Matthias Fischer

Paderborn, 8. April 2024
Abgabe: Di., 16. April, 11:15 Uhr

Hausaufgaben zur Vorlesung

Algorithmische Geometrie

SS 2024

Übungsblatt 1

Aufgabe 1 (5 Punkte)**:**

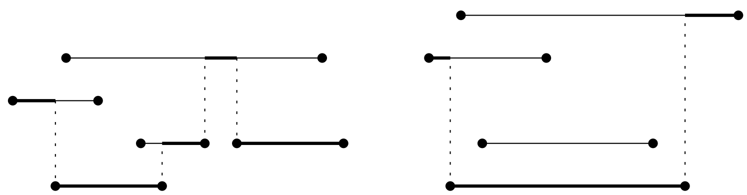

Abbildung 1: Example of a lower contour (the bold marked parts of the line segments).

We have n line segments in the plane. All these line segments are parallel to the x-axis, disjoint and all x-values are pairwise disjoint. Develop a sweep-algorithm that computes the lower contour of these line segments in time $\mathcal{O}(n\log(n))$ and prove the running time. You can assume that the line segments are in general position. This means that the line segments are disjoint.
The lower contour are the parts of the line segments visible from below. Figure 1 shows an example.

Aufgabe 2 (5 Punkte)**:**
Look at the running time estimation for the closest-pair algorithm from the lecture:
The update of *MinSoFar* required the comparison of the newly added point r to just a constant number of points within a specific rectangle. Because the Status Structure includes a balanced searchtree, the points located in this rectangle can be found quickly. In this exercise, we assume that the Status Structure only contains an unsorted list without a searchtree.

(a) What is the resulting worst case running time of the whole algorithm?

(b) Construct a scalable input that leads to the worst case running time of (a).

(c) How large do the areas *dead points* and *active points* become during the calculation of the worst case (b)?

Algorithmische Geometrie H01

You

April 16, 2024

A1

Assumptions

- The Segments are modelled by an Array S consisting of (x, y) coordinate Tuples. S will be given as input
- The balanced binary search tree T can hold tuples (x, y) as elements and is sorted by the y-value
- $T.min.x$, $T.min.y$ references the x and y values of the coordinate stored in the left most leaf of the tree. If the T is empty, $T.min.x$, $T.min.y$ each return 'infinite'

Algorithm 1 ALG(S)

```
Initialize a balanced binary search tree T
coordinates ← []                                ▷ Array that holds (x, y) coordinates
for i = 0 to 2n do
    old ← (T.min.x, T.min.y)
    if found(T, S[i]) then
        T.delete(S[i])
    else
        T.insert(S[i])
    end if
    if old ≠ (T.min.x, T.min.y) then
        coordinates.append(old)
    end if
    coordinates.append((T.min.x, T.min.y))
end for
return coordinates
```

Comments: The coordinates array holds (x, y) coordinates. (x_n, y_a), (x_m, y_a) are to be read in pairs, as the former describes the starting point of a segment and the latter the ending point. Added together these segments describe the lower contour. (∞, ∞) means that blank space is starting.

Runtime:

- for-loop: $\mathcal{O}(2n) = \mathcal{O}(n)$
- $T.delete$: $\mathcal{O}(\log(n))$
- $T.insert$: $\mathcal{O}(\log(n))$
- The rest has constant runtime.

Therefore the resulting runtime is $\mathcal{O}(n \log(n))$

0.1 A2

(a) The worst case running time of the whole algorithm is $\mathcal{O}(n^2)$.

Proof by construction of the worst case:

We describe the following set of points P with size n:

$$P = \{p_1, p_2, \ldots, p_n\}$$

$$p_i = (i/n, i)$$

One can immediately see that the Algorithms Sweep-Line processes the points in the order $p_1, p_2, \ldots, p_n$.

When processing p_i, all previous closest pairs are $|p_1p_2| = \sqrt{1 + (\frac{1}{n})^2}$ apart.

And since $|p_1p_2| \geq 1$ and all points lie on the strip $[0, 1] \times \mathbb{R}$, one must check all previous i points for p_i.

Giving us a total of $\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} \in \mathcal{O}(n^2)$ comparisons.

This gives us a running time of $\mathcal{O}(n^2)$.

(b) See above for the construction of the worst case.

(c) No point ever becomes a dead point, as all points are active points even until p_n is processed, since all x-coordinates lie in the interval $[0, 1]$, and the closest pair is always $|p_1p_2| = \sqrt{1 + (\frac{1}{n})^2} \geq 1$ apart.