Sweeping Lines and Line Segments with a Heap

Julien Basch Leonidas J. Guibas* G.D. Ramkumar

Computer Science Department
Stanford University
Stanford, CA 94305
{jbasch,guibas}@cs.stanford.edu

Research & Development Division
Hitachi America Ltd.
3101, Tasman Drive, MS: 120
Santa Clara, CA 95054
ramkumar@hitachi.com

Abstract

Given n line segments in the plane, the Bentley-Ottmann sweep maintains the exact ordering of the intersections of the segments with a vertical line, as this line sweeps the plane from left to right. To accomplish this, every intersection between two segments must be processed, and the running time of the sweep can be $\Omega(n^2)$. In this paper, it is shown how a heap on the intersections can be maintained during the sweep. This new type of sweep processes $O(n \log^2 n)$ intersections when sweeping over lines and $O(n\sqrt{n \log n})$ intersections when sweeping over line segments. A lower bound of $\Omega(n \log n)$ is also established.

1 Introduction

One of the common introductory problems in geometric algorithms is that of finding all pairwise intersections in a family of line segments in the plane. The first non-trivial solution to this problem was given by Bentley and Ottmann [BO79], who introduced in 1979 the now familiar line sweep paradigm. A vertical line is moved ('swept') from the far left to the far right of the plane, and the exact order of the intersections with the segments is maintained as the line sweeps along. All vertices in the arrangement of segments are detected in this process.

From the point of view of the sweep line, the segments can be viewed as points which move up and down the line. The Bentley-Ottmann algorithm requires that we maintain the full sorted list of these points along the line and update it as new points are inserted (segment appearance), old ones are deleted (segment disappearance), and adjacent segments interchange position (intersection). In the same vein, we may be interested in maintaining other order-related information of these points along the sweep line, for example the topmost point, the k-th point from the top, etc. Since these variants do not require full sorting of the points during the sweep line, we may hope for an algorithm more efficient than a normal sweep. Indeed, if we just want to know the topmost point at all times, we can merely compute the upper envelope of the line segments, and it is known how to do this in $O(n \log n)$ time for an input of size n [Her89], irrespective of how many intersections the segments may have.

But can we actually compute the upper envelope of the segments using a line sweep method? One possibility is to put the segments intersecting the sweep line in a binary heap, where the priority of such a segment is the vertical coordinate of the intersection with the sweep line. At any given time, the validity of the heap structure can be guaranteed by one comparison per edge in the heap. The set of comparisons is called the *certificate list* of the structure. As the priorities of the heap elements change continuously with time, a comparison will fail exactly when the two segments involved intersect. For each edge of the heap, this time can be computed and put into an event queue, and the sweep can proceed from event to event exactly as in the case of the Bentley-Ottman sweep. The processing of an event requires a swap between the two intersecting segments in the heap. It also requires the descheduling and rescheduling of at most four events corresponding to the edges affected by the swap (Figure 1). A segment insertion in the sweep line is processed by inserting the segment into the heap, and rescheduling all events corresponding to edges disturbed by this operation.

What we have described here is a *kinetic heap*, an example of a *kinetic data structure* (a data structure for continuously changing data) as introduced by Basch, Guibas, and Hershberger [BGH97]. The kinetic heap is a data structure for maintaining the maximum of a collection of continuously changing numbers, where insertion of a new number into the collection, and dele-

*Supported in part by National Science Foundation grant CCR–9623851 and by US Army MURI grant DAAH04–96–1–0007.

To appear in the 13th Symposium of Computational Geometry, 1997.

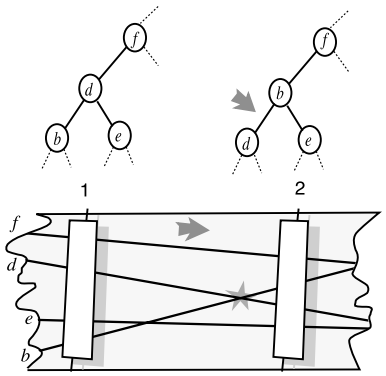

Figure 1: At the first position sweep line this figure, segments b and e are not in the same subtree, hence their intersection is not scheduled as an event. The first intersection to be scheduled is bd. At this point, the heap property is maintained by a swap between b and d. The events corresponding to the four edges around this pair are descheduled before the swap and rescheduled with the new identities of the lines involved after the swap

tion of an old number from it, are also allowed. Two other data structures have been proposed for the same purpose: the *heater* [BGR96], and the *kinetic tournament* [BGH97].

We measure the performance of a kinetic data structure by the number of events that it needs to process. On a set of n segments in the plane, the heater is a randomized structure that processes $O(n\alpha(n)\log n)$ events in expectation, while the kinetic tournament achieves the same bounds in the worst case.

The kinetic heap is arguably the most natural kinetic structure for maintaining the maximum. Its analysis, however, appears to be much less natural than that of its competitors. The primary reason for this state of affairs is that there is a complicated dependence between the exact structure of the heap at a given time and the prior history of the sweep. Nevertheless, we can show that if the sweep is done over entire lines (or segments with the same x-projection), then the number of events is $O(n\log^2 n)$ in the worst case. If the sweep is over line segments, we can only prove the weaker upper bound of $O(n\sqrt{n\log n})$. The best lower bound we can prove is $\Omega(n\log n)$ in both cases.

2 Notation

In what follows, we say that a vertex is *red* if it is processed as an event during the sweep, and *green* otherwise. Thus, a vertex is red if the two segments that define it are parent and child in the heap at the time they intersect. We are interested in bounding the number of red vertices. For a time t, corresponding to a position of the sweep line, we denote by t^- and t^+ the times immediately before and after t.

We assume that the lines or segments are in general position, so that no two events (intersections or endpoints) occur at the same time, and no three lines or segments intersect at a common point. We write $a <_t b$ to mean that the line or segment a is below b at time t.

3 Sweeping over lines

We now consider a binary heap sweeping over a set of n infinite lines. The heap structure remains fixed during the sweep, even though the node contents change over time. The *level* of a line a at time t, denoted by $\lambda_t(a)$, is the distance from the node containing a at t to the bottom level. Thus the level of a line is an integer between 1 and $\lceil \lg n \rceil$. We denote by $\hat{\lambda}_t(a)$ the highest level ever attained in the heap by line a, from the initial time until time t.

Lemma 3.1 *If ab is a red vertex with $b <_{t^-} a$, then $\hat{\lambda}_t(a) \geq \hat{\lambda}_t(b) + 1$.*

Proof: Consider the last time τ before t when b was at its maximum level $\hat{\lambda}_t(b)$. At time τ, line b was in some node ν of of the heap and from then on b only moved within the subtree U rooted at ν. If b is in ν at t^-, then a is in the parent of ν at that time, and the claim trivially holds.

Otherwise, as $b <_\tau a$, line a could not be in U at τ. Therefore, the path followed by a in the heap from τ to t^- goes from the complement of U into U, and thus passes through the parent of ν. The conclusion follows. □

Theorem 3.2 *The number of red vertices encountered during the maintenance of a standard binary heap in a sweep over n lines is $O(n\log^2 n)$.*

Proof: We use a potential function argument. For a line a in the heap, define its potential $\phi_t(a)$ at time t to be

$$\phi_t(a) = \hat{\lambda}_t(a)(\lambda_t(a) - \hat{\lambda}_t(a)).$$

The potential of the entire heap is the sum of the potentials of all the lines. Note that this potential is 0 before the start of the sweep (for every a, $\lambda_t(a) = \hat{\lambda}_t(a)$); it is non-positive for all t (because $\lambda_t(a) \leq \hat{\lambda}_t(a)$) and at the end of the sweep its absolute value is at most $O(n\log^2 n)$. We now show that the occurrence of a red vertex decreases the total potential by at least 1.

Consider a swap at time t between a parent a and a child b in the heap. The potential of any line other than a or b doesn't change. Also, the quantity $\hat{\lambda}_{t^-}(a)$ doesn't change, so that the potential change for a is $\phi_{t^+}(a) - \phi_{t^-}(a) = -\hat{\lambda}_{t^-}(a)$.

There are two cases for b: either it breaks its current $\hat{\lambda}$ record (i.e. $\lambda_{t^-}(b) = \hat{\lambda}_{t^-}(b)$), or it does not. In the first case, $\hat{\lambda}_{t^-}(b) = \lambda_{t^-}(b)$ and $\hat{\lambda}_{t^+}(b) = \lambda_{t^+}(b)$, so $\phi_{t^-}(b) = \phi_{t^+}(b) = 0$, and b's potential doesn't change. The decrease in potential is therefore $\hat{\lambda}_{t^-}(a)$, which is at least 1 because a is not on the bottom level before the swap.

In the second case, the potential of b increases by $\hat{\lambda}_{t^-}(b)$, and the net potential change is $\hat{\lambda}_{t^-}(b) - \hat{\lambda}_{t^-}(a)$, which is at most -1 by Lemma 3.1. □

4 Sweeping over line segments

The analysis used in the previous section completely breaks down in the case of line segments, as Lemma 3.1 no longer hold. Instead, we use another potential argument reminiscent of the one used independently by several authors [ELSS73, Gus79, EW85] for proving upper bounds on the k-level of an arrangement of lines.

Lemma 4.1 *The number of red vertices processed during the maintenance of a standard binary heap in a sweep over n line segments is $O(n\sqrt{n \log n})$.*

Proof: Let us order the n segments by increasing slope, and denote by $r(s)$ the rank of a segment s in this ordering. Let $H(\nu)$ be the subtree rooted at a node ν, and let $r_t(\nu)$ be the rank of the segment in node ν at time t.

For a node ν in the heap, define its potential $\phi_t(\nu)$ at time t to be:

$$\phi_t(\nu) = \sum_{\mu \in H(\nu)} r_t(\mu) .$$

In other words, the potential of a node is the sum of the ranks of all segments in the subtree rooted at that node. The potential of the entire heap is the sum of the potentials of all the nodes of the heap. It is 0 at both ends of the sweep (when the heap is empty).

If two nodes μ, ν are parent and child, a swap of their contents at t changes the potential by $r_{t^-}(\mu) - r_{t^-}(\nu)$, which is at most n. In a binary heap, a segment insertion/deletion is implemented by $O(\log n)$ swaps and an insertion/deletion at the bottom of the heap. This sequence of operation change the potential by $O(n \log n)$. Hence, during the entire sweep, the total potential *increase* due to endpoints is $O(n^2 \log n)$.

Let R be the set of red vertices. When $v \in S$ is processed, the potential can only decrease, as it is the higher ranked segment that goes up one level in the heap, and it will always remain non-negative. Hence, if we denote by $\delta(v)$ the absolute difference in rank between the two segments that define a vertex v, we have:

$$\sum_{v \in R} \delta(v) \leq O(n^2 \log n) .$$

Therefore, for any fixed B, the number of vertices $v \in S$ with $\delta(v) \geq B$ is $O(\frac{n^2 \log n}{B})$. Moreover, there are at most nB vertices with $\delta(v) \leq B$. Choosing appropriately $B = \sqrt{n \log n}$ gives the desired bound. □

5 Lower bound

We describe an arrangement of n lines where the number of red vertices seen while sweeping with a heap is $\Omega(n \log n)$. Place each line so that it is tangent to an upward facing parabola. Each line appears on the upper envelope exactly once. In particular, a line that starts at the bottom of the heap has to appear at the root at the time it is on the upper envelope. This cannot be done in less than $\log n$ swaps. As there are $n/2$ lines that start at the bottom of the heap. As a swap increases the level of only one line at a time, the total number of swaps generated by this arrangement is $\Omega(n \log n)$.

6 Conclusion

In this paper, we established bounds on the number of events that need to be processed to maintain a heap while sweeping over an arrangement of lines and line segments. As the kinetic heap is the most natural structure to maintain the extremum of a set of points moving along a line, we would like to see the bounds tightened and extended to the case of algebraic curves or arcs of low degree.

References

[BGH97] J. Basch, L. J. Guibas, and J. Hershberger. Data structures for mobile data. In *8th Symposium on Discrete Algorithms*, pages 747–756, 1997.

[BGR96] J. Basch, L.J. Guibas, and G.D. Ramkumar. Reporting red-blue intersections between connected sets of line segments. In J. Diaz and M. Serna, editors, *Algorithms — ESA '96*, LNCS 1136, pages 302–319, sep 1996.

[BO79] J. L. Bentley and T. A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.*, C-28:643–647, 1979.

[ELSS73] P. Erdős, L. Lovász, A. Simmons, and E. Straus. Dissection graphs of planar point sets. In J. N. Srivastava, editor, *A Survey of Combinatorial Theory*, pages 139–154. North-Holland, Amsterdam, Netherlands, 1973.

[EW85] H. Edelsbrunner and E. Welzl. On the number of line separations of a finite set in the plane. *J. Combin. Theory Ser. A*, pages 15–29, 1985.

[Gus79] D. Gusfield. Bounds for the parametric spanning tree problem. In *Proc. Humbolt Conf. on Graph Theory, Combinatorics and Computing*, pages 173–183, 1979.

[Her89] J. Hershberger. Finding the upper envelope of n line segments in $O(n \log n)$ time. *Inform. Process. Lett.*, 33:169–174, 1989.