# 扫描线法（二）

主讲人：邓哲也

# 矩形面积并加强

二维平面上有 n 个矩形，告诉你第 i 个矩形的左上角和右下角的坐标 (x1[i],y1[i]), (x2[i], y2[i]).

求这些矩形的并的面积。

n <= 100000, 0 < x, y < $10^9$

# 矩形面积并加强

现在由于坐标的范围太大了，没法直接建 $10^9$ 的线段树。

运用离散化思想，因为不同的 x 坐标只有 2n 个，不妨把它们从小到大排序，将[1，$10^9$]映射到[1，2n]，这样就可以建线段树了。

同理 y 也是一样。

要注意的是计算答案的时候，要使用原来的 x 和 y。

# 代码实现

```
#define N 1000
#define ls (x << 1)
#define rs (x << 1 | 1)
double xbin[N], ybin[N];
int xcnt, ycnt, n;
double sum[N];
int val[N], tag[N];
struct rect{
    double x1, y1, x2, y2;
}r[N];
struct event{
    int x1, x2, v;
};
vector <event> g[N];
```

# 代码实现

```
void upd(int x){
    if (val[ls] == val[rs]){
        val[x] = val[ls];
        sum[x] = sum[ls] + sum[rs];
    } else {
        int f = (val[ls] < val[rs]) ? ls : rs;
        val[x] = val[f];
        sum[x] = sum[f];
    }
}
void down(int x){
    if (tag[x] != 0){
        val[ls] += tag[x]; val[rs] += tag[x];
        tag[ls] += tag[x]; tag[rs] += tag[x];
        tag[x] = 0;
    }
}
```

# 代码实现

```
void add(int A, int B, int v, int l, int r, int x){
    if (A <= l && r <= B) {
        tag[x] += v;
        val[x] += v;
        return;
    }
    down(x);
    int mid = (l + r) >> 1;
    if (A <= mid) add(A, B, v, l, mid, ls);
    if (mid < B)  add(A, B, v, mid + 1, r, rs);
    upd(x);
}
```

# 代码实现

```
void build(int l, int r, int x){
    val[x] = tag[x] = 0;
    if (l == r){
        sum[x] = xbin[l] - xbin[l - 1];
        return;
    }
    int mid = (l + r) >> 1;
    build(l, mid, ls);
    build(mid + 1, r, rs);
    upd(x);
}
```

# 代码实现

```
int main(){
    int tc = 0;
    while(scanf("%d", &n) != EOF){
        if (!n) break;
        xcnt = ycnt = 0;
        for(int i = 1;i <= n;i ++){
            scanf("%lf%lf%lf%lf", &r[i].x1, &r[i].y1, &r[i].x2, &r[i].y2);
            xbin[++ xcnt] = r[i].x1;
            xbin[++ xcnt] = r[i].x2;
            ybin[++ ycnt] = r[i].y1;
            ybin[++ ycnt] = r[i].y2;
        }
```

# 代码实现

```
sort(xbin + 1, xbin + xcnt + 1);
sort(ybin + 1, ybin + ycnt + 1);
xcnt = unique(xbin + 1, xbin + xcnt + 1) - xbin - 1;
ycnt = unique(ybin + 1, ybin + ycnt + 1) - ybin - 1;
for(int i = 1;i <= n;i ++){
    int x1 = lower_bound(xbin + 1, xbin + xcnt + 1, r[i].x1) - xbin;
    int x2 = lower_bound(xbin + 1, xbin + xcnt + 1, r[i].x2) - xbin;
    int y1 = lower_bound(ybin + 1, ybin + ycnt + 1, r[i].y1) - ybin;
    int y2 = lower_bound(ybin + 1, ybin + ycnt + 1, r[i].y2) - ybin;
    if (x1 < x2){
        g[y1].push_back((event){x1 + 1, x2, 1});
        g[y2].push_back((event){x1 + 1, x2, -1});
    }
}
```

# 代码实现

```
            xbin[0] = xbin[1];
            ybin[0] = ybin[1];
            build(1, xcnt, 1);
            double ans = 0;
            for (int i = 1;i <= ycnt;i ++){
                double tmp = sum[1];
                if (val[1] > 0) tmp = 0;
                ans += (ybin[i] - ybin[i - 1]) * (xbin[xcnt] - xbin[1] - tmp);
                for (int j = 0;j < g[i].size();j ++){
                    add(g[i][j].x1, g[i][j].x2, g[i][j].v, 1, xcnt, 1);
                }
            }
            printf("Test case #%d\nTotal explored area: %.2lf\n\n", ++ tc, ans);
            for(int i = 1;i <= ycnt;i ++) g[i].clear();
        }
    }
```

# 下节课再见