rt,int l,int r){
66      int x=0,y=0,z=0;
67      split(rt,x,y,l-1);
68      split(y,y,z,r-l+1);
69      t[y].rev^=1;
70      merge(y,y,z);
```

```
71      merge(rt,x,y);
72  }
73
74  void dfs(int now){
75      if(!now) return;
76      if(t[now].rev) putdown(now);
77      dfs(ls(now));
78      printf("%d ",t[now].val);
79      dfs(rs(now));
80  }
81
82  int n,m,rt;
83
84  int main(){
85      srand(19260817);
86      n=input(),m=input();
87      for(int i=1;i<=n;i++){
88          merge(rt,rt,newnode(i));
89      }
90
91      for(int i=1;i<=m;i++){
92          int l=input(),r=input();
93          reverse(rt,l,r);
94      }
95      dfs(rt);
96  }
```

2.14.3 线段树套 Treap

```
1   #include <bits/stdc++.h>
2
3   using namespace std;
4
5   #define ll long long
6   ll input(){
7       ll x=0,f=0;char ch=getchar();
8       while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
9       while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
10      return f? -x:x;
11  }
12
13  const int N=5e4+7;
14  const int INF=0x7fffffff;
15
16  #define ls(x) t[x].ch[0]
17  #define rs(x) t[x].ch[1]
18
19  struct node{
20      int ch[2],val,key,size;
21  }t[N*25];
22  int tot;
23
```

```
24 void put(int rt){
25    t[rt].size=t[ls(rt)].size+t[rs(rt)].size+1;
26 }
27
28 void split(int rt,int &x,int &y,int val){
29    if(!rt){x=y=0;return;}
30    if(t[rt].val<=val) x=rt,split(rs(rt),rs(x),y,val);
31    else y=rt,split(ls(rt),x,ls(y),val);
32    put(rt);
33 }
34
35 void merge(int &rt,int x,int y){
36    if(!x||!y){rt=x+y;return;}
37    if(t[x].key<t[y].key) rt=x,merge(rs(rt),rs(x),y);
38    else rt=y,merge(ls(rt),x,ls(y));
39    put(rt);
40 }
41
42 void Ins(int &rt,int val){
43    int x=0,y=0,z=++tot;
44    t[z].size=1,t[z].val=val,t[z].key=rand();
45    split(rt,x,y,val);
46    merge(x,x,z);
47    merge(rt,x,y);
48 }
49
50 void erase(int &rt,int val){
51    int x=0,y=0,z=0;
52    split(rt,x,y,val);
53    split(x,x,z,val-1);
54    merge(z,ls(z),rs(z));
55    merge(x,x,z);
56    merge(rt,x,y);
57 }
58
59 int k_th(int rt,int k){
60    while(t[ls(rt)].size+1!=k){
61       if(t[ls(rt)].size>=k) rt=ls(rt);
62       else k-=(t[ls(rt)].size+1),rt=rs(rt);
63    }
64    return t[rt].val;
65 }
66
67 int get_rank(int &rt,int val){
68    int x=0,y=0,res;
69    split(rt,x,y,val-1);
70    res=t[x].size;
71    merge(rt,x,y);
72    return res;
73 }
74
75 int pre(int &rt,int val){
76    int x=0,y=0,res=-INF;
```

```
77      split(rt,x,y,val-1);
78      if(t[x].size) res=k_th(x,t[x].size);
79      merge(rt,x,y);
80      return res;
81  }
82
83  int suf(int &rt,int val){
84      int x=0,y=0,res=INF;
85      split(rt,x,y,val);
86      if(t[y].size)res=k_th(y,1);
87      merge(rt,x,y);
88      return res;
89  }
90
91  int T[N<<2],a[N];
92
93  void Add(int rt,int l,int r,int pos,int val){
94      Ins(T[rt],val);
95      if(l==r) return;
96      int mid=(l+r)>>1;
97      if(pos<=mid) Add(rt<<1,l,mid,pos,val);
98      else Add(rt<<1|1,mid+1,r,pos,val);
99  }
100
101 void Del(int rt,int l,int r,int pos,int val){
102     erase(T[rt],val);
103     if(l==r) return;
104     int mid=(l+r)>>1;
105     if(pos<=mid) Del(rt<<1,l,mid,pos,val);
106     else Del(rt<<1|1,mid+1,r,pos,val);
107 }
108
109 int query_rank(int rt,int l,int r,int ql,int qr,int val){
110     if(ql<=l&&r<=qr) return get_rank(T[rt],val);
111     int mid=(l+r)>>1,res=0;
112     if(mid>=ql) res+=query_rank(rt<<1,l,mid,ql,qr,val);
113     if(mid<qr) res+=query_rank(rt<<1|1,mid+1,r,ql,qr,val);
114     return res;
115 }
116
117 int query_pre(int rt,int l,int r,int ql,int qr,int val){
118     if(ql<=l&&r<=qr) return pre(T[rt],val);
119     int mid=(l+r)>>1,res=-INF;
120     if(mid>=ql) res=max(res,query_pre(rt<<1,l,mid,ql,qr,val));
121     if(mid<qr) res=max(res,query_pre(rt<<1|1,mid+1,r,ql,qr,val));
122     return res;
123 }
124
125 int query_suf(int rt,int l,int r,int ql,int qr,int val){
126     if(ql<=l&&r<=qr) return suf(T[rt],val);
127     int mid=(l+r)>>1,res=INF;
128     if(mid>=ql) res=min(res,query_suf(rt<<1,l,mid,ql,qr,val));
129     if(mid<qr) res=min(res,query_suf(rt<<1|1,mid+1,r,ql,qr,val));
```

```
130     return res;
131 }
132 
133 int query_kth(int n,int ql,int qr,int k){
134     int l=0,r=1e8;
135     int res=0;
136     while(l<r){
137         int mid=(l+r+1)>>1;
138         if(query_rank(1,1,n,ql,qr,mid)<k) l=mid;
139         else r=mid-1;
140     }
141     return l;
142 }
143 
144 int n,m,opr,l,r,k;
145 
146 int main(){
147     srand(19260817);
148     n=input(),m=input();
149     for(int i=1;i<=n;i++){
150         a[i]=input();
151         Add(1,1,n,i,a[i]);
152     }
153 
154     for(int i=1;i<=m;i++){
155         opr=input();
156         if(opr==1){
157             l=input(),r=input(),k=input();
158             printf("%d\n",query_rank(1,1,n,l,r,k)+1);
159         }
160         if(opr==2){
161             l=input(),r=input(),k=input();
162             printf("%d\n",query_kth(n,l,r,k));
163         }
164         if(opr==3){
165             l=input(),k=input();
166             Del(1,1,n,l,a[l]);
167             a[l]=k;
168             Add(1,1,n,l,k);
169         }
170         if(opr==4){
171             l=input(),r=input(),k=input();
172             printf("%d\n",query_pre(1,1,n,l,r,k));
173         }
174         if(opr==5){
175             l=input(),r=input(),k=input();
176             printf("%d\n",query_suf(1,1,n,l,r,k));
177         }
178     }
179 }
```

2.15 GSS(最大连续子段和)

2.15.1 GSS1

题目大意：给出一个数列，多次询问区间最长连续子段和题解：线段树维护区间最长连续子段和 gss，区间从最左元素开始的最长连续子段和 lgss 区间以最右元素为结尾的最长连续子段和 rgss 以及区间和 s，信息传递并合并即可

```
#include <bits/stdc++.h>

using namespace std;

#define ll long long
ll input(){
    ll f=0,x=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

const int N=50007;

ll n,m;
ll a[N];
struct node{
    ll mxl,mxr,mxs,sum;
}t[N*4];
node merge(node a,node b){
    node res;
    res.sum=a.sum+b.sum;
    res.mxs=max(a.mxr+b.mxl,max(a.mxs,b.mxs));
    res.mxl=max(a.mxl,a.sum+b.mxl);
    res.mxr=max(b.mxr,b.sum+a.mxr);
    return res;
}

void build(int rt,int l,int r){
    if(l==r){
        t[rt].sum=t[rt].mxl=t[rt].mxr=t[rt].mxs=a[l];
        return;
    }

    int mid=(l+r)>>1;
    build(rt<<1,l,mid),build(rt<<1|1,mid+1,r);
    t[rt]=merge(t[rt<<1],t[rt<<1|1]);
}

node query(int rt,int l,int r,int ql,int qr){
    if(ql<=l&&r<=qr)
        return t[rt];
    node res;
    int mid=(l+r)>>1;
    if(qr<=mid) res=query(rt<<1,l,mid,ql,qr);
```

```
    else if(ql>mid) res=query(rt<<1|1,mid+1,r,ql,qr);
    else res=merge(query(rt<<1,l,mid,ql,qr),query(rt<<1|1,mid+1,r,ql,qr));
    return res;
}

int main(){
    n=input();
    for(int i=1;i<=n;i++) a[i]=input();
    build(1,1,n);
    m=input();
    for(int i=1;i<=m;i++){
        int l=input(),r=input();
        printf("%lld\n",query(1,1,n,l,r).mxs);
    }
}
```

2.15.2 GSS2

题目大意：给出一个数列，多次询问区间最大连续子段和，计数时重复元素只算进子段和中一次
题解：考虑离线对询问进行扫描线，线段树 i 位置表示 i 时间一直到当前的时间节点的累加值，区间最大值 nmx 表示当前时间为终点的后缀最大子段和，记 mx 为区间最大子段和，我们发现区间最大子段和 mx 其实就是后缀最大子段和的历史最大值，每次处理到节点 i，只要在区间加上 a[i]，就能更新所有的后缀和 nmx，对于非重元素，更新区间为 [1,i]，对于重复元素，更新区间为 [lst[a[i]]+1,i], 区间更新需要打 tag，相对的，我们还要维护一个历史最大标记 mxtag，历史最大值的是需要当前的最大值加上下传的历史最大标记来更新的，在历史最大值更新之后才能用 tag 去更新当前的最大值，因为线段树标记发生变化的前提条件是上方没有任何标记，所以上下标记作用的时间域不会有交集，从而能保证正确性。

```
#include <bits/stdc++.h>

using namespace std;

#define ll long long
ll input(){
    ll x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

#define ls rt<<1
#define rs rt<<1|1
#define N 800007

int n,m;
ll a[N],pre[N];
ll Ans[N];

struct seg{
    ll p,s,lazy,tag;
```

```
}t[N];

struct Q{
    int l,r,id;
    friend bool operator <(Q a,Q b){
        return a.r<b.r;
    }
}q[N];

void pushdown(int rt){
    if(!t[rt].tag&&!t[rt].lazy) return;
    t[ls].p=max(t[ls].p,t[ls].s+t[rt].lazy);
    t[ls].lazy=max(t[ls].lazy,t[ls].tag+t[rt].lazy);
    t[ls].s+=t[rt].tag,t[ls].tag+=t[rt].tag;
    t[rs].p=max(t[rs].p,t[rs].s+t[rt].lazy);
    t[rs].lazy=max(t[rs].lazy,t[rs].tag+t[rt].lazy);
    t[rs].s+=t[rt].tag;t[rs].tag+=t[rt].tag;
    t[rt].tag=0,t[rt].lazy=0;
}

void put(int rt){
    t[rt].s=max(t[ls].s,t[rs].s);
    t[rt].p=max(t[ls].p,t[rs].p);
}

void updata(int rt,int l,int r,int ul,int ur,int x){
    if(ul<=l&&r<=ur){
        t[rt].tag+=x,t[rt].s+=x;
        t[rt].lazy=max(t[rt].lazy,t[rt].tag);
        t[rt].p=max(t[rt].s,t[rt].p);
        return;
    }
    pushdown(rt);
    int mid=(l+r)>>1;
    if(ul<=mid) updata(ls,l,mid,ul,ur,x);
    if(ur>mid) updata(rs,mid+1,r,ul,ur,x);
    put(rt);
}

ll query(int rt,int l,int r,int ql,int qr){
    if(ql<=l&&r<=qr) return t[rt].p;
    pushdown(rt);
    ll ans=-100005;
    int mid=(l+r)>>1;
    if(ql<=mid) ans=query(ls,l,mid,ql,qr);
    if(qr>mid) ans=max(ans,query(rs,mid+1,r,ql,qr));
    return ans;
}

int main(){
    n=input();
    for(int i=1;i<=n;i++) a[i]=input();

```

```
    m=input();
    for(int i=1;i<=m;i++){
        q[i].l=input(),q[i].r=input(),q[i].id=i;
    }

    sort(q+1,q+1+m);

    int tail=1;
    for(int i=1;i<=n;i++){
        if(tail>m) break;
        int tmp=a[i]+100005;
        updata(1,1,n,pre[tmp]+1,i,a[i]);
        pre[tmp]=i;
        while(q[tail].r==i) Ans[q[tail].id]=query(1,1,n,q[tail].l,i),tail++;
    }
    for(int i=1;i<=m;i++){
        printf("%lld\n",Ans[i]);
    }
}
```

2.15.3 GSS3

题目大意：给出一个数列，多次询问区间最长连续子段和，支持单点修改题解：线段树维护区间最长连续子段和 gss，区间从最左元素开始的最长连续子段和 lgss 区间以最右元素为结尾的最长连续子段和 rgss 以及区间和 s，信息传递并合并即可单点修改最多影响树上 log 个点，重新合并计算即可

```
#include <bits/stdc++.h>

using namespace std;

#define ll long long
ll input(){
    ll f=0,x=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

const int N=50007;

ll n,m;
ll a[N];
struct node{
    ll mxl,mxr,mxs,sum;
}t[N*4];
node merge(node a,node b){
    node res;
    res.sum=a.sum+b.sum;
    res.mxs=max(a.mxr+b.mxl,max(a.mxs,b.mxs));
    res.mxl=max(a.mxl,a.sum+b.mxl);
    res.mxr=max(b.mxr,b.sum+a.mxr);
```

```
26      return res;
27  }
28
29  void build(int rt,int l,int r){
30      if(l==r){
31          t[rt].sum=t[rt].mxl=t[rt].mxr=t[rt].mxs=a[l];
32          return;
33      }
34
35      int mid=(l+r)>>1;
36      build(rt<<1,l,mid),build(rt<<1|1,mid+1,r);
37      t[rt]=merge(t[rt<<1],t[rt<<1|1]);
38  }
39
40  void updata(int rt,int l,int r,int pos,int x){
41      if(l==r){
42          t[rt].sum=t[rt].mxl=t[rt].mxr=t[rt].mxs=x;
43          return;
44      }
45      int mid=(l+r)>>1;
46      if(pos<=mid) updata(rt<<1,l,mid,pos,x);
47      else updata(rt<<1|1,mid+1,r,pos,x);
48      t[rt]=merge(t[rt<<1],t[rt<<1|1]);
49  }
50
51  node query(int rt,int l,int r,int ql,int qr){
52      if(ql<=l&&r<=qr)
53          return t[rt];
54      node res;
55      int mid=(l+r)>>1;
56      if(qr<=mid) res=query(rt<<1,l,mid,ql,qr);
57      else if(ql>mid) res=query(rt<<1|1,mid+1,r,ql,qr);
58      else res=merge(query(rt<<1,l,mid,ql,qr),query(rt<<1|1,mid+1,r,ql,qr));
59      return res;
60  }
61
62  int main(){
63      n=input();
64      for(int i=1;i<=n;i++) a[i]=input();
65      build(1,1,n);
66      m=input();
67      for(int i=1;i<=m;i++){
68          int o=input();
69          if(o==0){
70              int x=input(),y=input();
71              updata(1,1,n,x,y);
72          }else{
73              int l=input(),r=input();
74              printf("%lld\n",query(1,1,n,l,r).mxs);
75          }
76      }
77  }
```

2.15.4 GSS4

题目：给出 N 个数，两种操作，更新操作为把区间的每一个数，都开方，查询操作为区间求和
做法：这题的操作为开方操作，可以 Lazy，可是基本没有什么效果，操作不能进行合并是一个大问题但是也正因为这是开方操作，也是突破点。一个 64 位整数，最多开方 7 次就变成 1 了，之后的开方操作也就没有效果了，这就是突破口如果一个区间全是 1 了，那就没必要执行开方操作了，而最多只有 7 次，那就对区间的每一个暴力加一点优化就是区间的和等于区间的个数，即区间全为 1，就不更新，或者记录一个最值，最值为 1 不更新

```cpp
#include <bits/stdc++.h>

using namespace std;

#define ll long long
ll input(){
   ll x=0,f=0;char ch=getchar();
   while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
   while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
   return f? -x:x;
}

const int N=1e5+7;

int n,m;
ll a[N];

struct seg{
   ll sum,mx;
}t[N*4];

void put(int rt){
   t[rt].mx=max(t[rt<<1].mx,t[rt<<1|1].mx);
   t[rt].sum=t[rt<<1].sum+t[rt<<1|1].sum;
}

void build(int rt,int l,int r){
   if(l==r){
      t[rt].mx=t[rt].sum=a[l];
      return;
   }
   int mid=(l+r)>>1;
   build(rt<<1,l,mid),build(rt<<1|1,mid+1,r);
   put(rt);
}

void update(int rt,int l,int r,int ul,int ur){
   if(l==r){
      t[rt].mx=sqrt(t[rt].mx);
      t[rt].sum=sqrt(t[rt].sum);
      return;
   }
   if(t[rt].mx==1) return;
```

```
    int mid=(l+r)>>1;
    if(ur<=mid) update(rt<<1,l,mid,ul,ur);
    else if(mid<ul) update(rt<<1|1,mid+1,r,ul,ur);
    else update(rt<<1,l,mid,ul,ur),update(rt<<1|1,mid+1,r,ul,ur);
    put(rt);
}

ll query(int rt,int l,int r,int ql,int qr){
    if(ql<=l&&r<=qr) return t[rt].sum;
    int mid=(l+r)>>1;
    if(qr<=mid) return query(rt<<1,l,mid,ql,qr);
    if(mid<ql) return query(rt<<1|1,mid+1,r,ql,qr);
    return query(rt<<1,l,mid,ql,qr)+query(rt<<1|1,mid+1,r,ql,qr);
}

int main(){
    int cas=0;
    while(~scanf("%d",&n)){
        for(int i=1;i<=n;i++)
            a[i]=input();
        build(1,1,n);
        m=input();
        printf("Case #%d:\n",++cas);
        for(int i=1;i<=m;i++){
            int f=input(),l=input(),r=input();
            if(l>r) swap(l,r);
            if(f==0) update(1,1,n,l,r);
            else printf("%lld\n",query(1,1,n,l,r));
        }
    }
}
```

2.15.5 GSS5

题目大意：给出一个数列，多次询问区间最长连续子段和，要求区间的左端点在 x1,y1 之间，右端点在 x2,y2 之间题解：线段树维护区间最长连续子段和 gss，区间从最左元素开始的最长连续子段和 lgss 区间以最右元素为结尾的最长连续子段和 rgss 以及区间和 s，信息传递并合并即可考虑询问区间的特殊性，如果两个端点区间不相交，则答案为 [x1,y1].rgss+[y1+1,x2-1].s+[x2,y2].lgss 如果区间相交，则中间区间 [x2,y1] 中必须有取到的部分，剩余两个区间可取可不取，但是必须与中间取到部分连接才能用于更新答案

```
#include <bits/stdc++.h>

using namespace std;

#define ll long long
ll input(){
    ll x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
```

```
}

const int N=10007;

int n,m;
int a[N];

#define debug printf("PASS IN LINE:%d\n",__LINE__)

struct node{
    ll ls,rs,ms,ss;
}t[N*4];

node merge(node l,node r){
    node res;
    res.ss=l.ss+r.ss;
    res.ls=max(l.ls,l.ss+r.ls);
    res.rs=max(r.rs,l.rs+r.ss);
    res.ms=max(l.rs+r.ls,max(l.ms,r.ms));
    return res;
}

void build(int rt,int l,int r){
    if(l==r){
        t[rt].ls=t[rt].rs=t[rt].ss=t[rt].ms=a[l];
        return;
    }
    int mid=(l+r)>>1;
    build(rt<<1,l,mid),build(rt<<1|1,mid+1,r);
    t[rt]=merge(t[rt<<1],t[rt<<1|1]);
}

node _query(int rt,int l,int r,int ql,int qr){
    if(ql<=l&&r<=qr)
        return t[rt];
    int mid=(l+r)>>1;
    if(qr<=mid) return _query(rt<<1,l,mid,ql,qr);
    else if(ql>mid) return _query(rt<<1|1,mid+1,r,ql,qr);
    else return merge(_query(rt<<1,l,mid,ql,qr),_query(rt<<1|1,mid+1,r,ql,qr));
}

node query(int l,int r){
    if(l>r) return t[0];
    return _query(1,1,n,l,r);
}

int main(){
    int T=input();
    while(T--){
        n=input();
        for(int i=1;i<=n;i++) a[i]=input();
        build(1,1,n);
        m=input();
```

```
64        for(int i=1;i<=m;i++){
65            int x1=input(),y1=input(),x2=input(),y2=input();
66            if(y1<x2)printf("%d\n",query(x1,y1).rs+query(y1+1,x2-1).ss+query(x2,y2).ls
                  );
67            else printf("%d\n",max(query(x2,y1).ms,max((query(x1,y1).rs+query(y1+1,y2)
                  .ls),(query(x1,x2-1).rs+query(x2,y2).ls))));
68        }
69    }
70 }
```

3 图论

3.1 最短路

3.1.1 Dijkstra

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  inline int input(){
6      int x=0,f=0;char ch=getchar();
7      while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
8      while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
9      return f? -x:x;
10 }
11
12 #define N (int)1e5+7
13 #define clr(a,b) memset(a,b,sizeof a)
14
15 struct edge{int v,w,next;}e[5*N];
16 int head[N],Cnt=0;
17 int n,m,s;
18 int dis[N];
19 struct node{
20     int u,d;
21     bool operator <(const node& rhs) const {return d>rhs.d;}
22 };
23
24 void Ins(int u,int v,int w){
25     e[++Cnt]=(edge){v,w,head[u]};
26     head[u]=Cnt;
27 }
28
29 priority_queue <node> Q;
30
31 void Dijkstra(){
32     clr(dis,0x3f);
33     dis[s]=0;
34     Q.push((node){s,0});
35     while(!Q.empty()){
36         node fr=Q.top(); Q.pop();
```

```
37        int u=fr.u,d=fr.d,v,w;
38        if(d!=dis[u]) continue;
39        for(int i=head[u];i;i=e[i].next){
40            if(dis[u]+(w=e[i].w)<dis[v=e[i].v])
41                dis[v]=dis[u]+w,Q.push((node){v,dis[v]});
42        }
43    }
44 }
45
46 int main(){
47    n=input(),m=input(),s=input();
48    for(int i=1;i<=m;i++){
49        int u=input(),v=input(),w=input();
50        Ins(u,v,w);
51    }
52    Dijkstra();
53    for(int i=1;i<=n;i++) printf("%d ",dis[i]);
54    return 0;
55 }
```

3.1.2 spfa

```
1  #include<iostream>
2  #include<cstdio>
3  #include<queue>
4  #define INF 19260817
5  using namespace std;
6  struct edge{
7     int to,next,w;
8  }map[2007];
9  int head[1007];
10 int mapl[1007];//原点到i点的距离
11 bool mapb[1007];//是否在队列中
12 int c=0;
13 int n,m;
14 void makemap(int a,int b,int d){//建图
15    map[++c].to=b;
16    map[c].w=d;
17    map[c].next=head[a];
18    head[a]=c;
19 }
20 void BEGIN(){//初始化
21    for(int i=1;i<=n;i++){
22        mapl[i]=INF;
23        mapb[i]=0;
24        head[i]=-1;
25    }
26 }
27 queue <int> q;
28 void spfa(int sta){
29    int now;
30    mapl[sta]=0;
```

```
    q.push(sta);
    while(!q.empty()){
        now=q.front();
        q.pop();
        mapb[now]=0;
        for(int i=head[now];i!=-1;i=map[i].next){
            if(map[i].to==now) continue; //双向边防止回到起始点
            if(mapl[now]+map[i].w<mapl[map[i].to]){//松弛操作
                mapl[map[i].to]=mapl[now]+map[i].w;
                if(mapb[map[i].to]==0){//判断是否入队
                    q.push(map[i].to);
                    mapb[map[i].to]=1;
                }
            }
        }
    }
}
int main(){
    BEGIN();
    cin>>n>>m;
    int x,y,z;
    for(int i=1;i<=m;i++){
        cin>>x>>y>>z;
        makemap(x,y,z);
        makemap(y,x,z);//双向边存图
    }
    spfa(1);
    for(int i=1;i<=n;i++) printf("%d ",mapl[i]);
}
```

3.1.3 k 短路

问题描述:

给定一个有 n 个结点，m 条边的有向图，求从 s 到 t 的所有不同路径中的第 k 短路径的长度。

$A*$ 算法:

$A*$ 算法定义了一个对当前状态 x 的估价函数 $f(x)=g(x)+h(x)$ ，其中 $g(x)$ 为从初始状态到达当前状态的实际代价，$h(x)$ 为从当前状态到达目标状态的最佳路径的估计代价。每次取出 $f(x)$ 最优的状态 x ，扩展其所有子状态，可以用 ** 优先队列 ** 来维护这个值。

在求解 k 短路问题时，令 $h(x)$ 为从当前结点到达终点 t 的最短路径长度。可以通过在反向图上对结点 t 跑单源最短路预处理出对每个结点的这个值。

由于设计的距离函数和估价函数，对于每个状态需要记录两个值，为当前到达的结点 x 和已经走过的距离 $g(x)$ ，将这种状态记为 $(x,g(x))$ 。

开始我们将初始状态 $(s,0)$ 加入优先队列。每次我们取出估价函数 $f(x)=g(x)+h(x)$ 最小的一个状态，枚举该状态到达的结点 x 的所有出边，将对应的子状态加入优先队列。当我们访问到一个结点第 k 次时，对应的状态的 $g(x)$ 就是从 x 到该结点的第 k 短路。

优化：由于只需要求出从初始结点到目标结点的第 k 短路，所以已经取出的状态到达一个结

点的次数大于 k 次时，可以不扩展其子状态。因为之前 k 次已经形成了 k 条合法路径，当前状态不会影响到最后的答案。

当图的形态是一个 n 元环的时候，该算法最坏是 $O(nk\log n)$ 的。但是这种算法可以在相同的复杂度内求出从起始点 s 到每个结点的前 k 短路。

```
#include <algorithm>
#include <cstdio>
#include <cstring>
#include <queue>
using namespace std;
const int maxn = 5010;
const int maxm = 400010;
const int inf = 2e9;
int n, m, s, t, k, u, v, ww, H[maxn], cnt[maxn];
int cur, h[maxn], nxt[maxm], p[maxm], w[maxm];
int cur1, h1[maxn], nxt1[maxm], p1[maxm], w1[maxm];
bool tf[maxn];
void add_edge(int x, int y, double z) {
  cur++;
  nxt[cur] = h[x];
  h[x] = cur;
  p[cur] = y;
  w[cur] = z;
}
void add_edge1(int x, int y, double z) {
  cur1++;
  nxt1[cur1] = h1[x];
  h1[x] = cur1;
  p1[cur1] = y;
  w1[cur1] = z;
}
struct node {
  int x, v;
  bool operator<(node a) const { return v + H[x] > a.v + H[a.x]; }
};
priority_queue<node> q;
struct node2 {
  int x, v;
  bool operator<(node2 a) const { return v > a.v; }
} x;
priority_queue<node2> Q;
int main() {
  scanf("%d%d%d%d%d", &n, &m, &s, &t, &k);
  while (m--) {
    scanf("%d%d%d", &u, &v, &ww);
    add_edge(u, v, ww);
    add_edge1(v, u, ww);
  }
  for (int i = 1; i <= n; i++) H[i] = inf;
  Q.push({t, 0});
  while (!Q.empty()) {
    x = Q.top();
```

```
    Q.pop();
    if (tf[x.x]) continue;
    tf[x.x] = true;
    H[x.x] = x.v;
    for (int j = h1[x.x]; j; j = nxt1[j]) Q.push({p1[j], x.v + w1[j]});
  }
  q.push({s, 0});
  while (!q.empty()) {
    node x = q.top();
    q.pop();
    cnt[x.x]++;
    if (x.x == t && cnt[x.x] == k) {
      printf("%d\n", x.v);
      return 0;
    }
    if (cnt[x.x] > k) continue;
    for (int j = h[x.x]; j; j = nxt[j]) q.push({p[j], x.v + w[j]});
  }
  printf("-1\n");
  return 0;
}
```

3.2 树上问题

3.2.1 倍增求 LCA

```
#include<iostream>
#include<cstdio>
#include<cstring>
using namespace std;
struct node{
   int next;
   int to;
}map[6014];
int head[3007];
int c=1;
int p[3007][13]={0};
int n;
int deep[3007]={0};
int bcj[3007];
void makemap(int x,int y){
   map[c].next=head[x];
   map[c].to=y;
   head[x]=c;
   c++;
}
void init(){
   int i,j;
   for(j=1;(1<<j)<=n;j++){
      for(i=1;i<=n;i++){
         if(p[i][j-1]!=0){
            p[i][j]=p[p[i][j-1]][j-1];
```

```
            }
        }
    }
}
void dfs(int u){
    for(int i=head[u]; i ;i=map[i].next){
        if(deep[map[i].to]==0){
            deep[map[i].to]=deep[u]+1;
            p[map[i].to][0]=u;
            dfs(map[i].to);
        }
    }
}
int lca(int a,int b){
    int i,j;
    if(deep[a]<deep[b])swap(a,b);
    for(i=0;(1<<i)<=deep[a];i++);
    i--;
    for(j=i;j>=0;j--)
        if(deep[a]-(1<<j)>=deep[b])
            a=p[a][j];
    if(a==b)return a;
    for(j=i;j>=0;j--)
    {
        if(p[a][j]!=0&&p[a][j]!=p[b][j])
        {
            a=p[a][j];
            b=p[b][j];
        }
    }
    return p[a][0];
}
int main(){
    cin>>n;
    int x,y;
    for(int i=1;i<=n;i++){
        bcj[i]=i;
    }
    for(int i=1;i<n;i++){
        cin>>x>>y;
        bcj[x]=bcj[y];
        makemap(y,x);
    }
    dfs(bcj[1]);
    init();
    int a,b,t;
    cin>>t;
    for(int i=1;i<=t;i++){
        cin>>a>>b;
        cout<<lca(a,b)<<endl;
    }
    return 0;
}
```

3.2.2 树的重心

定义: 对于树上的每一个点，计算其所有子树中最大的子树节点数，这个值最小的点就是这棵树的重心。

（这里以及下文中的“子树”都是指无根树的子树，即包括“向上”的那棵子树，并且不包括整棵树自身。）

性质: 以树的重心为根时，所有子树的大小都不超过整棵树大小的一半。

树中所有点到某个点的距离和中，到重心的距离和是最小的；如果有两个重心，那么到它们的距离和一样。

把两棵树通过一条边相连得到一棵新的树，那么新的树的重心在连接原来两棵树的重心的路径上。

在一棵树上添加或删除一个叶子，那么它的重心最多只移动一条边的距离

```
// 这份代码默认节点编号从 1 开始，即 i ∈ [1,n]
int size[MAXN], // 这个节点的“大小”（所有子树上节点数 + 该节点）
    weight[MAXN], // 这个节点的“重量”
    centroid[2]; // 用于记录树的重心（存的是节点编号）
void GetCentroid(int cur, int fa) { // cur 表示当前节点 (current)
  size[cur] = 1;
  weight[cur] = 0;
  for (int i = head[cur]; i != -1; i = e[i].nxt) {
    if (e[i].to != fa) { // e[i].to 表示这条有向边所通向的节点。
      GetCentroid(e[i].to, cur);
      size[cur] += size[e[i].to];
      weight[cur] = max(weight[cur], size[e[i].to]);
    }
  }
  weight[cur] = max(weight[cur], n - size[cur]);
  if (weight[cur] <= n / 2) { // 依照树的重心的定义统计
    centroid[centroid[0] != 0] = cur;
  }
}
```

3.2.3 树链剖分

```
#include<iostream>
#include<cstdio>
#define mx(a,b) a>b? a:b
#define mi(a,b) a<b? a:b
#define INF 19260817
using namespace std;
int input(){
    int x=0,f=1;char ch=getchar();
    while(ch<'0'||ch>'9') {if(ch=='-') f=-1;ch=getchar();}
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f*x;
}
struct edge{
    int to,next,w;
```

```
}map[20007];
struct stree{
   int l,r;
}stree[10007];
int head[10007];
int w[10007];
int c=0;
int size[10007],top[10007],fa[10007],deep[10007],son[10007],rank[10007],tid[10007];
void makemap(int x,int y){
   map[++c].to=y;
   map[c].next=head[x];
   head[x]=c;
}
void dfs1(int now,int father){
   deep[now]=deep[father]+1;
   size[now]=1;
   fa[now]=father;
   for(int i=head[now];i;i=map[i].next){
      if(map[i].to==father) continue;
      dfs1(map[i].to,now);
      size[now]+=size[map[i].to];
      if(!son[now]||size[map[i].to]>size[son[now]]) son[now]=map[i].to;
   }
}
int tim=0;
void dfs2(int now,int tp){
   top[now]=tp;
   tid[now]=++tim;
   rank[tid[now]]=w[now];
   stree[now].l=tim;
   if(!son[now]) return;
   dfs2(son[now],tp);
   for(int i=head[now];i;i=map[i].next){
      if(map[i].to==fa[now]||map[i].to==son[now]) continue;
      dfs2(map[i].to,map[i].to);
   }
   stree[now].r=tim;
}
class segment_tree{
   public:
      void build(int rootr,int l,int r){
         tree[rootr].l=l;tree[rootr].r=r;
         if(l==r){tree[rootr].MX=tree[rootr].MI=tree[rootr].tot=rank[l];return;}
         int mid=(r-l>>1)+l;
         build(rootr*2,l,mid);
         build(rootr*2+1,mid+1,r);
         put(rootr);
      }
      int MX,MI,tot;
      void query(int l,int r){
         MX=0,MI=INF,tot=0;
         getans(1,l,r);
      }
```

```
68        void updata(int rootr,int l,int r,int add){
69            int L=tree[rootr].l,R=tree[rootr].r;
70            if(L>r||R<l)return;
71            if(l<=L&&R<=r){tree[rootr].tot+=add*(R-L);tree[rootr].MX+=add;tree[rootr].
                  MI+=add;tree[rootr].lazy+=add;return;}
72            putdown(rootr);
73            updata(rootr*2,l,r,add);
74            updata(rootr*2+1,l,r,add);
75            put(rootr);
76        }
77    private:
78    struct node{
79            int l,r,lazy,MX,MI,tot;
80        }tree[200007];
81    void put(int rootr){
82        tree[rootr].MX=tree[rootr*2].MX>tree[rootr*2+1].MX? tree[rootr*2].MX:tree[
              rootr*2+1].MX;
83        tree[rootr].MI=tree[rootr*2].MI<tree[rootr*2+1].MI? tree[rootr*2].MI:tree[
              rootr*2+1].MI;
84        tree[rootr].tot=tree[rootr*2].tot+tree[rootr*2+1].tot;
85    }
86    void putdown(int rootr){
87        if(tree[rootr].lazy!=0){
88            tree[rootr*2].lazy+=tree[rootr].lazy;
89            tree[rootr*2+1].lazy+=tree[rootr].lazy;
90            tree[rootr*2].MX+=tree[rootr].lazy;tree[rootr*2].MI+=tree[rootr].lazy;
91            tree[rootr*2+1].MX+=tree[rootr].lazy;tree[rootr*2+1].MI+=tree[rootr].lazy;
92            tree[rootr*2].tot+=tree[rootr].lazy*(tree[rootr*2].r-tree[rootr*2].l+1);
93            tree[rootr*2+1].tot+=tree[rootr].lazy*(tree[rootr*2+1].r-tree[rootr*2+1].l
                  +1);
94            tree[rootr].lazy=0;
95        }
96    }
97    void getans(int rootr,int l,int r){
98        int L=tree[rootr].l,R=tree[rootr].r;
99        if(L>r||R<l)return;
100       if(l<=L&&R<=r){tot+=tree[rootr].tot;MX=MX>tree[rootr].MX?MX:tree[rootr].MX;MI
              =MI<tree[rootr].MI?MI:tree[rootr].MI;return;}
101       putdown(rootr);
102       getans(rootr*2,l,r);
103       getans(rootr*2+1,l,r);
104       put(rootr);
105   }
106};
107segment_tree tree1;
108namespace ttree{
109   int cntT=0,cntX=0,cntI=INF;
110   void up(){cntT+=tree1.tot;cntI=mi(cntI,tree1.MI);cntX=mx(cntX,tree1.MX);}
111   void tquery(int u,int v){
112       cntT=0;cntX=0;cntI=INF;
113       while(top[u]!=top[v]){
114           if(deep[top[u]]<deep[top[v]]) swap(u,v);
115           tree1.query(tid[top[u]],tid[u]),up();
```

```
116             u=fa[top[u]];
117         }
118         if(deep[u]<deep[v]) swap(u,v);
119         tree1.query(tid[v],tid[u]),up;
120     }
121     void tupdata(int u,int v,int add){
122         while(top[u]!=top[v]){
123             if(deep[top[u]]<deep[top[v]]) swap(u,v);
124             tree1.updata(1,tid[top[u]],tid[u],add);
125             u=fa[top[u]];
126         }
127         if(deep[u]<deep[v]) swap(u,v);
128         tree1.updata(1,tid[v],tid[u],add);
129     }
130     int lca(int u,int v){
131         while(top[u]!=top[v]){
132             if(deep[top[u]]<deep[top[v]]) swap(u,v);
133             u=fa[top[u]];
134         }
135         return deep[u]>deep[v]? v:u;
136     }
137     void squery(int x){cntT=0;cntX=0;cntI=INF;tree1.query(stree[x].l,stree[x].r);}
138     void supdata(int x,int add){tree1.updata(stree[x].l,stree[x].r,add);}
139 }
140 int n;
141 int main(){
142     n=input();
143     int x,y;
144     for(int i=1;i<n;i++){
145         x=input();
146         y=input();
147         makemap(x,y);
148         makemap(y,x);
149     }
150     for(int i=1;i<=n;i++) w[i]=input();
151     dfs1(1,0);dfs2(1,1);
152     tree1.build(1,1,n);
153 
154     return 0;
155 }
```

3.2.4 点分治

```
1 #include <bits/stdc++.h>
2 
3 using namespace std;
4 
5 #define ll long long
6 ll input(){
7     ll x=0,f=0;char ch=getchar();
8     while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
9     while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
```

```
10      return f? -x:x;
11  }
12
13  #define PII pair<int,int>
14  #define fr first
15  #define sc second
16  #define mp make_pair
17  #define debug printf("PASS IN LINE %d\n",__LINE__)
18
19  const int N=10007;
20
21  vector<PII> G[N];
22
23  int n,m,k;
24  int rt,siz[N],f[N],dep[N],size,Ans;
25  vector <int> d;
26  bool vis[N];
27
28  void getrt(int u,int fa){
29      siz[u]=1,f[u]=0;
30      for(auto v:G[u]){
31          if(v.fr==fa||vis[v.fr]) continue;
32          getrt(v.fr,u);
33          siz[u]+=siz[v.fr];
34          f[u]=max(f[u],siz[v.fr]);
35      }
36      f[u]=max(f[u],size-siz[u]);
37      if(f[u]<f[rt]) rt=u;
38  }
39
40  void getdep(int u,int fa){
41      d.push_back(dep[u]);
42      siz[u]=1;
43      for(auto v:G[u]){
44          if(vis[v.fr]||v.fr==fa) continue;
45          dep[v.fr]=dep[u]+v.sc;
46          getdep(v.fr,u);
47          siz[u]+=siz[v.fr];
48      }
49  }
50
51  int calc(int u,int init){
52      d.clear(),dep[u]=init;
53      getdep(u,0);
54      sort(d.begin(),d.end());
55      int res=0;
56      for(int l=0,r=d.size()-1;l<r;)
57          if(d[l]+d[r]<=k) res+=r-l++;
58          else r--;
59      return res;
60  }
61
62  void Solve(int u){
```

```
63      Ans+=calc(u,0);
64      vis[u]=1;
65      for(auto v:G[u]){
66          if(vis[v.fr]) continue;
67          Ans-=calc(v.fr,v.sc);
68          f[0]=size=siz[u];
69          getrt(v.fr,rt=0);
70          Solve(rt);
71      }
72  }
73
74  int main(){
75      n=input(),k=input();
76      for(int i=1;i<=n;i++) G[i].clear();d.clear();
77      memset(vis,0,sizeof vis);
78      for(int i=1;i<n;i++){
79          int u=input(),v=input(),w=input();
80          G[u].push_back(mp(v,w));
81          G[v].push_back(mp(u,w));
82      }
83      f[0]=size=n;
84      getrt(1,rt=0);
85      Ans=0;
86      Solve(rt);
87      printf("%d\n",Ans);
88  }
89
90
91  ////////////////////////////////////////
92
93
94  // poj 1741
95
96  //求长度小于等于k的边的数量
97  #include <bits/stdc++.h>
98
99  using namespace std;
100
101 #define ll long long
102 ll input(){
103     ll x=0,f=0;char ch=getchar();
104     while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
105     while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
106     return f? -x:x;
107 }
108
109 #define PII pair<int,int>
110 #define fr first
111 #define sc second
112 #define mp make_pair
113 #define debug printf("PASS IN LINE %d\n",__LINE__)
114
115 const int N=10007;
```

```cpp

vector<PII> G[N];

int n,m,k;
int rt,siz[N],f[N],dep[N],size,Ans;
vector <int> d;
bool vis[N];

void getrt(int u,int fa){
    siz[u]=1,f[u]=0;
    for(auto v:G[u]){
    for(int i=0;i<G[u].size();i++){
        PII v=G[u][i];
        if(v.fr==fa||vis[v.fr]) continue;
        getrt(v.fr,u);
        siz[u]+=siz[v.fr];
        f[u]=max(f[u],siz[v.fr]);
    }
    f[u]=max(f[u],size-siz[u]);
    if(f[u]<f[rt]) rt=u;
}

void getdep(int u,int fa){
    d.push_back(dep[u]);
    siz[u]=1;
    for(auto v:G[u]){
    for(int i=0;i<G[u].size();i++){
        PII v=G[u][i];
        if(vis[v.fr]||v.fr==fa) continue;
        dep[v.fr]=dep[u]+v.sc;
        getdep(v.fr,u);
        siz[u]+=siz[v.fr];
    }
}

int calc(int u,int init){
    d.clear(),dep[u]=init;
    getdep(u,0);
    sort(d.begin(),d.end());
    int res=0;
    for(int l=0,r=d.size()-1;l<r;)
        if(d[l]+d[r]<=k) res+=r-l++;
        else r--;
    return res;
}

void Solve(int u){
    Ans+=calc(u,0);
    vis[u]=1;
    for(auto v:G[u]){
    for(int i=0;i<G[u].size();i++){
        PII v=G[u][i];
        if(vis[v.fr]) continue;
```

```
169         Ans-=calc(v.fr,v.sc);
170         f[0]=size=siz[u];
171         getrt(v.fr,rt=0);
172         Solve(rt);
173     }
174 }
175 
176 int main(){
177     while(1){
178         n=input(),k=input();
179         if(!n&&!k) break;
180         for(int i=1;i<=n;i++) G[i].clear();d.clear();
181         memset(vis,0,sizeof vis);
182         for(int i=1;i<n;i++){
183             int u=input(),v=input(),w=input();
184             G[u].push_back(mp(v,w));
185             G[v].push_back(mp(u,w));
186         }
187         f[0]=size=n;
188         getrt(1,rt=0);
189         Ans=0;
190         Solve(rt);
191         printf("%d\n",Ans);
192     }
193 }
```

3.2.5 树上启发式合并

```
1  #include <bits/stdc++.h>
2  
3  using namespace std;
4  
5  #define ll long long
6  ll input(){
7      ll x=0,f=0;char ch=getchar();
8      while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
9      while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
10     return f? -x:x;
11 }
12 
13 const int N=2e5+7;
14 
15 ll son[N],siz[N],a[N],color[N];
16 ll Ans[N];
17 
18 vector<int> G[N];
19 
20 void dfs(int u,int fa){
21     siz[u]=1;
22     for(auto v:G[u]){
23         if(v==fa) continue;
24         dfs(v,u);
```

```
25        siz[u]+=siz[v];
26        if(siz[son[u]]<siz[v]) son[u]=v;
27    }
28 }
29
30 ll mx,ans;
31
32 void work(int u,int fa,int x,int ch){
33    color[a[u]]+=x;
34    if(color[a[u]]>mx) mx=color[a[u]],ans=a[u];
35    else if(color[a[u]]==mx) ans+=a[u];
36
37    for(auto v:G[u]){
38       if(v==fa||v==ch) continue;
39       work(v,u,x,ch);
40    }
41 }
42
43 void dfs1(int u,int fa,bool flag){
44    for(auto v:G[u]){
45       if(v==fa||v==son[u]) continue;
46       dfs1(v,u,1);
47    }
48    if(son[u]) dfs1(son[u],u,0);
49
50    work(u,fa,1,son[u]);
51
52    Ans[u]=ans;
53
54    if(flag){
55       work(u,fa,-1,0);
56       mx=ans=0;
57    }
58 }
59
60 int main(){
61    int n=input();
62    for(int i=1;i<=n;i++) a[i]=input();
63
64    for(int i=1;i<n;i++){
65       int u=input(),v=input();
66       G[u].push_back(v);
67       G[v].push_back(u);
68    }
69
70    dfs(1,0);
71    dfs1(1,0,0);
72
73    for(int i=1;i<=n;i++){
74       printf("%lld%c",Ans[i],i==n? '\n':' ');
75    }
76 }
77
```

```cpp

////////////////////////////////////////////////


#include <bits/stdc++.h>

using namespace std;

#define ll long long
ll input(){
    ll x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

const int N=2e5+7;

ll son[N],color[N],all[N],a[N],siz[N],Fa[N];
ll Ans[N];

struct edge{
    int v,next;
}e[N<<1];

int head[N],cnt=0;

void Ins(int u,int v){
    e[++cnt]=(edge){v,head[u]},head[u]=cnt;
    e[++cnt]=(edge){u,head[v]},head[v]=cnt;
}

void dfs(int u,int fa){
    siz[u]=1;Fa[u]=fa;
    for(int i=head[u];i;i=e[i].next){
        int v=e[i].v;
        if(v==fa) continue;
        dfs(v,u);
        siz[u]+=siz[v];
        if(siz[son[u]]<siz[v]) son[u]=v;
    }
}

int ans=0;

void work(int u,int fa,int x,int ch){
    all[a[u]]-=x;
    if(color[a[u]]&&!all[a[u]]) ans--;
    if(!color[a[u]]&&all[a[u]]) ans++;
    color[a[u]]+=x;

    for(int i=head[u];i;i=e[i].next){
        int v=e[i].v;
```

```
131         if(v==fa||v==ch) continue;
132         work(v,u,x,ch);
133     }
134 }
135 
136 void dfs1(int u,int fa,bool flag){
137     for(int i=head[u];i;i=e[i].next){
138         int v=e[i].v;
139         if(v==fa||v==son[u]) continue;
140         dfs1(v,u,1);
141     }
142     if(son[u]) dfs1(son[u],u,0);
143 
144     work(u,fa,1,son[u]);
145 
146     Ans[u]=ans;
147 
148     if(flag){
149         work(u,fa,-1,0);
150         ans=0;
151     }
152 }
153 
154 #define PII pair <int,int>
155 #define fr first
156 #define sc second
157 #define mp make_pair
158 
159 vector<PII> E;
160 
161 int main(){
162     int n;
163     while(~scanf("%d",&n)){
164         E.clear();
165 
166         cnt=0;
167         for(int i=0;i<=n;i++){
168             color[i]=son[i]=all[i]=head[i]=0;
169         }
170         for(int i=1;i<=n;i++) a[i]=input(),all[a[i]]++;
171 
172         for(int i=1;i<n;i++){
173             int u=input(),v=input();
174             Ins(u,v);
175             E.push_back(mp(u,v));
176         }
177 
178         dfs(1,0);
179         dfs1(1,0,0);
180 
181         for(auto t:E){
182             int u=t.fr,v=t.sc;
183             if(Fa[v]==u) swap(u,v);
```

```
184             printf("%lld\n",Ans[u]);
185         }
186     }
187 }
```

3.3 图上问题

3.3.1 矩阵树定理

求一个图的生成树的数量。用高斯消元实现。

```
1  #include<bits/stdc++.h>
2  #define rep(i,a,b) for(int i=a;i<=(b);++i)
3  #define per(i,a,b) for(int i=a;i>=(b);--i)
4  #define mem(a,x) memset(a,x,sizeof(a))
5  #define pb emplace_back
6  #define pii pair<int,int>
7  #define mk make_pair
8  using namespace std;
9  typedef long long ll;
10 const ll mod = 998244353;
11 ll gcd(ll a, ll b) { return b?gcd(b,a%b):a;}
12 ll powmod(ll a,ll b) {ll res=1;a%=mod;
13 assert(b>=0); for(;b;b>>=1){if(b&1)res=res*a%mod;a=a*a%mod;}return res%mod;}
14 inline ll read()
15 {
16     ll x=0,w=1; char c=getchar();
17     while(c<'0'||c>'9') {if(c=='-') w=-1; c=getchar();}
18     while(c<='9'&&c>='0') {x=(x<<1)+(x<<3)+c-'0'; c=getchar();}
19     return w==1?x:-x;
20 }
21
22 const int N = 1e2 + 10, M=1e5 + 10;
23 int u[M], v[M];
24 ll w[M];
25 ll f[35], K[N][N];
26 ll Gauss(int n){
27     ll res=1;
28     for(int i=1;i<=n;i++){
29         for(int j=i;j<=n;j++)
30             if(K[j][i]){
31                 for(int k=i;k<=n;k++)swap(K[i][k],K[j][k]);
32                 if(i!=j)res=-res;
33                 break;
34             }
35         if(!K[i][i])return 0;
36         for(ll j=i+1,iv=powmod(K[i][i],mod-2);j<=n;j++){
37                 ll t=K[j][i]*iv%mod;
38                 for(int k=i;k<=n;k++)
39                     K[j][k]=(K[j][k]-t*K[i][k]%mod+mod)%mod;
40         }
41         res=(res*K[i][i]%mod+mod)%mod;
42     }
```

```
43      return res;
44  }
45  //void Gauss(int n){
46  // for(int i=1;i<=n;i++){
47  // int j=i;
48  // while(j<=n&&!a[j][i]) j++;
49  // if(j!=i) for(int k=1;k<=n+1;k++) swap(a[i][k],a[j][k]);
50  // int inv=Inv(a[i][i]);
51  // for(int j=i+1;j<=n;j++){
52  // int t=a[j][i]*inv%P;
53  // for(int k=i;k<=n+1;k++) a[j][k]=(a[j][k]-t*a[i][k]%P+P)%P;
54  // }
55  // }
56  // for(int i=n;i>=1;i--){
57  // for(int j=n;j>i;j--) a[i][n+1]=(a[i][n+1]-a[j][n+1]*a[i][j]%P+P)%P;
58  // a[i][n+1]=a[i][n+1]*Inv(a[i][i])%P;
59  // }
60  //}
61
62  int main()
63  {
64      f[0] = 1;
65      for(int i = 1;i < 31; ++i) f[i] = f[i-1] * 2;
66
67      int _ = read();
68      while (_--) {
69          int n = read(), m = read();
70          for (int i = 1; i <= m; i++) u[i] = read(), v[i] = read(), w[i] = read();
71
72          ll fz = 0;
73          for (int j=0; j<=30; j++) {
74              memset(K, 0, sizeof(K));
75
76              for (int i = 1; i <= m; i++) {
77                  K[u[i]][u[i]] += (w[i] >> j) & 1;
78                  K[v[i]][v[i]] += (w[i] >> j) & 1;
79                  K[u[i]][v[i]] -= (w[i] >> j) & 1;
80                  K[v[i]][u[i]] -= (w[i] >> j) & 1;
81              }
82              fz = (fz + Gauss(n - 1) * f[j] % mod) % mod;
83          }
84
85          memset(K, 0, sizeof(K));
86          for (int i = 1; i <= m; i++) {
87              K[u[i]][u[i]]++;
88              K[v[i]][v[i]]++;
89              K[u[i]][v[i]]--;
90              K[v[i]][u[i]]--;
91          }
92          ll fm = powmod(Gauss(n - 1), mod - 2);
93          ll ans = fz * fm % mod;
94          printf("%lld\n", ans);
95      }
```

```
    return 0;
}



//////////////////////////////////////////////////////////


const int N = 1e2 + 10, M=1e5 + 10;
int u[M], v[M], w[M];
ll f[35], K[N][N];
ll Gauss(int n){
    ll res=1;
    for(int i=1;i<=n;i++){
        for(int j=i;j<=n;j++)
            if(K[j][i]){
                for(int k=i;k<=n;k++)swap(K[i][k],K[j][k]);
                if(i!=j)res=-res;
                break;
            }
        if(!K[i][i])return 0;
        for(ll j=i+1,iv=powmod(K[i][i],mod-2);j<=n;j++){
                ll t=K[j][i]*iv%mod;
                for(int k=i;k<=n;k++)
                    K[j][k]=(K[j][k]-t*K[i][k]%mod+mod)%mod;
        }
        res=(res*K[i][i]%mod+mod)%mod;
    }
    return res;
}
```

3.3.2 最小生成树 (Kruskal)

```
#include<iostream>
#include<cstdio>
#include<algorithm>
using namespace std;
int input(){
    int x=0,f=1;char ch=getchar();
    while(ch<'0'||ch>'9') {if(ch=='-') f=-1;ch=getchar();}
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f*x;
}
int n,m;
class bcj{
    public:
        void BEGIN(){
            for(int i=1;i<=n;i++){
                fa[i]=i;
                rank[i]=1;
            }
        }
```

```
20        int find(int x){return x==fa[x]? x:(fa[x]=find(fa[x]));}
21        int merge(int x,int y){
22            x=find(x);y=find(y);
23            x==y? 0:(rank[x]>=rank[y]? (fa[y]=x,rank[x]+=rank[y]):(fa[x]=y,rank[y]+=
                  rank[x]));
24            return rank[x]>=rank[y]? rank[x]:rank[y];
25        }
26    private:
27        int fa[1007],rank[1007];
28 };
29 bcj bcj_;
30 struct edge{
31    int from,w,to;
32 }map[2024];
33 struct mst{
34    int next,w,to;
35 }tree[2024];
36 int head[1024];
37 int c=0;
38 int cnt=0;
39 bool cmp(edge a,edge b){
40    return a.w<b.w;
41 }
42 void makemap(int x,int y,int z){
43    tree[++c].to=y;
44    tree[c].w=z;
45    tree[c].next=head[x];
46    head[x]=c;
47 }
48 void kruskal(){
49    int size=0,now=1;
50    while(size<n){
51        if(bcj_.find(map[now].from)!=bcj_.find(map[now].to)){
52            size=bcj_.merge(map[now].from,map[now].to);
53            cnt+=map[now].w;
54            makemap(map[now].from,map[now].to,map[now].w);makemap(map[now].to,map[now
                  ].from,map[now].w);
55        }
56        now++;
57    }
58 }
59 int main(){
60    n=input();m=input();
61    bcj_.BEGIN();
62    for(int i=1;i<=m;i++){
63        map[i].from=input();map[i].to=input();map[i].w=input();
64    }
65    sort(map+1,map+1+n,cmp);
66    kruskal();
67    cout<<cnt<<endl;
68 }
```

3.3.3 二分图最大匹配 (匈牙利算法)

```cpp
#include <bits/stdc++.h>

using namespace std;

#define ll long long
ll input(){
    ll x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

const int N=507;

#define pb push_back

vector<int> G[N*2];
int n,m,e;
int match[N*2],vis[N*2];


bool dfs(int u){
    for(auto v:G[u]){
        if(!vis[v]){// 要求不在交替路中
            vis[v]=1;// 放入交替路
            if(!match[v]||dfs(match[v])){ // 如果是未盖点，说明交替路为增广路，则交换路径，并返回成功
                match[v]=u,match[u]=v;
                return 1;
            }
        }
    }
    return 0;
}

int main(){
    n=input(),m=input(),e=input();

    for(int i=1;i<=e;i++){
        int u=input(),v=input()+n;
        G[u].pb(v),G[v].pb(u);
    }

    int Ans=0;
    for(int i=1;i<=n;i++){
        if(!match[i]){
            memset(vis,0,sizeof(vis));
            if(dfs(i)) Ans++;
        }

    }
```

```
51
52      printf("%d\n",Ans);
53  }
```

3.3.4 最小异或生成树

```
1   #include <bits/stdc++.h>
2
3   using namespace std;
4
5   #define ll long long
6   ll input(){
7       ll x=0,f=0;char ch=getchar();
8       while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
9       while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
10      return f? -x:x;
11  }
12
13  const int N=2e5+7;
14  const int inf=0x3f3f3f3f;
15
16  #define ls(x) t[x][0]
17  #define rs(x) t[x][1]
18  int L[N*40],R[N*40],t[N*40][2],cnt;
19  int n,a[N],root;
20
21  void Insert(int &rt,int x,int dep){
22      if(!rt) rt=++cnt;
23      L[rt]=min(L[rt],x),R[rt]=max(R[rt],x);
24      if(dep<0) return;
25      int bit=a[x]>>dep&1;
26      Insert(t[rt][bit],x,dep-1);
27  }
28
29  int query(int rt,int val,int dep){
30      if(dep<0) return 0;
31      int bit=val>>dep&1;
32      if(t[rt][bit]) return query(t[rt][bit],val,dep-1);
33      else return query(t[rt][bit^1],val,dep-1)+(1<<dep);
34  }
35
36  ll dfs(int rt,int dep){
37      if(dep<0) return 0;
38      if(R[ls(rt)]&&R[rs(rt)]){
39          int mi=inf;
40          for(int i=L[ls(rt)];i<=R[ls(rt)];i++) mi=min(mi,query(rs(rt),a[i],dep-1));
41          return dfs(ls(rt),dep-1)+dfs(rs(rt),dep-1)+mi+(1<<dep);
42      }
43      if(R[ls(rt)]) return dfs(ls(rt),dep-1);
44      if(R[rs(rt)]) return dfs(rs(rt),dep-1);
45      return 0;
46  }
```

```

int main(){
    n=input();
    memset(L,0x3f,sizeof L);
    for(int i=1;i<=n;i++) a[i]=input();
    sort(a+1,a+1+n);
    for(int i=1;i<=n;i++) Insert(root,i,30);
    printf("%lld\n",dfs(root,30));
}
```

3.4 tarjan

3.4.1 求强连通分量

```
#include <bits/stdc++.h>

using namespace std;

int input(){
    int x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f?-x:x;
}

#define N (int)1e6

int dfn[N],low[N],color[N];
bool vis[N];
stack <int> s;
int n,m;
struct edge{
    int v,next;
}e[N];
int head[N],top=0,col=0;

void Ins(int u,int v){
    e[++top]=(edge){v,head[u]},head[u]=top;
}

int dep=0;

void tarjan(int x){
    dfn[x]=++dep;
    low[x]=dep;
    vis[x]=1;
    s.push(x);
    for(int i=head[x];i;i=e[i].next){
        // cout<<x<<" "<<e[i].v<<endl;
        if(!dfn[e[i].v]){
            tarjan(e[i].v);
            low[x]=min(low[e[i].v],low[x]);
```

```
39          }else if(vis[e[i].v]){
40              low[x]=min(low[e[i].v],low[x]);
41          }
42      }
43      if(dfn[x]==low[x]){
44          color[x]=++col;
45          vis[x]=0;
46          while(s.top()!=x){
47              color[s.top()]=col;
48              vis[s.top()]=0;
49              s.pop();
50          }
51          s.pop();
52      }
53  }
54
55  int main(){
56      n=input(),m=input();
57      int u,v;
58      for(int i=1;i<=m;i++){
59          u=input(),v=input();
60          Ins(u,v);
61      }
62      tarjan(1);
63      for(int i=1;i<=n;i++){
64          printf("%d ",color[i]);
65      }
66      printf("\n");
67      return 0;
68  }
```

3.4.2 求割点

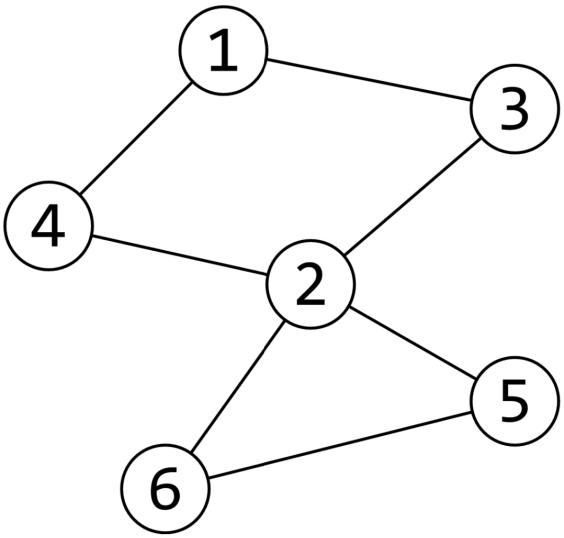
割点

> 对于一个无向图，如果把一个点删除后这个图的极大连通分量数增加了，那么这个点就是这个图的割点（又称割顶）。

如何实现？

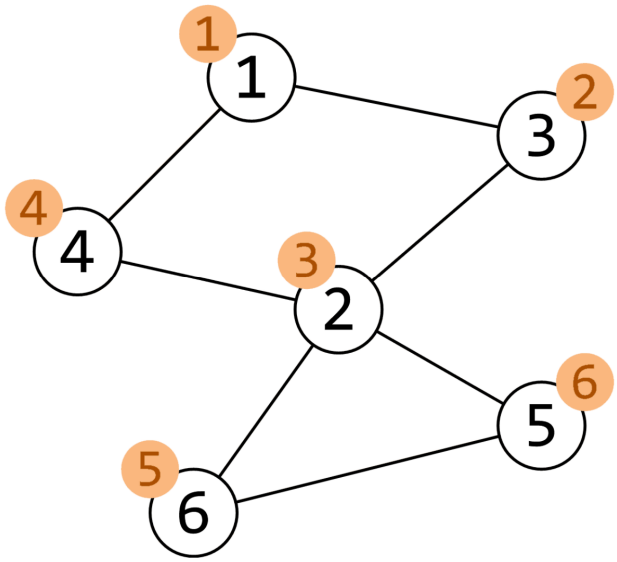
如果我们尝试删除每个点，并且判断这个图的连通性，那么复杂度会特别的高。所以要介绍一个常用的算法：Tarjan。

首先，我们上一个图：



很容易的看出割点是 2，而且这个图仅有这一个割点。

首先，我们按照 DFS 序给他打上时间戳（访问的顺序）。



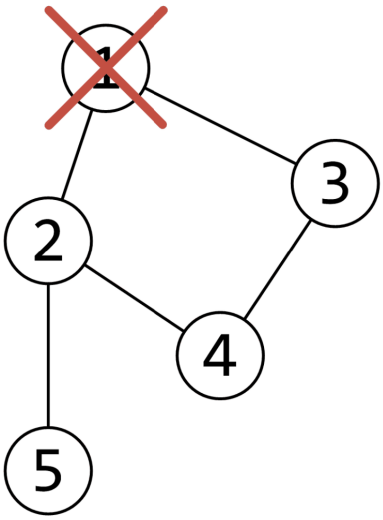
这些信息被我们保存在一个叫做 ‘num‘ 的数组中。

还需要另外一个数组 low ，用它来存储不经过其父亲能到达的最小的时间戳。

例如 low[2] 的话是 1，low[5] 和 low[6] 是 3。

然后我们开始 DFS，我们判断某个点是否是割点的根据是：对于某个顶点 u ，如果存在至少一个顶点 v （u 的儿子），使得 $low_v \geq num_u$ ，即不能回到祖先，那么 u 点为割点。

另外，如果搜到了自己（在环中），如果他有两个及以上的儿子，那么他一定是割点了，如果只有一个儿子，那么把它删掉，不会有任何的影响。比如下面这个图，此处形成了一个环，从树上来讲它有 2 个儿子：



我们在访问 1 的儿子时候，假设先 DFS 到了 2，然后标记用过，然后递归往下，来到了 4，4 又来到了 3，当递归回溯的时候，会发现 3 已经被访问过了，所以不是割点。

更新 low 的伪代码如下：

```
如果 v 是 u 的儿子 low[u] = min(low[u], low[v]);
否则
low[u] = min(low[u], num[v]);
```

例题洛谷 P3388【模板】割点（割顶）

```cpp
/*
洛谷 P3388 【模板】割点（割顶）
*/
#include <bits/stdc++.h>
using namespace std;
int n, m; // n：点数 m：边数
int num[100001], low[100001], inde, res;
// num：记录每个点的时间戳
// low：能不经过父亲到达最小的编号，inde：时间戳，res：答案数量
bool vis[100001], flag[100001]; // flag：答案 vis：标记是否重复
vector<int> edge[100001]; // 存图用的
void Tarjan(int u, int father) { // u 当前点的编号，father 自己爸爸的编号
  vis[u] = true; // 标记
  low[u] = num[u] = ++inde; // 打上时间戳
  int child = 0; // 每一个点儿子数量
  for (auto v : edge[u]) { // 访问这个点的所有邻居 （C++11）

    if (!vis[v]) {
      child++; // 多了一个儿子
      Tarjan(v, u); // 继续
      low[u] = min(low[u], low[v]); // 更新能到的最小节点编号
      if (father != u && low[v] >= num[u] &&
          !flag
             [u]) // 主要代码
                  // 如果不是自己，且不通过父亲返回的最小点符合割点的要求，并且没有被标记过
                  // 要求即为：删了父亲连不上去了，即为最多连到父亲
```

```
27      {
28        flag[u] = true;
29        res++; // 记录答案
30      }
31    } else if (v != father)
32      low[u] =
33          min(low[u], num[v]); // 如果这个点不是自己，更新能到的最小节点编号
34  }
35  if (father == u && child >= 2 &&
36      !flag[u]) { // 主要代码，自己的话需要 2 个儿子才可以
37    flag[u] = true;
38    res++; // 记录答案
39  }
40 }
41 int main() {
42  cin >> n >> m; // 读入数据
43  for (int i = 1; i <= m; i++) { // 注意点是从 1 开始的
44    int x, y;
45    cin >> x >> y;
46    edge[x].push_back(y);
47    edge[y].push_back(x);
48  } // 使用 vector 存图
49  for (int i = 1; i <= n; i++) // 因为 Tarjan 图不一定连通
50    if (!vis[i]) {
51      inde = 0; // 时间戳初始为 0
52      Tarjan(i, i); // 从第 i 个点开始，父亲为自己
53    }
54  cout << res << endl;
55  for (int i = 1; i <= n; i++)
56    if (flag[i]) cout << i << " "; // 输出结果
57  return 0;
58 }
```

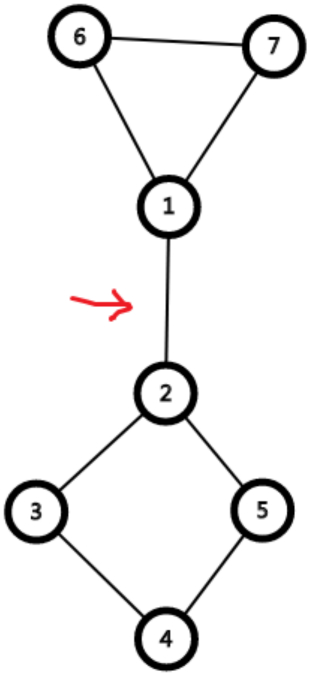
3.4.3 求桥

割边

和割点差不多，叫做桥。

> 对于一个无向图，如果删掉一条边后图中的连通分量数增加了，则称这条边为桥或者割边。严谨来说，就是：假设有连通图 $G=\{V,E\}$，e 是其中一条边（即 $e\in E$），如果 $G-e$ 是不连通的，则边 e 是图 G 的一条割边（桥）。

比如说，下图中，



红色箭头指向的就是割边。

实现

和割点差不多，只要改一处：$low_v > num_u$ 就可以了，而且不需要考虑根节点的问题。

割边是和是不是根节点没关系的，原来我们求割点的时候是指点 v 是不可能不经过父节点 u 为回到祖先节点（包括父节点），所以顶点 u 是割点。如果 $low_v = num_u$ 表示还可以回到父节点，如果顶点 v 不能回到祖先也没有另外一条回到父亲的路，那么 $u - v$ 这条边就是割边。

代码实现

下面代码实现了求割边，其中，当 isbridge[x] 为真时，(father[x],x) 为一条割边。

```
int low[MAXN], dfn[MAXN], iscut[MAXN], dfs_clock;
bool isbridge[MAXN];
vector<int> G[MAXN];
int cnt_bridge;
int father[MAXN];

void tarjan(int u, int fa) {
  father[u] = fa;
  low[u] = dfn[u] = ++dfs_clock;
  for (int i = 0; i < G[u].size(); i++) {
    int v = G[u][i];
    if (!dfn[v]) {
      tarjan(v, u);
      low[u] = min(low[u], low[v]);
      if (low[v] > dfn[u]) {
        isbridge[v] = true;
        ++cnt_bridge;
      }
    } else if (dfn[v] < dfn[u] && v != fa) {
      low[u] = min(low[u], dfn[v]);
```

```
    }
  }
}
```

3.5 网络流

3.5.1 Dinic

```
#include <bits/stdc++.h>

using namespace std;

int input(){
   int x=0,f=0;char ch=getchar();
   while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
   while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
   return f? -x:x;
}

#define N 20007
#define M 200007
#define INF 999999999
#define clr(a,b) memset(a,b,sizeof(a))

int n,m,s,t;
int cnt=0;
int head[N];
int dis[N],q[N];
struct edge{
   int v,w,next;
}e[M];

void Ins(int u,int v,int w){
   e[cnt]=(edge){v,w,head[u]};head[u]=cnt++;
   e[cnt]=(edge){u,0,head[v]};head[v]=cnt++;
}

int MkLevel(){
   int x,l=0,r=0;
   clr(dis,0);
   dis[s]=1,q[r++]=s;
   while(l<r){
      x=q[l++];
      for(int i=head[x];i!=-1;i=e[i].next){
         if(e[i].w&&!dis[e[i].v]){
            dis[e[i].v]=dis[x]+1;
            if(e[i].v==t) return 1;
            q[r++]=e[i].v;
         }
      }
   }
   return 0;
```

```
}

int extend(int s,int Lim){
   if(s==t) return Lim;
   int tmp,cost=0;
   for(int i=head[s];i!=-1;i=e[i].next){
      if(e[i].w&&dis[s]==dis[e[i].v]-1){
         tmp=extend(e[i].v,min(Lim-cost,e[i].w));
         if(tmp>0){
            e[i].w-=tmp,e[i^1].w+=tmp,cost+=tmp;
            if(Lim==cost) break;
         }else dis[e[i].v]=-1;
      }
   }
   return cost;
}

int Dinic(){
   int ans=0;
   while(MkLevel()) ans+=extend(s,INF);
   return ans;
}

void Solve(){
   n=input(),m=input(),s=input(),t=input();
   int u,v,w;
   cnt=0;
   clr(head,-1);
   for(int i=1;i<=m;i++){
      u=input(),v=input(),w=input();
      Ins(u,v,w);
   }
   printf("%d\n",Dinic());
}

int main(){
   Solve();
}
```

3.5.2 ISAP

```
struct Edge {
  int from, to, cap, flow;
  Edge(int u, int v, int c, int f) : from(u), to(v), cap(c), flow(f) {}
};

bool operator<(const Edge& a, const Edge& b) {
  return a.from < b.from || (a.from == b.from && a.to < b.to);
}

struct ISAP {
  int n, m, s, t;
```

```
12  vector<Edge> edges;
13  vector<int> G[maxn];
14  bool vis[maxn];
15  int d[maxn];
16  int cur[maxn];
17  int p[maxn];
18  int num[maxn];
19
20  void AddEdge(int from, int to, int cap) {
21    edges.push_back(Edge(from, to, cap, 0));
22    edges.push_back(Edge(to, from, 0, 0));
23    m = edges.size();
24    G[from].push_back(m - 2);
25    G[to].push_back(m - 1);
26  }
27
28  bool BFS() {
29    memset(vis, 0, sizeof(vis));
30    queue<int> Q;
31    Q.push(t);
32    vis[t] = 1;
33    d[t] = 0;
34    while (!Q.empty()) {
35      int x = Q.front();
36      Q.pop();
37      for (int i = 0; i < G[x].size(); i++) {
38        Edge& e = edges[G[x][i] ^ 1];
39        if (!vis[e.from] && e.cap > e.flow) {
40          vis[e.from] = 1;
41          d[e.from] = d[x] + 1;
42          Q.push(e.from);
43        }
44      }
45    }
46    return vis[s];
47  }
48
49  void init(int n) {
50    this->n = n;
51    for (int i = 0; i < n; i++) G[i].clear();
52    edges.clear();
53  }
54
55  int Augment() {
56    int x = t, a = INF;
57    while (x != s) {
58      Edge& e = edges[p[x]];
59      a = min(a, e.cap - e.flow);
60      x = edges[p[x]].from;
61    }
62    x = t;
63    while (x != s) {
64      edges[p[x]].flow += a;
```

```
    edges[p[x] ^ 1].flow -= a;
    x = edges[p[x]].from;
   }
   return a;
  }

  int Maxflow(int s, int t) {
   this->s = s;
   this->t = t;
   int flow = 0;
   BFS();
   memset(num, 0, sizeof(num));
   for (int i = 0; i < n; i++) num[d[i]]++;
   int x = s;
   memset(cur, 0, sizeof(cur));
   while (d[s] < n) {
    if (x == t) {
     flow += Augment();
     x = s;
    }
    int ok = 0;
    for (int i = cur[x]; i < G[x].size(); i++) {
     Edge& e = edges[G[x][i]];
     if (e.cap > e.flow && d[x] == d[e.to] + 1) {
      ok = 1;
      p[e.to] = G[x][i];
      cur[x] = i;
      x = e.to;
      break;
     }
    }
    if (!ok) {
     int m = n - 1;
     for (int i = 0; i < G[x].size(); i++) {
      Edge& e = edges[G[x][i]];
      if (e.cap > e.flow) m = min(m, d[e.to]);
     }
     if (--num[d[x]] == 0) break;
     num[d[x] = m + 1]++;
     cur[x] = 0;
     if (x != s) x = edges[p[x]].from;
    }
   }
   return flow;
  }
};
```

3.5.3 最小费用最大流

```
// dijkstra
#include <bits/stdc++.h>

```

```cpp
using namespace std;

#define ll long long
ll input(){
    ll x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

const int N=5007;
const int INF=0x3f3f3f3f;
#define PII pair <int,int>
#define fr first
#define sc second
#define mp make_pair

struct MinCostFlow{
    struct edge{
        int v,f,c,r;
        edge(int _v, int _f, int _c, int _r) :v(_v), f(_f), c(_c), r(_r) {}
    };
    int V=0,h[N],dis[N],pv[N],pe[N];
    vector <edge> G[N];

    inline void init(int n){
        for(int i=0;i<=V;i++) G[i].clear();
        V=n;
    }

    inline void Ins(int u,int v,int f,int c){
        G[u].push_back(edge(v,f,c,G[v].size() ));
        G[v].push_back(edge(u,0,-c,G[u].size()-1 ));
    }

    PII MCMF(int s,int t,int Flow){
        int cost=0,flow=0,newflow;fill(h,h+1+V,0);
        while(Flow){
            priority_queue <PII,vector<PII>,greater<PII> > Q;
            fill(dis,dis+V+1,INF);
            dis[s]=0,Q.push(mp(0,s));
            while(!Q.empty()){
                PII now=Q.top();Q.pop();
                int u=now.sc;
                if(dis[u]<now.fr) continue;
                for(int i=0;i<G[u].size();i++){
                    edge &e=G[u][i];
                    if(e.f>0&&dis[e.v]>dis[u]+e.c+h[u]-h[e.v]){
                        dis[e.v]=dis[u]+e.c+h[u]-h[e.v];
                        pv[e.v]=u,pe[e.v]=i;
                        Q.push(PII(dis[e.v],e.v));
                    }
                }
```

```
57          }
58          if(dis[t]==INF) break;
59          for(int i=0;i<=V;i++) h[i]+=dis[i];
60          newflow=Flow;
61          for(int x=t;x!=s;x=pv[x]) newflow=min(newflow,G[pv[x]][pe[x]].f);
62          Flow-=newflow,flow+=newflow,cost+=newflow*h[t];
63          for(int x=t;x!=s;x=pv[x]){
64              edge &e=G[pv[x]][pe[x]];
65              e.f-=newflow;
66              G[x][e.r].f+=newflow;
67          }
68      }
69      return mp(flow,cost);
70  }
71  }A;
72
73  int main(){
74      int n=input(),m=input(),s=input(),t=input();
75      A.init(n);
76      for(int i=1;i<=m;i++){
77          int u=input(),v=input(),w=input(),c=input();
78          A.Ins(u,v,w,c);
79      }
80      PII Ans=A.MCMF(s,t,INF);
81      printf("%d %d\n",Ans.fr,Ans.sc);
82  }
83
84  ////////////////////////////////////////////////////////////////////////////////
85
86  //spfa
87  #include <bits/stdc++.h>
88
89  using namespace std;
90
91  #define ll long long
92  ll input(){
93      ll x=0,f=0;char ch=getchar();
94      while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
95      while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
96      return f? -x:x;
97  }
98
99  const int N=5007;
100 const int M=50007;
101 #define clr(a,b) memset(a,b,sizeof(a))
102 #define PII pair <int,int>
103 #define fr first
104 #define sc second
105 #define mp make_pair
106 #define debug printf("PASS IN LINE:%d\n",__LINE__)
107
108 struct edge{
109     int v,w,c,next;
```

```
}e[M*2];
int head[N],cnt=-1;
int n,m;

void Ins(int u,int v, int w,int c){
    e[++cnt]=(edge){v,w,c,head[u]},head[u]=cnt;
    e[++cnt]=(edge){u,0,-c,head[v]},head[v]=cnt;
}

PII MCMF(int s,int t){
    int dis[N],pv[N],pe[N],flow[N],maxflow=0,mincost=0;
    bool vis[N];
    queue <int> q;
    while(1){
        clr(dis,0x3f),clr(flow,0x3f),clr(vis,0);
        q.push(s),vis[s]=1,dis[s]=0,pv[t]=-1;
        while(!q.empty()){
            int now=q.front();q.pop();
            vis[now]=0;
            for(int i=head[now];~i;i=e[i].next){
                if(e[i].w&&dis[now]+e[i].c<dis[e[i].v]){
                    dis[e[i].v]=dis[now]+e[i].c;
                    pv[e[i].v]=now,pe[e[i].v]=i;
                    flow[e[i].v]=min(flow[now],e[i].w);
                    if(!vis[e[i].v]){
                        vis[e[i].v]=1;
                        q.push(e[i].v);
                    }
                }
            }
        }

        if(pv[t]==-1) break;

        maxflow+=flow[t];
        mincost+=flow[t]*dis[t];

        for(int x=t;x!=s;x=pv[x]) e[pe[x]].w-=flow[t],e[pe[x]^1].w+=flow[t];
    }
    return mp(maxflow,mincost);
}

int main(){
    clr(head,-1),clr(e,-1),cnt=-1;
    int s,t;
    n=input(),m=input(),s=input(),t=input();
    for(int i=1;i<=m;i++){
        int u=input(),v=input(),w=input(),c=input();
        Ins(u,v,w,c);
    }
    PII Ans=MCMF(s,t);
    printf("%d %d\n",Ans.fr,Ans.sc);

```

```
163 }
164
165 ////////////////////////////////////////////////////////////////////////////
166
167
168 //MCMF->init(N) -> MCMF.add(u,v,flow,cost) -> (flow,cost) = MCMF.run(s,t,UP)
169 //其中UP参数表示求s到t的流量不超过UP的情况下的最小费用，无要求填INF即可
170 //所有节点的id范围[0,N]
171 const int MAXN = 10005;
172 const int INF = 0x7fffffff;
173 struct MinCostMaxFlow {
174     struct edge {
175         int v, f, c, r;
176         edge(int _v, int _f, int _c, int _r) :v(_v), f(_f), c(_c), r(_r) {}
177     };
178     int V = 0, H[MAXN + 5], dis[MAXN + 5], PreV[MAXN + 5], PreE[MAXN + 5];
179     vector<edge> G[MAXN + 5];
180     inline void init(int n) {
181         for (int i = 0; i <= V; ++i) G[i].clear();
182         V = n;
183     }
184     inline void add(int u, int v, int f, int c) {
185         G[u].push_back(edge(v, f, c, G[v].size()));
186         G[v].push_back(edge(u, 0, -c, G[u].size() - 1));
187     }
188     typedef pair<int, int> P;
189     P run(int s, int t, int f) {
190         int cost = 0, flow = 0, tf; fill(H, H + 1 + V, 0);
191         while (f) {
192             priority_queue <P, vector<P>, greater<P> > q;
193             fill(dis, dis + 1 + V, INF);
194             dis[s] = 0; q.push(P(0, s));
195             while (!q.empty()) {
196                 P now = q.top(); q.pop();
197                 int v = now.second;
198                 if (dis[v] < now.first)continue;
199                 for (int i = 0; i < G[v].size(); ++i) {
200                     edge& e = G[v][i];
201                     if (e.f > 0 && dis[e.v] > dis[v] + e.c + H[v] - H[e.v]) {
202                         dis[e.v] = dis[v] + e.c + H[v] - H[e.v];
203                         PreV[e.v] = v, PreE[e.v] = i;
204                         q.push(P(dis[e.v], e.v));
205                     }
206                 }
207             }
208             if (dis[t] == INF)break;
209             for (int i = 0; i <= V; ++i) H[i] += dis[i];
210             tf = f;
211             for (int v = t; v != s; v = PreV[v])
212                 tf = min(tf, G[PreV[v]][PreE[v]].f);
213             f -= tf, flow += tf, cost += tf*H[t];
214             for (int v = t; v != s; v = PreV[v]) {
215                 edge& e = G[PreV[v]][PreE[v]];
```

```
216             e.f -= tf;
217             G[v][e.r].f += tf;
218         }
219     }
220     return P(flow, cost);
221   }
222 }MCMF;
```

3.5.4 最小割

概念

割

对于一个网络流图 $G=(V,E)$，其割的定义为一种 ** 点的划分方式 ** ：将所有的点划分为 S 和 $T=V-S$ 两个集合，其中源点 $s\in S$，汇点 $t\in T$。

* *

割的容量

我们的定义割 (S,T) 的容量 $c(S,T)$ 表示所有从 S 到 T 的边的容量之和，即 $c(S,T)=\sum_{u\in S,v\in T}c(u,v)$。当然我们也可以用 $c(s,t)$ 表示 $c(S,T)$。

* *

最小割

最小割就是求得一个割 (S,T) 使得割的容量 $c(S,T)$ 最小。

* *

证明

最大流最小割定理

定理：$f(s,t)_{\max}=c(s,t)_{\min}$

对于任意一个可行流 $f(s,t)$ 的割 (S,T)，我们可以得到：

$$f(s,t)=S\text{出边的总流量}-S\text{入边的总流量}\leq S\text{出边的总流量}=c(s,t)$$

如果我们求出了最大流 f，那么残余网络中一定不存在 s 到 t 的增广路经，也就是 S 的出边一定是满流，S 的入边一定是零流，于是有：

$$f(s,t)=S\text{出边的总流量}-S\text{入边的总流量}=S\text{出边的总流量}=c(s,t)$$

结合前面的不等式，我们可以知道此时 f 已经达到最大。

* *

代码：

通过最大流最小割定理，我们可以直接得到如下代码：

```
1  #include <algorithm>
2  #include <cstdio>
3  #include <cstring>
4  #include <queue>
```

```
5
6   const int N = 1e4 + 5, M = 2e5 + 5;
7   int n, m, s, t, tot = 1, lnk[N], ter[M], nxt[M], val[M], dep[N], cur[N];
8
9   void add(int u, int v, int w) {
10    ter[++tot] = v, nxt[tot] = lnk[u], lnk[u] = tot, val[tot] = w;
11  }
12  void addedge(int u, int v, int w) { add(u, v, w), add(v, u, 0); }
13  int bfs(int s, int t) {
14    memset(dep, 0, sizeof(dep));
15    memcpy(cur, lnk, sizeof(lnk));
16    std::queue<int> q;
17    q.push(s), dep[s] = 1;
18    while (!q.empty()) {
19      int u = q.front();
20      q.pop();
21      for (int i = lnk[u]; i; i = nxt[i]) {
22        int v = ter[i];
23        if (val[i] && !dep[v]) q.push(v), dep[v] = dep[u] + 1;
24      }
25    }
26    return dep[t];
27  }
28  int dfs(int u, int t, int flow) {
29    if (u == t) return flow;
30    int ans = 0;
31    for (int &i = cur[u]; i && ans < flow; i = nxt[i]) {
32      int v = ter[i];
33      if (val[i] && dep[v] == dep[u] + 1) {
34        int x = dfs(v, t, std::min(val[i], flow - ans));
35        if (x) val[i] -= x, val[i ^ 1] += x, ans += x;
36      }
37    }
38    if (ans < flow) dep[u] = -1;
39    return ans;
40  }
41  int dinic(int s, int t) {
42    int ans = 0;
43    while (bfs(s, t)) {
44      int x;
45      while ((x = dfs(s, t, 1 << 30))) ans += x;
46    }
```

```
47      return ans;
48    }
49    int main() {
50      scanf("%d%d%d%d", &n, &m, &s, &t);
51      while (m--) {
52        int u, v, w;
53        scanf("%d%d%d", &u, &v, &w);
54        addedge(u, v, w);
55      }
56      printf("%d\n", dinic(s, t));
57      return 0;
58    }
```

方案：

我们可以通过从源点 s 开始 DFS ，每次走残量大于 0 的边，找到所有 S 点集内的点。

```
1  void dfs(int u) {
2    vis[u] = 1;
3    for (int i = lnk[u]; i; i = nxt[i]) {
4      int v = ter[i];
5      if (!vis[v] && val[i]) dfs(v);
6    }
7  }
```

* *

割边数量

只需要将每条边的容量变为 1 ，然后重新跑 Dinic 即可。

问题模型

有 n 个物品和两个集合 A, B ，如果将一个物品放入 A 集合会花费 a_i ，放入 B 集合会花费 b_i ；还有若干个形如 u_i, v_i, w_i 限制条件，表示如果 u_i 和 v_i 同时不在一个集合会花费 w_i 。每个物品必须且只能属于一个集合，求最小的代价。

这是一个经典的 ** 二者选其一 ** 的最小割题目。我们对于每个集合设置源点 s 和汇点 t ，第 i 个点由 s 连一条容量为 a_i 的边、向 t 连一条容量为 b_i 的边。对于限制条件 u, v, w ，我们在 u, v 之间连容量为 w 的双向边。

注意到当源点和汇点不相连时，代表这些点都选择了其中一个集合。如果将连向 s 或 t 的边割开，表示不放在 A 或 B 集合，如果把物品之间的边割开，表示这两个物品不放在同一个集合。

最小割就是最小花费。

4 字符串

4.1 字符串 hash

```cpp
//hdu1711
#include <bits/stdc++.h>

using namespace std;

#define ll long long
ll input(){
    ll x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

const int N=1e6+7;
int base[2]={19260817,(int)1e5+7},mod[2]={(int)1e9+7,998244353};
#define debug printf("PASS IN LINE:%d\n",__LINE__)

ll p[2][N],h[2][N];

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

ll gethash(int l,int r,int id){
    return ((h[id][r]-h[id][l-1]*p[id][r-l+1])%mod[id]+mod[id])%mod[id];
}

int main(){
    init();
    int T=input();
    while(T--){
        int n=input(),m=input(),x;
        for(int i=1;i<=n;i++){
            x=input()+N;
            h[0][i]=(h[0][i-1]*base[0]+x)%mod[0];
            h[1][i]=(h[1][i-1]*base[1]+x)%mod[1];
        }

        ll H[2]={0,0};

        for(int i=1;i<=m;i++){
            x=input()+N;
            H[0]=(H[0]*base[0]+x)%mod[0];
            H[1]=(H[1]*base[1]+x)%mod[1];
        }

        int Ans=-1;

        for(int i=1;i+m-1<=n;i++){
```

```

        if(gethash(i,i+m-1,0)==H[0]&&gethash(i,i+m-1,1)==H[1]){
            Ans=i;break;
        }
    }

    printf("%d\n",Ans);
  }
}
```

4.2 KMP

```cpp
#include <bits/stdc++.h>

using namespace std;

#define ll long long
ll input(){
  ll x=0,f=0;char ch=getchar();
  while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
  while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
  return f? -x:x;
}

const int N=1e6+7;

int nxt[N];

void get_nxt(char *s){
  int len=strlen(s+1);

  for(int i=2,j=0;i<=len;i++){
    while(j&&s[i]!=s[j+1]) j=nxt[j];
    if(s[i]==s[j+1]) j++;
    nxt[i]=j;
  }
}

char s[N],t[N];

int main(){
  scanf("%s%s",s+1,t+1);

  get_nxt(t);
  int ls=strlen(s+1),lt=strlen(t+1);

  for(int i=1,j=0;i<=ls;i++){
    while(j&&s[i]!=t[j+1]) j=nxt[j];
    if(s[i]==t[j+1]) j++;
    if(j==lt){printf("%d\n",i-j+1);j=nxt[j];}
  }

```

```
41      for(int i=1;i<=lt;i++) printf("%d%c",nxt[i],i==lt? '\n':' ');
42  }
```

4.3 tire 树

```
1   void insert(char *s)
2   {
3       int p = 0;
4       for(int i=0;s[i];i++)
5       {
6           int k = s[i] - 'a'; // 根据题意改动
7           if(!tr[p][k]) tr[p][k] = ++ tot;
8           p = tr[p][k];
9       }
10      vis[p] = 1; // 代表该节点有一个单词
11  }
12  void qu(char * s)
13  {
14      int p = 0 ;
15      for(int i=0;s[i];i++)
16      {
17          int k = s[i] - 'a' ; //根据题意改动
18          int now = tr[p][k];
19          if(vis[now]) ans ++ ;
20          p = tr[p][k];
21      }
22  }
```

4.4 AC 自动机

```
1   #include <bits/stdc++.h>
2
3   using namespace std;
4
5   #define ll long long
6   ll input(){
7       ll x=0,f=0;char ch=getchar();
8       while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
9       while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
10      return f? -x:x;
11  }
12
13  const int N=1e6+7;
14  int n;
15
16  namespace AC{
17      int tr[N][26],tot;
18      int e[N],fail[N];
19      void Ins(char *s){
20          int u=0;
21          for(int i=1;s[i];i++){
```

```
22          if(!tr[u][s[i]-'a'])tr[u][s[i]-'a']=++tot;
23          u=tr[u][s[i]-'a'];
24       }
25       e[u]++;
26    }
27
28    void build(){
29       queue<int> q;
30       for(int i=0;i<26;i++)
31          if(tr[0][i]) q.push(tr[0][i]);
32       while(!q.empty()){
33          int u=q.front();q.pop();
34          for(int i=0;i<26;i++){
35             if(tr[u][i])
36                fail[tr[u][i]]=tr[fail[u]][i],q.push(tr[u][i]);
37             else tr[u][i]=tr[fail[u]][i];
38          }
39       }
40    }
41
42    int query(char *t){
43       int u=0,res=0;
44       for(int i=1;t[i];i++){
45          u=tr[u][t[i]-'a'];
46          for(int j=u;j&&e[j]!=-1;j=fail[j]){
47             res+=e[j],e[j]=-1;
48          }
49       }
50       return res;
51    }
52 }
53 char s[N];
54 int main(){
55    n=input();
56    for(int i=1;i<=n;i++){
57       scanf("%s",s+1);
58       AC::Ins(s);
59    }
60    scanf("%s",s+1);
61    AC::build();
62    printf("%d\n",AC::query(s));
63    return 0;
64 }
65
66
67
68 //////////加强版////////////////
69 #include <bits/stdc++.h>
70
71 using namespace std;
72
73 #define ll long long
74 ll input(){
```

```
    ll x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

const int N=156,L=1e6+7;
namespace AC{
    const int SZ=N*80;
    int tot,tr[SZ][26];
    int fail[SZ],idx[SZ],val[SZ];
    int cnt[N];

    void init(){
        memset(fail,0,sizeof(fail));
        memset(tr,0,sizeof(tr));
        memset(val,0,sizeof(val));
        memset(cnt,0,sizeof(cnt));
        memset(idx,0,sizeof(idx));
        tot=0;
    }

    void Ins(char *s,int id){
        int u=0;
        for(int i=1;s[i];i++){
            if(!tr[u][s[i]-'a']) tr[u][s[i]-'a']=++tot;
            u=tr[u][s[i]-'a'];
        }
        idx[u]=id;
    }

    void build(){
        queue<int> q;
        for(int i=0;i<26;i++){
            if(tr[0][i]) q.push(tr[0][i]);
        }
        while(!q.empty()){
            int u=q.front();q.pop();
            for(int i=0;i<26;i++){
                if(tr[u][i])
                    fail[tr[u][i]]=tr[fail[u]][i],q.push(tr[u][i]);
                else
                    tr[u][i]=tr[fail[u]][i];
            }
        }
    }

    int query(char *t){
        int u=0,res=0;
        for(int i=1;t[i];i++){
            u=tr[u][t[i]-'a'];
            for(int j=u;j;j=fail[j]) val[j]++;
        }
```

```
128         for(int i=0;i<=tot;i++)
129             if(idx[i]) res=max(res,val[i]),cnt[idx[i]]=val[i];
130         return res;
131     }
132 }
133 
134 int n;
135 char s[N][100],t[L];
136 int main(){
137     while(~scanf("%d",&n)){
138         if(n==0) break;
139         AC::init();
140         for(int i=1;i<=n;i++)
141             scanf("%s",s[i]+1),
142             AC::Ins(s[i],i);
143         AC::build();
144         scanf("%s",t+1);
145         int x=AC::query(t);
146         printf("%d\n",x);
147         for(int i=1;i<=n;i++){
148             if(AC::cnt[i]==x) printf("%s\n",s[i]+1);
149         }
150     }
151     return 0;
152 }
```

4.5 后缀数组

```
1  // DA算法 时间复杂度O(nlogn)
2  #include<cstring>
3  #include<cstdio>
4  #include<algorithm>
5  using namespace std;
6  typedef long long ll;
7  const int N=100010;
8  int wa[N],wb[N],wv[N],wss[N],cal[N];
9  int rak[N],height[N],sa[N];
10 char s[N];
11 int cmp(int *r,int a,int b,int l)
12 {return r[a]==r[b]&&r[a+l]==r[b+l];}
13 //sa
14 void da(int *r,int *sa,int n,int M) {
15     int i,j,p,*x=wa,*y=wb,*t;
16     for(i=0;i<M;i++) wss[i]=0;
17     for(i=0;i<n;i++) wss[x[i]=r[i]]++;
18     for(i=1;i<M;i++) wss[i]+=wss[i-1];
19     for(i=n-1;i>=0;i--) sa[--wss[x[i]]]=i;
20     for(j=1,p=1;p<n;j*=2,M=p) {
21         for(p=0,i=n-j;i<n;i++) y[p++]=i;
22         for(i=0;i<n;i++) if(sa[i]>=j) y[p++]=sa[i]-j;
23         for(i=0;i<n;i++) wv[i]=x[y[i]];
24         for(i=0;i<M;i++) wss[i]=0;
```

```
25        for(i=0;i<n;i++) wss[wv[i]]++;
26        for(i=1;i<M;i++)wss[i]+=wss[i-1];
27        for(i=n-1;i>=0;i--) sa[--wss[wv[i]]]=y[i];
28        for(t=x,x=y,y=t,p=1,x[sa[0]]=0,i=1;i<n;i++)
29        x[sa[i]]=cmp(y,sa[i-1],sa[i],j)?p-1:p++;
30    }
31    return;
32 }
33 //height
34 void calheight(int *r,int *sa,int n) {
35    int i,j,k=0;
36    for(i=1;i<=n;i++) rak[sa[i]]=i;
37    for(i=0;i<n;height[rak[i++]]=k)
38    for(k?k--:0,j=sa[rak[i]-1];r[i+k]==r[j+k];k++);
39    for(int i=n;i;i--)rak[i]=rak[i-1],sa[i]++;
40 }
41 int n,k;
42 //RMQ
43 int lg[N], bin[30], d[32][N];
44 void rmq() {
45    bin[0]=1;lg[0]=-1;
46    for(int i=1;i<=30;i++)bin[i]=(bin[i-1]<<1);
47    for(int i=1;i < N;i++)lg[i]=lg[i>>1]+1;
48    int t = lg[n];
49    for(int i=1;i<=n;i++)d[0][i]=height[i];
50    for(int i=1;i<=t;i++)
51       for(int j=1;j+bin[i]-1<=n;j++)
52          d[i][j] = min(d[i-1][j], d[i-1][j+bin[i-1]]);
53 }
54 int query(int a, int b) {
55    a = rak[a], b = rak[b];
56    if (a > b) swap(a, b); ++a;
57    int t = lg[b-a+1];
58    return min(d[t][a], d[t][b-bin[t]+1]);
59 }
60 int ok(int x)
61 {
62    int cnt = 0 ;
63    for(int i=1;i<=n;i++)
64    {
65       if(height[i]>=x) cnt ++ ;
66       else {
67          if(cnt + 1 >= k) return 1;
68          cnt = 0;
69       }
70    }
71    return 0 ;
72 }
73 int main(){
74    int cas=1;
75    scanf("%d%d",&n,&k);
76    for(int i=1;i<=n;i++)
77       scanf("%d",&cal[i]);
```

```
78      cal[n+1]=0;//不能删除
79      da(cal+1,sa,n+1,200);
80      calheight(cal+1,sa,n);
81      // rmq();
82      int l = 1 ,r = n , ans = 0;
83      while(l<=r){
84          int mid = (l+r)/2;
85          if(ok(mid)){
86            ans = mid ;
87            l = mid + 1;
88          }else r = mid - 1;
89      }
90      printf("%d\n",ans);
91  }
92
93
94  ////////////////////////////////////////////////////
95
96  //DC3算法,时间复杂度O(n),空间复杂度O(3*n)
97
98
99  #include <cstdio>
100 #include <cstring>
101 #include <algorithm>
102 #define F(x) ((x) / 3 + ((x) % 3 == 1 ? 0 : tb))
103 #define G(x) ((x) < tb ? (x) * 3 + 1 : ((x) - tb) * 3 + 2)
104 using namespace std;
105 const int N = 3*(1e5+5);
106 int wa[N], wb[N], ws[N], wv[N], sa[N];
107 int rak[N], height[N], cal[N],n;
108 char s[N],ans[N],s1[N];
109 int cnt[N];
110 int c0(int *r, int a, int b) {
111     return r[a] == r[b] && r[a + 1] == r[b + 1] && r[a + 2] == r[b + 2];
112 }
113 int c12(int k, int *r, int a, int b) {
114     if (k == 2)
115         return r[a] < r[b] || r[a] == r[b] && c12(1, r, a + 1, b + 1);
116     return r[a] < r[b] || r[a] == r[b] && wv[a + 1] < wv[b + 1];
117 }
118 void Rsort(int *r, int *a, int *b, int n, int m) {
119     for (int i = 0; i < n; i++) wv[i] = r[a[i]];
120     for (int i = 0; i < m; i++) ws[i] = 0;
121     for (int i = 0; i < n; i++) ws[wv[i]]++;
122     for (int i = 1; i < m; i++) ws[i] += ws[i - 1];
123     for (int i = n - 1; i >= 0; i--) b[--ws[wv[i]]] = a[i];
124 }
125 void dc3(int *r, int *sa, int n, int m) {
126     int i, j, *rn = r + n, *san = sa + n, ta = 0, tb = (n + 1) / 3, tbc = 0, p;
127     r[n] = r[n + 1] = 0;
128     for (i = 0; i < n; i++) if (i % 3 != 0) wa[tbc++] = i;
129     Rsort(r + 2, wa, wb, tbc, m);
130     Rsort(r + 1, wb, wa, tbc, m);
```

```
    Rsort(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;
    Rsort(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 calheight(int *r, int *sa, int n) {
    int i, j, k = 0;
    for (i = 1; i <= n; i++) rak[sa[i]] = i;
    for (i = 0; i < n; height[rak[i++]] = k)
        for (k ? k-- : 0, j = sa[rak[i] - 1]; r[i + k] == r[j + k]; k++);
    for(int i=n;i;i--) rak[i]=rak[i-1];
    for(int i=n;i;i--) sa[i]++;
}
int main() {
    while (scanf("%s", s+1)&&s[1] != '#') {
        n = strlen(s+1);
        for (int i = 1; i <= n; i++)
            cal[i] = s[i] - 'a' + 1;
        cal[n+1] = 0;
        dc3(cal+1, sa, n + 1, 30);
        calheight(cal+1, sa, n);
        for(int i=1;i<=n;i++) {
            printf("%d ",height[i]);
        }puts("");
    }
    return 0;
}
```

5 动态规划

5.1 背包问题

```
#include<iostream>
#include<cstdio>
#define DEBUG printf("PASS [%s] IN LINE %d\n",__FUNCTION__,__LINE__)
using namespace std;
int input(){
    int f=1,x=0;char ch=getchar();
    while(ch<'0'||ch>'9'){if(ch=='-') f=-1;ch=getchar();}
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f*x;
}
```

```
11 //有N件物品和一个容量为V的背包。第i件物品的费用是c[i]，价值是w[i]。求解将哪些物品装入背包可使价值总和最大。
12 class bag1{
13    public:
14       void solve(){
15          n=input();v=input();
16          int c,w;
17          for(int i=1;i<=n;i++){
18                c=input();w=input();
19             for(int j=v;j>=c;j--){
20                dp[j]=dp[j]>dp[j-c]+w? dp[j]:dp[j-c]+w;
21             }
22          }
23          printf("%lld",dp[v]);
24       }
25    private:
26       int dp[1007];
27       int n,v;
28 };
29 //有N种物品和一个容量为V的背包，每种物品都有无限件可用。第i种物品的费用是c[i]，价值是w[i]。求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，且价值总和最大。
30 class bag2{
31    public:
32       void solve(){
33          n=input();v=input();
34          int c,w;
35          for(int i=1;i<=n;i++){
36                c=input();w=input();
37             for(int j=c;j<=v;j++){
38                dp[j]=dp[j]<dp[j-c]+w? dp[j]:dp[j-c]+w;
39             }
40          }
41          printf("%d\n",dp[v]);
42       }
43    private:
44       int dp[1007];
45       int n,v;
46 };
47 // 有N种物品和一个容量为V的背包。第i种物品最多有n[i]件可用，每件费用是c[i]，价值是w[i]。求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，且价值总和最大。
48 class bag3{
49    public:
50       void solve(){
51          l=1,r=0;
52          n=input();v=input();
53          int c,w,k;
54          for(int i=1;i<=n;i++){
55             c=input();w=input();k=input();
56             if(v/c<k) k=v/c;
57             for(int d=0;d<c;d++){
58                l=1,r=0;
59                for(int j=0;j<=(v-d)/c;j++){
60                   insert(j,dp[j*c+d]-j*w);
```

```
61                      if(Q[l]<j-k) l++;
62                      dp[j*c+d]=q[l]+j*w;
63                  }
64              }
65          }
66          printf("%d\n",dp[v]);
67      }
68  private:
69      int n,v;
70      int dp[1007];
71      int q[1007],Q[1007];
72      int l,r;
73      void insert(int x,int y){while(l<=r&&(q[r]<y))r--;q[++r]=y;Q[r]=x;}
74  };
75  // 如果将P01、P02、P03混合起来。也就是说，有的物品只可以取一次（01背包），有的物品可以取无
        限次（完全背包），有的物品可以取的次数有一个上限（多重背包）。应该怎么求解呢？
76  class bag4{
77  public:
78      void solve(){
79          l=1,r=0;
80          n=input();v=input();
81          int w,c,k;
82          for(int i=1;i<=n;i++){
83              c=input();w=input();k=input();
84              if(k==1) solve1(c,w);
85              if(k==-1) solve2(c,w);
86              if(k>1) solve3(c,w,k);
87          }
88          printf("%d\n",dp[v]);
89      }
90  private:
91      int n,v;
92      int dp[1007];
93      int q[1007],Q[1007];
94      int l,r;
95      void insert(int x,int y){while(l<=r&&(q[r]<y))r--;q[++r]=y;Q[r]=x;}
96      void solve1(int c,int w){
97          for(int j=v;j>=c;j--){
98                  dp[j]=dp[j]>dp[j-c]+w? dp[j]:dp[j-c]+w;
99              }
100     }
101     void solve2(int c,int w){
102         for(int j=c;j<=v;j++){
103                 dp[j]=dp[j]>dp[j-c]+w? dp[j]:dp[j-c]+w;
104             }
105     }
106     void solve3(int c,int w,int k){
107         if(v/c<k) k=v/c;
108         for(int d=0;d<c;d++){
109             l=1,r=0;
110             for(int j=0;j<=(v-d)/c;j++){
111                 insert(j,dp[j*c+d]-j*w);
112                 if(Q[l]<j-k) l++;
```

```
                dp[j*c+d]=q[l]+j*w;
            }
        }
    }
};
// 二维费用的背包问题是指：对于每件物品，具有两种不同的费用；选择这件物品必须同时付出这两种代价；对于每种代价都有 一个可付出的最大值（背包容量）。问怎样选择物品可以得到最大的价值。设这两种代价分别为代价1和代价2，第i件物品所需的两种代价分别为a[i]和 b[i]。两种代价可付出的最大值（两种背包容量）分别为V和U。物品的价值为w[i]。
class bag5{
   public:
      void solve(){
         vmax=input();mmax=input();n=input();
            int v,m,k;
         for(int l=1;l<=n;l++){
            v=input();m=input();k=input();
            for(int i=vmax;i>=v;i--){
               for(int j=mmax;j>=m;j--){
                  dp[i][j]=dp[i][j]>dp[i-v][j-m]+k? dp[i][j]:dp[i-v][j-m]+k;
               }
            }
         }
         printf("%d",dp[vmax][mmax]);
      }
   private:
      int dp[1007][1007];
   int n;
   int vmax,mmax;
};
// 有N件物品和一个容量为V的背包。第i件物品的费用是c[i]，价值是w[i]。这些物品被划分为若干组，每组中的物品互相冲突，最多选一件。求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，且价值总和最大。
class bag6{
   public:
      void solve(){
         k=input();v=input();
         int cc,w,c;
         for(int l=1;l<=k;l++){
            cc=input();
            for(int j=v;j>=0;j--){
               for(int i=1;i<=cc;i++){
                  w=input();c=input();
                  dp[j]=dp[j]>dp[j-c]+w? dp[j]:dp[j-c]+w;
               }
            }
         }
         printf("%d\n",dp[v]);
      }
   private:
      int dp[1007];
      int v,k;
};
//bag1 bag1;
```

```
//bag2 bag2;
//bag3 bag3;
//bag4 bag4;
//bag5 bag5;
//bag6 bag6;
int main(){
// bag1.solve();
// bag2.solve();
// bag3.solve();
// bag4.solve();
// bag5.solve();
// bag6.solve();
}
```

5.2 数位 dp

5.2.1 数位 dp 模板

```
/*
HDU 2089 不要62
入门题。就是数位上不能有4也不能有连续的62，没有4的话在枚举的时候判断一下，不枚举4就可以保证状态合法了，所以这个约束没有记忆化的必要，而对于62的话，涉及到两位，当前一位是6或者不是6这两种不同情况我计数是不相同的，所以要用状态来记录不同的方案数。
dp[pos][sta]表示当前第pos位，前一位是否是6的状态，这里sta只需要去0和1两种状态就可以了，不是6的情况可视为同种，不会影响计数。
*/
#include<iostream>
#include<cstdio>
#include<cstring>
#include<string>
using namespace std;
typedef long long ll;
int a[20];
int dp[20][2];
int dfs(int pos,int pre,int sta,bool limit)
{
    if(pos==-1) return 1;
    if(!limit && dp[pos][sta]!=-1) return dp[pos][sta];
    int up=limit ? a[pos] : 9;
    int tmp=0;
    for(int i=0;i<=up;i++)
    {
        if(pre==6 && i==2)continue;
        if(i==4) continue;//都是保证枚举合法性
        tmp+=dfs(pos-1,i,i==6,limit && i==a[pos]);
    }
    if(!limit) dp[pos][sta]=tmp;
    return tmp;
}
int solve(int x)
{
    int pos=0;
```

```
    while(x)
    {
        a[pos++]=x%10;
        x/=10;
    }
    return dfs(pos-1,-1,0,true);
}
int main()
{
    int le,ri;
    //memset(dp,-1,sizeof dp);可优化
    while(~scanf("%d%d",&le,&ri) && le+ri)
    {
        memset(dp,-1,sizeof dp);
        printf("%d\n",solve(ri)-solve(le-1));
    }
    return 0;
}
```

5.2.2 数位 dp 例题

```
typedef long long ll;
int a[20];
ll dp[20][state];//不同题目状态不同
ll dfs(int pos,/*state变量*/,bool lead/*前导零*/,bool limit/*数位上界变量*/)//不是每个题都要判断前导零
{
    //递归边界，既然是按位枚举，最低位是0，那么pos==-1说明这个数我枚举完了
    if(pos==-1) return 1;/*这里一般返回1，表示你枚举的这个数是合法的，那么这里就需要你在枚举时必须每一位都要满足题目条件，也就是说当前枚举到pos位，一定要保证前面已经枚举的数位是合法的。不过具体题目不同或者写法不同的话不一定要返回1 */
    //第二个就是记忆化(在此前可能不同题目还能有一些剪枝)
    if(!limit && !lead && dp[pos][state]!=-1) return dp[pos][state];
    /*常规写法都是在没有限制的条件记忆化，这里与下面记录状态是对应，具体为什么是有条件的记忆化后面会讲*/
    int up=limit?a[pos]:9;//根据limit判断枚举的上界up;这个的例子前面用213讲过了
    ll ans=0;
    //开始计数
    for(int i=0;i<=up;i++)//枚举，然后把不同情况的个数加到ans就可以了
    {
        if() ...
        else if()...
        ans+=dfs(pos-1,/*状态转移*/,lead && i==0,limit && i==a[pos]) //最后两个变量传参都是这样写的
        /*这里还算比较灵活，不过做几个题就觉得这里也是套路了
        大概就是说，我当前数位枚举的数是i，然后根据题目的约束条件分类讨论
        去计算不同情况下的个数，还有要根据state变量来保证i的合法性，比如题目
        要求数位上不能有62连续出现,那么就是state就是要保存前一位pre,然后分类，
        前一位如果是6那么这意味就不能是2，这里一定要保存枚举的这个数是合法*/
    }
    //计算完，记录状态
    if(!limit && !lead) dp[pos][state]=ans;
```

```
    /*这里对应上面的记忆化，在一定条件下时记录，保证一致性，当然如果约束条件不需要考虑lead，
        这里就是lead就完全不用考虑了*/
    return ans;
}
ll solve(ll x)
{
    int pos=0;
    while(x)//把数位都分解出来
    {
        a[pos++]=x%10;//个人老是喜欢编号为[0,pos),看不惯的就按自己习惯来，反正注意数位边界
            就行
        x/=10;
    }
    return dfs(pos-1/*从最高位开始枚举*/,/*一系列状态 */,true,true);//刚开始最高位都是有
        限制并且有前导零的，显然比最高位还要高的一位视为0嘛
}
int main()
{
    ll le,ri;
    while(~scanf("%lld%lld",&le,&ri))
    {
        //初始化dp数组为-1,这里还有更加优美的优化,后面讲
        printf("%lld\n",solve(ri)-solve(le-1));
    }
}
```

5.3 树形 dp

基本的 dp 方程

选择节点类

$$\begin{cases} dp[i][0] = dp[j][1] \\ dp[i][1] = \max / \min(dp[j][0], dp[j][1]) \end{cases}$$

树形背包类

$$\begin{cases} dp[v][k] = dp[u][k] + val \\ dp[u][k] = max(dp[u][k], dp[v][k-1]) \end{cases}$$

以下是一些树形 dp 的例题形式。

5.3.1 例题 1(换根)

```
#include <bits/stdc++.h>

using namespace std;

#define ll long long
```

```
ll input(){
    ll x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

#define debug printf("PASS IN LINE:%d\n",__LINE__)

const int N=1e5+7;

struct node{
    int x,y,len;
}d[N];

vector <int> G[N];

vector <node> nxt[N];

int ans[N];

void Ins(int u,int v){
    G[u].push_back(v);
    G[v].push_back(u);
}

int top[N],dep[N],fa[N],siz[N],son[N];;

namespace LCA{
    void dfs1(int now,int father){
        dep[now]=dep[father]+1;
        siz[now]=1;
        fa[now]=father;
        for(auto v:G[now]){
            if(v==father) continue;
            dfs1(v,now);
            siz[now]+=siz[v];
            if(!son[now]||siz[v]>siz[son[now]])
                son[now]=v;
        }
    }

    void dfs2(int now,int tp){
        top[now]=tp;
        if(!son[now]) return;
        dfs2(son[now],tp);
        for(auto v:G[now]){
            if(v==fa[now]||v==son[now]) continue;
            dfs2(v,v);
        }
    }
}

```

```
59
60  int lca(int u,int v){
61      while(top[u]!=top[v]){
62          if(dep[top[u]]<dep[top[v]]) swap(u,v);
63          u=fa[top[u]];
64      }
65      return dep[u]>dep[v]? v:u;
66  }
67
68  int dis(int u,int v){
69      return dep[u]+dep[v]-2*dep[lca(u,v)]+1;
70  }
71
72  node merge(node a,node b){
73      node tmp=a.len>b.len? a:b;
74
75      if(dis(a.x,b.y)>tmp.len)
76          tmp=(node){a.x,b.y,dis(a.x,b.y)};
77
78      if(dis(a.x,b.x)>tmp.len)
79          tmp=(node){a.x,b.x,dis(a.x,b.x)};
80
81      if(dis(a.y,b.y)>tmp.len)
82          tmp=(node){a.y,b.y,dis(a.y,b.y)};
83
84      if(dis(a.y,b.x)>tmp.len)
85          tmp=(node){a.y,b.x,dis(a.y,b.x)};
86
87      return tmp;
88  }
89
90  void dfs1(int u){
91      d[u]=(node){u,u,1};
92      for(auto v:G[u]){
93          if(v==fa[u]) continue;
94          dfs1(v);
95          d[u]=merge(d[u],d[v]);
96      }
97  }
98
99  void putans(int x,int y){
100     if(x>y) swap(x,y);
101     ans[x]=max(ans[x],y);
102 }
103
104 void dfs2(int u){
105     nxt[u].push_back(d[u]);
106     for(int i=G[u].size()-1;i>=0;i--){
107         if(G[u][i]==fa[u])continue;
108         nxt[u].push_back(merge(nxt[u][nxt[u].size()-1],d[G[u][i]]));
109     }
110
111     node pre=d[u];
```

```
112     int i=nxt[u].size()-2;
113     for(auto v:G[u]){
114         if(v==fa[u]) continue;
115         node tmp=merge(pre,nxt[u][i]);
116         putans(d[v].len,tmp.len);
117         pre=merge(pre,d[v]);
118         d[v]=merge(tmp,(node){v,v,1});
119         dfs2(v);
120         i--;
121     }
122 }
123 
124 int main(){
125     int T=input();
126     while(T--){
127         int n=input();
128 
129         for(int i=1;i<=n+3;i++)
130             G[i].clear(),nxt[i].clear(),ans[i]=0,son[i]=0;
131 
132         for(int i=1;i<n;i++){
133             int u=input(),v=input();
134             Ins(u,v);
135         }
136 
137         LCA::dfs1(1,-1);
138         LCA::dfs2(1,1);
139 
140         dfs1(1);
141         d[1]=(node){1,1,1};
142         dfs2(1);
143 
144         ll Ans=0;
145         for(int i=n;i;i--){
146             ans[i]=max(ans[i],ans[i+1]);
147             if(ans[i]) Ans+=1+(ans[i]-i)*2;
148         }
149 
150 
151         printf("%lld\n",Ans);
152     }
153 }
```

5.3.2 例题 2(换根)

```
1  #include <bits/stdc++.h>
2  
3  using namespace std;
4  
5  #define ll long long
6  ll input(){
7      ll x=0,f=0;char ch=getchar();
```

```
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

#define PII pair <ll,ll>
#define fr first
#define sc second
#define mp make_pair

const int N=1e5+7;
const ll inf=1e18;

vector<int> G[N];
ll dp[N][2];
ll c[N];
vector<PII> nxt[N];
int n;
ll Ans;

void dfs1(int u,int fa){
    dp[u][0]=inf,dp[u][1]=-inf;
    for(auto v:G[u]){
        if(v==fa) continue;
        dfs1(v,u);
        dp[u][0]=min(dp[u][0],dp[v][1]);
        dp[u][1]=max(dp[u][1],dp[v][0]);
    }
    if(dp[u][0]==inf)
        dp[u][0]=dp[u][1]=0;
    dp[u][0]+=c[u],dp[u][1]+=c[u];
}


void dfs2(int u,int fa){
    dp[u][0]=inf,dp[u][1]=-inf;
    for(int i=G[u].size()-1;i>=0;i--){
        int v=G[u][i];
        dp[u][0]=min(dp[u][0],dp[v][1]);
        dp[u][1]=max(dp[u][1],dp[v][0]);
        nxt[u].push_back(mp(dp[u][0],dp[u][1]));
    }

    dp[u][0]+=c[u];
    dp[u][1]+=c[u];
    Ans=max(Ans,dp[u][0]);

    int i=G[u].size()-1;
    ll pre1=inf,pre2=-inf;

    for(auto v:G[u]){
        dp[u][0]=pre1;
        dp[u][1]=pre2;
```

```
61          if(i){
62              dp[u][0]=min(dp[u][0],nxt[u][i-1].fr);
63              dp[u][1]=max(dp[u][1],nxt[u][i-1].sc);
64          }
65          if(dp[u][0]==inf) dp[u][0]=dp[u][1]=0;
66          dp[u][0]+=c[u];
67          dp[u][1]+=c[u];
68          pre1=min(pre1,dp[v][1]);
69          pre2=max(pre2,dp[v][0]);
70          if(v!=fa)
71              dfs2(v,u);
72          i--;
73      }
74  }
75
76  int main(){
77      int T=input();
78      while(T--){
79          n=input();
80          for(int i=1;i<=n;i++)
81              c[i]=input();
82          for(int i=1;i<=n;i++)
83              c[i]-=input();
84          for(int i=1;i<=n;i++)
85              G[i].clear(),nxt[i].clear();
86
87          for(int i=1;i<n;i++){
88              int u=input(),v=input();
89              G[u].push_back(v);
90              G[v].push_back(u);
91          }
92
93          Ans=-inf;
94
95          dfs1(1,0);
96          dfs2(1,0);
97
98          printf("%lld\n",Ans);
99      }
100 }
```

5.3.3 例题 3(树上背包型 dp)

```
1  //二叉苹果树
2  //设f[u][i]表示uu的子树上保留ii条边，至多保留的苹果数目
3  #include <bits/stdc++.h>
4
5  using namespace std;
6
7  #define ll long long
8  ll input(){
9      ll x=0,f=0;char ch=getchar();
```

```
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

const int N=107;

struct egde{
    int v,w,next;
}e[N*2];
int head[N],cnt=0;

void Ins(int u,int v,int w){
    e[++cnt]=(egde){v,w,head[u]},head[u]=cnt;
    e[++cnt]=(egde){u,w,head[v]},head[v]=cnt;
}

int siz[N];

int dp[N][N];

int n,q;

void dfs(int u,int fa){
    for(int i=head[u];i;i=e[i].next){
        int v=e[i].v,w=e[i].w;
        if(v==fa) continue;
        dfs(v,u);
        siz[u]+=siz[v]+1;
        for(int j=min(siz[u],q);j>=0;j--){
            for(int k=min(siz[v],j-1);k>=0;k--){
                dp[u][j]=max(dp[u][j],dp[u][j-k-1]+dp[v][k]+w);
            }
        }
    }
}

int main(){
    n=input(),q=input();

    for(int i=1;i<n;i++){
        int u=input(),v=input(),w=input();
        Ins(u,v,w);
    }

    dfs(1,0);

    printf("%d\n",dp[1][q]);
}
```

5.3.4 例题 4(树上背包型 dp)

```cpp
#include <bits/stdc++.h>

using namespace std;

#define ll long long
ll input(){
    ll x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

const int N=2007;

struct edge{
    int v,w,next;
}e[N<<1];
int head[N],cnt;

void Ins(int u,int v,int w){
    e[++cnt]=(edge){v,w,head[u]},head[u]=cnt;
    e[++cnt]=(edge){u,w,head[v]},head[v]=cnt;
}

ll dp[N][N];
int siz[N];
int n,m;

void dfs(int u,int fa){
    siz[u]=1;dp[u][0]=dp[u][1]=0;
    for(int i=head[u];i;i=e[i].next){
        int v=e[i].v,w=e[i].w;
        if(v==fa) continue;
        dfs(v,u);siz[u]+=siz[v];
        for(int j=min(siz[u],m);j>=0;j--){
            if(dp[u][j]!=-1) dp[u][j]+=dp[v][0]+1ll*siz[v]*(n-m-siz[v])*w;
            for(int k=min(siz[v],j);k>0;k--){
                if(dp[u][j-k]==-1) continue;
                ll tot=1ll*(m-k)*k+(n-m-(siz[v]-k))*(siz[v]-k);
                dp[u][j]=max(dp[u][j],dp[u][j-k]+dp[v][k]+tot*w);
            }
        }
    }
}

int main(){
    memset(dp,-1,sizeof dp);

    n=input(),m=input();

    for(int i=1;i<n;i++){
        int u=input(),v=input(),w=input();

```

```
54          Ins(u,v,w);
55      }
56
57      dfs(1,0);
58
59      printf("%lld\n",dp[1][m]);
60  }
```

5.3.5 例题 5(维护树链)

题意让我们用最少的代价把叶子节点到根节点的距离调成相同

显然，我们调整靠近根节点的树枝，其下叶子节点距离根节点的距离都会增加，所以，调整越靠根节点的树枝调整的代价越少。

为了方便作图，效果直观，在此我们用节点深度类比距离。

所以我们可以先找到最深的叶子节点。

再从最小的子树开始，把所有子节点调整到同一深度，再调整子树上面的树枝

```
1   #include <bits/stdc++.h>
2
3   using namespace std;
4
5   #define ll long long
6   ll input(){
7       ll x=0,f=0;char ch=getchar();
8       while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
9       while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
10      return f? -x:x;
11  }
12
13  const int N=5e5+7;
14
15  struct edge{
16      ll v,w,next;
17  }e[N<<1];
18
19  int head[N],cnt;
20
21  void Ins(int u,int v,int w){
22      e[++cnt]=(edge){v,w,head[u]},head[u]=cnt;
23      e[++cnt]=(edge){u,w,head[v]},head[v]=cnt;
24  }
25
26  ll dp[N],Ans=0;
27
28  void dfs(int u,int fa){
29      dp[u]=0;
30      for(int i=head[u];i;i=e[i].next){
31          ll v=e[i].v,w=e[i].w;
32          if(v==fa) continue;
33          dfs(v,u);
34          dp[u]=max(dp[u],dp[v]+w);
```

```
    }

    for(int i=head[u];i;i=e[i].next){
        int v=e[i].v,w=e[i].w;
        if(v==fa) continue;
        Ans+=dp[u]-(dp[v]+w);
    }
}

int main(){
    int n=input(),s=input();

    for(int i=1;i<n;i++){
        int u=input(),v=input(),w=input();
        Ins(u,v,w);
    }

    dfs(s,0);

    printf("%lld\n",Ans);
}
```

5.3.6 例题 6(维护树链)

```
// 二分可以选的边的上限选择一些边进行dp，类似与例题5
#include <bits/stdc++.h>

using namespace std;

#define ll long long
ll input(){
    ll x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

const int N=1007;

int dp[N];

struct edge{
    int v,w,next;
}e[N<<1];
int head[N],cnt=0;

void Ins(int u,int v,int w){
    e[++cnt]=(edge){v,w,head[u]},head[u]=cnt;
    e[++cnt]=(edge){u,w,head[v]},head[v]=cnt;
}

void dfs(int u,int fa,int limit){
```

```
    dp[u]=0;int flag=0;
    for(int i=head[u];i;i=e[i].next){
        int v=e[i].v,w=e[i].w;
        if(v==fa) continue;
        flag=1;
        dfs(v,u,limit);
        if(w<=limit) dp[u]+=min(dp[v],w);
        else dp[u]+=dp[v];
    }
    if(!flag) dp[u]=1000001;
}

int main(){
    while(1){
        int n=input(),m=input();
        if(n==0&&m==0) break;
        for(int i=1;i<=n;i++){
            head[i]=0;
        }cnt=0;

        for(int i=1;i<n;i++){
            int u=input(),v=input(),w=input();
            Ins(u,v,w);
        }

        dfs(1,0,m);
        if(dp[1]>m){
            printf("-1\n");
            continue;
        }

        int l=1,r=m,Ans;
        while(l<=r){
            int mid=(l+r)>>1;
            dfs(1,0,mid);
            if(1<=dp[1]&&dp[1]<=m) Ans=mid,r=mid-1;
            else l=mid+1;
        }

        printf("%d\n",Ans);
    }
}
```

5.3.7 例题 7(类似于换根)

```
#include <bits/stdc++.h>

using namespace std;

#define ll long long
ll input(){
    ll x=0,f=0;char ch=getchar();
```

```cpp
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

const int N=2e5+7;

struct edge{
    int _u,_v,v,w,next;
}e[N<<1];
int head[N],cnt;

void Ins(int u,int v){
    e[++cnt]=(edge){u,v,v,0,head[u]},head[u]=cnt;
    e[++cnt]=(edge){u,v,u,0,head[v]},head[v]=cnt;
}

int dp[N];
int Ans[N];

void dfs1(int u,int fa){
    for(int i=head[u];i;i=e[i].next){
        int v=e[i].v,w;
        if(e[i]._u==u&&e[i]._v==v) w=e[i].w=0;
        else w=e[i].w=1;
        if(v==fa) continue;
        dfs1(v,u);
        dp[u]+=dp[v]+w;
    }
}

void dfs2(int u,int fa){
    Ans[u]=dp[u];
    for(int i=head[u];i;i=e[i].next){
        int v=e[i].v,w=e[i].w;
        if(v==fa) continue;
        int tmp1=dp[u],tmp2=dp[v];
        dp[u]=dp[u]-dp[v]-w;
        dp[v]=dp[v]+dp[u]+(!w);
        dfs2(v,u);
        dp[u]=tmp1;
        dp[v]=tmp2;
    }
}

vector<int> ans;

int main(){
    int n=input();

    for(int i=1;i<n;i++){
        int u=input(),v=input();
        Ins(u,v);
```

```
    }

    dfs1(1,0);
    dfs2(1,0);

    int mi=5e5;
    for(int i=1;i<=n;i++){
        if(Ans[i]<mi){
            mi=Ans[i];
            ans.clear();
            ans.push_back(i);
        }else if(Ans[i]==mi){
            ans.push_back(i);
        }
    }

    printf("%d\n",mi);
    for(int i=0;i<ans.size();i++) printf("%d%c",ans[i],i==ans.size()-1? '\n':' ');
}
```

5.3.8 例题 8(维护树链)

```
//维护当前节点向下的最大值和次大值，并且用这些信息去更新向上的节点信息
// #include <bits/stdc++.h>
#include <iostream>
#include <cstdio>
#include <cstring>

using namespace std;

#define ll long long
ll input(){
    ll x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

const int N=1e6+7;

struct edge{
    ll v,w,next;
}e[N<<1];
ll head[N],cnt=0;

void Ins(ll u,ll v,ll w){
    e[++cnt]=(edge){v,w,head[u]};head[u]=cnt;
    e[++cnt]=(edge){u,w,head[v]};head[v]=cnt;
}

ll dp[N][3];

```

```
31 void dfs(ll u,ll fa){
32    for(ll i=head[u];i;i=e[i].next){
33       ll v=e[i].v,w=e[i].w;
34       if(v==fa) continue;
35       dfs(v,u);
36       if(dp[v][0]+w>=dp[u][0]){
37          dp[u][1]=dp[u][0];
38          dp[u][0]=dp[v][0]+w;
39       }else if(dp[v][0]+w>dp[u][1]){
40          dp[u][1]=dp[v][0]+w;
41       }
42    }
43 }
44 
45 void dfs2(ll u,ll fa){
46    for(ll i=head[u];i;i=e[i].next){
47       ll v=e[i].v,w=e[i].w;
48       if(v==fa) continue;
49       if(dp[v][0]+w==dp[u][0]) dp[v][2]=max(dp[u][1],dp[u][2])+w;
50       else dp[v][2]=max(dp[u][0],dp[u][2])+w;
51       dfs2(v,u);
52    }
53 }
54 
55 ll ans[N];
56 ll Ans;
57 
58 ll mx[N*4],mi[N*4];
59 
60 void build(ll rt,ll l,ll r){
61    if(l==r){
62       mx[rt]=ans[l];mi[rt]=ans[l];
63       return;
64    }
65    ll mid=(l+r)>>1;
66    build(rt<<1,l,mid);build(rt<<1|1,mid+1,r);
67    mx[rt]=max(mx[rt<<1],mx[rt<<1|1]);
68    mi[rt]=min(mi[rt<<1],mi[rt<<1|1]);
69 }
70 
71 ll querymx(ll rt,ll l,ll r,ll ql,ll qr){
72    if(ql<=l&&r<=qr) return mx[rt];
73    ll mid=(l+r)>>1;
74    if(qr<=mid) return querymx(rt<<1,l,mid,ql,qr);
75    if(mid<ql) return querymx(rt<<1|1,mid+1,r,ql,qr);
76    return max(querymx(rt<<1,l,mid,ql,qr),querymx(rt<<1|1,mid+1,r,ql,qr));
77 }
78 
79 ll querymi(ll rt,ll l,ll r,ll ql,ll qr){
80    if(ql<=l&&r<=qr) return mi[rt];
81    ll mid=(l+r)>>1;
82    if(qr<=mid) return querymi(rt<<1,l,mid,ql,qr);
83    if(mid<ql) return querymi(rt<<1|1,mid+1,r,ql,qr);
```

```
84      return min(querymi(rt<<1,l,mid,ql,qr),querymi(rt<<1|1,mid+1,r,ql,qr));
85  }
86
87  int main(){
88      ll n,m;
89      while(~scanf("%lld%lld",&n,&m)){
90          for(int i=1;i<=n;i++) head[i]=0;
91          cnt=0;
92          memset(dp,0,sizeof dp);
93
94          for(ll i=2;i<=n;i++){
95              ll fa=input(),w=input();
96              Ins(i,fa,w);
97          }
98
99          dfs(1,0);
100         dfs2(1,0);
101
102         for(ll i=1;i<=n;i++){
103             ans[i]=max(dp[i][0],dp[i][2]);
104         }
105
106         build(1,1,n);
107
108         Ans=0;
109         for(ll r=1,l=1;r<=n;r++){
110             while(l<r&&querymx(1,1,n,l,r)-querymi(1,1,n,l,r)>m) l++;
111             Ans=max(Ans,r-l+1);
112         }
113
114         printf("%lld\n",Ans);
115     }
116 }
```

5.3.9 例题 9(维护树链)

```
1   //同T8
2   #include <bits/stdc++.h>
3
4   using namespace std;
5
6   #define ll long long
7   ll input(){
8       ll x=0,f=0;char ch=getchar();
9       while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
10      while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
11      return f? -x:x;
12  }
13
14  const int N=1e5+7;
15
16  struct edge{
```

```
17      int v,w,next;
18  }e[N<<1];
19  int head[N],cnt=0;
20
21  void Ins(int u,int v,int w){
22      e[++cnt]=(edge){v,w,head[u]},head[u]=cnt;
23      e[++cnt]=(edge){u,w,head[v]},head[v]=cnt;
24  }
25
26  int val[N];
27  int dp_down[2][N],dp_up[2][N];
28
29  void dfs1(int u,int fa){
30      dp_down[0][u]=dp_down[1][u]=val[u];
31      for(int i=head[u];i;i=e[i].next){
32          ll v=e[i].v,w=e[i].w;
33          if(v==fa) continue;
34          dfs1(v,u);
35          if(dp_down[0][v]-w*2>=0){
36              dp_down[0][u]+=dp_down[0][v]-w*2;
37          }
38      }
39
40      ll mx=0;
41      for(int i=head[u];i;i=e[i].next){
42          ll v=e[i].v,w=e[i].w;
43          if(v==fa) continue;
44          if(dp_down[0][v]-2*w>0){
45              mx=max(mx,(dp_down[1][v]-w)-(dp_down[0][v]-2*w));
46          }else{
47              mx=max(mx,dp_down[1][v]-w);
48          }
49      }
50      dp_down[1][u]=dp_down[0][u]+mx;
51  }
52
53  void dfs2(int u,int fa){
54      int mx1=0,mx2=0,tmp;
55      for(int i=head[u];i;i=e[i].next){
56          int v=e[i].v,w=e[i].w;
57          if(v==fa) continue;
58          if(dp_down[0][v]-2*w>0) tmp=(dp_down[1][v]-w)-(dp_down[0][v]-2*w);
59          else tmp=dp_down[1][v]-w;
60          if(mx1<tmp){
61              mx2=mx1;
62              mx1=tmp;
63          }else if(mx2<tmp){
64              mx2=tmp;
65          }
66      }
67
68      for(int i=head[u];i;i=e[i].next){
69          int v=e[i].v,w=e[i].w;
```

```
70          if(v==fa) continue;
71
72          int tmp2;
73          if(dp_down[0][v]-2*w>0){
74              tmp2=dp_down[0][u]-(dp_down[0][v]-2*w);
75          }else{
76              tmp2=dp_down[0][u];
77          }
78
79          int mx=max(dp_up[0][u]-2*w,tmp2-2*w);
80          mx=max(mx,dp_up[0][u]+tmp2-2*w-val[u]);
81          dp_up[0][v]=val[v]+max(0,mx);
82
83          if(dp_down[0][v]-2*w>0){
84              if(mx1==(dp_down[1][v]-w)-(dp_down[0][v]-2*w)) tmp=dp_down[1][u]-(dp_down
                    [1][v]-w)+mx2;
85              else tmp=dp_down[1][u]-(dp_down[0][v]-2*w);
86          }else if(dp_down[1][v]-w>0){
87              if(mx1==dp_down[1][v]-w) tmp=dp_down[1][u]-(dp_down[1][v]-w)+mx2;
88              else tmp=dp_down[1][u];
89          }else tmp=dp_down[1][u];
90          mx=max(dp_up[1][u]-w,tmp-w);
91          mx=max(mx,max(dp_up[0][u]+tmp-w-val[u],dp_up[1][u]+tmp2-w-val[u]));
92          dp_up[1][v]=val[v]+max(0,mx);
93
94          dfs2(v,u);
95      }
96  }
97
98  int main(){
99      int T=input(),cas=0;
100     while(T--){
101         memset(head,0,sizeof head);
102         cnt=0;
103
104         int n=input();
105         for(int i=1;i<=n;i++) val[i]=input();
106
107         for(int i=1;i<n;i++){
108             int u=input(),v=input(),w=input();
109             Ins(u,v,w);
110         }
111
112         dfs1(1,1);
113
114         dp_up[0][1]=dp_up[1][1]=val[1];
115
116         dfs2(1,1);
117
118         printf("Case #%d:\n",++cas);
119
120         for(int i=1;i<=n;i++){
121             printf("%d\n",max(dp_up[0][i]+dp_down[1][i],dp_up[1][i]+dp_down[0][i])-val
```

```
            [i]);
122     }
123   }
124 }
```

6 杂项

6.1 离散化

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  #define PII pair <int, int>
6  #define fr first
7  #define sc second
8  #define mp make_pair
9  #define N 100050
10
11 // Input
12 int val[N], n;
13
14 // Output
15 int Cnt, sg[N];
16 PII li[N];
17
18 void Solve() {
19   for (int i=1;i<=n;i++)
20     li[i]=mp(val[i],i);
21   sort(li+1,li+n+1);
22
23   li[0].fr=-1;
24   for (int i=1;i<=n;i++)
25     sg[Cnt+=li[i].fr!=li[i-1].fr]=li[i].fr,
26     val[li[i].sc]=Cnt;
27   return;
28 }
```

6.2 归并排序

```
1  #include<stdio.h>
2  int a[1000005],temp[1000005];
3  long long count;
4  void Merge_sort(int L,int mid,int R)
5  {
6    int i=L,j=mid+1,k=L;
7    while(i<=mid&&j<=R)
8    {
9      if(a[i]<=a[j])
10     temp[k++]=a[i++];
11     else
```

```
12          {
13              temp[k++]=a[j++];
14              count+=(mid-i+1);
15          }
16      }
17      while(i<=mid)
18      temp[k++]=a[i++];
19      while(j<=R)
20      temp[k++]=a[j++];
21      for(i=L;i<=R;i++)
22      a[i]=temp[i];
23  }
24  void Merge(int L,int R)
25  {
26      if(L<R)
27      {
28          int mid=(L+R)/2;
29          Merge(L,mid);
30          Merge(mid+1,R);
31          Merge_sort(L,mid,R);
32      }
33  }
34  int main()
35  {
36      int N;
37      while(~scanf("%d",&N)&&N)
38      {
39          int i;
40          for(i=0;i<N;i++)
41          scanf("%d",&a[i]);
42          count=0;
43          Merge(0,N-1);
44          printf("%lld\n",count);
45      }
46  }
```

6.3 struct

```
1   #include <set>
2   #include <cstdio>
3   #include <cstdlib>
4   #include <iostream>
5   #include <algorithm>
6   using namespace std;
7
8   struct node {
9       int l, r, mid;
10      node *lef, *rig;
11      node (int l, int r) {
12          this->l = l;
13          this->r = r;
14          mid = (l + r) >> 1;
```

```
15
16          if (l == r)
17              return;
18          lef = new node(l, mid);
19          rig = new node(mid + 1, r);
20      }
21  };
22  struct node2 {
23      int l, r, mid;
24      node2 *lef, *rig;
25      void build(int l, int r) {
26          this->l = l;
27          this->r = r;
28          mid = (l + r) >> 1;
29
30          if (l == r)
31              return;
32          (lef = new node2)->build(l, mid);
33          (rig = new node2)->build(mid + 1, r);
34      }
35      int query(int x) {
36          if (l == r)
37              return l;
38          if (x <= mid)
39              return lef->query(x);
40          else
41              return rig->query(x);
42      }
43  };
44  struct matrix {
45      int v[4][4];
46      int *operator [] (int x) { return v[x]; }
47  };
48  matrix operator * (matrix &a, matrix &b) {
49      matrix ret;
50      for (int i = 0; i < 4; ++i)
51          for (int j = 0; j < 4; ++j)
52              for (int k = 0; k < 4; ++k)
53                  ret[i][j] += a[i][k] * b[k][j];
54      return ret;
55  }
56  struct element {
57      int x, y;
58  };
59  struct bignum {
60      int n, a[10];
61      void operator += (bignum &t) {
62          n = max(n, t.n);
63          for (int i = 1; i <= n; ++i) {
64              a[i] += t.a[i];
65              if (a[i] >= 1000) {
66                  a[i] -= 1000;
67                  a[i + 1] ++;
```

```
68          }
69       }
70       if (a[n + 1] > 0)
71          ++n;
72    }
73 };
74
75 bool cmp_element(const element &a, const element &b) {
76    return a.x < b.x || (a.x == b.x && a.y < b.y);
77 }
78
79 struct cmp {
80    bool operator () (element *a, element *b) {
81       if (a->x != b->x)
82          return a->x < b->x;
83       if (a->y != b->y)
84          return a->y < b->y;
85       return a < b;
86    }
87 };
88
89 set <element*, cmp> heap;
90
91 int main() {
92    node *root = new node(1, 100);
93
94    node2 *root2 = new node2;
95    root2->build(1, 100);
96
97    cout << root2->query(5) << endl;
98
99    matrix a, b, c;
100   a = b * c;
101
102   element s[100];
103   sort(s + 1, s + 100, cmp_element);
104
105   bignum x, y;
106   x += y;
107 }
```

6.4 0/1 分数规划

1. 概述对于形如

$$\frac{\sum a_i}{\sum b_i} = c$$

的式子，要求最大/最小化 c。

2. 解法可以考虑贪心，但是发现所求的是分式结构，不好操作。

我们采用更优的一种方式：01 分数规划。

我们可以把每个 i 看作一个物品，a_i,b_i 分别为价值和体积。

因为最大化的是比值，没有办法直接背包。

我们可以在每个物品前加一个系数 0/1，表示该物品选或不选。

选的话上下同加上当前的 a 和 b。

这只是理论部分。

3. 实现先经过移项：

$$\sum a_i = \sum b_i \times c \sum a_i - b_i \times c = 0$$

有一个显然的性质：c 是单调的。

也就是说，我们可以二分查找 c 的取值。

对于每一个 c，我们可以进行验证，利用 $a_i - b_i \times c$。至于具体的 check 方式，有 SPFA 找负环，贪心，dp 等，因题而定。一般 c 都是小数，所以记住用 double。

例题:

某地方有 N 个工厂，有 N-1 条路连接它们，且它们两两都可达。每个工厂都有一个产量值和一个污染值。现在工厂要进行规划，拆除其中的 M 个工厂，使得剩下的工厂依然连成一片且总产量/总污染的值最大。

Solution: 很经典的分数规划。

把产量看成 a，污染值看成 b，就是上面的式子。

考虑验证，即在联通条件下，验证 $a_i + b_i \times c$ 与 0 的关系。

如果大于 0，说明可以继续扩大；反之，则不能。

那么现在要求一个 $n - m$ 的连通块，最大化 $a_i + b_i \times c$。

又因为图是棵树，所以采用树上背包。

```
#include <bits/stdc++.h>

using namespace std;

#define ll long long
int input(){
   int x=0,f=0;char ch=getchar();
   while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
   while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
   return f? -x:x;
}

#define N 107
const double eps=1e-4;
const int INF=1<<30;

double a[N],b[N],d[N];
struct edge{
   int v,next;
}e[2*N];
int head[N],tot;
double dp[N][N];
```

```
23 int siz[N];
24 int n,m;
25
26
27 void Ins(int u,int v){
28    e[++tot]=(edge){v,head[u]},head[u]=tot;
29    e[++tot]=(edge){u,head[v]},head[v]=tot;
30 }
31
32 void dfs(int x,int fa){
33    dp[x][0]=0;
34    siz[x]=1;
35    for(int i=head[x];i;i=e[i].next){
36       if(e[i].v==fa) continue;
37       dfs(e[i].v,x);
38       siz[x]+=siz[e[i].v];
39       for(int j=min(siz[x],m);j>=0;j--){
40          for(int k=0;k<=min(j,siz[x]);k++){
41             dp[x][j]=max(dp[x][j-k]+dp[e[i].v][k],dp[x][j]);
42          }
43       }
44    }
45    for(int i=min(m,siz[x]);i>0;i--){
46       dp[x][i]=dp[x][i-1]+d[x];
47    }
48 }
49
50 bool check(){
51    for(int i=1;i<=n;i++){
52       for(int j=1;j<=m;j++){
53          dp[i][j]=-INF;
54       }
55    }
56
57    dfs(1,0);
58
59    double res=-INF;
60    for(int i=1;i<=n;i++){
61       res=max(dp[i][m],res);
62    }
63
64    return res>0;
65 }
66
67 int main(){
68    n=input(),m=n-input();
69    for(int i=1;i<=n;i++) a[i]=input();
70    for(int i=1;i<=n;i++) b[i]=input();
71
72    int u,v;
73    for(int i=1;i<n;i++){
74       u=input(),v=input();
75       Ins(u,v);
```

```
76      }
77
78      double l=0.0,r=10000.0,mid;
79      while(r-l>eps){
80          mid=(l+r)/2;
81          for(int i=1;i<=n;i++) d[i]=a[i]-mid*b[i];
82          check()? l=mid:r=mid;
83      }
84      printf("%.11f\n",mid);
85  }
```

6.5 莫队 (小 z 的袜子)

作为一个生活散漫的人，小 Z 每天早上都要耗费很久从一堆五颜六色的袜子中找出一双来穿。终于有一天，小 Z 再也无法忍受这恼人的找袜子过程，于是他决定听天由命……

具体来说，小 Z 把这 N 只袜子从 1 到 N 编号，然后从编号 L 到 R (L 尽管小 Z 并不在意两只袜子是不是完整的一双，甚至不在意两只袜子是否一左一右，他却很在意袜子的颜色，毕竟穿两只不同色的袜子会很尴尬。

你的任务便是告诉小 Z，他有多大的概率抽到两只颜色相同的袜子。当然，小 Z 希望这个概率尽量高，所以他可能会询问多个 (L,R) 以方便自己选择。

然而数据中有 L=R 的情况，请特判这种情况，输出 0/1。

```
1   #include <bits/stdc++.h>
2
3   using namespace std;
4
5   #define ll long long
6   ll input(){
7       ll x=0,f=0;char ch=getchar();
8       while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
9       while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
10      return f? -x:x;
11  }
12
13  #define debug printf("PASS IN LINE:%d\n",__LINE__)
14
15  #define N (int)(50007)
16  int n,m,k;
17  int a[N];
18  struct Query{
19      int l,r,id;
20  }qu[N];
21
22  bool operator <(Query &a,Query &b) { return a.l/k==b.l/k? a.r<b.r:a.l/k<b.l/k; }
23
24  ll Ans[N],Ans_[N];
25
26  ll l=0,r=0,now=0;
27
28  int cnt[N];
```

```
29
30 ll GCD(ll a, ll b){ return b?GCD(b, a%b):a; }
31
32 void move(int pos,int sign){
33    now=now+2*sign*cnt[a[pos]]+1;
34    cnt[a[pos]]+=sign;
35 }
36
37 void Solve(){
38    k=ceil(pow(n,0.5));
39    sort(qu+1,qu+m+1);
40    l=1,r=0,now=0;
41    for(int i=1;i<=m;i++){
42       while(l > qu[i].l) move(--l,1);
43       while(r < qu[i].r) move(++r,1);
44       while(l < qu[i].l) move(l++,-1);
45       while(r > qu[i].r) move(r--,-1);
46       if(l==r) Ans[qu[i].id]=0,Ans_[qu[i].id]=1;
47       else {
48          Ans[qu[i].id]=1ll*now-(r-l+1),Ans_[qu[i].id]=1ll*(r-l+1)*(r-l);
49          ll d=GCD(Ans[qu[i].id],Ans_[qu[i].id]);
50          Ans[qu[i].id]/=d,Ans_[qu[i].id]/=d;
51       }
52    }
53 }
54
55 int main(){
56    n=input(),m=input();
57    for(int i=1;i<=n;i++){
58       a[i]=input();
59    }
60
61    for(int i=1;i<=m;i++){
62       qu[i].l=input(),qu[i].r=input(),qu[i].id=i;
63    }
64
65    Solve();
66
67    for(int i=1;i<=m;i++){
68       printf("%lld/%lld\n",Ans[i],Ans_[i]);
69    }
70 }
```

6.6 尺取法

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 #define ll long long
6 ll input(){
7    ll x=0,f=0;char ch=getchar();
```

```
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

const int N=1e5+7;

int a[1007],n;
bitset <N> bt;

int fd(int L,int R){
    if(L>=R) return 0;
    int l=L,r=L,res=0;
    while(1){
        while(bt[a[r]]==0&&r<R)
            bt[a[r++]]=1,res=max(res,r-l);
        if(l==R&&r==R) break;
        bt[a[l++]]=0;
        if(l==R&&r==R) break;
    }
    return res;
}

int main(){
    int T=input(),cas=0;
    while(T--){
        n=input();
        for(int i=0;i<n;i++) a[i]=input();

        int l=0,r=0,Ans=0;

        while(1){
            while(bt[a[r]]==0&&r<n)
                bt[a[r++]]=1,Ans=max(Ans,(r-l)+max(fd(0,l),fd(r,n)));
            if(l==n&&r==n) break;
            bt[a[l++]]=0;
            Ans=max(Ans,(r-l)+max(fd(1,l),fd(r,n)));
            if(l==n&&r==n) break;
        }
        printf("Case #%d: %d\n",++cas,Ans);
    }
}
```

6.7 求最大子矩阵

6.7.1 悬线法

```
// // 悬线法O(NM)用于障碍点比较多的情况
// // luogu p1169
// 给定一个01棋盘,求其中01交错的最大正方形与矩形。
#include <bits/stdc++.h>

```

```cpp
using namespace std;

#define ll long long
ll input(){
    ll x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

const int N=2007;

int a[N][N];
int H[N][N],L[N][N],R[N][N];

int main(){
    int n=input(),m=input();

    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            a[i][j]=input();
            H[i][j]=1,L[i][j]=R[i][j]=j;
        }

        for(int j=2;j<=m;j++)
            if(a[i][j]!=a[i][j-1]) L[i][j]=L[i][j-1];//原题要求黑白相间，所以要不等于
        for(int j=m-1;j>=1;j--)
            if(a[i][j]!=a[i][j+1]) R[i][j]=R[i][j+1];//原题要求黑白相间，所以要不等于

        for(int j=1;j<=m;j++){
            if(i>1&&a[i][j]!=a[i-1][j]){//原题要求黑白相间，所以要不等于
                H[i][j]=H[i-1][j]+1;
                L[i][j]=max(L[i-1][j],L[i][j]),R[i][j]=min(R[i-1][j],R[i][j]);
            }
        }
    }

    int Ans1=0,Ans2=0;

    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            int a=H[i][j],b=R[i][j]-L[i][j]+1;
            Ans1=max(Ans1,min(a,b)*min(a,b));
            Ans2=max(Ans2,a*b);
        }
    }

    printf("%d\n%d\n",Ans1,Ans2);
}
```

6.7.2 单调队列

```cpp
//O(NM)单调栈做法
//luogu P4147
//它要找一块矩形土地，要求这片土地都标着 'F' 并且面积最大

#include <bits/stdc++.h>

using namespace std;

#define ll long long
ll input(){
    ll x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

const int N=1007;
int a[N],h[N];
stack <int> stk;
int Ans=0;

int main(){
    int n=input(),m=input();

    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            char ch[2];
            scanf("%s",ch);
            if(ch[0]=='F') h[j]=h[j]+1;
            else h[j]=0;

            a[j]=h[j];

            int top=j;
            while(!stk.empty()&&a[stk.top()]>a[j]){
                top=stk.top();stk.pop();
                Ans=max(Ans,(j-top)*a[top]);
            }
            a[top]=a[j];
            stk.push(top);
        }
        while(!stk.empty()){
            Ans=max(Ans,(m+1-stk.top())*a[stk.top()]);
            stk.pop();
        }
    }
    printf("%d\n",Ans*3);
}
```

6.7.3 王知昆论文做法

```cpp
//luogu p1578
```

```cpp
#include <bits/stdc++.h>

using namespace std;

#define ll long long
ll input(){
   ll x=0,f=0;char ch=getchar();
   while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
   while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
   return f? -x:x;
}

#define PII pair <int,int>
#define fr first
#define sc second
#define mp make_pair

const int N=5007;

PII a[N];
int n,m,k,Ans;

bool cmp(PII a,PII b){return a.sc<b.sc;}

int main(){
   n=input(),m=input(),k=input();

   for(int i=1;i<=k;i++){
      a[i].fr=input(),a[i].sc=input();
   }

   a[++k]=mp(0,0),a[++k]=mp(0,m);
   a[++k]=mp(n,0),a[++k]=mp(n,m);

   sort(a+1,a+k+1);

   for(int i=1;i<=k;i++){
      int h1=m,h2=0,v=n-a[i].fr,f=0;

      for(int j=i+1;j<=k;j++){
         if(h2<=a[j].sc&&a[j].sc<=h1){
            if(v*(h1-h2)<=Ans) break;

            Ans=max(Ans,(h1-h2)*(a[j].fr-a[i].fr));

            if(a[j].sc==a[i].sc){
               f=1;
               break;
            }

            if(a[j].sc>a[i].sc) h1=min(h1,a[j].sc);
            else h2=max(h2,a[j].sc);
         }
```

```
55          }
56          if(!f) Ans=max(Ans,v*(h1-h2));
57
58          h1=m,h2=0,v=a[i].fr,f=0;
59          for(int j=i-1;j>=1;j--){
60              if(h2<=a[j].sc&&a[j].sc<=h1){
61                  if(v*(h1-h2)<=Ans) break;
62
63                  Ans=max(Ans,(h1-h2)*(a[i].fr-a[j].fr));
64
65                  if(a[j].sc==a[i].sc){
66                      f=1;
67                      break;
68                  }
69
70                  if(a[j].sc>a[i].sc) h1=min(h1,a[j].sc);
71                  else h2=max(h2,a[j].sc);
72              }
73          }
74          if(!f) Ans=max(Ans,v*(h1-h2));
75      }
76
77      sort(a+1,a+k+1,cmp);
78      for(int i=1;i<k;i++){
79          Ans=max(Ans,(a[i+1].sc-a[i].sc)*n);
80      }
81      printf("%d\n",Ans);
82  }
```

6.8 整体二分

给你 n 个数，多次询问某段区间第 k 小的数。($1 \leq n \leq 100000, 1 \leq m \leq 5000$)

所谓整体二分，需要数据结构题满足以下性质:

1. 询问的答案具有可二分性。

2. 修改对判定答案的贡献相对独立，修改之间互不影响效果。

3. 修改如果对判定答案有贡献，则贡献为一确定的与判定标准无关的值。

4. 贡献满足交换律，结合律，具有可加性。

5. 题目允许离线操作。

询问的答案有可二分性质显然是前提，我们发现，因为修改对判定标准的贡献相对独立，且贡献的值 (如果有的话) 与判定标准无关，所以如果我们已经计算过某一个些修改对询问的贡献，那么这个贡献永远不会改变，我们没有必要当判定标准改变时再次计算这部分修改的贡献，只要记录下当前的总贡献，再进一步二分时，直接加上新的贡献即可。

这样的话，我们发现，处理的复杂度可以不再与序列总长度直接相关了，而可能只与当前待处理序列的长度相关。

接下来考虑本题。

对于单个查询而言，我们可以采用预处理 + 二分答案的方法解决，但现在我们要回答的是一系列的查询，对于查询而言我们都要重新预处理然后二分，时间复杂度无法承受，但是我们

仍然希望通过二分答案的思想来解决，整体二分就是基于这样一种想法，我们将所有操作 (包括修改和查询) 一起二分，进行分治。

我们时刻维护一个操作序列和对应的可能答案区间 [L,R]，我们先求的一个判定答案 mid=(L+R)»1，然后我们考虑操作序列的修改操作，将其中符合条件的修改对各个询问的贡献统计出来，然后我们对操作序列进行划分。

```
//k-th number
#include <bits/stdc++.h>

using namespace std;

#define ll long long
ll input(){
    ll x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') f|=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return f? -x:x;
}

#define debug printf("pass in line:%d in [%s]\n",__LINE__,__FUNCTION__)
#define lowbit(x) x&-x

const int N = 2e5+7;
const int inf=0x3f3f3f3f;

struct opr{
    int f,l,r,k,pos,val;
    //修改中 val:数值, f:操作类型(1), pos:在数组中的位置
    //查询中 l,r,k:输入, f:操作类型(2), pos:询问编号
};
opr q[N];
opr ql[N],qr[N];
int Ans[N];
int t[N];

int n,m;


//第一类操作是修改，也就是输入的 nn 个数。
// 这里用树状数组维护，最普通的吧，维护比一个数小的数的个数
void update(int pos,int x){
    for(;pos<=n;pos+=lowbit(pos)) t[pos]+=x;
}

int query(int l,int r){
    int res=0;l--;
    for(;l>0;l-=lowbit(l)) res-=t[l];
    for(;r>0;r-=lowbit(r)) res+=t[r];
    return res;
}

void solve(int l,int r,int head,int tail){
```

```
47      //head,tail:操作区间; l,r:答案区间
48      if(head>tail) return;
49      // //找到答案了
50      if(l==r){//[ql,qr] 表示操作区间，[L,R]表示答案区间。如果 L=R 就说明找到答案了，将 [ql
            ,qr] 中所有查询操作的答案赋为 L。
51          for(int i=head;i<=tail;i++)
52              if(q[i].f==2) Ans[q[i].pos]=l;
53          return;
54      }
55      int mid=(l+r)>>1;
56      int lcnt=0,rcnt=0;
57      for(int i=head;i<=tail;i++){
58          //但是并不是直接全都 add 进去。在二分的时候，如果 q[i].x<=mid，也就是会影响第 k 小
                的值，就 add(q[i].id,1)，并将这个操作存进 q1，因为在左区间也会用到，否则直接存
                进 q2，因为在右区间可能会用到。
59          if(q[i].f==1){
60              if(q[i].val<=mid){
61                  ql[++lcnt]=q[i];
62                  update(q[i].pos,1);
63              }else qr[++rcnt]=q[i];
64          }else{
65              int num=query(q[i].l,q[i].r);
66              if(num>=q[i].k) ql[++lcnt]=q[i];
67              else q[i].k-=num,qr[++rcnt]=q[i];
68          }
69
70          /*
71              第二类操作是查询。
72
73              如果当前累计贡献 cnt 比要求贡献大，也就是数的个数大于 k，说明 mid 过大，
74              满足标准的修改过多，我们需要给这个查询设置更小的答案区间，
75              于是二分到答案区间 [L,mid]，否则二分到区间 [mid+1,R]，
76              并将查询第 k 小改为查询第 k-cnt 小。
77          */
78      }
79      for(int i=1;i<=lcnt;i++){
80          if(ql[i].f==1)
81              update(ql[i].pos,-1);
82      }
83
84      // 最后把 q1 和 q2 再存回 q，进行二分就可以了。注意要把树状数组清空。
85      tail=head-1;
86      for(int i=1;i<=lcnt;i++) q[++tail]=ql[i];
87      int div=tail;
88      for(int i=1;i<=rcnt;i++) q[++tail]=qr[i];
89
90      solve(l,mid,head,div);
91      solve(mid+1,r,div+1,tail);
92  }
93
94  int main(){
95      while(~scanf("%d%d",&n,&m)){
96          memset(t,0,sizeof(t));
```

```
97        int cnt=0;
98        for(int i=1;i<=n;i++){
99            q[++cnt].val=input();
100           q[cnt].pos=i,q[cnt].f=1;
101       }
102
103       for(int i=1;i<=m;i++){
104           q[++cnt].l=input(),q[cnt].r=input(),q[cnt].k=input();
105           q[cnt].f=2,q[cnt].pos=i;
106       }
107
108       solve(-inf,inf,1,cnt);
109
110       for(int i=1;i<=m;i++){
111           printf("%d\n",Ans[i]);
112       }
113   }
114 }
```

7 附录

7.1 STL

7.1.1 vector

```
1  #include<vector>;
2
3  vector<int> v;
4  vector<int> a(10); //定义了10个整型元素的向量（尖括号中为元素类型名，它可以是任何合法的数
       据类型），但没有给出初值，其值是不确定的。
5  vector<int> a(10,1); //定义了10个整型元素的向量,且给出每个元素的初值为1
6  vector<int> a(b); //用b向量来创建a向量，整体复制性赋值
7  vector<int> a(b.begin(),b.begin+3); //定义了a值为b中第0个到第2个（共3个）元素
8  int b[7]={1,2,3,4,5,9,8};
9  vector<int> a(b,b+7); //从数组中获得初值
10
11
12 a.assign(b.begin(), b.begin()+3); //b为向量，将b的0~2个元素构成的向量赋给a
13 a.assign(4,2); //是a只含4个元素，且每个元素为2
14 a.back(); //返回a的最后一个元素
15 a.front(); //返回a的第一个元素
16 a[i]; //返回a的第i个元素，当且仅当a[i]存在2013-12-07
17 a.clear(); //清空a中的元素
18 a.empty(); //判断a是否为空，空则返回ture,不空则返回false
19 a.pop_back(); //删除a向量的最后一个元素
20 a.erase(a.begin()+1,a.begin()+3); //删除a中第1个（从第0个算起）到第2个元素，也就是说删除
       的元素从a.begin()+1算起（包括它）一直到a.begin()+3（不包括它）
21 a.push_back(5); //在a的最后一个向量后插入一个元素，其值为5
22 a.insert(a.begin()+1,5); //在a的第1个元素（从第0个算起）的位置插入数值5，如a为1,2,3,4，
       插入元素后为1,5,2,3,4
23 a.insert(a.begin()+1,3,5); //在a的第1个元素（从第0个算起）的位置插入3个数，其值都为5
```

```
a.insert(a.begin()+1,b+3,b+6); //b为数组，在a的第1个元素（从第0个算起）的位置插入b的第3个元素到第5个元素（不包括b+6），如b为1,2,3,4,5,9,8，插入元素后为1,4,5,9,2,3,4,5,9,8
a.size(); //返回a中元素的个数；
a.capacity(); //返回a在内存中总共可以容纳的元素个数
a.resize(10); //将a的现有元素个数调至10个，多则删，少则补，其值随机
a.resize(10,2); //将a的现有元素个数调至10个，多则删，少则补，其值为2
a.reserve(100); //将a的容量（capacity）扩充至100，也就是说现在测试a.capacity();的时候返回值是100.这种操作只有在需要给a添加大量数据的时候才显得有意义，因为这将避免内存多次容量扩充操作（当a的容量不足时电脑会自动扩容，当然这必然降低性能）
a.swap(b); //b为向量，将a中的元素和b中的元素进行整体性交换
a==b; //b为向量，向量的比较操作还有!=,>=,<=,>,<

sort(a.begin(),a.end()); //对a中的从a.begin()（包括它）到a.end()（不包括它）的元素进行从小到大排列
reverse(a.begin(),a.end()); //对a中的从a.begin()（包括它）到a.end()（不包括它）的元素倒置，但不排列，如a中元素为1,3,2,4,倒置后为4,2,3,1
copy(a.begin(),a.end(),b.begin()+1); //把a中的从a.begin()（包括它）到a.end()（不包括它）的元素复制到b中，从b.begin()+1的位置（包括它）开始复制，覆盖掉原有元素
find(a.begin(),a.end(),10); //在a中的从a.begin()（包括它）到a.end()（不包括它）的元素中查找10，若存在返回其在向量中的位置
```

7.1.2 deque

```
deque<int> a; // 定义一个int类型的双端队列a
deque<int> a(10); // 定义一个int类型的双端队列a，并设置初始大小为10
deque<int> a(10, 1); // 定义一个int类型的双端队列a，并设置初始大小为10且初始值都为1
deque<int> b(a); // 定义并用双端队列a初始化双端队列b
deque<int> b(a.begin(), a.begin()+3); // 将双端队列a中从第0个到第2个(共3个)作为双端队列b的初始值

int n[] = { 1, 2, 3, 4, 5 };
// 将数组n的前5个元素作为双端队列a的初值
// 说明：当然不包括arr[4]元素，末尾指针都是指结束元素的下一个元素，
// 这个主要是为了和deque.end()指针统一。
deque<int> a(n, n + 5);
deque<int> a(&n[1], &n[4]); // 将n[1]、n[2]、n[3]作为双端队列a的初值

deq.size();//容器大小
deq.max_size();//容器最大容量
deq.resize(x);//更改容器大小为x
deq.empty();//容器判空
deq.shrink_to_fit();//减少容器大小到满足元素所占存储空间的大小

// 遍历显示
for (it = deq.begin(); it != deq.end(); it++)
   cout << *it << " ";
cout << endl;

// 头部增加元素
deq.push_front(4);
// 末尾添加元素
deq.push_back(5);
```

```
// 任意位置插入一个元素
deque<int>::iterator it = deq.begin();
deq.insert(it, 2);
// 任意位置插入n个相同元素
it = deq.begin(); // 必须要有这句
deq.insert(it, 3, 9);
// 插入另一个向量的[forst,last]间的数据
deque<int> deq2(5,8);
it = deq.begin(); // 必须要有这句
deq.insert(it, deq2.end() - 1, deq2.end());

// 头部删除元素
deq.pop_front();
// 末尾删除元素
deq.pop_back();
// 任意位置删除一个元素
deque<int>::iterator it = deq.begin();
deq.erase(it);
// 删除[first,last]之间的元素
deq.erase(deq.begin(), deq.begin()+1);
// 清空所有元素
deq.clear();

// 下标访问
cout << deq[0] << endl;
// at方法访问
cout << deq.at(0) << endl;
// 访问第一个元素
cout << deq.front() << endl;
// 访问最后一个元素
cout << deq.back() << endl;

// 多个元素赋值
deque<int> deq;
deq.assign(3, 1);
deque<int> deq2;
deq2.assign(3, 2);

// 交换两个容器的元素
deq.swap(deq2);

deq.begin();//开始迭代器指针
deq.end(); //末尾迭代器指针,指向最后一个元素的下一个位置
deq.cbegin();//指向常量的开始迭代器指针,意思就是不能通过这个指针来修改所指的内容，但还是可
    以通过其他方式修改的，而且指针也是可以移动的。
deq.cend();//指向常量的末尾迭代器指针
deq.rbegin();//反向迭代器指针，指向最后一个元素
deq.rend();//反向迭代器指针，指向第一个元素的前一个元素
```

7.1.3 list

```
//List是stl实现的双向链表，与向量(vectors)相比，它允许快速的插入和删除，但是随机访问却比较慢.
list<int>lst1; //创建空list
list<int> lst2(5); //创建含有5个元素的list
list<int>lst3(3,2); //创建含有3个元素的list
list<int>lst4(lst2); //使用lst2初始化lst4
list<int>lst5(lst2.begin(),lst2.end()); //同lst4

Lst1.assign() //给list赋值
Lst1.back() //返回最后一个元素
Lst1.begin() //返回指向第一个元素的迭代器
Lst1.clear() //删除所有元素
Lst1.empty() //如果list是空的则返回true
Lst1.end() //返回末尾的迭代器
Lst1.erase() //删除一个元素
Lst1.front() //返回第一个元素
Lst1.get_allocator() //返回list的配置器
Lst1.insert() //插入一个元素到list中
Lst1.max_size() //返回list能容纳的最大元素数量
Lst1.merge() //合并两个list
Lst1.pop_back() //删除最后一个元素
Lst1.pop_front() //删除第一个元素
Lst1.push_back() //在list的末尾添加一个元素
Lst1.push_front() //在list的头部添加一个元素
Lst1.rbegin() //返回指向第一个元素的逆向迭代器
Lst1.remove() //从list删除元素
Lst1.remove_if() //按指定条件删除元素
Lst1.rend() //指向list末尾的逆向迭代器
Lst1.resize() //改变list的大小
Lst1.reverse() //把list的元素倒转
Lst1.size() //返回list中的元素个数
Lst1.sort() //给list排序
Lst1.splice() //合并两个list
Lst1.swap() //交换两个list
Lst1.unique() //删除list中重复的元素
```

7.1.4 (multi)set

```
begin()//返回指向第一个元素的迭代器
clear()//清除所有元素
count()//返回某个值元素的个数
empty()//如果集合为空，返回true
end()//返回指向最后一个元素的迭代器
equal_range()//返回集合中与给定值相等的上下限的两个迭代器
erase()//删除集合中的元素
find()//返回一个指向被查找到元素的迭代器
get_allocator()//返回集合的分配器
insert()//在集合中插入元素
lower_bound()//返回指向大于（或等于）某值的第一个元素的迭代器
key_comp()//返回一个用于元素间值比较的函数
max_size()//返回集合能容纳的元素的最大限值
rbegin()//返回指向集合中最后一个元素的反向迭代器
```

```
rend()//返回指向集合中第一个元素的反向迭代器
size()//集合中元素的数目
swap()//交换两个集合变量
upper_bound()//返回大于某个值元素的迭代器
value_comp()//返回一个用于比较元素间的值的函数
```

7.1.5 map

```
//map的基本构造函数
map<string , int >strMap;
map<int ,string >intMap;
map<sring, char>strMap;
map< char ,string>charMap;
map<char ,int>charMap;
map<int ,char >intMap;

//map添加数据
map<int ,string> maplive;

pair<int,string> value(1,"a"); maplive.insert(value);
//等价于maplive.insert(pair<int,string>(1,"a"));

maplive.insert(map<int,string>::value_type(1,"a"));

maplive[1]="a";//map中最简单最常用的插入添加!

map的基本操作函数:

begin() //返回指向map头部的迭代器
clear() //删除所有元素
count() //返回指定元素出现的次数
empty() //如果map为空则返回true
end() //返回指向map末尾的迭代器
equal_range() //返回特殊条目的迭代器对
erase() //删除一个元素
find() //查找一个元素
get_allocator()//返回map的配置器
insert() //插入元素
key_comp() //返回比较元素key的函数
lower_bound() //返回键值>=给定元素的第一个位置
max_size() //返回可以容纳的最大元素个数
rbegin() //返回一个指向map尾部的逆向迭代器
rend() //返回一个指向map头部的逆向迭代器
size() //返回map中元素的个数
swap() // 交换两个map
upper_bound() // 返回键值>给定元素的第一个位置
value_comp() // 返回比较元素value的函数


//遍历
map<string,int>::iterator strmap_iter2 = strMap.begin();
for(;strmap_iter2 !=strMap.end();strmap_iter2++)
```

```
{
    cout<<strmap_iter2->first<<' '<<strmap_iter2->second<<endl;
}
```

7.1.6 unordered_set

```
unordered_set<string> c//初始化容器
unordered_set<string> c//{ "aaa", "bbb", "ccc" }：初始化容器，并将"aaa", "bbb", "ccc"加入到容器中
unordered_set<string> c//{ 16 }：初始化容器，并设置16个桶

c.insert("dddd")//向容器添加元素”dddd"
a.insert({ "aaa","bbbb","cccc" })//向容器添加元素"aaa","bbbb","cccc"
a.insert(b.begin(), b.end())//b是一个存储着和a相同类型元素的向量，可将b中所有元素添加到a中

a.find("eeee")//查找元素"eeee"，返回结果为a.end()则表明没有找到，否则返回所对应元素
a.count("eeee")//查找元素"eeee"在a中有几个（由于unordered_set中没有相同的元素，所以结果通常为0或1）

a.bucket_count()//返回数据结构中桶的数量
a.bucket_size(i)//返回桶i中的大小
a.bucket("eeee")//返回元素"eeee"在哪个桶里

a.hash_function()("aaa")//返回"aaa"所对应的hash值
a.key_eq()("aaa","aaaa")//当元素相同时返回true，否则返回false

a.clear()//清除a中所有元素
a.erase("aaa")//清除元素"aaa"

a.size()//返回a中总的元素个数
a.max_size()//返回a中最大容纳元素
a.empty()//判断a中是否为空
```

7.1.7 unordered_map

```
若有unordered_map <int, int> mp;查找x是否在map中
方法1: 若存在 mp.find(x)!=mp.end()
方法2: 若存在 mp.count(x)!=0

mp.insert(Map::value_type(1,"Raoul"));

unordered_map<key,T>::iterator it;
(*it).first; //the key value
(*it).second //the mapped value
for(unordered_map<key,T>::iterator iter=mp.begin();iter!=mp.end();iter++)
     cout<<"key value is"<<iter->first<<" the mapped value is "<< iter->second;


//也可以这样
for(auto& v : mp)
```

```
    print v.first and v.second

//其它操作同map
```

7.1.8 stack

```
stack <int> s;

s.size() //返回栈中元素个数
s.top() //返回栈顶的元素
s.pop() //从栈中取出并删除元素
s.push(e) //向栈中添加元素e
s.empty() //栈为空时返回true
```

7.1.9 queue

```
queue <int> q;
q.back() //返回最后一个元素
q.empty() //如果队列空则返回真
q.front() //返回第一个元素
q.pop() //删除第一个元素
q.push() //在末尾加入一个元素
q.size() //返回队列中元素的个数
```

7.1.10 priority_queue

```
top() //访问队头元素
empty() //队列是否为空
size() //返回队列内元素个数
push() //插入元素到队尾 (并排序)
emplace() //原地构造一个元素并插入队列
pop() //弹出队头元素
swap() //交换内容

//升序队列
priority_queue <int,vector<int>,greater<int> > q;
//降序队列
priority_queue <int,vector<int>,less<int> >q;

//greater和less是std实现的两个仿函数（就是使一个类的使用看上去像一个函数。其实现就是类中实现一个operator()，这个类就有了类似函数的行为，就是一个仿函数类了）


// 基本类型例子
#include<iostream>
#include <queue>
using namespace std;
int main()
{
   //对于基础类型 默认是大顶堆
```

```cpp
    priority_queue<int> a;
    //等同于 priority_queue<int, vector<int>, less<int> > a;

    // 这里一定要有空格，不然成了右移运算符↓
    priority_queue<int, vector<int>, greater<int> > c; //这样就是小顶堆
    priority_queue<string> b;

    for (int i = 0; i < 5; i++)
    {
        a.push(i);
        c.push(i);
    }
    while (!a.empty())
    {
        cout << a.top() << ' ';
        a.pop();
    }
    cout << endl;

    while (!c.empty())
    {
        cout << c.top() << ' ';
        c.pop();
    }
    cout << endl;

    b.push("abc");
    b.push("abcd");
    b.push("cbd");
    while (!b.empty())
    {
        cout << b.top() << ' ';
        b.pop();
    }
    cout << endl;
    return 0;
}

//pari的比较，先比较第一个元素，第一个相等比较第二个
#include <iostream>
#include <queue>
#include <vector>
using namespace std;
int main()
{
    priority_queue<pair<int, int> > a;
    pair<int, int> b(1, 2);
    pair<int, int> c(1, 3);
    pair<int, int> d(2, 5);
    a.push(d);
    a.push(c);
    a.push(b);
    while (!a.empty())
```

```
77      {
78          cout << a.top().first << ' ' << a.top().second << '\n';
79          a.pop();
80      }
81  }
82
83  // 对于自定义类型
84  #include <iostream>
85  #include <queue>
86  using namespace std;
87
88  //方法1
89  struct tmp1 //运算符重载<
90  {
91      int x;
92      tmp1(int a) {x = a;}
93      bool operator<(const tmp1& a) const
94      {
95          return x < a.x; //大顶堆
96      }
97  };
98
99  //方法2
100 struct tmp2 //重写仿函数
101 {
102     bool operator() (tmp1 a, tmp1 b)
103     {
104         return a.x < b.x; //大顶堆
105     }
106 };
107
108 int main()
109 {
110     tmp1 a(1);
111     tmp1 b(2);
112     tmp1 c(3);
113     priority_queue<tmp1> d;
114     d.push(b);
115     d.push(c);
116     d.push(a);
117     while (!d.empty())
118     {
119         cout << d.top().x << '\n';
120         d.pop();
121     }
122     cout << endl;
123
124     priority_queue<tmp1, vector<tmp1>, tmp2> f;
125     f.push(c);
126     f.push(b);
127     f.push(a);
128     while (!f.empty())
129     {
```

```
130         cout << f.top().x << '\n';
131         f.pop();
132     }
133 }
```

7.1.11 bitset

```
1  bitset<4> bitset1;    //无参构造，长度为 4 ，默认每一位为 0
2  bitset<8> bitset2(12);    //长度为 8 ，二进制保存，前面用 0 补充
3  string s = "100101";
4  bitset<10> bitset3(s);    //长度为10，前面用 0 补充
5  char s2[] = "10101";
6  bitset<13> bitset4(s2);    //长度为13，前面用 0 补充
7  cout << bitset1 << endl;    //0000
8  cout << bitset2 << endl;    //00001100
9  cout << bitset3 << endl;    //0000100101
10 cout << bitset4 << endl;    //0000000010101
11
12 bitset<2> bitset1(12);    //12的二进制为1100（长度为 4 ），但bitset1的size=2，只取后面部
       分，即00
13 string s = "100101";
14 bitset<4> bitset2(s);    //s的size=6，而bitset的size=4，只取前面部分，即1001
15 char s2[] = "11101";
16 bitset<4> bitset3(s2);    //与bitset2同理，只取前面部分，即1110
17 cout << bitset1 << endl;    //00
18 cout << bitset2 << endl;    //1001
19 cout << bitset3 << endl;    //1110
20
21 bitset<4> foo (string("1001"));
22 bitset<4> bar (string("0011"));
23
24 cout << (foo^=bar) << endl; // 1010 (foo对bar按位异或后赋值给foo)
25 cout << (foo&=bar) << endl; // 0010 (按位与后赋值给foo)
26 cout << (foo|=bar) << endl; // 0011 (按位或后赋值给foo)
27
28 cout << (foo<<=2) << endl; // 1100 (左移 2 位，低位补 0 ，有自身赋值)
29 cout << (foo>>=1) << endl; // 0110 (右移 1 位，高位补 0 ，有自身赋值)
30
31 cout << (~bar) << endl; // 1100 (按位取反)
32 cout << (bar<<1) << endl; // 0110 (左移，不赋值)
33 cout << (bar>>1) << endl; // 0001 (右移，不赋值)
34
35 cout << (foo==bar) << endl; // false (0110==0011为false)
36 cout << (foo!=bar) << endl; // true (0110!=0011为true)
37
38 cout << (foo&bar) << endl; // 0010 (按位与，不赋值)
39 cout << (foo|bar) << endl; // 0111 (按位或，不赋值)
40 cout << (foo^bar) << endl; // 0101 (按位异或，不赋值)
41
42 //此外，可以通过 [ ] 访问元素(类似数组)，注意最低位下标为 0 ，如下：
43 bitset<4> foo ("1011");
44 cout << foo[0] << endl;    //1
```

```
cout << foo[1] << endl;    //1
cout << foo[2] << endl;    //0

bitset<8> foo ("10011011");
cout << foo.count() << endl;    //5    （count函数用来求bitset中1的位数，foo中共有 5 个 1
cout << foo.size() << endl;     //8    （size函数用来求bitset的大小，一共有 8 位
cout << foo.test(0) << endl;    //true    （test函数用来查下标处的元素是 0 还是 1，并返回 false或true，此处foo[0]为 1，返回true
cout << foo.test(2) << endl;    //false    （同理，foo[2]为 0，返回false
cout << foo.any() << endl;    //true    （any函数检查bitset中是否有 1
cout << foo.none() << endl;    //false    （none函数检查bitset中是否没有 1
cout << foo.all() << endl;    //false    （all函数检查bitset中是全部为 1

bitset<8> foo ("10011011");
cout << foo.flip(2) << endl;    //10011111    （flip函数传参数时，用于将参数位取反，本行代码将foo下标 2 处"反转"，即 0 变 1， 1 变 0
cout << foo.flip() << endl;     //01100000    （flip函数不指定参数时，将bitset每一位全部取反
cout << foo.set() << endl;       //11111111    （set函数不指定参数时，将bitset的每一位全部置为 1
cout << foo.set(3,0) << endl;    //11110111    （set函数指定两位参数时，将第一参数位的元素置为第二参数的值，本行对foo的操作相当于foo[3]=0
cout << foo.set(3) << endl;     //11111111    （set函数只有一个参数时，将参数下标处置为 1
cout << foo.reset(4) << endl;    //11101111    （reset函数传一个参数时将参数下标处置为 0
cout << foo.reset() << endl;     //00000000    （reset函数不传参数时将bitset的每一位全部置为 0

bitset<8> foo ("10011011");
string s = foo.to_string();    //将bitset转换成string类型
unsigned long a = foo.to_ulong();    //将bitset转换成unsigned long类型
unsigned long long b = foo.to_ullong();    //将bitset转换成unsigned long long类型
cout << s << endl;    //10011011
cout << a << endl;    //155
cout << b << endl;    //155
```

7.1.12 string

```
string s; //生成一个空字符串s
string s(str) //拷贝构造函数 生成str的复制品
string s(str,stridx) //将字符串str内“始于位置stridx”的部分当作字符串的初值
string s(str,stridx,strlen) //将字符串str内“始于stridx且长度顶多strlen”的部分作为字符串的初值
string s(cstr) //将C字符串作为s的初值
string s(chars,chars_len) //将C字符串前chars_len个字符作为字符串s的初值。
string s(num,c) //生成一个字符串，包含num个c字符
string s(beg,end) //以区间beg;end(不包含end)内的字符作为字符串s的初值
s.~string() //销毁所有字符，释放内存

=,assign() //赋以新值
swap() //交换两个字符串的内容
```

```
+=,append(),push_back() //在尾部添加字符
insert() //插入字符
erase() //删除字符
clear() //删除全部字符
replace() //替换字符
+ //串联字符串
==,!=,<,<=,>,>=,compare() //比较字符串
size(),length() //返回字符数量
max_size() //返回字符的可能最大个数
empty() //判断字符串是否为空
capacity() //返回重新分配之前的字符容量
reserve() //保留一定量内存以容纳一定数量的字符
[ ], at() //存取单一字符
>>,getline() //从stream读取某值
<< //将谋值写入stream
copy() //将某值赋值为一个C_string
c_str() //将内容以C_string返回
substr() //返回某个子字符串
begin() end() //提供类似STL的迭代器支持
rbegin() rend() //逆向迭代器
get_allocator() //返回配置器
```

7.1.13 二分查找

```
/*
lower_bound( )和upper_bound( )都是利用二分查找的方法在一个排好序的数组中进行查找的。
在从小到大的排序数组中，
lower_bound( begin,end,num)：从数组的begin位置到end-1位置二分查找第一个大于或等于num的数字，找到返回该数字的地址，不存在则返回end。通过返回的地址减去起始地址begin,得到找到数字在数组中的下标。
upper_bound( begin,end,num)：从数组的begin位置到end-1位置二分查找第一个大于num的数字，找到返回该数字的地址，不存在则返回end。通过返回的地址减去起始地址begin,得到找到数字在数组中的下标。
在从大到小的排序数组中，重载lower_bound()和upper_bound()
lower_bound( begin,end,num,greater<type>() ):从数组的begin位置到end-1位置二分查找第一个小于或等于num的数字，找到返回该数字的地址，不存在则返回end。通过返回的地址减去起始地址begin,得到找到数字在数组中的下标。
upper_bound( begin,end,num,greater<type>() ):从数组的begin位置到end-1位置二分查找第一个小于num的数字，找到返回该数字的地址，不存在则返回end。通过返回的地址减去起始地址begin,得到找到数字在数组中的下标。
*/
#include<bits/stdc++.h>
using namespace std;
const int maxn=100000+10;
const int INF=2*int(1e9)+10;
#define LL long long
int cmd(int a,int b){
   return a>b;
}
int main(){
   int num[6]={1,2,4,7,15,34};
   sort(num,num+6); //按从小到大排序
```

```
21      int pos1=lower_bound(num,num+6,7)-num; //返回数组中第一个大于或等于被查数的值
22      int pos2=upper_bound(num,num+6,7)-num; //返回数组中第一个大于被查数的值
23      cout<<pos1<<" "<<num[pos1]<<endl;
24      cout<<pos2<<" "<<num[pos2]<<endl;
25      sort(num,num+6,cmd); //按从大到小排序
26      int pos3=lower_bound(num,num+6,7,greater<int>())-num; //返回数组中第一个小于或等于
            被查数的值
27      int pos4=upper_bound(num,num+6,7,greater<int>())-num; //返回数组中第一个小于被查数
            的值
28      cout<<pos3<<" "<<num[pos3]<<endl;
29      cout<<pos4<<" "<<num[pos4]<<endl;
30      return 0;
31  }
```