1.离线算法

cdq分治

类似于求逆序对的归并排序，n次运算，每次运算只需运算>i的地方，可以考虑cdq分治，优化为 $O(nlogn)$

三维偏序

```
#include<bits/stdc++.h>
using namespace std;
const int N=2e5+10;
int n,k;//n为数组数量，k为树状数组大小
struct node{
    int x,y,z,num,ans;
}a[N];
bool cmp1(node a,node b){
    if (a.x==b.x){
        if (a.y==b.y)return a.z<b.z;
        return a.y<b.y;
    }return a.x<b.x;
}
bool cmp2(node a,node b){
    if(a.y==b.y)return a.z<b.z;
    else return a.y<b.y;
}
int c[N];
void add(int x,int y){
    for (;x<=k;c[x]+=y,x+=x&(-x));
}
int que(int x){
    int ans=0;for (;x;ans+=c[x],x-=x&(-x));return ans;
}
//node b[N];
void cdq(int l,int r){
    if(l==r)return;
    int mid=l+r>>1;
    cdq(l,mid);cdq(mid+1,r);
    sort(a+l,a+1+mid,cmp2);//第二维为关键字排序,sort(a+mid+1,a+1+r) mid+1~r,可以最后面归并排序来优化sort
```

```
    sort(a+1+mid,a+1+r,cmp2);
    int j=l;
    for(int i=mid+1;i<=r;++i){
        while(a[i].y>=a[j].y&&j<=mid){
            add(a[j].z,a[j].num);
            j++;
        }
        a[i].ans+=que(a[i].z);
    }
    for(int i=l;i<j;++i)add(a[i].z,-a[i].num);//清空树状数组
    /*//归并排序
    int l1=l,r1=mid,l2=mid+1,r2=r,cnt=l-1;
    while (l1<=r1&&l2<=r2){
        if (cmp2(a[l1],a[l2]))b[++cnt]=a[l1++];
        else b[++cnt]=a[l2++];
    }
    while (l1<=r1)b[++cnt]=a[l1++];
    while (l2<=r2)b[++cnt]=a[l2++];
    for (int i=l;i<=r;i++)a[i]=b[i];
    */
}
int ans[N];
signed main(){
    cin>>n>>k;
    for (int i=1;i<=n;i++)scanf("%d%d%d",&a[i].x,&a[i].y,&a[i].z);
    sort(a+1,a+1+n,cmp1);//第一维
    int cnt=0,num=0;
    for(int i=1;i<=n;++i){//去重
        ++num;
        if(a[i].x!=a[i+1].x||a[i].y!=a[i+1].y||a[i].z!=a[i+1].z){

a[++cnt].x=a[i].x;a[cnt].y=a[i].y;a[cnt].z=a[i].z;a[cnt].num=num;num=0;
        }
    }
    cdq(1,cnt);//cdq分治
    //cdq.a[i].ans为<a[i]的数的个数
    for(int i=1;i<=cnt;++i)ans[a[i].ans+a[i].num-1]+=a[i].num;
    for(int i=0;i<n;++i)printf("%d\n",ans[i]);
    return 0;
}
```

树上启发式合并(dsu on tree,静态链分治)

离线，重链信息保留，轻链删除，往往和线段树合并用与类似题型

1.给出一棵n个节点以1为根的树，节点u的颜色为c_u，现在对于每个结点u询问子树里一共出现了多少种不同的颜色$n \le 1e5$，$c \le 1e5$

$O(nlogn)$

```
#include<bits/stdc++.h>
using namespace std;
#define int long long
const int N=1e5+10;
int n,m,k;
int c[N],ans[N];
int sz[N],son[N];
vector<int>v[N];
```

```
void dfs1(int x,int fa){//预处理出重儿子
    sz[x]=1;
    for (int i:v[x]){
        if (i==fa)continue;
        dfs1(i,x);
        sz[x]+=sz[i];
        if (sz[son[x]]<sz[i])son[x]=i;
    }
}
int num[N],sum=0,nowson,mx=0;
void cal(int x,int fa,int val){
    num[c[x]]+=val;
    int t=num[c[x]];
    if (t>mx)mx=t,sum=c[x];
    else if (t==mx)sum+=c[x];
    for (int i:v[x]){
        if (i==fa||i==nowson)continue;
        cal(i,x,val);
    }
}
void dfs2(int x,int fa,int is){
    for(int i:v[x]){
        if (i==fa||i==son[x])continue;
        dfs2(i,x,0);
    }
    if(son[x])dfs2(son[x],x,1);
    nowson=son[x];//加入除重儿子外的值，重儿子已经加过了
    cal(x,fa,1);
    ans[x]=sum;
    if (!is){//不是重儿子，将子树所有值回退
        nowson=0;cal(x,fa,-1);mx=0;
    }
}
signed main(){
    int x,y;
    cin>>n;
    for (int i=1;i<=n;i++)scanf("%lld",&c[i]);
    for (int i=1;i<n;i++){
        scanf("%lld%lld",&x,&y);
        v[x].push_back(y);v[y].push_back(x);
    }
    dfs1(1,0);
    dfs2(1,0,1);
    for (int i=1;i<=n;i++)printf("%lld ",ans[i]);
    return 0;
}
```

莫队

1.普通莫队

例题：长度为n的颜色数组，q次询问，每次询问区间的颜色种数，$O(q\sqrt{n}+n\sqrt{n}+q\log q)$

分析：以l所在块为第一关键字排序，r为第二关键字排序，此处同时采用了奇偶优化

```
#include<bits/stdc++.h>
using namespace std;
const int N=1e5+10;
```

```
int n,q;
int a[N];
struct node{
    int l,r,id,block;
}b[N];
bool cmp(node a,node b){
    return a.block==b.block?(a.block&1)?a.r<b.r:a.r>b.r:a.block<b.block;
}
int c[N],cnt=0;
void add(int x){
    if (!c[a[x]])cnt++;
    c[a[x]]++;
}
void del(int x){
    c[a[x]]--;
    if (!c[a[x]])cnt--;
}
int ans[N];
signed main(){
    cin>>n;
    for (int i=1;i<=n;i++)scanf("%d",&a[i]);
    cin>>q;
    int block=sqrt(n);
    for (int i=1;i<=q;i++){
        scanf("%d%d",&b[i].l,&b[i].r);
        b[i].id=i;b[i].block=b[i].l/block;
    }
    sort(b+1,b+1+q,cmp);
    int l=0,r=0;
    for (int i=1;i<=q;i++){
        while (r<b[i].r)add(++r);
        while (r>b[i].r)del(r--);
        while (l>b[i].l)add(--l);
        while (l<b[i].l)del(l++);
        ans[b[i].id]=cnt;
    }
    for (int i=1;i<=q;i++)printf("%d\n",ans[i]);
    return 0;
}
```

2.滚动莫队

有些题目在区间转移时，可能会出现增加或者删除无法实现的问题。在只有增加不可实现或者只有删除不可实现的时候，就可以使用回滚莫队在$O(n\sqrt{n})$的时间内解决问题。回滚莫队的核心思想就是既然我只能实现一个操作，那么我就只使用一个操作，剩下的交给回滚解决。

例题：长度为n的数组，q个询问 ，每次询问一个区间$[l,r]$内重要度最大的数字，要求输出其重要度。一个数字重要度的定义为i乘上i在区间内出现的次数。

分析：在这个问题中，在增加的过程中更新答案是很好实现的，但是在删除的过程中更新答案是不好实现的。

代码中__代表临时。

以l所在块为第一关键字排序，r为第二关键字排序

当$[l,r]$在一个块内，直接用临时数组来暴力更新答案，$O(\sqrt{n})$

当这次询问的l和之前的询问的l在不同块内时，重新初始化莫队最多初始化$\sqrt{n}$边，复杂度$O(n\sqrt{n})$

对于r直接每次从小到大枚举，更新答案

对于l直接每次暴力，$O(\sqrt{n})$，运用临时变量记载，并且每次回滚初值

```
#include<bits/stdc++.h>
using namespace std;
const int N=1e5+10;
int n,q;
int a[N],c[N];
struct node{
    int l,r,id;
}b[N];
int L[N],R[N],pos[N];
bool cmp(node a,node b){
    if (pos[a.l]==pos[b.l])return a.r<b.r;
    return pos[a.l]<pos[b.l];
}
int cnt[N],__cnt[N];
inline void add(int x,long long& Ans) {
    ++cnt[x];
    Ans=max(Ans,1LL*cnt[x]*c[x]);
}
inline void del(int x){--cnt[x];}

long long ans[N];
int main(){
    cin>>n>>q;
    int cnt=0;
    for (int i=1;i<=n;i++)scanf("%d",&a[i]),c[++cnt]=a[i];
    for (int i=1;i<=q;i++)scanf("%d%d",&b[i].l,&b[i].r),b[i].id=i;
    int block=sqrt(n);
    int tot=n/block;
    for (int i=1;i<=tot;i++)L[i]=(i-1)*block+1,R[i]=i*block;
    if (R[tot]<n)L[++tot]=R[tot-1]+1,R[tot]=n;
    for (int i=1;i<=tot;i++)for (int j=L[i];j<=R[i];j++)pos[j]=i;
    sort(b+1,b+1+q,cmp);
    //离散化
    sort(c+1,c+1+cnt);
    cnt=unique(c+1,c+1+cnt)-c-1;
    for (int i=1;i<=n;i++)a[i]=lower_bound(c+1,c+1+cnt,a[i])-c;

    int l=1,r=0,last_block=0,__l;
    long long Ans=0,tmp;
    for (int i=1;i<=q;i++){
        //询问的左右端点同属于一个块则暴力扫描回答
        if (pos[b[i].l]==pos[b[i].r]) {
            for (int j=b[i].l;j<=b[i].r;j++)++__cnt[a[j]];
            for (int
j=b[i].l;j<=b[i].r;j++)ans[b[i].id]=max(ans[b[i].id],1LL*c[a[j]]*__cnt[a[j]]);
            for (int j=b[i].l;j<=b[i].r;j++)--__cnt[a[j]];
            continue;
        }
        //访问到了新的块则重新初始化莫队区间
        if (pos[b[i].l]!=last_block){
            while (l<r+1)del(a[r--]);
            r=R[pos[b[i].l]];l=R[pos[b[i].l]]+1;
            Ans=0;last_block=pos[b[i].l];
```

```
        }
        //扩展右端点
        while (r<b[i].r)add(a[++r],Ans);
        __l=l;tmp=Ans;
        //扩展左端点
        while (__l>b[i].l)add(a[--__l],tmp);
        ans[b[i].id]=tmp;
        //回滚
        while (__l<l)del(a[__l++]);
    }
    for (int i=1;i<=q;i++)printf("%lld\n",ans[i]);
    return 0;
}
```

2.杂项知识

小知识

排列逆序对数=环数+排列长度

图上点含有值和颜色(两种颜色)的属性，当交换相邻点的值时，若两点颜色相同则颜色都翻转。这个操作等价于交换两点的值和颜色，再翻转其两点颜色。

众数

给定一组数，询问有多少个子区间，其中一个数的个数严格大于其他数的个数

```

```

对分写法

```
while(l<=r){
    int mid=l+r>>1;
    if (check(mid))l=mid+1;
    else r=mid-1;
}//r
while(l<r){
    int mid=l+r>>1;
    if (check(mid))l=mid+1;
    else r=mid;
}//l
```

分块打表

定义$f(x)$为x的数位和，问[l,r]区间内$x\%f(x)=0$的数的个数，$1\le l,r\le 1e9$

原题可以数位dp

现在对$1e9$分块，打表出n个数，然后不包含整块的另算，设$f(x)$时间复杂度为r,$O(Tr1e9/n)$

如$n=10000$，时间复杂度为$O(1e5Tr)$

1e4，每个数4位数，打表差不多50KB，hdoj输入限制64KB

打表程序

```
#include<bits/stdc++.h>
using namespace std;
int f(int x){
```

```
    if (x==0)return 0;
    return f(x/10)+x%10;
}
signed main(){
    int T;
    ofstream cout;
    cout.open("1002.in");
    int block=1e5,num=1e4;
    for (int i=0;i<num;i++){
        int ans=0;
        for (int j=i*block+1;j<=(i+1)*block;j++){
            if (j%f(j)==0)ans++;
        }
        if(i!=num-1)cout<<ans<<",";
        else cout<<ans;
    }
    return 0;
}
```

```
#include<bits/stdc++.h>
using namespace std;
const int N=1e5+10;
int a[N]={};//存入打表数据
int f(int x){
    if (x==0)return 0;
    return f(x/10)+x%10;
}
int main(){
    int T;
    scanf("%d",&T);
    for(int cas=1;cas<=T;cas++){
        int l,r;
        scanf("%d%d",&l,&r);
        int ans=0;
        int ll,rr;
        int block=1e5,num=1e4;
        for (int i=0;i<num;i++){
            if (l<=i*block+1){
                ll=i;break;
            }
        }
        for (int i=num;i>=0;i--){
            if (r>=i*block){
                rr=i-1;break;
            }
        }
        if (ll<=rr){
            for (int i=ll;i<=rr;i++)ans+=a[i];
            for (int i=l;i<=ll*block;i++)if (i%f(i)==0)ans++;
            for (int i=(rr+1)*block+1;i<=r;i++)if (i%f(i)==0)ans++;
        }
        else{
            for (int i=l;i<=r;i++)if (i%f(i)==0)ans++;
        }
        printf("Case %d: %d\n",cas,ans);
    }
    return 0;
```

```
}
```

3.BM(查询时带上杜教筛)

BM，寻找线性关系，找规律

```
#include<bits/stdc++.h>
using namespace std;
#define rep(i,a,n) for (int i=a;i<n;i++)
#define pb push_back
typedef long long ll;
#define SZ(x) ((ll)(x).size())
typedef vector<ll> VI;
typedef pair<ll,ll> PII;
const ll mod=1000000007;
ll powmod(ll a,ll b) {ll res=1;a%=mod; assert(b>=0); for(;b;b>>=1)
{if(b&1)res=res*a%mod;a=a*a%mod;}return res;}
// head
ll _,n;
namespace linear_seq {
    const ll N=10010;
    ll res[N],base[N],_c[N],_md[N];

    vector<ll> Md;
    void mul(ll *a,ll *b,ll k) {
        rep(i,0,k+k) _c[i]=0;
        rep(i,0,k) if (a[i]) rep(j,0,k) _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
        for (ll i=k+k-1;i>=k;i--) if (_c[i])
            rep(j,0,SZ(Md)) _c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%mod;
        rep(i,0,k) a[i]=_c[i];
    }
    ll solve(ll n,VI a,VI b) {
        ll ans=0,pnt=0;
        ll k=SZ(a);
        assert(SZ(a)==SZ(b));
        rep(i,0,k) _md[k-1-i]=-a[i];_md[k]=1;
        Md.clear();
        rep(i,0,k) if (_md[i]!=0) Md.push_back(i);
        rep(i,0,k) res[i]=base[i]=0;
        res[0]=1;
        while ((1ll<<pnt)<=n) pnt++;
        for (ll p=pnt;p>=0;p--) {
            mul(res,res,k);
            if ((n>>p)&1) {
                for (ll i=k-1;i>=0;i--) res[i+1]=res[i];res[0]=0;
                rep(j,0,SZ(Md)) res[Md[j]]=(res[Md[j]]-res[k]*_md[Md[j]])%mod;
            }
        }
        rep(i,0,k) ans=(ans+res[i]*b[i])%mod;
        if (ans<0) ans+=mod;
        return ans;
    }
    VI BM(VI s) {
        VI C(1,1),B(1,1);
        ll L=0,m=1,b=1;
        rep(n,0,SZ(s)) {
            ll d=0;
```

```
            rep(i,0,L+1) d=(d+(ll)C[i]*s[n-i])%mod;
            if (d==0) ++m;
            else if (2*L<=n) {
                VI T=C;
                ll c=mod-d*powmod(b,mod-2)%mod;
                while (SZ(C)<SZ(B)+m) C.pb(0);
                rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
                L=n+1-L; B=T; b=d; m=1;
            } else {
                ll c=mod-d*powmod(b,mod-2)%mod;
                while (SZ(C)<SZ(B)+m) C.pb(0);
                rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
                ++m;
            }
        }
        return C;
    }
    ll gao(VI a,ll n) {
        VI c=BM(a);
        c.erase(c.begin());
        rep(i,0,SZ(c)) c[i]=(mod-c[i])%mod;
        return solve(n,c,VI(a.begin(),a.begin()+SZ(c)));
    }
};

int main() {
    int T;
    scanf("%d",&T);
    while (T--)
    {
        scanf("%lld",&n);
        vector<ll>v;
        v.push_back(3);
        v.push_back(9);
        v.push_back(20);
        v.push_back(46);
        v.push_back(106);
        v.push_back(244);
        v.push_back(560);
        v.push_back(1286);
        v.push_back(2956);
        v.push_back(6794);
        printf("%lld\n",linear_seq::gao(v,n-1));//输出第n项
    }
}
```