CDQ分治

重庆南开信竞基础课程

重庆南开信竞

分治 Divide and Conquer

一种军队中常见的战术

分(Divide)： 把一个大的复杂的问题，分割成两个或多个相似子问题

治(Conquer)： 逐个求解子问题，最终得到原问题的解

重庆南开信竞

CDQ分治

CDQ分治是一种**对时间分治**算法。

对于有**多个维度的问题，对其中一个维度进行分治**，把穿插的修改和查询操作化为统一修改后统一查询的问题的算法。

通过增加一个log的时间复杂度，从而**把问题降维。**

CDQ分治可以用来代替很多复杂数据结构（比如树套树）。但要求问题必须可以**离线处理。**

重庆南开信竞

陈丹琦 2008ioi金牌
普林斯顿大学计算机
科学系助理教授

CDQ分治

前置知识：树状数组 + 归并排序

重庆南开信竞

CDQ分治的基本套路

若要解决一系列问题，这些问题一般包含修改和查询操作，可以把这些问题排成一个序列，用一个区间[L,R]表示。

分：递归处理左边区间[L,M]和右边区间[M+1,R]的问题。

治：合并两个子问题，同时考虑到[L,M]内的修改对[M+1,R]内的查询产生的影响。即，用左边的子问题帮助解决右边的子问题。

和普通分治不同的地方在于，普通分治在合并两个子问题的过程中，[L,M]内的问题不会对[M+1,R]内的问题产生影响。

重庆南开信竞

简单地说：**一半贡一半**。计算前一部分子问题的修改操作对后一部分子问题的影响。

使用条件

1.问题必须可以离线

2.前一半做出决策后，后一半不可以再反过来影响前一半——也就是无后效性

3.需要维护的数据规模与修改/提问的次数同阶

归并排序求逆序对

```
void merge(int L,int Mid,int R)
{
        int i=L, j=Mid+1, k=L;
        while(i<=Mid&&j<=R)
        {
                if(s1[i]<=s1[j])s2[k++]=s1[i++];
                else
                {
                        cnt+=Mid-i+1          //计算前一半对后一半的贡献

                        s2[k++]=s1[j++];
                }
        }
        while(i<=Mid) s2[k++]=s1[i++];
        while(j<=R)   s2[k++]=s1[j++];
        for(i=L;i<=R;i++)s1[i]=s2[i];
}
```

CDQ与归并排序求逆序对

归并排序求逆序对就是CDQ思想的体现：

我们要求区间[L,R]中的逆序对。

分：递归求解左边区间[L,M]和右边区间[M+1,R]各自的逆序对数。

治：合并两个子问题，同时计算到[L,M]内的数字对[M+1,R]内的逆序对个数产生的贡献。

每次我们从右边区间的有序序列中取出一个元素时，要把“以该元素结尾的逆序对个数”加上“左边区间有多少个元素比它大”。即，用左边的子问题帮助解决右边的子问题。

上面求逆序对的过程实际上是解决**二维偏序**问题（偏序问题就是多约束条件的元素统计问题）。

重庆南开信竞

我们要求下列数组的逆序对数：

A={ 9, 3, 1, 7, 6 }

下标 1, 2, 3, 4, 5

我们可以把下标看作时间维度x,把数值看作数值维度y,

即共有5个点(1,9),(2,3),(3,1),(4,7),(5,6)

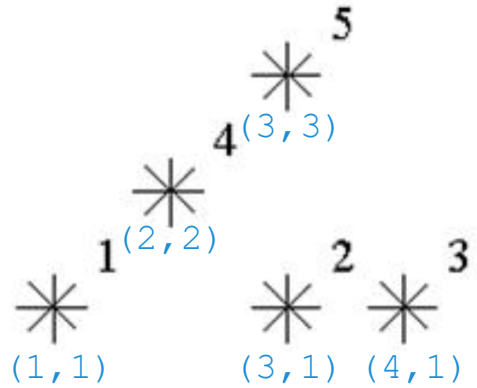
对于其中一个点(x_i,y_i),它贡献的逆序对数实际上是点(x_j,y_j)的个数： $x_i>=x_j$且$y_i<=y_j$

这就是一个二维偏序问题

二维偏序

数星星NKOJ1908

宇航员经常检测星图,在星图上，星星由点表示而且每颗星星都有笛卡尔坐标。星星的等级表示左下方(包含正左和正下方)星星的数量。宇航员想知道星星等级的分布。



例如，如上面图形所示，第5号星等级是3 (它由三个标记为1,2和4的星组成)。在此地图上，0等级的星星只有一个(1号)，1等级的有两个（2和4号），2等级的有一个(3号)，3等级的有一个（5号）。你设计一个程序，在给定地图上计算出每个等级星星的数量。

输入格式

第一行包含N个星星(1<=N<=200000)， 接下来的N行描绘星星的坐标（每行中有两个整数X和Y，由空格分开， 0<=X，Y<=32000）。 在图上的一点只能存在一颗星星，星星根据Y坐标的递增顺序排列，Y坐标相同的星星根据X坐标的递增顺序排列。

输出格式

包含N 行，一行一个数字，第一行包含0等级星星的数量，第二行包含1等级星星的数量，等等，最后一行包含N-1等级星星的数量。

做法一：按y排序，按x建树状数组，求“顺序对”，如果x，y的值很大，需要进行离散化。$O(nlog_2{}^n)$

做法二：CDQ分治，按y排序，对x分治$O(nlog_2{}^n)$

我们把y这一维度当作时间，按y排序，忽略掉其影响，将问题降为一维度。这样就变成了对于x求“顺序对”。

重庆南开信竞

数星星NKOJ1908

```
struct node{ int x,y,id;} A[N],B[N];
int n,Ans[200005],Star[200005];

int main()
{
    int i,j,k;
    scanf("%d",&n);
    for(i=1; i<=n; i++)
       scanf("%d%d",&A[i].x,&A[i].y),A[i].id=i;
    CDQ(1,n);
    for(i=1; i<=n; i++)Star[Ans[i]]++;
    for(i=0; i<n; i++)printf("%d\n",Star[i]);
}
```

```
void CDQ(int l,int r)
{
    if(l==r)return;
    int i,j,k,mid=l+r>>1;
    CDQ(l,mid);
    CDQ(mid+1,r);
    i=l;
    j=mid+1;
    k=0;
    while(i<=mid&&j<=r)
    {
        if(A[i].x<=A[j].x)B[++k]=A[i++];
        else Ans[A[j].id]+=i-l,B[++k]=A[j++];
    }
    while(i<=mid)B[++k]=A[i++];
    while(j<=r)Ans[A[j].id]+=mid-l+1,B[++k]=A[j++];
    for(i=1; i<=k; i++)A[l+i-1]=B[i];
}
```

重庆南开信竞

此题一开始y已经有序，所以直接对x分治。
否则应先对y排序。

数列操作 NKOJ1321

问题描述

假设有一列数{Ai}(1≤i≤n)，支持如下两种操作：
将Ak的值加D。（k, D是输入的数）
输出As+As+1+...+At。（s, t都是输入的数，S≤T）

输入格式

第一行一个整数n，
第二行为n个整数，表示{Ai}的初始值≤10000。
第三行为一个整数m，表示操作数
下接m行，每行描述一个操作，有如下两种情况：
ADD k d (表示将Ak加d，1<=k<=n，d为数,d的绝对值不超过10000)
SUM s t （表示输出As+...+At)

M,N<=100000

输出格式

对于每一个SUM提问，输出结果

做法一：树状数组 $O(nlog_2{}^n)$

做法二：线段树 $O(nlog_2{}^n)$

做法三：分块 $O(n\sqrt{n})$

做法四：CDQ分治 $O(nlog_2{}^n)$

重庆南开信竞
重庆南开信竞

数列操作 NKOJ1321

问题描述

假设有一列数{Ai}(1≤i≤n)，支持如下两种操作：
将Ak的值加D。（k, D是输入的数）
输出As+As+1+…+At。（s, t都是输入的数，S≤T）

输入格式

第一行一个整数n，
第二行为n个整数，表示{Ai}的初始值≤10000。
第三行为一个整数m，表示操作数
下接m行，每行描述一个操作，有如下两种情况：
ADD k d (表示将Ak加d，1<=k<=n，d为数,d的绝对值不超过10000)
SUM s t （表示输出As+…+At)

M,N<=100000

输出格式

对于每一个SUM提问，输出结果

我们按操作发生的时间离线处理操作：
用结构体`struct node{int Pos,Id,Type;}A[200005];`来记录每个操作。
用`tot`来记录当前是第几个操作(操作编号)

对于`ADD k d` 操作
```
tot++; A[tot].Pos=k,A[tot].Id=d,A[tot].Type=0;
```
其中`Pos`记录操作发生的位置`k`，
`Id`记录修改的值`d`，
`Type`记录操作类型`0`

对于`SUM s t` 操作
求区间和`Sum[1, t]-Sum[1, s-1]`，我们可以把它分解为两个操作：
求区间和`Sum[1, s-1]`，求区间和`Sum[1, t]`
```
Cnt++;
tot++; A[tot].Pos=s-1,A[tot].Id=Cnt, A[tot].Type=-1
tot++; A[tot].Pos=t,  A[tot].Id=Cnt, A[tot].Type=1
```
`Id=Cnt`记录它们同属第几个查询操作，
`Type=-1`表示查询时要减去该操作结果，
`Type=1`表示要加上该操作的结果

就把它转换成一个二维偏序问题：
用操作的编号`i`表示时间，用操作发生的位置`Pos`表示地点，这样每个操作对应一个二维信息`(i,Pos)`。
对于一个**询问操作**`i`，对于所有**`A[j].Pos<=A[i].Pos` 且 `j<=i`** 的**修改操作**，其`A[j].Id`都应该累加到`i`号查询的答案上 `Ans[A[i].Id]+=A[j].Id*A[i].Type;`

时间默认有序，我们对位置进行分治
对于一段操作编号区间`[L,R]`
`Mid=(L+R)/2`
首先递归处理`[L,Mid]`和`[Mid+1,R]`
　　此时，两个区间已经算完部分查询操作的贡献。`[L,Mid]`区间中的所有操作已按`Pos`值排序，`[Mid+1,R]`中的所有操作也已按`Pos`值排好序
然后，讨论`[L,Mid]`区间中的修改操作对`[Mid+1,R]`区间中的查询操作的贡献
　　对于`[Mid+1,R]`区间中的`j`号操作，我们只能保证`[L,Mid]`区间的所有操作时间上都发生在`j`之前。所以只要其中的修改操作发生位置也小于等于`j`的位置，那么该修改操作就会对`j`的查询产生贡献。

重庆南开信竞

数列操作 NKOJ1321

```
struct node{int Pos,Id,Type;}A[200005],B[200005];
int n,m,S[200005],Ans[200005],tot,cnt;

int main()
{
    int i,j,k,x,y;char c[4];
    scanf("%d",&n);
    for(i=1;i<=n;i++)scanf("%d",&S[i]),S[i]+=S[i-1];
    scanf("%d",&m);
    for(i=1;i<=m;i++)
    {
        scanf("\n%s %d %d",c,&x,&y);
        if(c[0]=='S')
        {
            cnt++;
            A[++tot].Pos=x-1; A[tot].Id=cnt; A[tot].Type=-1;
            A[++tot].Pos=y;   A[tot].Id=cnt; A[tot].Type=1;
            Ans[cnt]+=S[y]-S[x-1];
        }
        else A[++tot].Pos=x,A[tot].Id=y,A[tot].Type=0;
    }
    CDQ(1,tot);
    for(i=1;i<=cnt;i++)printf("%d\n",Ans[i]);
}
```

重庆南开信竞

```
void CDQ(int l,int r)
{
    if(l==r)return;
    int i,j,k,sum=0,Mid=l+r>>1;
    CDQ(l,Mid);
    CDQ(Mid+1,r);
    i=l; j=Mid+1; k=0;
    while(i<=Mid&&j<=r)
    {
        if(A[i].Pos<=A[j].Pos)
        {
            if(A[i].Type==0)sum+=A[i].Id;
            B[++k]=A[i++];
        }
        else
        {
            if(A[j].Type!=0)Ans[A[j].Id]+=A[j].Type*sum;
            B[++k]=A[j++];
        }
    }
    while(i<=Mid)B[++k]=A[i++];
    while(j<=r)
    {
        if(A[j].Type!=0)Ans[A[j].Id]+=A[j].Type*sum;
        B[++k]=A[j++];
    }
    for(i=1;i<=k;i++)A[l+i-1]=B[i];
}
```

三维偏序

菊花的故事1 NKOJ3655

问题描述

何老板很喜欢菊花，所以现在他买了n朵菊花.
每朵花有三个属性：花形(s)、颜色(c)、气味(m)，又三个整数表示。
现要对每朵花评级，一朵花的级别是它拥有的美丽能超过的花的数量。定义一朵花A比另一朵花B要美丽，
当且仅当Sa>=Sb,Ca>=Cb,Ma>=Mb。显然，两朵花可能有同样的属性。需要统计出评出每个等级的花的数量。

输入格式

第一行为N,K (1 <= N <= 100,000, 1 <= K <= 200,000), 分别表示花的数量和最大属性值。
以下N行，每行三个整数si, ci, mi (1 <= si, ci, mi <= K)，表示第i朵花的属性

输出格式

包含N行，分别表示评级为0...N-1的每级花的数量。

做法一：按x排序，按y，z建立树套树$O(nlog^2n)$

做法二：k-d tree $O(n^{5/3})$

做法三：CDQ分治
按x排序，对y分治，对z建立树状数组$O(nlog^2n)$

重庆南开信竞

菊花的故事1 NKOJ3655

三维偏序 第一维sort，第二维CDQ分治，第三维数据结构
即是 第一维x直接排序，第二维y用CDQ分治，第三维z用树状数组

我们在CDQ处理[L,R]区间时，讨论[L,Mid]对[Mid+1,R]的贡献。

1.事先按第属性x排过序，所以在处理[L,Mid]时，不管属性x怎么被打乱，[Mid+1,R]所有元素的x属性一定不小于[L,Mid]中所有元素的x属性。

当处理完CDQ(L,Mid)和CDQ(Mid+1,R)后，我们明确的信息有：

1)[L,Mid] 中的所有元素的属性x都是不大于[Mid+1,R]中所有元素的x属性的；

2)[L,Mid] 中的所有元素的属性x可能是乱序，但属性y是有序的；

3)[Mid+1,R]中的所有元素的属性x可能是乱序，但属性y是有序的；

2.当讨论到[L,Mid]中某个元素i,[Mid+1,R]中某个元素j时，若y[i]<=y[j],此时也有x[i]<=x[j]。
但z[i]与z[j]的关系是无序的，所以我们就把z[i]添加到树状数组里，以便后面统计贡献。

3.当讨论到[L,Mid]中某个元素i,[Mid+1,R]中某个元素j时，若y[i]>y[j],
因y是有序的，所以[i,Mid]中所有元素的y属性都是大于y[j]的，
[L,i-1]中任意元素k,都有 x[k]<=x[j]且y[k]<=y[j]， 如果z[k]<=z[j]，那么k就对j的答案有贡献。
此时，在树状数组里去查询[1,z[j]]区间的总和即是[L,i-1]区间对j的贡献。

4.处理完[L,R]区间后，[L,Mid]对[Mid+1,R]的贡献已统计完毕，树状数组里的信息就失效，我们应将树状数组还原。

重庆南开信竞
重庆南开信竞

菊花的故事1 NKOJ3655

```
int n,k,ans[100005],tot;
struct flower
{   int x,y,z,num,ans;} A[100005],B[100005];

int main()
{
    read(n); read(k);
    for(int i=1,x,y,z; i<=n; i++)
    {
        read(x); read(y); read(z);
        B[i]= {x,y,z,1,0};
    }
    sort(B+1,B+n+1,cmp);
    for(int i=1; i<=n; i++)
    {
        if(B[i].x!=A[tot].x||B[i].y!=A[tot].y||B[i].z!=A[tot].z)
            A[++tot]= {B[i].x,B[i].y,B[i].z,1,0};
        else A[tot].num++;
    }
    CDQ(1,tot);
    for(int i=1; i<=tot; i++) ans[A[i].ans+A[i].num-1]+=A[i].num;
    for(int i=0; i<n; i++) printf("%d\n",ans[i]);
}
```

```
void CDQ(int l,int r)
{
    if(l==r) return;
    int mid=(l+r)>>1;
    CDQ(l,mid);
    CDQ(mid+1,r);
    int i=l, j=mid+1, k=l;
    while(i<=mid && j<=r)
    {
        if(A[i].y<=A[j].y) Modify(A[i].z,A[i].num),B[k++]=A[i++];
        else A[j].ans+=GetSum(A[j].z),B[k++]=A[j++];
    }
    while(i<=mid) Modify(A[i].z,A[i].num),B[k++]=A[i++];
    while(j<=r) A[j].ans+=GetSum(A[j].z),B[k++]=A[j++];
    for(int i=l; i<=mid; i++) Modify(A[i].z,-A[i].num);
    for(int i=l; i<=r; i++) A[i]=B[i];
}
```

重庆南开信竞

动态逆序对 NKOJ2041

将问题转换为修改操作：
把原始的n个数字，看作n次添加操作；
把删除m个数字，看作m次删除操作；
用结构体来记录每个操作 `struct Node{ int Type,Num,Pos,Id;}A[200005];`
Type记录操作类型，1表示添加，-1表示删除；
Pos记录操作发生的位置；
Num记录操作针对的数字；
Id记录这是第几个删除操作；

这样总共有n+m个操作，编号1到n+m，先进行n次添加，再进行m次删除。
对于操作A[i],A[j]，若i<j,表示操作i发生在操作j之前。

对于i,j两次操作，满足下列条件之一，i就对j的逆序对有贡献：
i<j且A[i].Pos<A[j].Pos且A[i].Num>A[j].Num　　（i在j的左边，且i的值比j的值大）
i<j且A[i].Pos>A[j].Pos且A[i].Num<A[j].Num　　（i在j的右边，且i的值比j的值小）
这样就转换为了三维偏序问题，可以通过CDQ分治来解决。

第一维操作时间i默认有序
第二维操作位置Pos用CDQ分治
第三维操作数值Num用树状数组维护

总用时	最大用时	最大内存	判题结果
1168 MS	374 MS	6484 KB	Accepted (100)
3180 MS	889 MS	103244 KB	Accepted (100)
5863 MS	1591 MS	32472 KB	Accepted (100)

CDQ分治

树状数组套主席树

线段树套平衡树

问题描述

对于序列A，它的逆序对数定义为满足$i < j$，且$A_i > A_j$的数对(i, j)的个数。给1到n的一个排列，按照某种顺序依次删除m个元素，你的任务是在每次删除一个元素之前统计整个序列的逆序对数。

$n <= 100000$
$m <= 50000$

输入格式

输入第一行包含两个整数n和m，即初始元素的个数和删除的元素个数。
以下n行每行包含一个1到n之间的正整数，即初始排列。
以下m行每行一个正整数，依次为每次删除的元素。

输出格式

输出包含m行，依次为删除每个元素之前，逆序对的个数。

重庆南开信竞

动态逆序对 NKOJ2041

```
struct Node
{   int Type,Num,Pos,Id;} A[200005];
int n,m,tot;
int pos[200005],a[200005],c[200005];
long long ans[200005];

int main()
{
    n=read(),m=read();
    for (int i=1; i<=n; ++i)
    {
       a[i]=read();
       pos[a[i]]=i;
       A[++tot]=(Node){1,a[i],i,0};
    }
    for (int i=1,x; i<=m; ++i)
    {
       x=read();
       A[++tot]=(Node){-1,x,pos[x],i};
    }
    cdq(1,tot);
    for (int i=1; i<=m; ++i) ans[i]+=ans[i-1];
    for (int i=0; i<m; ++i) printf("%lld\n",ans[i]);
    return 0;
}
```

```
void cdq(int l,int r)
{
    if (l==r) return;
    int Mid=(l+r)>>1,j=l;
    cdq(l,Mid);
    cdq(Mid+1,r);
    sort(A+l,A+Mid+1,Cmp);  //按.Pos由小到大排序，也可以在归并时顺带排序
    sort(A+Mid+1,A+r+1,Cmp);

    //处理左边的贡献
    for (int i=Mid+1; i<=r; ++i)
    {
        while (j<=Mid&&A[j].Pos<=A[i].Pos)Modify(A[j].Num,A[j].Type),++j;
        ans[A[i].Id]+=A[i].Type*(GetSum(n)-GetSum(A[i].Num));
    }
    for (int i=l; i<j; ++i) Modify(A[i].Num,-A[i].Type);//还原树状数组

    //处理右边的贡献
    j=Mid;
    for (int i=r; i>Mid; --i)
    {
        while (j>=l&&A[j].Pos>=A[i].Pos)Modify(A[j].Num,A[j].Type),--j;
        ans[A[i].Id]+=A[i].Type*GetSum(A[i].Num-1);
    }
    for (int i=Mid; i>j; --i) Modify(A[i].Num,-A[i].Type);//还原树状数组
}
```

重庆南开信竞

四维偏序

四维偏序?

做法一：对x排序，树套树套树　　$O(nlog^3n)$

做法二：CDQ分治，按x排序，对y分治，对z，w建立树套树　　$O(nlog^3n)$

做法三：CDQ分治套CDQ分治，对x排序，对y分治，对z，w再次CDQ分治　　$O(nlog^3n)$

重庆南开信竞

CDQ小总结

CDQ分治

CDQ分治适用于满足一下两个条件的数据结构题：
1.修改操作对询问的贡献独立，修改操作之间互不影响效果；
2.题目允许使用离线算法；

重庆南开信竞

它的本质是按时间分治，即若要处理时间[L,R]上的修改与询问操作，
先递归处理区间[L,Mid]与[Mid+1,R]
再处理[L,Mid]上的修改对[Mid+1,R]上的询问的影响

重庆南开信竞

CDQ分治会使得思维难度与代码难度大大减小
通常利用 CDQ 分治能使得一个树套树实现的题目，能够去掉外层的树，改为用分治来进行求解。

CDQ分治是一种解决问题的思想。与常规分治对比，常规分治是通过解决左右两个子区间的问题，合并得到大区间问题的解。CDQ分治通过处理左区间对右区间的影响来得到大区间的解。
一般CDQ分治可以顶替一层数据结构，而且常数小，空间开销也小。缺点是需要离线。

奋斗吧少年

巨大的成功需要付出巨大的代价

no sacrifice, no success