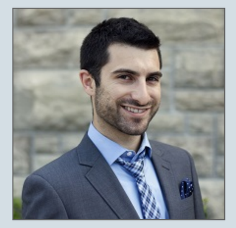
MADALGO
CENTER FOR MASSIVE DATA ALGORITHMICS



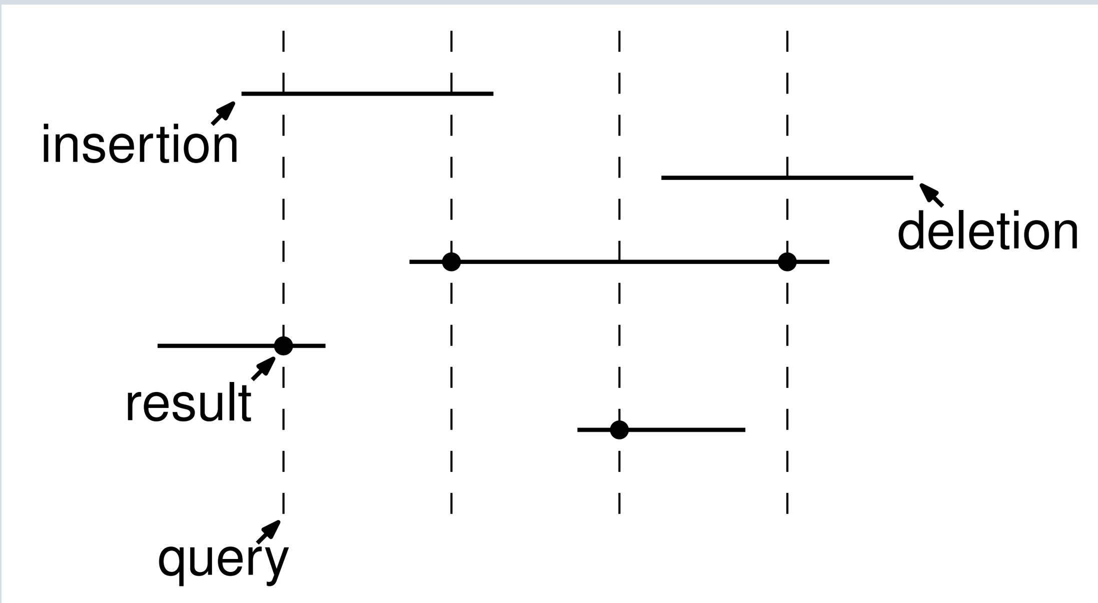
Bryan T. Wilkinson
Aarhus University

Danmarks Grundforskningsfond
Danish National Research Foundation

Offline Priority Queues, Lower Envelopes, and 2-D Visibility

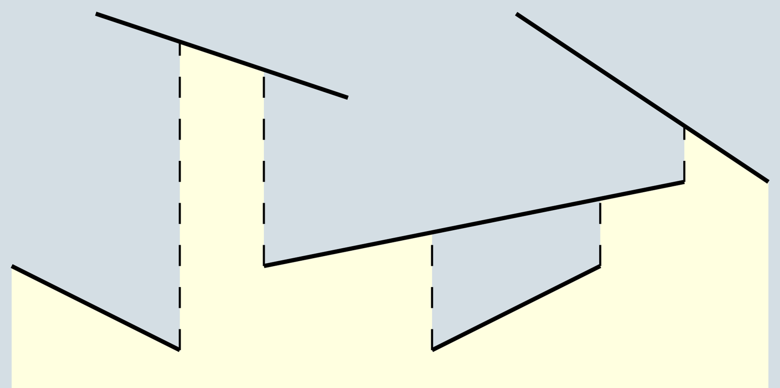
Problem

In the **priority queue** problem, we maintain a dynamic set of ordered elements in such a way that we always have efficient access to the minimum element. We consider the **offline problem** in which all insertions to, deletions from, and queries to the set are provided in advance in a list.

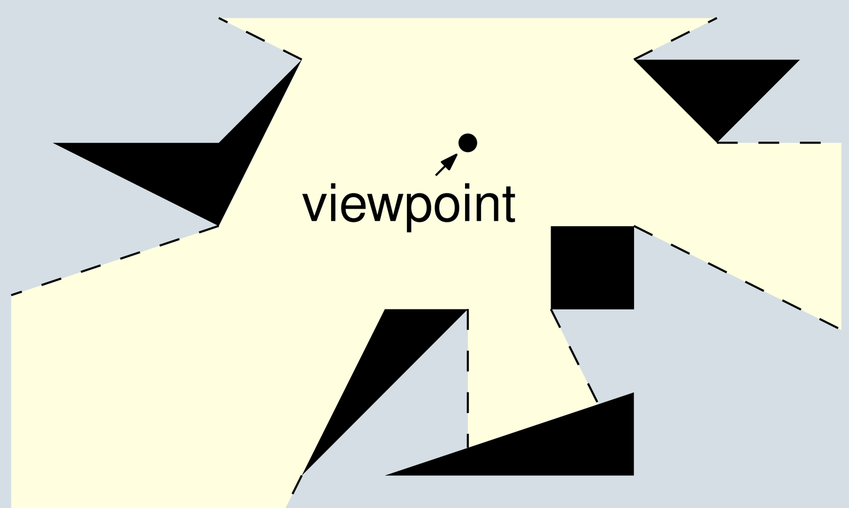


Motivation

If an algorithm solves the offline priority queue problem and only inspects elements via comparisons, then the algorithm can be adapted to compute the **lower envelope** of a set of **disjoint line segments**.



If we take the value of a segment to be its distance from a specific point (a viewpoint) instead of its height, the same algorithm can then be adapted to compute the **visible region** from the point amidst polygonal obstacles [4].



Previous Work

Efficient online priority queues support operations in $O(\log n)$ time. An algorithm for the offline problem that simply uses an online priority queue thus runs in $O(n \log n)$ time.

The lower envelope and the 2-D visibility problems have $\Omega(n \log n)$-time lower bounds via reduction from sorting [5]. These lower bounds hold in the pointer machine model but not the word RAM model.

Since we assume that the updates and queries of the offline priority queue problem are presorted in the time dimension, the same lower bound does not hold for the offline priority queue problem.

Eppstein and Muthukrishnan [1] give an algorithm for the offline priority queue problem that runs in $O(n)$ time when elements are restricted to the integers between 1 and n. The algorithm, thus, cannot be applied to solve the lower envelope and 2-D visibility problems.

New Results

We give several algorithms that run in $o(n \log n)$ time:

- Offline priority queues:
  - $O(n\alpha(n))$ time in the pointer machine model
  - $\Theta(n)$ time in the word RAM model
- Lower envelopes and 2-D visibility:
  - $\Theta(\mathrm{sort}(n))$ time in the word RAM model

In the word RAM model, the following bounds on $\mathrm{sort}(n)$ are known:

- $\Omega(n)$ time, since any algorithm must read the whole input
- $O(n \log\log n)$ deterministic time [2]
- $O(n\sqrt{\log\log n})$ randomized time [3]

We expect our word RAM algorithms to be practically efficient when implemented with a variant of radix sort.

References

[1] D. Eppstein and S. Muthukrishnan. *Internet packet filter management and rectangle geometry*. Symposium on Discrete Algorithms, 2001.
[2] Y. Han. *Deterministic sorting in $O(n \log\log n)$ time and linear space*. Journal of Algorithms, 2004.
[3] Y. Han and M. Thorup. *Integer sorting in $O(n\sqrt{\log\log n})$ expected time and linear space*. Symposium on Foundations of Computer Science, 2002.
[4] P. J. Heffernan and J. S. B. Mitchell. *An optimal algorithm for computing visibility in the plane*. Workshop on Algorithms and Data Structures, 1991.
[5] S. Suri and J. O'Rourke. *Worst-case optimal algorithms for constructing visibility polygons with holes*. Symposium on Computational Geometry, 1986.

New Approach

Our solutions combine efficient algorithms for two different variants of the original offline priority queue problem.

***Variant 1*: few queries**

We study the offline priority queue problem with an additional parameter: the number of queries, q, which we assume is less than n.

We obtain an algorithm that requires only $O(n + q \log q)$ time. We group elements that are deleted between the same queries, so that we only need to maintain q elements in an online priority queue.

***Variant 2*: hybrid data access model**

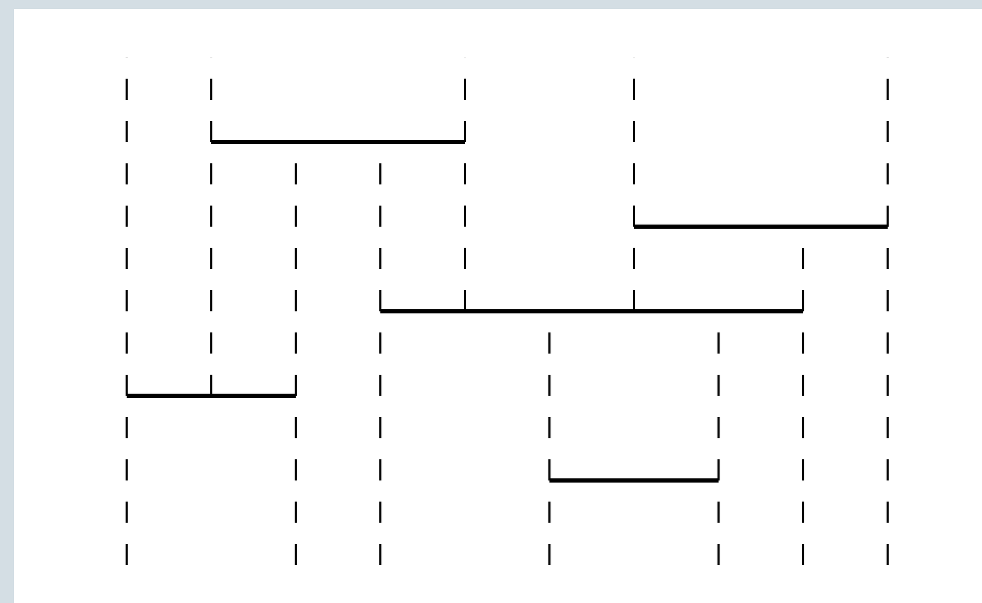
We consider a hybrid model in which we can exploit fast predecessor search data structures of the word RAM model for x-coordinates, but can only infer height information from comparisons.

We obtain an algorithm that requires only $O(n\,\mathrm{pred}(n))$ time, where $\mathrm{pred}(n)$ is the cost of predecessor search over x-coordinates.

Our final algorithm uses a solution to Variant 1 to decompose the problem into small subproblems in which the predecessor searches of Variant 2 require only $O(1)$ time.

Open Problem

The vertical decomposition of a set of segments is formed by shooting vertical rays from the endpoints of each segment. Note that the lower cells form the lower envelope of the segments.



The following is a question of great interest:

Given horizontal segments that have been presorted along both the x- and y-axes, can we compute the vertical decomposition in $O(n)$ time (or even $\mathrm{sort}(n)$ time) in the word RAM model?

MADALGO – Center for Massive Data Algorithmics, a Center of the Danish National Research Foundation