P3810 【模板】三维偏序（陌上花开） 题解

返回题目

本题目已不接受新题解提交

当题目的题解数量、做题思路已足够丰富，题目过于简单，或处于月赛保护期时，题解提交入口会被关闭。

不要把题解发布在题目讨论区。

题解仅供学习参考使用

抄袭、复制题解，以达到刷 AC 率/AC 数量或其他目的的行为，在洛谷是严格禁止的。

洛谷非常重视**学术诚信**。此类行为将会导致您成为**作弊者**。 具体细则请查看洛谷社区规则。

提交题解前请务必阅读《洛谷主题库题解规范》。

应用 »
题库
题单
比赛
记录
讨论

35 篇题解

默认排序 按E

echo6342 创建时间：2018-02-27 15:46:57

在 Ta 的

这并不是对劲的cdq分治......

如果想看更不对劲的，点这里-> :-)

cdq分治每次计算前一半对后一半的影响。具体地， 假设三维分别是x,y,z，先按x排序。分治时每次将前半边、后半边分别按y排序。虽然现 序被打乱了，但是前半边还是都小于后半边的，所以要是只计算前半边对后半边的偏序关系，是不会受到x的影响的。维护后一半的指针i，前 指针j，每次将i后移一位时，若y[j]<=y[i]则不断后移j，并不断将z[j]加入树状数组。然后再查询树状数组中有多少数小于等于z[i]。 最后要清 数组。

它有那么一些些眼熟，解一维偏序时就是归什么排序。

```
#include<iostream>
#include<iomanip>
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<cmath>
#include<algorithm>
#define maxn 100010
#define maxk 200010
#define ll long long
using namespace std;
inline int read()
{
    int x=0,f=1;
    char ch=getchar();
    while(isdigit(ch)==0 && ch!='-')ch=getchar();
    if(ch=='-')f=-1,ch=getchar();
    while(isdigit(ch))x=x*10+ch-'0',ch=getchar();
    return x*f;
}
inline void write(int x)
{
    int f=0;char ch[20];
    if(!x){puts("0");return;}
    if(x<0){putchar('-');x=-x;}
    while(x)ch[++f]=x%10+'0',x/=10;
    while(f)putchar(ch[f--]);
    putchar('\n');
}
typedef struct node
{
    int x,y,z,ans,w;
}stnd;
stnd a[maxn],b[maxn];
int n,cnt[maxk];
int k,n_;
bool cmpx(stnd u,stnd v)
{
    if(u.x==v.x)
    {
        if(u.y==v.y)
            return u.z<v.z;
        return u.y<v.y;
    }
    return u.x<v.x;
}
bool cmpy(stnd u,stnd v)
{
    if(u.y==v.y)
        return u.z<v.z;
    return u.y<v.y;
}
struct treearray
{
    int tre[maxk],kk;
    int lwbt(int x){return x&(-x);}
```

```
    int ask(int i){int ans=0; for(;i;i-=lwbt(i))ans+=tre[i];return ans;}
    void add(int i,int k){for(;i<=kk;i+=lwbt(i))tre[i]+=k;}
}t;
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+mid+1,cmpy);
    sort(a+mid+1,a+r+1,cmpy);
    int i=mid+1,j=l;
    for(;i<=r;i++)
    {
        while(a[j].y<=a[i].y && j<=mid)
            t.add(a[j].z,a[j].w),j++;
        a[i].ans+=t.ask(a[i].z);
    }
    for(i=l;i<j;i++)
        t.add(a[i].z,-a[i].w);
}
int main()
{
    n_=read(),k=read();t.kk=k;
    for(int i=1;i<=n_;i++)
        b[i].x=read(),b[i].y=read(),b[i].z=read();
    sort(b+1,b+n_+1,cmpx);
    int c=0;
    for(int i=1;i<=n_;i++)
    {
        c++;
        if(b[i].x!=b[i+1].x || b[i].y!=b[i+1].y || b[i].z!=b[i+1].z )
            a[++n]=b[i],a[n].w=c,c=0;
    }
    cdq(1,n);
    for(int i=1;i<=n;i++)
        cnt[a[i].ans+a[i].w-1]+=a[i].w;
    for(int i=0;i<n_;i++)
        write(cnt[i]);
    return 0;
}
```


Shadows 创建时间：2018-01-10 10:27:55 在 Ta 的

其实cdq分治可以一直嵌套下去，不一定需要数据结构维护

这样1d 排序，2d cdq，3d cdq统计答案，推而广之，四维偏序也是一样道理，用cdq统计答案，和用cdq外层嵌套几乎一样操作

```
#include<cstdio>
#include<cstring>
#include<iostream>
#include<algorithm>
#define maxn 100010
using namespace std;
int n,k,ans[maxn]={0},d[maxn]={0};
struct node{
    int x,y,z;
    bool b;
    int *ans;
    inline void get(){
        scanf("%d%d%d",&x,&y,&z);
        return;
    }
    bool operator==(const node &a)
    const{
        return x==a.x&&y==a.y&&z==a.z;
    }
}a[maxn],b[maxn],c[maxn];
inline bool cmp(const node &a, const node &b)  {
    return a.x<b.x||a.x==b.x&&a.y<b.y||a.x==b.x&&a.y==b.y&&a.z<b.z;
```

应用 »
题库
题单
比赛
记录
讨论

```
    }
    void merge2(int l,int r){
        if(l==r)return;
        int mid=(l+r)>>1;
        merge2(l,mid);
        merge2(mid+1,r);
        for(int i=l,j=l,k=mid+1,cnt=0;i<=r;++i){
            if((k>r||b[j].z<=b[k].z)&&j<=mid)
            c[i]=b[j++],cnt+=c[i].b;
            else{
                c[i]=b[k++];
                if(!c[i].b)*c[i].ans+=cnt;
            }
        }
        for(int i=l;i<=r;++i)b[i]=c[i];
    }
    void merge1(int l,int r){
        if(l==r)return;
        int mid=(l+r)>>1;
        merge1(l,mid);
        merge1(mid+1,r);
        for(int i=l,j=l,k=mid+1;i<=r;++i){
            if((k>r||a[j].y<=a[k].y)&&j<=mid)
            b[i]=a[j++],b[i].b=1;
            else
            b[i]=a[k++],b[i].b=0;
        }
        for(int i=l;i<=r;++i)a[i]=b[i];
        merge2(l,r);
    }
    int main(){
        scanf("%d%d",&n,&k);
        for(int i=1;i<=n;++i)
        a[i].get(),a[i].ans=&ans[i],ans[i]=0;
        sort(a+1,a+n+1,cmp);
        for(int i=n-1;i;--i)
        if(a[i]==a[i+1])
        *a[i].ans=*a[i+1].ans+1;
        merge1(1,n);
        for(int i=1;i<=n;++i)++d[ans[i]];
        for(int i=0;i<n;++i)
        printf("%d\n",d[i]);
        return 0;
}
```

139 57 条评论

FlashHu 创建时间：2018-07-28 10:38:41 在 Ta 的

安利蒟蒻CDQ分治总结

分治就是分治，“分而治之”的思想。

那为什么会有CDQ分治这样的称呼呢？

这一类分治有一个重要的思想——用一个子问题来计算对另一个子问题的贡献。

有了这种思想，就可以方便地解决更复杂的问题。

这样一句话怎样理解好呢？还是做做题目吧。

三维偏序问题，即给出若干元素，每个元素有三个属性值a,b,c，询问对于每个元素i，满足$a_j \le a_i, b_j \le b_i, c_j \le c_i$的$j$的个数

不用着急，先从简单的问题开始

试想一下二位偏序也就是$a_j \le a_i, b_j \le b_i$怎么做

先按a为第一关键字，b为第二关键字排序，那么我们就保证了第一维a的有序。

于是，对于每一个i，只可能1到$i-1$的元素会对它有贡献，那么直接查1到$i-1$的元素中满足$b_j \le b_i$的元素个数。

具体实现？动态维护b的树状数组，从前到后扫一遍好啦，$O(n \log n)$。

那么三维偏序呢？我们只有在保证前两位都满足的情况下才能计算答案了。

仍然按a为第一关键字，b为第二关键字，c为第三关键字排序，第一维保证左边小于等于右边了。

为了保证第二维也是左边小于等于右边，我们还需要排序。

想到归并排序是一个分治的过程，我们可不可以在归并的过程中，统计出在子问题中产生的对答案贡献呢？

现在我们有一个序列，我们把它递归分成两个子问题，子问题进行完归并排序，已经保证b有序。此时，两个子问题间有一个分界线，原来第
边小于等于右边，所以现在分界线左边的任意一个的a当然还是都小于右边的任意一个。那不等于说，只有分界线左边的能对右边的产生贡献

于是，问题降到了二维。我们就可以排序了，归并排序（左边的指针为j，右边的为i）并维护c的树状数组，如果当前$b_j \le b_i$，说明j可以对
入的满足$c_j \le c_i$的i产生贡献了，把c_j加入树状数组；否则，因为后面加入的j都不会对i产生贡献了，所以就要统计之前被给的所有贡献了，
状数组c_i的前缀和。

这是在分治中统计的子问题的答案，跟总答案有怎样的关系呢？容易发现，每个子问题统计的只有跨越分界线的贡献，反过来看，每一个能
献的i, j，有且仅有一个子问题，两者既同时被包含，又在分界线的异侧。那么所有子问题的贡献加起来就是总答案。

算法的大致思路就是这样啦。至于复杂度，$T(n) = O(n \log k) + 2T(\frac{2}{n}) = O(n \log n \log k)$。

当然还有不少细节问题。

最大的问题就在于，可能有完全相同的元素。这样的话，本来它们相互之间都有贡献，可是cdq的过程中只有左边的能贡献右边的。这可怎么

我们把序列去重，这样现在就没有相同的了。给现在的每个元素一个权值v等于出现的次数。中间的具体实现过程也稍有变化，在树状数组中
值是v而不是1了，最后统计答案时，也要算上相同元素内部的贡献，`ans+=v-1`。

写法上，为了防止sort和归并排序中空间移动太频繁，没有对每个元素封struct，这样的话就要膜改一下cmp函数（蒟蒻也是第一次发现cm
么写）

蒟蒻还是觉得开区间好写一些吧。。。当然闭区间好理解些。。。

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#define RG register
#define R RG int
using namespace std;
const int N=1e5+9,SZ=2.2e6;
char buf[SZ],*pp=buf-1;//fread必备
int k,a[N],b[N],c[N],p[N],q[N],v[N],cnt[N],ans[N],*e;
inline int in(){
    while(*++pp<'-');
    R x=*pp&15;
    while(*++pp>'-')x=x*10+(*pp&15);
    return x;
}
void out(R x){
    if(x>9)out(x/10);
    *++pp=x%10|'0';
}
inline bool cmp(R x,R y){//直接对数组排序，注意三关键字
    return a[x]<a[y]||(a[x]==a[y]&&(b[x]<b[y]||(b[x]==b[y]&&c[x]<c[y])));
}
inline void upd(R i,R v){//树状数组修改
    for(;i<=k;i+=i&-i)e[i]+=v;
}
inline int ask(R i){//树状数组查前缀和
    R v=0;
    for(;i;i-=i&-i)v+=e[i];
    return v;
}
void cdq(R*p,R n){//处理一个长度为n的子问题
    if(n==1)return;
    R m=n>>1,i,j,k;
    cdq(p,m);cdq(p+m,n-m);//递归处理
    memcpy(q,p,n<<2);//归并排序
    for(k=i=0,j=m;i<m&&j<n;++k){
        R x=q[i],y=q[j];
        if(b[x]<=b[y])upd(c[p[k]=x],v[x]),++i;//左边小，插入
        else  cnt[y]+=ask(c[p[k]=y])      ,++j;//右边小，算贡献
    }
    for(;j<n;++j)cnt[q[j]]+=ask(c[q[j]]);//注意此时可能没有完成统计
    memcpy(p+k,q+i,(m-i)<<2);
    for(--i;~i;--i)upd(c[q[i]],-v[q[i]]);//必须这样还原树状数组，memset是O(n^2)的
```

```
}
int main(){
    fread(buf,1,SZ,stdin);
    R n=in(),i,j;k=in();e=new int[k+9];
    for(i=0;i<n;++i)
        p[i]=i,a[i]=in(),b[i]=in(),c[i]=in();
    sort(p,p+n,cmp);
    for(i=1,j=0;i<n;++i){
        R x=p[i],y=p[j];++v[y];//模仿unique双指针去重，统计v
        if(a[x]^a[y]||b[x]^b[y]||c[x]^c[y])p[++j]=x;
    }
    ++v[p[j++]];
    cdq(p,j);
    for(i=0;i<j;++i)
        ans[cnt[p[i]]+v[p[i]]-1]+=v[p[i]];//答案算好
    for(pp=buf-1,i=0;i<n;++i)
        out(ans[i]),*++pp='\n';
    fwrite(buf,1,pp-buf+1,stdout);
}
```

124 18 条评论

1LoveNozomi 创建时间：2019-02-26 13:42:00 在 Ta 的

这里都没有bitset的写法，虽然bitset不能够过这道题目，但是对于维数更高或者要求强制在线的题目，bitset有很大优势，我们应该学习。是算法中使用的数据结构而已，这个算法的本质是暴力。然后用分块优化暴力。 这个算法能求出每个元素能与其他几个元素构成偏序关系，个元素，这个暴力算法的做法是：

1. 对于除自己以外的所有元素，把属性a的值小于等于自己的属性a的值的元素的编号组成一个集合。（反过来说，这个集合里存放的都是元号，里面的编号指向的元素的属性a的值都小于等于自己的属性a的值，且编号不在此集合的元素的属性a的值一定大于自己的属性a的值）
2. 同理构造b集合
3. 同理构造c集合
4. ……（多少种属性我们就处理出多少个集合。把元素的编号大小当作属性值也可以。）
5. 把构造的所有集合交在一起，得到的集合里面的每个编号对应的元素都和自己构成偏序关系。因为能够在交集中出现的元素，必定其每个都不超过自己对应的每个属性值，符合题意。

下面是代码：

```
#include <bits/stdc++.h>
using namespace std;
const int maxn=100000,maxk=3,maxblock=320,inf=0x3f3f3f3f;
int maxsize,n,k,length,belong[maxn+5],ans[maxn+5],temp[maxn+5];
struct Demension {
    int value[maxn+5],size;
    vector<int> list[maxn+5];
    bitset<maxn+5> PreBlock[maxblock+5];//储存前i个块的答案
    void Discretization() {//将元素在当前维度的属性值离散化
        int i;
        memcpy(temp+1,value+1,n<<2);
        sort(temp+1,temp+1+n);
        size=unique(temp+1,temp+1+n)-(temp+1);
        maxsize=max(maxsize,size);
        for(i=1; i<=size; ++i)
            list[i].clear();
        for(i=1; i<=n; ++i)
            list[(value[i]=lower_bound(temp+1,temp+1+size,value[i])-temp)].push_back(i);
    }
    bitset<maxn+5> Get(int p) {//当前维度下，属性值小于元素p的属性值的元素的编号集合
        int i,j,temp;
        p=value[p];
        bitset<maxn+5> res=PreBlock[belong[p]-1];
        for(i=(belong[p]-1)*length+1; i<=p; ++i) {
            temp=list[i].size();
            for(j=0; j<temp; ++j)
                res.set(list[i][j]);
        }
        return res;
    }
} demension[maxk+5];
inline void Calc();
inline void Input();
```

应用 »
题库
题单
比赛
记录
讨论

```
inline void Work();
inline void Read(int &x);
void Print(int x);
int main() {
    Input();
    Calc();
    Work();
    return 0;
}
void Calc() {
    int i,j,t,m;
    bitset<maxn+5> temp;
    for(i=1; i<=k; ++i)
        demension[i].Discretization();//离散化
    length=sqrt(maxsize);
    for(i=1; i<=maxsize; ++i)
        belong[i]=(i-1)/length+1;//值域分块
    for(i=1; i<=k; ++i) {
        demension[i].PreBlock[0].reset();
        temp.reset();
        for(j=1; j<=demension[i].size; ++j) {
            m=demension[i].list[j].size();
            for(t=0; t<m; ++t)
                temp.set(demension[i].list[j][t]);
            if(belong[j]^belong[j+1])//权值数组，把属性值的范围进行分块,第i块储存了前i块的信息
                demension[i].PreBlock[belong[j]]=temp;
        }
    }
}
void Work() {
    int i,j;
    bitset<maxn+5> temp;
    for(i=1; i<=n; ++i) {
        temp.set();
        for(j=1; j<=k; ++j)
            temp&=demension[j].Get(i);
        ++ans[temp.count()-1];
    }
    for(i=0; i<n; ++i) {
        Print(ans[i]);
        putchar('\n');
    }
}
inline void Input() {
    int i,j;
    scanf("%d%d",&n,&k);//n个元素，k个属性
    k=3;
    for(i=1; i<=n; ++i)
        for(j=1; j<=k; ++j)
            Read(demension[j].value[i]);
}
inline void Read(int &x) {
    x=0;
    char c=getchar();
    while(!isdigit(c))
        c=getchar();
    do {
        x=(x<<3)+(x<<1)+(48^c);
        c=getchar();
    } while(isdigit(c));
}
void Print(int x) {
    if(x>9)
        Print(x/10);
    putchar(48^(x%10));
}
```

Update 2019.10.27:修改原代码的码风

38 5 条评论

撤云 创建时间：2018-12-18 16:51:47 在 Ta 的

戳我食用效果更加 ####前言

戳我食用效果更加 ####前言

> 陌上花开，可缓缓归矣　　　　　　——吴越王

应用 »
题库
题单
比赛
记录
讨论

1. 寓意：意思是：田间阡陌上的花开了，你可以一边赏花，一边慢慢回来。
2. 隐意:春天都到了，你怎么还没有回来。形容吴越王期盼夫人早日归来的急切心情。

Ask:那么这和cdq有什么关系呢?
Answer:~~并没有什么关系，增强语文水平而已~~，现在来看一到题目:陌上花开。这就有关系了吧。

题目大意是:有n个元素，第i个元素有a_i, b_i, c_i三个属性，设$f(i)$表示满足$a_j \leq a_i$且$b_j \leq b_i$且$c_j \leq c_i$的j的数量。求$f(i) = d$的数量$d \in [$

做法1：暴力 $O(n^2)$的扫一遍求一下就好了。

```
#include<bits/stdc++.h>
int k,n,f[200001],a[200001],b[200001],c[200001],ans;
int main(){
    scanf("%d%d",&n,&k);
    for(int i=1;i<=n;i++)
        scanf("%d%d%d",&a[i],&b[i],&c[i]);
    for(int i=1;i<=n;i++){
        for(int j=1;j<=n;j++)
            if(a[j]<=a[i]&&b[j]<=b[i]&&c[j]<=c[i]&&i!=j)
                ans++;
        f[ans]++,ans=0;
    }
    for(int i=0;i<n;i++)
        printf("%d\n",f[i]);
}
```

这个应该不需要多讲吧,普及组的难度，但不要说不需要，你在对拍的时候就需要他了

做法2： K-DTree

~~不会,我tcl~~

做法三：cdq分治

现在来正式讲一讲cdq分治

cdq**分治**

####前置要求：

1. 树状数组
2. 基础分治
3. 树状数组求逆序对

逆序对的问题是二维的,我们只需要讲一维排序，然后在用树状数组维护即可。
那么对于三维的陌上花开呢?我们还是可以用这个方法，首先先将数列按第一位排序，这样我们只需要考虑两维的情况。于是我们可以分治做
某一个序列$[l,r]$,分成段$[l,mid]$和$[mid+1,r]$,然后在对$[l,r]$这段区间的第二维进行排序。若点在排序前属于$[l,mid]$,树状数组单点修改；
点在排序前属于$[m+1,r]$,便统计一次。(其实就是类似于树状数组求逆序对的操作)

一定要记得去重，否则会出事的

code

```
#include<bits/stdc++.h>
using namespace std;
const int N=200001;
struct node{
    int x,y,z,id;
}a[N];
int c[N<<2],k,n,b[N],bj[N],f[N];
int lowbit(int x){
    return x&(-x);
}
int read(){
    int x=0,f=1;
    char c=getchar();
    while(c<'0'||c>'9') f=(c=='-')?-1:1,c=getchar();
    while(c>='0'&&c<='9') x=x*10+c-48,c=getchar();
```

应用 »
题库
题单
比赛
记录
讨论

```
    return f*x;
}
void add(int x,int v){
    while(x<=k)
        c[x]+=v,x+=lowbit(x);
}
int sum(int x){
    int ans=0;
    while(x)
        ans+=c[x],x-=lowbit(x);
    return ans;
}
bool cmp1(const node & a , const node & b ){
    if(a.x!=b.x)
        return a.x<b.x;
    if(a.y!=b.y)
        return a.y<b.y;
    return a.z<b.z;
}
bool cmp2(const node & a , const node & b ){
    if(a.y!=b.y)
        return a.y<b.y;
    if(a.z!=b.z)
        return a.z<b.z;
    return a.x<b.x;
}
void cdq(int l,int r){
    if(l==r)
        return ;
    int mid=(l+r)>>1,flag;
    cdq(l,mid),cdq(mid+1,r);
    sort(a+l,a+r+1,cmp2);
    for(int i=l;i<=r;i++)
        (a[i].x<=mid)?add(a[i].z,1),flag=i:b[a[i].id]+=sum(a[i].z);
    for(int i=l;i<=r;i++)
        if(a[i].x<=mid)
            add(a[i].z,-1);
}
int main(){
    n=read(),k=read();
    for(int i=1;i<=n;i++)
        a[i].x=read(),a[i].y=read(),a[i].z=read(),a[i].id=i;
    sort(a+1,a+1+n,cmp1);
    for(int i=1;i<=n;){
        int j=i+1;
        while(j<=n&&a[j].x==a[i].x&&a[j].y==a[i].y&&a[j].z==a[i].z)
            j++;
        while(i<j)
            bj[a[i].id]=a[j-1].id,i++;
    }
    for(int i=1;i<=n;i++)
        a[i].x=i;
    cdq(1,n);
    for(int i=1;i<=n;i++)
        f[b[bj[a[i].id]]]++;
    for(int i=0;i<n;i++)
        printf("%d\n",f[i]);
}
```

41 15 条评论

Ireliaღ 创建时间：2019-07-19 15:11:50 在 Ta 的!

树状数组套值域线段树

三维偏序

看到三维偏序，最简单直接的思路就是先按照x把元素排序，然后按顺序把每个元素放进二维数据结构，统计二维前缀和，即为x、y、z都不
元素的元素个数。

以下为二维线段树（树套树）代码

应用 »
题库
题单
比赛
记录
讨论

```
// luogu-judger-enable-o2
#include <iostream>
#include <algorithm>

using namespace std;

const int MAXN = 1e5 + 5;
const int MAXK = 2e5;

int n, k, cnt[MAXN];

struct Data{
    int x, y, z;

    int operator < (const Data &o) const {
        return x != o.x ? (x < o.x) : (y != o.y ? (y < o.y) : (z < o.z));
    }

    int operator == (const Data &o) const {
        return x == o.x && y == o.y && z == o.z;
    }
}data[MAXN];

struct Seg{
    struct Node{
        int val;
        Node *ch[2];

        Node(int val = 0) : val(val) {
            ch[0] = ch[1] = NULL;
        }
    };

    Node *rt;

    Seg() {
        rt = NULL;
    }

    void Modify(Node *&now, int pos, int val = 1, int nl = 1, int nr = MAXK) {
        if (!now) now = new Node();
        if (nl == nr) {
            now->val += val;
            return;
        }
        int mid = nl + nr >> 1;
        if (pos <= mid) Modify(now->ch[0], pos, val, nl, mid);
        else Modify(now->ch[1], pos, val, mid + 1, nr);
        now->val = (now->ch[0] ? now->ch[0]->val : 0) + (now->ch[1] ? now->ch[1]->val : 0);
    }

    int Query(Node *now, int l, int r, int nl = 1, int nr = MAXK) {
        if (!now) return 0;
        if (l == nl && r == nr) return now->val;
        int mid = nl + nr >> 1;
        if (r <= mid) return Query(now->ch[0], l, r, nl, mid);
        else if (l > mid) return Query(now->ch[1], l, r, mid + 1, nr);
        return Query(now->ch[0], l, mid, nl, mid) + Query(now->ch[1], mid + 1, r, mid + 1, nr);
    }
};

Seg tree[MAXK * 4 + 5];

void Modify(int now, int posx, int posy, int val, int nl = 1, int nr = MAXK) {
    tree[now].Modify(tree[now].rt, posy, val);
    if (nl == nr) return;
    int mid = nl + nr >> 1;
    if (posx <= mid) Modify(now << 1, posx, posy, val, nl, mid);
    else Modify(now << 1 | 1, posx, posy, val, mid + 1, nr);
}

int Query(int now, int xl, int xr, int yl, int yr, int nl = 1, int nr = MAXK) {
    if (xl == nl && xr == nr) return tree[now].Query(tree[now].rt, yl, yr);
    int mid = nl + nr >> 1;
```

```
        if (xr <= mid) return Query(now << 1, xl, xr, yl, yr, nl, mid);
        else if (nl > mid) return Query(now << 1 | 1, xl, xr, yl, yr, mid + 1, nr);
        return Query(now << 1, xl, mid, yl, yr, nl, mid) + Query(now << 1 | 1, mid + 1, xr, yl, yr, mid + 1, nr);
    }

int main() {
    cin >> n >> k;
    for (int i = 1; i <= n; i++) cin >> data[i].x >> data[i].y >> data[i].z;
    sort(data + 1, data + n + 1);
    int sum = 1;
    for (int i = 1; i <= n; i++) {
        if (data[i + 1] == data[i]) {
            sum++;
            continue;
        }
        Modify(1, data[i].y, data[i].z, sum);
        int res = Query(1, 1, data[i].y, 1, data[i].z);
        cnt[res] += sum;
        sum = 1;
    }
    for (int i = 1; i <= n; i++) cout << cnt[i] << endl;
    return 0;
}
```

你 ~~`Ctrl+C` 一 `Ctrl+v` 交上去，发现TLE70~~

考虑到值域不大，并且线段树常数较大，可以把外层的非动态开点线段树换成树状数组，减小常数，以下为AC代码。

```
// luogu-judger-enable-o2
#include <iostream>
#include <algorithm>

using namespace std;

const int MAXN = 1e5 + 5;
const int MAXK = 2e5;

int n, k, cnt[MAXN];

struct Data{
    int x, y, z;

    int operator < (const Data &o) const {
        return x != o.x ? (x < o.x) : (y != o.y ? (y < o.y) : (z < o.z));
    }

    int operator == (const Data &o) const {
        return x == o.x && y == o.y && z == o.z;
    }
}data[MAXN];

struct Seg{
    struct Node{
        int val;
        Node *ch[2];

        Node(int val = 0) : val(val) {
            ch[0] = ch[1] = NULL;
        }
    };

    Node *rt;

    Seg() {
        rt = NULL;
    }

    void Modify(Node *&now, int pos, int val, int nl, int nr) {
        if (!now) now = new Node();
        if (nl == nr) {
            now->val += val;
            return;
        }
        int mid = nl + nr >> 1;
```

应用 »
题库
题单
比赛
记录
讨论

```
        if (pos <= mid) Modify(now->ch[0], pos, val, nl, mid);
        else Modify(now->ch[1], pos, val, mid + 1, nr);
        now->val = (now->ch[0] ? now->ch[0]->val : 0) + (now->ch[1] ? now->ch[1]->val : 0);
    }

    int Query(Node *now, int l, int r, int nl, int nr) {
        if (!now) return 0;
        if (l == nl && r == nr) return now->val;
        int mid = nl + nr >> 1;
        if (r <= mid) return Query(now->ch[0], l, r, nl, mid);
        else if (l > mid) return Query(now->ch[1], l, r, mid + 1, nr);
        return Query(now->ch[0], l, mid, nl, mid) + Query(now->ch[1], mid + 1, r, mid + 1, nr);
    }
};
/*
Seg tree[MAXK * 4 + 5];

void Modify(int now, int posx, int posy, int val, int nl = 1, int nr = MAXK) {
    tree[now].Modify(tree[now].rt, posy, val);
    if (nl == nr) return;
    int mid = nl + nr >> 1;
    if (posx <= mid) Modify(now << 1, posx, posy, val, nl, mid);
    else Modify(now << 1 | 1, posx, posy, val, mid + 1, nr);
}

int Query(int now, int xl, int xr, int yl, int yr, int nl = 1, int nr = MAXK) {
    if (xl == nl && xr == nr) return tree[now].Query(tree[now].rt, yl, yr);
    int mid = nl + nr >> 1;
    if (xr <= mid) return Query(now << 1, xl, xr, yl, yr, nl, mid);
    else if (nl > mid) return Query(now << 1 | 1, xl, xr, yl, yr, mid + 1, nr);
    return Query(now << 1, xl, mid, yl, yr, nl, mid) + Query(now << 1 | 1, mid + 1, xr, yl, yr, mid + 1, nr);
}
*/

Seg tree[MAXK + 5];

int LB(int x) {
    return x & (-x);
}

void Modify(int posx, int posy, int val) {
    for (int i = posx; i <= k; i += LB(i))
        tree[i].Modify(tree[i].rt, posy, val, 1, k);
}

int Query(int x, int y) {
    int ret = 0;
    for (int i = x; i; i -= LB(i)) ret += tree[i].Query(tree[i].rt, 1, y, 1, k);
    return ret;
}

int main() {
    cin >> n >> k;
    for (int i = 1; i <= n; i++) cin >> data[i].x >> data[i].y >> data[i].z;
    sort(data + 1, data + n + 1);
    int sum = 1;
    for (int i = 1; i <= n; i++) {
        if (data[i + 1] == data[i]) {
            sum++;
            continue;
        }
        Modify(data[i].y, data[i].z, sum);
        int res = Query(data[i].y, data[i].z);
        cnt[res] += sum;
        sum = 1;
    }
    for (int i = 1; i <= n; i++) cout << cnt[i] << endl;
    return 0;
}
```

29   5 条评论

于丰林 创建时间：2019-02-18 14:49:27 在 Ta 的

应用 »
题库
题单
比赛
记录
讨论

作为CDQ的一道板子题，我们先来明确一下到底什么是CDQ?

CDQ通俗一点说就是三句话：

1.递归前一半

2.判断前一半对于后一半的影响

3.递归后一半

那么我们来看这道经典的三维偏序题，说一下大体的思路：

首先我们可以以x作为第一关键字进行排序，（y和z作为次要关键字），这样我们就可以保证从左到右是以x单调递增的。

然后我们将这个区间一分为二，分别对于每一个序列以y作为第一关键字排序，这样一来我们得到的两个序列拥有以下的性质：

1.左序列的任意x值都小于右序列的任意x值

2.每一个序列里的y值都是单调递增的

最后我们分别在序列头和序列中点设定两个指针j和i，对于每一个i，j都从上一个不满足条件的地方开始枚举，将j对应的y小于i对应的y的j加入
组中，最后只要判断z值是否满足条件就可以了。

当然，这道题还有一些小细节：

1.由于原序列可能会有重复的数，我们要先进行去重，并储存每一个数都代表了几个相同的数。

2.每一次i的增加都要清空树状数组，只要加上之前加的相反数就可以了。

好像也就这些了。。。

最后，附上本题代码：

```
#include<cstdio>
#include<algorithm>
#define maxn 100010
#define maxk 200010
using namespace std;
struct node
{
    int x,y,z,ans,w;
};
node a[maxn],b[maxn];
int n,cnt[maxk];
int k,z;
bool cmp1(node u,node v)
{
    if(u.x==v.x)
    {
        if(u.y==v.y)
            return u.z<v.z;
        return u.y<v.y;
    }
    return u.x<v.x;
}
bool cmp2(node u,node v)
{
    if(u.y==v.y)
        return u.z<v.z;
    return u.y<v.y;
}
struct TREE
{
    int tre[maxk],kk;
    int lowbit(int x)
    {
        return x&(-x);
    }
    int ask(int i)
    {
        int ans=0;
        for(; i; i-=lowbit(i))
        {
            ans+=tre[i];
```

应用 »
题库
题单
比赛
记录
讨论

```
        }
        return ans;
    }
    void add(int i,int k)
    {
        for(; i<=kk; i+=lowbit(i))
        {
            tre[i]+=k;
        }
    }
} t;
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+mid+1,cmp2);
    sort(a+mid+1,a+r+1,cmp2);
    int i=mid+1,j=l;
    while(i<=r)
    {
        while(a[j].y<=a[i].y && j<=mid)
        {
            t.add(a[j].z,a[j].w);
            j++;
        }
        a[i].ans+=t.ask(a[i].z);
        i++;
    }
    for(i=l; i<j; i++)
    {
        t.add(a[i].z,-a[i].w);
    }
}
int main()
{
    scanf("%d%d",&z,&k);//z是数量，k是最大属性值
    t.kk=k;//设定上限，t是维护的树状数组
    for(int i=1; i<=z; i++)
    {
        scanf("%d%d%d",&b[i].x,&b[i].y,&b[i].z);//
    }
    sort(b+1,b+z+1,cmp1);//排序
    int c=0;
    for(int i=1; i<=z; i++)
    {
        c++;
        if(b[i].x!=b[i+1].x || b[i].y!=b[i+1].y || b[i].z!=b[i+1].z )
            a[++n]=b[i],a[n].w=c,c=0;
    }//去重
    cdq(1,n);//cdqaq
    for(int i=1; i<=n; i++)
    {
        cnt[a[i].ans+a[i].w-1]+=a[i].w;//这个地方不太好理解：cnt【x】就是储存f【i】= x的个数，x就等于i的答案加上它重复的个数（可以取等）减
    }
    for(int i=0; i<z; i++)
    {
        printf("%d\n",cnt[i]);
    }
    return 0;
}
```

28 11 条评论

Henry_he 创建时间：2018-02-04 16:46:08 在 Ta 的

为什么到紫题就很少有pascal题解了呢qwq

那我来写一份比较详细的吧（毕竟是算法模板题）

应用 »
题库
题单
比赛
记录
讨论

这道题是三维偏序的模板

1D 我们对第一维直接进行排序就好啦，这样可以保证第一维有序

2D: 这里我采用了cdq分治

cdq分治是一种神奇的分治算法

例如本题，将第二维的区间采取二分，首先递归计算左边内部对自己的贡献，然后计算左边对右边的贡献，最后递归计算右边自己内部贡献

所以重点在如何处理左边对右边的贡献

假设左边区间为[l,mid] 右边区间为[mid+1.r] (推荐使用闭区间，比较好判断边界~~

我们先将两个区间排序，注意要打上序号，这样等会才方便判断这个数是在左区间还是在右区间

然后将数组扫一遍，每当我们遇见左区间的数时统计下来，遇到右区间的数就将前面的左区间的数统计到该数的答案里

因为，左区间的数第一维一定小于右区间，而上述操作便统计了第二维也小的数

当然，到此为止是二维偏序的做法，而本题是三维偏序(见下面

3D 第三维当然也可以再套一层cdq分治，但推荐使用树状数组来维护,因为树状数组常数小，而二维树状数组额外空间过大，会MLE,至于tre
不会打qwq

接着第二步说，由于第三维的限制，不一定第一维，第二维都满足条件就可以， 所以我们用树状数组来维护统计答案步骤~~~

当我们访问到左区间的点时，我们将他的第三维对应的点加1，来统计左区间中第三维为此数的点，当我们访问到右区间的点时，就查询树状
来找到第三维也小的左节点数量

嗯就这样啦~~~

附码

```
var n,k,i:longint;
    x,y,z,f,b,s,p,m:array[0..100000]of longint;
    c:array[1..200000]of longint;
procedure sort(l,r:longint);
var a,b,i,j,c,d:longint;
begin
  a:=x[(l+r)div 2];
  c:=y[(l+r)div 2];
  d:=z[(l+r)div 2];
  i:=l;
  j:=r;
  repeat
    while (x[i]<a)or((x[i]=a)and(y[i]<c))or((x[i]=a)and(y[i]=c)and(z[i]<d)) do
      inc(i);
    while (x[j]>a)or((x[j]=a)and(y[j]>c))or((x[j]=a)and(y[j]=c)and(z[j]>d)) do
      dec(j);
    if i<=j then
    begin
      b:=x[i];
      x[i]:=x[j];
      x[j]:=b;
      b:=y[i];
      y[i]:=y[j];
      y[j]:=b;
      b:=z[i];
      z[i]:=z[j];
      z[j]:=b;
      inc(i);
      dec(j);
    end;
  until i>j;
  if i<r then sort(i,r);
  if l<j then sort(l,j);
end;
procedure sort2(l,r:longint);
var x,y,z,i,j:longint;
begin
  x:=b[(l+r)div 2];
  z:=s[(l+r)div 2];
  i:=l;
  j:=r;
```

```
    repeat
      while (b[i]<x)or((b[i]=x)and(s[i]<z)) do
        inc(i);
      while (x<b[j])or((b[j]=x)and(s[j]>z)) do
        dec(j);
      if i<=j then
      begin
        y:=b[i];
        b[i]:=b[j];
        b[j]:=y;
        y:=s[i];
        s[i]:=s[j];
        s[j]:=y;
        inc(i);
        dec(j);
      end;
    until i>j;
    if i<r then sort2(i,r);
    if l<j then sort2(l,j);
  end;
  procedure ins(x,y:longint);
  begin
    while x<=k do
    begin
      inc(c[x],y);
      x:=x+(x and -x);
    end;
  end;
  function query(x:longint):longint;
  begin
    query:=0;
    while x>0 do
    begin
      inc(query,c[x]);
      x:=x-(x and -x);
    end;
  end;
  procedure cdq(l,r:longint);
  var mid,i,j:longint;
  begin
    if l>=r then exit;
    mid:=(l+r)div 2;
    cdq(l,mid);//递归左区间的答案
    for i:=l to r do//标记+排序
    begin
      b[i]:=y[i];
      s[i]:=i;
    end;
    sort2(l,r);
    for i:=l to r do
    begin
      if s[i]=n then
      begin
        s[i]:=s[i];
      end;
      if s[i]<=mid then ins(z[s[i]],1);
      //如果是左区间的点，树状数组统计
      if s[i]>mid then f[s[i]]:=f[s[i]]+query(z[s[i]]);
      //右区间的点查询统计
    end;
    for i:=l to r do
      if s[i]<=mid then ins(z[s[i]],-1);
    cdq(mid+1,r);//递归右区间
  end;
  begin
    readln(n,k);
    for i:=1 to n do
      readln(x[i],y[i],z[i]);
    sort(1,n);//第一维排序啦啦啦
    cdq(1,n);//开始分治
    for i:=n-1 downto 1 do
      if (x[i]=x[i+1])and(y[i]=y[i+1])and(z[i]=z[i+1]) then f[i]:=f[i+1];
    for i:=1 to n do//依题意扫答案
      inc(p[f[i]]);
    for i:=0 to n-1 do
```

```
    writeln(p[i]);
end.
```


Ps:三维偏序中，cdq的分治统计右区间内部贡献和左区间对右区间的贡献并不冲突，所以交换顺序也是可以的，这样我们就可以在递归左右时候可以通过归并排序来减少常数（因为我们已经维护了树状数组，所以这点小优化不影响时间复杂度）

28 10 条评论

Mingoal 创建时间：2018-06-22 13:17:19 在 Ta 的

安利一下博客

题目 #题解： 这题我看没有k-d树的题解，我就来一发 三维k-d树O(n^(5/3))应该是过不了的，我T掉了5个点 我们可以把z排序，x和y用k-d树每次计算当前位置前面有多少个x和y符合条件（均小于当前位置的x和y） 但是，这实际上是错误的，当一些z相等时，要查询的范围可能在之后，于是，我们可以建两棵k-d树，分别维护，然后就大功告成了，时间复杂度O(n^(3/2)) #标程：

```
//T维护在当前位置前，有多少个x和y符合条件
//K维护所有z值相等的点中，有多少个x和y符合条件
#include<cstdio>
#include<algorithm>
using namespace std;
const int N=100002;
int n,m,ans,D,i,f[N],k,j,t;
inline char gc(){
    static char buf[100000],*p1=buf,*p2=buf;
    return p1==p2&&(p2=(p1=buf)+fread(buf,1,100000,stdin),p1==p2)?EOF:*p1++;
}
#define gc getchar
inline int read(){
    int x=0,fl=1;char ch=gc();
    for (;ch<48||ch>57;ch=gc())if(ch=='-')fl=-1;
    for (;48<=ch&&ch<=57;ch=gc())x=(x<<3)+(x<<1)+(ch^48);
    return x*fl;
}
inline void wri(int a){if(a>=10)wri(a/10);putchar(a%10|48);}
inline void wln(int a){if(a<0)a=-a,putchar('-');wri(a);puts("");}
inline void Min(int &x,int y){if (y<x) x=y;}
inline void Max(int &x,int y){if (y>x) x=y;}
struct kk{
    int x,y,z;
}a[N];
struct P{
    int l,r,d[2],mn[2],mx[2],v,sum;
    int &operator[](int x){
        return d[x];
    }
    friend bool operator==(P a,P b){
        return a.d[0]==b.d[0] && a.d[1]==b.d[1];
    }
    friend bool operator<(P a,P b){
        return a[D]<b[D];
    }
}p[N];
bool in(int x1,int y1,int x2,int y2,int X1,int Y1,int X2,int Y2){
    return x1<=X1 && X2<=x2 && y1<=Y1 && Y2<=y2;
}
bool out(int x1,int y1,int x2,int y2,int X1,int Y1,int X2,int Y2){
    return X2<x1 || X1>x2 || Y2<y1 || Y1>y2;
}
struct node{
    P now,t[N];
    int rt,cnt;
    void update(int k){
        int l=t[k].l,r=t[k].r;
        for (int i=0;i<2;i++){
            t[k].mn[i]=t[k].mx[i]=t[k][i];
            if (l) Min(t[k].mn[i],t[l].mn[i]),Max(t[k].mx[i],t[l].mx[i]);
            if (r) Min(t[k].mn[i],t[r].mn[i]),Max(t[k].mx[i],t[r].mx[i]);
        }
        t[k].sum=t[k].v+t[l].sum+t[r].sum;
    }
    void ins(int &k,bool D){
```

```
            if (!k){
                k=++cnt;
                t[k][0]=t[k].mn[0]=t[k].mx[0]=now[0];
                t[k][1]=t[k].mn[1]=t[k].mx[1]=now[1];
            }
            if (now==t[k]){
                t[k].v+=now.v;t[k].sum+=now.v;
                return;
            }
            if (now[D]<t[k][D]) ins(t[k].l,D^1);
            else ins(t[k].r,D^1);
            update(k);
        }
        int query(int k,int x1,int y1,int x2,int y2){
            if (!k) return 0;
            int tmp=0;
            if (in(x1,y1,x2,y2,t[k].mn[0],t[k].mn[1],t[k].mx[0],t[k].mx[1])) return t[k].sum;
            if (out(x1,y1,x2,y2,t[k].mn[0],t[k].mn[1],t[k].mx[0],t[k].mx[1])) return 0;
            if (in(x1,y1,x2,y2,t[k][0],t[k][1],t[k][0],t[k][1])) tmp+=t[k].v;
            tmp+=query(t[k].l,x1,y1,x2,y2)+query(t[k].r,x1,y1,x2,y2);
            return tmp;
        }
        int build(int l,int r,bool f){
            if (l>r) return 0;
            int mid=l+r>>1;
            D=f;nth_element(p+l,p+mid,p+r+1);
            t[mid]=p[mid];
            t[mid].l=build(l,mid-1,f^1);
            t[mid].r=build(mid+1,r,f^1);
            update(mid);
            return mid;
        }
    }T,K;
    bool cmp(kk a,kk b){
        return a.z<b.z;
    }
    int main(){
        n=read();k=read();m=10000;
        for (i=1;i<=n;i++) a[i].x=read(),a[i].y=read(),a[i].z=read();
        sort(a+1,a+n+1,cmp);
        for (i=1;i<=n;i++){
            k=i;
            while (k<n && a[k].z==a[k+1].z) k++;
            for (j=i;j<=k;j++) p[j-i+1][0]=a[j].x,p[j-i+1][1]=a[j].y,p[j-i+1].v=p[j-i+1].sum=1;
            K.rt=K.build(1,k-i+1,0);
            for (j=i;j<=k;j++){
                ans=T.query(T.rt,1,1,a[j].x,a[j].y)+K.query(K.rt,1,1,a[j].x,a[j].y);
                f[ans]++;//把当前位置也算进去了，所以f的输出范围是[1,n]
            }
            for (j=i;j<=k;j++){
                T.now[0]=a[j].x,T.now[1]=a[j].y,T.now.v=T.now.sum=1,T.ins(T.rt,0);
                if (T.cnt==m){
                    for (t=1;t<=T.cnt;t++) p[t]=T.t[t];
                    T.rt=T.build(1,T.cnt,0);m+=10000;
                }
            }
            i=k;
        }
        for (i=1;i<=n;i++) wln(f[i]);
    }
```

20 9 条评论

panyf 创建时间：2021-05-30 19:18:05 在 Ta 的博

提供一篇 bitset 的题解。

bitset 可以用来求解高维偏序问题，记 m 为维数，则时间复杂度为 $O(\frac{n^2m}{w})$。

开 n 个 bitset，$b_{i,j}=1$ 表示 j 每一维都不超过 i。初始化所有 $b_{i,j}=1$。

先枚举每一维，然后对所有数按这一维排序。

开一个新的 bitset s，按这一维从小到大处理所有数，处理到 i 时 $s_j=1$ 表示当前维 j 不超过 i。

j 是单调的，可以用一个指针维护。

每次 `b[i]&=s` 即可。

100000 的 bitset 开不下，要用分组 bitset。

就是将点分为若干组，每组 B 个，每次只求出这 B 个点对应的 bitset，这样只需要开 B 个 bitset。

可能略微卡常。

AC 提交记录

```
#include<bits/stdc++.h>
using namespace std;
const int N=1e5+3;
int p[3][N],a[3][N],w[N];
bitset<N>b[9999],s;
int main(){
    int*tp,*ta,n,l,r,i,j,k,o;
    for(i=1,scanf("%d%*d",&n);i<=n;++i)for(j=0;j<3;++j)scanf("%d",a[j]+i),p[j][i]=i;
    for(i=0;i<3;++i)sort(p[i]+1,p[i]+n+1,[=](int x,int y){return a[i][x]<a[i][y];});//按每一维排序
    for(l=1;l<=n;l=r+1){//分组bitset，每组9991个
        for(i=l,r=min(l+9991,n);i<=r;++i)b[i-l].set();
        for(i=0;i<3;++i)for(tp=p[i],ta=a[i],s.reset(),j=k=1;j<=n;++j){
            for(o=tp[j];k<=n&&ta[tp[k]]<=ta[o];)s[tp[k++]]=1;
            if(l<=o&&o<=r)b[o-l]&=s;
        }
        for(i=l;i<=r;++i)++w[b[i-l].count()];
    }
    for(i=1;i<=n;++i)printf("%d\n",w[i]);
    return 0;
}
```

16 3 条评论

共 4 页 1 2