# ACM模板

# ACM Template

学科专业：软件工程
本 科 生：刘 坤 鑫
指导教师：刘 坤 鑫

天津大学智能与计算学部
二〇一九 年九 月 二十 日

# 目　录

# 第一章　准备工作

## 1.1　头文件模板

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

#define REP(i,n) for(int i = 0; i < n; i++)
#define PER(i,n) for(int i = n-1; i >= 0; i--)
#define FOR(i,l,r) for(int i = l; i <= r; i++)
#define ROF(i,l,r) for(int i = r; i >= l; i--)
#define DEBUG(x) cout << #x << "=" << x << endl;
#define SHOW1(A,n) { REP(i,n) cout<<A[i]<<(i==n-1?'\n':' '); }
#define SHOW2(A,m,n) { REP(j,m) SHOW1(A[j],n) }
#define pb push_back
#define fi first
#define se second
#define ALL(x) x.begin(),x.end()
#define SZ(x) (int)((x).size())
typedef long long LL;
typedef unsigned long long ULL;
typedef pair<int, int> pii;
typedef pair<LL, LL> pll;
const int INF = 0x3f3f3f3f, MOD = 1000000007;
const double PI = acos(-1), EPS = 1e-15;
const int MAXN = 1e2+9, MAXM = 3e3+9;



int main()
{
#ifdef LOCAL
    //freopen("i.txt", "r", stdin);
    //freopen("o.txt", "w", stdout);
#endif //LOCAL



    return 0;
}
```

## 1.2 编译参数

```
g++ -O2 -std=c++11 liu.cpp -DLOCAL
```

## 1.3 IO外挂

适合读到文件末尾、负数

```
namespace fastIO {
#define BUF_SIZE 100000
    bool IOerror = 0;
    inline char nc() {
        static char buf[BUF_SIZE], *p1 = buf + BUF_SIZE, *pend = buf + BUF_SIZE;
        if(p1 == pend) {
            p1 = buf;
            pend = buf + fread(buf, 1, BUF_SIZE, stdin);
            if(pend == p1) {
                IOerror = 1;
                return -1;
            }
        }
        return *p1++;
    }
    template <class T>
    inline bool read(T &ret) {
        char c;
        if (c=nc(),c==EOF)return 0;
        while(c!='-'&&(c<'0'||c>'9'))c=nc();
        int sgn =(c=='-')?-1:1;
        ret=(c=='-')?0:(c - '0');
        while(c=nc(),c>='0'&&c<='9') ret=ret*10+(c-'0');
        ret *= sgn;
        return 1;
    }
    template <class T>
    inline void print(T x) {
        if(x>9) print(x/10);
        putchar(x%10+'0');
    }
#undef BUF_SIZE
};
```

适用于正整数

```
namespace IN {
    const int N=2e6+10;
```

```
    char str[N],*S=str,*T=str;
    inline char rdc()
    {
        if (S==T) {
            T=(S=str)+fread(str,1,N,stdin);
            if (S==T) return EOF; //exit(0);
        }
        return *S++;
    }
    inline int rd()
    {
        int x=0;
        char t=rdc();
        while (t>'9'||t<'0') t=rdc();
        while ('0'<=t&&t<='9') x=x*10+t-'0', t=rdc();
        return x;
    }
}
namespace OUT {
    const int N=2e6+10;
    char str[N],*T=str;
    inline void otc(char t)
    {
        if (T==str+N) {
            fwrite(str,1,N,stdout);
            T=str;
        }
        *T++=t;
    }
    inline void ot(int x)
    {
        if (x<10) otc('0'+x);
        else {
            ot(x/10);
            otc('0'+x%10);
        }
    }
    inline void otall()
    {
        fwrite(str,1,T-str,stdout);
    }
}
//example:
//int n=IN::rd();
//OUT::ot(n)
//OUT::otc('\n');
//OUT::otall();
```

## 1.4 Java大数

编译参数

```
> javac Main.java
> java Main
```

Product UVA - 10106
大数乘法
a*b

```
import java.math.*;
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner cin=new Scanner(System.in);
        while (cin.hasNext()) {
            BigInteger a=cin.nextBigInteger();
            BigInteger b=cin.nextBigInteger();
            System.out.println(a.multiply(b));
        }
    }
}
```

Integer Inquiry UVA - 424
大数加法
把数一直相加，直到读入0停止相加并输出结果

```
import java.math.*;
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner cin=new Scanner(System.in);
        BigInteger ans=new BigInteger("0");
        while (true) {
            BigInteger x=cin.nextBigInteger();
            if (x.equals(BigInteger.ZERO)) break;
            ans=ans.add(x);
        }
        System.out.println(ans);
    }
}
```

N! HDU - 1042
大数乘法
计算阶乘

```
import java.math.*;
import java.util.*;
```

```
public class Main {
    private static final int MAXN=10009;
    private static BigInteger[] fac=new BigInteger[MAXN];
    public static void main(String[] args) {
        fac[0]=BigInteger.ONE;
        for (int i=1; i<MAXN; i++) {
            fac[i]=fac[i-1].multiply(BigInteger.valueOf(i));
        }
        Scanner cin=new Scanner(System.in);
        while (cin.hasNext()) {
            int n=cin.nextInt();
            System.out.println(fac[n]);
        }
    }
}
```

If We Were a Child Again UVA - 10494
大数除法大数取模

```
import java.math.*;
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner cin=new Scanner(System.in);
        while (cin.hasNext()) {
            BigInteger a=cin.nextBigInteger();
            String op=cin.next();
            BigInteger b=cin.nextBigInteger();
            if (op.equals("/")) System.out.println(a.divide(b));
            else System.out.println(a.mod(b));
        }
    }
}
```

Exponentiation POJ - 1001
大数乘方
去除尾部的0，以及如果是0.*的形式，以.*的形式打印

```
import java.math.*;
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner cin=new Scanner(System.in);
        while (cin.hasNext()) {
            BigDecimal a=cin.nextBigDecimal();
            int b=cin.nextInt();
            String ans=a.pow(b).stripTrailingZeros().toPlainString();
            if (ans.startsWith("0")) ans=ans.substring(1);
```

```
            System.out.println(ans);
        }
    }
}
```

Eid LightOJ - 1024
最大公倍数大数/高精度 JAVA
需要System.gc()清理内存，另外大数自带gcd

```
import java.math.*;
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner cin=new Scanner(System.in);
        int T=cin.nextInt();
        for (int kase=1; kase<=T; kase++) {
            int n=cin.nextInt();
            BigInteger ans=cin.nextBigInteger();
            for (int i=1; i<n; i++) {
                BigInteger x=cin.nextBigInteger();
                ans=ans.divide(ans.gcd(x)).multiply(x);
            }
            System.out.println("Case "+kase+": "+ans);
            System.gc();
        }
    }
}
```

Cryptography Reloaded UVALive - 4353
大数开方 JAVA

```
import java.math.*;
import java.util.*;

public class Main {
    public static BigInteger sqrt(BigInteger x) {
        BigInteger div=BigInteger.ZERO.setBit(x.bitLength()/2);
        BigInteger div2=div;
        while (true) {
            BigInteger y=div.add(x.divide(div)).shiftRight(1);
            if (y.equals(div) || y.equals(div2)) {
                if (y.multiply(y).equals(x)) return y;
                else return BigInteger.valueOf(-1);
            }
            div2=div;
            div=y;
        }
    }
    public static void main(String[] args) {
        Scanner cin=new Scanner(System.in);
```

```
        int kase=0;
        while (cin.hasNext()) {
            BigInteger n,d,e,t,b,delta,p,q;
            n=cin.nextBigInteger();
            d=cin.nextBigInteger();
            e=cin.nextBigInteger();
            if (e.equals(BigInteger.ZERO)) break;
            d=d.multiply(e).subtract(BigInteger.ONE);
            for (int i=1; ; i++) {
                if (d.mod(BigInteger.valueOf(i)).signum()>0) continue;
                t=d.divide(BigInteger.valueOf(i));
                b=n.add(BigInteger.ONE).subtract(t);
                delta=b.multiply(b).subtract(BigInteger.valueOf(4).multiply(n));
                if (delta.signum()<0) continue;
                delta=sqrt(delta);
                if (delta.signum()<0) continue;
                p=b.subtract(delta).divide(BigInteger.valueOf(2));
                q=b.add(delta).divide(BigInteger.valueOf(2));
                if (p.multiply(q).equals(n)) {
                    System.out.println("Case #"+ ++kase+": "+p+" "+q);
                    break;
                }
            }
        }
    }
}
```

# 第二章 动态规划

## 2.1 数位DP

```
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));
    }
}
```

不要62 HDU - 2089

经典

统计[L,R]内不含4和62的个数。

优化一：

memset放外面，因为约束条件是62和4，是确定的，跟数本身没关系。

```
int L,R,a[10],d[10][2];
int dp(int pos, bool if6, bool limit)
{
    if (pos==-1) return 1;
    if (!limit && ~d[pos][if6]) return d[pos][if6];
    int ans=0;
    int n=limit?a[pos]:9;
    FOR(i,0,n) {
        if (i==4) continue;
        if (if6 && i==2) continue;
        ans+=dp(pos-1,i==6,limit&&i==a[pos]);
    }
    if (!limit) d[pos][if6]=ans;
    return ans;
}
int solve(int x)
{
    int cnt=0;
    while (x) {
        a[cnt++]=x%10;
        x/=10;
    }
    return dp(cnt-1,false,true);
}

int main()
{

    memset(d,-1,sizeof(d));
    while (scanf("%d%d", &L,&R)==2 && L&&R) {
        printf("%d\n", solve(R)-solve(L-1));
    }

    return 0;
}
```

F(x) HDU - 4734

定义了f(x)=d[n]*2^(n-1)+d[n-1]*2^(n-2)+...+d[2]*2+d[1]*1，其中d[k]表示正整数x的第k个数位。求[0,b]中满足f(i)¡f(a)的个数。

可以想到用d[pos][sum]表示状态，sum为当前的和，如果每次累加的话，需要每次清空d数组，会超时。

如果用d[pos][sum]表示pos位数不超过sum的数个数，每次累减，则只需要清空一次d数组即可。

```
int a,b,c[11],d[11][MAXN],ma;
int dp(int pos, int sum, bool limit)
{
    if (sum<0) return 0;
    if (pos==-1) return 1;
    if (!limit && ~d[pos][sum]) return d[pos][sum];
    int ans=0;
    int up=limit?c[pos]:9;
    FOR(i,0,up) {
        ans+=dp(pos-1,sum-i*(1<<pos),limit&&i==up);
    }
    if (!limit) d[pos][sum]=ans;
    return ans;
}

int main()
{

    memset(d,-1,sizeof(d));
    int T; scanf("%d", &T);
    FOR(i,1,T) {
        scanf("%d%d", &a,&b);
        ma=0;
        for (int j=0; a; j++) {
            ma+=(a%10)*(1<<j);
            a/=10;
        }
        int cnt=0;
        while (b) {
            c[cnt++]=b%10;
            b/=10;
        }
        printf("Case #%d: %d\n", i,dp(cnt-1,ma,true));
    }

    return 0;
}
```

Round Numbers POJ - 3252

模板

求二进制中0个数不少于1个数的数个数。类似的变形还有计数01个数相等的题目

如果0减一，如果1加一，最后判断sum不大于初始值即可。需要维护limit还需要维护一个前导零lead，以便得知什么时候能开始对sum加减。

```
int a[33],d[33][66];
int dp(int pos, int sum, bool limit, bool lead)
{
    if (pos==-1) return sum<=33;
    if (!limit && !lead && ~d[pos][sum]) return d[pos][sum];
    int ans=0, up=limit?a[pos]:1;
    FOR(i,0,up) {
        int add=i?1:-1;
        if (lead && !i) add=0;
        ans+=dp(pos-1,sum+add,limit&&i==up,lead&&!i);
    }
    if (!limit && !lead) d[pos][sum]=ans;
    return ans;
}
int solve(int x)
{
    int cnt=0;
    while (x) {
        a[cnt++]=(x&1);
        x>>=1;
    }
    return dp(cnt-1,33,true,true);
}

int main()
{

    memset(d,-1,sizeof(d));
    int L,R;
    while (scanf("%d%d", &L,&R)==2 && L&&R) {
        printf("%d\n", solve(R)-solve(L-1));
    }

    return 0;
}
```

## 2.2 斜率DP

$d_i = \min\left(-a_i x_j + y_j\right) + w_i$

$d_i = \max\left(-a_i x_j + y_j\right) + w_i$

令$b = -a_i x_j + y_j$

移项：$y_i = a_i x_j + b$

如果是求最小值，即求b最小值，问题即是在凸包的点中找一个点使得b最小

模板：

```
struct point {
    LL x,y;
    point operator-(const point& p) const {
        return (point){x-p.x,y-p.y};
    }
};
inline LL cross(const point& u, const point& v)
{
    return u.x*v.y-v.x*u.y;
}
struct dequeue {
    point q[MAXN];
    int st,ed;
    void init() { st=1; ed=0; }
    void push(const point& u) {
        while (st<ed && cross(q[ed]-q[ed-1],u-q[ed-1])<=0) ed--;
        q[++ed]=u;
    }
    point pop(const LL& k) {
        while (st<ed && -k*q[st].x+q[st].y >= -k*q[st+1].x+q[st+1].y) st++;
        return q[st];
    }
} Q;
// d[i] = -k*x[j] + y[j] + w
void solve()
{
    Q.init();
    //Q.push((point){0,0});
    FOR(i,1,n) {
        point t=Q.pop(k);
        d[i]=-k*p.x+t.y+w;
        Q.push((point){x[i],y[i]});
    }
}
```

Pearls HDU - 1300

n个珍珠要买a[i]个，价格为p[i]，单买要加上基础价10*p[i]，也可以和贵的珍珠一起买不需要付基础价

要么单买，要么连续地买，因为比如a,b,c三种，如果a要跳过b用c的价格买，那还不如用b的价格买。

d[i]=min(d[j]+(a[i]-a[j]+10)*p[i]), (j¡i)

可以用斜率优化，虽然规模其实很小。

```
int n,a[MAXN],d[MAXN];

int main()
{

    int T; scanf("%d", &T);
    while (T--) {
        scanf("%d", &n);
```

```
        FOR(i,1,n) {
            int p;
            scanf("%d%d", &a[i],&p);
            a[i]+=a[i-1];
            d[i]=INF;
            REP(j,i) d[i]=min(d[i],d[j]+(a[i]-a[j]+10)*p);
        }
        printf("%d\n", d[n]);
    }

    return 0;
}
```

MAX Average Problem HDU - 2993

经典，最大平均值问题

求区间长度至少为k的最大的区间平均值

NOI2004年周源的论文《浅谈数形结合思想在信息学竞赛中的应用》

参考白书第一章 Average, Seoul, UVALive - 4726，或参考紫书例题8-9

这道题很坑爹就是规模太大了，并且用int会WA

用了能读到文件末尾的输入挂，对代码常数进行了很细致的优化（详细对比汝佳源码和这份代码），需要用LL否则会WA

```
namespace IN {
    const int N=2e6+10;
    char str[N],*S=str,*T=str;
    inline char rdc()
    {
        if (S==T) {
            T=(S=str)+fread(str,1,N,stdin);
            if (S==T) exit(0);
        }
        return *S++;
    }
    inline int rd()
    {
        int x=0;
        char t=rdc();
        while (t>'9'||t<'0') t=rdc();
        while ('0'<=t&&t<='9') x=x*10+t-'0', t=rdc();
        return x;
    }
}
int n,L,a[MAXN];
int q[MAXN];
LL cmp(int x1, int x2, int x3, int x4)
{
    return (LL)(a[x2]-a[x1])*(x4-x3) - (LL)(a[x4]-a[x3])*(x2-x1);
}
bool cmp2(int x1, int x3, int x4)
{
    return (LL)(a[x4]-a[x1])*(x4-x3) >= (LL)(a[x4]-a[x3])*(x4-x1);
}
```

```
int main()
{

    while (1) {
        n=IN::rd(); L=IN::rd();
        FOR(i,1,n) a[i]=IN::rd(), a[i]+=a[i-1];
        int st=1, ed=0, ansL=1, ansR=L;
        FOR(i,L,n) {
            while (st<ed && cmp2(q[ed-1]-1,q[ed]-1,i-L)) ed--;
            q[++ed]=i-L+1;
            while (st<ed && cmp2(q[st+1]-1,q[st]-1,i)) st++;
            if (cmp(q[st]-1,i,ansL-1,ansR)>0) {
                ansL=q[st];
                ansR=i;
            }
        }
        printf("%.2f\n", (double)(a[ansR]-a[ansL-1])/(ansR-ansL+1));
    }

    return 0;
}
```

Picnic Cows HDU - 3045

给一些数，把他们分组，每组不少于T个数，每个组每个数分别减去这个组最小的数之后求和为这个组的费用，所有组费用加起来为总费用，求总费用最小。

先把数排序，斜率dp的套路就显现出来了——每个数要么单独选，要么与前面连续几个数一起选

d[i]=min(d[j]+S[i]-S[j]-(i-j)*a[j+1])

=min(-i*a[j+1]+j*a[j+1]+d[j]-S[j])+S[i]

套模板即可，初始化条件是(a[1],0)，另外由于要求每个组数不少于L个，所以先把d预处理为-1，当d[i-L+1]不是-1时才进队

```
struct point {
    LL x,y;
    point operator-(const point& p) const {
        return (point){x-p.x, y-p.y};
    }
};
inline LL cross(const point& u, const point& v) {
    return u.x*v.y - v.x*u.y;
}
struct dequeue {
    point q[MAXN];
    int st,ed;
    void init() { st=1, ed=0; }
    void push(const point& u) {
        while (st<ed && cross(q[ed]-q[ed-1],u-q[ed-1])<=0) ed--;
        q[++ed]=u;
    }
    point pop(const LL& k) {
```

```
        while (st<ed && -k*q[st].x+q[st].y >= -k*q[st+1].x+q[st+1].y) st++;
        return q[st];
    }
} Q;
int n,L;
LL a[MAXN],S[MAXN],d[MAXN];
void solve()
{
    Q.init();
    FOR(i,0,n) d[i]=-1;
    Q.push((point){a[1],0});
    FOR(i,L,n) {
        LL k=i;
        LL w=S[i];
        point t=Q.pop(k);
        d[i]=-k*t.x+t.y+w;
        int j=i-L+1;
        if (~d[j]) Q.push((point){a[j+1],j*a[j+1]-S[j]+d[j]});
    }
    printf("%lld\n", d[n]);
}

int main()
{

    while (scanf("%d%d", &n,&L)==2) {
        FOR(i,1,n) scanf("%lld", &a[i]);
        sort(a+1,a+1+n);
        FOR(i,1,n) S[i]=a[i]+S[i-1];
        solve();
    }

    return 0;
}
```

## 2.3 区间DP

四边形不等式优化

```
for (int i=0; i<=n; i++) {
    g[i][i]=i;
}
for (int len=1; len<=n; len++) {
    for (int i=1; i<=n-len; i++) {
        int j=i+len;
        f[i][j]=INF;
        for (int k=g[i][j-1]; k<=g[i+1][j]; k++) {
            if (f[i][j]>f[i][k]+f[k+1][j]+w[i][j]) {
                f[i][j]=f[i][k]+f[k+1][j]+w[i][j];
                g[i][j]=k;
```

```
            }
        }
    }
}
```

Palindrome subsequence HDU - 4632

求回文子序列个数

经典

f[i][j]=f[i+1][j]+f[i][j-1]-f[i+1][j-1]，由容斥原理得到

if (s[i]==s[j]) f[i][j]+=f[i+1][j-1]+1，如果s[i]==s[j]，说明区间[i+1,j-1]所有子序列加上i和j又可以得到新的子序列，个数为f[i+1][j-1]，还有加上1个s[i]和s[j]构成的子序列。

注意求模时产生负数的处理

```
int n,f[MAXN][MAXN];
char s[MAXN];

int main()
{

    int T; scanf("%d", &T);
    FOR(kase,1,T) {
        scanf("%s", s+1);
        n=strlen(s+1);
        FOR(i,1,n) f[i][i]=1;
        FOR(len,1,n)FOR(i,1,n-len) {
            int j=i+len;
            f[i][j]=f[i][j-1]+f[i+1][j]-f[i+1][j-1];
            if (s[i]==s[j]) f[i][j]+=f[i+1][j-1]+1;
            f[i][j]%=MOD;
        }
        printf("Case %d: %d\n", kase,(f[1][n]+MOD)%MOD);
    }

    return 0;
}
```

Brackets POJ - 2955

括号匹配

经典

给一个括号组成的序列，问最长的合法括号子序列有多长

如果序列两端是一对括号，那么方程可以由去掉这对括号的子序列转移而来，并且还有以下转移：d[i][j]=d[i][k]+d[k+1][j]

```
int main()
{

    while (scanf("%s", s+1)==1 && s[1]!='e') {
        n=strlen(s+1);
        FOR(i,1,n) {
```

```
            switch (s[i]) {
                case '(': a[i]=-2; break;
                case '[': a[i]=-1; break;
                case ']': a[i]=1; break;
                case ')': a[i]=2; break;
            }
        }
        FOR(len,1,n)FOR(i,1,n-len) {
            int j=i+len;
            d[i][j]=d[i+1][j-1];
            if (a[i]<0 && a[i]+a[j]==0) d[i][j]+=2;
            FOR(k,i,j) d[i][j]=max(d[i][j], d[i][k]+d[k+1][j]);
        }
        printf("%d\n", d[1][n]);
    }

    return 0;
}
```

# 第三章 数学

## 3.1 数论

### 3.1.1 素数筛

```
int vis[MAXN],prime[MAXN],tot;
void getPrime(int n)
{
    FOR(i,2,n) {
        if (!vis[i]) prime[tot++]=i;
        REP(j,tot) {
            if (i*prime[j]>n) break;
            vis[i*prime[j]]=1;
            if (i%prime[j]==0) break;
        }
    }
}
```

### 3.1.2 素性测试

test_time 为测试次数,建议设为不小于 8 的整数以保证正确率,但也不宜过大,否则会影响效率

```
bool millerRabbin(int n) {
  if (n < 3) return n == 2;
  int a = n - 1, b = 0;
  while (a % 2 == 0) a /= 2, ++b;
  for (int i = 1, j; i <= test_time; ++i) {
    int x = rand() % (n - 2) + 2, v = quickPow(x, a, n);
    if (v == 1 || v == n - 1) continue;
    for (j = 0; j < b; ++j) {
      v = (long long)v * v % n;
      if (v == n - 1) break;
    }
    if (j >= b) return 0;
  }
  return 1;
}
```

### 3.1.3 反素数

如果某个正整数 $n$ 满足如下条件，则称为是反素数：任何小于 $n$ 的正数的约数个数都小于 $n$ 的约数个数

https://blog.csdn.net/ACdreamers/article/details/25049767

Number With The Given Amount Of Divisors CodeForces - 27E

求约数个数为n的最小整数

```
typedef unsigned long long ull;
int p[16]={2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53};
ull ans;
int n;
void dfs(int dep, ull now, int num, int up)
{
    if (num>n || dep>=16) return;
    if (num==n && ans>now) {
        ans=now;
        return;
    }
    FOR(i,1,up) {
        if (now/p[dep]>ans) break;
        dfs(dep+1,now=now*p[dep],num*(i+1),i);
    }
}

int main()
{

    while (scanf("%d", &n)==1) {
        ans=~0ULL;
        dfs(0,1,1,60);
        printf("%llu\n", ans);
    }

    return 0;
}
```

More Divisors ZOJ - 2562

求n以内因子数最多的数

```
typedef unsigned long long ull;
int p[16]={2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53};
ull n,ans,ans_num;
void dfs(int dep, ull now, int num, int up)
{
    if (dep>=16 || now>n) return;
    if (num>ans_num || (num==ans_num && ans>now)) {
        ans=now;
        ans_num=num;
    }
```

```
    FOR(i,1,up) {
        if (now*p[dep]>n) break;
        dfs(dep+1,now*=p[dep],num*(i+1),i);
    }
}

int main()
{

    while (scanf("%llu", &n)==1) {
        ans_num=0;
        dfs(0,1,1,60);
        printf("%llu\n", ans);
    }

    return 0;
}
```

小明系列故事——未知剩余系 HDU - 4542
给出一个数K和两个操作
如果操作是0，就求出一个最小的正整数X，满足X的约数个数为K。
如果操作是1，就求出一个最小的X，满足X的约数个数为X-K。
https://blog.csdn.net/qq_37025443/article/details/75007451
上面的模板有错，以这个为准

```
const LL inf=(1LL<<62)+1;
int p[16]={2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53};
LL ans;
int op,n,d[MAXN];
void init()
{
    int m=MAXN-5;
    FOR(i,1,m) d[i]=i;
    FOR(i,1,m) {
        for (int j=i; j<=m; j+=i) d[j]--;
        if (!d[d[i]]) d[d[i]]=i;
        d[i]=0;
    }
}
void dfs(int dep, LL now, int num, int up)
{
    if (num>n) return;
    if (num==n && ans>now) { ans=now; return; }
    FOR(i,1,up) {
        if (ans/p[dep]<now || num*(i+1)>n) break;
        now*=p[dep];
        if (n%(num*(i+1))==0) dfs(dep+1,now,num*(i+1),i);
    }
}

int main()
{
```

```
    init();
    int T; scanf("%d", &T);
    FOR(kase,1,T) {
        scanf("%d%d", &op,&n);
        if (op) ans=d[n];
        else { ans=inf; dfs(0,1,1,62); }
        printf("Case %d: ", kase);
        if (!ans) puts("Illegal");
        else if (ans>=inf) puts("INF");
        else printf("%lld\n", ans);
    }

    return 0;
}
```

### 3.1.4 欧几里得算法

```
//ax+by=d=gcd(a,b)
LL gcd(LL a, LL b) { return b==0?a:gcd(b,a%b); }
LL lcm(LL a, LL b) { return a/gcd(a,b)*b; }
void ex_gcd(LL a, LL b, LL& d, LL& x, LL& y)
{
    if (!b) x=1, y=0, d=a;
    else ex_gcd(b,a%b,d,y,x), y-=x*(a/b);
}
```

### 3.1.5 快速幂取模

```
LL powMod(LL x, LL e)
{
    if (!e) return 1;
    return e&1 ? powMod(x,e-1)*x%MOD : powMod(x*x%MOD,e>>1);
}
LL powMod(LL x, LL e, LL p)
{
    if (!e) return 1;
    return e&1 ? powMod(x,e-1,p)*x%p : powMod(x*x%p,e>>1,p);
}
```

### 3.1.6 欧拉函数

```
int vis[MAXN],prime[MAXN],phi[MAXN],tot;
```

```
void getphi(int n)
{
    phi[1]=1;
    FOR(i,2,n) {
        if (!vis[i]) prime[tot++]=i, phi[i]=i-1;
        REP(j,tot) {
            if (i*prime[j]>n) break;
            vis[i*prime[j]]=1;
            if (i%prime[j]==0) {
                phi[i*prime[j]]=phi[i]*prime[j];
                break;
            } else phi[i*prime[j]]=phi[i]*(prime[j]-1);
        }
    }
}
```

### 3.1.7 逆元

```
void ex_gcd(LL a, LL b, LL& d, LL& x, LL& y)
{
    if (!b) x=1, y=0, d=a;
    else ex_gcd(b,a%b,d,y,x), y-=x*(a/b);
}
LL inv(LL a, LL p)
{
    LL d,x,y;
    ex_gcd(a,p,d,x,y);
    return d==1?(x+p)%p:-1;
}
LL inv(LL a) { return powMod(a,MOD-2); }
```

线性递推求乘法逆元

```
int inv[MAXN];
void getinv(int n, LL p)
{
    inv[1]=1;
    FOR(i,2,n) inv[i]=(p-p/i)*inv[p%i]%p;
}
```

阶乘逆元

```
LL powMod(LL x, LL e)
{
    if (!e) return 1;
    return e&1 ? powMod(x,e-1)*x%MOD : powMod(x*x%MOD,e>>1);
}
LL fac[MAXN],inv[MAXN];
```

```
void getinv(int n)
{
    fac[0]=1;
    FOR(i,1,n) fac[i]=fac[i-1]*i%MOD;
    inv[0]=1;
    inv[n]=powMod(fac[n],MOD-2);
    ROF(i,1,n-1) inv[i]=inv[i+1]*(i+1)%MOD;
}
```

### 3.1.8 模方程

模板
中国剩余定理，孙子定理
//n个方程：x=a[i](mod m[i]) (0¡=i¡n)

```
void ex_gcd(LL a, LL b, LL& d, LL& x, LL& y)
{
    if (!b) x=1, y=0, d=a;
    else ex_gcd(b,a%b,d,y,x), y-=x*(a/b);
}
LL china(int n, int* a, int* m)
{
    LL M=1,d,y,x=0;
    REP(i,n) M*=m[i];
    REP(i,n) {
        LL w=M/m[i];
        ex_gcd(m[i],w,d,d,y);
        x=(x+y*w*a[i])%M;
    }
    return (x+M)%M;
}
```

离散对数
又叫BSGS算法（Shank's Baby-Step-Giant-Step Algorithm，Shank的大步小步算法）
//求解模方程aŷ=b(mod n)。n为素数，无解返回-1

```
void ex_gcd(LL a, LL b, LL& d, LL& x, LL& y)
{
    if (!b) x=1, y=0, d=a;
    else ex_gcd(b,a%b,d,y,x), y-=x*(a/b);
}
LL inv(LL a, LL p)
{
    LL d,x,y;
    ex_gcd(a,p,d,x,y);
    return d==-1?(x+p)%p:-1;
}
LL powMod(LL x, LL e, LL p)
{
```

```
    if (!e) return 1;
    return e&1 ? powMod(x,e-1,p)*x%p : powMod(x*x%p,e>>1,p);
}
//a^x=b(mod n), n is prime
int log_mod(int a, int b, int n)
{
    int m=sqrt(n+0.5);
    int v=inv(powMod(a,m,n),n);
    map<int,int> x;
    x[1]=0;
    int e=1;
    for (int i=1; i<m; i++) {
        e=e*a%n;
        if (!x.count(e)) x[e]=i;
    }
    REP(i,m) {
        if (x.count(b)) return i*m+x[b];
        b=b*v%n;
    }
    return -1;
}
```

## 3.2 博弈论

https://www.bilibili.com/video/av9114486?from=search&seid=1678403368497689776
Brave Game HDU - 1846
Bash博弈
有n个石子，每人最多取m个，轮流操作
n%(m+1)

```
int main()
{

    int T; scanf("%d", &T);
    while (T--) {
        int a,b; scanf("%d%d", &a,&b);
        if (a%(b+1)) puts("first");
        else puts("second");
    }

    return 0;
}
```

取石子游戏 HDU - 2516
斐波那契博弈裸题
n个石子轮流取，第一次可以取任意多个但不能全部取完，以后每次取石子不能超过上次取数的两倍，取完者胜
如果是斐波那契数，必败，否则必胜

```
LL n,fib[55];

int main()
{

    fib[1]=1;
    FOR(i,2,54) fib[i]=fib[i-1]+fib[i-2];
    while (cin>>n && n) {
        bool ok=true;
        FOR(i,2,54) {
            if (fib[i]==n) ok=false;
            if (fib[i]>n) break;
        }
        puts(ok?"First win":"Second win");
    }

    return 0;
}
```

威佐夫博弈(Wythoff Game)：两堆物品，两人轮流从某堆或者同时在两堆物品中取同样多的物品，至少一个，不能操作者输

奇异局势：必败的局势形如(ak,bk)，ak为之前未出现过的最小自然数，bk=ak+k

判断奇异局势：(b-a)*(sqrt(5)+1)/2==a

取石子游戏 POJ - 1067

威佐夫博弈裸题

```
int main()
{

    int a,b;
    while (scanf("%d%d", &a,&b)==2) {
        if (a>b) swap(a,b);
        int c=floor((b-a)*(sqrt(5)+1)/2);
        puts(a==c?"0":"1");
    }

    return 0;
}
```

Nim游戏：

奇异局势：异或和等于0

奇异局势的转换：假设异或和为sum，对于任意一个数ai，如果sumâi¡ai，则可以拿走ai-(sumâi)个数。

Being a Good Boy in Spring Festival HDU - 1850

Nim游戏裸题

如果有解，输出第一步可行解的方案数

```

```

```
int n,a[MAXN];

int main()
{

    while (scanf("%d", &n)==1 && n) {
        int ans=0,cnt=0;
        REP(i,n) scanf("%d", &a[i]), ans^=a[i];
        if (!ans) puts("0");
        else {
            REP(i,n) {
                int t=ans^a[i];
                if (t<a[i]) cnt++;
            }
            printf("%d\n", cnt);
        }
    }

    return 0;
}
```

Good Luck in CET-4 Everybody!HDU - 1847

SG函数

SG函数由记忆化搜索得到

```
int n,sg[MAXN],a[22];
int mex(int i)
{
    if (~sg[i]) return sg[i];
    int vis[MAXN];
    memset(vis,0,sizeof(vis));
    REP(j,11) {
        int k=i-a[j];
        if (k<0) break;
        vis[mex(k)]=1;
    }
    REP(j,MAXN) if (!vis[j]) { sg[i]=j; break; }
    return sg[i];
}

int main()
{

    a[0]=1;
    FOR(i,1,10) a[i]=a[i-1]*2;
    memset(sg,-1,sizeof(sg));
    while (scanf("%d", &n)==1) {
        puts(mex(n)?"Kiki":"Cici");
    }

    return 0;
}
```

kuangbin博弈

Game of CS LightOJ - 1355

green博弈变形

参考了网上题解

green博弈：给定一颗树，每次可以删掉一些边并且连带删掉子树，不能操作者输。此题稍有不同，每条边有一个权值，每次只能使权值减一，减为零即可删掉整个子树

结论：如果边权为1，即green博弈，sg(u)≐sg(v)+1，如果边权大于1，只有是奇数的时候有贡献：sg(u)≐sg(v)Î

```
int to[MAXN],dis[MAXN],f[MAXN],nxt[MAXN],tot;
void init()
{
    tot=0;
    memset(f,-1,sizeof(f));
}
void add(int u, int v, int w)
{
    to[tot]=v;
    dis[tot]=w;
    nxt[tot]=f[u];
    f[u]=tot++;
}
int n;
int sg(int u, int fa)
{
    int ans=0;
    for (int i=f[u]; ~i; i=nxt[i]) {
        int v=to[i], w=dis[i];
        if (v==fa) continue;
        if (w==1) ans^=sg(v,u)+1;
        else ans^=sg(v,u)^(w&1);
    }
    return ans;
}

int main()
{

    int T; scanf("%d", &T);
    FOR(kase,1,T) {
        init();
        scanf("%d", &n);
        REP(i,n-1) {
            int x,y,z; scanf("%d%d%d", &x,&y,&z);
            add(x,y,z);
            add(y,x,z);
        }
        printf("Case %d: %s\n", kase,sg(0,-1)?"Emily":"Jolly");
    }

    return 0;
}
```

Misere Nim LightOJ - 1253
反Nim博弈裸题
反Nim博弈：取到最后的石子的人输
如果全是1，当且仅当异或和为0必胜，否则异或和非0必胜。

```
int a[MAXN];

int main()
{

    int T; scanf("%d", &T);
    FOR(kase,1,T) {
        int n; scanf("%d", &n);
        REP(i,n) scanf("%d", &a[i]);
        int sum=0;
        REP(i,n) sum^=a[i];
        bool flag=true;
        REP(i,n) if (a[i]!=1) flag=false;
        printf("Case %d: %s\n", kase,((flag&&!sum)||(!flag&&sum))?"Alice":"Bob");
    }

    return 0;
}
```

### 3.2.1 组合游戏总结

Nim游戏：简单来说可以是n个数，轮流操作，每次选择一个数减小或者删掉，不能操作者输。
1. 两个游戏者轮流操作
2. 游戏的状态集有限，并且不管双方怎么走，都不会再现以前出现过的状态。
3. 谁不能操作谁输。
相关术语：
1. 一个状态是必败状态当且仅当它的所有后继都是必败状态。
2. 一个状态是必胜状态当且仅当它至少有一个后继是必败状态。
暴力解法：根据状态集，dp
Ferguson游戏：Nim的特例？
Chomp!游戏：有一个m*n的棋盘，每次可以取走一个方块并拿走这个方块右边及上边的所有方块，拿到最后一块方块的人输。
结论：除非开局只有一个方块，否则先手必胜
证明：假设后手有必胜决策可以达到必胜状态，则先手必可以抢先后手达到这个必胜状态，矛盾。
约数游戏：有1 n个数，轮流拿一个数并且拿走拿走它的约数，拿到最后一个数的胜。
结论：类似Chomp!游戏，先手必胜。
Bouton定理：状态(x_1,x_2,x_3)是必败状态当且仅当x_1x̂_2x̂_3=0，这个结果也称Nim和，这个定理可以推广到n维状态。

先手胜操作步骤：异或结果最高位的1记为第k位，找一个第k位为1的数，把它修改（注意第k位变为0，数肯定减小了），使得异或结果为0

证明：归纳法

组合游戏的和：由k个组合游戏构成的新游戏。

SG函数：SG(x)=mex(S)，S为状态x的后继状态集，mex函数表示不在”S所有元素的SG函数值”中的最小非负整数。

SG定理：SG(x)的值等于各子游戏的Nim和。

## 3.3 矩阵

例题 22 递推关系 Recurrences UVA - 10870

矩阵递推式友矩阵(companion matrix) 经典模板

```
const int maxn=16;
int n,x,mo,a[22],f[22];
struct mat {
    LL a[maxn][maxn];
    mat(LL x=0){ memset(a,0,sizeof(a)); REP(i,n) a[i][i]=x; }
    mat operator*(const mat& T) const {
        mat res;
        REP(i,n)REP(k,n) {
            LL r=a[i][k];
            REP(j,n) res.a[i][j]+=r*T.a[k][j];
        }
        REP(i,n)REP(j,n) res.a[i][j]%=mo;
        return res;
    }
    mat operator^(LL x) const {
        mat res(1), bas{*this};
        for (; x; x>>=1) {
            if (x&1) res=res*bas;
            bas=bas*bas;
        }
        return res;
    }
    void print(){
        REP(i,n){
            REP(j,n) printf("%lld\t", a[i][j]);
            puts("");
        }
        puts("");
    }
};

int main()
{

    while (scanf("%d%d%d", &n,&x,&mo)==3) {
        if (n==0 && x==0 && mo==0) break;
        REP(i,n) scanf("%d", &a[i]), a[i]%=mo;
```

```
        REP(i,n) scanf("%d", &f[i]), f[i]%=mo;
        if (x<=n) { printf("%d\n", f[x-1]); continue; }
        mat M;
        REP(i,n-1) M.a[i][i+1]=1;
        REP(j,n) M.a[n-1][j]=a[n-1-j];
        M=M^(x-n);
        LL ans=0;
        REP(j,n) ans+=M.a[n-1][j]*f[j];
        printf("%lld\n", ans%mo);
    }

    return 0;
}
```

例题 23 细胞自动机 Cellular Automaton, NEERC 2006, UVALive - 3704
矩阵循环矩阵模板
分析见白书。循环矩阵的性质使得循环矩阵复杂度可降为O(n2̂lgk)

```
int n,m,d,k,a[555];
struct mat {
    static const int maxn=505;
    LL a[maxn];
    LL& operator()(int i, int j) { return a[(j-i+n)%n]; }
    mat(LL x=0){ memset(a,0,sizeof(a)); a[0]=x; }
    mat operator*(mat& T) {
        mat res;
        REP(i,1)REP(k,n) {
            LL r=(*this)(i,k);
            REP(j,n) res(i,j)+=r*T(k,j);
        }
        REP(i,1)REP(j,n) res(i,j)%=m;
        return res;
    }
    mat operator^(LL x) const {
        mat res(1), bas{*this};
        for (; x; x>>=1) {
            if (x&1) res=res*bas;
            bas=bas*bas;
        }
        return res;
    }
    void print() {
        REP(i,n) printf("%lld ", a[i]);
        puts("");
    }
};

int main()
{

    while (scanf("%d%d%d%d", &n,&m,&d,&k)==4) {
        REP(i,n) scanf("%d", &a[i]);
        mat M(1);
```

```
        REP(i,d) M.a[1+i]=M.a[n-1-i]=1;
        M=M^k;
        REP(j,n) {
            LL ans=0;
            REP(i,n) ans+=M(i,j)*a[i];
            printf("%lld%c", ans%m,j==n-1?'\n':' ');
        }
    }

    return 0;
}
```

例题 24 随机程序 Back to Kernighan-Ritchie UVA - 10828

线性方程组高斯消元高斯-约当消元模板

含无穷解和0解，当A[i][i]=A[i][n]=0时，x[i]=0，当A[i][i]=0时，x[i]为正无穷

```
typedef double Matrix[MAXN][MAXN];
void gauss_jordan(Matrix A, int n)
{
    REP(i,n) {
        int r=i;
        FOR(j,i+1,n-1) if (fabs(A[j][i])>fabs(A[r][i])) r=j;
        if (fabs(A[r][i])<eps) continue;
        if (r!=i) FOR(j,0,n) swap(A[r][j], A[i][j]);
        REP(k,n) if (k!=i) ROF(j,i,n) A[k][j]-=A[k][i]/A[i][i]*A[i][j];
    }
}
Matrix A;
int n,d[MAXN],kase;
vector<int> pre[MAXN];
int inf[MAXN];

int main()
{

    while (scanf("%d", &n)==1 && n) {
        memset(d,0,sizeof(d));
        REP(i,n) pre[i].clear();
        int a,b;
        while (scanf("%d%d", &a,&b)==2 && a) {
            a--, b--;
            d[a]++;
            pre[b].pb(a);
        }
        memset(A,0,sizeof(A));
        REP(i,n) {
            A[i][i]=1;
            REP(j,SZ(pre[i])) A[i][pre[i][j]]-=1.0/d[pre[i][j]];
            if (i==0) A[i][n]=1;
        }
        gauss_jordan(A,n);
        memset(inf,0,sizeof(inf));
        PER(i,n) {
```

```
            if (fabs(A[i][i])<eps && fabs(A[i][n])>eps) inf[i]=1;
            FOR(j,i+1,n-1) if (fabs(A[i][j])>eps && inf[j]) inf[i]=1;
        }
        int q; scanf("%d", &q);
        printf("Case #%d:\n", ++kase);
        while (q--) {
            int u; scanf("%d", &u); u--;
            if (inf[u]) puts("infinity");
            else printf("%.3f\n", fabs(A[u][u])<eps ? 0 : A[u][n]/A[u][u]);
        }
    }

    return 0;
}
```

# 第四章 数据结构

## 4.1 并查集与最小生成树

```
int n, m, u[MAXM], v[MAXM], w[MAXM], r[MAXM], p[MAXN];
bool cmp(const int i, const int j) { return w[i] < w[j]; }
int Find(int x) { return p[x] == x ? x : p[x] = Find(p[x]); }
bool Union(int i, int j) { int x = Find(i), y = Find(j); if (x != y) p[x] = y; return x !=
    y; }
int Kruskal()
{
    REP(i,m) r[i] = i;
    sort(r, r+m, cmp);
    FOR(i,1,n) p[i] = i;
    int ans = 0;
    REP(i,m) {
        int e = r[i];
        if (Union(u[e], v[e])) ans += w[e];
    }
    return ans;
}

int main()
{

    while (~scanf("%d", &n)) {
        m = 0;
        FOR(i,1,n)FOR(j,1,n) {
            scanf("%d", &w[m]);
            if (j > i) {
                u[m] = i, v[m] = j;
                m++;
            }
        }
        printf("%d\n", Kruskal());
    }

    return 0;
}
```

POJ - 1182
经典题目：食物链
带权并查集，种类并查集
https://blog.csdn.net/niushuai666/article/details/6981689
不仅维护父结点，还维护与父结点的关系relation。相应的Find和Union操作不仅要更新集

合域，还要更新关系域。此题巧妙利用0，1，2三个数表示关系，并利用矢量可加性来维护关系域。

POJ数据有问题，输入只有一组。多组输入会WA到去世。要不是去了洛谷OJ上交了一发发现有一组数据有多余输出，要没看到POJ上的讨论，简直WA到怀疑人生。

```
int n, m, p[MAXN], r[MAXN]; //0 equal, 1 eat, 2 eated
int Find(int x)
{
    if (p[x] == x) return x;
    int root = Find(p[x]);
    r[x] = (r[x]+r[p[x]])%3;
    return p[x] = root;
}
bool Union(int op, int i, int j)
{
    int x = Find(i), y = Find(j);
    if (x == y) return r[i] == (op-1+r[j])%3;
    p[x] = y;
    r[x] = (-r[i]+op+2+r[j])%3;
    return true;
}

int main()
{

    scanf("%d%d", &n, &m);
    FOR(i,1,n) p[i] = i;
    int ans = 0;
    while (m--) {
        int op, x, y; scanf("%d%d%d", &op, &x, &y);
        if (x > n || y > n) ans++;
        else if (op == 2 && x == y) ans++;
        else if (!Union(op,x,y)) ans++;
    }
    printf("%d\n", ans);

    return 0;
}
```

## 4.2 树状数组

```
struct BIT {
    LL n,c[MAXN];
    void init(int n){ this->n=n; memset(c,0,sizeof(c)); }
    inline int lowbit(int x){ return x&-x; }
    void add(int x, LL d){ for (; x<=n; x+=lowbit(x)) c[x]+=d; }
    LL sum(int x){ LL ans=0; for (; x; x-=lowbit(x)) ans+=c[x]; return ans; }
    LL sum(int l, int r){ return sum(r)-sum(l-1); }
```

```
} bit;
```

## 4.3 RMQ

如果是求下标而不是值，可以重载min函数

```
int n, a[MAXN], d[MAXN][20];
void RMQ_init()
{
    REP(i,n) d[i][0] = a[i];
    for (int j = 1; (1<<j) < n; j++) {
        for (int i = 0; i+(1<<j)-1 < n; i++) {
            d[i][j] = min(d[i][j-1], d[i+(1<<(j-1))][j-1]);
        }
    }
}
int RMQ(int L, int R)
{
    int k = 0;
    while ((1<<(k+1)) <= R-L+1) k++;
    return min(d[L][k], d[R-(1<<k)+1][k]);
}
```

## 4.4 线段树

单点加法区间查询最大子段和

```
#define ls x<<1
#define rs x<<1|1
#define mid (l+r)/2
struct node {
    LL sum,ma,lma,rma;
    void update(LL x) {
        sum+=x;
        ma+=x;
        lma+=x;
        rma+=x;
    }
} T[MAXN<<2];
void up(int x) {
    T[x].sum=T[ls].sum+T[rs].sum;
    T[x].ma=max(T[ls].rma+T[rs].lma,max(T[ls].ma,T[rs].ma));
    T[x].lma=max(T[ls].lma,T[ls].sum+T[rs].lma);
    T[x].rma=max(T[rs].rma,T[rs].sum+T[ls].rma);
}
void build(int x, int l, int r) {
```

```
    if (l==r) T[x]={0,0,0,0};
    else {
        build(ls,l,mid);
        build(rs,mid+1,r);
        up(x);
    }
}
void modify(int x, int l, int r, int p, LL val) {
    if (l==r) T[x].update(val);
    else {
        if (p<=mid) modify(ls,l,mid,p,val);
        else modify(rs,mid+1,r,p,val);
        up(x);
    }
}
node query(int x, int l, int r, int ql, int qr) {
    if (ql<=l && r<=qr) return T[x];
    if (qr<=mid) return query(ls,l,mid,ql,qr);
    if (mid<ql) return query(rs,mid+1,r,ql,qr);
    node ll=query(ls,l,mid,ql,qr);
    node rr=query(rs,mid+1,r,ql,qr);
    return {
        ll.sum+rr.sum,
        max(ll.rma+rr.lma,max(ll.ma,rr.ma)),
        max(ll.lma,ll.sum+rr.lma),
        max(rr.rma,rr.sum+ll.rma),
    };
}
```

单点修改区间最值区间求和

```
int n,a[MAXN],val[MAXN];
#define ls x<<1
#define rs x<<1|1
#define mid (l+r)/2
struct node {
    int sum,ma;
} T[MAXN<<2];
void up(int x) {
    T[x].sum=T[ls].sum+T[rs].sum;
    T[x].ma=max(T[ls].ma,T[rs].ma);
}
void build(int x, int l, int r) {
    if (l==r) T[x]={a[l],a[l]};
    else {
        build(ls,l,mid);
        build(rs,mid+1,r);
        up(x);
    }
}
void update(int x, int l, int r, int p, int val) {
    if (l==r) T[x]={val,val};
    else {
```

```
        if (p<=mid) update(ls,l,mid,p,val);
        else update(rs,mid+1,r,p,val);
        up(x);
    }
}
int query_sum(int x, int l, int r, int ql, int qr) {
    if (ql<=l && r<=qr) return T[x].sum;
    else {
        int ans=0;
        if (ql<=mid) ans+=query_sum(ls,l,mid,ql,qr);
        if (mid<qr) ans+=query_sum(rs,mid+1,r,ql,qr);
        return ans;
    }
}
int query_max(int x, int l, int r, int ql, int qr) {
    if (ql<=l && r<=qr) return T[x].ma;
    else {
        int ans=-INF;
        if (ql<=mid) ans=max(ans,query_max(ls,l,mid,ql,qr));
        if (mid<qr) ans=max(ans,query_max(rs,mid+1,r,ql,qr));
        return ans;
    }
}
```

区间修改区间求和

```
int n;
#define ls x<<1
#define rs x<<1|1
#define mid (l+r)/2
struct node {
    int l,r,sum,lazy;
    void update(int val) {
        sum=val*(r-l+1);
        lazy=val;
    }
} T[MAXN<<2];
void up(int x) {
    T[x].sum=T[ls].sum+T[rs].sum;
}
void down(int x) {
    if (~T[x].lazy) {
        T[ls].update(T[x].lazy);
        T[rs].update(T[x].lazy);
        T[x].lazy=-1;
    }
}
void build(int x, int l, int r) {
    T[x]={l,r,0,-1};
    if (l<r) {
        build(ls,l,mid);
        build(rs,mid+1,r);
    }
```

```
}
void update(int x, int l, int r, int ql, int qr, int val) {
    if (ql<=l && r<=qr) T[x].update(val);
    else {
        down(x);
        if (ql<=mid) update(ls,l,mid,ql,qr,val);
        if (mid<qr) update(rs,mid+1,r,ql,qr,val);
        up(x);
    }
}
int query_sum(int x, int l, int r, int ql, int qr) {
    if (ql<=l && r<=qr) return T[x].sum;
    else {
        int ans=0;
        down(x);
        if (ql<=mid) ans+=query_sum(ls,l,mid,ql,qr);
        if (mid<qr) ans+=query_sum(rs,mid+1,r,ql,qr);
        up(x);
        return ans;
    }
}
struct edge {
    int to,nxt;
} e[MAXM];
int tot,f[MAXN];
void add(int u, int v) {
    e[tot]={v,f[u]}; f[u]=tot++;
}
int sz[MAXN],fa[MAXN],d[MAXN],son[MAXN],top[MAXN],clk,in[MAXN];
void dfs(int u) {
    sz[u]=1; d[u]=d[fa[u]]+1; son[u]=0;
    for (int i=f[u]; ~i; i=e[i].nxt) {
        int v=e[i].to;
        if (v!=fa[u]) {
            fa[v]=u; dfs(v);
            sz[u]+=sz[v];
            if (sz[v]>sz[son[u]]) son[u]=v;
        }
    }
}
void dfs(int u, int tp) {
    in[u]=++clk;
    top[u]=tp;
    if (son[u]) dfs(son[u],tp);
    for (int i=f[u]; ~i; i=e[i].nxt) {
        int v=e[i].to;
        if (v!=fa[u] && v!=son[u]) dfs(v,v);
    }
}
int install(int x) {
    int ans=0;
    while (x) {
        ans+=(in[x]-in[top[x]]+1)-query_sum(1,1,n,in[top[x]],in[x]);
        update(1,1,n,in[top[x]],in[x],1);
        x=fa[top[x]];
```

```
    }
    return ans;
}
void init() {
    tot=fa[1]=clk=0;
    memset(f,-1,sizeof(f));
}

int main()
{

    while (scanf("%d", &n)==1) {
        init();
        FOR(i,2,n) {
            int x; scanf("%d", &x);
            add(x+1,i);
        }
        dfs(1);
        dfs(1,1);
        build(1,1,n);
        int q; scanf("%d", &q);
        while (q--) {
            char op[11]; int x; scanf("%s%d", op,&x); x++;
            if (op[0]=='i') printf("%d\n", install(x));
            else {
                printf("%d\n", query_sum(1,1,n,in[x],in[x]+sz[x]-1));
                update(1,1,n,in[x],in[x]+sz[x]-1,0);
            }
        }
    }

    return 0;
}
```

K - Transformation HDU - 4578

涉及区间加、乘、定制修改，区间求和、平方和、立方和，虽是裸体但很有考验

与一般线段树无异，修改和查询都是先push_down，修改还要push_up，区别在于懒标记优先级：最高是set，其次add和mul相等，初始值时set=add=0,mul=1。更新set时，需将add,mul设为初始值；更新mul时，需将原mul和add乘以更新值；更新add时，只需简单加上add即可。区间求平方和、立方和，把多项式展开即可得到表达式。

线段树模板 1778ms

```
#define ls x<<1
#define rs x<<1|1
struct node {
    int l,r;
    LL sum[4],lazy[4];
    void update(int op, LL k)
    {
        LL tmp;
        switch (op) {
        case 1:
```

```
            lazy[1] = (lazy[1]+k)%MOD;
            sum[3] += 3*k*sum[2] + 3*k*k*sum[1] + k*k*k*(r-l+1);
            sum[2] += 2*k*sum[1] + k*k*(r-l+1);
            sum[1] += k*(r-l+1);
            FOR(i,1,3) sum[i] %= MOD;
            break;
        case 2:
            lazy[1] = lazy[1]*k%MOD;
            lazy[2] = lazy[2]*k%MOD;
            tmp = k;
            FOR(i,1,3) sum[i] = sum[i]*tmp%MOD, tmp *= k;
            break;
        case 3:
            lazy[1] = 0;
            lazy[2] = 1;
            lazy[3] = k;
            tmp = k;
            FOR(i,1,3) sum[i] = tmp*(r-l+1)%MOD, tmp *= k;
            break;
        }
    }
} tree[MAXN<<2];
void push_up(int x)
{
    FOR(i,1,3) tree[x].sum[i] = (tree[ls].sum[i] + tree[rs].sum[i]) % MOD;
}
void push_down(int x)
{
    ROF(i,1,3) {
        LL& t = tree[x].lazy[i];
        LL tag = i==2 ? 1 : 0;
        if (t != tag) {
            tree[ls].update(i,t);
            tree[rs].update(i,t);
            t = tag;
        }
    }
}
void build(int x, int l, int r)
{
    tree[x].l = l, tree[x].r = r;
    FOR(i,1,3) tree[x].sum[i] = tree[x].lazy[i] = 0;
    tree[x].lazy[2] = 1;
    if (l<r) {
        int m = (l+r)>>1;
        build(ls,l,m);
        build(rs,m+1,r);
    }
}
void update(int x, int op, int l, int r, int val)
{
    int L = tree[x].l, R = tree[x].r;
    if (l <= L && R <= r) {
        tree[x].update(op,val);
    } else {
```

```
        push_down(x);
        int m = (L+R)>>1;
        if (l<=m) update(ls,op,l,r,val);
        if (m<r) update(rs,op,l,r,val);
        push_up(x);
    }
}
LL query(int x, int l, int r, int p)
{
    int L = tree[x].l, R = tree[x].r;
    if (l <= L && R <= r) {
        return tree[x].sum[p];
    } else {
        push_down(x);
        LL ans = 0;
        int m = (L+R)>>1;
        if (l<=m) ans += query(ls,l,r,p);
        if (m<r) ans += query(rs,l,r,p);
        return ans % MOD;
    }
}

int main()
{

    int n, m, op, l, r, val;
    while (scanf("%d%d", &n, &m) == 2 && n && m) {
        build(1,1,n);
        while (m--) {
            scanf("%d%d%d%d", &op,&l,&r,&val);
            if (op != 4) update(1,op,l,r,val);
            else printf("%I64d\n", query(1,l,r,val));
        }
    }

    return 0;
}
```

P - Atlantis HDU - 1542

扫描线热身

线段树、面积并、离散化

https://blog.csdn.net/wlxsq/article/details/47254571

维护一个覆盖次数cover，一个当前区间被覆盖的长度len。扫描线算法步骤：先把纵坐标离散化。从左到右扫描x坐标，每扫到矩形的左边的“竖边”加到线段树中，扫到右边的“竖边”即从线段树中删掉，每次把当前的得到的纵坐标覆盖长度乘上相邻两点横坐标之差。

离散化采用vector，下标从0开始，所以采用左开右闭区间（因为线段树下标从1开始）表示线段，即线段树上一个下标i，表示区间(Y[i-1],Y[i]]这条线段。

正常整数区间离散都是二元组(x1,x2-1)表示一个区间，上述表示方法则不再局限整数区间，实数区间也可以离散化表示。

不是很明白无需push_down操作，可能每次区间修改必对应两次一正一反的操作？或者查询只查询树根？如果一般线段树写法维护cover,

面积并模板

```
struct Line {
    double x,y1,y2;
    int sign;
    bool operator<(const Line& o) const { return x < o.x; }
} line[MAXN];
vector<double> Y;
int ID(double y) { return lower_bound(ALL(Y),y)-Y.begin(); }

#define ls x<<1
#define rs x<<1|1
struct node {
    int l,r,cover;
    double len;
} tree[MAXN<<2];
void push_up(int x)
{
    int L = tree[x].l, R = tree[x].r;
    if (tree[x].cover) tree[x].len = Y[R]-Y[L-1];
    else if (L == R) tree[x].len = 0;
    else tree[x].len = tree[ls].len + tree[rs].len;
}
void build(int x, int l, int r)
{
    tree[x].l = l, tree[x].r = r;
    tree[x].cover = tree[x].len = 0;
    if (l<r) {
        int m = (l+r)>>1;
        build(ls,l,m);
        build(rs,m+1,r);
    }
}
void update(int x, int l, int r, int val)
{
    int L = tree[x].l, R = tree[x].r;
    if (l <= L && R <= r) {
        tree[x].cover += val;
    } else {
        int m = (L+R)>>1;
        if (l<=m) update(ls,l,r,val);
        if (m<r) update(rs,l,r,val);
    }
    push_up(x);
}

int main()
{

    int n, kase = 0;
    double x1,x2,y1,y2;
    while (~scanf("%d", &n) && n) {
        Y.clear();
        REP(i,n) {
```

```
            scanf("%lf%lf%lf%lf", &x1,&y1,&x2,&y2);
            line[i<<1] = (Line){x1,y1,y2,1};
            line[i<<1|1] = (Line){x2,y1,y2,-1};
            Y.pb(y1), Y.pb(y2);
        }
        n <<= 1;
        sort(line,line+n);
        sort(ALL(Y)); Y.erase(unique(ALL(Y)),Y.end());
        build(1,1,SZ(Y));
        double ans = 0;
        REP(i,n-1) {
            int l = ID(line[i].y1)+1, r = ID(line[i].y2);
            update(1,l,r,line[i].sign);
            ans += tree[1].len*(line[i+1].x-line[i].x);
        }
        printf("Test case #%d\nTotal explored area: %.2f\n\n", ++kase, ans);
    }

    return 0;
}
```

O - 覆盖的面积 HDU - 1255

线段树、面积交、离散化

https://www.cnblogs.com/lxjshuju/p/7040186.html

https://blog.csdn.net/zearot/article/details/47762543

与面积并的区别在于push_up， 线段树多维护里一个len表示覆盖两次以上的长度，而push_up的更新有所变化

```
struct Line {
    double x,y1,y2;
    int sign;
    bool operator<(const Line& o) const { return x<o.x; }
} line[MAXN];
vector<double> Y;
int ID(double y) { return lower_bound(ALL(Y),y)-Y.begin(); }

#define ls x<<1
#define rs x<<1|1
struct node {
    int l,r,cover;
    double len[3];
} tree[MAXN<<2];
void push_up(int x)
{
    FOR(i,1,2) {
        int j = i-tree[x].cover;
        if (j <= 0) tree[x].len[i] = tree[x].len[0];
        else if (tree[x].l == tree[x].r) tree[x].len[i] = 0;
        else tree[x].len[i] = tree[ls].len[j]+tree[rs].len[j];
    }
}
void build(int x, int l, int r)
```

```
{
    tree[x] = (node){l,r,0,Y[r]-Y[l-1],0,0};
    if (l<r) {
        int m = (l+r)>>1;
        build(ls,l,m);
        build(rs,m+1,r);
    }
}
void update(int x, int l, int r, int val)
{
    int L = tree[x].l, R = tree[x].r;
    if (l <= L && R <= r) {
        tree[x].cover += val;
    } else {
        int m = (L+R)>>1;
        if (l<=m) update(ls,l,r,val);
        if (m<r) update(rs,l,r,val);
    }
    push_up(x);
}

int main()
{

    double x1,x2,y1,y2;
    int T; scanf("%d", &T);
    while (T--) {
        int n; scanf("%d", &n);
        Y.clear();
        REP(i,n) {
            scanf("%lf%lf%lf%lf", &x1,&y1,&x2,&y2);
            line[i<<1] = (Line){x1,y1,y2,1};
            line[i<<1|1] = (Line){x2,y1,y2,-1};
            Y.pb(y1), Y.pb(y2);
        }
        n <<= 1;
        sort(line,line+n);
        sort(ALL(Y)); Y.erase(unique(ALL(Y)),Y.end());
        double ans = 0;
        build(1,1,SZ(Y)-1);
        REP(i,n-1) {
            int l = ID(line[i].y1)+1, r = ID(line[i].y2);
            update(1,l,r,line[i].sign);
            ans += tree[1].len[2]*(line[i+1].x-line[i].x);
        }
        printf("%.2f\n", ans);
    }

    return 0;
}
```

N - Picture POJ - 1177

求覆盖面积周长

https://www.cnblogs.com/AC-King/p/7789013.html

相比求面积交，多维护：lines表示区间有多少个线段，每个线段贡献两个端点，lf、rf表示区间端点是否被覆盖，这样写push_up区间合并时：（详见代码。。）

矩形并周长模板（覆盖面周长模板）

```
struct Line {
    int x,y1,y2,sign;
    bool operator<(const Line& o) const { return x<o.x; }
} line[MAXN];
vector<int> Y;
int ID(int y) { return lower_bound(ALL(Y),y)-Y.begin(); }

#define ls x<<1
#define rs x<<1|1
struct node {
    int l,r,cover,lines,len[2],lf,rf;
} tree[MAXN<<2];
void push_up(int x)
{
    if (tree[x].cover) {
        tree[x].len[1] = tree[x].len[0];
        tree[x].lf = tree[x].rf = tree[x].lines = 1;
    } else {
        tree[x].lines = tree[ls].lines + tree[rs].lines - (tree[ls].rf && tree[rs].lf);
        tree[x].len[1] = tree[ls].len[1] + tree[rs].len[1];
        tree[x].lf = tree[ls].lf, tree[x].rf = tree[rs].rf;
    }
}
void build(int x, int l, int r)
{
    tree[x] = (node){l,r,0,0,Y[r]-Y[l-1],0,0,0};
    if (l<r) {
        int m = (l+r)>>1;
        build(ls,l,m);
        build(rs,m+1,r);
    }
}
void update(int x, int l, int r, int val)
{
    int L = tree[x].l, R = tree[x].r;
    if (l <= L && R <= r) {
        tree[x].cover += val;
    } else {
        int m = (L+R)>>1;
        if (l<=m) update(ls,l,r,val);
        if (m<r) update(rs,l,r,val);
    }
    push_up(x);
}

int main()
{

    int n,x1,x2,y1,y2;
```

```
    while (~scanf("%d", &n)) {
        Y.clear();
        REP(i,n) {
            scanf("%d%d%d%d", &x1,&y1,&x2,&y2);
            line[i<<1] = (Line){x1,y1,y2,1};
            line[i<<1|1] = (Line){x2,y1,y2,-1};
            Y.pb(y1), Y.pb(y2);
        }
        n <<= 1;
        sort(line, line+n);
        sort(ALL(Y)); Y.erase(unique(ALL(Y)),Y.end());
        LL ans = 0;
        build(1,1,SZ(Y)-1);
        REP(i,n-1) {
            int l = ID(line[i].y1)+1, r = ID(line[i].y2);
            int preL = tree[1].len[1];
            update(1,l,r,line[i].sign);
            ans += abs(tree[1].len[1]-preL) + tree[1].lines*2*(line[i+1].x-line[i].x);
        }
        ans += tree[1].len[1];
        printf("%I64d\n", ans);
    }

    return 0;
}
```

## 4.5 树链剖分

Query on a tree SPOJ - QTREE

树链剖分边权单点修改区间最值

给的是边权，可以通过下放边权到深度较深的那个结点解决，树剖查询的时候，如果top结点相同，查询时不查祖先那个点，如果只有祖先一个点，直接返回

要保存每条边的两个结点，然后通过深度判断第i条边的边权是下放到哪个结点去了

```
int n,a[MAXN],val[MAXN];
#define ls x<<1
#define rs x<<1|1
#define mid (l+r)/2
struct node {
    int ma;
} T[MAXN<<2];
void up(int x) {
    T[x].ma=max(T[ls].ma,T[rs].ma);
}
void build(int x, int l, int r) {
    T[x]={a[l]};
    if (l<r) {
        build(ls,l,mid);
        build(rs,mid+1,r);
```

```
        up(x);
    }
}
void update(int x, int l, int r, int p, int val) {
    if (l==r) T[x]={val};
    else {
        if (p<=mid) update(ls,l,mid,p,val);
        else update(rs,mid+1,r,p,val);
        up(x);
    }
}
int query_max(int x, int l, int r, int ql, int qr) {
    if (ql<=l && r<=qr) return T[x].ma;
    else {
        int ans=-INF;
        if (ql<=mid) ans=max(ans,query_max(ls,l,mid,ql,qr));
        if (mid<qr) ans=max(ans,query_max(rs,mid+1,r,ql,qr));
        return ans;
    }
}
struct edge {
    int to,nxt,w;
} e[MAXM];
int tot,f[MAXN],uu[MAXN],vv[MAXN];
void add(int u, int v, int w) {
    e[tot]={v,f[u],w}; f[u]=tot++;
}
int sz[MAXN],fa[MAXN],d[MAXN],son[MAXN],top[MAXN],clk,in[MAXN];
void dfs(int u) {
    sz[u]=1; d[u]=d[fa[u]]+1; son[u]=0;
    for (int i=f[u]; ~i; i=e[i].nxt) {
        int v=e[i].to;
        if (v!=fa[u]) {
            fa[v]=u; val[v]=e[i].w; dfs(v);
            sz[u]+=sz[v];
            if (sz[v]>sz[son[u]]) son[u]=v;
        }
    }
}
void dfs(int u, int tp) {
    a[++clk]=val[u]; in[u]=clk;
    top[u]=tp;
    if (son[u]) dfs(son[u],tp);
    for (int i=f[u]; ~i; i=e[i].nxt) {
        int v=e[i].to;
        if (v!=fa[u] && v!=son[u]) dfs(v,v);
    }
}
int tree_max(int x, int y) {
    int ans=-INF;
    while (top[x]!=top[y]) {
        if (d[top[x]]<d[top[y]]) swap(x,y);
        ans=max(ans,query_max(1,1,n,in[top[x]],in[x]));
        x=fa[top[x]];
    }
```

```
    if (x==y) return ans;
    if (in[x]>in[y]) swap(x,y);
    ans=max(ans,query_max(1,1,n,in[x]+1,in[y]));
    return ans;
}
void init() {
    tot=fa[1]=clk=0;
    memset(f,-1,sizeof(f));
    val[1]=-INF;
}

int main()
{

    int kase; scanf("%d", &kase);
    while (kase--) {
        init();
        scanf("%d", &n);
        FOR(i,1,n-1) {
            int z;
            scanf("%d%d%d", &uu[i],&vv[i],&z);
            add(uu[i],vv[i],z);
            add(vv[i],uu[i],z);
        }
        dfs(1);
        dfs(1,1);
        build(1,1,n);
        FOR(i,1,n-1) {
            if (d[uu[i]]>d[vv[i]]) swap(uu[i],vv[i]);
        }
        char op[11];
        while (scanf("%s", op), op[0]!='D') {
            int x,y; scanf("%d%d", &x,&y);
            if (op[0]=='C') update(1,1,n,in[vv[x]],y);
            else printf("%d\n", tree_max(x,y));
        }
    }

    return 0;
}
```

P3379 【模板】最近公共祖先（LCA）

```
int to[MAXM],nxt[MAXM],f[MAXN],tot;
void init() {
    tot=0;
    memset(f,-1,sizeof(f));
}
void add(int u, int v) {
    to[tot]=v;
    nxt[tot]=f[u];
    f[u]=tot++;
}
```

```
int n,q,root;
int sz[MAXN],d[MAXN],fa[MAXN],son[MAXN],top[MAXN];
void init1() {
    d[0]=0;
    fa[root]=0; //start at 0
    memset(son,0,sizeof(son));
}
void dfs(int u) {
    sz[u]=1; d[u]=d[fa[u]]+1;
    for (int i=f[u]; ~i; i=nxt[i]) {
        int v=to[i];
        if (v==fa[u]) continue;
        fa[v]=u; dfs(v);
        sz[u]+=sz[v];
        if (!son[u]||sz[v]>sz[son[u]]) son[u]=v;
    }
}
void dfs(int u, int tp) {
    top[u]=tp;
    if (son[u]) dfs(son[u],tp);
    for (int i=f[u]; ~i; i=nxt[i]) {
        int v=to[i];
        if (v==son[u]||v==fa[u]) continue;
        dfs(v,v);
    }
}
int LCA(int x, int y) {
    while (top[x]!=top[y]) d[top[x]]>=d[top[y]]?x=fa[top[x]]:y=fa[top[y]];
    return d[x]>=d[y]?y:x;
}

int main()
{

    while (scanf("%d%d%d", &n,&q,&root)==3) {
        init();
        REP(i,n-1) {
            int x,y; scanf("%d%d", &x,&y);
            add(x,y); add(y,x);
        }
        init1();
        dfs(root);
        dfs(root,root);
        while (q--) {
            int x,y; scanf("%d%d", &x,&y);
            printf("%d\n", LCA(x,y));
        }
    }

    return 0;
}
```

P3384 【模板】树链剖分

支持以下操作：

操作1：格式： 1 x y z 表示将树从x到y结点最短路径上所有节点的值都加上z
操作2：格式： 2 x y 表示求树从x到y结点最短路径上所有节点的值之和
操作3：格式： 3 x z 表示将以x为根节点的子树内所有节点值都加上z
操作4：格式： 4 x 表示求以x为根节点的子树内所有节点值之和
树链剖分可以把树上两个结点之间的路径转化为O(logn)个区间，方法是：第二遍dfs的时候维护dfs序，

```
int n,q,root,p;
LL a[MAXN],val[MAXN];
#define ls x<<1
#define rs x<<1|1
#define mid (l+r)/2
struct node {
    int l,r;
    LL sum,add;
    void update(LL x) {
        sum+=x*(r-l+1); sum%=p;
        add+=x; add%=p;
    }
} T[MAXN<<2];
void up(int x) {
    T[x].sum=(T[ls].sum+T[rs].sum)%p;
}
void down(int x) {
    if (T[x].add) {
        T[ls].update(T[x].add);
        T[rs].update(T[x].add);
        T[x].add=0;
    }
}
void build(int x, int l, int r) {
    T[x]={l,r,0,0};
    if (l==r) T[x].sum=a[l];
    else {
        build(ls,l,mid);
        build(rs,mid+1,r);
        up(x);
    }
}
void update(int x, int l, int r, int ql, int qr, LL val) {
    if (ql<=l && r<=qr) T[x].update(val);
    else {
        down(x);
        if (ql<=mid) update(ls,l,mid,ql,qr,val);
        if (mid<qr) update(rs,mid+1,r,ql,qr,val);
        up(x);
    }
}
LL query(int x, int l, int r, int ql, int qr) {
    if (ql<=l && r<=qr) return T[x].sum;
    else {
        LL ans=0;
```

```
        down(x);
        if (ql<=mid) ans+=query(ls,l,mid,ql,qr);
        if (mid<qr) ans+=query(rs,mid+1,r,ql,qr);
        up(x);
        return ans%p;
    }
}
struct edge {
    int to,nxt;
} e[MAXN<<1];
int tot,f[MAXN];
void add(int u, int v) {
    e[tot]={v,f[u]}; f[u]=tot++;
}
int sz[MAXN],d[MAXN],fa[MAXN],son[MAXN],top[MAXN],clk,in[MAXN];
void dfs(int u) {
    sz[u]=1; d[u]=d[fa[u]]+1; son[u]=0;
    for (int i=f[u]; ~i; i=e[i].nxt) {
        int v=e[i].to;
        if (v!=fa[u]) {
            fa[v]=u; dfs(v);
            sz[u]+=sz[v];
            if (sz[v]>sz[son[u]]) son[u]=v;
        }
    }
}
void dfs(int u, int tp) {
    a[++clk]=val[u]; in[u]=clk;
    top[u]=tp;
    if (son[u]) dfs(son[u],tp);
    for (int i=f[u]; ~i; i=e[i].nxt) {
        int v=e[i].to;
        if (v!=fa[u] && v!=son[u]) dfs(v,v);
    }
}
void tree_add(int x, int y, LL val) {
    while (top[x]!=top[y]) {
        if (d[top[x]]<d[top[y]]) swap(x,y);
        update(1,1,n,in[top[x]],in[x],val);
        x=fa[top[x]];
    }
    if (in[x]>in[y]) swap(x,y);
    update(1,1,n,in[x],in[y],val);
}
LL tree_sum(int x, int y) {
    LL ans=0;
    while (top[x]!=top[y]) {
        if (d[top[x]]<d[top[y]]) swap(x,y);
        ans+=query(1,1,n,in[top[x]],in[x]);
        x=fa[top[x]];
    }
    if (in[x]>in[y]) swap(x,y);
    ans+=query(1,1,n,in[x],in[y]);
    return ans%p;
}
```

```
void init() {
    tot=clk=fa[root]=0;
    memset(f,-1,sizeof(f));
}

int main()
{

    while (scanf("%d%d%d%d", &n,&q,&root,&p)==4) {
        init();
        FOR(i,1,n) scanf("%lld", &val[i]), val[i]%=p;
        REP(i,n-1) {
            int x,y; scanf("%d%d", &x,&y);
            add(x,y); add(y,x);
        }
        dfs(root);
        dfs(root,root);
        build(1,1,n);
        while (q--) {
            int op,x,y; LL z; scanf("%d", &op);
            switch (op) {
                case 1:
                    scanf("%d%d%lld", &x,&y,&z);
                    tree_add(x,y,z%p);
                    break;
                case 2:
                    scanf("%d%d", &x,&y);
                    printf("%lld\n", tree_sum(x,y));
                    break;
                case 3:
                    scanf("%d%lld", &x,&z);
                    update(1,1,n,in[x],in[x]+sz[x]-1,z%p);
                    break;
                case 4:
                    scanf("%d", &x);
                    printf("%lld\n", query(1,1,n,in[x],in[x]+sz[x]-1));
                    break;
            }
        }
    }

    return 0;
}
```

P2486 [SDOI2011]染色

两种操作：1.将两结点之间路径染色。2.询问两结点之间路径连续颜色段数。

首先考虑区间染色问题，需要支持区间修改，即要下推标记；需要区间合并，则线段树需要维护左端点颜色和右端点颜色以及连续颜色段数，合并时需要检查两区间中间颜色是否相同。

其次考虑树上的区间合并，树链剖分思想是两个结点往上跳，所以开两个维护答案，最后合并答案，需要考虑区间的反转，具体可以手推一下就出来了。

```
int n,q,a[MAXN],val[MAXN];
#define ls x<<1
#define rs x<<1|1
#define mid (l+r)/2
struct node {
    int lc,rc,cnt,lazy;
    void update(int val) {
        lc=rc=lazy=val;
        cnt=1;
    }
} T[MAXN<<2];
node cal(node a, node b) {
    return {a.lc,b.rc,a.cnt+b.cnt-(a.rc==b.lc),0};
}
void up(int x) {
    T[x]=cal(T[ls],T[rs]);
}
void down(int x) {
    if (T[x].lazy) {
        T[ls].update(T[x].lazy);
        T[rs].update(T[x].lazy);
        T[x].lazy=0;
    }
}
void build(int x, int l, int r) {
    if (l==r) T[x]={a[l],a[l],1,0};
    else {
        build(ls,l,mid);
        build(rs,mid+1,r);
        up(x);
    }
}
void update(int x, int l, int r, int ql, int qr, int val) {
    if (ql<=l && r<=qr) T[x].update(val);
    else {
        down(x);
        if (ql<=mid) update(ls,l,mid,ql,qr,val);
        if (mid<qr) update(rs,mid+1,r,ql,qr,val);
        up(x);
    }
}
node query(int x, int l, int r, int ql, int qr) {
    if (ql<=l && r<=qr) return T[x];
    else {
        node ans;
        down(x);
        if (ql<=mid && mid<qr) {
            node a=query(ls,l,mid,ql,qr);
            node b=query(rs,mid+1,r,ql,qr);
            ans=cal(a,b);
        } else {
            if (ql<=mid) ans=query(ls,l,mid,ql,qr);
            else ans=query(rs,mid+1,r,ql,qr);
        }
        up(x);
```

```
        return ans;
    }
}
struct edge {
    int to,nxt;
} e[MAXM];
int f[MAXN],tot;
void add(int u, int v) {
    e[tot]={v,f[u]}; f[u]=tot++;
}
int sz[MAXN],fa[MAXN],d[MAXN],son[MAXN],top[MAXN],clk,in[MAXN];
void dfs(int u) {
    sz[u]=1; d[u]=d[fa[u]]+1; son[u]=0;
    for (int i=f[u]; ~i; i=e[i].nxt) {
        int v=e[i].to;
        if (v!=fa[u]) {
            fa[v]=u; dfs(v);
            sz[u]+=sz[v];
            if (sz[v]>sz[son[u]]) son[u]=v;
        }
    }
}
void dfs(int u, int tp) {
    a[++clk]=val[u]; in[u]=clk;
    top[u]=tp;
    if (son[u]) dfs(son[u],tp);
    for (int i=f[u]; ~i; i=e[i].nxt) {
        int v=e[i].to;
        if (v!=fa[u] && v!=son[u]) dfs(v,v);
    }
}
void tree_update(int x, int y, int val) {
    while (top[x]!=top[y]) {
        if (d[top[x]]<d[top[y]]) swap(x,y);
        update(1,1,n,in[top[x]],in[x],val);
        x=fa[top[x]];
    }
    if (in[x]>in[y]) swap(x,y);
    update(1,1,n,in[x],in[y],val);
}
int tree_query(int x, int y) {
    node a{0,0,0}, b{0,0,0};
    while (top[x]!=top[y]) {
        if (d[top[x]]<d[top[y]]) swap(x,y), swap(a,b);
        if (a.cnt) a=cal(query(1,1,n,in[top[x]],in[x]),a);
        else a=query(1,1,n,in[top[x]],in[x]);
        x=fa[top[x]];
    }
    if (in[x]>in[y]) swap(x,y), swap(a,b);
    if (b.cnt) b=cal(query(1,1,n,in[x],in[y]),b);
    else b=query(1,1,n,in[x],in[y]);
    swap(a.lc,a.rc);
    return cal(a,b).cnt;
}
void init() {
```

```
    tot=fa[1]=clk=0;
    memset(f,-1,sizeof(f));
}

int main()
{

    while (scanf("%d%d", &n,&q)==2) {
        init();
        FOR(i,1,n) scanf("%d", &val[i]);
        REP(i,n-1) {
            int x,y; scanf("%d%d", &x,&y);
            add(x,y); add(y,x);
        }
        dfs(1);
        dfs(1,1);
        build(1,1,n);
        while (q--) {
            char op[2]; int x,y,z; scanf("%s%d%d", op,&x,&y);
            if (op[0]=='Q') printf("%d\n", tree_query(x,y));
            else scanf("%d", &z), tree_update(x,y,z);
        }
    }

    return 0;
}
```

## 4.6 平衡树fhq_treap

区间加法

```
struct fhq_treap {
    #define lt T[x].ls
    #define rt T[x].rs
    int root,tot,tmp;
    struct node {
        int rnd,sz,ls,rs;
        LL val,sum,add;
    } T[MAXN];
    void build(int& x, LL val){ T[x=++tot]={rand(),1,0,0,val,val}; }
    void addone(int x, LL val) {
        if (x) {
            T[x].val+=val; T[x].add+=val;
            T[x].sum+=T[x].sz*val;
        }
    }
    void init() {
        srand(19981111);
        root=tot=0;
        T[0]={INF};
```

```
    }
    void up(int x) {
        if (x) {
            T[x].sz=T[lt].sz+T[rt].sz+1;
            T[x].sum=T[lt].sum+T[rt].sum+T[x].val;
        }
    }
    void down(int x) {
        if (x) {
            if (T[x].add) addone(lt,T[x].add), addone(rt,T[x].add);
            T[x].add=0;
        }
    }
    void Merge(int& x, int l, int r) {
        if (!l || !r) x=l+r;
        else if (T[l].rnd<T[r].rnd) down(x=l), Merge(rt,rt,r), up(x);
        else down(x=r), Merge(lt,l,lt), up(x);
    }
    void build(int n) {
        REP(i,n) {
            LL x; scanf("%lld", &x);
            build(tmp,x); Merge(root,root,tmp);
        }
    }
    void split(int x, int& l, int& r, int k) {
        if (!x) { l=r=0; return; }
        if (k<=T[lt].sz) down(r=x), split(lt,l,lt,k), up(x);
        else down(l=x), split(rt,rt,r,k-T[lt].sz-1), up(x);
    }
    void add(int l, int r, LL val) {
        int x,y,z;
        split(root,x,y,r); split(x,x,z,l-1);
        addone(z,val);
        Merge(x,x,z); Merge(root,x,y);
    }
    LL query(int l, int r) {
        int x,y,z;
        split(root,x,y,r); split(x,x,z,l-1);
        LL ans=T[z].sum;
        Merge(x,x,z); Merge(root,x,y);
        return ans;
    }
} solve;
```

Tyvj 1728 普通平衡树 BZOJ- 3224 588ms

经典：

1. 插入x数
2. 删除x数(若有多个相同的数，因只删除一个)
3. 查询x数的排名(若有多个相同的数，因输出最小的排名)
4. 查询排名为x的数
5. 求x的前驱(前驱定义为小于x，且最大的数)
6. 求x的后继(后继定义为大于x，且最小的数)

```
struct fhq_treap {
    #define lt T[x].ls
    #define rt T[x].rs
    int root,tot,tmp;
    struct node {
        int rnd,sum,sz,ls,rs;
    } T[MAXN];
    void init()
    {
        srand(19981111);
        root = tot = 0;
        T[0].rnd = T[0].sum = INF;
    }
    void build(int& x, int val) { T[x=++tot] = node{rand()<<15|rand(),val,1}; }
    void up(int x) { if (x) T[x].sz = T[lt].sz+T[rt].sz+1; }
    void Merge(int& x, int l, int r)
    {
        if (!l || !r) x = l+r;
        else if (T[l].rnd<T[r].rnd) x=l, Merge(rt,rt,r), up(x);
        else x=r, Merge(lt,l,lt), up(x);
    }
    void split(int x, int& l, int& r, int k)
    {
        if (!k) l=0, r=x;
        else if (k==T[x].sz) l=x, r=0;
        else if (k<=T[lt].sz) r=x, split(lt,l,lt,k), up(x);
        else l=x, split(rt,rt,r,k-T[lt].sz-1), up(x);
    }
    int Rank(int x, int val)
    {
        if (!x) return 0;
        if (T[x].sum>=val) return Rank(lt,val);
        return T[lt].sz+1+Rank(rt,val);
    }
    void Insert(int val)
    {
        int x,y,rk = Rank(root,val);
        split(root,x,y,rk);
        build(tmp,val);
        Merge(x,x,tmp); Merge(root,x,y);
    }
    void del(int val)
    {
        int x,y,z,rk = Rank(root,val);
        split(root,x,y,rk+1); split(x,x,z,rk);
        Merge(root,x,y);
    }
    int Kth(int k)
    {
        int x,y,z;
        split(root,x,y,k); split(x,x,z,k-1);
        int ans = T[z].sum;
        Merge(x,x,z); Merge(root,x,y);
        return ans;
```

```
    }
    int pre(int val)
    {
        int x,y,z,rk = Rank(root,val);
        split(root,x,y,rk); split(x,x,z,rk-1);
        int ans = T[z].sum;
        Merge(x,x,z); Merge(root,x,y);
        return ans;
    }
    int succ(int val)
    {
        int x,y,z,rk = Rank(root,val+1);
        split(root,x,y,rk+1); split(x,x,z,rk);
        int ans = T[z].sum;
        Merge(x,x,z); Merge(root,x,y);
        return ans;
    }
};

fhq_treap solve;
int n,op,x;

int main()
{

    scanf("%d", &n);
    solve.init();
    while (n--) {
        scanf("%d%d", &op,&x);
        switch (op) {
        case 1: solve.Insert(x);
            break;
        case 2: solve.del(x);
            break;
        case 3: printf("%d\n", solve.Rank(solve.root,x)+1);
            break;
        case 4: printf("%d\n", solve.Kth(x));
            break;
        case 5: printf("%d\n", solve.pre(x));
            break;
        case 6: printf("%d\n", solve.succ(x));
            break;
        }
    }

    return 0;
}
```

SuperMemo POJ - 3580
区间加、翻转、旋转，插入、删除，求最值
加了push_up、push_down
模板 891ms

```
struct fhq_treap {
    #define lt T[x].ls
    #define rt T[x].rs
    int root,tot,tmp;
    struct node {
        int rnd,sum,mi,sz,add,rev,ls,rs;
    } T[MAXN];
    void build(int& x, int val) { T[x=++tot] = node{rand()<<15|rand(),val,val,1}; }
    void addone(int x, int val) { if (x) T[x].sum+=val, T[x].mi+=val, T[x].add+=val; }
    void revone(int x) { T[x].rev ^= 1, swap(lt,rt); }
    void init()
    {
        srand(19981111);
        root = tot = 0;
        T[0] = node{INF,INF,INF};
    }
    void up(int x)
    {
        if (x) {
            T[x].mi = min(T[x].sum, min(T[lt].mi, T[rt].mi));
            T[x].sz = T[lt].sz + T[rt].sz + 1;
        }
    }
    void down(int x)
    {
        if (x) {
            if (T[x].add) addone(lt,T[x].add), addone(rt,T[x].add);
            if (T[x].rev) revone(lt), revone(rt);
            T[x].add = T[x].rev = 0;
        }
    }
    void Merge(int& x, int l, int r)
    {
        if (!l || !r) x=l+r;
        else if (T[l].rnd<T[r].rnd) down(x=l), Merge(rt,rt,r), up(x);
        else down(x=r), Merge(lt,l,lt), up(x);
    }
    void build(int n)
    {
        while (n--) {
            int x; scanf("%d", &x);
            build(tmp,x); Merge(root,root,tmp);
        }
    }
    void split(int x, int& l, int& r, int k)
    {
        if (!k) l=0, r=x;
        else if (k==T[x].sz) l=x, r=0;
        else if (k<=T[lt].sz) down(r=x), split(lt,l,lt,k), up(x);
        else down(l=x), split(rt,rt,r,k-T[lt].sz-1), up(x);
    }
    void Insert(int pos, int val)
    {
        int x,y;
```

```
        split(root,x,y,pos);
        build(tmp,val);
        Merge(x,x,tmp), Merge(root,x,y);
    }
    void del(int pos)
    {
        int x,y,z;
        split(root,x,z,pos), split(x,x,y,pos-1);
        Merge(root,x,z);
    }
    void add(int l, int r, int val)
    {
        int x,y,z;
        split(root,x,z,r), split(x,x,y,l-1);
        addone(y,val);
        Merge(x,x,y), Merge(root,x,z);
    }
    void Reverse(int l, int r)
    {
        int x,y,z;
        split(root,x,z,r), split(x,x,y,l-1);
        revone(y);
        Merge(x,x,y), Merge(root,x,z);
    }
    void revole(int l, int r, int k)
    {
        k %= r-l+1;
        int x,y,z,h;
        split(root,x,h,r), split(x,x,z,r-k), split(x,x,y,l-1);
        Merge(x,x,z), Merge(x,x,y), Merge(root,x,h);
    }
    int query_mi(int l, int r)
    {
        int x,y,z;
        split(root,x,z,r), split(x,x,y,l-1);
        int ans = T[y].mi;
        Merge(x,x,y), Merge(root,x,z);
        return ans;
    }
};

fhq_treap solve;
int n,m,x,y,z;
char op[10];

int main()
{

    scanf("%d", &n);
    solve.init();
    solve.build(n);
    scanf("%d", &m);
    while (m--) {
        scanf("%s", op);
        switch (op[0]) {
```

```
        case 'A': scanf("%d%d%d", &x,&y,&z), solve.add(x,y,z);
            break;
        case 'R': scanf("%d%d", &x,&y);
            if (op[3] == 'E') solve.Reverse(x,y);
            else scanf("%d", &z), solve.revole(x,y,z);
            break;
        case 'I': scanf("%d%d", &x,&y), solve.Insert(x,y);
            break;
        case 'D': scanf("%d", &x), solve.del(x);
            break;
        case 'M': scanf("%d%d", &x,&y), printf("%d\n", solve.query_mi(x,y));
            break;
        }
    }

    return 0;
}
```

递归打印全体（中序遍历）

```
void print(int x)
{
    if (x) {
        down(x);
        print(lt);
        printf("%d ", T[x].sum);
        print(rt);
    }
}
```

treap线性构树的方法：跟笛卡尔树一样

https://blog.sengxian.com/algorithms/treap

Version Controlled IDE UVA - 12538

线性建树、pos插入k个、pos删除k个、pos后打印k个、强制在线

线性建树与笛卡尔树建树一样，用一个栈保存最右边一条链。可持久化：fhq_treap由于不修改结点，故每次操作都新建一个结点，并把每次操作的根保存，根的编号即是版本号。此处还需要用到内存池保存结点——重载new运算符。

一直RE，把rand()¡¡15—rand()改成rand()就好了!!!???

100ms，优秀

```
int tot,cntC;
struct node *null,*pit;
struct node {
    int key,rnd,sz;
    node *l,*r;
    node(){}
    node(int key, int rnd = rand()):key(key),rnd(rnd),sz(1){ l=r=null; }
    void* operator new(size_t){ return pit++; }
    void upd(){ sz = l->sz+r->sz+1; }
} pool[4000010], *root[MAXN];
```

```
node* newnode(node* o) { return o==null ? o : new node{*o}; }
node* Merge(node* a, node* b)
{
    if (a==null) return newnode(b);
    if (b==null) return newnode(a);
    node* tmp;
    if (a->rnd < b->rnd) {
        tmp = newnode(a);
        tmp->r = Merge(a->r,b);
    } else {
        tmp = newnode(b);
        tmp->l = Merge(a,b->l);
    }
    tmp->upd();
    return tmp;
}
typedef pair<node*, node*> droot;
droot split(node* o, int k)
{
    droot d(null,null);
    if (o==null) return d;
    if (!k) return droot(null,newnode(o));
    if (k==o->sz) return droot(newnode(o),null);
    int sz = o->l->sz;
    node* tmp = newnode(o);
    if (k<=sz) {
        d = split(o->l,k);
        tmp->l = d.se;
        tmp->upd();
        d.se = tmp;
    } else {
        d = split(o->r,k-sz-1);
        tmp->r = d.fi;
        tmp->upd();
        d.fi = tmp;
    }
    return d;
}
node* stk[MAXN];
node* build(char* p)
{
    node* rt = new node(-INF,-INF);
    stk[0] = rt; int sz = 1;
    for (; *p; p++) {
        node* now = new node(*p); int top = sz-1;
        while (stk[top]->rnd > now->rnd) stk[top--]->upd();
        if (top != sz-1) now->l = stk[top+1];
        stk[top]->r = now; sz = top+1;
        stk[sz++] = now;
    }
    while (sz) stk[--sz]->upd();
    return rt->r;
}
void print(node* o)
{
```

```
    if (o==null) return;
    print(o->l);
    putchar(o->key);
    if (o->key == 'c') cntC++;
    print(o->r);
}
void init()
{
    srand(time(NULL));
    pit = pool; null = new node();
    null->sz = tot = cntC = 0;
    root[0] = null;
}
void Insert(int pos, char* s)
{
    droot l = split(root[tot],pos);
    root[++tot] = Merge(Merge(l.fi,build(s)),l.se);
}
void Remove(int pos, int k)
{
    droot l = split(root[tot],pos-1);
    droot r = split(l.se,k);
    root[++tot] = Merge(l.fi,r.se);
}
void print(int v, int pos, int k)
{
    droot l = split(root[v],pos-1);
    droot r = split(l.se,k);
    print(r.fi); puts("");
}

int main()
{

    char s[MAXN];
    init();
    int n; scanf("%d", &n);
    REP(i,n) {
        int op; scanf("%d", &op);
        int v,p,k;
        if (op == 1) scanf("%d%s", &p,s), Insert(p-cntC,s);
        else if (op == 2) scanf("%d%d", &p,&k), Remove(p-cntC,k-cntC);
        else scanf("%d%d%d", &v,&p,&k), print(v-cntC,p-cntC,k-cntC);
    }

    return 0;
}
```

## 4.7 可持久化平衡树

P5055 【模板】可持久化文艺平衡树

模板

可持久化

支持：

1. 在第 p 个数后插入数 xx 。
2. 删除第 p 个数。
3. 翻转区间 [l,r][l,r]
4. 查询区间 [l,r][l,r] 中所有数的和。

以上操作基于某个历史版本

可持久化只需在原来fhq_treap基础上添加：凡是涉及到修改全部newnode，并复制原来结点信息

注意这里Merge操作中并没有新开结点，因为split和merge总是成对出现

```
struct fhq_treap {
    #define lt T[x].ls
    #define rt T[x].rs
    struct node {
        int rnd,sz,ls,rs,rev;
        LL val,sum;
    } T[MAXN<<7];
    int root[MAXN],tot;
    void init() {
        srand(19981111);
        tot=0;
        memset(root,0,sizeof(root));
    }
    int newnode(LL val){ T[++tot]={rand(),1,0,0,0,val,val}; return tot; }
    int cp(int x) { T[++tot]=T[x]; return tot; }
    void revone(int& x) {
        if (x) {
            x=cp(x);
            T[x].rev^=1;
            swap(lt,rt);
        }
    }
    void up(int x) {
        if (x) {
            T[x].sz=T[lt].sz+T[rt].sz+1;
            T[x].sum=T[lt].sum+T[rt].sum+T[x].val;
        }
    }
    void down(int x) {
        if (x) {
            if (T[x].rev) revone(lt), revone(rt);
            T[x].rev=0;
        }
    }
    void split(int x, int& l, int& r, int k) {
        if (!x){ l=r=0; return; }
        down(x);
        if (k<=T[lt].sz) r=cp(x), split(lt,l,T[r].ls,k), up(r);
        else l=cp(x), split(rt,T[l].rs,r,k-T[lt].sz-1), up(l);
```

```
    }
    void Merge(int& x, int l, int r) {
        if (!l||!r) x=l+r;
        else if (T[l].rnd<T[r].rnd) down(x=l), Merge(rt,rt,r), up(x);
        else down(x=r), Merge(lt,l,lt), up(x); //down(x=cp(r))
    }
    void Insert(int now, int v, int pos, LL val) {
        int x,y;
        split(root[v],x,y,pos);
        Merge(x,x,newnode(val)); Merge(root[now],x,y);
    }
    void del(int now, int v, int pos) {
        int x,y,z;
        split(root[v],x,y,pos); split(x,x,z,pos-1);
        Merge(root[now],x,y);
    }
    void Reverse(int now, int v, int l, int r) {
        int x,y,z;
        split(root[v],x,y,r); split(x,x,z,l-1);
        revone(z);
        Merge(x,x,z); Merge(root[now],x,y);
    }
    LL query_sum(int now, int v, int l, int r) {
        int x,y,z;
        split(root[v],x,y,r); split(x,x,z,l-1);
        LL ans=T[z].sum;
        Merge(x,x,z); Merge(root[now],x,y);
        return ans;
    }
} solve;

int main()
{

    int q;
    while (scanf("%d", &q)==1) {
        solve.init();
        LL ans=0;
        FOR(i,1,q) {
            int v,op; scanf("%d%d", &v,&op);
            LL x,y;
            if (op==1) {
                scanf("%lld%lld", &x,&y);
                x^=ans, y^=ans;
                solve.Insert(i,v,x,y);
            } else if (op==2) {
                scanf("%lld", &x);
                x^=ans;
                solve.del(i,v,x);
            } else if (op==3) {
                scanf("%lld%lld", &x,&y);
                x^=ans, y^=ans;
                solve.Reverse(i,v,x,y);
            } else {
                scanf("%lld%lld", &x,&y);
```

```
            x^=ans, y^=ans;
            ans=solve.query_sum(i,v,x,y);
            printf("%lld\n", ans);
        }
    }
}

return 0;
}
```

P3835 【模板】可持久化平衡树

模板

2.42s（O2）

支持历史版本，支持：

1. 插入x数

2. 删除x数(若有多个相同的数，因只删除一个，如果没有请忽略该操作)

3. 查询x数的排名(排名定义为比当前数小的数的个数+1。若有多个相同的数，因输出最小的排名)

4. 查询排名为x的数

5. 求x的前驱(前驱定义为小于x，且最大的数，如不存在输出-2147483647)

6. 求x的后继(后继定义为大于x，且最小的数，如不存在输出2147483647)

```
struct fhq_treap {
    #define lt T[x].ls
    #define rt T[x].rs
    struct node {
        int rnd,sz,ls,rs,sum;
    } T[MAXN<<6];
    int root[MAXN],tot;
    void init() {
        srand(19981111);
        tot=0;
    }
    int newnode(int val){ T[++tot]={rand(),1,0,0,val}; return tot; }
    int cp(int x){ T[++tot]=T[x]; return tot; }
    void up(int x){ if (x) T[x].sz=T[lt].sz+T[rt].sz+1; }
    void split(int x, int& l, int& r, int k) {
        if (!x) { l=r=0; return; }
        if (k<=T[lt].sz) r=cp(x), split(lt,l,T[r].ls,k), up(r);
        else l=cp(x), split(rt,T[l].rs,r,k-T[lt].sz-1), up(l);
    }
    void Merge(int& x, int l, int r) {
        if (!l||!r) x=l+r;
        else if (T[l].rnd<T[r].rnd) x=l, Merge(rt,rt,r), up(x);
        else x=r, Merge(lt,l,lt), up(x);
    }
    //the max pos of x s.t. x<val
    int Rank(int x, int val) {
        if (!x) return 0;
        if (T[x].sum>=val) return Rank(lt,val);
        return T[lt].sz+1+Rank(rt,val);
```

```
    }
    void Insert(int now, int v, int val) {
        int x,y;
        split(root[v],x,y,Rank(root[v],val));
        Merge(x,x,newnode(val)); Merge(root[now],x,y);
    }
    void del(int now, int v, int val) {
        int x,y,z,rnk=Rank(root[v],val);
        if (rnk>=T[root[v]].sz) { root[now]=root[v]; return; }
        split(root[v],x,y,rnk+1), split(x,x,z,rnk);
        if (T[z].sum!=val) Merge(x,x,z);
        Merge(root[now],x,y);
    }
    int kth(int now, int v, int k) {
        int x,y,z;
        split(root[v],x,y,k); split(x,x,z,k-1);
        int ans=T[z].sum;
        Merge(x,x,z); Merge(root[now],x,y);
        return ans;
    }
    int query_rank(int now, int v, int val) {
        root[now]=root[v];
        return Rank(root[v],val)+1;
    }
    int pre(int now, int v, int val) {
        int x,y,z,rnk=Rank(root[v],val);
        if (rnk<1) {
            root[now]=root[v];
            return -2147483647;
        } else {
            split(root[v],x,y,rnk); split(x,x,z,rnk-1);
            int ans=T[z].sum;
            Merge(x,x,z); Merge(root[now],x,y);
            return ans;
        }
    }
    int succ(int now, int v, int val) {
        int x,y,z,rnk=Rank(root[v],val+1);
        if (rnk>=T[root[v]].sz) {
            root[now]=root[v];
            return 2147483647;
        } else {
            split(root[v],x,y,rnk+1), split(x,x,z,rnk);
            int ans=T[z].sum;
            Merge(x,x,z); Merge(root[now],x,y);
            return ans;
        }
    }
} solve;

int main()
{

    int n;
    while (scanf("%d", &n)==1) {
```

```
        solve.init();
        FOR(i,1,n) {
            int v,op,val; scanf("%d%d%d", &v,&op,&val);
            switch (op) {
                case 1: solve.Insert(i,v,val); break;
                case 2: solve.del(i,v,val); break;
                case 3: printf("%d\n", solve.query_rank(i,v,val)); break;
                case 4: printf("%d\n", solve.kth(i,v,val)); break;
                case 5: printf("%d\n", solve.pre(i,v,val)); break;
                case 6: printf("%d\n", solve.succ(i,v,val)); break;
            }
        }
    }

    return 0;
}
```

## 4.8 主席树

P3834 【模板】可持久化线段树 1（主席树）
区间第k大裸题

```
#define mid (l+r)/2
#define lt T[x].ls
#define rt T[x].rs
struct node {
    int sz,ls,rs;
} T[MAXN*20];
int root[MAXN],tot;
void init(){ tot=0; root[0]=0; T[0]={0,0,0}; }
int cp(int x){ T[++tot]=T[x]; return tot; }
void update(int& x, int v, int l, int r, int val) {
    x=cp(v);
    T[x].sz++;
    if (l<r) {
        if (val<=mid) update(lt,lt,l,mid,val);
        else update(rt,rt,mid+1,r,val);
    }
}
int query(int u, int v, int l, int r, int k) {
    if (l>=r) return l;
    int sz=T[T[v].ls].sz-T[T[u].ls].sz;
    if (k<=sz) return query(T[u].ls,T[v].ls,l,mid,k);
    else return query(T[u].rs,T[v].rs,mid+1,r,k-sz);
}
int n,m,q,a[MAXN],b[MAXN];
int ID(int x){ return lower_bound(b+1,b+1+m,x)-b; }

int main()
{
```

```
    while (scanf("%d%d", &n,&q)==2) {
        init();
        FOR(i,1,n) scanf("%d", &a[i]), b[i]=a[i];
        sort(b+1,b+1+n);
        m=unique(b+1,b+1+n)-b-1;
        FOR(i,1,n) update(root[i],root[i-1],1,m,ID(a[i]));
        while (q--) {
            int l,r,k; scanf("%d%d%d", &l,&r,&k);
            printf("%d\n", b[query(root[l-1],root[r],1,m,k)]);
        }
    }

    return 0;
}
```

K-th Closest Distance HDU - 6621
区间小于数x的个数模板

```
//the number of x s.t. x<k
int query(int u, int v, int l, int r, int k) {
    if (k<=l) return 0;
    if (k>r) return sum[v]-sum[u];
    return query(L[u],L[v],l,mid,k)+query(R[u],R[v],mid+1,r,k);
}
```

P3919 【模板】可持久化数组（可持久化线段树/平衡树）
可持久化线段树单点修改单点查询

```
#define mid (l+r)/2
#define lt T[x].ls
#define rt T[x].rs
struct node {
    int val,ls,rs;
} T[MAXN<<5];
int root[MAXN],tot;
void init(){ tot=0; }
int newnode(int val){ T[++tot]={val,0,0}; return tot; }
int cp(int x){ T[++tot]=T[x]; return tot; }
int build(int l, int r) {
    int x=newnode(0);
    if (l==r) scanf("%d", &T[x].val);
    else {
        lt=build(l,mid);
        rt=build(mid+1,r);
    }
    return x;
}
int update(int v, int l, int r, int p, int val) {
    int x=cp(v);
    if (l==r) T[x].val=val;
```

```
    else {
        if (p<=mid) lt=update(lt,l,mid,p,val);
        else rt=update(rt,mid+1,r,p,val);
    }
    return x;
}
int query(int x, int l, int r, int p) {
    if (l==r) return T[x].val;
    else {
        if (p<=mid) return query(lt,l,mid,p);
        else return query(rt,mid+1,r,p);
    }
}

int main()
{

    int n,q;
    while (scanf("%d%d", &n,&q)==2) {
        init();
        root[0]=build(1,n);
        FOR(i,1,q) {
            int v,op,p,val; scanf("%d%d%d", &v,&op,&p);
            if (op==1) {
                scanf("%d", &val);
                root[i]=update(root[v],1,n,p,val);
            } else {
                root[i]=root[v];
                printf("%d\n", query(root[v],1,n,p));
            }
        }
    }

    return 0;
}
```

P3567 [POI2014]KUR-Couriers

查询区间中出现超过区间长度一半的数，没有输出0

当且仅当一个结点size超过区间长度一半才有答案，查询左右孩子size，递归下去即可

（如果查询最小，查询最大，查询中间大，貌似会超时，常数大）

```
int query(int u, int v, int l, int r, int len) {
    if (l==r) return l;
    int lsz=T[T[v].ls].sz-T[T[u].ls].sz;
    int rsz=T[T[v].rs].sz-T[T[u].rs].sz;
    if (2*lsz>len) return query(T[u].ls,T[v].ls,l,mid,len);
    if (2*rsz>len) return query(T[u].rs,T[v].rs,mid+1,r,len);
    return 0;
}
```

P2633 Count on a tree

树上第k大模板

从根到某个结点可以看成一个区间，这样可以转化为区间第k大，方法是sum[u]+sum[v]-sum[lca]-sum[lca_fa]

需要求出lca，用了树链剖分（常数小），而且dfs的时候顺便更新了主席树

```
int n,m,q,a[MAXN],b[MAXN];
int to[MAXM],nxt[MAXM],f[MAXN],tot;
void add(int u, int v) {
    to[tot]=v;
    nxt[tot]=f[u];
    f[u]=tot++;
}
#define mid (l+r)/2
int sum[MAXN<<5],L[MAXN<<5],R[MAXN<<5],root[MAXN],cnt;
int cp(int x){ sum[++cnt]=sum[x]; L[cnt]=L[x]; R[cnt]=R[x]; return cnt; }
void update(int& x, int v, int l, int r, int val) {
    x=cp(v); sum[x]++;
    if (l<r) {
        if (val<=mid) update(L[x],L[x],l,mid,val);
        else update(R[x],R[x],mid+1,r,val);
    }
}
int ID(int x){ return lower_bound(b+1,b+1+m,x)-b; }
int sz[MAXN],d[MAXN],fa[MAXN],son[MAXN],top[MAXN];
void dfs(int u) {
    sz[u]=1; d[u]=d[fa[u]]+1; son[u]=0;
    update(root[u],root[fa[u]],1,m,ID(a[u]));
    for (int i=f[u]; ~i; i=nxt[i]) {
        int v=to[i];
        if (v==fa[u]) continue;
        fa[v]=u; dfs(v);
        sz[u]+=sz[v];
        if (sz[v]>sz[son[u]]) son[u]=v;
    }
}
void dfs(int u, int tp) {
    top[u]=tp;
    if (son[u]) dfs(son[u],tp);
    for (int i=f[u]; ~i; i=nxt[i]) {
        int v=to[i];
        if (v==fa[u]||v==son[u]) continue;
        dfs(v,v);
    }
}
int LCA(int x, int y) {
    while (top[x]!=top[y]) d[top[x]]>=d[top[y]]?x=fa[top[x]]:y=fa[top[y]];
    return d[x]>=d[y]?y:x;
}
void init() {
    tot=0;
    memset(f,-1,sizeof(f));
    cnt=fa[1]=sz[0]=0;
}
int query(int u, int v, int lca, int lca_fa, int l, int r, int k) {
```

```
    if (l>=r) return l;
    int sz=sum[L[u]]+sum[L[v]]-sum[L[lca]]-sum[L[lca_fa]];
    if (k<=sz) return query(L[u],L[v],L[lca],L[lca_fa],l,mid,k);
    return query(R[u],R[v],R[lca],R[lca_fa],mid+1,r,k-sz);
}

int main()
{

    while (scanf("%d%d", &n,&q)==2) {
        init();
        FOR(i,1,n) scanf("%d", &a[i]), b[i]=a[i];
        sort(b+1,b+1+n);
        m=unique(b+1,b+1+n)-b-1;
        REP(i,n-1) {
            int x,y; scanf("%d%d", &x,&y);
            add(x,y); add(y,x);
        }
        dfs(1); dfs(1,1);
        int ans=0;
        while (q--) {
            int x,y,z; scanf("%d%d%d", &x,&y,&z);
            x^=ans;
            int lca=LCA(x,y);
            ans=b[query(root[x],root[y],root[lca],root[fa[lca]],1,m,z)];
            printf("%d\n", ans);
        }
    }

    return 0;
}
```

Sequence II HDU - 5919

主席树区间不同数个数区间不同数第一次出现位置第k个位置强制在线模板

假设一个区间出现了k个不同的数，每个数第一次出现的位置从小到大记为p[i]，求p[(k+1)/2]

方法一（TLE）：预处理出每个数上一个出现的位置pre[i]，这样预处理出pre数组，求一个区间[l,r]不同数个数即求这个pre数组区间种小于l的数有多少个，对于求中位数，二分一个m，求区间[l,m]的不同数个数。复杂度为O(nlogn$\hat{2}$)，在hdu上会超时

方法二：只维护第一次出现的位置，逆序遍历，如果第一次出现，该位置加1，否则还要把之前出现的位置减1。这样相当于维护了n个版本的线段树，每次单点修改。查询一个区间有多少个不同数个数，即相当于版本root[l]的中小于等于r的个数，即区间求和，跟线段树求和一样。假如有num个不同，现在要查第(num+1)/2的位置，即第k大，方法跟主席树一样，比较左右两个子树的size递归遍历下去即可。复杂度为O(nlogn)

```
int n,q,a[MAXN];
#define mid (l+r)/2
int sum[MAXN<<6],L[MAXN<<6],R[MAXN<<6],root[MAXN],tot;
void init(){ tot=0; }
int cp(int x){ sum[++tot]=sum[x]; L[tot]=L[x]; R[tot]=R[x]; return tot; }
void update(int& x, int v, int l, int r, int p, int d) {
```

```
    x=cp(v); sum[x]+=d;
    if (l<r) {
        if (p<=mid) update(L[x],L[x],l,mid,p,d);
        else update(R[x],R[x],mid+1,r,p,d);
    }
}
int query_sum(int x, int l, int r, int ll, int rr) {
    if (ll<=l && r<=rr) return sum[x];
    int ans=0;
    if (ll<=mid) ans+=query_sum(L[x],l,mid,ll,rr);
    if (mid<rr) ans+=query_sum(R[x],mid+1,r,ll,rr);
    return ans;
}
int kth(int x, int l, int r, int k) {
    if (l==r) return l;
    int sz=sum[L[x]];
    if (k<=sz) return kth(L[x],l,mid,k);
    return kth(R[x],mid+1,r,k-sz);
}

int main()
{

    int T; scanf("%d", &T);
    FOR(kase,1,T) {
        init();
        printf("Case #%d:", kase);
        scanf("%d%d", &n,&q);
        FOR(i,1,n) scanf("%d", &a[i]);
        map<int,int> mp;
        root[n+1]=0;
        ROF(i,1,n) {
            update(root[i],root[i+1],1,n,i,1);
            if (mp.count(a[i])) update(root[i],root[i],1,n,mp[a[i]],-1);
            mp[a[i]]=i;
        }
        int ans=0;
        while (q--) {
            int l,r; scanf("%d%d", &l,&r);
            l=(l+ans)%n+1, r=(r+ans)%n+1;
            if (l>r) swap(l,r);
            int num=query_sum(root[l],1,n,l,r);
            ans=kth(root[l],1,n,(num+1)/2);
            printf(" %d", ans);
        }
        puts("");
    }

    return 0;
}
```

P3168 [CQOI2015]任务查询系统

差分区间前k小数之和强制在线模板

给n个区间，每个区间有一个权值，问某个点所在所有区间中前k小权值之和为多少

容易想到差分，对于每个区间，在l点+1，在r+1点-1，询问前缀和，区间前k小权值和用主席树维护。

注意有一个容易错的地方，当有多个相同的数时，前k小不能仅仅用size判断了

主席树不用离散化的方法：l=1,r=1e18，r理论大于可能的最大值即可。观察update函数，由于是动态开点，每次update消耗log(r-l)的空间，共有n次update，所以空间复杂度是O(nlog(max_r-min_l))，所以r取足够大也不怕

```
#define mid (l+r)/2
int sz[MAXN<<7],L[MAXN<<7],R[MAXN<<7],root[MAXN],tot;
LL sum[MAXN<<7];
void init(){ tot=0; }
int cp(int x){ sz[++tot]=sz[x]; L[tot]=L[x]; R[tot]=R[x]; sum[tot]=sum[x]; return tot; }
void update(int& x, int v, int l, int r, int val) {
    x=cp(v); sz[x]+=(val>0?1:-1); sum[x]+=val;
    if (l<r) {
        if (abs(val)<=mid) update(L[x],L[x],l,mid,val);
        else update(R[x],R[x],mid+1,r,val);
    }
}
LL query(int x, int l, int r, int k) {
    if (k<=0) return 0;
    if (sz[x]<=k) return sum[x];
    if (l==r) return (LL)k*l;
    return query(L[x],l,mid,k)+query(R[x],mid+1,r,k-sz[L[x]]);
}

int main()
{

    init();
    int n,q; scanf("%d%d", &n,&q);
    map<int,vector<pii> > mp;
    FOR(i,1,n) {
        int l,r,val; scanf("%d%d%d", &l,&r,&val);
        mp[l].pb({l,val});
        mp[r+1].pb({r+1,-val});
    }
    FOR(i,1,n) {
        root[i]=cp(root[i-1]);
        for (auto& j:mp[i]) {
            update(root[i],root[i],1,1e7,j.se);
        }
    }
    LL ans=1;
    while (q--) {
        int x,a,b,c; scanf("%d%d%d%d", &x,&a,&b,&c);
        ans=query(root[x],1,1e7,1+(a*ans+b)%c);
        printf("%lld\n", ans);
    }

    return 0;
}
```

## 4.9 树套树

https://blog.csdn.net/xyc1719/article/details/83614580（splay好板子）
P3380【模板】二逼平衡树（树套树）
线段树套平衡树
支持以下操作：
1. 查询k在区间内的排名
2. 查询区间内排名为k的值
3. 修改某一位值上的数值
4. 查询k在区间内的前驱(前驱定义为严格小于x，且最大的数，若不存在输出-2147483647)
5. 查询k在区间内的后继(后继定义为严格大于x，且最小的数，若不存在输出2147483647)
线段树维护区间信息，平衡树维护值域信息
具体来说，线段树每个结点开一颗平衡树，这个平衡树能支持rank,kth,insert,del,pre,succ操作。
操作1，写一个函数lt，求区间中小于val的数有多少个，答案即为lt+1
操作2，适用操作1的lt函数进行二分，复杂度为O(logn$\hat{3}$)，树状数组套主席树能搞到O(logn$\hat{2}$)
操作3，对包含这个位置的区间全部暴力修改，O(logn$\hat{2}$)
操作4,5，递归求出前驱后继，然后合并答案
对 https://blog.csdn.net/xyc1719/article/details/83614580 代码简化的一些学习：线段树只涉及单点修改，不需要维护l，r信息，递归函数中带上l,r即可

```
struct treap_node {
    int rnd,sz,ls,rs,sum;
} T[MAXN<<5];
int tot;
void init(){ srand(19981111); tot=0; }
namespace treap {
    #define lt T[x].ls
    #define rt T[x].rs
    int newnode(int val){ T[++tot]={rand(),1,0,0,val}; return tot; }
    void up(int x){ if (x) T[x].sz=T[lt].sz+T[rt].sz+1; }
    void split(int x, int& l, int& r, int k) {
        if (!x) { l=r=0; return; }
        if (k<=T[lt].sz) r=x, split(lt,l,lt,k), up(x);
        else l=x, split(rt,rt,r,k-T[lt].sz-1), up(x);
    }
    void Merge(int& x, int l, int r) {
        if (!l||!r) x=l+r;
        else if (T[l].rnd<T[r].rnd) x=l, Merge(rt,rt,r), up(x);
        else x=r, Merge(lt,l,lt), up(x);
    }
    int Rank(int x, int val) { //the number less than val
        if (!x) return 0;
        if (val<=T[x].sum) return Rank(lt,val);
        return T[lt].sz+1+Rank(rt,val);
```

```
    }
    void Insert(int& root, int val) {
        int x,y;
        split(root,x,y,Rank(root,val));
        Merge(x,x,newnode(val)); Merge(root,x,y);
    }
    void del(int& root, int val) {
        int x,y,z,rnk=Rank(root,val);
        split(root,x,y,rnk+1); split(x,x,z,rnk);
        Merge(root,x,y);
    }
    int pre(int& root, int val) {
        int x,y,z,rnk=Rank(root,val);
        if (rnk<1) return -2147483647;
        split(root,x,y,rnk); split(x,x,z,rnk-1);
        int ans=T[z].sum;
        Merge(x,x,z); Merge(root,x,y);
        return ans;
    }
    int succ(int& root, int val) {
        int x,y,z,rnk=Rank(root,val+1);
        if (rnk>=T[root].sz) return 2147483647;
        split(root,x,y,rnk+1); split(x,x,z,rnk);
        int ans=T[z].sum;
        Merge(x,x,z); Merge(root,x,y);
        return ans;
    }
    void show(int x) {
        if (x) {
            show(lt);
            printf("%d ", T[x].sum);
            show(rt);
        }
    }
    #undef lt
    #undef rt
}
int n,q,a[MAXN];
namespace tree {
    #define ls x<<1
    #define rs x<<1|1
    struct tree_node {
        int l,r,root;
    } tree[MAXN<<2];
    void build(int x, int l, int r) {
        tree[x].l=l, tree[x].r=r;
        tree[x].root=0;
        FOR(i,l,r) treap::Insert(tree[x].root,a[i]);
        if (l<r) {
            int mid=(l+r)/2;
            build(ls,l,mid);
            build(rs,mid+1,r);
        }
    }
    int lt(int x, int l, int r, int val) {
```

```
    int L=tree[x].l, R=tree[x].r;
    if (l<=L&&R<=r) return treap::Rank(tree[x].root,val);
    else {
        int mid=(L+R)/2;
        int ans=0;
        if (l<=mid) ans+=lt(ls,l,r,val);
        if (mid<r) ans+=lt(rs,l,r,val);
        return ans;
    }
}
int Rank(int l, int r, int val) {
    return lt(1,l,r,val)+1;
}
int kth(int l, int r, int k) {
    int ll=-1, rr=1e8+9, ans;
    while (ll<=rr) {
        int mm=(ll+rr)/2;
        if (lt(1,l,r,mm)>=k) ans=mm-1, rr=mm-1;
        else ll=mm+1;
    }
    return ans;
}
void modify(int x, int pos, int val) {
    treap::del(tree[x].root,a[pos]);
    treap::Insert(tree[x].root,val);
    int L=tree[x].l, R=tree[x].r;
    if (L<R) {
        int mid=(L+R)/2;
        if (pos<=mid) modify(ls,pos,val);
        else modify(rs,pos,val);
    }
}
void modify(int pos, int val) {
    modify(1,pos,val);
    a[pos]=val;
}
int pre(int x, int l, int r, int val) {
    int L=tree[x].l, R=tree[x].r;
    if (l<=L&&R<=r) return treap::pre(tree[x].root,val);
    else {
        int mid=(L+R)/2;
        int ans=-2147483647;
        if (l<=mid) ans=max(ans,pre(ls,l,r,val));
        if (mid<r) ans=max(ans,pre(rs,l,r,val));
        return ans;
    }
}
int pre(int l, int r, int val) { return pre(1,l,r,val); }
int succ(int x, int l, int r, int val) {
    int L=tree[x].l, R=tree[x].r;
    if (l<=L&&R<=r) return treap::succ(tree[x].root,val);
    else {
        int mid=(L+R)/2;
        int ans=2147483647;
        if (l<=mid) ans=min(ans,succ(ls,l,r,val));
```

```
            if (mid<r) ans=min(ans,succ(rs,l,r,val));
            return ans;
        }
    }
    int succ(int l, int r, int val){ return succ(1,l,r,val); }
    void show(int x) {
        int L=tree[x].l, R=tree[x].r;
        printf("[%d,%d]: ", L,R);
        treap::show(tree[x].root); puts("");
        if (L<R) show(ls), show(rs);
    }
    #undef ls
    #undef rs
}
using namespace tree;

int main()
{

    while (scanf("%d%d", &n,&q)==2) {
        init();
        FOR(i,1,n) scanf("%d", &a[i]);
        build(1,1,n);
        while (q--) {
            int op,l,r,k; scanf("%d%d%d", &op,&l,&r);
            if (op!=3) scanf("%d", &k);
            switch (op) {
                case 1: printf("%d\n", Rank(l,r,k)); break;
                case 2: printf("%d\n", kth(l,r,k)); break;
                case 3: modify(l,r); break;
                case 4: printf("%d\n", pre(l,r,k)); break;
                case 5: printf("%d\n", succ(l,r,k)); break;
            }
        }
    }

    return 0;
}
```

bzoj1901洛谷P2617 Dynamic Rankings
树状数组套主席树模板动态第k大
支持单点修改，区间询问第k大，O(logn$\hat{2}$)
常数略大，需要开O2优化才能过

```
#define mid (l+r)/2
inline int lowbit(int x){ return x&-x; }
int sum[MAXN<<8],L[MAXN<<8],R[MAXN<<8];
int xx[MAXN],yy[MAXN],rt[MAXN],a[MAXN],b[MAXN<<1],ca[MAXN],cb[MAXN],cc[MAXN];
int n,q,m,tot,totx,toty;
void init(){ tot=totx=toty=0; }
int ID(int x){ return lower_bound(b+1,b+1+m,x)-b; }
int cp(int x){ sum[++tot]=sum[x]; L[tot]=L[x]; R[tot]=R[x]; return tot; }
void update(int& x, int v, int l, int r, int p, int d) {
```

```
    x=cp(v); sum[x]+=d;
    if (l<r) {
        if (p<=mid) update(L[x],L[x],l,mid,p,d);
        else update(R[x],R[x],mid+1,r,p,d);
    }
}
int query(int l, int r, int k) {
    if (l==r) return b[l];
    int sz=0;
    FOR(i,1,totx) sz-=sum[L[xx[i]]];
    FOR(i,1,toty) sz+=sum[L[yy[i]]];
    if (k<=sz) {
        FOR(i,1,totx) xx[i]=L[xx[i]];
        FOR(i,1,toty) yy[i]=L[yy[i]];
        return query(l,mid,k);
    } else {
        FOR(i,1,totx) xx[i]=R[xx[i]];
        FOR(i,1,toty) yy[i]=R[yy[i]];
        return query(mid+1,r,k-sz);
    }
}
void add(int x, int y) {
    for (int i=x; i<=n; i+=lowbit(i)) update(rt[i],rt[i],1,m,ID(a[x]),y);
}

int main()
{

    while (scanf("%d%d", &n,&q)==2) {
        init();
        FOR(i,1,n) scanf("%d", &a[i]), b[i]=a[i];
        m=n;
        FOR(i,1,q) {
            char op[2]; scanf("%s%d%d", op,&ca[i],&cb[i]);
            if (op[0]=='Q') scanf("%d", &cc[i]);
            else b[++m]=cb[i], cc[i]=0;
        }
        sort(b+1,b+1+m);
        m=unique(b+1,b+1+m)-b-1;
        FOR(i,1,n) add(i,1);
        FOR(i,1,q) {
            if (cc[i]) {
                totx=toty=0;
                for (int j=ca[i]-1; j; j-=lowbit(j)) xx[++totx]=rt[j];
                for (int j=cb[i]; j; j-=lowbit(j)) yy[++toty]=rt[j];
                printf("%d\n", query(1,m,cc[i]));
            } else add(ca[i],-1), a[ca[i]]=cb[i], add(ca[i],1);
        }
    }

    return 0;
}
```

P3157 [CQOI2011]动态逆序对

支持两种操作，打印当前序列逆序对数；删掉一个数（凡是相同的全部删掉）

方法一：

线段树套平衡树在线

先求出总逆序对数，没删掉一个数，假设该数的位置为p，计算出区间[1,p-1]大于a[p]的数个数和区间[p+1,n]小于a[p]的数个数，更新答案，然后从序列中删掉该数。复杂度O(nlogn$\hat{2}$)

TLE（常数大）

```
struct treap_node {
    int rnd,sz,ls,rs,sum;
} T[MAXN<<5];
int tot;
void init() { srand(19981111); tot=0; }
struct treap {
    int root;
    void init(){ root=0; }
    #define lt T[x].ls
    #define rt T[x].rs
    int newnode(int val){ T[++tot]={rand(),1,0,0,val}; return tot; }
    void up(int x){ if (x) T[x].sz=T[lt].sz+T[rt].sz+1; }
    void split(int x, int& l, int& r, int k) {
        if (!x) { l=r=0; return; }
        if (k<=T[lt].sz) r=x, split(lt,l,lt,k), up(x);
        else l=x, split(rt,rt,r,k-T[lt].sz-1), up(x);
    }
    void Merge(int& x, int l, int r) {
        if (!l||!r) x=l+r;
        else if (T[l].rnd<T[r].rnd) x=l, Merge(rt,rt,r), up(x);
        else x=r, Merge(lt,l,lt), up(x);
    }
    int LT(int x, int val) {
        if (!x) return 0;
        if (val<=T[x].sum) return LT(lt,val);
        return T[lt].sz+1+LT(rt,val);
    }
    int GT(int x, int val) {
        if (!x) return 0;
        if (val>=T[x].sum) return GT(rt,val);
        return T[rt].sz+1+GT(lt,val);
    }
    void Insert(int val) {
        int x,y;
        split(root,x,y,LT(root,val));
        Merge(x,x,newnode(val)); Merge(root,x,y);
    }
    void del(int val) {
        int x,y,z,rnk=LT(root,val);
        split(root,x,y,rnk+1); split(x,x,z,rnk);
        Merge(root,x,y);
    }
    int LT(int val) { return LT(root,val); }
    int GT(int val) { return GT(root,val); }
```

```
    void show(int x) {
        if (x) {
            show(lt);
            printf(" %d", T[x].sum);
            show(rt);
        }
    }
    void show(){ show(root); }
} tree[MAXN<<2];
int n,a[MAXN];
#define ls x<<1
#define rs x<<1|1
#define mid (l+r)/2
void build(int x, int l, int r) {
    tree[x].init();
    FOR(i,l,r) tree[x].Insert(a[i]);
    if (l<r) {
        build(ls,l,mid);
        build(rs,mid+1,r);
    }
}
void del(int x, int l, int r, int p) {
    tree[x].del(a[p]);
    if (l<r) {
        if (p<=mid) del(ls,l,mid,p);
        else del(rs,mid+1,r,p);
    }
}
//[p+1,n] a[i] s.t. a[i]<a[p]
int LT(int x, int l, int r, int p) {
    if (p<l) return tree[x].LT(a[p]);
    int ans=LT(rs,mid+1,r,p);
    if (p<mid) ans+=LT(ls,l,mid,p);
    return ans;
}
//[1,p-1] a[i] s.t. a[i]>a[p]
int GT(int x, int l, int r, int p) {
    if (r<p) return tree[x].GT(a[p]);
    int ans=GT(ls,l,mid,p);
    if (mid+2<=p) ans+=GT(rs,mid+1,r,p);
    return ans;
}
int b[MAXN];
LL ans;
void merge_sort(int* A, int l, int r) {
    static int T[MAXN];
    if (r-l>1) {
        int m=(l+r)>>1;
        merge_sort(A,l,m);
        merge_sort(A,m,r);
        int i=l, j=m, k=l;
        while (i<m && j<r) {
            if (A[i]<=A[j]) T[k++] = A[i++];
            else {
                T[k++] = A[j++];
```

```
                ans+=m-i;
            }
        }
        while (i<m) T[k++] = A[i++];
        while (j<r) T[k++] = A[j++];
        memcpy(A+l,T+l,sizeof(int)*(r-l));
    }
}
void show(int x, int l, int r) {
    printf("[%d,%d]:", l,r); tree[x].show(); puts("");
    if (l<r) {
        show(ls,l,mid);
        show(rs,mid+1,r);
    }
}

int main()
{

    int n,q; scanf("%d%d", &n,&q);
    FOR(i,1,n) scanf("%d", &a[i]), b[i]=a[i];
    ans=0;
    merge_sort(b,1,n+1);
    map<int,vector<int> > mp;
    FOR(i,1,n) {
        mp[a[i]].pb(i);
    }
    init();
    build(1,1,n);
    while (q--) {
        int val; scanf("%d", &val);
        printf("%lld\n", ans);
        for (auto& p:mp[val]) {
            if (p!=1) ans-=GT(1,1,n,p);
            if (p!=n) ans-=LT(1,1,n,p);
            del(1,1,n,p);
        }
    }

    return 0;
}
```

P3759 [TJOI2017]不勤劳的图书管理员

动态逆序对树状数组套主席树模板

给定一排书，每本书有它本应该属于的位置和页数，每次可以选择两本书交换，要求计算每次交换后的贡献，每两本逆序的书产生它们的页数和的贡献。

树状数组套主席树减少空间的一个重要优化：更新的时候只有当结点为空结点时才动态开点。

树状数组套主席树就是应该像树状数组套一层主席树，写法简洁了不少

```
int n,q,a[MAXN],b[MAXN];
inline void addmod(int& a, int b){ a=((LL)a+b+MOD)%MOD; }
```

```
#define mid ((l+r)>>1)
int sz[MAXN<<8],sum[MAXN<<8],L[MAXN<<8],R[MAXN<<8],tot,rt[MAXN];
void init(){ tot=0; }
void update(int& x, int l, int r, int p, int val, int sgn) {
    if (!x) x=++tot; //!!!
    sz[x]+=sgn; addmod(sum[x],sgn*val);
    if (l<r) {
        if (p<=mid) update(L[x],l,mid,p,val,sgn);
        else update(R[x],mid+1,r,p,val,sgn);
    }
}
void LT(int x, int l, int r, int k, int& ans, int& cnt) {
    while (l<k) {
        if (mid<k) {
            addmod(ans,sum[L[x]]);
            cnt+=sz[L[x]];
            x=R[x];
            l=mid+1;
        } else {
            x=L[x];
            r=mid;
        }
    }
}
void GT(int x, int l, int r, int k, int& ans, int& cnt) {
    while (k<r) {
        if (k<=mid) {
            addmod(ans,sum[R[x]]);
            cnt+=sz[R[x]];
            x=L[x];
            r=mid;
        } else {
            x=R[x];
            l=mid+1;
        }
    }
}
inline int lowbit(int x){ return x&-x; }
void add(int p, int sgn) {
    for (int i=p; i<=n; i+=lowbit(i)) update(rt[i],1,n,a[p],b[p],sgn);
}
void sol(int& ans, int p, int sgn) {
    int res,cnt;
    res=cnt=0;
    for (int i=p-1; i; i-=lowbit(i)) GT(rt[i],1,n,a[p],res,cnt);
    addmod(ans,sgn*((res+(LL)cnt*b[p])%MOD));
    res=cnt=0;
    for (int i=p; i; i-=lowbit(i)) LT(rt[i],1,n,a[p],res,cnt);
    res*=-1; cnt*=-1;
    for (int i=n; i; i-=lowbit(i)) LT(rt[i],1,n,a[p],res,cnt);
    addmod(ans,sgn*((res+(LL)cnt*b[p])%MOD));
    add(p,sgn);
}

int main()
```

```
{

    while (scanf("%d%d", &n,&q)==2) {
        init();
        int ans=0;
        FOR(i,1,n) {
            scanf("%d%d", &a[i],&b[i]);
            sol(ans,i,1);
        }
        while (q--) {
            int l,r; scanf("%d%d", &l,&r);
            if (l>r) swap(l,r);
            if (l<r) {
                sol(ans,l,-1);
                sol(ans,r,-1);
                swap(a[l],a[r]);
                swap(b[l],b[r]);
                sol(ans,l,1);
                sol(ans,r,1);
            }
            printf("%d\n", ans);
        }
    }

    return 0;
}
```

# 第五章 图论

## 5.1 最大流dinic

洛谷 P3376
最大流模板 dinic

```
struct dinic {
    const static LL inf=0x3f3f3f3f3f3f3f3f;
    struct edge {
        int from,to;
        LL cap,flow;
    };
    int n,m,s,t;
    vector<edge> edges;
    vector<int> G[MAXN];
    bool vis[MAXN];
    int d[MAXN],cur[MAXN];
    void init(int n) {
        this->n=n;
        edges.clear();
        FOR(i,0,n) G[i].clear();
    }
    void add(int from, int to, LL cap) {
        edges.pb({from,to,cap,0});
        edges.pb({to,from,0,0});
        m=SZ(edges);
        G[from].pb(m-2);
        G[to].pb(m-1);
    }
    bool bfs() {
        memset(vis,0,sizeof(vis));
        queue<int> Q;
        Q.push(s); d[s]=0; vis[s]=1;
        while (!Q.empty()) {
            int u=Q.front(); Q.pop();
            REP(i,SZ(G[u])) {
                edge& e=edges[G[u][i]];
                if (!vis[e.to] && e.cap>e.flow) {
                    vis[e.to]=1;
                    d[e.to]=d[u]+1;
                    Q.push(e.to);
                }
            }
        }
        return vis[t];
```

```
    }
    LL dfs(int u, LL a) {
        if (u==t || !a) return a;
        LL flow=0, f;
        for (int& i=cur[u]; i<SZ(G[u]); i++) {
            edge& e=edges[G[u][i]];
            if (d[e.to]==d[u]+1 && (f=dfs(e.to,min(a,e.cap-e.flow)))>0) {
                e.flow+=f;
                edges[G[u][i]^1].flow-=f;
                flow+=f;
                a-=f;
                if (!a) break;
            }
        }
        return flow;
    }
    LL maxflow(int s, int t) {
        this->s=s, this->t=t;
        LL flow=0;
        while (bfs()) {
            memset(cur,0,sizeof(cur));
            flow+=dfs(s,inf);
        }
        return flow;
    }
} solve;

int main()
{

    int n,m,s,t;
    while (scanf("%d%d%d%d", &n,&m,&s,&t)==4) {
        solve.init(n+1);
        REP(i,m) {
            int x,y,z; scanf("%d%d%d", &x,&y,&z);
            solve.add(x,y,z);
        }
        printf("%lld\n", solve.maxflow(s,t));
    }

    return 0;
}
```

Reactor Cooling ZOJ - 2314

无源无汇有容量下界网络的可行流

新增源点s、汇点t，t到s添加一条无穷大的边，对于每条弧u-¿v，[b,c]，拆成3条：u-¿t (b), u-¿v (c-b), s-¿v (b)，有可行流当且仅当所有附加弧满载

注意此题并没有考虑重边的情况

```
int main()
{
```

```
    int T; scanf("%d", &T);
    while (T--) {
        int n,m; scanf("%d%d", &n,&m);
        solve.init(n+5);
        int s=0,t=n+1;
        solve.add(t,s,dinic::inf);
        map<pii,int> bbak;
        REP(i,m) {
            int u,v,b,c; scanf("%d%d%d%d", &u,&v,&b,&c);
            bbak[{u,v}]=b;
            solve.add(u,t,b);
            solve.add(u,v,c-b);
            solve.add(s,v,b);
        }
        solve.maxflow(s,t);
        bool ok=true;
        for (auto& e:solve.edges) {
            if ((e.from==s || e.to==t) && e.cap>e.flow) ok=false;
        }
        if (!ok) puts("NO");
        else {
            puts("YES");
            for (auto& e:solve.edges) {
                if (bbak.count({e.from,e.to})) {
                    printf("%lld\n", e.flow+bbak[{e.from,e.to}]);
                }
            }
        }
        if (T) puts("");
    }

    return 0;
}
```

## 5.2 网络流MCMF

P3381 【模板】最小费用最大流

```
template<typename Type>
struct MCMF {
    const static Type inf=0x3f3f3f3f3f3f3f3f;
    struct edge {
        int to,nxt;
        LL cap,flow;
        Type cost;
    } e[MAXM];
    int n,f[MAXN],tot,p[MAXN];
    Type d[MAXN];
    bool inq[MAXN];
    void init(int n) {
```

```
        this->n=n;
        tot=0;
        memset(f,-1,sizeof(f));
    }
    void add(int u, int v, LL cap, Type cost) {
        e[tot]={v,f[u],cap,0,cost}; f[u]=tot++;
        e[tot]={u,f[v],0,0,-cost}; f[v]=tot++;
    }
    bool spfa(int s, int t) {
        queue<int> Q;
        FOR(i,0,n) d[i]=inf;
        memset(inq,0,sizeof(inq));
        memset(p,-1,sizeof(p));
        Q.push(s), d[s]=0, inq[s]=1;
        while (!Q.empty()) {
            int u=Q.front(); Q.pop();
            inq[u]=0;
            for (int i=f[u]; ~i; i=e[i].nxt) {
                int v=e[i].to;
                if (e[i].cap>e[i].flow && d[v]>d[u]+e[i].cost) {
                    d[v]=d[u]+e[i].cost;
                    p[v]=i;
                    if (!inq[v]) {
                        inq[v]=1;
                        Q.push(v);
                    }
                }
            }
        }
        return ~p[t];
    }
    LL mcmf(int s, int t, Type& cost) {
        LL flow=0;
        cost=0;
        while (spfa(s,t)) {
            LL mi=1e18;
            for (int i=p[t]; ~i; i=p[e[i^1].to]) {
                if (mi>e[i].cap-e[i].flow) mi=e[i].cap-e[i].flow;
            }
            for (int i=p[t]; ~i; i=p[e[i^1].to]) {
                e[i].flow+=mi;
                e[i^1].flow-=mi;
                cost+=e[i].cost*mi;
            }
            flow+=mi;
        }
        return flow;
    }
};
MCMF<LL> solve;

int main()
{

    int n,m,s,t;
```

```
    while (scanf("%d%d%d%d", &n,&m,&s,&t)==4) {
        solve.init(n);
        REP(i,m) {
            int u,v,w,f; scanf("%d%d%d%d", &u,&v,&w,&f);
            solve.add(u,v,w,f);
        }
        LL cost;
        LL ans=solve.mcmf(s,t,cost);
        printf("%lld %lld\n", ans,cost);
    }

    return 0;
}
```

dijkstra优化，优化到1/3速度

```
template<typename Type>
struct MCMF {
    const static Type inf=0x3f3f3f3f3f3f3f3f;
    struct edge {
        int to,nxt;
        LL cap,flow;
        Type cost;
    } e[MAXM];
    int n,f[MAXN],tot,p[MAXN];
    Type d[MAXN],H[MAXN];
    void init(int n) {
        this->n=n;
        tot=0;
        memset(f,-1,sizeof(f));
    }
    void add(int u, int v, LL cap, Type cost) {
        e[tot]={v,f[u],cap,0,cost}; f[u]=tot++;
        e[tot]={u,f[v],0,0,-cost}; f[v]=tot++;
    }
    bool dij(int s, int t) {
        fill(d,d+1+n,inf);
        fill(p,p+1+n,-1);
        priority_queue<pll,vector<pll>,greater<pll> > pq;
        pq.push({d[s]=0,s});
        while (!pq.empty()) {
            pll now=pq.top(); pq.pop();
            int u=now.se;
            if (d[u]<now.fi) continue;
            for (int i=f[u]; ~i; i=e[i].nxt) {
                int v=e[i].to;
                if (e[i].cap>e[i].flow && d[v]>d[u]+e[i].cost+H[u]-H[v]) {
                    d[v]=d[u]+e[i].cost+H[u]-H[v];
                    p[v]=i;
                    pq.push({d[v],v});
                }
            }
        }
```

```
        return ~p[t];
    }
    LL mcmf(int s, int t, Type& cost) {
        memset(H,0,sizeof(H));
        LL flow=0;
        cost=0;
        while (dij(s,t)) {
            FOR(i,0,n) H[i]+=d[i];
            LL mi=1e18;
            for (int i=p[t]; ~i; i=p[e[i^1].to]) {
                if (mi>e[i].cap-e[i].flow) mi=e[i].cap-e[i].flow;
            }
            for (int i=p[t]; ~i; i=p[e[i^1].to]) {
                e[i].flow+=mi;
                e[i^1].flow-=mi;
            }
            flow+=mi;
            cost+=mi*H[t];
        }
        return flow;
    }
};
MCMF<LL> solve;
```

多校赛 K Subsequence HDU - 6611

卡常毒瘤

可以建边上优化

```
struct MCMF {
    struct edge {
        int to,rev,cap,cost;
    };
    int n,H[MAXN],dis[MAXN],p[MAXN],pe[MAXN];
    vector<edge> G[MAXN];
    void init(int n) {
        this->n=n;
        FOR(i,0,n) G[i].clear();
    }
    void add(int u, int v, int cap, int cost) {
        G[u].pb({v,SZ(G[v]),cap,cost});
        G[v].pb({u,SZ(G[u])-1,0,-cost});
    }
    int mcmf(int s, int t, int& cost) {
        fill(H,H+1+n,0);
        cost=0;
        int flow=0, f=INF;
        while (f) {
            fill(dis,dis+1+n,INF);
            priority_queue<pii,vector<pii>,greater<pii> > pq;
            pq.push({dis[s]=0,s});
            while (!pq.empty()) {
                pii now=pq.top(); pq.pop();
                int v=now.se;
```

```
            if (dis[v]<now.fi) continue;
            REP(i,SZ(G[v])) {
                edge& e=G[v][i];
                if (e.cap>0 && dis[e.to]>dis[v]+e.cost+H[v]-H[e.to]) {
                    dis[e.to]=dis[v]+e.cost+H[v]-H[e.to];
                    p[e.to]=v;
                    pe[e.to]=i;
                    pq.push({dis[e.to],e.to});
                }
            }
        }
        if (dis[t]==INF) break;
        FOR(i,0,n) H[i]+=dis[i];
        int d=f;
        for (int v=t; v!=s; v=p[v]) d=min(d,G[p[v]][pe[v]].cap);
        f-=d; flow+=d; cost+=d*H[t];
        for (int v=t; v!=s; v=p[v]) {
            edge& e=G[p[v]][pe[v]];
            e.cap-=d;
            G[v][e.rev].cap+=d;
        }
    }
    return flow;
  }
} solve;
```

P4001 [ICPC-Beijing 2006]狼抓兔子 / BZOJ 1001: [BeiJing2006]狼抓兔子

经典平面图最小割转对偶图最短路

对于原图的每一个面，对应对偶图的一个点；对于原图的每一个边，对应对偶图的一个边，连接两个面（如果只连通一个面，那个这条边是自环）。源点和汇点连一条边，把外部面分为两块，一个面为对偶图的起点，另一个面为终点，然后建图，跑最短路即是原图的最小割即最大流

详细参考2008国家集训队论文周冬《两极相通——浅析最大—最小定理在信息学竞赛中的应用》

洛谷oj的数据有问题，用多组数据输入会WA

```
int main()
{

    int m,n;
    scanf("%d%d", &m,&n);
    if (m==1 && n==1) { puts("0"); return 0; }
    init();
    int s=0, t=2*m*n+1, mn=m*n;
    REP(i,m)REP(j,n-1) {
        int w; scanf("%d", &w);
        int u=i*n+j, v=mn+i*n+j;
        if (!i) u=s;
        if (i==m-1) v=t;
        add(u,v,w);
        add(v,u,w);
```

```
    }
    REP(i,m-1)REP(j,n) {
        int w; scanf("%d", &w);
        int u=mn+i*n+j-1, v=(i+1)*n+j;
        if (!j) u=t;
        if (j==n-1) v=s;
        add(u,v,w);
        add(v,u,w);
    }
    REP(i,m-1)REP(j,n-1) {
        int w; scanf("%d", &w);
        int u=mn+i*n+j, v=(i+1)*n+j;
        add(u,v,w);
        add(v,u,w);
    }
    printf("%d\n", dij(s,t));

    return 0;
}
```

最大权闭合子图

https://blog.csdn.net/winter2121/article/details/80076806

闭合图：给定一个有向图，对于每个结点，如果该结点的所有出边连接的结点也属于这个图，称为闭合图

最大权闭合子图：给定一个有向图，每个结点有一个权重，有正有负，求最大闭合子图，使得权值和最大

求解方法：新增源点s，汇点t，对于所有权值为正的结点u，连一条s到u容量为结点权值的边，对于所有权值为负的结点u，连一条u到t容量为结点权值绝对值的边，对于所有边u到v，连一条u到v容量为无穷大的边，答案为正权值之和减去最大流，可以理解为能得到的最小损失为最小割。

P3410 拍照

最大权闭合子图

老板有n个下属，带每个下属出去要花费给定的钱，有m个人愿意给钱，如果带了他指定的下属，就会给钱，求最大利益

如果带下属不需要花钱，全部带上就能得到最大利益，但是要钱，考虑损失的最少的钱，转化为最大权闭合子图，每个出钱的人是一个正权值结点，每个下属是一个负权值结点，每个出钱的人指定的下属就连一条边，这样求最小割即可，答案是正权值之和减去最小割

```
int main()
{

    int m,n; scanf("%d%d", &m,&n);
    int s=0, t=m+n+1;
    solve.init(t);
    LL sum=0;
    FOR(i,1,m) {
        int w; scanf("%d", &w);
        sum+=w;
        solve.add(s,i,w);
```

```
        int x;
        while (scanf("%d", &x)==1 && x) {
            solve.add(i,m+x,solve.inf);
        }
    }
    FOR(i,1,n) {
        int w; scanf("%d", &w);
        solve.add(m+i,t,w);
    }
    printf("%lld\n", sum-solve.maxflow(s,t));

    return 0;
}
```

P2774 方格取数问题

经典二分图最大独立集=点权和-最大权匹配

给一个网格，每个格点有一个数，现在要求取一些数使得所有的数两两不相邻并且它们的和尽可能地大

源点连奇数点，容量为边权，偶数点连汇点，容量为边权，奇数点对于相邻的偶数点，连一条边，容量为无穷大，答案为所有点的和减去最大流

```
int m,n;
LL a[111][111];
const int r[]{0,0,1,-1};
const int c[]{1,-1,0,0};
inline bool vld(int i, int j){ return 0<=i&&i<m&&0<=j&&j<n; }

int main()
{

    while (scanf("%d%d", &m,&n)==2) {
        LL sum=0;
        REP(i,m)REP(j,n) scanf("%lld", &a[i][j]), sum+=a[i][j];
        int s=m*n, t=m*n+1;
        solve.init(t);
        REP(i,m)REP(j,n) {
            if ((i+j)&1) {
                solve.add(i*n+j,t,a[i][j]);
            } else {
                solve.add(s,i*n+j,a[i][j]);
                REP(k,4) {
                    int x=i+r[k], y=j+c[k];
                    if (vld(x,y)) {
                        solve.add(i*n+j,x*n+y,solve.inf);
                    }
                }
            }
        }
        printf("%lld\n", sum-solve.maxflow(s,t));
    }

    return 0;
```

```
}
```

P1231 教辅的组成

拆点

有几本书，几本答案，几本练习册，书有与某些答案和练习册有一个配对关系，求最多能配对多少个三元组

因为一本答案一本练习册都只能与一本书对应，所以把书拆点，结点容量为1

```
int main()
{

    int n1,n2,n3,m;
    while (scanf("%d%d%d", &n1,&n2,&n3)==3) {
        int n=n1+n2+n3;
        int s=0, t=n+n1+1;
        solve.init(t);
        FOR(i,1,n2) solve.add(s,n1+i,1);
        FOR(i,1,n3) solve.add(n1+n2+i,t,1);
        FOR(i,1,n1) solve.add(i,n+i,1);
        scanf("%d", &m);
        while (m--) {
            int x,y; scanf("%d%d", &x,&y);
            solve.add(n1+y,x,1);
        }
        scanf("%d", &m);
        while (m--) {
            int x,y; scanf("%d%d", &x,&y);
            solve.add(n+x,n1+n2+y,1);
        }
        printf("%lld\n", solve.maxflow(s,t));
    }

    return 0;
}
```

P1345 [USACO5.4]奶牛的电信Telecowmunication

拆点最小割点

问最少删除多少个点，使得两个点不能连通

拆点，拆成入点和出点，点容量为1，两个点有边就连一条无穷大容量的边

```
int main()
{

    int n,m,s,t;
    while (scanf("%d%d%d%d", &n,&m,&s,&t)==4) {
        s+=n;
        solve.init(2*n);
        FOR(i,1,n) solve.add(i,i+n,1);
        REP(i,m) {
            int x,y; scanf("%d%d", &x,&y);
```

```
            solve.add(x+n,y,solve.inf);
            solve.add(y+n,x,solve.inf);
        }
        printf("%lld\n", solve.maxflow(s,t));
    }

    return 0;
}
```

## 5.3 二分图

https://www.bilibili.com/video/av16362938?from=search&seid=1697643797530960926
过山车 HDU - 2063
二分图裸题无权值无向二分图模板

```
int to[MAXM],f[MAXN],nxt[MAXM],tot;
void init()
{
    tot=0;
    memset(f,-1,sizeof(f));
}
void add(int u, int v)
{
    to[tot]=v;
    nxt[tot]=f[u];
    f[u]=tot++;
}
int linker[MAXN],used[MAXN],uN;
bool dfs(int u)
{
    for (int i=f[u]; ~i; i=nxt[i]) {
        int v=to[i];
        if (!used[v]) {
            used[v]=1;
            if (linker[v]==-1 || dfs(linker[v])) {
                linker[v]=u;
                return true;
            }
        }
    }
    return false;
}
int hungary()
{
    int ans=0;
    memset(linker,-1,sizeof(linker));
    FOR(i,1,uN) {
        memset(used,0,sizeof(used));
        if (dfs(i)) ans++;
    }
```

```
    return ans;
}

int main()
{

    int k,m,n;
    while (scanf("%d%d%d", &k,&m,&n)==3 && k) {
        init();
        uN=m;
        REP(i,k) {
            int x,y; scanf("%d%d", &x,&y);
            add(x,y+m), add(y+m,x);
        }
        printf("%d\n", hungary());
    }

    return 0;
}
```

奔小康赚大钱 HDU - 2255
带权二分图模板

```
int nx,ny,g[MAXN][MAXN];
int linker[MAXN],lx[MAXN],ly[MAXN];
int slack[MAXN];
bool visx[MAXN],visy[MAXN];
bool dfs(int x)
{
    visx[x]=true;
    REP(y,ny) {
        if (visy[y]) continue;
        int tmp=lx[x]+ly[y]-g[x][y];
        if (!tmp) {
            visy[y]=1;
            if (linker[y]==-1 || dfs(linker[y])) {
                linker[y]=x;
                return true;
            }
        } else if (slack[y]>tmp) slack[y]=tmp;
    }
    return false;
}
int KM()
{
    memset(linker,-1,sizeof(linker));
    memset(ly,0,sizeof(ly));
    REP(i,nx) {
        lx[i]=-INF;
        REP(j,ny) lx[i]=max(lx[i],g[i][j]);
    }
    REP(x,nx) {
        REP(i,ny) slack[i]=INF;
```

```
        while (true) {
            memset(visx,0,sizeof(visx));
            memset(visy,0,sizeof(visy));
            if (dfs(x)) break;
            int d=INF;
            REP(i,ny) if (!visy[i] && d>slack[i]) d=slack[i];
            REP(i,nx) if (visx[i]) lx[i]-=d;
            REP(i,ny) {
                if (visy[i]) ly[i]+=d;
                else slack[i]-=d;
            }
        }
    }
    int ans=0;
    REP(i,ny) if (~linker[i]) ans+=g[linker[i]][i];
    return ans;
}

int main()
{

    int n;
    while (scanf("%d", &n)==1) {
        memset(g,0,sizeof(g));
        REP(i,n)REP(j,n) scanf("%d", &g[i][j]);
        nx=ny=n;
        printf("%d\n", KM());
    }

    return 0;
}
```

## 5.4 spfa

poj 2139 （acm题解里面）

poj 3259

spfa判负环，判断是否存在负环，加上vis数组判断是否遍历了图的所有结点

spfa判正环，d数组初始化为0

bfs版dfs已经够快，除非数据被针对，否则只能用dfs版spfa，一个简单而有效的改进是，所有结点进队次数大于T*(n+m)次以上即可判断负环，T根据实际情况一般取2，此算法牺牲了正确性，不过大多数情况下是正确的

```
int T, n, m, w, d[MAXN], inq[MAXN], cnt[MAXN], kase, vis[MAXN];
vector<pii> E[MAXN];

bool spfa(int s)
{
    memset(d+1, 0x3f, sizeof(int) * n);
    memset(cnt+1, 0, sizeof(int) * n);
```

```
    queue<int> Q;
    Q.push(s); inq[s] = kase; d[s] = 0; vis[s] = kase;
    while (!Q.empty()) {
        int u = Q.front(); Q.pop();
        inq[u] = 0;
        REP(i,SZ(E[u])) {
            int v = E[u][i].fi, dist = E[u][i].se;
            if (d[v] > d[u] + dist) {
                d[v] = d[u] + dist;
                if (inq[v] != kase) {
                    vis[v] = kase;
                    inq[v] = kase;
                    Q.push(v);
                    if (++cnt[v] > n) return false;
                }
            }
        }
    }
    return true;
}

int main()
{

    scanf("%d", &T);
    for (kase = 1; kase <= T; kase++) {
        scanf("%d%d%d", &n, &m, &w);
        FOR(i,1,n) E[i].clear();
        REP(i,m) {
            int x, y, z; scanf("%d%d%d", &x, &y, &z);
            E[x].pb(pii(y,z)); E[y].pb(pii(x,z));
        }
        REP(i,w) {
            int x, y, z; scanf("%d%d%d", &x, &y, &z);
            E[x].pb(pii(y,-z));
        }
        bool ans = true;
        FOR(i,1,n) if (vis[i] != kase) ans &= spfa(i);
        puts(ans ? "NO" : "YES");
    }

    return 0;
}
```

POJ - 2240

判正环，spfa_dfs版875ms，可能是边的存储慢了，或者数据被针对？或者dfs版写错了？还是输入或者map的问题？

```
int n, m, vis[MAXN], ins[MAXN], kase;
double E[MAXN][MAXN], d[MAXN];
map<string, int> mp;

bool spfa_dfs(int u)
```

```
{
    vis[u] = ins[u] = kase;
    FOR(v,1,n) if (d[v] < d[u]*E[u][v]) {
        d[v] = d[u]*E[u][v];
        if (ins[v] == kase || spfa_dfs(v)) return true;
    }
    ins[u] = 0;
    return false;
}

int main()
{

    while (scanf("%d", &n), n) {
        printf("Case %d: ", ++kase);
        mp.clear();
        FOR(i,1,n) {
            string s; cin >> s;
            mp[s] = i;
        }
        scanf("%d", &m);
        memset(E, 0, sizeof(E));
        FOR(i,1,m) {
            string s, t; double r; cin >> s >> r >> t;
            E[mp[s]][mp[t]] = r;
        }
        bool ok = false;
        FOR(i,1,n) if (vis[i] != kase) {
            FOR(i,1,n) d[i] = 0;
            d[i] = 1;
            ok |= spfa_dfs(i);
            if (ok) break;
        }
        puts(ok ? "Yes" : "No");
    }

    return 0;
}
```

# 第六章 字符串

## 6.1 KMP

P3375 【模板】KMP字符串匹配

一些细节：next数组第m个元素也是有值的，方便匹配成功后继续next。此题无需优化，最简单写法即可，next数组打印从1到m而不是从0到m-1。

```
char T[MAXN],P[MAXN];
int nxt[MAXN],n,m;
void getnext()
{
    int j = 0, k = nxt[0] = -1;
    while (j < m) {
        //if (k==-1 || P[j]==P[k]) j++,k++,nxt[j] = P[j]!=P[k] ? k : nxt[k];
        if (k==-1 || P[j]==P[k]) nxt[++j] = ++k;
        else k = nxt[k];
    }
}
void KMP()
{
    int i = 0, j = 0;
    while (i < n) {
        if (j==-1 || T[i]==P[j]) i++,j++;
        else j = nxt[j];
        if (j==m) printf("%d\n", i-j+1);
    }
}

int main()
{

    scanf("%s%s", T,P);
    n = strlen(T), m = strlen(P);
    getnext(); KMP();
    FOR(i,1,m-1) printf("%d ", nxt[i]); printf("%d\n", nxt[m]);

    return 0;
}
```

Period UVALive - 3026

训练指南第三章例题13

KMPnext数组、最小循环节

如果前i个字符组成一个周期串，则“错位”部分恰好是一个循环节：满足nxt[i]¿0，i

此处next数组定义为“最长相同前后缀”

```
int n,nxt[MAXN],kase;
char a[MAXN];
void get_next(char* P)
{
    int j=0, k=nxt[0]=-1;
    while (j<n) {
        if (k==-1 || P[j]==P[k]) nxt[++j] = ++k;
        else k = nxt[k];
    }
}

int main()
{

    while (scanf("%d", &n)==1 && n) {
        scanf("%s", a);
        get_next(a);
        printf("Test case #%d\n", ++kase);
        FOR(i,2,n) if (nxt[i]>0 && i%(i-nxt[i])==0)
            printf("%d %d\n", i,i/(i-nxt[i]));
        puts("");
    }

    return 0;
}
```

D - Cyclic Nacklace HDU - 3746

最小循环节

i-nxt[i]的含义是前i个元素中最小循环节的长度。

```
int n,nxt[MAXN],kase;
char a[MAXN];
void get_next(char* P)
{
    int j=0, k=nxt[0]=-1;
    while (j<n) {
        if (k==-1 || P[j]==P[k]) nxt[++j] = ++k;
        else k = nxt[k];
    }
}

int main()
{

    int T; scanf("%d", &T);
    while (T--) {
        scanf("%s", a);
        n=strlen(a), get_next(a);
        int cir = n-nxt[n];
        if (n%cir==0 && n/cir>1) puts("0");
        else printf("%d\n", cir-n%cir);
```

```
    }

    return 0;
}
```

扩展KMP模板

扩展KMP解决S所有后缀子串与T的最长公共前缀长度问题。

next数组含义：T[i,m-1]与T的最长相同前缀长度

extend数组含义：S[i,n-1]与T的最长相同前缀长度

```
int n,m,nxt[MAXN],ext[MAXN];
void get_next(char* T)
{
    int a=0, p=0;
    nxt[0] = m;
    FOR(i,1,m-1) {
        if (i>=p || i+nxt[i-a]>=p) {
            if (i>=p) p=i;
            while (p<m && T[p]==T[p-i]) p++;
            nxt[i] = p-i;
            a = i;
        } else nxt[i] = nxt[i-a];
    }
}
void extKMP(char* S, char* T)
{
    int a=0, p=0;
    REP(i,n) {
        if (i>=p || i+nxt[i-a]>=p) {
            if (i>=p) p=i;
            while (p<n && p-i<m && S[p]==T[p-i]) p++;
            ext[i] = p-i;
            a = i;
        } else ext[i] = nxt[i-a];
    }
}
```

# 第七章 其他

## 7.1 cdq分治

归并排序统计逆序对

```
void merge_sort(int* A, int l, int r)
{
    static int T[MAXN];
    if (r-l>1) {
        int m=(l+r)>>1;
        merge_sort(A,l,m);
        merge_sort(A,m,r);
        int i=l, j=m, k=l;
        while (i<m && j<r) {
            if (A[i]<=A[j]) T[k++] = A[i++];
            else {
                T[k++] = A[j++];
                //ans+=m-i;
            }
        }
        while (i<m) T[k++] = A[i++];
        while (j<r) T[k++] = A[j++];
        memcpy(A+l,T+l,sizeof(int)*(r-l));
    }
}
```

P3810 【模板】三维偏序（陌上花开）

统计每个元素满足a[i]¡=a[j]&&b[i]¡=b[j]&&c[i]¡=c[j]的有多少个

cdq分治，离线，先按x,y,z排序去重并统计重复元素个数（要考虑重复元素的贡献），因为题目要求的是¡=，然后像归并排序那样将y排序，合并的时候由于x已经是有序，合并y时用树状数组像统计逆序对那样统计符合条件z的元素个数

使用间接排序，一定程度减少了排序复制元素的时间

```
struct BIT {
    int n,c[MAXN*2];
    void init(int n){ this->n=n; memset(c,0,sizeof(c)); }
    inline int lowbit(int x){ return x&-x; }
    void add(int x, int d){ for (int i=x; i<=n; i+=lowbit(i)) c[i]+=d; }
    int sum(int x){ int ans=0; for (; x; x-=lowbit(x)) ans+=c[x]; return ans; }
} bit;
struct node {
    int x,y,z,c,ans;
    bool operator!=(const node& o) const {
```

```
        return x!=o.x || y!=o.y || z!=o.z;
    }
} a[MAXN],b[MAXN];
int n,k,cnt[MAXN];
bool cmpx(node& x, node& y) {
    if (x.x!=y.x) return x.x<y.x;
    if (x.y!=y.y) return x.y<y.y;
    return x.z<y.z;
}
bool cmpy(node& x, node& y) {
    return x.y<y.y;
}
void cdq(int l, int r) {
    static node T[MAXN];
    if (r-l>1) {
        int m=(l+r)/2;
        cdq(l,m);
        cdq(m,r);
        int i=l;
        for (int j=m; j<r; j++) {
            while (i<m && a[i].y<=a[j].y) bit.add(a[i].z,a[i].c), i++;
            a[j].ans+=bit.sum(a[j].z);
        }
        while (--i>=l) bit.add(a[i].z,-a[i].c);
        merge(a+l,a+m,a+m,a+r,T+l,cmpy);
        memcpy(a+l,T+l,sizeof(T[l])*(r-l));
    }
}

int main()
{

    scanf("%d%d", &n,&k);
    bit.init(k);
    REP(i,n) {
        int x,y,z; scanf("%d%d%d", &x,&y,&z);
        b[i]={x,y,z,0,0};
    }
    sort(b,b+n,cmpx);
    int m=0;
    REP(i,n) {
        if (!m || a[m-1]!=b[i]) {
            a[m++]=b[i];
        }
        a[m-1].c++;
    }
    cdq(0,m);
    memset(cnt,0,sizeof(cnt));
    REP(i,m) cnt[a[i].ans+a[i].c-1]+=a[i].c;
    REP(i,n) printf("%d\n", cnt[i]);

    return 0;
}
```

## 7.2 整体二分

P3834【模板】可持久化线段树 1（主席树）

静态区间第k大整体二分模板

巨坑爹的地方是，只能用(l+r)¿¿1而不能用(l+r)/2，因为有负数，导致取整方向不对，导致二分死循环。

```
struct query {
    int l,r,k,op,ind;
} q[MAXN<<1],q1[MAXN<<1],q2[MAXN<<1];
struct BIT {
    int n,tree[MAXN];
    void init(int n){ this->n=n; memset(tree,0,sizeof(tree)); }
    void add(int x, int d){ for (; x<=n; x+=x&-x) tree[x]+=d; }
    int sum(int x){ int ans=0; for (; x; x-=x&-x) ans+=tree[x]; return ans; }
} bit;
int n,m,res[MAXN];
void solve(int l, int r, int ql, int qr) {
    if (ql>qr) return;
    if (l==r) {
        FOR(i,ql,qr) if (q[i].op==2) res[q[i].ind]=l;
        return;
    }
    int mid=(l+r)>>1,cnt1=0,cnt2=0;
    FOR(i,ql,qr) {
        if (q[i].op==1) {
            if (q[i].l<=mid) q1[++cnt1]=q[i], bit.add(q[i].ind,q[i].r);
            else q2[++cnt2]=q[i];
        } else {
            int x=bit.sum(q[i].r)-bit.sum(q[i].l-1);
            if (q[i].k<=x) q1[++cnt1]=q[i];
            else q[i].k-=x, q2[++cnt2]=q[i];
        }
    }
    FOR(i,1,cnt1) if (q1[i].op==1) bit.add(q1[i].ind,-q[i].r);
    FOR(i,1,cnt1) q[ql+i-1]=q1[i];
    FOR(i,1,cnt2) q[ql+i-1+cnt1]=q2[i];
    solve(l,mid,ql,ql+cnt1-1);
    solve(mid+1,r,ql+cnt1,qr);
}

int main()
{


    while (scanf("%d%d", &n,&m)==2) {
        FOR(i,1,n) {
            int x; scanf("%d", &x);
            q[i]={x,0,0,1,i};
        }
        FOR(i,1,m) {
            int l,r,k; scanf("%d%d%d", &l,&r,&k);
```

```
            q[n+i]={l,r,k,2,i};
        }
        bit.init(n);
        solve(-INF,INF,1,n+m);
        FOR(i,1,m) printf("%d\n", res[i]);
    }

    return 0;
}
```

P2617 Dynamic Rankings

单点修改区间第k大整体二分模板

不吸氧比吸氧的带修主席树快一倍，吸氧用时是吸氧的带修主席树的1/5

```
struct query {
    int l,r,k,ind,op;
} q[MAXN*3],q1[MAXN*3],q2[MAXN*3];
struct BIT {
    int n,tree[MAXN];
    void init(int n){ this->n=n; memset(tree,0,sizeof(tree)); }
    void add(int x, int d){ for (; x<=n; x+=x&-x) tree[x]+=d; }
    int sum(int x){ int ans=0; for (; x; x-=x&-x) ans+=tree[x]; return ans; }
} bit;
int n,m,a[MAXN],cnt,tot,res[MAXN];
void solve(int l, int r, int ql, int qr) {
    if (ql>qr) return;
    if (l==r) {
        FOR(i,ql,qr) if (q[i].op==2) res[q[i].ind]=l;
        return;
    }
    int mid=(l+r)>>1,cnt1=0,cnt2=0;
    FOR(i,ql,qr) {
        if (q[i].op==1) {
            if (q[i].l<=mid) q1[++cnt1]=q[i], bit.add(q[i].ind,q[i].r);
            else q2[++cnt2]=q[i];
        } else {
            int x=bit.sum(q[i].r)-bit.sum(q[i].l-1);
            if (q[i].k<=x) q1[++cnt1]=q[i];
            else q[i].k-=x, q2[++cnt2]=q[i];
        }
    }
    FOR(i,1,cnt1) if (q1[i].op==1) bit.add(q1[i].ind,-q1[i].r);
    FOR(i,1,cnt1) q[ql+i-1]=q1[i];
    FOR(i,1,cnt2) q[ql+i-1+cnt1]=q2[i];
    solve(l,mid,ql,ql+cnt1-1);
    solve(mid+1,r,ql+cnt1,qr);
}

int main()
{

    while (scanf("%d%d", &n,&m)==2) {
        cnt=tot=0;
```

```
        FOR(i,1,n) {
            scanf("%d", &a[i]);
            q[++cnt]={a[i],1,0,i,1};
        }
        FOR(i,1,m) {
            char op[2]; scanf("%s", op);
            if (op[0]=='Q') {
                int l,r,k; scanf("%d%d%d", &l,&r,&k);
                q[++cnt]={l,r,k,++tot,2};
            } else {
                int p,x; scanf("%d%d", &p,&x);
                q[++cnt]={a[p],-1,0,p,1};
                q[++cnt]={a[p]=x,1,0,p,1};
            }
        }
        bit.init(n);
        solve(-INF,INF,1,cnt);
        FOR(i,1,tot) printf("%d\n", res[i]);
    }

    return 0;
}
```

P3332 [ZJOI2013]K大数查询

区间加区间第k大整体二分模板

题意：区间加上一个数，不是累加，是添一个数

```
namespace ST {
    #define ls x<<1
    #define rs x<<1|1
    #define mid ((l+r)>>1)
    struct node {
        int l,r;
        LL sum,add;
        void update(LL val) {
            sum+=val*(r-l+1);
            add+=val;
        }
    } tree[MAXN<<2];
    void down(int x) {
        if (tree[x].add) {
            tree[ls].update(tree[x].add);
            tree[rs].update(tree[x].add);
            tree[x].add=0;
        }
    }
    void up(int x) {
        tree[x].sum=tree[ls].sum+tree[rs].sum;
    }
    void build(int x, int l, int r) {
        tree[x]={l,r,0,0};
        if (l<r) {
            build(ls,l,mid);
```

```
            build(rs,mid+1,r);
        }
    }
    void update(int x, int l, int r, int ql, int qr, LL val) {
        if (ql<=l && r<=qr) tree[x].update(val);
        else {
            down(x);
            if (ql<=mid) update(ls,l,mid,ql,qr,val);
            if (mid<qr) update(rs,mid+1,r,ql,qr,val);
            up(x);
        }
    }
    LL query(int x, int l, int r, int ql, int qr) {
        if (ql<=l && r<=qr) return tree[x].sum;
        else {
            LL ans=0;
            down(x);
            if (ql<=mid) ans+=query(ls,l,mid,ql,qr);
            if (mid<qr) ans+=query(rs,mid+1,r,ql,qr);
            up(x);
            return ans;
        }
    }
    #undef mid
}
struct query {
    int l,r;
    LL k;
    int ind,op;
} q[MAXN],q1[MAXN],q2[MAXN];
int n,m,tot,res[MAXN];
void solve(int l, int r, int ql, int qr) {
    if (ql>qr) return;
    if (l==r) {
        FOR(i,ql,qr) if (q[i].op==2) res[q[i].ind]=l;
        return;
    }
    int mid=(l+r)>>1,cnt1=0,cnt2=0;
    FOR(i,ql,qr) {
        if (q[i].op==1) {
            if (q[i].k<=mid) q1[++cnt1]=q[i];
            else q2[++cnt2]=q[i], ST::update(1,1,n,q[i].l,q[i].r,1);
        } else {
            LL x=ST::query(1,1,n,q[i].l,q[i].r);
            if (q[i].k<=x) q2[++cnt2]=q[i];
            else q[i].k-=x, q1[++cnt1]=q[i];
        }
    }
    FOR(i,1,cnt2) if (q2[i].op==1) ST::update(1,1,n,q2[i].l,q2[i].r,-1);
    FOR(i,1,cnt1) q[ql+i-1]=q1[i];
    FOR(i,1,cnt2) q[ql+i-1+cnt1]=q2[i];
    solve(l,mid,ql,ql+cnt1-1);
    solve(mid+1,r,ql+cnt1,qr);
}
```

```
int main()
{

    while (scanf("%d%d", &n,&m)==2) {
        tot=0;
        FOR(i,1,m) {
            int op,l,r;
            LL x; scanf("%d%d%d%lld", &op,&l,&r,&x);
            q[i]={l,r,x,0,op};
            if (op==2) q[i].ind=++tot;
        }
        ST::build(1,1,n);
        solve(-n,n,1,m);
        FOR(i,1,tot) printf("%d\n", res[i]);
    }

    return 0;
}
```

## 7.3 莫队

P1494 [国家集训队]小Z的袜子

普通莫队最终模板

询问区间随机取两个数相同的概率是多少，输出最简分数

组合数C(n+1,2)-C(n,2)=n，然后就很简单了，注意L==R的时候要特判，这时候分子分母都是0，gcd为0，除0会RE

```
int n,m,a[MAXN];
struct query {
    int l,r,ind;
} q[MAXN];
int bsz,belong[MAXN],cnt[MAXN];
LL tot,ansa,ansb,resa[MAXN],resb[MAXN];
bool cmp(query& a, query& b) {
    if (belong[a.l]!=belong[b.l]) return belong[a.l]<belong[b.l];
    return (belong[a.l]&1)?(a.r>b.r):(a.r<b.r);
}
void prelude() {
    bsz=sqrt(n);
    FOR(i,1,n) belong[i]=i/bsz;
    sort(q+1,q+1+m,cmp);
}
void add(int x) {
    ansa+=cnt[x];
    cnt[x]++;
    ansb+=tot;
    tot++;
}
void del(int x) {
```

```
    tot--;
    ansb-=tot;
    cnt[x]--;
    ansa-=cnt[x];
}
LL gcd(LL a, LL b){ return b==0?a:gcd(b,a%b); }
void mt() {
    int l=1,r=0;
    ansa=ansb=tot=0;
    memset(cnt,0,sizeof(cnt));
    FOR(i,1,m) {
        int ql=q[i].l, qr=q[i].r, ind=q[i].ind;
        while (l<ql) del(a[l++]);
        while (l>ql) add(a[--l]);
        while (r<qr) add(a[++r]);
        while (r>qr) del(a[r--]);
        LL g=gcd(ansa,ansb);
        if (!g) {
            resa[ind]=0;
            resb[ind]=1;
            continue;
        }
        resa[ind]=ansa/g;
        resb[ind]=ansb/g;
    }
}

int main()
{

    while (scanf("%d%d", &n,&m)==2) {
        FOR(i,1,n) scanf("%d", &a[i]);
        FOR(i,1,m) {
            int l,r; scanf("%d%d", &l,&r);
            q[i]={l,r,i};
        }
        prelude();
        mt();
        FOR(i,1,m) printf("%lld/%lld\n", resa[i],resb[i]);
    }

    return 0;
}
```

https://www.cnblogs.com/WAMonster/p/10118934.html
P1903 [国家集训队]数颜色 / 维护队列
带修改询问区间不同数个数带修莫队模板
被卡常了只能用O2

```
int n,q,a[MAXN],belong[MAXN],res[MAXN];
int bsz,cntq,cntm;
struct query {
    int l,r,t,ind;
```

```
} b[MAXN];
bool cmp(query& a, query& b) {
    if (belong[a.l]!=belong[b.l]) return belong[a.l]<belong[b.l];
    if (belong[a.r]!=belong[b.r]) return belong[a.r]<belong[b.r];
    return a.t<b.t;
}
struct modify {
    int pos,color;
} c[MAXN];
int ans,cnt[MAXM];
inline void add(int p) {
    if (cnt[a[p]]++==0) ans++;
}
inline void del(int p) {
    if (--cnt[a[p]]==0) ans--;
}
inline void update(int t, int ql, int qr) {
    int p=c[t].pos;
    if (ql<=p && p<=qr) {
        if (--cnt[a[p]]==0) ans--;
        if (cnt[c[t].color]++==0) ans++;
    }
    swap(a[p],c[t].color);
}

int main()
{

    scanf("%d%d", &n,&q);
    bsz=pow(n,2.0/3);
    FOR(i,1,n) {
        scanf("%d", &a[i]);
        belong[i]=i/bsz;
    }
    cntq=cntm=0;
    FOR(i,1,q) {
        char op[2]; scanf("%s", op);
        if (op[0]=='Q') {
            int l,r; scanf("%d%d", &l,&r);
            if (l>r) swap(l,r);
            cntq++;
            b[cntq]={l,r,cntm,cntq};
        } else {
            int p,x; scanf("%d%d", &p,&x);
            c[++cntm]={p,x};
        }
    }
    sort(b+1,b+1+cntq,cmp);
    int l=1, r=0, t=0;
    ans=0;
    memset(cnt,0,sizeof(cnt));
    FOR(i,1,cntq) {
        int ql=b[i].l, qr=b[i].r, qt=b[i].t, ind=b[i].ind;
        while (l<ql) del(l++);
        while (l>ql) add(--l);
```

```
        while (r<qr) add(++r);
        while (r>qr) del(r--);
        while (t<qt) update(++t,ql,qr);
        while (t>qt) update(t--,ql,qr);
        res[ind]=ans;
    }
    FOR(i,1,cntq) {
        printf("%d\n", res[i]);
    }

    return 0;
}
```

SP10707 COT2 - Count on a tree II

树上莫队模板树上两个结点路径不同结点权值个数

树上两结点路径转区间方法：先生成欧拉序——dfs时候进入是记录一下时间戳，出来的时候记录一下时间戳，并按时间戳顺序记录结点编号，共2*n长度的序列。对于结点u,v，如果in[u]¿in[v]，交换u和v；如果lca(u,v)==u，直接询问欧拉序的区间[in(u),out(v)]；否则，询问区间(out(u),in(v))，并且加上一个点lca(u,v)int to[MAXN],nxt[MAXN],f[MAXN],tot;

```
void add(int u, int v) {
    to[tot]=v;
    nxt[tot]=f[u];
    f[u]=tot++;
}
int n,m,a[MAXN];
int sz[MAXN],d[MAXN],fa[MAXN],son[MAXN],top[MAXN];
int b[MAXN],L[MAXN],R[MAXN],clk;
void prelude() {
    int cnt=0;
    unordered_map<int,int> mp;
    FOR(i,1,n) {
        if (!mp.count(a[i])) mp[a[i]]=++cnt;
    }
    FOR(i,1,n) a[i]=mp[a[i]];
}
void dfs(int u) {
    b[++clk]=u;
    L[u]=clk;
    sz[u]=1; d[u]=d[fa[u]]+1; son[u]=0;
    for (int i=f[u]; ~i; i=nxt[i]) {
        int v=to[i];
        if (v!=fa[u]) {
            fa[v]=u; dfs(v);
            sz[u]+=sz[v];
            if (sz[v]>sz[son[u]]) son[u]=v;
        }
    }
    b[++clk]=u;
    R[u]=clk;
}
void dfs(int u, int tp) {
```

```
    top[u]=tp;
    if (son[u]) dfs(son[u],tp);
    for (int i=f[u]; ~i; i=nxt[i]) {
        int v=to[i];
        if (v!=son[u] && v!=fa[u]) dfs(v,v);
    }
}
int LCA(int x, int y) {
    while (top[x]!=top[y]) d[top[x]]>=d[top[y]]?x=fa[top[x]]:y=fa[top[y]];
    return d[x]>=d[y]?y:x;
}
void init() {
    tot=d[0]=fa[1]=clk=0;
    memset(f,-1,sizeof(f));
}
int belong[MAXN],res[MAXN],cnt[MAXN],vcnt[MAXN],bsz,ans;
struct query {
    int l,r,lca,ind;
} q[MAXN];
bool cmp(query& a, query& b) {
    if (belong[a.l]!=belong[b.l]) return belong[a.l]<belong[b.l];
    return (belong[a.l]&1)?(a.r>b.r):(a.r<b.r);
}
void add(int u) {
    if (cnt[u]++==0) {
        if (vcnt[a[u]]++==0) ans++;
    } else {
        if (--vcnt[a[u]]==0) ans--;
    }
}
void del(int u) {
    if (--cnt[u]==0) {
        if (--vcnt[a[u]]==0) ans--;
    } else {
        if (vcnt[a[u]]++==0) ans++;
    }
}

int main()
{

    while (scanf("%d%d", &n,&m)==2) {
        init();
        FOR(i,1,n) scanf("%d", &a[i]);
        REP(i,n-1) {
            int u,v; scanf("%d%d", &u,&v);
            add(u,v);
            add(v,u);
        }
        prelude();
        dfs(1);
        dfs(1,1);
        FOR(i,1,m) {
            int u,v; scanf("%d%d", &u,&v);
            if (L[u]>L[v]) swap(u,v);
```

```
            int lca=LCA(u,v);
            if (lca==u) q[i]={L[u],L[v],0,i};
            else q[i]={R[u],L[v],lca,i};
        }
        bsz=sqrt(2*n);
        FOR(i,1,2*n) belong[i]=i/bsz;
        sort(q+1,q+1+m,cmp);
        int l=1,r=0;
        ans=0;
        memset(cnt,0,sizeof(cnt));
        memset(vcnt,0,sizeof(vcnt));
        FOR(i,1,m) {
            int ql=q[i].l, qr=q[i].r, lca=q[i].lca, ind=q[i].ind;
            while (l<ql) del(b[l++]);
            while (l>ql) add(b[--l]);
            while (r<qr) add(b[++r]);
            while (r>qr) del(b[r--]);
            if (lca) add(lca);
            res[ind]=ans;
            if (lca) del(lca);
        }
        FOR(i,1,m) {
            printf("%d\n", res[i]);
        }
    }

    return 0;
}
```

AT1219 歴史の研究

回滚莫队模板

询问区间max(某数x乘上该数的出现次数)

对于添加操作容易实现，但删除操作不太可能，可以使用回滚莫队。

区别普通莫队：1.需要维护多一个块区间的最右端点br数组；2.记录上一个块的bid，每次遍历到一个新的块，把cnt数组清空（这里复杂度O(n)，共需清空sqrt(n)次），把左右指针指向当前块的最右端。3.如果右指针小于当前块最右端，添加元素，保存这时的答案（后面要恢复用），枚举左边部分的元素，逐个添加（注意右端点位置），更新答案，然后逐个删除，复原答案。 @int n,m,a[MAXN]

```
vector<LL> val;
struct query {
    int l,r,ind;
} q[MAXN];
int bsz,cnt[MAXN],belong[MAXN],br[MAXN];
LL ans,res[MAXN];
void prelude() {
    val.clear();
    unordered_map<int,int> mp;
    FOR(i,1,n) {
        if (!mp.count(a[i])) {
            val.pb(a[i]);
```

```
            mp[a[i]]=SZ(val)-1;
        }
        a[i]=mp[a[i]];
    }
    bsz=sqrt(n);
    FOR(i,1,n) {
        belong[i]=i/bsz;
        br[i/bsz]=i;
    }
}
bool cmp(query& a, query& b) {
    if (belong[a.l]!=belong[b.l]) return belong[a.l]<belong[b.l];
    return a.r<b.r;
}
void add(int x) {
    cnt[x]++;
    ans=max(ans,cnt[x]*val[x]);
}
void del(int x) {
    cnt[x]--;
}
void mt() {
    sort(q+1,q+1+m,cmp);
    int pre=-1;
    ans=0;
    int l,r;
    FOR(i,1,m) {
        int ql=q[i].l, qr=q[i].r, ind=q[i].ind;
        int bid=belong[ql];
        if (bid!=pre) {
            memset(cnt,0,sizeof(cnt[0])*SZ(val));
            ans=0;
            r=br[bid];
            l=br[bid];
            pre=bid;
        }
        while (r<qr) add(a[++r]);
        LL pans=ans;
        int ed=min(l,qr);
        FOR(j,ql,ed) add(a[j]);
        res[ind]=ans;
        FOR(j,ql,ed) del(a[j]);
        ans=pans;
    }
}

int main()
{

    while (scanf("%d%d", &n,&m)==2) {
        FOR(i,1,n) scanf("%d", &a[i]);
        FOR(i,1,m) {
            int l,r; scanf("%d%d", &l,&r);
            q[i]={l,r,i};
        }
```

```
        prelude();
        mt();
        FOR(i,1,m) {
            printf("%lld\n", res[i]);
        }
    }

    return 0;
}
```