Optimal Randomized Parallel Algorithms for Computational Geometry

John Reif Sandeep Sen

Presented by Pat Morin

J. Reif, S. Sen, "Optimal Randomized Parallel Algorithms for Computational Geometry," in *Algorithmica*, 7, pp. 91–117, 1992.

Contents

- Random mating and Kirkpatrick's algorithm
- Segment trees and plane sweep trees
- Nested plane sweep trees
- Applications of nested plane sweep trees
- Conclusions

Kirkpatrick's Point Location Algorithm

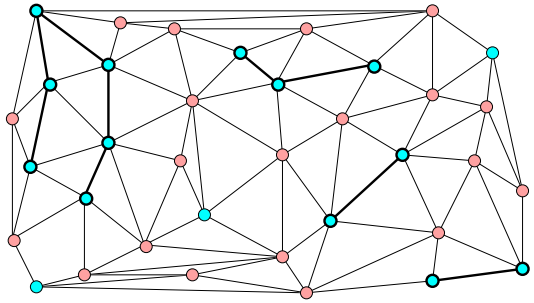
Recall Kirkpatrick's algorithm for point location in a triangulated planar straight line graph.

- During each round, an independent set of bounded degree vertices is chosen and deleted.
- The number of vertices chosen is a constant fraction of the total number of vertices.
- $\Rightarrow$ The algorithm completes after $O(\log n)$ rounds.
- Finding the independent set takes time linear in the current number of vertices, yielding an $O(n \log n)$ time algorithm.

Parallelizing Kirkpatrick's Algorithm

1. Associate a processor with each vertex and each edge.
2. Randomly assign a sex (male/female) to each vertex v with $degree(v) \leq d$ with equal probability. The *males* are our candidates for the independent set.
3. For each edge, if it connects two males then mark both males as dead.
4. The independent set consists of all the living males.
5. Delete the living males, retriangulate the holes created, and repeat.

Example



Analysis of Algorithm

- Choosing our independent set takes constant time.
- There exists an independent set of size αn, $O < \alpha < 1$ in which the shortest path between any two vertices in the set is of distance ≥ 3.
- With high probability, roughly half of these vertices are male.
- A male node survives with probability $\geq \frac{1}{2^d}$.
- $\Rightarrow$ With high probability, the size of the independent set is a constant fraction of the number of vertices in the graph.
- $\Rightarrow$ the running time of the algorithm is $\tilde{O}(\log n)$.

Applications

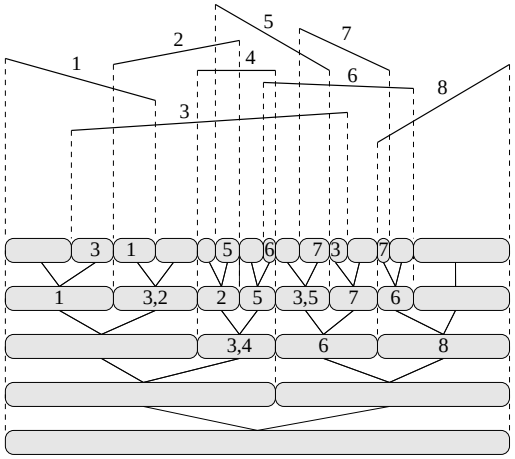
- Using concurrent read capability, n queries can be executed on a planar subdivision of size n in $\tilde{O}(\log n)$ time using $O(n)$ processors.
- The Voronoi diagram of $O(n)$ points in the plane can be constructed in $O(\log^2 n)$ expected time using $O(n)$ processors (Aggarwal *et. al.* 1985).

Segment Trees

A segment tree is a data structure used to store 2D non-intersecting line segments. It has the following properties:

- It is a complete binary tree on the ranges defined by the projection of the $2n$ endpoints on the x axis.
- The range of a parent is the union of the range of its two children.
- Assocated with each node is a *catalogue* containing the list of segments which cover the node's range, but do not cover the range of the node's parent.
- The catalogues are sorted by the *below* relationship.

Segment Trees (cont'd)



Segment Trees (cont'd)

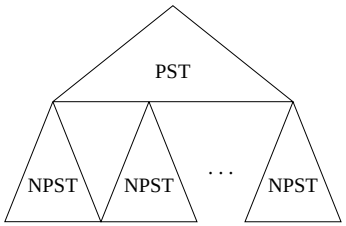
- A segment appears in at most 2 node on each level.
- A segment appears at most once in any path from the root to a leaf.
- A *plane sweep tree* is a segment tree augmented with some extra information.
- A parallel plane sweep tree construction algorithm exists which runs in $O(\log^2 n)$ time using $O(n)$ processors.
- The same algorithm can construct a plane sweep tree in $O(\log n)$ time using $O(n^{1+\epsilon})$ processors.

Applications of Segment Trees

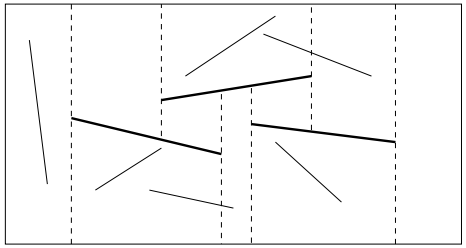
- Next element search
- Planar point location
- Trapezoidal decomposition of non-intersecting line segments
- Bichromatic line segment intersection reporting
- Line segment intersection detection

The Nested Plane Sweep Tree

- Construct a plane sweep tree from a sample of $\sqrt{n}$ segments.
- This partitions the plane into $O(\sqrt{n})$ trapezoids.
- For each trapezoid, recursively construct a nested plane sweep tree for all the remaining segments which intersect it.



The Nested Plane Sweep Tree (cont'd)



Constructing the Nested Plane Sweep Tree

- We want a trapezoidal decomposition such no trapezoid intersects more than $O(\sqrt{n}\log n)$ segments.
- In this case the running time is $T(n,n) = T(\sqrt{n}, \sqrt{n}\log n) + O(\log n) = O(\log n)$.
- With probability $\frac{1}{2}$, a randomly chosen set of line segments will satisfy this condition.
- $\Rightarrow$ If we choose $c\log n$ sets of segments, then the probability of at least one of them satisfying this condition is $1 - \frac{1}{n^c}$.

Choosing the Sample Segments

- The problem is we don't have time to completely test each set of segments.
- We can however, test a random sample of size $O(\frac{n}{\log n})$.
- If X is the number of segments which intersect a given trapezoid, then we expect to find $O(\frac{Xn}{\log n})$ of these segments in our random sample.
- We can use Chernoff bounds to show that this expected value occurs with high probability.

Choosing the Sample Segments (cont'd)

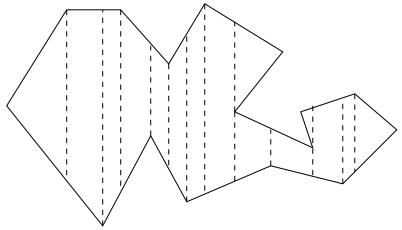
Algorithm Sample Select

1. Choose $O(\log n)$ sets of samples, where each set has $\sqrt{n}$ segments in it.
2. Partition the processors into $O(\log n)$ groups of size $O(\frac{n}{\log n})$.
3. Have each group of processors test a set of samples using $O(\frac{n}{\log n})$ randomly selected segments.

- Step 1 and Step 2 take constant time.
- Step 3 takes $O(\log n)$ time since each processor executes $O(1)$ queries on a plane sweep tree of size $\sqrt{n}$.

Trapezoidal Decomposition

Trapezoidal decomposition of a simple polygon can be done in $\tilde{O}(\log n)$ time using $O(n)$ processors.



1. Build the nested plane sweep tree for the edges of the polygon.
2. Locate the edge directly above and below each vertex of the polygon.

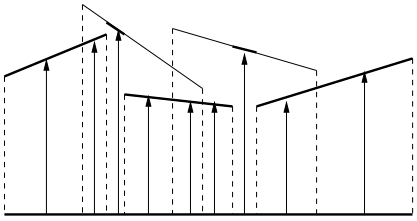
Triangulation

Triangulation of a simple polygon can be done in $\tilde{O}(\log n)$ time using $O(n)$ processors.

1. Compute a trapezoidal decomposition of the polygon.
2. Decompose the polygon into one-sided monotone polygons using the trapezoidal decomposition.
3. Each of the one sided monotone polygons can be triangulated in linear time (Attalah and Goodrich 1986).

Visibility from a Point

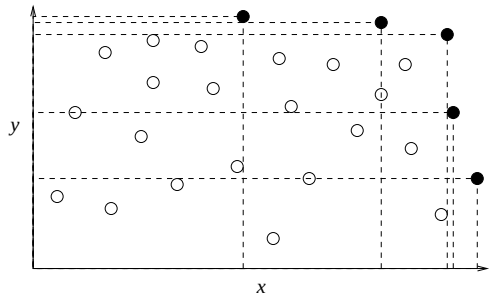
- The visibility polygon of a point can be computed in $\tilde{O}(\log n)$ time using $O(n)$ processors.



1. Sort the endpoints of the segments on their x-coordinate, creating $2n-1$ intervals.
2. For each interval, I, determine which segment is visible by performing a next element query of the form $(mid(I), -\infty)$.

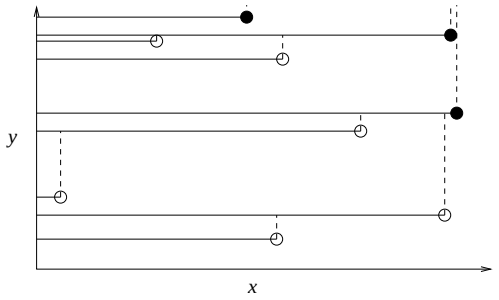
The Maxima Problem

Given a set of points in $\Re^d$ determine the subset of these which are not dominated on all dimensions by any other point.



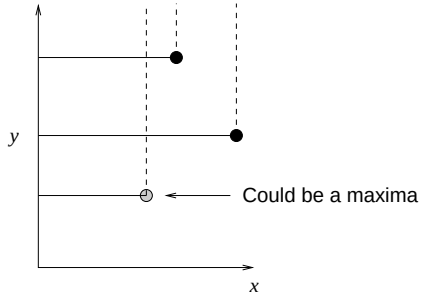
The 2-D Maxima Problem

The 2-D maxima problem can be solved using the nested plane sweep tree in $\tilde{O}(\log n)$ time using $O(n)$ processors.



The 3-D Maxima Problem

Note that a point is a 3-D maxima if none of the segments above it have a larger z coordinate.



The 3-D Maxima Problem (cont'd)

- For each catalogue segment, we store the maximum z-coordinate of all the items above it in the catalogue. This is easily computed using a prefix sum operation.
- There are at most $O(n \log n)$ catalogue segments in all, and any single catalogue contains at most n segments.
- A prefix sum of n items can be computed in $O(\log n)$ time using $O(\frac{n}{\log n})$ processors.
- $\Rightarrow$ The tree can be augmented with this information using only an additional $O(\log n)$ time, yielding an $O(\log n)$ time algorithm for 3-D maxima.

The 2-D Two Set Dominance Problem

Given two sets of points, U and V, of size n, for each point in V count the number of points in U which dominate it.

1. Make segments from the points in U and build a nested plane sweep tree on these segments.
2. Search the tree for each point in V, but when searching a catalogue, sum the number of items in the tree which fall above the point.

This algorithm performs 2 set dominance counting in time $O(\log n)$ using $O(n)$ processors.

Conclusions

- The segment tree and its parallel counterparts the plane sweep tree and nested plane sweep tree are useful data structures for solving a wide variety of problems in computational geometry.
- The nested plane sweep could be adapted to the the more realistic coarse grained setting.
- The segment tree can be scaled to higher dimensions (range trees), can the nested plane sweep tree?