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）：

```cpp
#include<cstdio>
#include<algorithm>

using namespace std;

const int N = 200005;

int n, xid;
double tx1, ty1, tx2, ty2, x[N], ans;
struct ScanLine{
    double x1, x2, y;
    int k; // k == 1 or -1
    int dx1, dx2; // after discretization
    bool operator < (const ScanLine &A) const{
        return y == A.y ? k > A.k : y < A.y;
    }
}a[N];

inline void disc(){
    sort(x+1, x+xid+1);
    xid = unique(x+1, x+xid+1) - (x+1);
    for(int i = 1; i <= n; i++){
        a[i].dx1 = lower_bound(x+1, x+xid+1, a[i].x1) - x;
        a[i].dx2 = lower_bound(x+1, x+xid+1, a[i].x2) - x;
    }
}

struct SegTree{
    int l, r, cnt;
    double length;
}tr[N<<2];
#define lid id<<1
#define rid id<<1|1
#define mid ((tr[id].l + tr[id].r) >> 1)
#define len(id) (tr[id].r - tr[id].l + 1)
inline void pushup(int id){
    if(tr[id].cnt > 0)  tr[id].length = x[tr[id].r + 1] - x[tr[id].l];
    else{
        if(tr[id].l == tr[id].r)    tr[id].length = 0;
        else    tr[id].length = tr[lid].length + tr[rid].length;
    }
}
void build(int id, int l, int r){
    tr[id].l = l, tr[id].r = r;
    tr[id].cnt = 0, tr[id].length = 0;
    if(tr[id].l == tr[id].r)    return;
    build(lid, l, mid);
    build(rid, mid+1, r);
    pushup(id);
}
void add(int id, int l, int r, int val){
```

```
    if(tr[id].l == l && tr[id].r == r){
        tr[id].cnt += val;
        pushup(id);
        return;
    }
    if(r <= mid) add(lid, l, r, val);
    else if(l > mid) add(rid, l, r, val);
    else    add(lid, l, mid, val), add(rid, mid+1, r, val);
    pushup(id);
}

int main(){
    scanf("%d", &n);
    for(int i = 1; i <= n; i++){
        scanf("%lf%lf%lf%lf", &tx1, &ty1, &tx2, &ty2);
        a[i] = (ScanLine){tx1, tx2, ty1, 1};
        a[i+n] = (ScanLine){tx1, tx2, ty2, -1};
        x[++xid] = tx1, x[++xid] = tx2;
    }
    n <<= 1;
    disc();
    sort(a+1, a+n+1);
    build(1, 1, xid-1);
    for(int i = 1; i < n; i++){
        add(1, a[i].dx1, a[i].dx2 - 1, a[i].k);
        ans += tr[1].length * (a[i+1].y - a[i].y);
    }
    printf("%.2f\n", ans);
    return 0;
}
```