Gruplama Çözüm

Öncelikle soruyu en basit haliyle çözelim: y=0, yani bütün noktalar tek bir satır üzerinde. K=1, yani tek bir grup var ve bizden istenen gruptaki uzaklık toplamı. n <= 1000, yani O(n^2)'de çözebiliriz. Bu durumda çözüm, her nokta ikilisini 2 for ile gezip aradaki |x-x2| uzaklığını toplamak olur.

Şimdi yavaş yavaş kısıtlamalardan kurtulalım. Öncelikle y=0'ı ele alalım. Formül'de fark ettiğimiz üzere X ekseni ile Y ekseni birbirinden bağımsız. Yani X ekseni için yaptığımız şeyin aynısını Y ekseni için de yapıp değerleri toplayabiliriz. Bu şekilde K=1 ve n <=1000 için çözebilmiş oluruz.

İkinci olarak n <= 1000 den kurtulmaya çalışalım. Yani O(n)'e çevirmeye çalışalım. 1 noktanın o gruptaki bütün elemanlara olan uzaklık toplamını O(1)'da bulursak çözmüş oluruz. Önceki gözlemi kullanarak tek eksen üzerinde çalışalım, yani her noktanın tek değeri olacak. Her noktayı o değerlerine göre sıralarsak, 1 noktanın kendisinden önce gelenlere olan uzaklık toplamı bütün değerlerin toplamıyla kendi değerinin değer sayısıyla çarpımının farkına eşittir. Aynısını kendisinden sonra gelenler için yapabiliriz. Yani:

[1, 3, 4, 6, 8, 10, 11] dizisinin, 9'a olan uzaklık toplamı:

- Önce gelenler: (9-1)+(9-3)+(9-4)+(9-6)+(9-8) = 9*5 - (1+3+4+6+8) = 45-22 = 23
- Sonra gelenler: (10-9)+(11-9) = (11+10) - 9*2 = 21-18 = 3

23+3 = 26 şeklinde bulunabilir. Aralık toplamı ise prefix sum kullanarak bulunabilir. Yani:

- Normal dizi: [1, 3, 4, 6, 8, 10, 11]
- Prefix sum dizisi: [1, 4, 8, 14, 22, 32, 43]

Dizilerini kullanarak herhangi iki aralık O(1)'da bulunabilir. Mesela normal dizideki [4, 6, 8] aralığının toplamı Prefix sum dizisinden başa ve sona karşılık gelen 22 ve 4'ü kullanarak bulunabilir.
(4+6+8 = 18 = 22-4)
Bu diziyi bulunan tek grup (K=1) için kurduğumuzda her sayı için 1'den ona kadar ve ondan sona kadar olan aralıkları sorgulayarak O(1)'da toplamı bulabiliriz.

Son olarak K=1'i kaldıralım. İkinci adımda yaptıklarımızın aynısını burdaki her grup için yapıyoruz. Bir noktanın grubunu değiştirecekken o noktanın o anki gruba katkısını ve değiştirdiğinde yeni gruba olan katkısını bulursak çözeriz. Katkısını hesaplamak içinse yeni grupta hangilerinin önce geldiğini ve hangilerinin sonra geldiğini bulmak yeterli. Elemanlar her grupta sıralı zaten. O yüzden yeni elemanın konumunu bulmak için İkili Arama (Binary Search) yöntemi kullanılabilir.

Bu şekilde herhangi bir noktanın bir gruptaki bütün noktalara olan uzaklık toplamını O(logn)'de bulabiliriz. Bunu her noktanın olası her gruba geçme durumu için yaparsak, herhangi bir işlemin cevabın ne kadar değiştiğini, toplamda O(n*K*logn)'de bulabiliriz. Başlangıçtaki cevabı ise yine bu fonksiyonu kullanarak, bir noktanın farklı gruplara olan uzaklığını toplayıp aynı gruba olan uzaklığını çıkartarak bulabiliriz.

Çözüm Kodu:

```
#include <bits/stdc++.h>
#define all(x) x.begin() , x.end()
#define fi first
#define se second
#define pb push_back
#define umax( x , y ) x = max( x , (y) )
#define umin( x , y ) x = min( x , (y) )

using namespace std;

typedef long long Lint;
typedef pair<int,int> ii;
const int maxn = 100020;

Lint val[maxn][102], ans, base;
vector<Lint> w[2][102];
vector<Lint> sum[2][102];
ii ar[maxn];
int g[maxn], a, b;

Lint get( int i, int j, int l, int r ) {
    l--;
    if( r < 0 ) return 0;
    if( l >= 0 ) return sum[i][j][r] - sum[i][j][l];
    return sum[i][j][r];
}

Lint find( int i, int j, Lint x ) {
    int t = lower_bound( all( w[i][j] ), x ) - w[i][j].begin();
    return get( i, j, t, w[i][j].size()-1 ) - get( i, j, 0, t-1 )
+ x * (t+t-w[i][j].size());
}
```

```
void go( int axis ) {
    for(int i=1;i<=a;i++) {
        Lint t = ar[i].fi;
        if( x ) t = ar[i].se;
        Lint own = find( axis, g[i], t );
        for(int j=1;j<=b;j++) {
            Lint changed = find( axis, j, t );
            if( g[i] == j ) base -= changed;
            else base += changed;
            val[i][j] += 2*(own-changed);
            if( axis ) umax( ans, val[i][j] );
        }
    }
}

int main() {
    scanf("%d %d",&a,&b);
    for(int i=1,x,y,t;i<=a;i++) {
        scanf("%d %d %d",&x,&y,&t);
        ar[i] = ii( x, y );
        g[i] = t;
        w[0][t].pb( j );
        w[1][t].pb( k );
    }
    for(int axis=0;axis<2;axis++)
        for(int i=1;i<=b;i++) {
            sort( all( w[axis][i] ) );
            Lint k = 0;
            for(Lint h: w[axis][i]) {
                k += h;
                sum[axis][i].pb( k );
            }
        }
    go(0);
    go(1);
    base /= 2;
    cout << base+ans << endl;
    return 0;
}
```