CDQ 分治

算法简介

一种离线处理多维偏序问题的算法。

算法模板

洛谷p3810

题意

三维空间中给定 n 个点，编号为 $1 \sim n$，定义 $f[i]$ 表示恰好有 i 个元素满足 $x_i\le x_j,y_i\le y_j,z_i\le z_j$ 且 $i\ne j$ 的 j 的个数。

要求输出 $f[0 \sim n-1]$。

题解 1

先考虑所有元素互异的情况。

先将第一维作为第一关键字，第二维作为第二关键字，第三维作为第三关键字对序列进行第一轮排序。

排序后将第二维作为第一关键字，第三维作为第二关键字对序列进行归并排序，利用分治过程计算答案贡献。

分治过程中受第一轮排序结果影响，只有左区间元素能为右区间元素做贡献，而右区间不能为左区间做贡献。

同时受第二轮排序结果影响，左区间和右区间元素的第二维单调不减，于是考虑用树状数组来维护左区间的第三维对右区间的贡献。

需要注意每轮分治完成后需要重置树状数组，考虑再次减去左区间贡献，不能暴力重置。

最后也可以发现，这样不会遗漏任何贡献。关于重复元素，最后特殊处理一下即可。

总时间复杂度 $O(n\log n\log v)$。

```
const int MAXN=1e5+5,MAXV=2e5+5;
struct Node{
    int a,b,c;
    bool operator < (const Node &y)const{
        if(a!=y.a)return a<y.a;
        if(b!=y.b)return b<y.b;
        return c<y.c;
    }
```

```
    bool operator == (const Node &y)const{
        return a==y.a&&b==y.b&&c==y.c;
    }
}node[MAXN],node2[MAXN];
int n,m,cnt[MAXN],ans[MAXN],c[MAXV];
int Id[MAXN],temp[MAXN];
#define lowbit(x) (x)&(-x)
void add(int pos,int v){
    while(pos<=m){
        c[pos]+=v;
        pos+=lowbit(pos);
    }
}
int query(int pos){
    int ans=0;
    while(pos){
        ans+=c[pos];
        pos-=lowbit(pos);
    }
    return ans;
}
bool cmp(int a,int b){
    if(node2[Id[a]].b!=node2[Id[b]].b)return node2[Id[a]].b<node2[Id[b]].b;
    return node2[Id[a]].c<=node2[Id[b]].c;
}
void cdq(int lef,int rig){
    if(lef==rig)return;
    int mid=lef+rig>>1;
    cdq(lef,mid);cdq(mid+1,rig);
    int pos1=lef,pos2=mid+1,pos3=lef;
    while(pos1<=mid&&pos2<=rig){
        if(cmp(pos1,pos2)){
            add(node2[Id[pos1]].c,cnt[Id[pos1]]);
            temp[pos3++]=Id[pos1++];
        }
        else{
            ans[Id[pos2]]+=query(node2[Id[pos2]].c);
            temp[pos3++]=Id[pos2++];
        }
    }
    while(pos1<=mid){
        add(node2[Id[pos1]].c,cnt[Id[pos1]]);
        temp[pos3++]=Id[pos1++];
    }
    while(pos2<=rig){
        ans[Id[pos2]]+=query(node2[Id[pos2]].c);
        temp[pos3++]=Id[pos2++];
    }
    _rep(i,lef,mid)add(node2[Id[i]].c,-cnt[Id[i]]);
    _rep(i,lef,rig)Id[i]=temp[i];
```

```
}
int f[MAXN];
int main()
{
    n=read_int(),m=read_int();
    _for(i,0,n)
    node[i].a=read_int(),node[i].b=read_int(),node[i].c=read_int();
    sort(node,node+n);
    memcpy(node2,node,sizeof(node2));
    int t=unique(node2,node2+n)-node2;
    for(int i=0,j=0;i<n;j++){
        while(node[i]==node2[j]&&i<n)
        cnt[j]++,i++;
        Id[j]=j;
    }
    cdq(0,t-1);
    _for(i,0,t)
    f[ans[i]+cnt[i]-1]+=cnt[i];
    _for(i,0,n)
    enter(f[i]);
    return 0;
}
```

题解 2

考虑两次嵌套 CDQ 分治来解决上述问题。

第一轮排序同样将第一维作为第一关键字，第二维作为第二关键字，第三维作为第三关键字。

第二轮排序将第二维作为关键字进行归并排序，同时标记每个元素第一维属于左区间还是右区间。

第三轮排序将第三维作为关键字进行归并排序，同时依照上一轮排序的元素的标记计算贡献。

时间复杂度 $O(n\log^2 n)$□理论上该方法适用于任意维偏序问题，每多一次嵌套时间复杂度增加 $O(\log n)$□

```
const int MAXN=1e5+5,MAXV=2e5+5;
struct Node{
    int a,b,c,id;
    bool lower;
    bool operator < (const Node &y)const{
        if(a!=y.a)return a<y.a;
        if(b!=y.b)return b<y.b;
        return c<y.c;
    }
    bool operator == (const Node &y)const{
        return a==y.a&&b==y.b&&c==y.c;
    }
}node[MAXN],node2[MAXN],temp[MAXN],temp2[MAXN];
int n,m,cnt[MAXN],ans[MAXN];
```

```
void cdq2(int lef,int rig){
    if(lef==rig)return;
    int mid=lef+rig>>1;
    cdq2(lef,mid);cdq2(mid+1,rig);
    int pos1=lef,pos2=mid+1,pos3=lef,s=0;
    while(pos1<=mid&&pos2<=rig){
        if(temp[pos1].c<=temp[pos2].c){
            if(temp[pos1].lower)s+=cnt[temp[pos1].id];
            temp2[pos3++]=temp[pos1++];
        }
        else{
            if(!temp[pos2].lower)ans[temp[pos2].id]+=s;
            temp2[pos3++]=temp[pos2++];
        }
    }
    while(pos1<=mid){
        if(temp[pos1].lower)s+=cnt[temp[pos1].id];
        temp2[pos3++]=temp[pos1++];
    }
    while(pos2<=rig){
        if(!temp[pos2].lower)ans[temp[pos2].id]+=s;
        temp2[pos3++]=temp[pos2++];
    }
    _rep(i,lef,rig)temp[i]=temp2[i];
}
void cdq1(int lef,int rig){
    if(lef==rig)return;
    int mid=lef+rig>>1;
    cdq1(lef,mid);cdq1(mid+1,rig);
    int pos1=lef,pos2=mid+1,pos3=lef;
    while(pos1<=mid&&pos2<=rig){
        if(node2[pos1].b<=node2[pos2].b)
        node2[pos1].lower=true,temp[pos3++]=node2[pos1++];
        else
        node2[pos2].lower=false,temp[pos3++]=node2[pos2++];
    }
    while(pos1<=mid)node2[pos1].lower=true,temp[pos3++]=node2[pos1++];
    while(pos2<=rig)node2[pos2].lower=false,temp[pos3++]=node2[pos2++];
    _rep(i,lef,rig)node2[i]=temp[i];
    cdq2(lef,rig);
}
int f[MAXN];
int main()
{
    n=read_int(),m=read_int();
    _for(i,0,n)
    node[i].a=read_int(),node[i].b=read_int(),node[i].c=read_int();
    sort(node,node+n);
    memcpy(node2,node,sizeof(node2));
    int t=unique(node2,node2+n)-node2;
```

```
    for(int i=0,j=0;i<n;j++){
        while(node[i]==node2[j]&&i<n)
        cnt[j]++,i++;
        node2[j].id=j;
    }
    cdq1(0,t-1);
    _for(i,0,t)
    f[ans[i]+cnt[i]-1]+=cnt[i];
    _for(i,0,n)
    enter(f[i]);
    return 0;
}
```

算法练习

习题一

题意

题解