Line Sweep Algorithms – Handout

(by Schalk-Willem Krüger)

1. Closest Pair: Sample Source Code

```cpp
#include <stdio.h>
#include <set>
#include <algorithm>
#include <cmath>
using namespace std;
#define px second
#define py first
typedef pair<long long, long long> pairll;
int n;
pairll pnts [100000];
set<pairll> box;
double best;
int compx(pairll a, pairll b) { return a.px<b.px; }
int main () {
    scanf("%d", &n);
    for (int i=0;i<n;++i) scanf("%lld %lld", &pnts[i].px, &pnts[i].py);
    sort(pnts, pnts+n, compx);
    best = 1500000000; // INF
    box.insert(pnts[0]);
    int left = 0;
    for (int i=1;i<n;++i) {
        while (left<i && pnts[i].px-pnts[left].px > best) box.erase(pnts[left++]);
        for (typeof(box.begin()) it=box.lower_bound(make_pair(pnts[i].py-best, pnts[i].px-best));
             it!=box.end() && pnts[i].py+best>=it->py; it++)
                 best = min(best, sqrt(pow(pnts[i].py - it->py, 2.0)+pow(pnts[i].px - it->px, 2.0)));
        box.insert(pnts[i]);
    printf("%.2f\n", best);
    }
    return 0;
}
```

2. Union of rectangles: Sample Source Code

```cpp
#include <cstdio>
#include <algorithm>
using namespace std;
struct event {
        int ind;      // Index of rectangle in rects
        bool type;    // Type of event: 0 = Lower-left ; 1 = Upper-right
        event() {};
        event(int ind, int type) : ind(ind), type(type) {};
};
struct point {
        int x, y;
};
int n, e; // n = number of rectangles; e = number of edges
point rects [1000][2];  // Each rectangle consists of 2 points: [0] = lower-left ; [1] = upper-right
event events_v [2000];  // Events of horizontal sweep line
event events_h [2000];  // Events of vertical sweep line
bool compare_x(event a, event b) { return rects[a.ind][a.type].x<rects[b.ind][b.type].x; }
bool compare_y(event a, event b) { return rects[a.ind][a.type].y<rects[b.ind][b.type].y; }
bool in_set [10000];    // Boolean array in place of balanced binary tree (set)
long long area;         // The output: Area of the union
int main() {   /// x -> v; y -> h
        scanf("%d", &n);
        for (int i=0;i<n;++i) {
                scanf("%d %d %d %d", &rects[i][0].x, &rects[i][0].y,   // Lower-left coordinate
                                     &rects[i][1].x, &rects[i][1].y);  // Upper-right coordinate
                events_v[e] = event(i, 0);
                events_h[e++] = event(i, 0);
                events_v[e] = event(i, 1);
                events_h[e++] = event(i, 1);
        }
        sort(events_v, events_v+e, compare_x);
        sort(events_h, events_h+e, compare_y);  // Pre-sort set of horizontal edges
        in_set[events_v[0].ind] = 1;
        for (int i=1;i<e;++i) {  // Vertical sweep line
                event c = events_v[i];
                int cnt = 0;  // Counter to indicate how many rectangles are currently overlapping
                // Delta_x: Distance between current sweep line and previous sweep line
                int delta_x = rects[c.ind][c.type].x - rects[events_v[i-1].ind][events_v[i-1].type].x;
                int begin_y;
                if (delta_x==0) continue;
                for (int j=0;j<e;++j) if (in_set[events_h[j].ind]==1) {  // Horizontal sweep line
                        if (events_h[j].type==0) {
                                if (cnt==0) begin_y = rects[events_h[j].ind][0].y;  // Block starts
                                ++cnt;
                        } else {
                                --cnt;
                                if (cnt==0) { // Block ends
                                        int delta_y = (rects[events_h[j].ind][1].y-begin_y);
                                        area+=delta_x * delta_y;
                                }
                        }
                }
                in_set[c.ind] = (c.type==0);
        }
        printf("%lld\n", area);
        return 0;
}
```