6. Top-Down Sampling

Top-Down Sampling is a divide-and-conquer method of building search structures based on random sampling.

A *randomized binary tree* is the simplest search structure based on random sampling

1. Choose a random point p from the given set N of points
2. p divides N into two subsets, N_1 and N_2, of roughly equal size
3. Label the root of the search tree with p
4. The children of this root are the recursively built trees for N_1 and N_2.

General geometric search problem: Given a set N of objects in R^d, construct the induced complex (partition) $H(N)$ and a geometric search structure $\tilde{H}(N)$ that can be used to answer the queries over $H(N)$ quickly.

- a point location query in a planar subdivision

Assumption
The complex $H(N)$ satisfies the bounded degree property.

- Every face of $H(N)$, at least of the dimension that matters, is defined by a bounded number of objects in N
- This assumption is needed to make the randome sampling technique
- If partition does not satisfy the assumption, a suitable refinement is needed
  - Vertical trapezoidal decomposition for the arrangement.

General Process

1. Choose a random subset $R \subset N$ of a large enough constant r
2. Build $H(R)$ and a search structure for $H(R)$
   - Since the size of R is a constant , the search structure is typically trivial.
3. Build conflicts of all faces of $H(R)$ of relevant dimensions
   - The notion of a conflict depends on the problem under consideration.
4. For each such face $\triangle \in H(R)$, recursively build a search structure for $N(\triangle)$, which is the set of objects in N in conflict with $\triangle$.
5. Build an ascent structure, denoted by $\text{ascent}(N, R)$.
   - It is used in queries described latter.

The queries are answered as bellow

- The original query is over the set N
- We answer the query over the smaller set R using the trivial search structure associated with $H(R)$
- If $\triangle \in H(R)$ is the answer to this smaller query, we recurisvely answer the query over the set $N(\triangle)$ of conflicting objects
- After reaching the bottommost face, using the ascent structure $\text{ascent}(N, R)$, we determine the answer over the set N

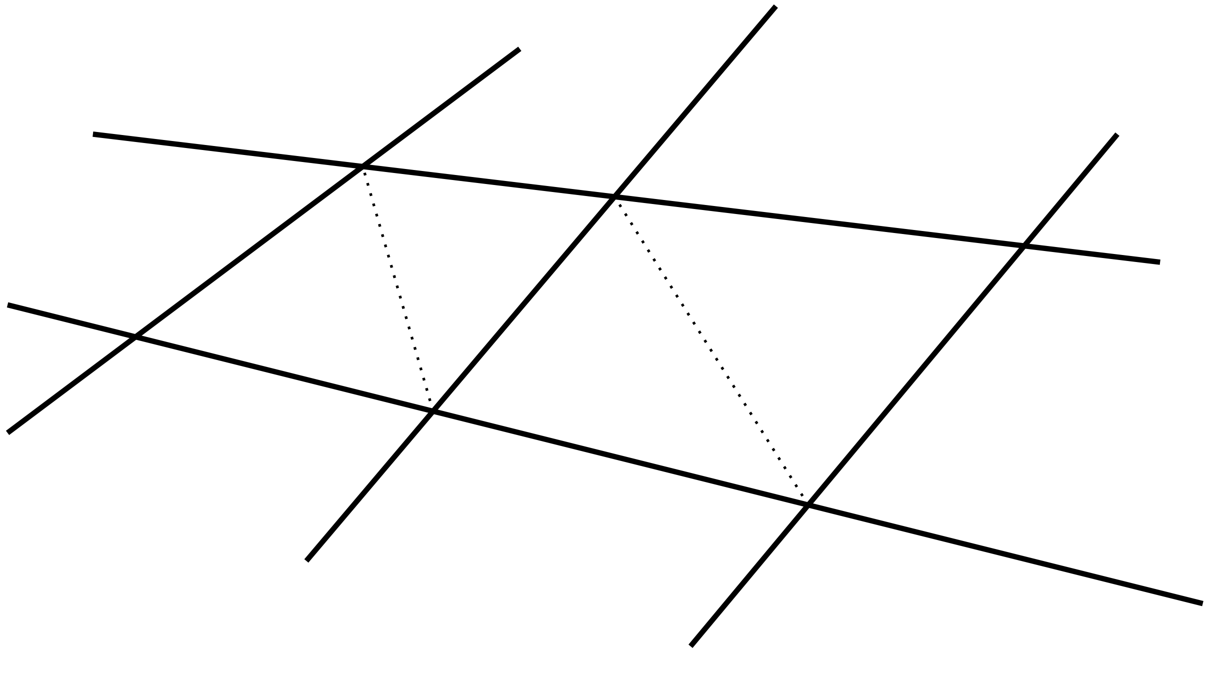
Arrangment of lines

- N is a set of n given lines in the plane
- $G(N)$ is the arrangement formed by N
- Γ is a fixed triangle in the plane
  - At the root level, the vertices of Γ is assumed to be at infinity, i.e., $\Gamma = \mathbb{R}^2$
- For any given query point q in Γ, answer the face in the intersection $\Gamma \cap G(N)$ containing q

Canonical Triangulation

$H(N)$ is the canonical for $G(N) \cap \Gamma$

- For a (possibly unbounded) convex polygon C, the canonical triangulation of C is a refinement for C by linking its bottom vertex to its other vertices (break ties arbitrarily)
- The canonical triangulation for $G(N) \cap \Gamma$ is a refinement for it by applying the canonical triangulation to each of its faces.
- $H(N)$ can be constructed in $O(n^2)$ time and space
- $H(N)$ has the bounded degree



Top-Dowm Sampling for the search structure

1. Let Γ be the root and compute $G(N) \cap \Gamma$
2. Select a random sample R of N of size r, where r is a large enough constant.
3. Construction $H(R)$
4. For each triangle $\triangle \in H(R)$, compute $N(\triangle)$, where $N(\triangle)$ denotes its conflict list, i.e., the set of lines in $N \setminus R$ intersecting $\triangle$.
5. If one triangle of $H(R)$ has a conflict size large than $b(n/r) \log r$, for an appropriate constant b, repeat step 2–4.
6. For each triangle $\triangle \in H(R)$, recur the computation on $G(N(\triangle) \cap \triangle$
7. For each $\triangle \in H(R)$, associate with every face of $G(N(\triangle)) \cap triangle$ a parent pointer to the face containing it in $G(N) \cap \Gamma$.

The construction time without recursive call

1. $O(n^2)$ time to construct $G(N)$
2. $O(n)$ to pick a random sample because r is a constant
3. $O(1)$ to construct $H(R)$ because r is a contant
4. $O(n)$ to compute $N(\triangle)$ for all triangle in $H(R)$ because $H(R)$ has $O(1)$ triangle
5. The expected number of repetition is $O(1)$, so step2–5 take $O(n)$ expected time
   - With probability at least $1/2$, the conflict size of each triangle in $H(R)$ is less than $b(n/r) \log r$
   - If the probability of success in each trial is at least $1/2$, the expected number of required trails is $O(1)$
6. $O(r^2)$ recursive calls and the size of each call is at most $O(b(n/r) \log r)$
7. $O(n^2)$ to make parent pointers (Could be an Exercise)

Point Location using the search structure

For a query point p in Γ, locate the face in $G(N) \cap \Gamma)$ that contains p

1. Locate the triangle $\triangle$ in $H(R)$ containing p
   - $O(1)$ time because $H(R)$ has $O(1)$ triangles
2. Recursively locate the face of $G(N(\triangle))\triangle$ containing p
3. Use the parent pointer associated with the recursively found face to tell the face of $G(N) \cap \Gamma)$ containing p

The **query time** is $\boldsymbol{O(\log n)}$

- Let $q(n)$ be the query time of locating a point in an arrangement formed by n lines.
- If n is less than a threshold, $q(n) = 1$
- Otherwise,

$$q(n) = O(1) + q(b\frac{n}{r}\log r)$$

- If r is sufficiently large constant, the statement follows.

The **expected construction time** is $\boldsymbol{O(n^{2+\epsilon})}$

- Let $t(n)$ be the expected time to construct the search structure for an arrangement formed by n lines
- If n is less than a threshold, $t(n) = 1$
- otherwise,

$$t(n) = O(n^2) + \sum_{\triangle \in H(R)} t(|N(\triangle)|) = O(n^2) + O(r^2) \cdot t(b\frac{n}{r}\log r).$$

- The depth of recursion is $O(\log_r n)$
- $t(n) = n^2 c^{\log_r n}$, where c is a constant that is sufficiently larger than b and the constant within the Big-Oh bound
- For any real number $\epsilon > 0$, we can choose r large enough such that, $t(n) = O(n^{2+\epsilon})$.

(The last two derivations will be an exercise)

The **size** of the search structure is $\boldsymbol{O(n^{2+\epsilon})}$

- It follows from the same derivation as the construction time but the complexity is deterministic.

Theorem

For every arragement of n lines in the plane and for any real number $\epsilon > 0$, one can construct a point location structure of $O(n^{2+\epsilon})$ size, guaranteeing $O(\log n)$ query time, in $O(n^{2+\epsilon})$ expected time