COL 758: Advanced Algorithms

Lecture 14

Randomized Incremental Construction

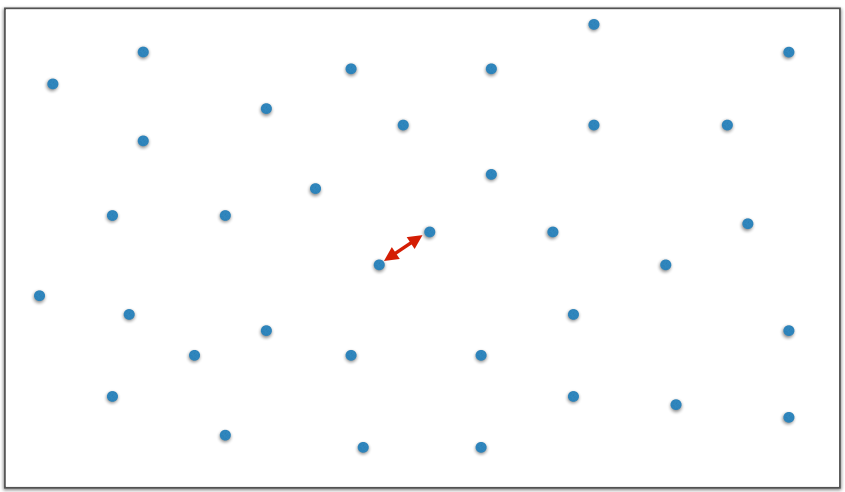
Minimum pairwise distance

Closest Pair of Points (or Minimum pairwise distance)

Given: A set S of n points in x-y plane.

Output: A pair of points in S at minimum distance, or $\min_{a \neq b \in S} distance(a, b)$.

Example:



Deterministic algorithm (COL 351):

- $O(n \log n)$: Divide and Conquer algorithm

Randomized algorithm:

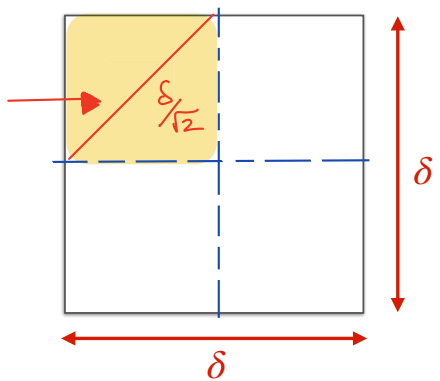
- $O(n)$: Randomized Incremental Construction

Subproblem

Given: A set S of points in x-y plane satisfying $\min_{\substack{(a,b) \in S \times S \\ a \neq b}} distance(a,b) \geq \delta$

Question: What is the maximum number of points that can fit in a square of size $\delta \times \delta$? Ans : (≤ 4)

At most one point can lie in this square



Randomized Algorithm: Main Idea

Notations:

- S : Input set of n points
- $(p_1, \ldots, p_n)$: Random permutation of points in S
- S_i : Set of first i points, i.e. $(p_1, \ldots, p_i)$
- δ_i : Minimum distance between points in set S_i

Question: What is probability that $\delta_i \neq \delta_{i-1}$? Answer : $\leq 2/i$

Solution : Let a, b be a pair of points at least distance in set S_i.

$S_i \not\subseteq S_{i-1}$, that is, min-dist$(S_i) \not\leq$ min-dist(S_{i-1})

only if point $p_i = a$ or $p_i = b$.

Now, Prob$(p_i = a$ or $p_i = b) = 2/|S_i| = 2/i$, so answer is $\leq 2/i$

Randomized Algorithm

Given: A set S of n points in x-y plane.

Output: A pair of points in S at minimum distance, or $\min_{a \neq b \in S} distance(a, b)$.

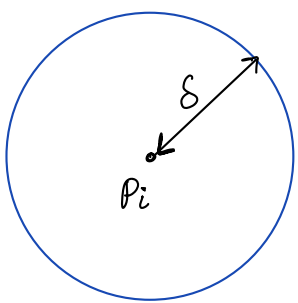
Algorithm

1. Let $P = (p_1, \ldots, p_n)$ be a random permutation of the n points.
2. Initialize $\delta = distance(p_1, p_2)$.
3. For $(i = 3$ to $n)$:

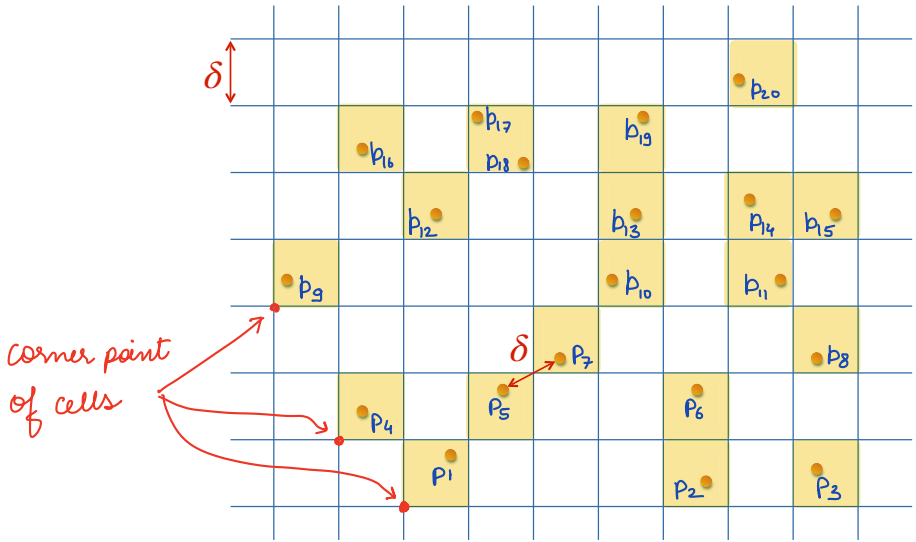
   Add point p_i, and update the minimum-distance δ.

Question: How to quickly detect if there is a change in δ on insertion of point "p_i" ?

Observation: If $\delta = \text{min-dist}(S_{i-1})$, i.e. δ is minimum distance encountered till now, then on addition of p_i the min-distance decreases iff $S_{i-1} \cap \text{Ball}(p_i, \text{radius} = \delta)$ is not $\emptyset$.



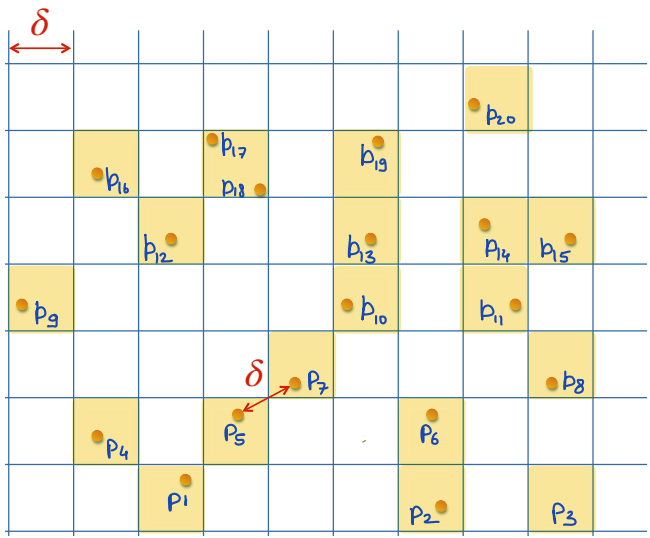
Detecting if there is a change in δ on insertion of new point :



Note : We can store in Balanced BST structure corner points of all the cells containing one or more points.

Make a grid of size δ containing "all points considered till now"

Grid Data Structure (using Balanced BST)



O(|Si| log n)

- Build-Grid(S_i, δ):
  Build grid for S_i with parameter δ.
- Find-Cell(p): (search coordinate $\frac{x}{\delta}$ $\frac{y}{\delta}$)
  Find the cell in which p belongs.
- Report-Points-in-Cell(c): (At most 4 pointers)
  Report all points in cell c.
- Insert-Point(p):
  Insert point p.

O(log n)

Grid Data Structure

	Height Balanced Binary search tree	Dynamic Hashing
Find-Cell(p)	$O(\log n)$	$O(1)$
Report-Points-in-Cell(c)	$O(\log n)$	$O(1)$
Insert-Point(p)	$O(\log n)$	$O(1)$ expected
Build-Grid(S_i, δ)	$O(\lvert S_i \rvert) * \log n$	$O(\lvert S_i \rvert)$ expected

Randomized Algorithm

Algorithm

1. Let $P = (p_1, \ldots, p_n)$ be a random permutation of the n points.
2. Initialize $\delta = distance(p_1, p_2)$.
3. Build-Grid($\{p_1, p_2\}, \delta$)
4. For ($i = 3$ to n):

   /* Add point p_i and update the minimum-distance δ.*/

   (i) Let c = Find-Cell(p_i). — If $p_i = (x_i\ y_i)$ then search $(\frac{x_i}{\delta}, \frac{y_i}{\delta})$

   (ii) Compare distance of p_i from all point in c, and its neighbouring cells.

   (we use function Report-Points-in-Cell).

   At most 9 non-empty cells, so at most 36 points

   (iii) **If** (δ is not updated) **then** Insert-Point(p_i).

   Else Build-Grid($\{p_1, \ldots, p_i\}, \delta$).

Randomized Algorithm (using dynamic hashing data-structure)

Algorithm

1. Let $P = (p_1, \ldots, p_n)$ be a random permutation of the n points.
2. Initialize $\delta = distance(p_1, p_2)$.
3. Build-Grid($\{p_1, p_2\}, \delta$)
4. For ($i = 3$ to n):

   /* Add point p_i and update the minimum-distance δ.*/

   (i) Let c = Find-Cell(p_i). — $O(1)$

   (ii) Compare distance of p_i from all point in c, and its neighbouring cells. — $O(1)$

   (we use function Report-Points-in-Cell).

   (iii) **If** (δ is not updated) **then** Insert-Point(p_i). — $O(1)$

   Else Build-Grid($\{p_1, \ldots, p_i\}, \delta$). — $O(i)$

Expected Time Complexity of i^{th} iteration

Expected time complexity of Round i $= O(1) + O(i) * Prob(\delta_i < \delta_{i-1})$

$\leq O(1) + O(i) * \frac{2}{i} = O(1)$

Thus, expected time complexity of algorithm is $O(n)$.