6. Bottom-Up Sampling

Bottom-Up Sampling is a gradational method of building search structures based on random sampling.

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.

For a set N of n objects, a ***gradation*** of N
is a sequence of sets, $N_1, N_2, \ldots, N_r$, such that

- $N = N_1 \supseteq N_2 \subseteq \cdots \supseteq N_{r-} \supseteq N_r = \emptyset$
- N_{i+1} is obtained from N_i by flipping a fair coin independently for each object in N_i and retaining only those objects for which the toss was a head.

Complexity with High Probability

$$f(n) = \tilde{O}(g(n)),$$

if for some positive constant c, $f(n) < cg(n)$, with probability $1 - 1/p(n)$, where $p(n)$ is a polynomial whose degree depends on c.

- The degree of $p(n)$ can be very high by choosing the constant c large enough.
- With probability $1/p(n)$, $f(n) > cg(n)$

Lemma

The expected value of $r = O(\log n)$

- $N = \{S_1, S_2, \ldots, S_n\}$
- For $1 \leq j \leq n$, let X_j be the random variable that $X_j = i$ if S_j belongs to N_i not to N_{i+1}.
- X_j is a geometric distribution that $\Pr(X_j = i) = (1/2)^i$
- X_1, X_2, ..., and X_n are independent and identical.
- Let T_n be a random variable representing $\max_{1 \leq j \leq n} X_j$.
- $E[r] = E[T_n]$
- Then

$$E[T_n] = \sum_{i \geq 1} \Pr(T_n \geq i) = \sum_{i \geq 1} 1 - \Pr(T_n < i)$$

$$= \sum_{i \geq 1} 1 - (1 - \frac{1}{2^i})^n = O(\int_1^\infty 1 - (1 - \frac{1}{2^x})^n dx) = O(\log n)$$

Lemma

$$r = \tilde{O}(\log n)$$

- For each object in N, the probability that this object belongs to N_{k+1} is $(1/2)^k$
- The probability that $r > k$ is bouneded by

$$\frac{n}{2^k}.$$

  - The probability of union of events is bounded by the sum of probabilities of those event
- By choose $k = c \log_2 n$, the probablity that $r > c \log_2 n$ is bounded by

$$\frac{n}{n^c} = n^{c-1}.$$

- With probability at least $1 - 1/n^{c-1}$,

$$r \leq c \log n.$$

Lemma

$$E[\sum_{i=1}^{r} |N_i|] = O(n).$$

- It is equivalent to the expected sum of the maximum level of all objects
  - If the maximum level of an object is i, it contributes i to the quantity
- The probability that the maximum level of an object is at least i is $1/2^{i-1}$
- The expected maximum level of an object is

$$\sum_{i \geq 1} \frac{1}{2^{i-1}} = O(1).$$

- Due to linearity of expectation, we have

$$E[\sum_{i=1}^{r} |N_i|] = \sum_{1 \leq j \leq n} O(1) = O(n).$$

General Idea for Bottom-Up Sampling

The search structure sample(N) is constructed as follow

1. Generate a gradation, $(N_1, N_2, \ldots, N_n)$, of N
2. Build $H(N_i)$ for $1 \leq i \leq r$
3. For $2 \leq i \leq r$, associate each face of $H(N_i)$ with the conflict list of objects in $N_{i-1} \setminus N_i$ that conflicting it.
4. Between two successive levels i and $i+1$, compute a descent structure descent$(i+1, i)$ which will be used in answering queries.

Queries for Bottom-Up Sampling

Given a query point q, answer which configuration of $H(N)$ contains q

- From $i = r$ to $i = 1$, locate the configuration of $H(N_i)$ contains q
- From $i = r$, it is trivial
- When descending from N_i to N_{i-1}, use descent(i, i-1) to find out the configuration of $H(N_{i-1})$ contains q

Skip List

For a set N of n points on the real line, the skip list sample(N) is a search structure based on Bottom-Up Random Sample to answer which interval in $H(N)$ contains a fiven query point q efficiently.

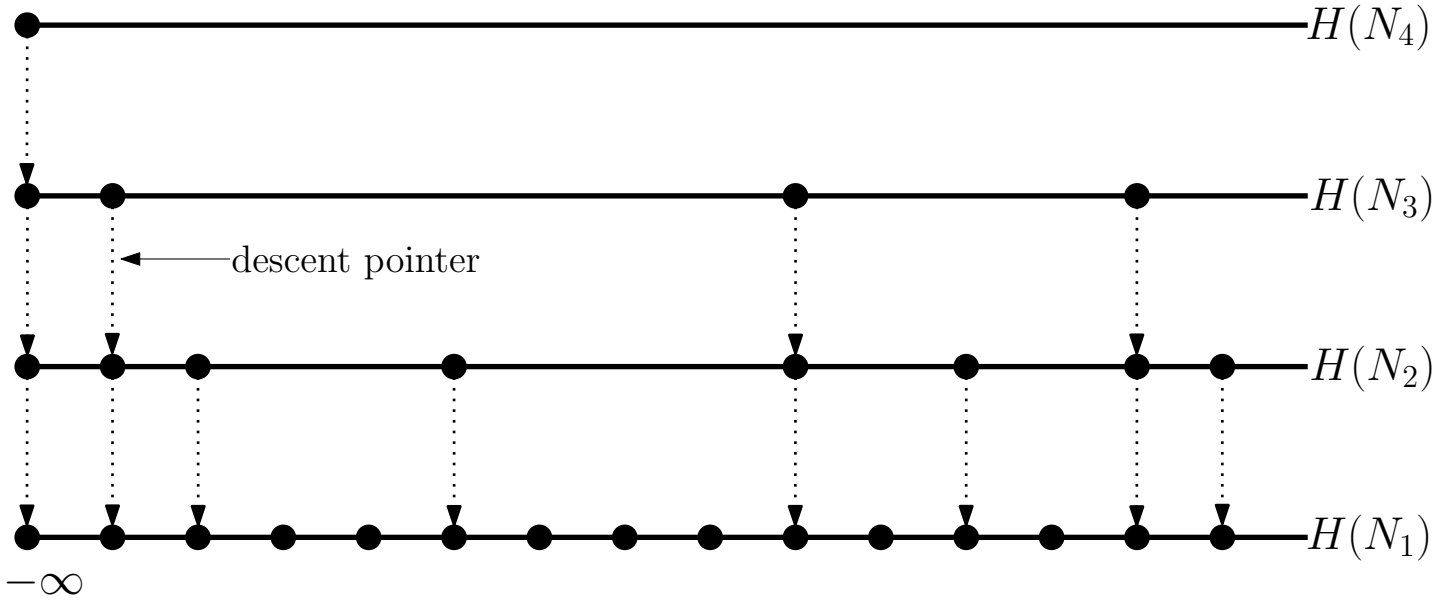
- The skip list is the simpliest search structure based on Bottom-Up Random Sampling
- There is always a dummy node associated with $-\infty$
- For an interval I of $H(N_i)$, the intervals in $H(N_{i-1})$ which are contained in I are called the *children* of I.

Construction of Skip List

- For $1 \leq i \leq r$, compute $H(N_i)$ by sorting points in N_i
- For $2 \leq i \leq r$, each point N_i is linked to its counterpart in N_{i-1} by a descent pointer.
- The construction time is

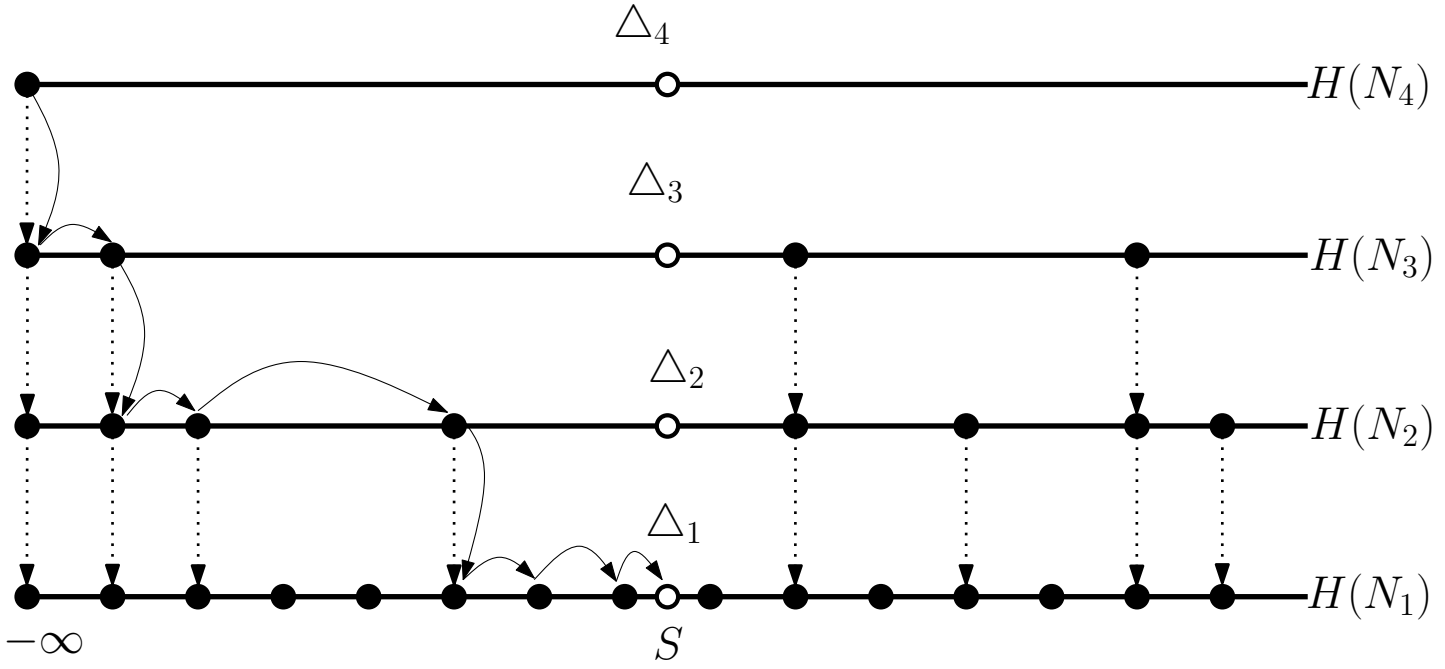
$$\sum_{i=1}^{r} O(|N_i| \log |N_i|) = O(\log n) \sum_{i=1}^{r} O(|N_i|) = O(n \log n).$$

- The bound $\tilde{O}(n \log n)$ can be obtained similarly.



A point location query for q

1. Let the single interval of $H(N_r)$ be $\triangle_r$. $\triangle_r$ clearly contains p
2. From $i = r$ to $i = 2$,
   (a) Let $\triangle_i$ be the interval in $H(N_i)$ that contains q
   (b) Use the descent pointer assocaited with the left endpoint of $\triangle$ to search through all children of $\triangle_i$. Actually, we search from left to right and not all children are visited.



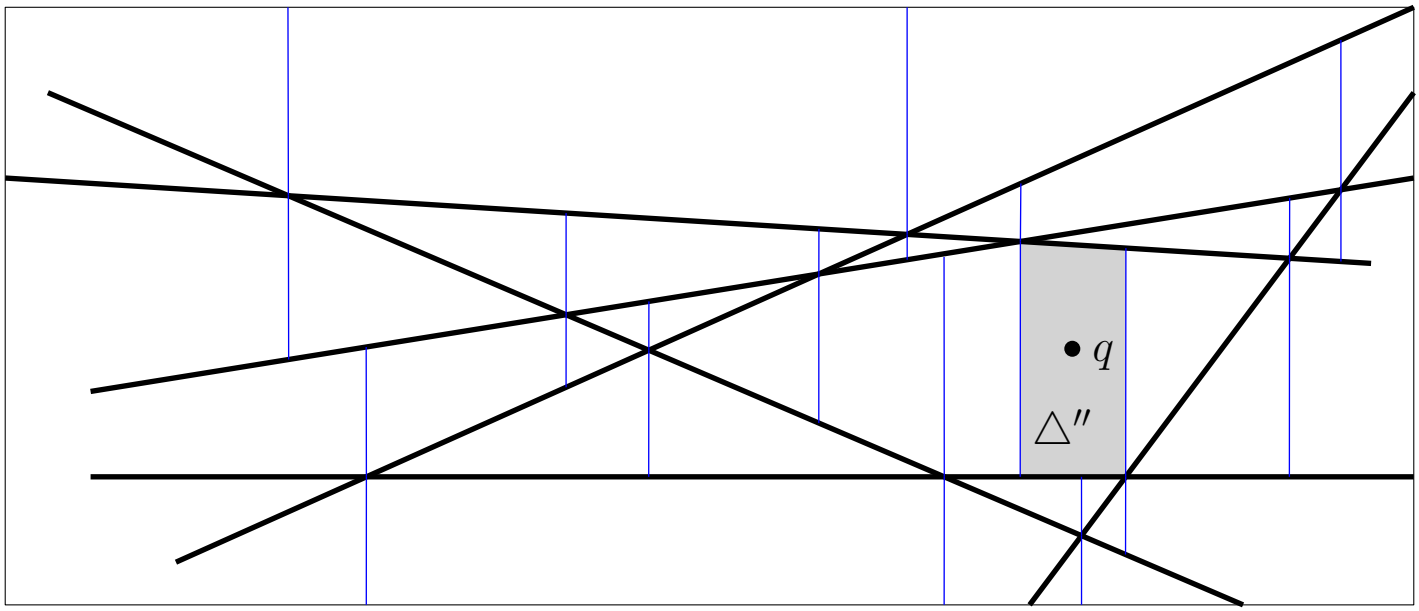
Query time is $O(\log n)$ and $\tilde{O}(\log n)$

- let $l(\triangle_i)$ be the number of children of $\triangle_i$
  - Let $l_0(\triangle_i)$ and $l_1(\triangle_i)$ be the number of points in $N_{i-1} \setminus N_i$ that are contained in $\triangle_i$ left and right to q, respectively.
  - $l(\triangle_i) = l_0(\triangle_i) + l_1(\triangle_i)$
- When N_{i-1} is fixed, $l_0(\triangle_i)$ is distributed according to the geometric distribution with probability $1/2$
  - because $l_0(\triangle_i) = k$ if and only if for exactly the k nearest points in N_{i-1} to the left of q, the tosses are all failures.
- $E[l_0(\triangle_i)] = O(1)$, so does $E[l_1(\triangle_i)] = O(1)$
- $E[\sum_{i=2}^{r} l(\triangle_i)]$ is $O(\log n)$ and $\tilde{O}(\log n)$ because r is $O(\log n)$ and $\tilde{O}(\log n)$.

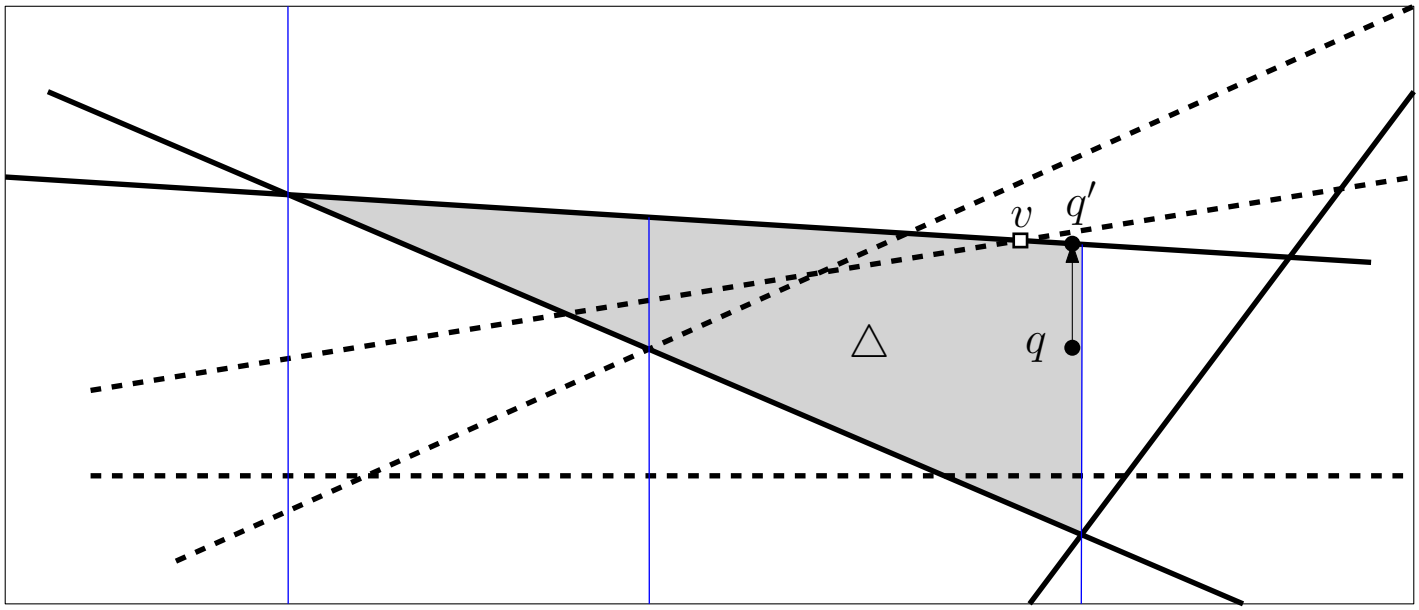
Point Location Search Structure in Arrangements

Given a set N of n lines in the plane, the point location search structure Sample(N) for the arragnement $G(N)$ of N is constructed as follows:

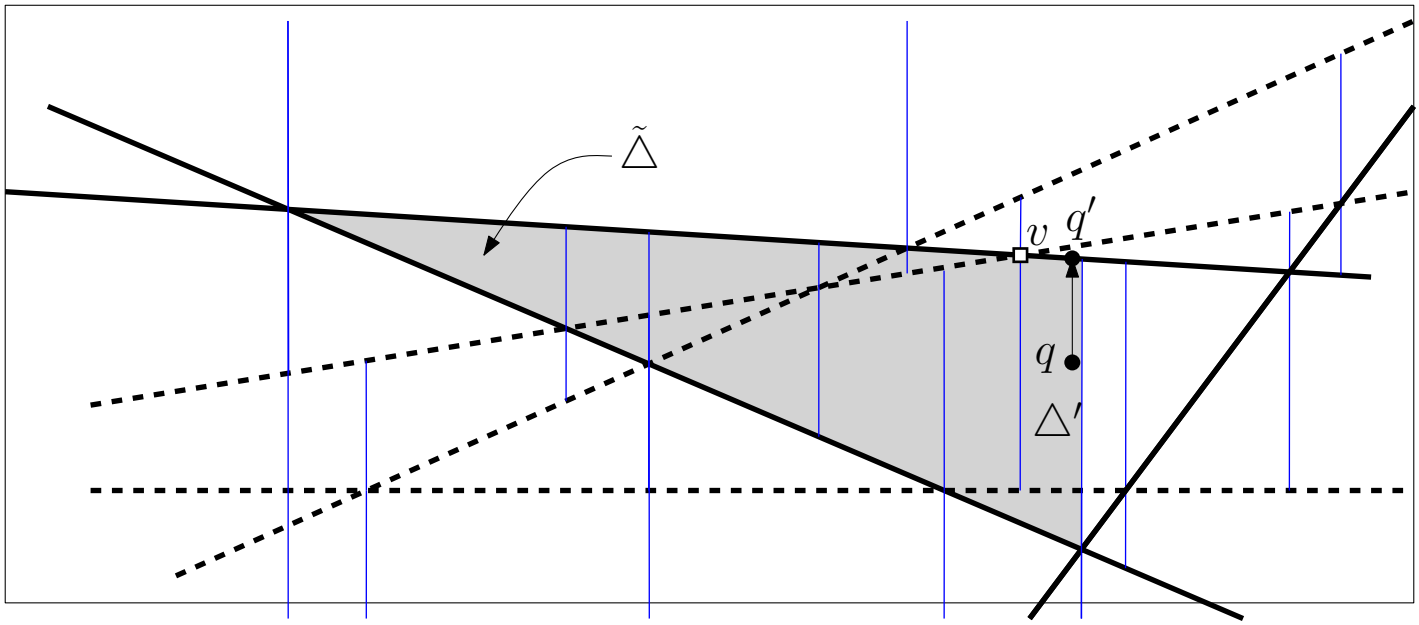
- Generate a gradation of N, $N_1, N_2, \ldots, N_r$
- For $1 \leq i \leq r$, compute the arrangment $G(N_i)$ and the trapezoidal decomposition $H(N_i)$
- For each trapezoid $\triangle$ of $H(N_i)$, store a conflict list $L(\triangle)$ of the line in $N_{i-1} \setminus N_i$ interseting $\triangle$
- Compute the descent structure descent$(i+1, i)$ as the partition $H(N_{i+1}) \oplus H(N_i)$ by superimpossing $H(N_{i+1})$ and $H(N_i)$ on each other
- Associate with each trapezoid in descent$(i+1, i)$ a pointer to the unique trapezoid $H(N_i)$ that contains it.



$H(N_l)$



$H(N_{l+1})$



$\mathrm{descent}(l+1, l) = H(N_{l+1}) \oplus H(N_l)$

Locate a point q in $H(N)$ using the search structure

1. It is trivial to locate q in $H(N_r)$ because it contains only one trapezoid, i.e., the entire plane.
2. From $i = r - 1$ to $i = 1$, assume that we have located q in $H(N_{i+1})$, and use descent$(i+1, i)$ to locate q in $H(N_i)$ as follows
   (a) Let $\triangle$ be the trapezoid in $H(N_{i+1})$ that contains q.
   (b) Let $\tilde{\triangle}$ denote the restriction of the superimposed partition descent$(i+1, i)$ within $\triangle$.
   (c) Project a vertical ray from q to hit a line $Q \in N$ and let q' be the hitting point. It is clear than Q either from the boundary of $\triangle$ or belongs to $L(\triangle)$.
   (d) Let v be the intersection of Q with a line in $L(\triangle)$ or the boundary of $\triangle$, which is nearest to q' on its left side.
   (e) It is easy to see that computing q' and v' take $O(|L(\triangle)| + 1)$ time
   (f) By walking from v to q', we can use $\tilde{\triangle}$ to find out the trapezod in $H(N_i)$ that contains q.

Intuitive Analysis for Query Time

- Since $E[|N_{i+1}|] = \frac{1}{2}E[|N_i|]$, at high probability, for each interval $\triangle \in H(N_{i+1})$, $L(\triangle)$ is $O(\log n)$.
- Since there are $\tilde{O}(\log n)$ levels, the total search time is $\tilde{O}(\log n^2)$.

Advanced Analysis for Query Time (A Sktech)

- For all i, $|L(\triangle_i)| = \mathrm{NB}(4)$, where $\mathrm{NB}(s)$ denotes the random variable that is equal to the number of tails obtained before obtaining s heads.
  - (Rough thought) A trapezoid is defined by at most 4 lines, and each line corresponds to one direction (up, down, right, left).
  - In up direction, it is equivalent to sequentialy draw a coin for lines above q accordering to vertical distance to q, i.e., equal to $\mathrm{NB}(1)$.
  - $\mathrm{NB}(1) = \tilde{O}(1)$ and $\mathrm{NB}(4) = 4\mathrm{NB}(1) = \tilde{O}(1)$
- $\tilde{O}(\log n) \times \tilde{O}(1) = \tilde{O}(\log n)$

Construction Time

- For $1 \leq i \leq r$, $H(N_l)$ can be constructed in $O(n_i^2)$ time, where n_i is the number of lines in N_i.
- Constructing conflict lists takes $O(n_i * n_{i-1})$ time
  - For a fixed line $Q \in N_{i-1}$, the number of trapezoids in $H(N_i)$ that are intersected by Q is $O(n_i)$. (Zone theorem)
  - Let S be any fixed line in N_i
  - Locate $Q \cap S$ in $H(N_i)$ by searching along S.
  - Search along Q from $Q \cap S$ in both directly will find all trapezoids in $H(N_i)$ that are conflicted by Q
  - It takes $O(n_i)$ time for Q, and thus $O(n_i * n_{i-1})$ time for total
- Building descent$(i, i-1)$ takes $O(n_i * n_{i-1})$ time
  - It is equivalent to computing $H(N_i) \oplus H(N_{i-1})$.
  - $H(N_i) \oplus H(N_{i-1})$ can be refined trivially from $H(N_i) \oplus G(N_{i-1})$
  - $H(N_i) \oplus G(N_{i-1})$ can be obtained by "drawing" lines in $N_{i-1} \setminus N_i$ on the top of $H(N_i)$
  - We add to $H(N_i)$ the lines in $N_i \setminus N_{i-1}$, one at a time, in any oder.
  - By the conflict information, adding a line takes $O(n_i)$ time
- The total time is $\sum_{i=1}^{r} O(n_i * n_i)$.
- Since $\sum_{i=1}^{r} O(n_i) = \tilde{O}(n)$, we have

$$\sum_{i=1}^{r} O(n_i * n_i) = O((\sum_{l=1}^{r} O(n_i))^2) = \tilde{O}(n^2).$$

Summary

Given a set N of n lines, a search structure for the arrangment formed by N can be constructed in $\tilde{O}(n^2)$ time and space such that the point location query can be conducted in $\tilde{O}(\log n)$ time.