Review

Last time we defined the 2 properties of Delaunay triangulation:
-  Global
-  Local

Recall that a Delaunay triangulation satisfies the local property if for any triangle abc and abd, d is not inside the circumcircle of abc. Any triangle that satisfies the local property also satisfies the global property.

Today we’ll discuss the Algorithms for computing Delaunay Triangulation. There are several ways to compute Delaunay triangulation:

a)  Edge flip
b)  Incremental Algorithm
c)  Divide and conquer.

Edge Flip

Simply put, an edge flip involves the transformation of a triangulation by deleting the diagonal of 2 neighboring triangles and inserting another. This algorithm is only defined for internal edges. We don’t flip on the convex hull.

Consider two adjacent triangles of T:
If the 2 triangles form a convex quadrilateral, we could have an alternate triangulation by performing an edge flip on their shared edge. If an edge flip improves the triangulation the first edge is called illegal. In other words, an edge is illegal if we can locally increase the smallest angle by flipping that edge. Also an edge is illegal if it fails the $\pi$ test.
**Prove:** Flipping an illegal edge improves the smallest angle of the two triangles.

This follows from Thale’s theorem. Let \( C \) be a circle, and \( l \) a line intersecting \( C \) at points \( a \) and \( b \). Let \( p, q, r \) and \( s \) be points lying on the same side of \( l \), where \( p \) and \( q \) are on \( C \), \( r \) inside \( C \) and \( s \) outside \( C \). Then observe that since \( r \) is inside \( C \) then it will subtend a larger angle than \( s \) that is outside \( C \). Hence \( \anglearb > \angleapb = \angleaqb > \angleasb \)

Suppose we have two adjacent triangles \( p_i, p_i, p_j \) and \( p_i, p_j, p_k \) (as shown below) such that their union forms a convex quadrilateral \( p_i, p_i, p_k, p_j \). Suppose triangle \( p_i, p_i, p_j \) violates the \( \pi \) test such that the edge \( p_i, p_j \) lies inside the circumcircle of \( p_i, p_j, p_k \) then if we flip the edge \( p_i, p_j \) it will follow that resulting triangles pass the \( \pi \) test (ie. That the circumcircles of the two resulting triangles \( p_i, p_k, p_j \) and \( p_i, p_k, p_j \) are now empty and the observation above about Thale’s theorem proves that the minimum angle increases at the same time.
Algorithm
a) Construct a triangulation of point set P.
b) while ( \( \exists \) internal edge \( e = (u, v) \) that fails the \( \pi \) test) flip (e).
c) This algorithm although slow will eventually terminate since there is a finite number of triangulations.

Analysis of Edge Flip Algorithm
An edge is subject to flip operation at most once. Since there are at most \( \binom{n}{2} \) edges of a graph with \( n \) vertices, there are fewer than \( \binom{n}{2} \) edge flip operations. Hence it takes \( O(n^2) \) time to construct Delaunay Triangulation, starting from an arbitrary triangulation.

Theorem: \( \forall \) point sets \( P \), \( DT(p) \) maximizes the minimum angle in the triangulation.

Suppose we have a triangulation \( T \), let \( A(T) \) be an angle vector of the sorted angles of \( T \) \( (\alpha_1, \alpha_2, \alpha_3, ..., \alpha_{3n}) \) where \( \alpha_1 \) is the smallest angle. Imagine \( T \) is not locally Delaunay, and so there is a triangulation \( T' \) resulting from an edge flip. Then \( A(T) \) is larger than \( A(T') \) if and only if there exists \( \alpha_i = \alpha'_i \) for all \( j < i \) and \( \alpha_i > \alpha'_i \). So the resulting Delaunay triangulation is lexicographically maximum. Since the number of triangulations is finite, and each triangulation generated is distinct, the algorithm terminates.

Incremental Algorithm
Basic Idea: Randomly order the points, then insert the points one by one, each time updating the triangulation so it is Delaunay. We can do this by:

a) Locating the triangle containing the point to be added
b) Inserting the point and connecting it to nearby vertices.
c) Making edge flips to return the triangulation to Delaunay.
d) Test adjacent triangles to each triangle which changes. Initially this is just the three incident on the new point, but edge flips can propagate into neighboring triangles.
Initially, we introduce three new points \{ p_{-2}, p_{-1}, p_0 \} chosen large enough so that in $\text{DT}\{ p_{-2}, p_{-1}, p_0, p_1, p_2, \ldots, p_n \}$ the convex hull $\text{CH}(p_1, p_2, \ldots, p_n)$ are DT edges. We produce DTs one after the other.

\[
\begin{align*}
\text{DT}_0 & \{ p_{-2}, p_{-1}, p_0 \} \\
\text{DT}_1 & \{ p_{-2}, p_{-1}, p_0, p_1 \} \\
\text{DT}_2 & \{ p_{-2}, p_{-1}, p_0, p_1, p_2 \} \\
& \vdots \\
\text{DT}_i & \{ p_{-2}, p_{-1}, p_0, p_1, p_2, \ldots, p_i \} \\
& \vdots \\
\text{DT}_n & \{ p_{-2}, p_{-1}, p_0, p_1, p_2, \ldots, p_n \}
\end{align*}
\]

**Algorithm**

Input: a point set $P = \{ p_1, p_2, \ldots, p_n \}$ of any $n$ points in the plane.
Output: A Delaunay triangulation of $P$.

a) Let $p_{-1}, p_{-2}, p_{-3}$ be a set of three points such that $P$ is contained in the triangle $p_{-1}, p_{-2}, p_{-3}$.
b) Initialize $T$ as the triangulation containing the single triangle $p_{-1}, p_{-2}, p_{-3}$.
c) Compute a random permutation $p_1, p_2, \ldots, p_n$ of $P$.
d) For $r = 1 \ldots n$ do (* insert $p_r$ into $T$: *)
   Find a triangle $p_i p_j p_k \in T$ containing $p_r$
   If $p_r$ lies in the interior of the triangle $p_i p_j p_k$
       Then add edges from $p_r$ to the three vertices of $p_i p_j p_k$
       thereby splitting $p_i p_j p_k$ into three triangles.
       Insert $(p_r p_i), (p_j p_k), (p_k p_i)$ in a queue to check for $\pi$-test.
   Else (* $p_r$ lies on an edge of $p_i p_j p_k$ say the edge $p_i p_j$ *)
       Add edges from $p_r$ to $p_k$ and to the third vertex $p_1$ of the other triangle that is incident to $p_i p_j$ thereby splitting the two triangles incident to $p_i p_j$ into four triangles.
       Insert $(p_r, p_i), (p_1 p_j), (p_j p_k)$ in a queue to check for $\pi$-test.
While the queue is not empty dequeue an edge () from the queue and test for \( \pi \) condition. If illegal \(
\text{FLIP}(p_r, p_k, p_i, T)
\) discard \(p_1, p_2, p_3\) with all their incident edges from \(T\).
Return \(T\).

The next drawing illustrates the two cases in the If and Else statements above.

- \(p_r\) lies in the interior of a triangle
- \(p_r\) falls on an edge

The incremental algorithm consists of two main parts:

1. Locate a triangle (or an edge), containing the inserted point.
2. Insert the point into the current triangulation, making the necessary adjustments.

The Delaunay criterion can be reduced in the second step to a simple empty circle test: if a circumcircle of a triangle contains another triangulation vertex in its circumcenter, then the edge between those two triangles should be ``flipped'' so that two new triangles are produced. The testing is done in a recursive fashion consistent with the incremental nature of the algorithm. When a new node is inserted inside a triangle, three new triangles are created, and three edges need to be tested. When the node falls on an edge, four triangles are created, and four edges are tested. In the case of test failure, a pair of triangles is replaced by the flip operation with another pair, producing two more edges to test.

**How do we find the triangle containing the point \(p_r\)?** There are several ways to do this. We can check all triangles that we have in the triangulation if they contain \(p_r\). But this is inefficient. A far more efficient method is the following:

**Basic Idea:** We either flip or split to locate \(p_r\).

We can keep a history of all triangles added and all edge flips in \(D\), a Directed Acyclic Graph (DAG). Each node will represent a triangle present at some phase of construction.

When a new point is added the node for its triangle will point to 3 child nodes representing the subdivided triangle.
When an edge flip occurs the two adjacent triangles will each point to the two new triangles.

We initialize D as a DAG with a single leaf node, which corresponds to initial triangle $p_1$, $p_2$, $p_3$.

Now suppose that at some point we split the $p_i p_j p_k$ triangle of the current triangulation into three (or two) triangles. The corresponding change in D is to add three (or two) new leaves to D, and to make the leaf for $p_i p_j p_k$ into an internal node with outbound pointers to the three (or two) leaves. Similarly, when we replace the two triangles $p_i p_j p_k$ and $p_i p_j p_1$ by triangles $p_i p_k p_i$ and $p_i p_k p_1$ by an edge flip, we create leaves for the two triangles, and the nodes of $p_i p_j p_k$ and $p_i p_j p_1$ get pointers to the two leaves.

To locate the next point $p_r$ to be added to the current triangulation in D, we start at the root. We check the three children of the root to see which triangle $p_r$ lies and we descend to the corresponding child. We then check the children of this node, descend to a child whose triangle contains $p_r$ and so on, until we reach a leaf of D. This leaf corresponds to the triangle on the current triangulation that contains $p_r$.

**Question**

1) How many triangles will be produced in this algorithm assuming $p$ is randomly permuted?
2) How many flips do we need?
3) How many steps to locate $p_r$?

**Analysis**

Each node has at most 3 children.
Each node in the path represents a triangle in D that contains $p_r$.
Therefore, the algorithm takes $O(\# \text{ of triangles in D that contain } p_r)$.
The expected running time of the algorithm is $O(n \log n)$:
- Time to create the expected number of triangles (the for loop) makes it at least $O(n)$.
- Time spent on point location (operations in the loop) are at most $O(\log n)$. 
Inserting a New Point to the Current Triangulation