Algorithms and Data Structures
Spring 2018
Exercises for Units 31

1. Construct some set of n segments along with some insertion(deletion) order for the incremental construction of a search structure so that for some query point the query time is $\Omega(n)$.

2. In the lecture we discussed a randomized method for building a point location query structure for a trapezoidation so that for any point in the plane the expected query time in $O(\log n)$. As a matter of fact we proved a more precise bound saying that for any query point q in expectation at most $5H_n \approx 5 \ln n$ basic comparisons are necessary to locate q.

   This problem is about achieving a query time guarantee of this sort for the worst case and not just in expectation.

   We make the following non-degeneracy assumptions: no two segements of S intersect at all (not even at endpoints); no two segment endpoints have the same y-coordinate.

   One approach of achieving good worst case query time proceeds as follows: Instead of removing just one segment s from S and building trapezuidation $\mathcal{T}(S)$ and query structure $\mathcal{Q}(S)$ from $\mathcal{T}(S \setminus \{s\})$ and $\mathcal{Q}(S \setminus \{s\})$ you could remove an entire subset $A \subset S$ and construct the desired structure from $\mathcal{T}(S \setminus A)$ and $\mathcal{Q}(S \setminus A)$. If the segments in A are sufficiently independent, i.e. no two touch the same trapezoid of $\mathcal{T}(S)$, then after locating a query point q in $\mathcal{T}(S \setminus A)$ it would take just constantly many more basic comparison to locate it in $\mathcal{T}(S)$ (at most one X-comparison and at most two Y-comparisons).

   (a) Convince yourself of this fact.

   Of course you would like to make A large, if possible a fixed fraction of S. This turns out to be possible: Let's make our lifes simpler and when removing the segments of A from $\mathcal{T}(S)$ we will actually leave the endpoints of those segments along with their horizontal extensions as part of the trapezoidation of $\mathcal{T}(S \setminus A)$. This means the independence condition on the segments of A is now that no two segments must bound the same trapezoid of $\mathcal{T}(S)$. Moreover, just at most one X-comparison is needed to locate a query point q in $\mathcal{T}(S)$ after knowing its location in $\mathcal{T}(S \setminus A)$.

   (b) Prove that for any set S of n segments there must be a subset A of at least $n/4$ independent segements, i.e. no two segments of A bound the same trapezoid of $\mathcal{T}(S)$.
   Hint: Create a graph whose nodeset is S and that for each trapezoid τ in $\mathcal{T}(S)$ has an edge that joins the two segments that bound the trapezoid (and remove duplicate edges). Argue that this graph is planar and hence has an independent set of the desired size.

   This independent segment subset removal step could be recursively repated $\log_{4/3} n$ times to remove all segments and be left with the trapezoids formed by the $2n$ extensions of segment endpoints. Since those extensions are ordered, $1 + \log_2 n$ comparisons would suffice to locate a query point in those trapezoids. This would lead to a total worst cae query time of $1 + \log_2 n + \log_{4/3} n \approx 1 + 4.92 \ln n$ for locating any query point in $\mathcal{T}(S)$, actually counting the worst case number of basic comparisons. However, there is a drawback:

   (c) Show that if the above approach is taken the space required for the query structure $\mathcal{Q}(S)$ can be $\Theta(n \log n)$.
   Hint: How many trapezoids does the trapezoidation at level i in this hierarchy contain? How many decision nodes may have to be employed $\mathcal{Q}()$ in order to go between two levels of the query structure.

A way around this space problem is to periodically just remove independent extensions (leading to Y-nodes in $\mathcal{Q}()$).

(d) Prove that always at least half of the extensions can be simultaneously removed without affecting any trapezoid with more than one extension.

Hint: What does the constraint graph for extensions that should not be removed simultaneously look like?

This leads to the following recursive approach (*):

Perform two levels of independent segment removals (one quarter each) followed by one level of extension removal (one half), and repeat recursively.

Note that the segment removals reduce the number of segments but increase the number of extensions, whereas the extension removals just decrease the number of extensions.

(e) Show that at each level of the recursion the number of extensions is at most 7 times the number of segments.

(f) Prove that if approach (*) is used the total space used for the query strusture is $O(n)$.

(g) What worst case query time does approach (*) guarantee? Try to be specific about the constants.