Randomized Incremental Construction of Trapezoidation

Input: Set N of n of segments.

Output: $H(N)$ is the trapezoidal decomposition.

Main Steps in the Randomized Incremental Algorithm

1. Take a random permutation of set of segments N.

2. Incrementally compute the trapezoidal decomposition, $H(N^0), \cdots, H(N^n)$.

3. Assume that we have $H(N^i)$ and want to add the next segment $S = S_{i+1}$. Let the end-points of S be s_0 and s_1.

Anil Maheswari
School of Computer Science
Carleton University

Travel and Split:

4. Assume that we can locate which trapezoid the endpoint s_0 lies (*using conflict lists*).

5. Travel in $H(N^i)$ from s_0 to s_1 along S. Say, we encounter the faces $R_0, R_1, \cdots$ in $H(N^i)$ one by one.

6. If $f \in R^i$, then traversal along f requires $O(face - length(f))$ time. *Face-length(f)* denotes the number of vertices of $H(N^i)$ adjacent to f.

7. Split each of the traversed face by S, by raising vertical walls where-ever appropriate (at end-points of S and at the intersection point of S with any segments of N^i. Notice that a face f can be split in at most 4 pieces an can be achieved in $O(face - length(f))$ time.

8. Also contract the vertical walls in $H(N^i)$ that are intersected by S. This is done in $O(1)$ time and the faces are merged which are adjacent to the erased segment.

Proposition 3.1.1: Once we know the trapezoid in $H(N^i)$ containing one of the end-points of S, $H(N^i)$ can be updated to $H(N^{i+1})$ in time proportional to $\sum_f face-length(f)$, where f ranges over all trapezoids in $H(N^i)$ intersecting S.

Anil Maheswari
School of Computer Science
Carleton University

Conflict-List $L(f)$

How do we locate an end-point of S in $H(N^i)$?

1. Maintain an unordered conflict list $L(f)$ with each trapezoid $f \in H(N^i)$. $L(f)$ contains the endpoints within f of the segments in $N \setminus N^i$

2. For each segment in $N \setminus N^i$, maintain a conflict pointer.

3. End-point location problem becomes easy! How do we update the conflict lists due to split and merge of faces?

4. When we split/merge a face f due to addition of S, we should also split/merge $L(f)$.

Anil Maheswari
School of Computer Science
Carleton University

5. Split is done in time proportional to the size of the conflict lists. If f is split in f_1 and f_2, for each point in $L(f)$ check whether it lies is in f_1 or f_2.

6. Merging of the conflict list is done simply concatenating them in $O(1)$ time, which takes $O(1)$ time per merge.

Proposition 3.1.2: The cost of updating conflict lists is $O(\sum_f l(f))$, where f ranges over all trapezoids in $H(N^i)$ intersecting S.

Analysis

1. Cost of adding $S = S_{i+1}$ is $O(\sum_f face - length(f) + l(f))$, where f ranges over all trapezoids in $H(N^i)$ intersecting S.

2. (*Backward Analysis*) Fix N^{i+1}, and compute the cost of deleting a random segment S. Conditional expected cost is proportional to

$$\frac{1}{i+1} \sum_{S \in N^{i+1}} \sum_{g} (face - length(g) + l(g))$$

where g ranges over the trapezoids of $H(N^{i+1})$ adjacent to S.

Anil Maheswari
School of Computer Science
Carleton University

3. Each trapezoid is adjacent to at most 4 segments and the total conflict list size is $2(n-i)$. Therefore the sum is proportional to

$$\frac{n-i+|H(N^{i+1})|}{i+1} = O(\frac{n+k_{i+1}}{i+1})$$

   where k_{i+1} denotes the number of intersections among the segments of N^{i+1}.

4. Expected cost of $(i+1)$th addition, conditional on a fixed N^{i+1}, is $O(n/j + kj/n^2)$, where $j = i+1$.

5. Above bound does-not depend upon N^{i+1}, hence the expected cost of the whole algorithm is $O(n\sum 1/j + k\sum j/n^2) = O(n\log n + k)$.

Lemma 3.1.3: Fix $j > 0$. The expected value of k_j, assuming that N^j is a random sample of size i, is $O(kj^2/n^2)$.

Proof:
-Consider the set of all intersections among the segments in N.
-For a fixed intersection v in this set, define a r.v. I_v.
-$I_v = 1$ if and only if v occurs in $H(N^j)$.
-$k_j = \sum I_v$.
-This happens with probability $O(j^2/n^2)$.
-The expected value of I_v is $O(j^2/n^2)$.

Theorem 3.1.4: A trapezoidal decomposition formed by n- segments in the plane can be constructed in $O(n \log n + k)$ expected time/ (k is the total number of intersection points)

Introducing History

Incremental but not online! Requires $N \setminus N^i$

Objective: An *online search structure* in place of the conflict lists. (Similar to the History used in randomized search structure).

1. Remember the history, history(i), of previous additions. Leaves of history(i) will point to the trapezoids in $H(N^i)$, and given a query point (end-point of segment) we will be able to locate the trapezoid in $O(\log i)$ with h.p.b.

2. Thus the total cost of the point location will be $O(\sum_i \log i) = O(n \log n)$ time, and therefore the total cost of the algorithm remains $O(n \log n + k)$.

3. What is history(i)?

History(0) is the root, corresponds to the whole plane.
Inductively, assume history(i) has been defined.
History(i+1) is defined as follows.

4. If S intersects a trapezoid $f \in H(N^i)$, say f is *killed*. The leaf corrsponding to f in history(i) is marked killed by S (Note that these leaves are not removed!).

5. Allocate new nodes for the new trapezoids. If f is killed and g is the newly created trapezoid, then g is a child of f if $f \cap g$ is nonempty. Note that the number of children of any killed leaf in history(i) is bounded (≤ 4).

6. History(i) is a directed acyclic graph with 1 root, outdegree of any node < 4.

7. Its trivial to see how query will be answered.

Lemma 3.1.5 For a fixed query point, the length of the search path in history(i) is $\tilde{O}(\log i)$.

Proof: Very similar to the analysis of randomized search trees.

1. Let p be the query point. Follow search path backwards.

2. Think of removing segments from N^i, one at a time, in the reverse order, $S_i, \cdots, S_1$.

3. Define a 0-1 random variable, V_j, so that $V_j = 1$ iff the trapezoid in $H(N^j)$ containing p is adjacent to S_j. (This trapezoid will be destroyed in this imaginary deletion of S_j)

4. The length of the search path in history(i)=$\sum_{j+1}^{i} V_j$.

5. Every segment in N_j is equally likely to occur as S_j.

6. Probability $(V_j = 1) \leq 4/j$, since at most 4 segemnts in N^j are adjacent to the trapezoid in $H(N^j)$ containing p

7. Expected Value of V_j is $O(1/j)$.

8. By linearity of expectation, it follows that the expected length of the search path is $O(\sum 1/j) = \log n)$.

9. The h.p.b can be shown using the Chernoff techniques.

The proof works because of the following **IMPORTANT** observation.

Bounded Degree Property: The answer to every query over $H(N^i)$ is defined by a bounded number of objects in N^i.

How do we do for arbitrary queries?

Its easy, since total number of search paths are polynomial, bounded by $O(i^2)$, and search paths terminating in a trapezoid are identical. This can be made true if we put vertical attachments everywhere!

Two interesting and Special Cases: Planar Graphs and Simple Polygons.

Convex Polytopes

Input: Set N of n half-spaces in $\Re^d$, $d = 2, 3$.

Output: $H(N)$= Convex Polytope defined by these half spaces.

Randomized Incremental Algorithm

1. Assume that i half-spaces have been added and we have $H(N^i)$. Assume that $H(N^i)$ is always bounded (e.g. by a big box!). Assume $H(N^i)$ is represented by its facial lattice.

2. Consider addition of a (i+1)st random half-space $S = S_{i+1}$.

3. Obtain $H(N^{i+1})$ from $H(N^i)$ by splitting off the cap given by $cap(S_{i+1}) = H(N^i \cap \bar{S})$.

Anil Maheswari
School of Computer Science
Carleton University

4. Assume we know of a vertex $p \in H(N^i)$ that conflicts with S, $p \in \bar{S}$. If no vertex in conflict with S then S is redundant.

5. Consider the edge skeleton of $H(N^i)$ and consider the portion the edge skeleton lying in $\bar{S}$. This will be a connected subgraph, containing p.

6. Its straight forward to remove the portion of $H(N^i)$ in $\bar{S}$, and we obtain $H(N^{i+1})$

Analysis

1. Assume that we are given p, the initial conflicting vertex.

2. Addition of $S = S_{i+1}$ takes time proportional to the removal of destroyed vertices in $H(N^i)$ and the time to create new vertices in $H(N^{i+1})$.

3. Each created vertex can be destroyed once, and hence we just count the number of created vertices!

4. Amortized cost of (i+1)st addition is proportional to the number of newly created vertices in $H(N^{i+1})$. These are the vertices that are contained in δS_{i+1}, and let this number be $m(S_{i+1}, N^{i+1})$.

5. (*Do Backward Analysis*) Fix N^{i+1}, imagine deleting S_{i+1}. The conditional expected cost will be

$$\frac{1}{i+1} \sum_{S \in N^{i+1}} m(S, N^{i+1}).$$

6. Each vertex in $H(N^{i+1})$ is contained in two bounding lines in 2-dimension, and 3 bounding planes in 3-dimension (assume general position).

7. Since each vertex is adjacent to bounded number of objects, the sum is bounded by d times the number of active vertices on the convex polytope $H(N^{i+1}$.

8. The number of active vertices is $O(i)$, for $i = 2, 3$. Thus the expected amortized cost of (i+1)st addition, conditional of a fixed N^{i+1}, is $O(1)$.

9. Since bound is independent of N^{i+1}, it holds in general.

Lemma 3.2.1: (*Summary so far*) The expected number of newly created vertices in the jth random addition is $O(1)$, assuming that we are given a vertex in the conflict!