306-14-CDQ分治-陌上花开

题目大意

对于给遗传给定的序列：

$$(x,y,z)_1,(x,y,z)_2,(x,y,z)_3,\cdots,(x,y,z)_n \tag{1}$$

求：

$$\sum_{x_i<x_j,\ y_i<y_j,\ z_i<z_j,\ i\neq j} 1 \tag{2}$$

题解：

CDQ分治，顾名思义就是要进行分治，但是它可以解决比普通分治更多的问题。CDQ分治的整体思想，是：

1. 对于一个需要解决问题的区间$[l,r)$，将其一分为二，变为$[l,mid),[mid,r)$。
2. 对于这两个被分开的区间，分别进行递归解决。
3. 完成递归解决之后，计算左边对右边的贡献（有的时候可能要计算右边对左边的计算或者互相都要算）
4. 完成，统计答案。

在这道题当中，需要解决的是三维偏序的问题，而首先想到的是使用二维树状数组进行求逆序对，将这三维当中的两维当作下标，对最后一维求逆序对。但是这种做法显而易见，空间会爆掉，是开不下的。所以我们需要采取某种算法或者数据结构来代替树状数组的这意味空间。

在这里，就是用了CDQ分治。

首先解决第一维的问题：以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，进行排序。比较函数如下：

```cpp
bool cmp1(Point a, Point b) {
    if (a.x == b.x && a.y == b.y) return a.z < b.z;
    if (a.x == b.x) return a.y < b.y;
    return a.x < b.x;
}
```

在进行排序之后，就保证了整个序列单调不下降。

随后进行CDQ分治：`solve(0,n)`

1. `solve(0,mid)`，`solve(mid, n)`
2. 计算左边对右边的影响（原因：整个序列对于第一维变量单调不下降，所以右边的大数无法对小数产生答案上的贡献）
   - 对$[l,mid),[mid,r)$分别进行以第二维作为第一关键字，第三维作为第二关键字的排序：

```
bool cmp2(Point a, Point b) {
    if (a.y == b.y) return a.z < b.z;
    return a.y < b.y;
}
```

注意，这里是直接对原数组进行排序，而且实际上不会产生任何问题，因为被排序的两个子序列的这两个子问题，共同属于同一个问题，所以在进行其父亲问题的解决时还会进行重拍，届时并不会产生元素跨界问题。

  - 进行双指针扫描，双指针扫描的目的就是直接使用树状数组对每个元素进行答案统计。

整个问题解决完毕。

但是其存在一定的问题：由于CDQ分治这里我们讨论过了，应该只需要计算左边对右边的影响，但是忽略了一种情况，就是两个元素完全相同，那么左右应该共享影响，但是这里是非常难做到的，所以对于输入的时候，直接进行去重操作，并且赋予每个元素重复次数的权值，方便计算。

注意：在统计答案的时候，会看到下式：

```
ans[A[i].ans + A[i].cnt - 1] += A[i].cnt
```

这里还要加上`A[i].cnt-1`的原因其实也不难想到，就是去重的逆运算。

历届悲惨代码如下：

第一次（WA，0分，没有去重）：

```
#include <iostream>
#include <algorithm>

using namespace std;

typedef long long ll;
typedef pair<ll, pair<ll, ll> > Point;

const ll maxn = 200005;

Point A[maxn];
ll n, K, ord[maxn], f[maxn], T[maxn], ans[maxn];

bool cmp(ll x, ll y) {
    return A[x].second < A[y].second;
}

ll lb(ll i) { return i & (-i); }

void add(ll i, ll delta) {
    while (i <= K) {
        T[i] += delta;
        i += lb(i);
    }
}

ll sum(ll i) {
    ll ans = 0;
    while (i > 0) {
        ans += T[i];
        i -= lb(i);
    }
    return ans;
}

// 左闭右开
void solve(ll left, ll right) {
    if (right - left == 1) return;
    ll mid = (right + left) / 2;
    solve(left, mid); solve(mid, right);
    for (ll i = left; i < right; i ++)
        ord[i] = i;
    sort(ord + left, ord + right, cmp);
    // 左边对右边的理想
    for (ll i = left; i < right; i ++) {
        ll k = ord[i];
        if (k < mid) add(A[k].second.second, 1);
        else         f[k] += sum(A[k].second.second);
    }
    for (ll i = left; i < right; i ++)
        if (ord[i] < mid)
            add(A[ord[i]].second.second, -1);
}

int main() {
    // input;
    cin >> n >> K;
    for (ll i = 0; i < n; i ++)
        cin >> A[i].first >> A[i].second.first >> A[i].second.second;
    sort(A, A + n);
    solve(0, n);
    for (ll i = 0; i < n; i ++)
        ans[f[i]] ++;
    for (ll i = 0; i < n; i ++)
        cout << ans[i] << endl;
    return 0;
}

```

第二次（WA，0分，去重了，答案统计没做逆运算）：

```
#include <iostream>
#include <algorithm>
#define inf 10000000

using namespace std;

typedef long long ll;
struct Point {
    ll x, y, z, cnt;
    friend bool operator == (const Point& a, const Point& b) {
```

```cpp
        return (a.x == b.x) && (a.y == b.y) && (a.z == b.z);
    }
    friend bool operator < (const Point& a, const Point& b) {
        if (a.x == b.x && a.y == b.y && a.z == b.z) return a.cnt < b.cnt;
        if (a.x == b.x && a.y == b.y) return a.z < b.z;
        if (a.x == b.x) return a.y < b.y;
        return a.x < b.x;
    }
};

const ll maxn = 200005;

Point A[maxn];
ll n, K, ord[maxn], f[maxn], T[maxn], ans[maxn];

inline bool cmp(ll x, ll y) {
    if (A[x].y == A[y].y) return A[x].z < A[y].z;
    return A[x].y < A[y].y;
}

inline ll lb(ll i) { return i & (-i); }

inline void add(ll i, ll delta) {
    while (i <= K) {
        T[i] += delta;
        i += lb(i);
    }
}

inline ll sum(ll i) {
    ll ans = 0;
    while (i > 0) {
        ans += T[i];
        i -= lb(i);
    }
    return ans;
}

// 左闭右开
void solve(ll left, ll right) {
    if (right - left == 1) return;
    ll mid = (right + left) / 2;
    solve(left, mid); solve(mid, right);
    for (ll i = left; i < right; i ++)
        ord[i] = i;
    sort(ord + left, ord + right, cmp);
    // 左边对右边的贡献
    for (ll i = left; i < right; i ++) {
        ll k = ord[i];
        if (k < mid) add(A[k].z, A[k].cnt);
        else         f[k] += sum(A[k].z);
    }
    for (ll i = left; i < right; i ++) {
        ll k = ord[i];
        if (k < mid)
```

```
            add(A[k].z, (-1 * A[k].cnt));
    }
}

int main() {
    // input;
    cin >> n >> K;
    for (ll i = 0; i < n; i ++) {
        cin >> A[i].x >> A[i].y >> A[i].z;
        A[i].cnt = 1;
    }

    sort(A, A + n); ll multiple_count = 0;
    for (ll i = 1; i < n; i ++)
        if (A[i] == A[i - 1]) {
            A[i].cnt = A[i - 1].cnt + 1;
            multiple_count ++;
            A[i - 1] = (Point) {inf, inf, inf, inf};
        }
    sort(A, A + n);
    cout << "-----------------" << endl;
    for (int i = 0; i < n - multiple_count; i ++)
        cout << A[i].x << " " << A[i].y << " " << A[i].z << " " << A[i].cnt
<< " " << endl;
    solve(0, n - multiple_count);
    for (ll i = 0; i < n - multiple_count; i ++)
        ans[f[i]] += A[i].cnt;
    for (ll i = 0; i < n; i ++)
        cout << ans[i] << endl;
    return 0;
}

```

第三次（WA，10分，不知道哪里错了）：

```
#include <iostream>
#include <algorithm>
#define inf 10000000

using namespace std;

typedef long long ll;
struct Point {
    ll x, y, z, cnt;
    friend bool operator == (const Point& a, const Point& b) {
        return (a.x == b.x) && (a.y == b.y) && (a.z == b.z);
    }
    friend bool operator < (const Point& a, const Point& b) {
        if (a.x == b.x && a.y == b.y && a.z == b.z) return a.cnt < b.cnt;
        if (a.x == b.x && a.y == b.y) return a.z < b.z;
        if (a.x == b.x) return a.y < b.y;
        return a.x < b.x;
    }
};
```

```cpp

const ll maxn = 200005;

Point A[maxn];
ll n, K, ord[maxn], f[maxn], T[maxn], ans[maxn];

inline bool cmp(ll x, ll y) {
    if (A[x].y == A[y].y) return A[x].z < A[y].z;
    return A[x].y < A[y].y;
}

inline ll lb(ll i) { return i & (-i); }

inline void add(ll i, ll delta) {
    while (i <= K) {
        T[i] += delta;
        i += lb(i);
    }
}

inline ll sum(ll i) {
    ll ans = 0;
    while (i > 0) {
        ans += T[i];
        i -= lb(i);
    }
    return ans;
}

// 左闭右开
void solve(ll left, ll right) {
    if (right - left == 1) return;
    ll mid = (right + left) / 2;
    solve(left, mid); solve(mid, right);
    for (ll i = left; i < right; i ++)
        ord[i] = i;
    sort(ord + left, ord + right, cmp);
    // 左边对右边的贡献
    for (ll i = left; i < right; i ++) {
        ll k = ord[i];
        if (k < mid) add(A[k].z, A[k].cnt);
        else         f[k] += sum(A[k].z);
    }
    for (ll i = left; i < right; i ++) {
        ll k = ord[i];
        if (k < mid)
            add(A[k].z, (-1 * A[k].cnt));
    }
}

int main() {
    // input;
    cin >> n >> K;
    for (ll i = 0; i < n; i ++) {
        cin >> A[i].x >> A[i].y >> A[i].z;
```

```
            A[i].cnt = 1;
        }

    sort(A, A + n); ll multiple_count = 0;
    for (ll i = 1; i < n; i ++)
        if (A[i] == A[i - 1]) {
            A[i].cnt = A[i - 1].cnt + 1;
            multiple_count ++;
            A[i - 1] = (Point) {inf, inf, inf, inf};
        }
    sort(A, A + n);
    cout << "-----------------" << endl;
    for (int i = 0; i < n - multiple_count; i ++)
        cout << A[i].x << " " << A[i].y << " " << A[i].z << " " << A[i].cnt
<< " " << endl;
    solve(0, n - multiple_count);
    for (ll i = 0; i < n - multiple_count; i ++) {
      f[i] += A[i].cnt - 1;
    }
    for (ll i = 0; i < n - multiple_count; i ++)
        ans[f[i]] += A[i].cnt;
    for (ll i = 0; i < n; i ++)
        cout << ans[i] << endl;
    return 0;
}
```

第四次，放弃了，使用双指针扫描（AC）

```
#include <iostream>
#include <algorithm>
#define inf 1000000000

using namespace std;

typedef long long ll;
struct Point {
    ll x, y, z, cnt, ans;
    friend bool operator == (const Point& a, const Point& b) {
        return (a.x == b.x) && (a.y == b.y) && (a.z == b.z);
    }
};

const int maxn = 300005;

Point A[maxn];
ll n, K, ord[maxn], T[maxn], ans[maxn];

inline bool cmp2(Point a, Point b) {
    if (a.y == b.y) return a.z < b.z;
    return a.y < b.y;
}

inline bool cmp1(Point a, Point b) {
    if (a.x == b.x && a.y == b.y) return a.z < b.z;
```

```cpp
    if (a.x == b.x) return a.y < b.y;
    return a.x < b.x;
}

inline ll lb(ll i) { return i & (-i); }

inline void add(ll i, ll delta) {
    while (i <= K) {
        T[i] += delta;
        i += lb(i);
    }
}

inline ll sum(ll i) {
    ll ans = 0;
    while (i > 0) {
        ans += T[i];
        i -= lb(i);
    }
    return ans;
}

// 左闭右开
void solve(ll left, ll right) {
    if (right - left == 1) return;
    ll mid = (right + left) >> 1;
    solve(left, mid); solve(mid, right);
    sort(A + left, A + mid, cmp2);
    sort(A + mid, A + right, cmp2);
    ll i, j = left;
    for (i = mid; i < right; i ++) {
        while (A[i].y >= A[j].y && j < mid) {
          add(A[j].z, A[j].cnt); j ++;
        }
        A[i].ans += sum(A[i].z);
    }
    for (i = left; i < j; i ++)
        add(A[i].z, -A[i].cnt);
}

int main() {
    // Input
    cin >> n >> K;
    for (ll i = 0; i < n; i ++) {
        cin >> A[i].x >> A[i].y >> A[i].z;
        A[i].cnt = 1;
    }
    // Unique
    sort(A, A + n, cmp1); ll multiple_count = 0;
    for (ll i = 1; i < n; i ++)
        if (A[i] == A[i - 1]) {
            A[i].cnt = A[i - 1].cnt + 1;
            multiple_count ++;
            A[i - 1] = (Point) {inf, inf, inf, inf};
        }
    sort(A, A + n, cmp1);
//    cout << "-----------------" << endl;
//    for (ll i = 0; i < n - multiple_count; i ++)
//    cout << A[i].x << " " << A[i].y << " " << A[i].z << " " << A[i].cnt << " \t";
//    cout << endl;
    solve(0, n - multiple_count);
    for (ll i = 0; i < n - multiple_count; i ++)
        ans[A[i].ans + A[i].cnt - 1] += A[i].cnt;
    for (ll i = 0; i < n; i ++)
        cout << ans[i] << endl;
    return 0;
}
```

第五次，“我是专业的”的著名金牌OI教练解决了该问题（树状数组）

原因：在排序的过程中，对于区间：

$$\text{left}:[1]:(1,1,1),[2]:(1,2,1),[3]:(3,1,3),[4]:(3,3,6);$$
$$\text{right}:[5]:(1,2,1),[6]:(2,4,3),[7]:(3,1,3),[8]:(3,3,6)$$

在重新排序之后就会变为：

$$[1]:(1,1,1),[5]:(1,2,1),[2]:(1,2,1),[6]:(2,4,3),[7]:(3,1,3),[3]:(3,1,3),[8]:(3,3,6),[4]:(3,3,6)$$

这种排列方式显然就是错误的因为$2,5$；$3,7$；$4,8$这三组数据位置颠倒了。而解决这个问题的方式就是更正排序方式：

1. 将y作为第一排序关键字
2. 将z作为第二排序关键字
3. 将x作为第三排序关键字

```cpp
#include <iostream>
#include <algorithm>
#define inf 1000000000

using namespace std;

typedef long long ll;
struct Point {
    ll x, y, z, cnt;
    friend bool operator == (const Point& a, const Point& b) {
        return (a.x == b.x) && (a.y == b.y) && (a.z == b.z);
    }
};

const int maxn = 300005;

Point A[maxn];
ll n, K, ord[maxn], f[maxn], T[maxn], ans[maxn];

inline bool cmp2(ll x, ll y) {
  if (A[x].y == A[y].y && A[x].z == A[y].z) return A[x].x < A[y].x;
    if (A[x].y == A[y].y) return A[x].z < A[y].z;
    return A[x].y < A[y].y;
```

```cpp
}

inline bool cmp1(Point a, Point b) {
    if (a.x == b.x && a.y == b.y) return a.z < b.z;
    if (a.x == b.x) return a.y < b.y;
    return a.x < b.x;
}

inline ll lb(ll i) { return i & (-i); }

inline void add(ll i, ll delta) {
    while (i <= K) {
        T[i] += delta;
        i += lb(i);
    }
}

inline ll sum(ll i) {
    ll ans = 0;
    while (i > 0) {
        ans += T[i];
        i -= lb(i);
    }
    return ans;
}

// 左闭右开
void solve(ll left, ll right) {
    if (right - left == 1) return;
    ll mid = (right + left) >> 1;
    solve(left, mid); solve(mid, right);
    // 根据y的值对数组进行排序，和求逆序对时的构造离散化类似
  // 即在这里，y是求逆序对当中的时序
  for (ll i = left; i < right; i ++)
        ord[i] = i;
    sort(ord + left, ord + right, cmp2);
    // 左边对右边的贡献
    for (ll i = left; i < right; i ++) {
        ll k = ord[i];
        if (k < mid)  add(A[k].z, A[k].cnt);
        else      f[k] += sum(A[k].z);
    }
  // Clear Binary Index Tree
    for (ll i = left; i < right; i ++) {
        ll k = ord[i]; ord[i] = 0;
        if (k < mid)    add(A[k].z, (-1 * A[k].cnt));
    }
}

int main() {
    // Input
    cin >> n >> K;
    for (ll i = 0; i < n; i ++) {
        cin >> A[i].x >> A[i].y >> A[i].z;
        A[i].cnt = 1;
    }
    // Unique
    sort(A, A + n, cmp1); ll multiple_count = 0;
    for (ll i = 1; i < n; i ++)
        if (A[i] == A[i - 1]) {
            A[i].cnt = A[i - 1].cnt + 1;
            multiple_count ++;
            A[i - 1] = (Point) {inf, inf, inf, inf};
        }
    sort(A, A + n, cmp1);
//    cout << "------------------" << endl;
//    for (ll i = 0; i < n - multiple_count; i ++)
//    cout << A[i].x << " " << A[i].y << " " << A[i].z << " " << A[i].cnt << " \t";
//  cout << endl;
    solve(0, n - multiple_count);
  for (ll i = 0; i < n - multiple_count; i ++) {
    f[i] += A[i].cnt - 1;
  }
    for (ll i = 0; i < n - multiple_count; i ++)
        ans[f[i]] += A[i].cnt;
    for (ll i = 0; i < n; i ++)
      cout << ans[i] << endl;
    return 0;
}
```