

Review of Computational Geometry Problems

Prof. Dr. M. Böhlen

May 19 2008

It is recommended that you solve the tasks without auxiliary material to mirror the setting at the exam.

Task 1:

Consider the five line segments in Figure 1 and the plane sweep algorithm that determines if any pair of segments intersects. The lower left corner has coordinate $(0,0)$ and the upper right corner coordinate $(20,20)$. The plane sweep algorithm maintains the sweep line status. Show the ordered content of the sweep line status after each change of the sweep line status.

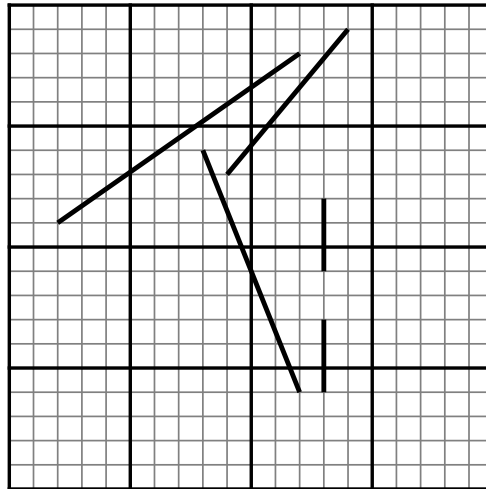


Figure 1: Set of Five Line Segments

Task 2:

Given is an array $A[n]$ with 2D points and a 2D point q . Assume that the y coordinate of q is lower than the y coordinate of all points in A . The task is to sort the points in A according to their polar angle wrt point q . Provide a complete sorting algorithm that achieves this task in $O(n \log n)$ time.

Task 3:

Explain Jarvis's march to determine the convex hull of a set of points. What is the computational complexity of Jarvis's march? Consider the following implementation of Jarvis's march. P is an array with n 2D points; $pmin$ is the index of the point with the lowest y coordinate; and $pmax$ is the index of the point with the largest y coordinate. List and explain all cases that lead to a failure of the algorithm. Explain how to correct the algorithm.

```
void JarvisMarch(point P[], int n, int pmin, int pmax) {
    int p0, pn, i, d;
    p0 = pmin;
    while (p0 != pmax) {
        pn = pmax;
        for (i=0; i<n; i++) {
            d = segDir(p0, i, pn);
            if (d < 0) { pn = i; }
        }
        printf("(%d,%d)", P[pn].x, P[pn].y);
        p0 = pn;
    }
    p0 = pmax;
    while (p0 != pmin) {
        pn = pmin;
        for (i=0; i<n; i++) {
            d = segDir(p0, i, pn);
            if (d < 0) { pn = i; }
        }
        printf("(%d,%d)", P[pn].x, P[pn].y);
        p0 = pn;
    }
}
```

Task 4:

Consider the plane sweep algorithm that determines if any pair of segments intersects. Assume the algorithm is modified as follows: instead of terminating upon finding an intersection it prints the segments that intersect and continues with the next iteration of the for loop. Show that the first intersection found by this algorithm is not necessarily the leftmost one and that the algorithm will fail to find all intersections.

Task 5:

Argue that plane sweep algorithm that determines if any pair of segments intersects algorithm works correctly even if three or more segments intersect at the same point.

Task 6:

Assume a red-black tree is used to maintain the sweep line status in the plane sweep algorithm that checks if in any pair of segments intersects. The red-black tree implementation provides the following functions:

```
struct rb_tree* RBcreate();
struct rb_node* RBinsert(struct rb_tree* T, int key);
struct rb_node* RBsearch(struct rb_tree* T, int key);
struct rb_node* RBmin(struct rb_tree* T);
struct rb_node* RBsucc(struct rb_tree* T, struct rb_node* x);
struct rb_node* RBpred(struct rb_tree* T, struct rb_node* x);
void RBdelete(struct rb_tree* T, struct rb_node* z);
```

Explain what parts of a regular red-black tree implementation have to be adjusted to solve the segment intersection problem. Describe the necessary changes to the code of a red-black tree implementation. Give precise code fragments.