Analysis of Algorithms

second assignment

Thom Wiggers

March 12, 2014

1 Exercise 1

Find an algorithm in $O(n \lg n)$ which finds the x-coordinate of a line which intersects the maximum amount of line segments.

This is a slightly modified version of the "Any-Segment-Intersect" algorithm of "Introduction To Algoritms", page 1025. It keeps for each point a score of how many lines are intersected.

Algorithm 1.1

```
(1)  Find the place with the most intersections
(3)  begin
(5)    Sort the endpoints of the segments in S from left to right,
(6)      breaking ties by putting left endpoints before rightendpoints and
(7)      breaking further ties by putting points with lower y-coordinates first.
(8)    score = 0                          we first intersect nothing, obviously
(9)    best = 0                                          Best score so far
(10)   point = (−∞, −∞)                   Best point so far, (x,y)
(11)   foreach point p in the sorted list of endpoints
(12)     if p is the left endpoint of a segment s
(13)       then
(14)            inc(score)                                  increment score
(15)            if score > best
(16)              then
(17)                   point = p
(18)                   best = score
(19)            fi
(20)        else
(21)            dec(score)                                  We're leaving a line
(22)     fi
(23)   end
(24)   return point.x
(25) end
```

As I've shown, no data structure is needed.

1.1 Complexity

This program has complexity $O(n \lg n)$.

The initial sorting has complexity $O(n \lg n)$. Then the algorithm has complexity n:

If set S contains n segments, then lines 8 - 10 run in $O(1)$. The foreach loop of lines 11 - 23 iterates at most once per point, and with $2n$ points it iterates $2n$ times. All instructions in the for loop only take $O(1)$ time, since they are only comparisons and assignments. So the second part of the algoritm only takes $O(n)$ time.

1.2 Correctness

I choose the following loop invariant: 'point' always contains a point with the most intersections left of the sweep line (1), 'best' contains the amount of intersections at 'point' (2) and 'score' always contains the amount of intersections at the current p (3).

This is true prior to iteration because there are no points left to the sweep line, and thus the 'best' and current amount of intersections is '0' (line 8 and 9), and the point is minus ∞ left of the sweep line.

In the loop there are two main cases: either p is a left endpoint or p is a right endpoint.

Assume that p is a left endpoint. Then we've arrived at a new line which our sweep line intersects. We increments 'score', which contained the amount of intersections (3), so it contains the new amount of intersections. (3) thus still holds after this iteration, because it is not changed again. Now, if 'score' is greater than 'best', p is better than 'point', because 'best' contained the amount of intersections at 'point'(2). So 'point' and 'best' are updated, again satisfying (1) and (2). If 'score' isn't greater than 'best', 'best' and 'point' still are correct.

Assume that p is a right endpoint. Then we've left a line. To satisfy (3) again we decrement 'score'. Because we're leaving a point, 'best' and 'point' still satisfy (1) and (2).

So in every case we satisfy the loop invariant. After completing the loop, all points are to the left of the sweep line, and 'point' has a point where the most intersections happen.

2 Exercise 2

CH(P) already is sorted, sorting $\{p\}$ into that list can be done in $O(n)$ time. You then only need to execute lines 3 – 11 of the Graham-Scan algoritm. The only question that needs to be answered is if p is an outer point or not, therefore there is no need to look at all the points of P again: it can only be an outer point if it's outside the current outer points which are in CH(P). Lines 3–11 are $O(n)$ (page 1036, introduction to algoritms).