ACM板子合集

字符串

KMP

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

using namespace std;

const int N=1e6+10;
const int INF=0x3f3f3f3f;

int a[N],b[N];
int n,m;

inline int read(){
    int res=0,sign=1;char ch=getchar();
    for(;ch>'9'||ch<'0';ch=getchar()){
        if(ch=='-')sign=-sign;
    }
    for(;'0'<=ch&&ch<='9';ch=getchar()){
        res=(res<<1)+(res<<3)+(ch^'0');
    }
    return res*sign;
}

int nxt[N];

void getnxt(){
    for(int i=2,j;i<=m;++i){
        for(j=nxt[i-1];j&&b[i]!=b[j+1];)j=nxt[j];
        if(b[i]==b[j+1])++j;
        nxt[i]=j;
    }
}

void kmp(){
    int flag=0;
    for(int i=1,j=0;i<=n;++i){
        while(j&&a[i]!=b[j+1])j=nxt[j];
        if(a[i]==b[j+1])++j;
        if(j==m){
            cout<<i-m+1<<'\n';
            flag=1;
            break;
        }
    }
    if(!flag)cout<<-1<<'\n';
}

void solve(){
    n=read(),m=read();
```

```
    for(int i=1;i<=n;++i)a[i]=read();
    for(int i=1;i<=m;++i)b[i]=read();
    getnxt();
    kmp();
}

int main(){
    int t;cin>>t;
    while(t--)solve();
    return 0;
}
```

Manacher

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

using namespace std;

const int N=1.1e7+10;

char s[N<<1],dat[N<<1];

int len,cnt;

int p[N<<1],ans;

int main(){
    scanf("%s",s+1);len=strlen(s+1);
    dat[0]='~',dat[cnt=1]='#';
    for(int i=1;i<=len;++i){
        dat[++cnt]=s[i],dat[++cnt]='#';
    }
    for(int i=1,mid=0,r=0;i<=cnt;++i){
        if(i<=r)p[i]=min(p[(mid<<1)-i],r-i+1);
        else p[i]=1;
        while(dat[i-p[i]]==dat[i+p[i]])++p[i];
        if(i+p[i]>r)r=i+p[i]-1,mid=i;
        ans=max(ans,p[i]);
    }
    printf("%d\n",ans-1);
    return 0;
}
```

Trie

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

using namespace std;

const int MAXN=1e4+2;

struct trie{
    int nxt[MAXN<<5][30],cnt;
    bool exist[MAXN<<5],vis[MAXN<<5];
```

```
    void insert(char s[],int len){
        int p=0;
        for(int i=0;i<len;++i){
            int c=s[i]-'a';
            if(!nxt[p][c])nxt[p][c]=++cnt;
            p=nxt[p][c];
        }
        exist[p]=1;
    }

    int find(char s[],int len){
        int p=0;
        for(int i=0;i<len;++i){
            int c=s[i]-'a';
            if(!nxt[p][c])return 0;
            p=nxt[p][c];
        }
        if(exist[p]){
            if(!vis[p])return vis[p]=true,1;
            else return 2;
        }else return 0;
    }
};

int n,m;
trie t;
string tmp;
char _tmp[60];

int main(){
    cin>>n;
    for(int i=1;i<=n;++i){
        cin>>tmp;
        for(int i=0;i<tmp.size();++i)_tmp[i]=tmp[i];
        t.insert(_tmp,tmp.size());
    }
    cin>>m;
    for(int i=1;i<=m;++i){
        cin>>tmp;
        for(int i=0;i<tmp.size();++i)_tmp[i]=tmp[i];
        int op=t.find(_tmp,tmp.size());
        if(op==0)cout<<"WRONG"<<'\n';
        else if(op==1)cout<<"OK"<<'\n';
        else if(op==2)cout<<"REPEAT"<<'\n';
    }
    return 0;
}
```

ACAM

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

using namespace std;

const int MAXN=1e6+2;
```

```
struct trie{
    int nxt[MAXN][30],cnt,end[MAXN],fail[MAXN];
    void insert(string s){
        int len=s.size();
        int p=0;
        for(int i=0;i<len;++i){
            int c=s[i]-'a';
            if(!nxt[p][c])nxt[p][c]=++cnt;
            p=nxt[p][c];
        }
        end[p]+=1;
    }

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

    int query(string s){
        int p=0,ret=0,len=s.size();
        for(int i=0,c;i<len;++i){
            c=s[i]-'a';
            p=nxt[p][c];
            for(int j=p;j&&end[j]!=-1;j=fail[j]){
                ret+=end[j],end[j]=-1;
            }
        }
        return ret;
    }
};

trie t;
int n;
string tmp;

int main(){
    cin>>n;
    for(int i=1;i<=n;++i){
        cin>>tmp;
        t.insert(tmp);
    }
    t.build();
    cin>>tmp;
    cout<<t.query(tmp)<<'\n';
    return 0;
```

```
}
```

ACAM + TOPO

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

using namespace std;

const int N=2e5+10;
const int PN=2e5+10;
const int TN=2e6+10;

struct aca_t{
    int n;
    char mods[N],txt[TN];
    int tot,vis[N],rev[N],deg[PN],ans;

    struct node_t{
        int son[27],fail,flag,ans;

        void init(){
            memset(son,0,sizeof son);
            fail=flag=0;
        }
    }trie[PN];

    void init(){
        for(int i=0;i<=tot;++i)trie[i].init();
        for(int i=1;i<=n;++i)vis[i]=0;
        tot=1;
        ans=0;
    }

    void insert(char* s,int len,int idx){
        int p=1;
        for(int i=1;i<=len;++i){
            int c=s[i]-'a';
            if(!trie[p].son[c])trie[p].son[c]=++tot;
            p=trie[p].son[c];
        }
        if(!trie[p].flag)trie[p].flag=idx;
        rev[idx]=trie[p].flag;
    }

    void getfail(){
        queue<int> q;
        for(int i=0;i<26;++i)trie[0].son[i]=1;
        q.push(1);
        trie[1].fail=0;
        while(!q.empty()){
            int u=q.front();q.pop();
            int Fail=trie[u].fail;
            for(int i=0;i<26;++i){
                int v=trie[u].son[i];
                if(v){
```

```
                    trie[v].fail=trie[Fail].son[i];
                    ++deg[trie[Fail].son[i]];
                    q.push(v);
                }else trie[u].son[i]=trie[Fail].son[i];
            }
        }
    }

    void query(char* s,int len){
        int p=1;
        for(int i=1;i<=len;++i){
            int c=s[i]-'a';
            p=trie[p].son[c];
            ++trie[p].ans;
        }
    }

    void topo(){
        queue<int> q;
        for(int i=1;i<=tot;++i){
            if(!deg[i])q.push(i);
        }
        while(!q.empty()){
            int u=q.front();q.pop();
            vis[trie[u].flag]=trie[u].ans;
            int v=trie[u].fail;
            trie[v].ans+=trie[u].ans;
            if(!(--deg[v]))q.push(v);
        }
    }

    void solve(){
        scanf("%d",&n);
        init();
        for(int i=1;i<=n;++i){
            scanf("%s",mods+1);
            int len=strlen(mods+1);
            insert(mods,len,i);
        }
        getfail();
        scanf("%s",txt+1);
        int len=strlen(txt+1);
        query(txt,len);
        topo();
        for(int i=1;i<=n;++i)printf("%d\n",vis[rev[i]]);
    }
}aca;

int main(){
    aca.solve();
    return 0;
}
```

数学

预处理 1~n 的因子 + 欧拉函数（$O(n \log n)$）

```
int phi[N];
vector<int> dv[N];

void init(){
    for(int i=1;i<N;++i)phi[i]=i;
    for(int i=1;i<N;++i){
        for(int j=i;j<N;j+=i){
            dv[j].push_back(i);
            if(j>i)phi[j]-=phi[i];
        }
    }
}
```

EXGCD

```
LL __exgcd(LL a,LL b,LL& x,LL& y){
    //矩阵法
    int x1=1,x2=0,x3=0,x4=1;
    while(b){
        LL c=a/b;
        tie(x1,x2,x3,x4,a,b)=make_tuple(x3,x4,x1-c*x3,x2-c*x4,b,a-c*b);
    }
    x=x1,y=x2;
    return a;
}
```

线性筛 + 求积性函数（例子为 μ，约数个数，约数和）

```
int prime[N],idx;
int vis[N];
int mu[N],smu[N];

void init(){
    mu[1]=1;
    for(int i=2;i<N;++i){
        if(!vis[i])prime[++idx]=i,mu[i]=-1;
        for(int j=1;j<=idx;++j){
            int p=prime[j];
            if(1ll*i*p>=N)break;
            vis[i*p]=1;
            if(i%p==0){
                // p一定为i的最小质因子
                mu[i*p]=0;
                break;
            }
            mu[i*p]=-mu[i];
        }
    }
    for(int i=1;i<N;++i){
```

```
        smu[i]=smu[i-1]+mu[i];
    }
}
```

```
int prime[N],idx,vis[N],g[N],f[N];
// g约数个数，f约数和

void init(){
    g[1]=1,f[1]=1;
    for(int i=2;i<N;++i){
        if(!vis[i])prime[++idx]=i,g[i]=2,f[i]=i+1;
        for(int j=1;j<=idx;++j){
            int p=prime[j];
            if(1ll*i*p>=N)break;
            vis[i*p]=1;
            if(i%p==0){
                g[i*p]=2*g[i]-g[i/p];
                f[i*p]=(p+1)*f[i]-p*f[i/p];
                break;
            }
            g[i*p]=g[i]*g[p];
            f[i*p]=f[i]*f[p];
        }
    }
}
```

推导(对于约数个数的部分):

> 当$i \mod p == 0, i = a * p^c$, 其中$gcd(a,p) = 1$(互质), $g(i*p) = g(a) * g(p^{c+1})$, $g(i) = g(a) * g(p^c), g(i/p) = g(a) * g(p^{c-1})$, 解得$g(i*p) = 2 * g(i) - g(i/p)$
>
> 当$i \mod p \neq 0, g(i*p) = g(i) * g(p)$
>
> 总体思路就是按照积性函数的性质去求解递推式

数论分块

举个例子

$ans = \sum_{i=1}^{n} \sum_{j=1}^{m} a[i] \times \lfloor n/i \rfloor \times \lfloor m/i \rfloor$

sum数组是a的前缀和

j代表的是n/i和m/i的值不发生改变的区间右界，i则是左界

```
long long getans(int n,int m){
    long long res=0;
    for(int i=1,j;i<=min(n,m);i=j+1){
        j=min((n/(n/i)),(m/(m/i)));
        res+=1ll*(smu[j]-smu[i-1])*(n/i)*(m/i);
    }
    return res;
}
```

莫比乌斯

$\epsilon = \mu * 1$

$id = \varphi * 1$

$\varphi = \mu * id$

$if\ f = g * 1, then\ g = f * \mu$

例子求

$\sum_{i=1}^{n} \sum_{j=1}^{m} [gcd(i,j) = k]$

化简后为

$\sum_{d=1}^{min(\lfloor n/k \rfloor, \lfloor m/k \rfloor)} \mu(d) \lfloor n/kd \rfloor \lfloor m/kd \rfloor$

逆元

预处理

```
inv[1]=1;
for(int i=2;i<=n;++i){
    inv[i]=((-inv[p%i]*(p/i))%p+p)%p;
}
```

原理

> $p = i \times a + b$
>
> $i^{-1} = p - a \times b^{-1}(\mod p)$

直接算

p是质数才能按下面这种方法算，费马小定理

```
inv[n]=qpow(n,p-2,p);
```

质数模意义下的组合数

```
const int MAXN=1e6+2;
const int mod=1e9+7;

int n,m,fac[MAXN],inv[MAXN];

int qpow(int a,int b){
    int ret=1;
    for(;b;b>>=1,a=1ll*a*a%mod)if(b&1){
        ret=1ll*ret*a%mod;
    }
    return ret%mod;
}

void pre(){
    fac[0]=1,inv[0]=1;
```

```
    for(int i=1;i<=MAXN-1;++i)fac[i]=1ll*fac[i-1]*i%mod,inv[i]=qpow(fac[i],mod-
2);
}

int binom(int n,int k){
    return 1ll*fac[n]*inv[n-k]%mod*inv[k]%mod;
}
```

质因子分解

Pollard Rho

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

using namespace std;

#define LL long long

const int MAXN=100;

LL gcd(LL a,LL b){
    if(!b)return a;
    return gcd(b,a%b);
}

LL qpow(LL a,LL b,LL mod){
    LL res=1,base=a%mod;
    for(;b;b>>=1,base=(__int128_t)base*base%mod)if(b&1){
        res=(__int128_t)res*base%mod;
    }
    return res;
}

LL MR(LL p){
    if(p<2)return 0;
    if(p==2)return 1;
    if(p==3)return 1;
    LL d=p-1,r=0;
    while(!(d&1))d>>=1,++r;
    for(LL k=1;k<=10;++k){
        LL a=rand()%(p-2)+2;
        LL x=qpow(a,d,p);
        if(x==1||x==p-1)continue;
        for(int i=1;i<=r-1;++i){
            x=(__int128_t)x*x%p;
            if(x==p-1)break;
        }
        if(x!=p-1)return 0;
    }
    return 1;
}

LL PR(LL x){
    LL s=0,t=0;
    LL c=1ll*rand()%(x-1)+1;
```

```
    int step=0,goal=1;
    LL val=1;
    for(goal=1;;goal<<=1,s=t,val=1){
        for(step=1;step<=goal;++step){
            t=((__int128_t)t*t+c)%x;
            val=(__int128_t)val*abs(t-s)%x;
            if((step%127)==0){
                LL d=gcd(val,x);
                if(d>1)return d;
            }
        }
        LL d=gcd(val,x);
        if(d>1)return d;
    }
}

// P4718

// int t;
// LL maxfac,n;

// void fac(LL x){
//     if(x<=maxfac||x<2)return;
//     if(MR(x)){
//         maxfac=max(maxfac,x);
//         return;
//     }
//     LL p=x;
//     while(p>=x)p=PR(x);
//     while((x%p)==0)x/=p;
//     fac(x),fac(p);
// }

// int main(){
//     cin>>t;
//     for(int e=1;e<=t;++e){
//         srand((unsigned)time(nullptr));
//         maxfac=0;
//         cin>>n;
//         fac(n);
//         if(maxfac==n)cout<<"Prime"<<'\n';
//         else cout<<maxfac<<'\n';
//     }
//     return 0;
// }

LL fac[MAXN],cnt[MAXN],tot,len,ufac[MAXN];

void divide(LL n){
    if(n<2)return;
    if(MR(n)){
        fac[++tot]=n;
        return;
    }
    LL d=n;
    while(d>=n)d=PR(n);
```

```
    divide(d);
    divide(n/d);
}

void brk(int n){
    memset(cnt,0,sizeof cnt);
    tot=0;
    len=0;
    divide(n);
    sort(fac+1,fac+tot+1);
    for(int i=1;i<=tot;++i){
        if(fac[i]!=fac[i-1])ufac[++len]=fac[i];
        ++cnt[len];
    }
}

int main(){
    srand((unsigned)time(nullptr));
    brk(100000000);
    for(int i=1;i<=len;++i){
        cout<<ufac[i]<<' '<<cnt[i]<<'\n';
    }
    return 0;
}
```

数据结构

Splay

1. 插入 x 数
2. 删除 x 数(若有多个相同的数，应只删除一个)
3. 查询 x 数的排名(排名定义为比当前数小的数的个数 $+1$)
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)

```
#include <iostream>

using namespace std;
const int MAXN=1e5+10;

int read(){
    int res=0,sign=1;char ch=getchar();
    for(;ch<'0'||ch>'9';ch=getchar())if(ch=='-')sign=-sign;
    for(;ch>='0'&&ch<='9';ch=getchar())res=(res<<3)+(res<<1)+(ch^'0');
    return res*sign;
}

int rt,tot,fa[MAXN],ch[MAXN][2],val[MAXN],cnt[MAXN],sz[MAXN];

void maintain(int x){sz[x]=sz[ch[x][0]]+sz[ch[x][1]]+cnt[x];}
```

```
bool get(int x){return x==ch[fa[x]][1];}

void clear(int x){fa[x]=ch[x][0]=ch[x][1]=val[x]=cnt[x]=sz[x]=0;}

void rotate(int x){
    int y=fa[x],z=fa[y],chk=get(x);

    ch[y][chk]=ch[x][chk^1];
    if(ch[x][chk^1])fa[ch[x][chk^1]]=y;

    ch[x][chk^1]=y;
    fa[y]=x;

    if(z)ch[z][y==ch[z][1]]=x;
    fa[x]=z;

    maintain(y);
    maintain(x);
}

void splay(int x){
    for(int f=fa[x];f=fa[x],f;rotate(x)){
        if(fa[f])rotate(get(x)==get(f)?f:x);
    }
    rt=x;
}

void ins(int k){
    if(!rt){
        val[++tot]=k;
        cnt[tot]++;
        rt=tot;
        maintain(rt);
        return;
    }
    int cur=rt,f=0;
    while(1){
        if(val[cur]==k){
            cnt[cur]++;
            maintain(cur);
            maintain(f);
            splay(cur);
            break;
        }
        f=cur;
        cur=ch[cur][val[cur]<k];
        if(!cur){
            val[++tot]=k;
            cnt[tot]++;
            fa[tot]=f;
            ch[f][val[f]<k]=tot;
            maintain(tot);
            maintain(f);
            splay(tot);
            break;
        }
```

```
    }
}

int rk(int k){
    int res=0,cur=rt;
    while(1){
        if(val[cur]>k){
            cur=ch[cur][0];
        }else{
            res+=sz[ch[cur][0]];
            if(val[cur]==k){
                splay(cur);
                return res+1;
            }
            res+=cnt[cur];
            cur=ch[cur][1];
        }
    }
}

int kth(int k){
    int cur=rt;
    while(1){
        if(ch[cur][0]&&k<=sz[ch[cur][0]]){
            cur=ch[cur][0];
        }else{
            k-=sz[ch[cur][0]]+cnt[cur];
            if(k<=0){
                splay(cur);
                return val[cur];
            }
            cur=ch[cur][1];
        }
    }
}

int pre(){
    int cur=ch[rt][0];
    if(!cur)return cur;
    while(ch[cur][1])cur=ch[cur][1];
    splay(cur);
    return cur;
}

int nxt(){
    int cur=ch[rt][1];
    if(!cur)return cur;
    while(ch[cur][0])cur=ch[cur][0];
    splay(cur);
    return cur;
}

void del(int k){
    rk(k);
    if(cnt[rt]>1){
        cnt[rt]--;
```

```
        maintain(rt);
        return;
    }
    if(!ch[rt][0]&&!ch[rt][1]){
        clear(rt);
        rt=0;
        return;
    }
    if(!ch[rt][0]){
        int cur=rt;
        rt=ch[rt][1];
        fa[rt]=0;
        clear(cur);
        return;
    }
    if(!ch[rt][1]){
        int cur=rt;
        rt=ch[rt][0];
        fa[rt]=0;
        clear(cur);
        return;
    }
    int cur=rt,x=pre();
    fa[ch[cur][1]]=x;
    ch[x][1]=ch[cur][1];
    clear(cur);
    maintain(rt);
}

int n;

int main(){
    n=read();
    for(int i=1;i<=n;++i){
        int op=read();
        if(op==1){
            int x=read();
            ins(x);
        }else if(op==2){
            int x=read();
            del(x);
        }else if(op==3){
            int x=read();
            ins(x);
            printf("%d\n",rk(x));
            del(x);
        }else if(op==4){
            int x=read();
            printf("%d\n",kth(x));
        }else if(op==5){
            int x=read();
            ins(x);
            rk(x);
            printf("%d\n",val[pre()]);
            del(x);
        }else if(op==6){
```

```
            int x=read();
            ins(x);
            rk(x);
            printf("%d\n",val[nxt()]);
            del(x);
        }
    }
    return 0;
}
```

ST表

```
void pre(){
    //对数取整
    _log_2[1]=0;
    _log_2[2]=1;
    for(int i=3;i<MAXN;++i){
        _log_2[i]=_log_2[i/2]+1;
    }
}

void _pre(){
    for(int k=1;k<=LOGN;++k){
        for(int i=1;i+(1<<k)-1<=n;++i){
            f[i][k]=max(f[i][k-1],f[i+(1<<(k-1))][k-1]);
        }
    }
}

int getmax(int l,int r){
    int ret;
    int s=_log_2[r-l+1];
    ret=max(f[l][s],f[r-(1<<s)+1][s]);
    return ret;
}

int main(){
    n=read(),m=read();
    for(int i=1;i<=n;++i)f[i][0]=read();
    pre();
    _pre();
    for(int i=1;i<=m;++i){
        int l,r;l=read(),r=read();
        printf("%d\n",getmax(l,r));
    }
    return 0;
}
```

线段树染色

贴报纸问题

```
#include <iostream>
#include <cstring>
```

```cpp
using namespace std;

const int MAXN=1e5+2;
const int INF=0x7fffffff;

struct node_t{
    int ls,rs;
    int tag;
}d[MAXN<<5];

int rt,cnt,vis[MAXN];

#define MID int m=s+((t-s)>>1)
#define lson d[p].ls
#define rson d[p].rs

inline void pushdown(int p,int s,int t){
    MID;
    if(s!=t&&d[p].tag){
        if(!lson)lson=++cnt;
        if(!rson)rson=++cnt;
        d[lson].tag=d[p].tag,d[rson].tag=d[p].tag;
        d[p].tag=0;
    }
}

void modify(int& p,int s,int t,int l,int r,int c){
    if(!p)p=++cnt;
    if(l<=s&&t<=r){
        d[p].tag=c;
        return;
    }
    MID;
    pushdown(p,s,t);
    if(l<=m)modify(lson,s,m,l,r,c);
    if(r>m)modify(rson,m+1,t,l,r,c);
}

int query(int p,int s,int t,int l,int r){
    if(!p)return 0;
    if(d[p].tag){
        if(vis[d[p].tag])return 0;
        else return vis[d[p].tag]=1;
    }
    MID;
    pushdown(p,s,t);
    int sum=0;
    if(l<=m)sum+=query(lson,s,m,l,r);
    if(r>m)sum+=query(rson,m+1,t,l,r);
    return sum;
}

int n;

void solve(){
```

```
    rt=cnt=0;
    memset(d,0,sizeof d);
    memset(vis,0,sizeof vis);
    cin>>n;
    for(int i=1,l,r;i<=n;++i){
        cin>>l>>r;
        modify(rt,1,INF,l,r,i);
    }
    cout<<query(rt,1,INF,1,INF)<<'\n';
}

int main(){
    int t;cin>>t;
    while(t--){
        solve();
    }
    return 0;
```

扫描线

面积

线段树中闭区间$[l,r]$表示的实际数轴上$[l,r+1]$的线段，数轴上每一个小区间$[x,x+1]$表示线段树的数组中下标为x的叶子节点

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

const int MAXN=1e5+2;
const int INF=0x7fffffff;

struct line_t{
    int l,r,h,mark;

    line_t(int _l,int _r,int _h,int _mark):l(_l),r(_r),h(_h),mark(_mark){}

    bool operator<(const line_t& o)const{
        return h<o.h;
    }
};

vector<line_t> lines;

int n,cnt,rt;

#define ls d[p].lson
#define rs d[p].rson
#define MID int m=s+((t-s)>>1)

struct node_t{
    int lson,rson,tag;
    int len;
}d[MAXN<<7];
```

```
void pushup(int p,int s,int t){
    if(d[p].tag){
        d[p].len=t-s+1;
    }else d[p].len=d[ls].len+d[rs].len;
}

void modify(int& p,int s,int t,int l,int r,int c){
    if(!p)p=++cnt;
    if(l<=s&&t<=r){
        d[p].tag+=c;
        pushup(p,s,t);
        return;
    }
    MID;
    if(l<=m)modify(ls,s,m,l,r,c);
    if(r>m)modify(rs,m+1,t,l,r,c);
    pushup(p,s,t);
}

int main(){
    cin>>n;
    for(int i=1,x,y,_x,_y;i<=n;++i){
        cin>>x>>y>>_x>>_y;
        lines.emplace_back(x,_x,y,1);
        lines.emplace_back(x,_x,_y,-1);
    }
    sort(lines.begin(),lines.end());

    LL ans=0;

    for(int i=0;i<lines.size()-1;++i){
        modify(rt,0,INF,lines[i].l,lines[i].r-1,lines[i].mark);
        ans+=1ll*d[1].len*(lines[i+1].h-lines[i].h);
    }cout<<ans<<'\n';
    return 0;
}
```

周长

```
#include <iostream>
#include <cstdio>
#include <vector>
#include <algorithm>

#define LL long long
using namespace std;

const int MAXN=5e3+2;
const int INF=0x7fffffff;

struct line_t{
    int l,r,h,mark;

    line_t(int _l,int _r,int _h,int _mark):l(_l),r(_r),h(_h),mark(_mark){}
```

```
    bool operator<(const line_t& o)const{
        if(h==o.h)return mark>o.mark;
        return h<o.h;
    }
};

vector<line_t> lines;

int n,cnt,rt;

#define ls d[p].lson
#define rs d[p].rson
#define MID int m=1ll*s+((1ll*t-s)>>1)

struct node_t{
    int lson,rson,tag;//左儿子，右儿子，区间是否被线段完全覆盖
    int c;// 区间线段个数
    bool lc,rc;//区间左右端点是否被覆盖
    int len;//区间内的线段总长度
}d[MAXN<<7];

void pushup(int p,int s,int t){
    if(d[p].tag){
        d[p].len=t-s+1;
        d[p].lc=d[p].rc=true;
        d[p].c=1;
    }else{
        d[p].len=d[ls].len+d[rs].len;
        d[p].lc=d[ls].lc,d[p].rc=d[rs].rc;
        d[p].c=d[ls].c+d[rs].c;
        if(d[ls].rc&&d[rs].lc){
            d[p].c-=1;
        }
    }
}

void modify(int& p,int s,int t,int l,int r,int c){
    if(!p)p=++cnt;
    if(l<=s&&t<=r){
        d[p].tag+=c;
        pushup(p,s,t);
        return;
    }
    MID;
    if(l<=m)modify(ls,s,m,l,r,c);
    if(r>m)modify(rs,m+1,t,l,r,c);
    pushup(p,s,t);
}

int main(){
    cin>>n;
    for(int i=1,x,y,_x,_y;i<=n;++i){
        cin>>x>>y>>_x>>_y;
        // lines.emplace_back(x,_x,y,1);
        // lines.emplace_back(x,_x,_y,-1);
        lines.push_back({x,_x,y,1});
```

```
        lines.push_back({x,_x,_y,-1});
    }
    sort(lines.begin(),lines.end());

    LL ans=0;
    int pre=0;

    for(int i=0;i<lines.size()-1;++i){
        modify(rt,-INF,INF,lines[i].l,lines[i].r-1,lines[i].mark);
        ans+=abs(d[1].len-pre);
        ans+=2*d[1].c*(lines[i+1].h-lines[i].h);
        pre=d[1].len;
    }

    ans+=lines.back().r-lines.back().l;

    cout<<ans<<'\n';
    return 0;
}
```

主席树

主席树，全名可持久化线段树，又名hjt tree，是一种保留了多个历史版本的权值线段树。插入节点的时候去和上一个版本的权值线段树做连接，计算前缀。询问的时候带着两个版本一起跳儿子，统计前缀的差来得到区间信息。

区间第k大

值域到了`int`，先做离散化。

这里先算出左儿子区间上的点的个数 x，如果排名 k 小于 x，那么就往做左儿子跳，否则往右儿子跳找排名为 k-x 的节点，和平衡树的查找很像。

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

using namespace std;

int read(){
    int res=0,sign=1;char ch=getchar();
    for(;ch<'0'||ch>'9';ch=getchar())if(ch=='-')sign=-sign;
    for(;ch>='0'&&ch<='9';ch=getchar())res=(res<<3)+(res<<1)+(ch^'0');
    return res*sign;
}

const int N=2e5+10;

int n,m;
int tot,sum[N<<5],rt[N],ls[N<<5],rs[N<<5];

int a[N],idx[N],len;

#define MID int m=s+((t-s)>>1)

int build(int s,int t){
    int p=++tot;
```

```
    if(s==t)return p;
    MID;
    ls[p]=build(s,m);
    rs[p]=build(m+1,t);
    return p;
}

int ins(int s,int t,int x,int _p){
    int p=++tot;
    ls[p]=ls[_p],rs[p]=rs[_p],sum[p]=sum[_p]+1;
    if(s==t)return p;
    MID;
    if(x<=m)ls[p]=ins(s,m,x,ls[p]);
    else rs[p]=ins(m+1,t,x,rs[p]);
    return p;
}

int query(int s,int t,int u,int v,int k){
    int x=sum[ls[v]]-sum[ls[u]];
    if(s==t)return s;
    MID;
    if(k<=x)return query(s,m,ls[u],ls[v],k);
    else return query(m+1,t,rs[u],rs[v],k-x);
}

int getid(int x){
    return lower_bound(idx+1,idx+1+len,x)-idx;
}

void init(){
    for(int i=1;i<=n;++i)idx[i]=a[i];
    sort(idx+1,idx+1+n);
    len=unique(idx+1,idx+1+n)-idx-1;
    rt[0]=build(1,len);
    for(int i=1;i<=n;++i){
        rt[i]=ins(1,len,getid(a[i]),rt[i-1]);
    }
}

void solve(){
    for(int i=1;i<=m;++i){
        int l=read(),r=read(),k=read();
        int ans=idx[query(1,len,rt[l-1],rt[r],k)];
        printf("%d\n",ans);
    }
}

int main(){
    n=read(),m=read();
    for(int i=1;i<=n;++i){
        a[i]=read();
    }
    init();
    solve();
    return 0;
}
```

区间[l, r]中属于某一个值域[L, R]中的点的个数

一个序列 $a_i\ (i=1,2,\ldots,n)$，在 $a_l, a_{l+1}, \ldots, a_r$ 中有多少个 a_i 满足 $L <= a_i <= R$？

带着两个版本的树跳，然后统计区间中满足的点的个数就好了。

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

using namespace std;

const int N=1e5+10;

int read(){
    int res=0,sign=1;
    char ch=getchar();
    for(;ch<'0'||ch>'9';ch=getchar())if(ch=='-'){sign=-sign;}
    for(;ch>='0'&&ch<='9';ch=getchar()){res=(res<<3)+(res<<1)+(ch^'0');}
    return res*sign;
}

int n,q;

vector<int> G[N];
int ver[N];
int tin[N],tout[N],timer;

void dfs(int u,int fno){
    tin[u]=++timer;
    for(auto&& v:G[u])if(v!=fno){
        dfs(v,u);
    }
    tout[u]=timer;
}

int ls[N<<5],rs[N<<5],rt[N<<5],sum[N<<5],tot;

#define MID int m=s+((t-s)>>1)

int ins(int s,int t,int x,int _p){
    int p=++tot;
    ls[p]=ls[_p],rs[p]=rs[_p],sum[p]=sum[_p]+1;
    if(s==t)return p;
    MID;
    if(x<=m)ls[p]=ins(s,m,x,ls[p]);
    else rs[p]=ins(m+1,t,x,rs[p]);
    return p;
}

int query(int s,int t,int u,int v,int l,int r){
    if(l<=s&&t<=r)return sum[v]-sum[u];
    MID;
    int res=0;
    if(l<=m)res+=query(s,m,ls[u],ls[v],l,r);
    if(r>m)res+=query(m+1,t,rs[u],rs[v],l,r);
```

```
    return res;
}

void solve(){
    n=read(),q=read();
    for(int i=1;i<=n;++i)G[i].clear();
    tot=0,timer=0;
    for(int i=1,u,v;i<=n-1;++i){
        u=read(),v=read();
        G[u].push_back(v);
        G[v].push_back(u);
    }
    dfs(1,0);
    for(int i=1;i<=n;++i)ver[i]=read(),rt[i]=ins(1,n,tin[ver[i]],rt[i-1]);
    for(int _=1;_<=q;++_){
        int l=read(),r=read(),x=read();
        int L=tin[x],R=tout[x];
        int res=query(1,n,rt[l-1],rt[r],L,R);
        if(res>=1)puts("YES");
        else puts("NO");
    }
    puts("");
}

int main(){
    int t=read();
    while(t--)solve();
    return 0;
}
```

分块

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

using namespace std;

#define LL long long

int read(){
    int res=0,sign=1;char ch=getchar();
    for(;ch<'0'||ch>'9';ch=getchar())if(ch=='-')sign=-sign;
    for(;ch>='0'&&ch<='9';ch=getchar())res=(res<<3)+(res<<1)+(ch^'0');
    return res*sign;
}

const int N=5e4+10;

int n;

int id[N],len;
LL a[N],b[N],s[N];

void add(int l,int r,int c){
    int op=id[l],ed=id[r];
    if(op==ed){
```

```
        for(int i=l;i<=r;++i)a[i]+=c,s[op]+=c;
        return;
    }
    for(int i=l;id[i]==op;++i)a[i]+=c,s[op]+=c;
    for(int i=op+1;i<ed;++i)b[i]+=c,s[i]+=1ll*c*len;
    for(int i=r;id[i]==ed;--i)a[i]+=c,s[ed]+=c;
}

int query(int l,int r,int mod){
    int op=id[l],ed=id[r],res=0;
    if(op==ed){
        for(int i=l;i<=r;++i)res=(1ll*res+a[i]+b[op]+mod)%mod;
        return res;
    }
    for(int i=l;id[i]==op;++i)res=(1ll*res+a[i]+b[op]+mod)%mod;
    for(int i=op+1;i<ed;++i)res=(1ll*res+s[i]+mod)%mod;
    for(int i=r;id[i]==ed;--i)res=(1ll*res+a[i]+b[ed]+mod)%mod;
    return res;
}

int main(){
    n=read();len=sqrt(n);
    for(int i=1;i<=n;++i){
        a[i]=read();
        id[i]=(i-1)/len+1;
        s[id[i]]+=a[i];
    }
    for(int i=1,op,l,r,c;i<=n;++i){
        op=read(),l=read(),r=read(),c=read();
        if(op==0){
            add(l,r,c);
        }else{
            printf("%d\n",query(l,r,c+1));
        }
    }
    return 0;
}
```

图论

判断是否是一个点子树

通过$tin[v] >= tin[u] \&\& tin[v] <= tout[u]$判断

LCA

倍增版本

```
void dfs(int x,int fno){
    fa[x][0]=fno;
    dep[x]=dep[fno]+1;
    for(int i=1;i<=30;++i){
        fa[x][i]=fa[fa[x][i-1]][i-1];
    }
```

```
    for(auto&& v:p[x])if(v!=fno){
        dfs(v,x);
    }
}

int lca(int x,int y){
    if(dep[x]>dep[y])swap(x,y);
    for(int i=30;i>=0;--i)if(dep[fa[y][i]]>=dep[x]){
        y=fa[y][i];
    }
    if(x==y)return x;
    for(int i=30;i>=0;--i)if(fa[x][i]!=fa[y][i]){
        x=fa[x][i],y=fa[y][i];
    }
    return fa[x][0];
}
```

树剖版本

```
void dfs(int u,int fno){
    fa[u]=fno;
    siz[u]=1;
    dep[u]=dep[fno]+1;
    son[u]=-1;
    for(auto&& v:G[u])if(v!=fno){
        dfs(v,u);
        siz[u]+=siz[v];
        if(son[u]==-1||siz[son[u]]<siz[v])son[u]=v;
    }
}

void _dfs(int u,int ufno){
    top[u]=ufno;
    if(son[u]==-1)return;
    _dfs(son[u],ufno);
    for(auto&& v:G[u])if(v!=fa[u]&&v!=son[u]){
        _dfs(v,v);
    }
}

int lca(int u,int v){
    while(top[u]!=top[v]){
        if(dep[top[u]]<dep[top[v]])v=fa[top[v]];
        else u=fa[top[u]];
    }
    if(dep[u]<dep[v])return u;
    else return v;
}
```

最短路

SPFA

```
bool spfa(int n,int s){
    fill_n(dis+1,n,INF);
    dis[s]=0,isv[s]=1;
    q.push(s);
    while(!q.empty()){
        int u=q.front();q.pop();
        isv[u]=0;
        for(auto& [v,w]:p[u]){
            if(dis[v]>dis[u]+w){
                dis[v]=dis[u]+w;
                cnt[v]=cnt[u]+1;
                if(cnt[v]>=n)return false;
                if(!isv[v])q.push(v),isv[v]=1;
            }
        }
    }
    return true;
}
```

Dijstra

```
void dijkstra(int n,int s){
    fill_n(dis+1,n,INF);
    dis[s]=0;
    q.push({s,0});
    while(!q.empty()){
        int u=q.top().u;
        q.pop();
        if(vis[u])continue;
        vis[u]=1;
        for(auto& [v,w]:p[u]){
            if(dis[v]>dis[u]+w){
                dis[v]=dis[u]+w;
                q.push({v,dis[v]});
            }
        }
    }
}
```

差分约束

给出一组包含 m 个不等式，有 n 个未知数的形如：

$$\begin{cases} x_{c_1} - x_{c_1'} \le y_1 \\ x_{c_2} - x_{c_2'} \le y_2 \\ \cdots \\ x_{c_m} - x_{c_m'} \le y_m \end{cases}$$

的不等式组，求任意一组满足这个不等式组的解。

这个题的意思就是将c'向c建边，权值为y，看这个图有没有负环。用spfa算法处理即可。

MST

Kruskal贪心加边即可

次小生成树

方法是先找出最小生成树，建立最小生成树。然后倍增维护最小生成树的祖先，路径上的最大权值，次大权值。

最后在看所有的非最小生成树的边的两点在最小生成树中的路径上的最大值是多少，并且与未使用的该边替换，更新答案。如果最大值等于未使用的该边的权值，那么直接找次大值替换。

最后的答案等于所有替换方案的最小值。

```
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int MAXN=1e5+2;
const int MAXM=3e5+2;
const int INF=0x7fffffffffffffff;

struct Tr{
private:
    struct edge{
        int v,w;
    };
    vector<edge> p[MAXN];
    int fa[MAXN][31],dep[MAXN],m[MAXN][31],mm[MAXN][31];

public:
    void add(int u,int v,int w){
        p[u].push_back({v,w});
        p[v].push_back({u,w});
    }

    void dfs(int x,int fno){
        if(fno)dep[x]=dep[fno]+1,fa[x][0]=fno,mm[x][0]=-INF;

        for(int i=1;i<=30;++i){
            fa[x][i]=fa[fa[x][i-1]][i-1];
            int tmp[4]={m[x][i-1],m[fa[x][i-1]][i-1],
                        mm[x][i-1],mm[fa[x][i-1]][i-1]};
            sort(tmp,tmp+4);
            m[x][i]=tmp[3];
            int ptr=2;
            for(;ptr>=0&&tmp[ptr]==tmp[3];){--ptr;}
            mm[x][i]=ptr==-1?-INF:tmp[ptr];
        }

        for(auto& edge:p[x]){
            int v=edge.v,w=edge.w;
            if(v==fno)continue;
            m[v][0]=w;
            dfs(v,x);
        }
```

```
    }

    int lca(int x,int y){
        if(dep[x]>dep[y])swap(x,y);
        for(int i=30;i>=0;--i){
            if(dep[fa[y][i]]>=dep[x])y=fa[y][i];
        }
        if(x==y)return x;
        for(int i=30;i>=0;--i){
            if(fa[x][i]!=fa[y][i]){
                x=fa[x][i],y=fa[y][i];
            }
        }
        return fa[x][0];
    }

    int query(int x,int y,int w){
        int ret=-INF;
        for(int i=30;i>=0;--i){
            if(dep[fa[x][i]]>=dep[y]){
                if(w!=m[x][i])
                    ret=max(ret,m[x][i]);
                else
                    ret=max(ret,mm[x][i]);
                x=fa[x][i];
            }
        }
        return ret;
    }
};

Tr tr;

struct _edge_{
    int u,v,w;
    bool operator<(const _edge_& other)const{return w<other.w;}
};

vector<_edge_> edges;
int fa[MAXN],n,m,used[MAXM];

int find(int x){return fa[x]==x?x:fa[x]=find(fa[x]);}
void unite(int u,int v){int fu=find(u),fv=find(v);if(fu!=fv)fa[fu]=fv;}

int kruskal(){
    int ans=0,cnt=0;
    for(int i=1;i<=n;++i)fa[i]=i;
    sort(edges.begin(),edges.end());
    for(int i=0;i<edges.size();++i){
        _edge_ e=edges[i];
        int u=e.u,v=e.v,w=e.w;
        if(find(u)!=find(v)){
            unite(u,v);
            ans+=w,++cnt;
            used[i]=1;
            tr.add(u,v,w);
```

```
        }
        if(cnt==n-1)break;
    }
    return ans;
}

signed main(){
    cin>>n>>m;
    for(int i=1;i<=m;++i){
        int u,v,w;cin>>u>>v>>w;
        edges.push_back({u,v,w});
    }
    int ans_1=kruskal(),ans_2=INF;
    tr.dfs(1,0);
    for(int i=0;i<edges.size();++i){
        if(!used[i]){
            int u=edges[i].u,v=edges[i].v,w=edges[i].w;
            int _lca=tr.lca(u,v);
            int tmp1=tr.query(u,_lca,w);
            int tmp2=tr.query(v,_lca,w);
            int _m=max(tmp1,tmp2);
            if(_m>-INF)
            ans_2=min(ans_2,ans_1-_m+w);
        }
    }
    if(ans_2!=INF)cout<<ans_2<<endl;
    else cout<<-1<<endl;
}
```

Tarjan

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

using namespace std;

const int MAXN=1e4+2;

int n,m;
int dfn[MAXN],low[MAXN],cnt;
int bel[MAXN],ins[MAXN],siz[MAXN],idx;
int st[MAXN],top;

vector<int> p[MAXN];

void add(int u,int v){
    p[u].push_back(v);
}

void tarjan(int x){
    dfn[x]=low[x]=++cnt;
    ins[st[++top]=x]=1;
    for(auto v:p[x]){
        if(!dfn[v]){
            tarjan(v);
            low[x]=min(low[x],low[v]);
```

```
        }else if(ins[v]){
            low[x]=min(low[x],dfn[v]);
        }
    }
    if(low[x]==dfn[x]){
        int v;++idx;
        do{
            ins[v=st[top--]]=0;
            bel[v]=idx;
            ++siz[idx];
        }while(v!=x);
    }
}

int main(){
    cin>>n>>m;
    for(int i=1,u,v;i<=m;++i){
        cin>>u>>v;
        add(u,v);
    }
    for(int i=1;i<=n;++i){
        if(!dfn[i])tarjan(i);
    }
    int ans=0;
    for(int i=1;i<=idx;++i){
        //siz>1的强连通分量个数
        ans+=siz[i]>1;
    }cout<<ans<<'\n';
    return 0;
}
```

若是无向图，则上面的`else if`就直接`else`

网络流

最大流

EK算法

通过`bfs()`查找增广路来更新最大流

关键思想

`lst`：在`bfs`过程中找到的增广路的某个点，它上个点的索引

`pre`：在`bfs`过程中找到的增广路的某个点，它上个点与自己连成的边在上个点的邻接表中的索引

`flow`：某个点的通过的流量

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

const int MAXN=2e2+2;
const int INF=0x3f3f3f3f3f3f3f3f;
```

```cpp
struct edge_t{
    int v,w,rid;
};

int n,m,s,t;
vector<edge_t> G[MAXN];
bool vis[MAXN];
int lst[MAXN],pre[MAXN],flow[MAXN];

inline void add(int u,int v,int w){
    int id=G[u].size();
    int rid=G[v].size();
    G[u].push_back({v,w,rid});
    G[v].push_back({u,0,id});
}

bool bfs(){
    memset(vis,0,sizeof vis);
    queue<int> q;
    q.push(s);
    vis[s]=true;
    flow[s]=INF;
    while(!q.empty()){
        int u=q.front();q.pop();
        for(int i=0;i<G[u].size();++i){
            int v=G[u][i].v,w=G[u][i].w;
            if(!vis[v]&&w>0){
                vis[v]=true;
                flow[v]=min(flow[u],w);
                lst[v]=u;
                pre[v]=i;
                if(v==t)return true;
                q.push(v);
            }
        }
    }
    return false;
}

int ek(){
    int res=0;
    while(bfs()){
        res+=flow[t];
        for(int i=t;i!=s;i=lst[i]){
            int u=lst[i],v=i,rid=G[u][pre[v]].rid;
            G[u][pre[v]].w-=flow[t];
            G[v][rid].w+=flow[t];
        }
    }
    return res;
}

signed main(){
    cin>>n>>m>>s>>t;
    for(int i=1,u,v,w;i<=m;++i){
        cin>>u>>v>>w;
```

```
        add(u,v,w);
    }
    cout<<ek()<<'\n';
}
```

Dinic算法

通过`bfs()`分层后，考虑当前弧优化和剪枝的情况下做`dfs()`去更新最大流，下面`dfs()`有两种写法，推荐未注释写法。

关键思想

`dep`：`bfs`分层结果

`cur`：当前弧优化，就是将不需要搜的边排除掉所用到的

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

const int MAXN=2e2+2;
const long long INF=0x3f3f3f3f3f3f3f3f;

int n,m,s,t;
int dep[MAXN],cur[MAXN];

struct edge_t{
    int v,w,rid;
};

vector<edge_t> G[MAXN];

inline void add(int u,int v,int w){
    int id=G[u].size();
    int rid=G[v].size();
    G[u].push_back({v,w,rid});
    G[v].push_back({u,0,id});
}

bool bfs(){
    memset(dep,-1,sizeof dep);
    queue<int> q;
    q.push(s);dep[s]=0;
    while(!q.empty()){
        int u=q.front();q.pop();
        for(auto&& [v,w,rid]:G[u])if(dep[v]==-1&&w>0){
            dep[v]=dep[u]+1;
            q.push(v);
        }
    }
    return dep[t]!=-1;
}

// int dfs(int u,int flow){
//     if(u==t)return flow;
//     for(int i=cur[u];i<G[u].size();++i){
```

```
//         cur[u]=i;
//         int v=G[u][i].v,w=G[u][i].w,rid=G[u][i].rid,res;
//         if(dep[v]==dep[u]+1&&w>0){
//             if(res=dfs(v,min(w,flow))){
//                 G[u][i].w-=res;
//                 G[v][rid].w+=res;
//                 return res;
//             }else dep[v]=-1;
//         }
//     }
//     return 0;
// }

int dfs(int u,int flow){
    if(u==t)return flow;
    int ans=0,res;
    for(int i=cur[u];i<G[u].size()&&flow>0;++i){
        cur[u]=i;
        int v=G[u][i].v,w=G[u][i].w,rid=G[u][i].rid;
        if(dep[v]==dep[u]+1&&w>0){
            res=dfs(v,min(flow,w));
            if(res==0)dep[v]=-1;
            G[u][i].w-=res;
            G[v][rid].w+=res;
            ans+=res;
            flow-=res;
        }
    }
    return ans;
}

int dinic(){
    int ans=0,res;
    while(bfs()){
        memset(cur,0,sizeof cur);
        while(res=dfs(s,INF))ans+=res;
    }
    return ans;
}

signed main(){
    cin>>n>>m>>s>>t;
    for(int i=1,u,v,w;i<=m;++i){
        cin>>u>>v>>w;
        add(u,v,w);
    }
    cout<<dinic()<<'\n';
}
```

最小割

最小割等于最大流

最小费用最大流

最大流情况下的最小费用（因为最大流的分配情况不唯一），实现就是用`EK`算法的基础将`bfs`改成最短路`spfa`

关键思想（`EK`之外的）

`dis`：当前增广路的单位费用和

`vis`：`spfa`算法中的入队判断

反向建边的单位费用为相反数

```
#include <bits/stdc++.h>
//ek中bfs改成spfa
using namespace std;

const int MAXN=5e3+10;
const int INF=0x3f3f3f3f;

int n,m,s,t,minc,maxf;
int flow[MAXN],lst[MAXN],pre[MAXN];
int dis[MAXN],vis[MAXN];

struct edge_t{
    int v,w,c,rid;
};

vector<edge_t> G[MAXN];

inline void add(int u,int v,int w,int c){
    int id=G[u].size();
    int rid=G[v].size();
    G[u].push_back({v,w,c,rid});
    G[v].push_back({u,0,-c,id});
}

bool spfa(){
    queue<int> q;
    memset(dis,0x3f,sizeof dis);
    memset(vis,0,sizeof vis);
    q.push(s);dis[s]=0;vis[s]=1;flow[s]=INF;
    while(!q.empty()){
        int u=q.front();q.pop();vis[u]=0;
        for(int i=0;i<G[u].size();++i){
            int v=G[u][i].v,w=G[u][i].w,c=G[u][i].c,rid=G[u][i].rid;
            if(dis[v]>dis[u]+c&&w>0){
                dis[v]=dis[u]+c;
                flow[v]=min(flow[u],w);
                lst[v]=u;
                pre[v]=i;
                if(!vis[v])q.push(v),vis[v]=1;
            }
        }
    }
    if(dis[t]==INF)return false;
```

```
    return true;
}

void mcmf(){
    while(spfa()){
        maxf+=flow[t];
        minc+=dis[t]*flow[t];
        for(int i=t;i!=s;i=lst[i]){
            int u=lst[i],v=i,rid=G[u][pre[v]].rid;
            G[u][pre[v]].w-=flow[t];
            G[v][rid].w+=flow[t];
        }
    }
}

int main(){
    cin>>n>>m>>s>>t;
    for(int i=1,u,v,w,c;i<=m;++i){
        cin>>u>>v>>w>>c;
        add(u,v,w,c);
    }
    mcmf();
    cout<<maxf<<' '<<minc<<'\n';
    return 0;
}
```

杂项

珂朵莉树

- $1\ l\ r\ x$：将$[l,r]$区间所有数加上x
- $2\ l\ r\ x$：将$[l,r]$区间所有数改成x
- $3\ l\ r\ x$：输出将$[l,r]$区间从小到大排序后的第x个数是的多少(即区间第x小，数字大小相同算多次，保证$1 \leq x \leq r-l+1$)
- $4\ l\ r\ x\ y$：输出$[l,r]$区间每个数字的x次方的和模y的值(即$(\sum_{i=l}^{r} a_i^x) \mod y$)

```
#include <bits/stdc++.h>
#define LL long long
#define int long long
using namespace std;

const int MAXN=1e5+2;
const int mod7=1e9+7;

int n,m,seed,vmax,a[MAXN];

int rnd(){
    int ret=seed;
    seed=(1ll*seed*7+13)%mod7;
    return ret;
}

int qpow(LL a,int b,int mod){
```

```
    int base=1ll*a%mod,res=1;
    for(;b;b>>=1,base=1ll*base*base%mod)if(b&1){
        res=1ll*res*base%mod;
    }
    return res%mod;
}

namespace odt{
struct node_t{
    int l,r;
    mutable LL v;
    node_t(int _l,int _r,LL _v):l(_l),r(_r),v(_v){}
    bool operator<(const node_t& o)const{return l<o.l;}
};

set<node_t> tree;

auto split(int x){
    auto it=tree.lower_bound(node_t{x,0,0});
    if(it!=tree.end()&&it->l==x)return it;
    --it;
    int l=it->l,r=it->r;
    LL v=it->v;
    tree.erase(it);
    tree.insert(node_t(l,x-1,v));
    return tree.insert(node_t(x,r,v)).first;
}

void assign(int l,int r,LL v){
    auto ed=split(r+1),op=split(l);
    tree.erase(op,ed);
    tree.insert(node_t(l,r,v));
}

void add(int l,int r,LL v){
    auto ed=split(r+1),op=split(l);
    for(auto it=op;it!=ed;++it){
        it->v+=v;
    }
}

int kth(int l,int r,int k){
    auto ed=split(r+1),op=split(l);
    vector<pair<int,int>> vec;
    for(auto it=op;it!=ed;++it){
        vec.push_back(make_pair(it->v,it->r-it->l+1));
    }
    sort(vec.begin(),vec.end());
    for(auto&& i:vec){
        k-=i.second;
        if(k<=0)return i.first;
    }
    return -1;
}

int sum_of_pow(int l,int r,int x,int y){
```

```
    auto ed=split(r+1),op=split(l);
    int res=0;
    for(auto it=op;it!=ed;++it){
        res=1ll*(1ll*res+1ll*qpow(it->v,x,y)*(it->r-it->l+1))%y;
    }
    return res;
}
}

void start(){
    for(int i=1;i<=n;++i){
        a[i]=(rnd()%vmax)+1;
        odt::tree.insert(odt::node_t(i,i,a[i]));
    }
    odt::tree.insert(odt::node_t(n+1,n+1,0));
}

signed main(){
    cin>>n>>m>>seed>>vmax;
    start();
    for(int i=1,op,l,r,x,y;i<=m;++i){
        op=(rnd()%4)+1;
        l=(rnd()%n)+1;
        r=(rnd()%n)+1;
        if(l>r)swap(l,r);

        if(op==3){
            x=(rnd()%(r-l+1))+1;
        }else{
            x=(rnd()%vmax)+1;
        }

        if(op==4){
            y=(rnd()%vmax)+1;
        }

        if(op==1){
            odt::add(l,r,x);
        }else if(op==2){
            odt::assign(l,r,x);
        }else if(op==3){
            cout<<odt::kth(l,r,x)<<'\n';
        }else if(op==4){
            cout<<odt::sum_of_pow(l,r,x,y)<<'\n';
        }
    }
}
```

普通莫队

1到N之间的格子，每个都有一个颜色c，区间[l, r]之间取两个格子颜色相同的概率

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

using namespace std;
```

```cpp
#define int long long

int read(){
    int res=0,sign=1;char ch=getchar();
    for(;ch<'0'||ch>'9';ch=getchar())if(ch=='-')sign=-sign;
    for(;ch>='0'&&ch<='9';ch=getchar())res=(res<<3)+(res<<1)+(ch^'0');
    return res*sign;
}

const int N=5e4+10;

int n,m,len;
int c[N],id[N];

struct query_t{
    int l,r,idx;
    bool operator<(const query_t& o)const{
        if(id[l]!=id[o.l])return l<o.l;
        return (id[l]&1)?r<o.r:r>o.r;
    }
};

int gcd(int a,int b){
    if(!b)return a;
    return gcd(b,a%b);
}

struct ans_t{
    int p,q;

    void setans(int _p,int _q){
        int _gcd=gcd(_p,_q);
        p=_p/_gcd,q=_q/_gcd;
    }
};

query_t q[N];
ans_t ans[N];

int sum,cnt[N];

void add(int i){
    sum+=cnt[i];
    ++cnt[i];
}

void del(int i){
    --cnt[i];
    sum-=cnt[i];
}

signed main(){
    // freopen("C:\\Users\\ASUS\\Downloads\\P1494_7.in","r",stdin);
    n=read(),m=read();
    len=sqrt(n);
```

```
    for(int i=1;i<=n;++i){
        c[i]=read();
        id[i]=(i-1)/len+1;
    }
    for(int i=1;i<=m;++i){
        q[i].l=read(),q[i].r=read(),q[i].idx=i;
    }
    sort(q+1,q+1+m);

    for(int i=1,l=1,r=0;i<=m;++i){
        int idx=q[i].idx;
        if(q[i].l==q[i].r){
            ans[idx].setans(0,1);
            continue;
        }
        while(l>q[i].l)add(c[--l]);
        while(r<q[i].r)add(c[++r]);
        while(l<q[i].l)del(c[l++]);
        while(r>q[i].r)del(c[r--]);
        ans[idx].setans(sum,(r-l+1)*(r-l)/2);
    }
    for(int i=1;i<=m;++i){
        printf("%lld/%lld\n",ans[i].p,ans[i].q);
    }
}
```