

CDQ 分治，偏序问题

CDQ 分治

Idea: 普通的分治中子问题之间互不影响，CDQ 分治中左区间可以对右区间产生影响。**必须离线。**

Steps: 分 \Rightarrow 递归处理左、右区间 \Rightarrow 处理左区间对右区间的影响，调整答案

Application: 点对有关问题（偏序，动态逆序对），1D/1D 动态规划的优化.....

二维偏序

Idea#1: 先按第一维排序，然后第二维值域树状数组维护

Idea#2: 先按第一维排序，然后 CDQ 分治

注：逆序对的两种实现分别对应上述两种做法

三维偏序

Idea#1: 先按第一维排序，

Idea#2: 先按第一维排序，然后 CDQ 分治：solve(l, r) 时，递归进行 solve(l, mid) 和 solve(mid+1, r)，考虑如何统计 $l \leq i \leq mid, mid+1 \leq j \leq r$ 的满足 $b_i < b_j$ 且 $c_i < c_j$ 的点对数。把所有 $[l, r]$ 区间的元素拿出来按第二维排序，遍历一遍，遇到左区间的就按第三维丢到值域树状数组里去，遇到右区间的就查询。

ATT: 如果题目存在重复元素，需要去重（因为 CDQ 分治只能统计左区间对右区间的答案），丢值域树状数组的时候丢重复次数。

Complexity: $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + O(n \lg n) = O(n \lg^2 n)$

Code (有重复元素) :

```
1  #include<algorithm>
2  #include<cstdio>
3
4  using namespace std;
5
6  const int N = 100005;
7  const int K = 200005;
8
9  int n, k, tot, ans[N];
10 struct Node{
11     int a, b, c, dc;    // dc is c after discretization
12     int newid, cnt, num; // num is the number of elements which are less than this node
13 }node[N];
14 bool cmpa(const Node &A, const Node &B){
15     return A.a == B.a ? (A.b == B.b ? A.c < B.c : A.b < B.b) : A.a < B.a;
16 }
17 bool cmpb(const Node &A, const Node &B){
18     return A.b == B.b ? (A.c == B.c ? A.a < B.a : A.c < B.c) : A.b < B.b;
19 }
20
21 int tc[N];
22 void disc(){
23     sort(tc+1, tc+tc[0]+1);
24     tc[0] = unique(tc+1, tc+tc[0]+1) - (tc+1);
```

```

25     for(int i = 1; i <= n; i++)
26         node[i].dc = lower_bound(tc+1, tc+tc[0]+1, node[i].c) - tc;
27 }
28
29 int c[N];
30 inline int lowbit(int x){ return x & -x; }
31 void add(int x, int val){ while(x <= n){ c[x] += val; x += lowbit(x); } }
32 int sum(int x){
33     int res = 0;
34     while(x){ res += c[x]; x -= lowbit(x); }
35     return res;
36 }
37
38 void cdq(int l, int r){
39     if(l == r) return;
40     int mid = (l + r) >> 1;
41     cdq(l, mid);
42     cdq(mid+1, r);
43     sort(node+l, node+r+1, cmpb);
44     for(int i = l; i <= r; i++){
45         if(node[i].newid <= mid) add(node[i].dc, node[i].cnt);
46         else node[i].num += sum(node[i].dc);
47     }
48     for(int i = l; i <= r; i++) // clear BIT
49         if(node[i].newid <= mid)
50             add(node[i].dc, -node[i].cnt);
51 }
52
53 int main(){
54     scanf("%d%d", &n, &k);
55     for(int i = 1; i <= n; i++){
56         scanf("%d%d%d", &node[i].a, &node[i].b, &node[i].c);
57         tc[++tc[0]] = node[i].c;
58     }
59     disc();
60     sort(node+1, node+n+1, cmpa);
61     for(int i = 1; i <= n; i++){ // unique
62         if(node[i-1].a != node[i].a || node[i-1].b != node[i].b || node[i-1].c != node[i].c)
63             node[++tot] = node[i], node[tot].newid = tot, node[tot].cnt = 1;
64         else node[tot].cnt++;
65     }
66     cdq(1, tot);
67     for(int i = 1; i <= tot; i++) ans[node[i].num + node[i].cnt - 1] += node[i].cnt;
68     for(int i = 0; i < n; i++) printf("%d\n", ans[i]);
69     return 0;
70 }

```

四维偏序

Idea: CDQ 套 CDQ 分治。先按第一维 a 排序，然后对第一维 CDQ 分治：递归回来后根据第一维的位置打上左右标记，然后按 b 排序，得到一系列形如 $(L/R, b, c, d)$ 的元素且 b 递增；复制一下，对第二维 CDQ 分治并在同时对第三维排序：递归回来后左子区间都是 $(L/R, L, c, d)$ ，右子区间都是 $(L/R, R, c, d)$ ，且各自区间内的 c 是递增的，然后双指针把第四维丢值域树状数组里或者查询，丢的时候记得参考第一维的 L/R 情况。

实现时用与 CDQ 浑然一体的归并排序而非 `sort` 来减少一些常数。

ATT: 清空树状数组时能少清就少清，这对减少常数很关键。

Complexity: $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + O(n \lg^2 n) = O(n \lg^3 n)$

Code:

```
1 #include<algorithm>
```

```

2  #include<cstdio>
3
4  using namespace std;
5
6  template<typename T>void read(T&x){x=0;int fl=1;char ch=getchar();while(ch<'0' || ch>'9'){if(ch=='-')
7  fl=-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=(x<<1)+(x<<3)+ch-'0';ch=getchar();}x*=fl;}
8  template<typename T,typename...Args>inline void read(T&t,Args&...args){read(t);read(args...);}
9
10 const int N = 50005;
11
12 int T, q, a[N], tot, ans[N];
13
14 #define LEFT 0
15 #define RIGHT 1
16 struct Node{
17     int t, x, y, z, k;
18     bool mark;
19 }node[N<<3];
20 bool cmp1(const Node &A, const Node &B){
21     if(A.t == B.t) if(A.x == B.x) if(A.y == B.y) return A.z < B.z;
22     else return A.y < B.y; else return A.x < B.x; else return A.t < B.t;
23 }
24
25 int t[N<<3];
26 void disc(){
27     sort(t+1, t+t[0]+1);
28     t[0] = unique(t+1, t+t[0]+1) - (t+1);
29     for(int i = 1; i <= tot; i++)
30         node[i].z = lower_bound(t+1, t+t[0]+1, node[i].z) - t;
31 }
32
33 int c[N<<3];
34 inline int lowbit(int x){ return x & -x; }
35 inline void add(int x, int val){ while(x <= q * 8){ c[x] += val; x += lowbit(x); } }
36 inline int sum(int x){ int res = 0; while(x){ res += c[x]; x -= lowbit(x); } return res; }
37
38 Node tmp[N<<3], tmp2[N<<3];
39 void cdq2(int l, int r){
40     if(l == r) return;
41     int mid = (l + r) >> 1;
42     cdq2(l, mid), cdq2(mid+1, r);
43     int ptl = l, ptr = mid+1, tid = l-1, lastl;
44     while(ptl <= mid && ptr <= r){
45         if(tmp[ptl].y <= tmp[ptr].y){
46             if(tmp[ptl].k == 0 && tmp[ptl].mark == LEFT)
47                 add(tmp[ptl].z, 1);
48             tmp2[++tid] = tmp[ptl++];
49         }
50         else{
51             if(tmp[ptr].k != 0 && tmp[ptr].mark == RIGHT)
52                 ans[tmp[ptr].t] += tmp[ptr].k * sum(tmp[ptr].z);
53             tmp2[++tid] = tmp[ptr++];
54         }
55     }
56     lastl = ptl - 1;
57     while(ptl <= mid) tmp2[++tid] = tmp[ptl++];
58     while(ptr <= r){
59         if(tmp[ptr].k != 0 && tmp[ptr].mark == RIGHT)
60             ans[tmp[ptr].t] += tmp[ptr].k * sum(tmp[ptr].z);
61         tmp2[++tid] = tmp[ptr++];
62     }
63     for(int i = l; i <= lastl; i++) // crucial for decreasing constant
64         if(tmp[i].k == 0 && tmp[i].mark == LEFT)
65             add(tmp[i].z, -1);
66     for(int i = l; i <= r; i++) tmp[i] = tmp2[i];
67 }
68 void cdq1(int l, int r){

```

```

69     if(l == r) return;
70     int mid = (l + r) >> 1;
71     cdq1(l, mid), cdq1(mid+1, r);
72     int ptl = l, ptr = mid+1, tid = l-1; // tid must be l-1 or it can't be used in cdq2
73     while(ptl <= mid && ptr <= r){
74         if(node[ptl].x <= node[ptr].x){
75             node[ptl].mark = LEFT;
76             tmp[++tid] = node[ptl++];
77         }
78         else{
79             node[ptr].mark = RIGHT;
80             tmp[++tid] = node[ptr++];
81         }
82     }
83     while(ptl <= mid) node[ptl].mark = LEFT, tmp[++tid] = node[ptl++];
84     while(ptr <= r) node[ptr].mark = RIGHT, tmp[++tid] = node[ptr++];
85     for(int i = l; i <= r; i++) node[i] = tmp[i];
86     cdq2(l, r);
87 }
88
89 inline void init(){
90     tot = t[0] = 0;
91     for(int i = 1; i <= q; i++) ans[i] = 0;
92 }
93
94 int main(){
95     read(T);
96     while(T--){
97         read(q);
98         init();
99         int _x1, _y1, _z1, _x2, _y2, _z2;
100         for(int i = 1; i <= q; i++){
101             read(a[i], _x1, _y1, _z1);
102             if(a[i] == 1){
103                 node[++tot] = (Node){i, _x1, _y1, _z1, 0};
104                 t[++t[0]] = _z1;
105             }
106             else{
107                 read(_x2, _y2, _z2);
108                 node[++tot] = (Node){i, _x2, _y2, _z2, 1};
109                 node[++tot] = (Node){i, _x1-1, _y2, _z2, -1};
110                 node[++tot] = (Node){i, _x2, _y1-1, _z2, -1};
111                 node[++tot] = (Node){i, _x2, _y2, _z1-1, -1};
112                 node[++tot] = (Node){i, _x1-1, _y1-1, _z2, 1};
113                 node[++tot] = (Node){i, _x1-1, _y2, _z1-1, 1};
114                 node[++tot] = (Node){i, _x2, _y1-1, _z1-1, 1};
115                 node[++tot] = (Node){i, _x1-1, _y1-1, _z1-1, -1};
116                 t[++t[0]] = _z2, t[++t[0]] = _z1-1;
117             }
118         }
119         disc();
120         sort(node+1, node+tot+1, cmpt);
121         cdq1(1, tot);
122         for(int i = 1; i <= q; i++)
123             if(a[i] == 2)
124                 printf("%d\n", ans[i]);
125     }
126     return 0;
127 }

```

