

扫描线

Scan Line

Idea: (以求矩形面积并为例) 用平行于 x 轴的直线去扫描图形, 每扫描至一个高度计算图形被扫描到的宽度——用线段树维护。具体地, 每至一个高度, 若该高度有一个矩形的底边, 则相应线段树区间 $+1$, 若有一个矩形的顶边, 则相应线段树区间 -1 。注意, 假设所有 x 坐标排序后形成序列 $\{x_n\}$, 则线段树的第 i 个叶节点代表的是区间 $[x_i, x_{i+1})$ 而非点 x_i 。

ATT: 线段树没有 `pushdown` 操作 (也不好 `pushdown`), 我们需要深入理解 `cnt` 标记的含义:

- `cnt == 0`: 仅从这个节点无法知道区间信息 (可能没被覆盖, 可能部分覆盖, 可能全被覆盖, 要从祖先节点的信息推知)。于是更新 `length` 信息时, 必须从左右儿子更新上来。
- `cnt >= 1`: 该区间被完全覆盖了至少 1 次 (可能 1 次, 可能 2 次.....), `length` 信息即区间长度。

Application: 求图形面积、周长.....

Complexity: $O(n \lg n)$

Code (以求矩形面积并为例):

```
1  #include<cstdio>
2  #include<algorithm>
3
4  using namespace std;
5
6  const int N = 200005;
7
8  int n, xid;
9  double tx1, ty1, tx2, ty2, x[N], ans;
10 struct ScanLine{
11     double x1, x2, y;
12     int k; // k == 1 or -1
13     int dx1, dx2; // after discretization
14     bool operator < (const ScanLine &A) const{
15         return y == A.y ? k > A.k : y < A.y;
16     }
17 }a[N];
18
19 inline void disc(){
20     sort(x+1, x+xid+1);
21     xid = unique(x+1, x+xid+1) - (x+1);
22     for(int i = 1; i <= n; i++){
23         a[i].dx1 = lower_bound(x+1, x+xid+1, a[i].x1) - x;
24         a[i].dx2 = lower_bound(x+1, x+xid+1, a[i].x2) - x;
25     }
26 }
27
28 struct SegTree{
29     int l, r, cnt;
30     double length;
31 }tr[N<<2];
32 #define lid id<<1
33 #define rid id<<1|1
34 #define mid ((tr[id].l + tr[id].r) >> 1)
35 #define len(id) (tr[id].r - tr[id].l + 1)
36 inline void pushup(int id){
37     if(tr[id].cnt > 0) tr[id].length = x[tr[id].r + 1] - x[tr[id].l];
38     else{
39         if(tr[id].l == tr[id].r) tr[id].length = 0;
40         else tr[id].length = tr[lid].length + tr[rid].length;
41     }
```

```

42 }
43 void build(int id, int l, int r){
44     tr[id].l = l, tr[id].r = r;
45     tr[id].cnt = 0, tr[id].length = 0;
46     if(tr[id].l == tr[id].r) return;
47     build(lid, l, mid);
48     build(rid, mid+1, r);
49     pushup(id);
50 }
51 void add(int id, int l, int r, int val){
52     if(tr[id].l == l && tr[id].r == r){
53         tr[id].cnt += val;
54         pushup(id);
55         return;
56     }
57     if(r <= mid) add(lid, l, r, val);
58     else if(l > mid) add(rid, l, r, val);
59     else add(lid, l, mid, val), add(rid, mid+1, r, val);
60     pushup(id);
61 }
62
63 int main(){
64     scanf("%d", &n);
65     for(int i = 1; i <= n; i++){
66         scanf("%lf%lf%lf%lf", &tx1, &ty1, &tx2, &ty2);
67         a[i] = (ScanLine){tx1, tx2, ty1, 1};
68         a[i+n] = (ScanLine){tx1, tx2, ty2, -1};
69         x[++xid] = tx1, x[++xid] = tx2;
70     }
71     n <= 1;
72     disc();
73     sort(a+1, a+n+1);
74     build(1, 1, xid-1);
75     for(int i = 1; i < n; i++){
76         add(1, a[i].dx1, a[i].dx2 - 1, a[i].k);
77         ans += tr[1].length * (a[i+1].y - a[i].y);
78     }
79     printf("%.2f\n", ans);
80     return 0;
81 }

```