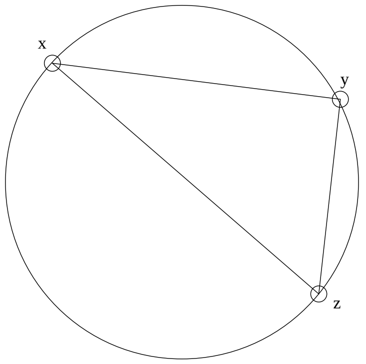
Properties of a Delaunay Triangulation

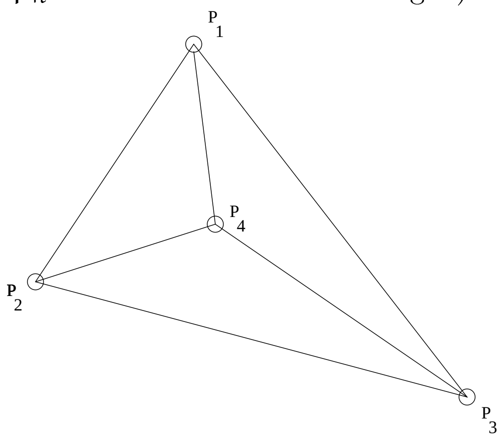
1. Given a set S of n points in $\mathbb{R}^2$ in general position, a triangulation of these points is Delaunay, if and only if the circumcircle of each triangle is empty.



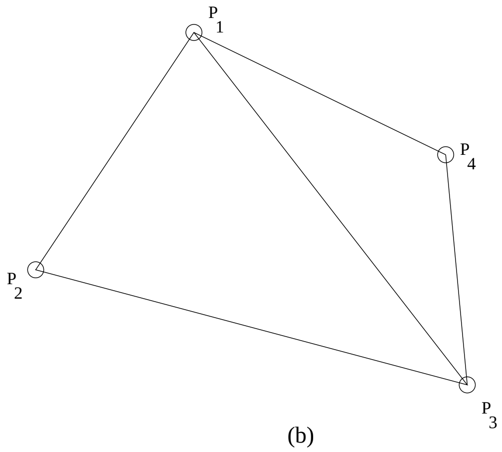
2. Edge xy is in Delaunay(S) if and only if there exists a circle with edge xy on the boundary that is empty.

Incremental Delaunay Algorithm

Insert points one at a time. That is, we insert point p_n in Delaunay($\{p_1, p_2, \ldots, p_{n-1}\}$). We assume that p_n lies inside some triangle, abc, in Delaunay($\{p_1, p_2, \ldots, p_{n-1}\}$). (If not, we can add three points that lie far enough outside of S that they don't change Delaunay(S) and yet p_n now lies inside some triangle.) We then insert p_n and add the edges ap_n, bp_n, cp_n.



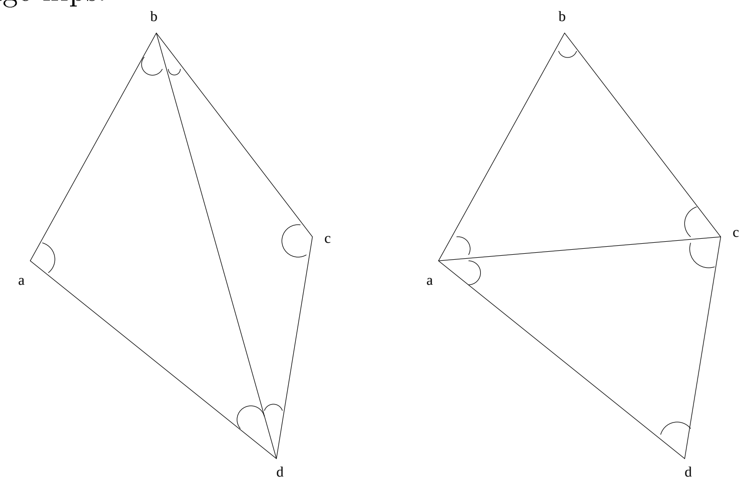
(a)



(b)

We assume case (a) and ignore case (b).

We now have a new triangulation. However, it may not be Delaunay. To fix this, we use edge flips.



Edge e in the triangulation is *locally Delaunay* if both triangles sharing e have empty circumcircles.

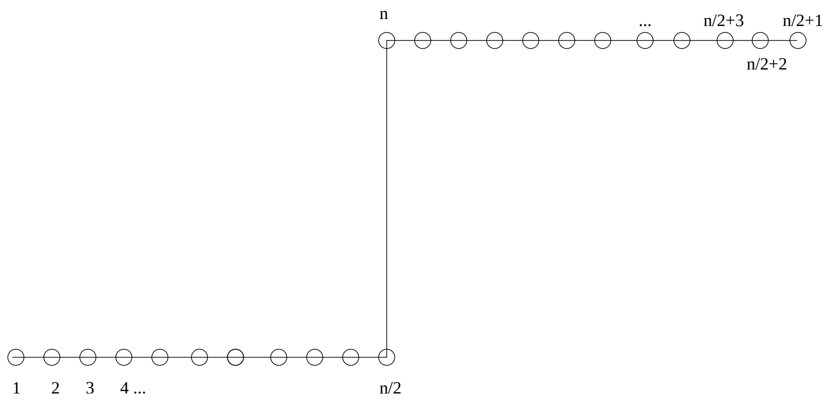
The vector of angles in the triangulation, ordered from smallest to largest is called the *angle vector*.

Lemma 0.1. *Flipping an edge increases the angle vector.*

Theorem 0.2. *The process of flipping an edge that is not locally Delaunay will terminate.*
Proof: There are finitely many triangulations. Each time we flip an edge we move forward in the lexicographic order of the angle vectors of all these triangulations (Thus, the Delaunay triangulation is the one which has its angle vector lexicographically last).

RecursiveDelaunay(S)
1. Recursively construct T_{n-1}
2. Given point p_n, locate Δxyz in T_{n-1}
3. Draw edges $(p_n, x), (p_n, y), (p_n, z)$
4. Repeatedly flip non-Delaunay edges until done

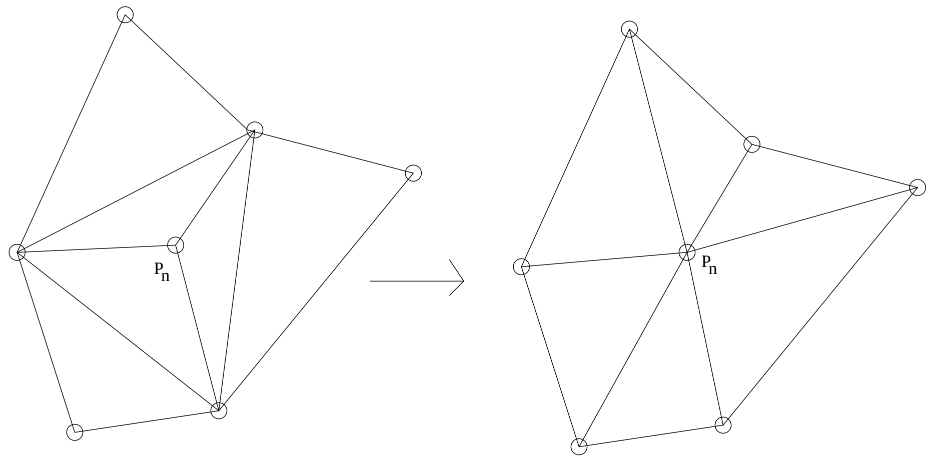
Since steps 2 and 4 run in $O(n)$ time (and step 3 runs in constant time), we get $T_n = T_{n-1} + O(n)$. Thus, we get $T_n = O(n^2)$. Below is an example that requires $\Omega(n^2)$ time.



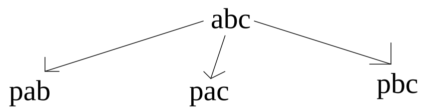
When point $n/2+k$ is inserted, it lies in the circumcircle of at least $n/2-k$ triangles, so the algorithm runs in $\Omega(n^2)$ time. As is often the case, randomization helps. The randomized incremental algorithm is exactly the same as the version just discussed except that the order of the points is randomly permuted, before they are inserted. There is nothing tricky about the algorithm itself, the work comes in analyzing it.

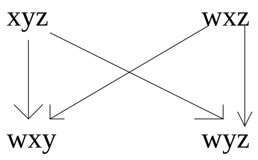
Analysis of Randomized Incremental

To analyze this algorithm, we run it in reverse; that is we imagine deleting point p_n from T_n. The key observation is that the number of flips that we make when we insert p_n is $\deg(p_n) - 3$.



$$
\begin{aligned}
&E[\text{number of flips after inserting } p_n] \\
&= E[\deg(p_n) \text{ in } T_n] - 3 \\
&= \frac{1}{n}\sum_{i=1}^{n} \deg(p_i) - 3 \\
&\leq \frac{1}{n} 6(n-2) - 3 \\
&< 6 \\
&= O(1)
\end{aligned}
$$





For the point location (finding which triangle a point lies in), we use a data structure called a *Directed Acyclic Graph*. Above are two pieces of a DAG. The first comes from a point insertion. The second comes from an edge flip. Every node of the DAG is a triangle, and it stores the history of the triangulation. The root is a large triangle containing all the points. If a triangle is destroyed by edge flips, the children of that triangle are the newly constructed triangles. We locate a point by following the edges from the root. The total number of triangles created is $O(n)$. To analyze the cost of point location, we define the *Scope* of a triangle as the number of points in the interior of its circumcircle. Point location cost is given by:

$$
\begin{aligned}
&O\Big(n + \sum_{k=0}^{n-3} \sum_{\text{triangles with scope } k} k \cdot \mathtt{prob}[\Delta \in \text{Delaunay}(S)] \cdot [\#\text{ of } \Delta \text{ with scope } k]\Big) \\
&= O\Big(n + \sum_{k=0}^{n-3} \sum_{\text{triangles with scope } k} \frac{6k[\text{number of triangles with scope} \leq k]}{(k+1)(k+2)(k+3)}\Big) \\
&= O\Big(n + \sum_{k \geq 0} \frac{[\text{number of triangles with scope} \leq k]}{(k+2)(k+3)(k+4)}\Big) \\
&= O\Big(n + 12n \sum_{k=0}^{n} \frac{1}{k+1}\Big) \\
&= O(n \log n)
\end{aligned}
$$

Thus, the point location requires $O(n \log n)$ time and the number of edge flips is linear, so the algorithm runs in $O(n \log n)$ time.