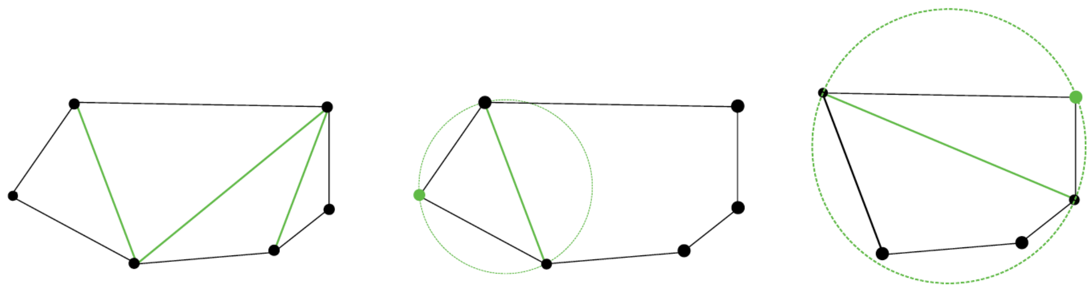
1 Delaunay Triangulation of Convex Polygons

- Assume no 4 circular points, find Delaunay triangulation of a convex polygon.
- Let S = ccw list of n vertices of convex hulls, $DT(S)$ be the Delaunay triangulation of S.

To implement a linear time complexity algorithm, we randomly select one point from S and form a triangle with the selected point and reduce the set S. However, the formed triangle is not always guaranteed to be a Delaunay triangle.



DT(S) good edge bad edge

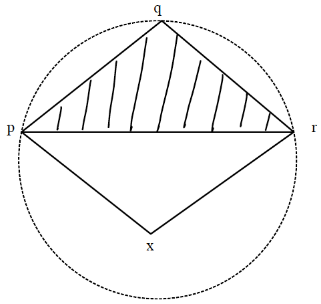
1.1 Algorithm of DT(S)

DT(S)

1. if $|S| = 3$, return $\triangle$ with vertices of S
2. pick q from S, let p, r be its neighbours.
3. T=DT($S\backslash\{q\}$)+$\triangle pqr$
4. Flip(T, q, rp)

Flip(T, q, rp)

1. if $\overline{rp}$ is bad i.e. not a Delaunay edge of T (which is equivalent to $x \in \bigcirc pqr$ (see Figure below))
   - remove $\bar{rp}$ from T
   - add $\bar{qx}$ to T
   - Flip(T, q, rx)
   - Flip(T, q, xp)
2. if $\bar{rp}$ is good, do nothing and return



1.2 Time Complexity of DT(S)

To compute the run time of the randomized algorithm DT(S), use backward analysis.

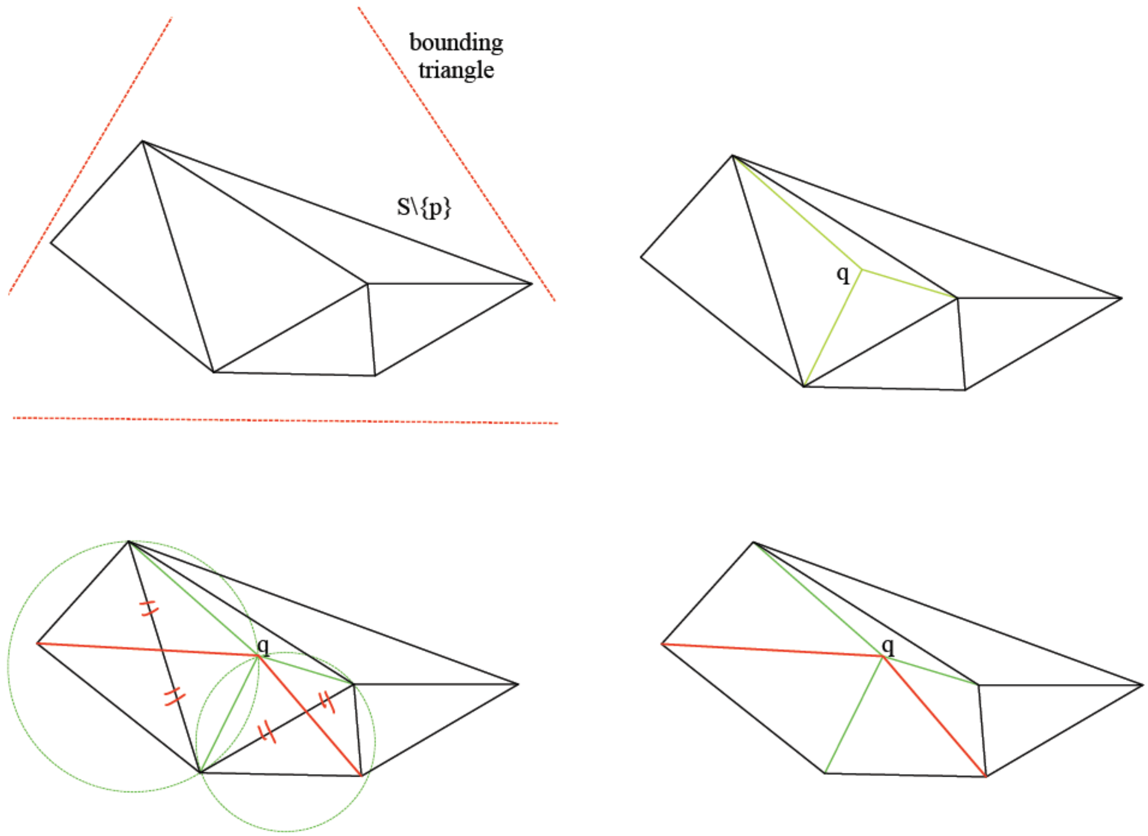
- For Flip(T, q, rp), the number of runs is $deg(q)$ in DT(S)
  - average degree of a vertex q in DT(S), where $|S| = n$ is $\frac{\sum deg(q)}{n} = \frac{2(\# \text{ of edges in DT}(S))}{n}$
  - deg(q) $= \frac{2(2n-3)}{n} = 4 - \frac{6}{n}$, Flip(T,q,rp) is takes expected constant time.
- Each recursive call takes expected $O(1)$ time in addition to the time for one more call on a smaller problem. Thus the total runtime is expected $O(n)$.
- DT(S) is $O(n)$

2 Incremental Delaunay Triangulation of point set S

2.1 Algorithm for General Point Sets

With the same algorithm for convex polygons, in each iteration,

- add a point $p \in S$ randomly
- add edges from p to three vertices of the triangle that p falls inside of.
- flip the edges if the added edges are bad



2.2 Time Complexity of General Point Sets

For the rest of the algorithm, we proved for linear complexity. We only need to figure out: how to know which triangle does the selected point q falls inside?

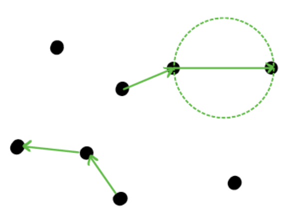
- option1: maintain search structure for $DT(S \backslash \{q\})$
- option2: re-bucketing remaining points to be added into newly created triangles.

For option 2, in ith iteration, what is the probability that a point x is re-bucketed when $|S| = i$?

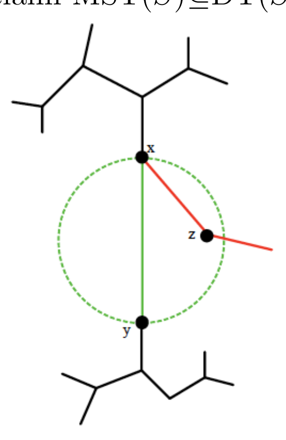
- x is re-bucketed when the triangle containing x in DT(S) is created by adding q, thus the probability is $\frac{3}{i}$
- $\mathbb{E}[\#\text{re-buckets of } x] \leq \sum_{i=1}^{n} \frac{3}{i} = O(\log n)$
- in total, for n points, option 2 have complexity of $O(n \log n)$

3 Relatives of Delaunay Triangulation

1. Nearest neighbour graph of S: NN(S)
   - draw edge $x \to y$ if y is closest to x, $x, y \in S$
   - claim: NN(S)⊆DT(S)

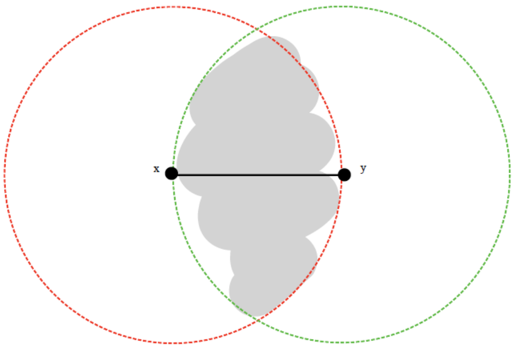
   - proof: if there is a point z other than x, y in circle with diameter $\bar{xy}$, x, y are not the nearest neighbour of each other.
2. Euclidean minimum spanning tree of S: MST(S)
   - claim MST(S)⊆DT(S)

- proof: let z be a point inside circle with diameter $\bar{xy}$, z must be in one of the connected components with root x or y if disconnect $\bar{xy}$. Suppose z is in the connected component of y, connect x, z will generate a new spanning tree but smaller.

3. Relative neighbourhood graph: RNG(S)

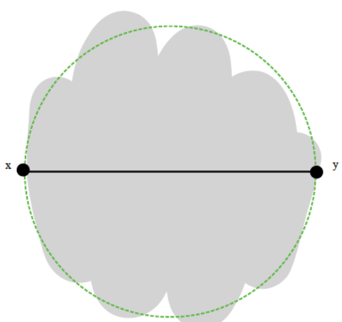
   - connect x, y if the intersection area of circles centred at x, y and radius of $|xy|$ is empty.



   - claim RNG(S)$\subseteq$DT(S)

4. Gabriel graph: GG(S)

   - connect x, y iff circle of diameter $\bar{xy}$ is empty.



   - claim GG(S)$\subseteq$DT(S)

5. NN(S)$\subseteq$MST(S)$\subseteq$RNG(S)$\subseteq$GG(S)$\subseteq$DT(S)