Computational Geometry

Lecture #4 — Closest Pairs II

Peter Kramer | November 12th, 2024

Recap:

Master Theorem

“Median of Medians” Algorithm

Closest pair — Bounds

Closest Pair — Divide and conquer

Master Theorem

Analysis of recursive algorithms

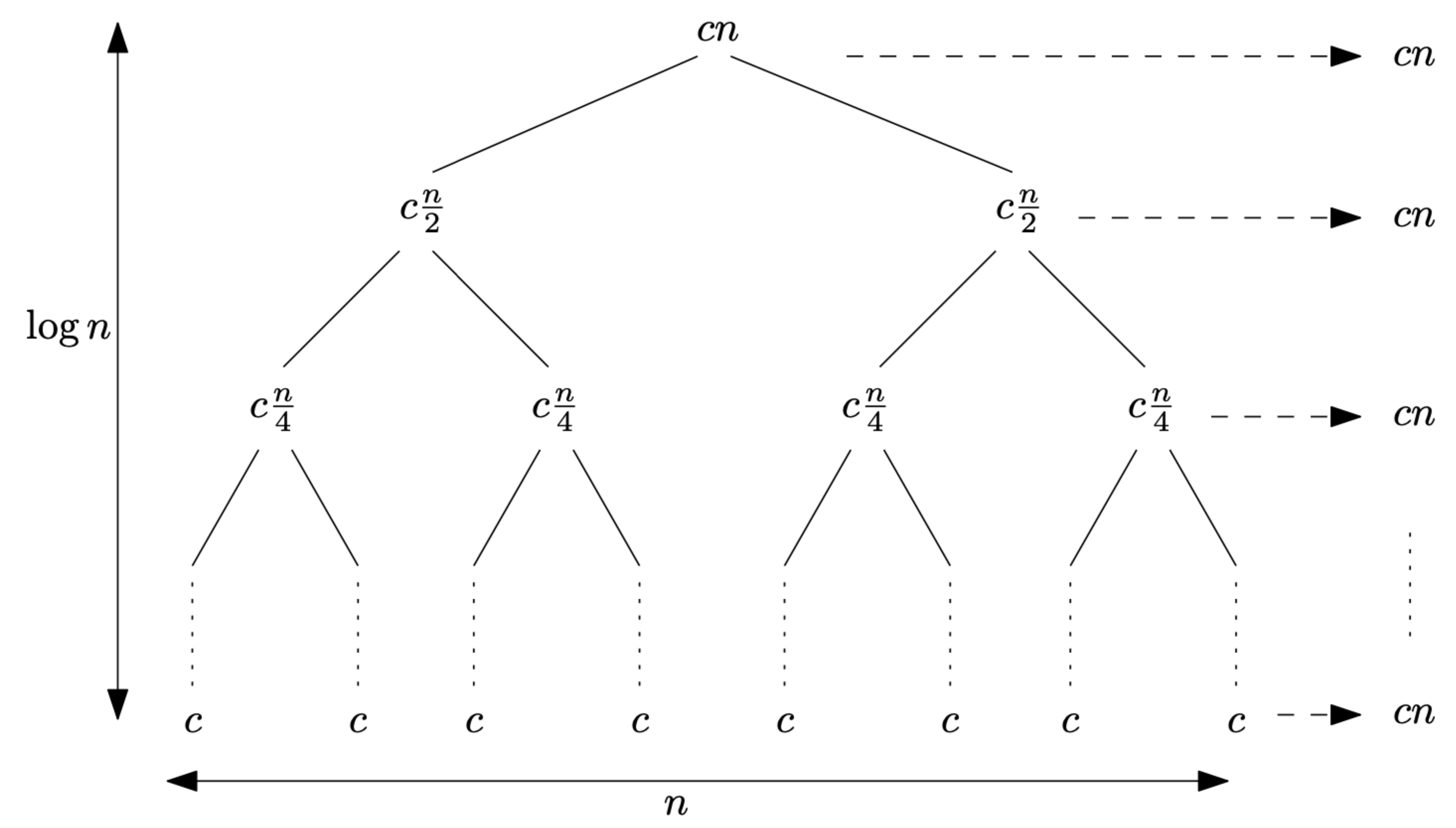
- Runtime based on subproblem size reduction before recursion, in sum:
  - "less than original" $\Rightarrow \Theta(n^k)$
  - "same as original" $\Rightarrow \Theta(n^k \log n)$
  - "more than original" $\Rightarrow \Theta(n^c)$
- k determined by cost of "merge" step
- e.g., Mergesort has $\Theta(n \log n)$

Theorem 3.3 (Master Theorem) *Let* $T : \mathbb{N} \to \mathbb{R}$ *with*

$$T(n) = \sum_{i=1}^{m} T(\alpha_i \cdot n) + \Theta(n^k),$$

where $\alpha_i \in \mathbb{R}$ *with* $0 < \alpha_i < 1,\ m \in \mathbb{N}$ *and* $k \in \mathbb{R}$. *Then*

$$T(n) \in \begin{cases} \Theta(n^k) & \textit{for } \sum_{i=1}^{m} \alpha_i^k < 1 \\ \Theta(n^k \log(n)) & \textit{for } \sum_{i=1}^{m} \alpha_i^k = 1 \\ \Theta(n^c) & \textit{with } \sum_{i=1}^{m} \alpha_i^c = 1 \textit{ for } \sum_{i=1}^{m} \alpha_i^k > 1 \end{cases}$$



Recap:

“Median of Medians” Algorithm

Closest pair — Bounds

Closest Pair — Divide and conquer

“Median of Medians” Algorithm
Medians [Blum, Floyd, Pratt, Rivest, Tarjan 1973]

Theorem 3.4: A median of n numbers can be computed in $\mathcal{O}(n)$.

“Median of Medians” Algorithm
Medians [Blum, Floyd, Pratt, Rivest, Tarjan 1973]

Theorem 3.4: A median of n numbers can be computed in $\mathcal{O}(n)$.

- Group the numbers into sets of 5.

1	6	11	17	3
22	18	16	5	9
10	21	2	12	15
13	4	20	19	7
24	25	8	14	23

$\mathcal{O}(n)$

“Median of Medians” Algorithm

Medians [Blum, Floyd, Pratt, Rivest, Tarjan 1973]

Theorem 3.4: A median of n numbers can be computed in $\mathcal{O}(n)$.

- Group the numbers into sets of 5.

1	6	11	17	3
22	18	16	5	9
10	21	2	12	15
13	4	20	19	7
24	25	8	14	23

$\mathcal{O}(n)$

- Sort these quintuples individually.

1	4	2	5	3
10	6	8	12	7
13	18	11	14	9
22	21	16	17	15
24	25	20	19	23

$\mathcal{O}(n)$

"Median of Medians" Algorithm
Medians [Blum, Floyd, Pratt, Rivest, Tarjan 1973]

Theorem 3.4: A median of n numbers can be computed in $\mathcal{O}(n)$.

- Group the numbers into sets of 5.

1	6	11	17	3
22	18	16	5	9
10	21	2	12	15
13	4	20	19	7
24	25	8	14	23

$\mathcal{O}(n)$

- Sort these quintuples individually.

1	4	2	5	3
10	6	8	12	7
13	18	11	14	9
22	21	16	17	15
24	25	20	19	23

$\mathcal{O}(n)$

- Compute the median of each group.

1	4	2	5	3
10	6	8	12	7
13	18	11	14	9
22	21	16	17	15
24	25	20	19	23

$T(\frac{n}{5})$

“Median of Medians” Algorithm

Medians [Blum, Floyd, Pratt, Rivest, Tarjan 1973]

Theorem 3.4: A median of n numbers can be computed in $\mathcal{O}(n)$.

- Group the numbers into sets of 5.

1	6	11	17	3
22	18	16	5	9
10	21	2	12	15
13	4	20	19	7
24	25	8	14	23

$\mathcal{O}(n)$

- Sort these quintuples individually.

1	4	2	5	3
10	6	8	12	7
13	18	11	14	9
22	21	16	17	15
24	25	20	19	23

$\mathcal{O}(n)$

- Compute the median of each group.

1	4	2	5	3
10	6	8	12	7
13	18	11	14	9
22	21	16	17	15
24	25	20	19	23

$T(\frac{n}{5})$

- Use the *median of medians* as a pivot.

≥ n/4 numbers

≥ n/4 numbers

3	2	1	5	4
7	8	10	12	6
9	11	13	14	18
15	16	22	17	21
23	20	24	19	25

$T(\frac{3n}{4})$

"Median of Medians" Algorithm

Medians [Blum, Floyd, Pratt, Rivest, Tarjan 1973]

Theorem 3.4: A median of n numbers can be computed in $\mathcal{O}(n)$.

$$T(n) = T(\frac{n}{5}) + T(\frac{3n}{4}) + \Theta(n)$$

$$\sum_{i=1}^{m} \alpha_i^k = \frac{1}{5} + \frac{3}{4} = \frac{19}{20} < 1.$$

Theorem 3.3 (Master Theorem) *Let* $T : \mathbb{N} \to \mathbb{R}$ *with*

$$T(n) = \sum_{i=1}^{m} T(\alpha_i \cdot n) + \Theta(n^k),$$

where $\alpha_i \in \mathbb{R}$ *with* $0 < \alpha_i < 1$, $m \in \mathbb{N}$ *and* $k \in \mathbb{R}$. *Then*

$$T(n) \in \begin{cases} \Theta(n^k) & \textit{for } \sum_{i=1}^{m} \alpha_i^k < 1 \\ \Theta(n^k \log(n)) & \textit{for } \sum_{i=1}^{m} \alpha_i^k = 1 \\ \Theta(n^c) & \textit{with } \sum_{i=1}^{m} \alpha_i^c = 1 \textit{ for } \sum_{i=1}^{m} \alpha_i^k > 1 \end{cases}$$

"Median of Medians" Algorithm

Medians [Blum, Floyd, Pratt, Rivest, Tarjan 1973]

Theorem 3.4: A median of n numbers can be computed in $\mathcal{O}(n)$.

$$T(n) = T(\frac{n}{5}) + T(\frac{3n}{4}) + \Theta(n)$$

$$\sum_{i=1}^{m} \alpha_i^k = \frac{1}{5} + \frac{3}{4} = \frac{19}{20} < 1.$$

Theorem 3.3 (Master Theorem) *Let* $T : \mathbb{N} \to \mathbb{R}$ *with*

where α_i … *and* $k \in \mathbb{R}$. *Then*

$$T(n) \in \Theta(n^k) = \Theta(n)$$

$$T(n) \in \begin{cases} \Theta(n^k) & \text{for } \sum_{i=1}^m \alpha_i^k < 1 \\ \Theta(n^k \log(n)) & \text{for } \sum_{i=1}^m \alpha_i^k = 1 \\ \Theta(n^c) & \text{with } \sum_{i=1}^m \alpha_i^c = 1 \text{ for } \sum_{i=1}^m \alpha_i^k > 1 \end{cases}$$

Recap:

Closest pair — Bounds

Closest Pair — Divide and conquer

Lower bound for the Closest pair Problem

Via SORTING and ELEMENT UNIQUENESS

Lower bound for the Closest pair Problem

Via SORTING and ELEMENT UNIQUENESS

- **Known:** SORTING has a lower bound of $\Omega(n \log n)$.

Lower bound for the Closest pair Problem

Via SORTING and ELEMENT UNIQUENESS

- **Known:** SORTING has a lower bound of $\Omega(n \log n)$.
- **Related:** ELEMENT UNIQUENESS asks whether all elements of an unsorted list are unique, i.e., whether no element occurs twice.

Lower bound for the Closest pair Problem

Via SORTING and ELEMENT UNIQUENESS

- **Known:** SORTING has a lower bound of $\Omega(n \log n)$.
- **Related:** ELEMENT UNIQUENESS asks whether all elements of an unsorted list are unique, i.e., whether no element occurs twice.
- Both require at least $\Omega(n \log n)$ comparisons to solve! $\Rightarrow$ Decision trees

Lower bound for the Closest pair Problem

Via Sorting and Element Uniqueness

- **Known:** Sorting has a lower bound of $\Omega(n \log n)$.
- **Related:** Element Uniqueness asks whether all elements of an unsorted list are unique, i.e., whether no element occurs twice.
- Both require at least $\Omega(n \log n)$ comparisons to solve! $\Rightarrow$ Decision trees
- **We showed:** Closest pair solves the Element Uniqueness Problem: If closest pair has distance zero, the are not unique.

Lower bound for the Closest pair Problem

Via SORTING and ELEMENT UNIQUENESS

- **Known:** SORTING has a lower bound of $\Omega(n \log n)$.
- **Related:** ELEMENT UNIQUENESS asks whether all elements of an unsorted list are unique, i.e., whether no element occurs twice.
- Both require at least $\Omega(n \log n)$ comparisons to solve! $\Rightarrow$ Decision trees
- **We showed:** CLOSEST PAIR solves the ELEMENT UNIQUENESS PROBLEM: If closest pair has distance zero, the are not unique.

Relevant Paper: Ben-Or, "Lower Bounds For Algebraic Computation Trees"

Upper bound for the Closest pair Problem

Upper bound for the Closest pair Problem

- **Theorem 3.7:** A closest pair for n numbers in $\mathbb{R}$ can be found in $\mathcal{O}(n \log n)$.

Upper bound for the Closest pair Problem

- **Theorem 3.7:** A closest pair for n numbers in $\mathbb{R}$ can be found in $\mathcal{O}(n \log n)$.
- **Idea:** Sort, then do a linear scan — closest pair consists of adjacent elements.

Upper bound for the Closest pair Problem

- **Theorem 3.7:** A closest pair for n numbers in $\mathbb{R}$ can be found in $\mathcal{O}(n \log n)$.
- **Idea:** Sort, then do a linear scan — closest pair consists of adjacent elements.

- Apply this to two dimensions, sorting along one axis.

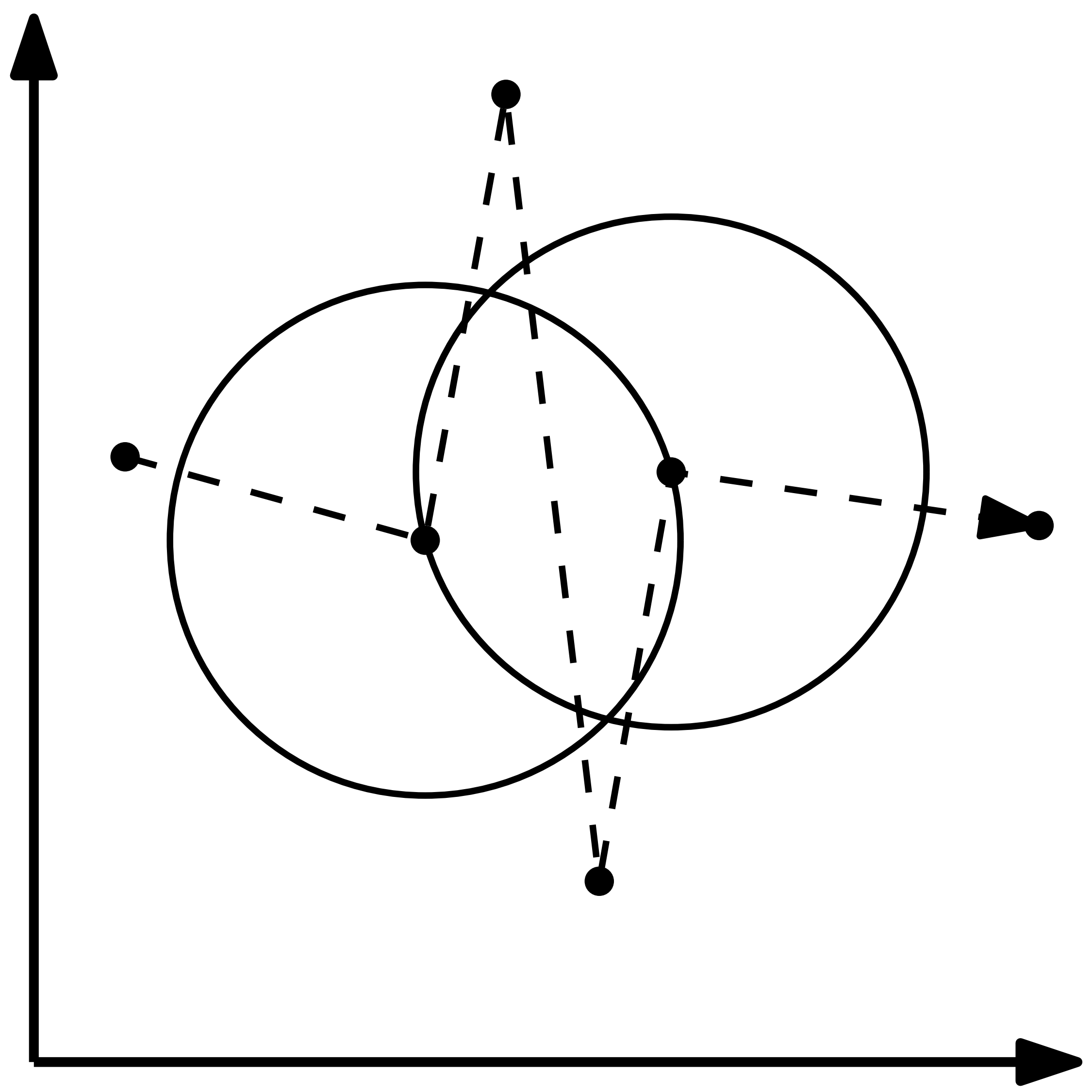
Upper bound for the Closest pair Problem

- **Theorem 3.7:** A closest pair for n numbers in $\mathbb{R}$ can be found in $\mathcal{O}(n \log n)$.
- **Idea:** Sort, then do a linear scan — closest pair consists of adjacent elements.

- Apply this to two dimensions, sorting along one axis.
- **Issue:** This is not true of $\mathbb{R}^2$, see counterexample.

Upper bound for the Closest pair Problem

- **Theorem 3.7:** A closest pair for n numbers in $\mathbb{R}$ can be found in $\mathcal{O}(n \log n)$.
- **Idea:** Sort, then do a linear scan — closest pair consists of adjacent elements.



- Apply this to two dimensions, sorting along one axis.
- **Issue:** This is not true of $\mathbb{R}^2$, see counterexample.

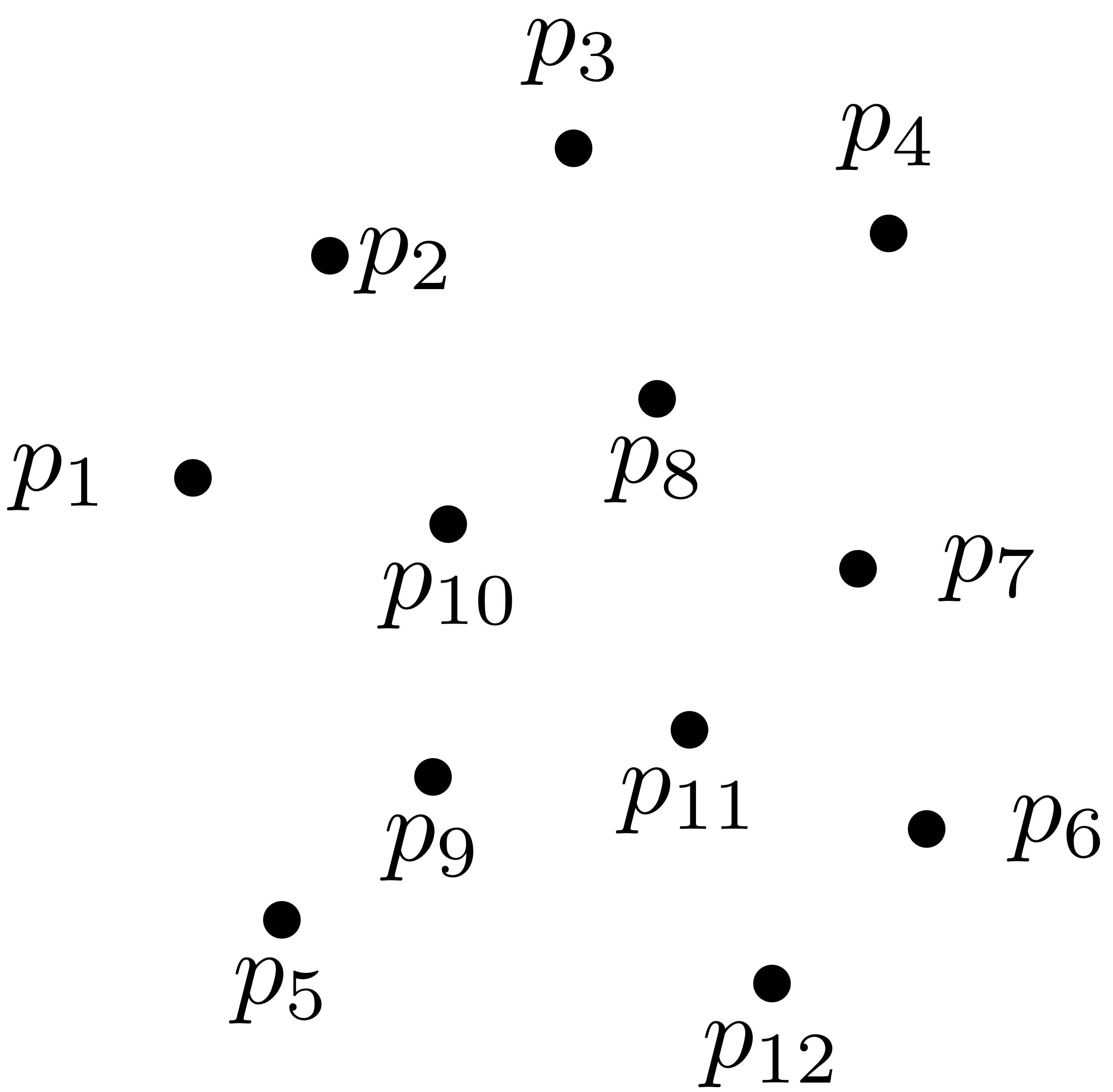
Recap:

Closest Pair — Divide and conquer

Peter Kramer | November 12th, 2024

Closest Pair — Divide and conquer Bentley & Shamos, 1976

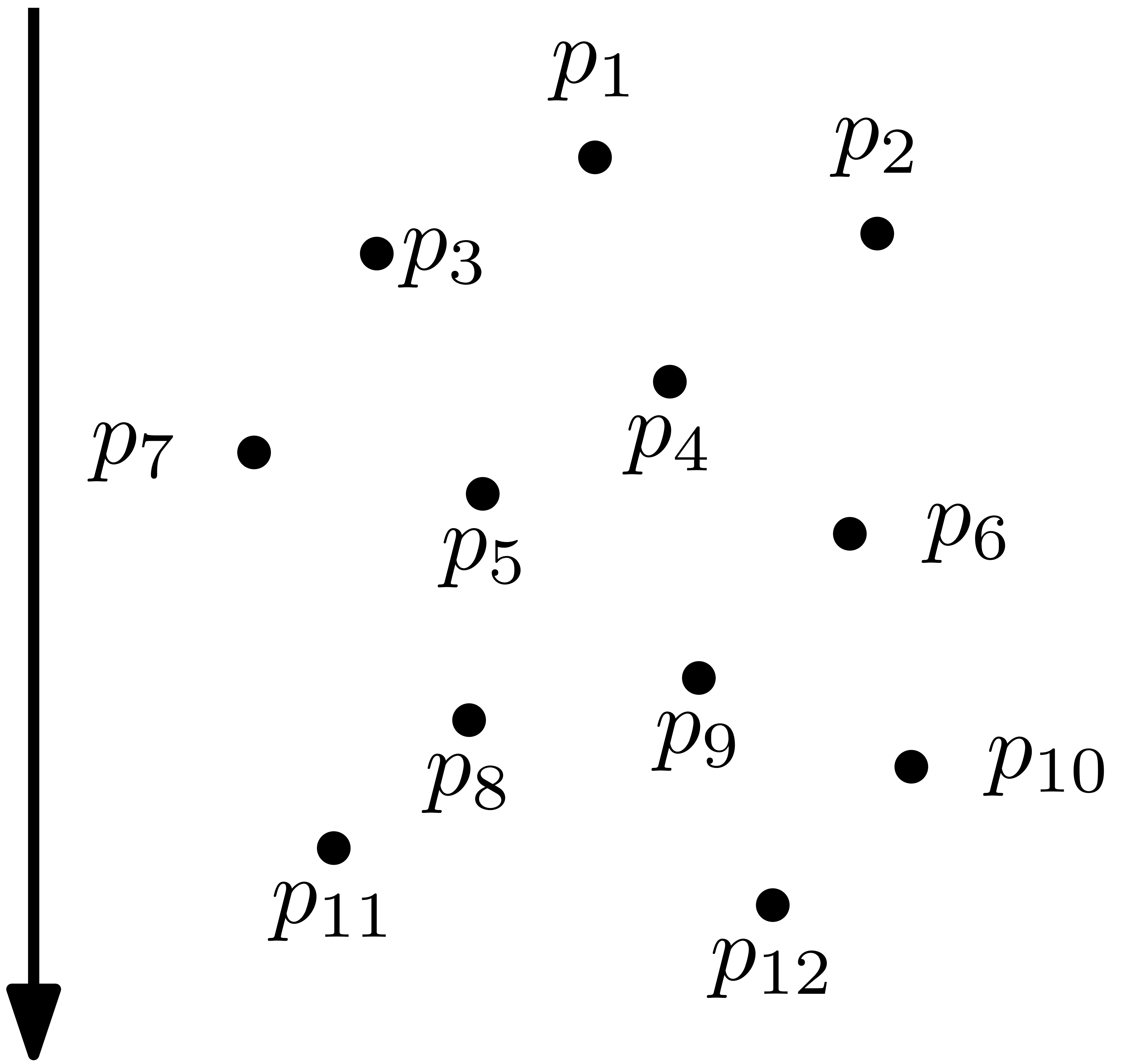
- **Theorem 3.8:** A closest pair for n numbers in $\mathbb{R}^2$ can be found in $\mathcal{O}(n \log n)$.



Closest Pair — Divide and conquer
Bentley & Shamos, 1976

- **Theorem 3.8:** A closest pair for n numbers in $\mathbb{R}^2$ can be found in $\mathcal{O}(n \log n)$.

1. Sort $\mathscr{P}$ by y-coordinates, i.e., $\geq_y$.

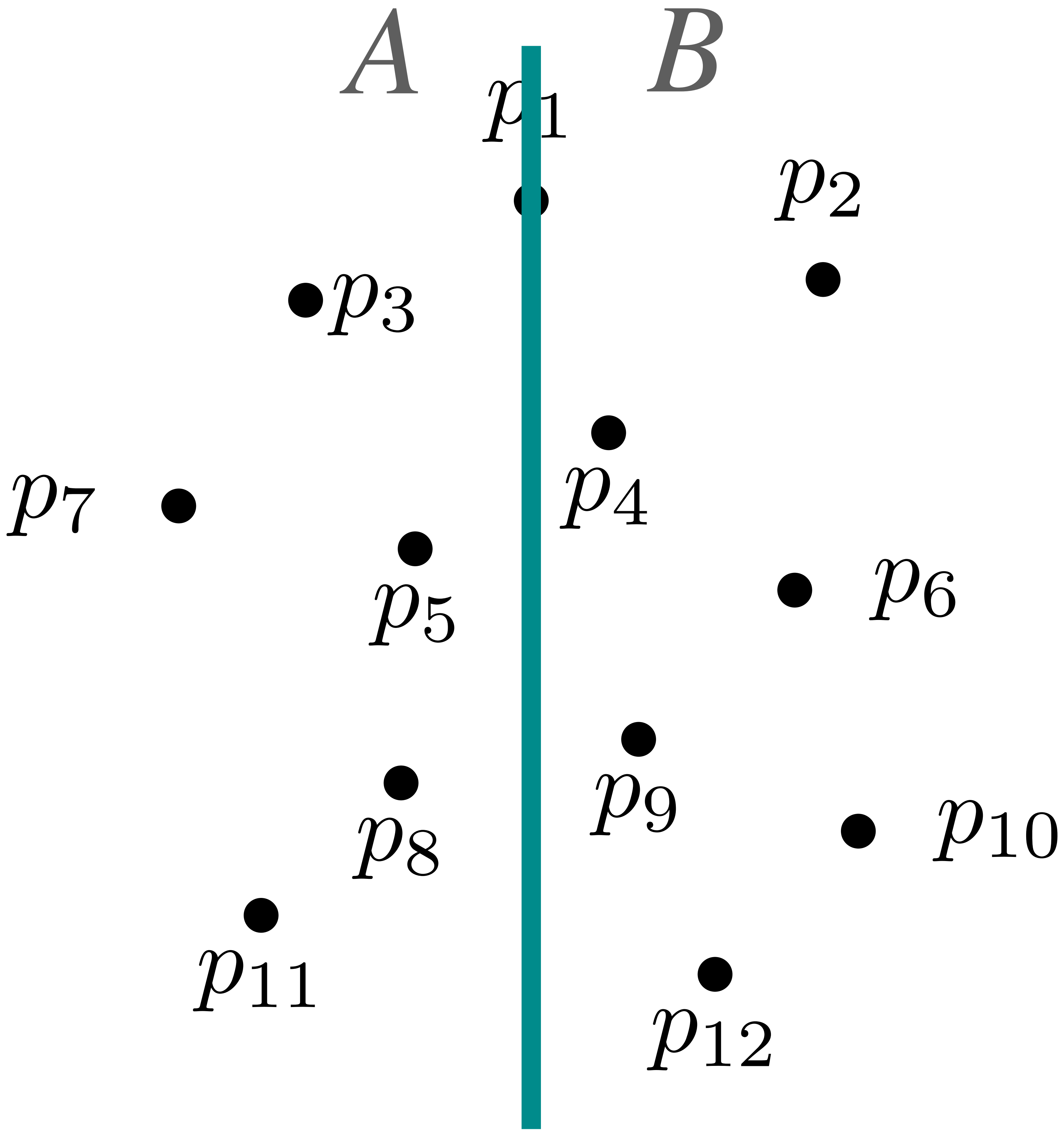


Closest Pair — Divide and conquer
Bentley & Shamos, 1976

- **Theorem 3.8:** A closest pair for n numbers in $\mathbb{R}^2$ can be found in $\mathcal{O}(n \log n)$.

1. Sort $\mathcal{P}$ by y-coordinates, i.e., $\geq_y$.

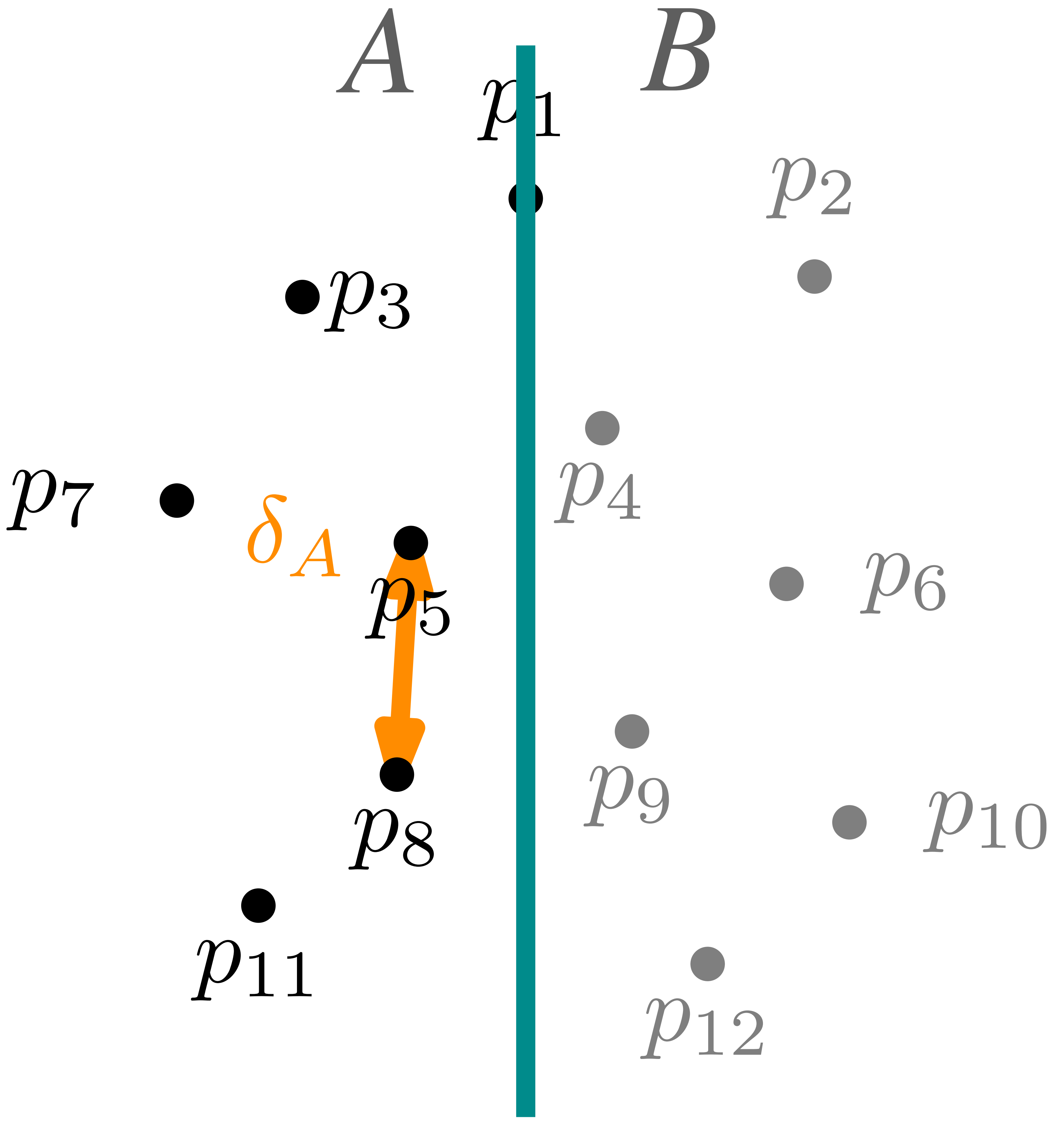
2. Compute an x-median of $\mathcal{P}$, pivot.
   We get disjoint sets $A \cup B = \mathcal{P}$.



Closest Pair — Divide and conquer
Bentley & Shamos, 1976

- **Theorem 3.8:** A closest pair for n numbers in $\mathbb{R}^2$ can be found in $\mathcal{O}(n \log n)$.

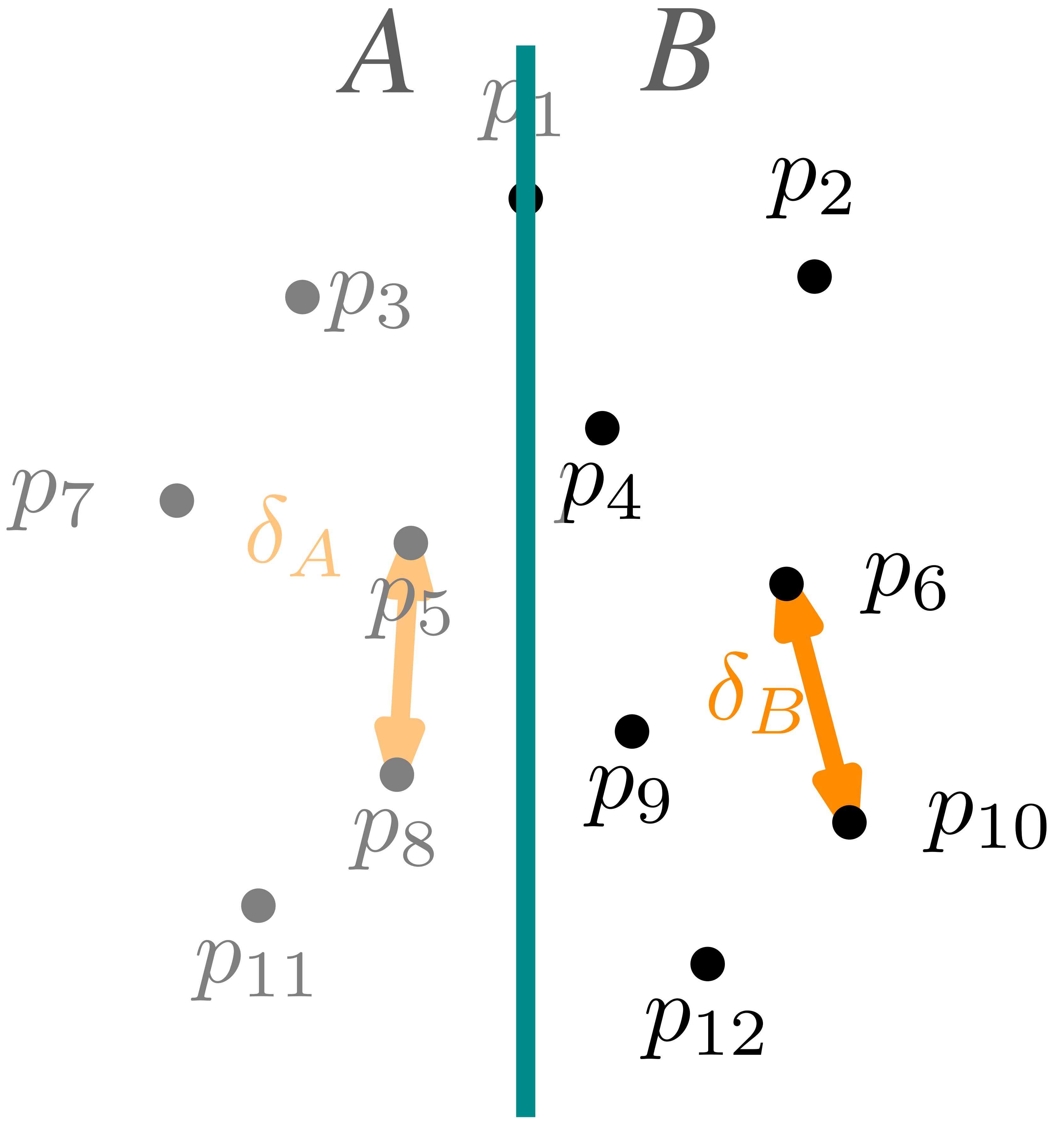
1. Sort $\mathscr{P}$ by y-coordinates, i.e., $\geq_y$.
2. Compute an x-median of $\mathscr{P}$, pivot.
   We get disjoint sets $A \cup B = \mathscr{P}$.
3. Compute δ_A and δ_B recursively.



Closest Pair — Divide and conquer Bentley & Shamos, 1976

- **Theorem 3.8:** A closest pair for n numbers in $\mathbb{R}^2$ can be found in $\mathcal{O}(n \log n)$.

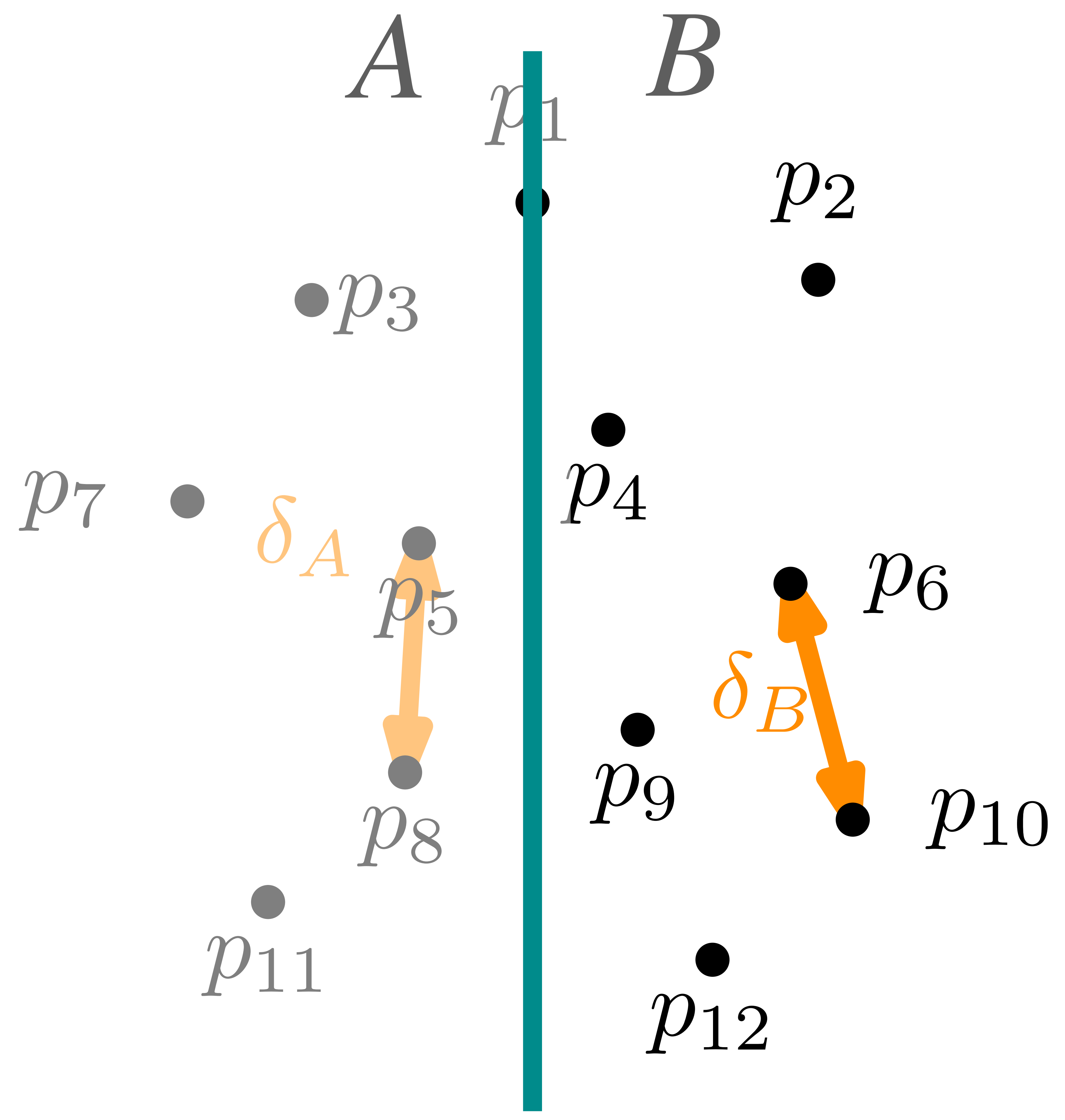
1. Sort $\mathscr{P}$ by y-coordinates, i.e., $\geq_y$.
2. Compute an x-median of $\mathscr{P}$, pivot. We get disjoint sets $A \cup B = \mathscr{P}$.
3. Compute δ_A and δ_B recursively.



Closest Pair — Divide and conquer
Bentley & Shamos, 1976

- **Theorem 3.8:** A closest pair for n numbers in $\mathbb{R}^2$ can be found in $\mathcal{O}(n \log n)$.

1. Sort $\mathscr{P}$ by y-coordinates, i.e., $\geq_y$.
2. Compute an x-median of $\mathscr{P}$, pivot. We get disjoint sets $A \cup B = \mathscr{P}$.
3. Compute δ_A and δ_B recursively.
4. … merge by checking pairs in $A \times B$.

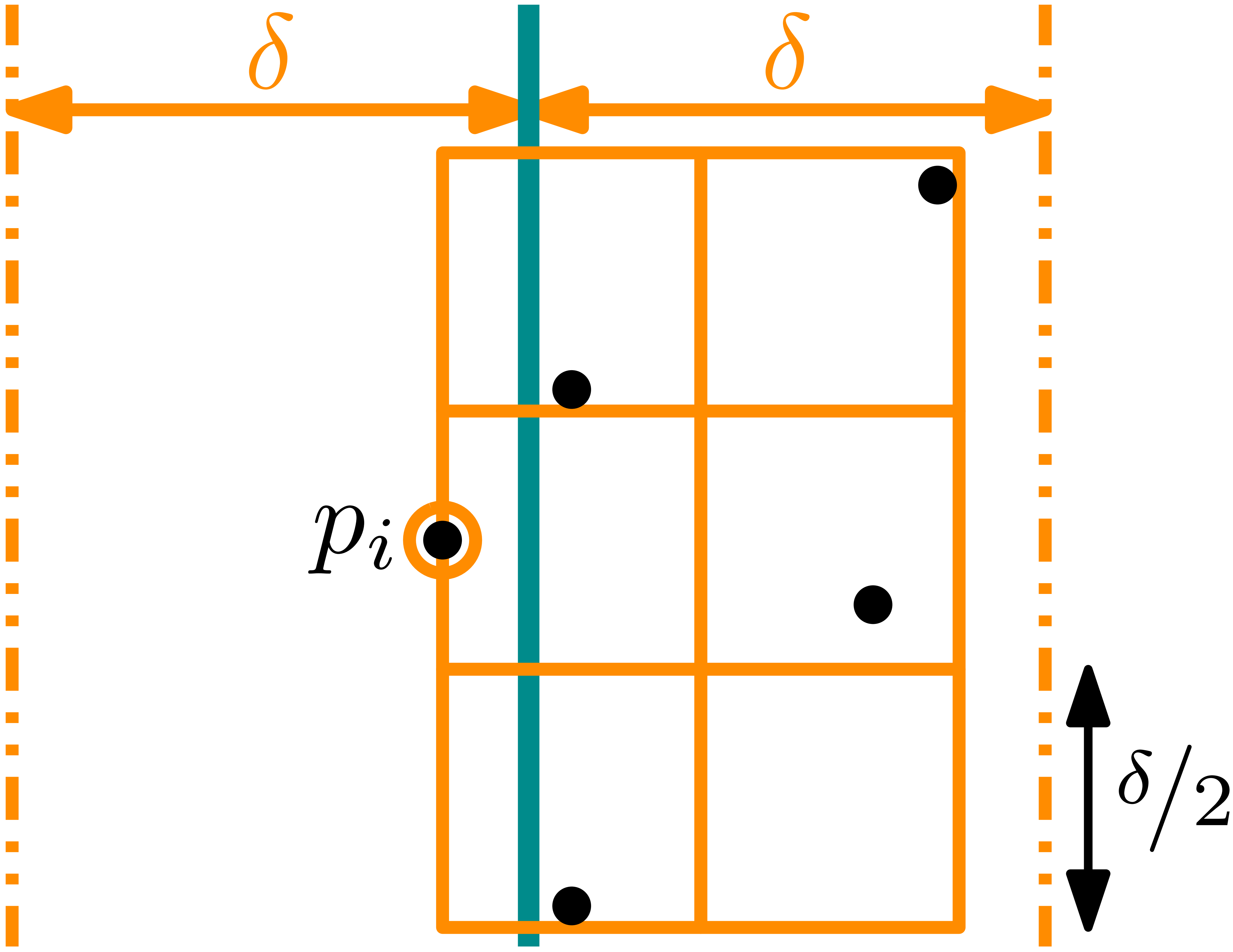


Closest Pair — Divide and conquer
Bentley & Shamos, 1976

- **Theorem 3.8:** A closest pair for n numbers in $\mathbb{R}^2$ can be found in $\mathcal{O}(n \log n)$.

4. … merge by checking pairs in $A \times B$?

- Candidate pairs in $A \times B$ must have distance at most δ to the median line.
- Point pairs in A (B) have distance $\geq \delta$.
- Therefore: For each point in $p_i \in A$, there can be only constantly many points in B that are closer than δ!

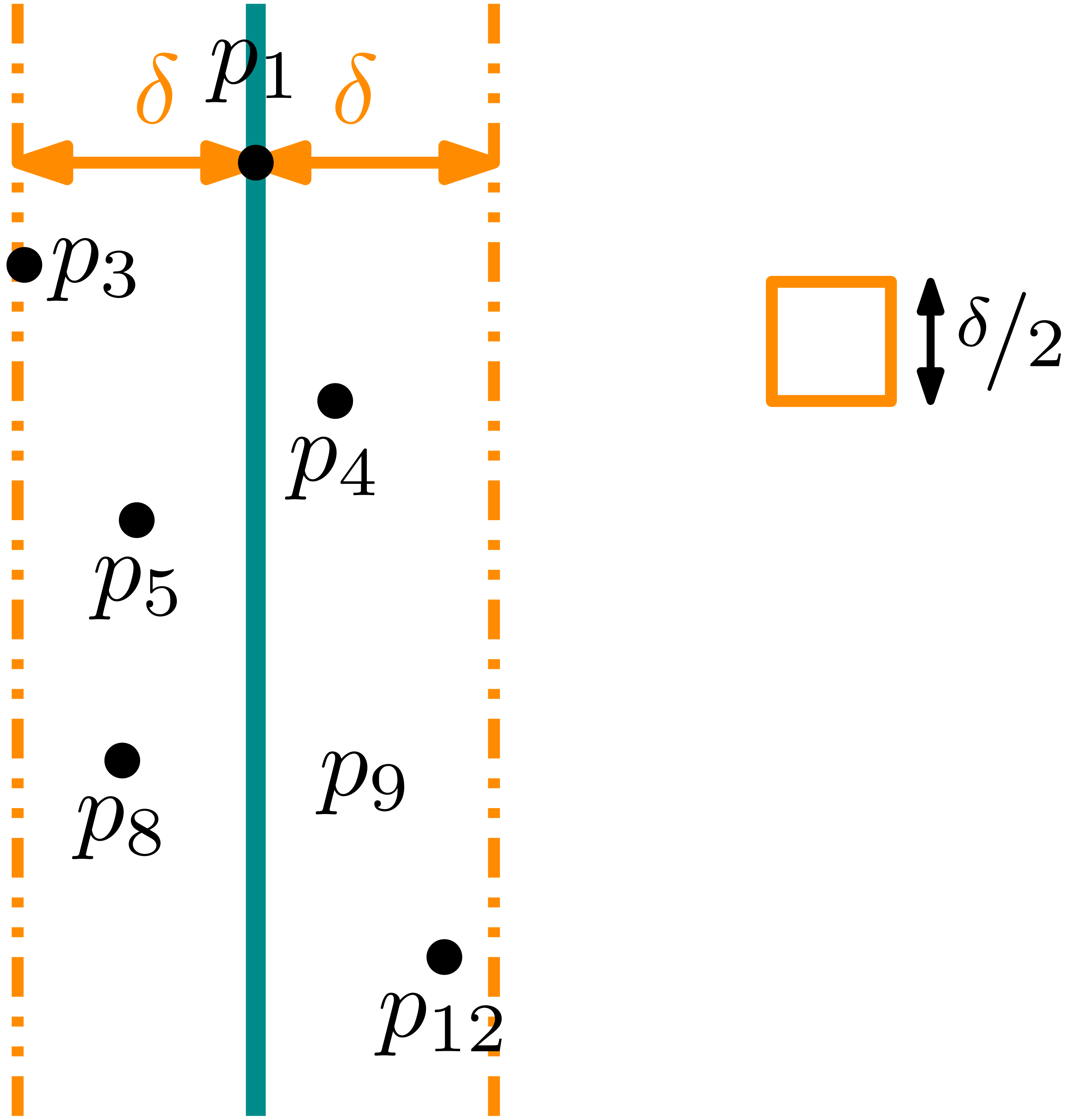


Closest Pair — Divide and conquer Bentley & Shamos, 1976

- **Theorem 3.8:** A closest pair for n numbers in $\mathbb{R}^2$ can be found in $\mathcal{O}(n \log n)$.

4. … merge by checking pairs in $A \times B$:

- Filter based on distance to median line, remove if farther than $\delta = \min(\delta_A, \delta_B)$.
- Linear scan along one side, y-maximal to y-minimal coordinates.
- Keep track of candidates on the other side of the median line.
- This idea generalizes to higher dimensions!

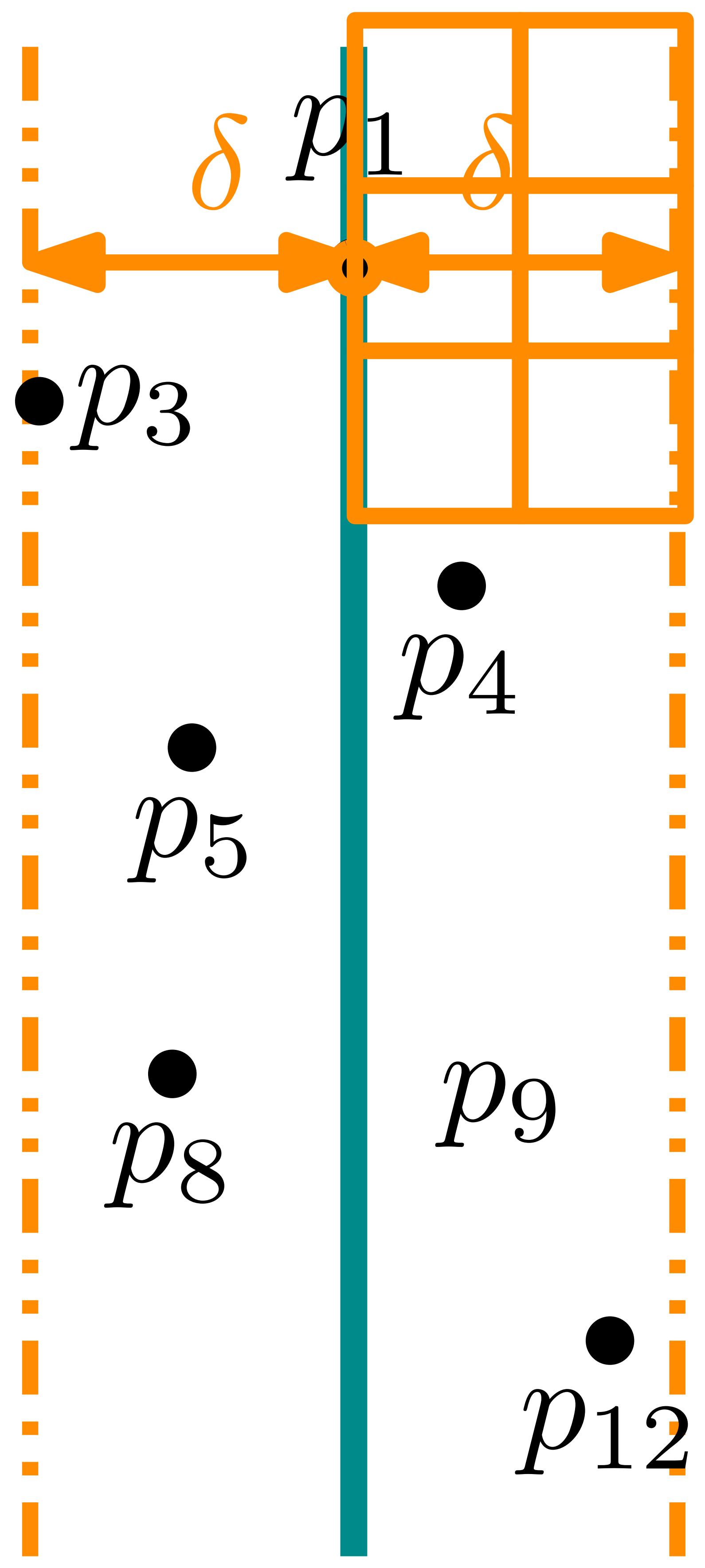


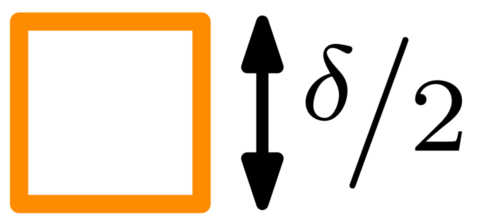
Closest Pair — Divide and conquer
Bentley & Shamos, 1976

- **Theorem 3.8:** A closest pair for n numbers in $\mathbb{R}^2$ can be found in $\mathcal{O}(n \log n)$.

4. … merge by checking pairs in $A \times B$:

- Filter based on distance to median line, remove if farther than $\delta = \min(\delta_A, \delta_B)$.
- Linear scan along one side, y-maximal to y-minimal coordinates.
- Keep track of candidates on the other side of the median line.
- This idea generalizes to higher dimensions!



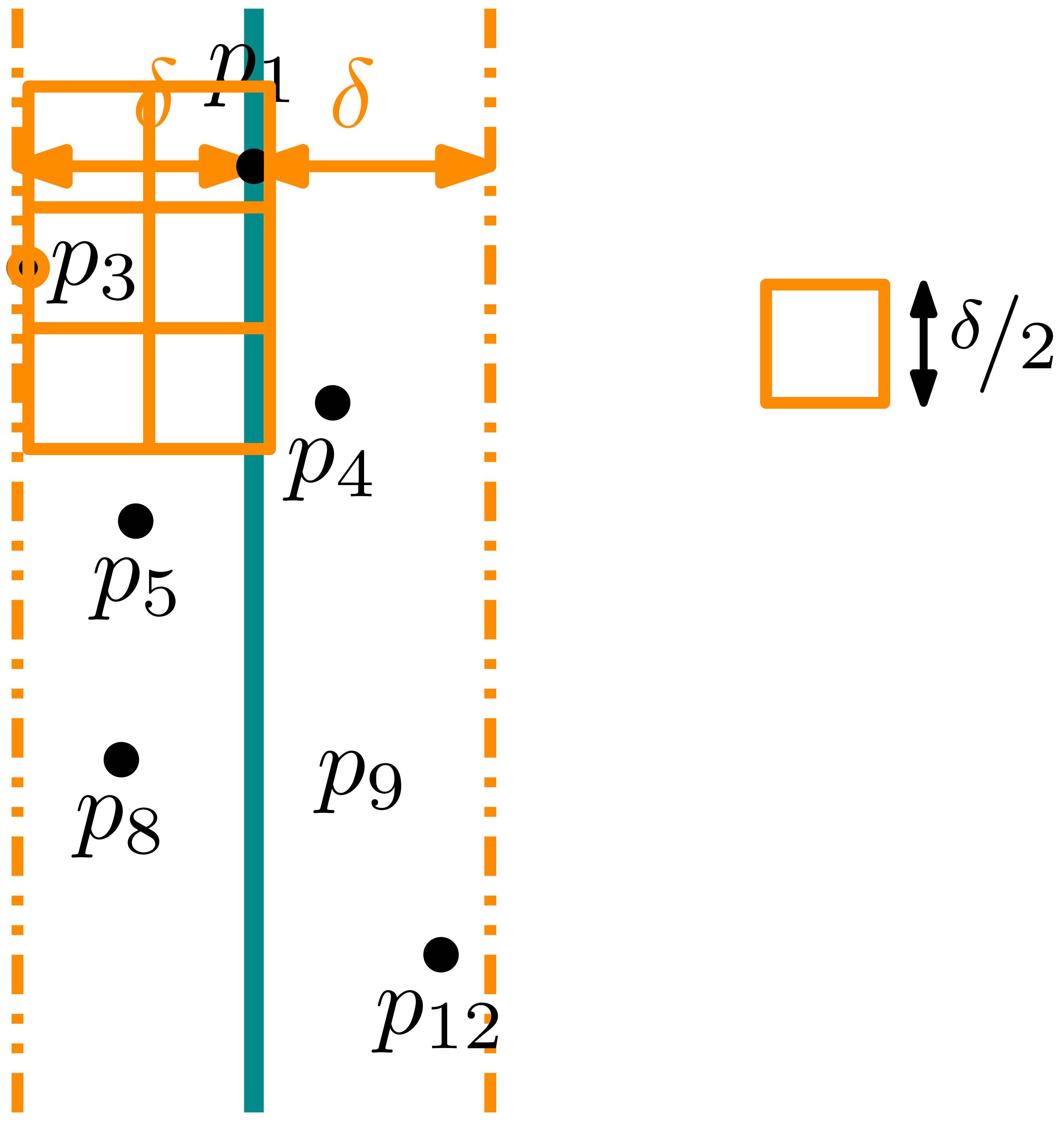


Closest Pair — Divide and conquer
Bentley & Shamos, 1976

- **Theorem 3.8:** A closest pair for n numbers in $\mathbb{R}^2$ can be found in $\mathcal{O}(n \log n)$.

4. … merge by checking pairs in $A \times B$:

- Filter based on distance to median line, remove if farther than $\delta = \min(\delta_A, \delta_B)$.
- Linear scan along one side, y-maximal to y-minimal coordinates.
- Keep track of candidates on the other side of the median line.
- This idea generalizes to higher dimensions!

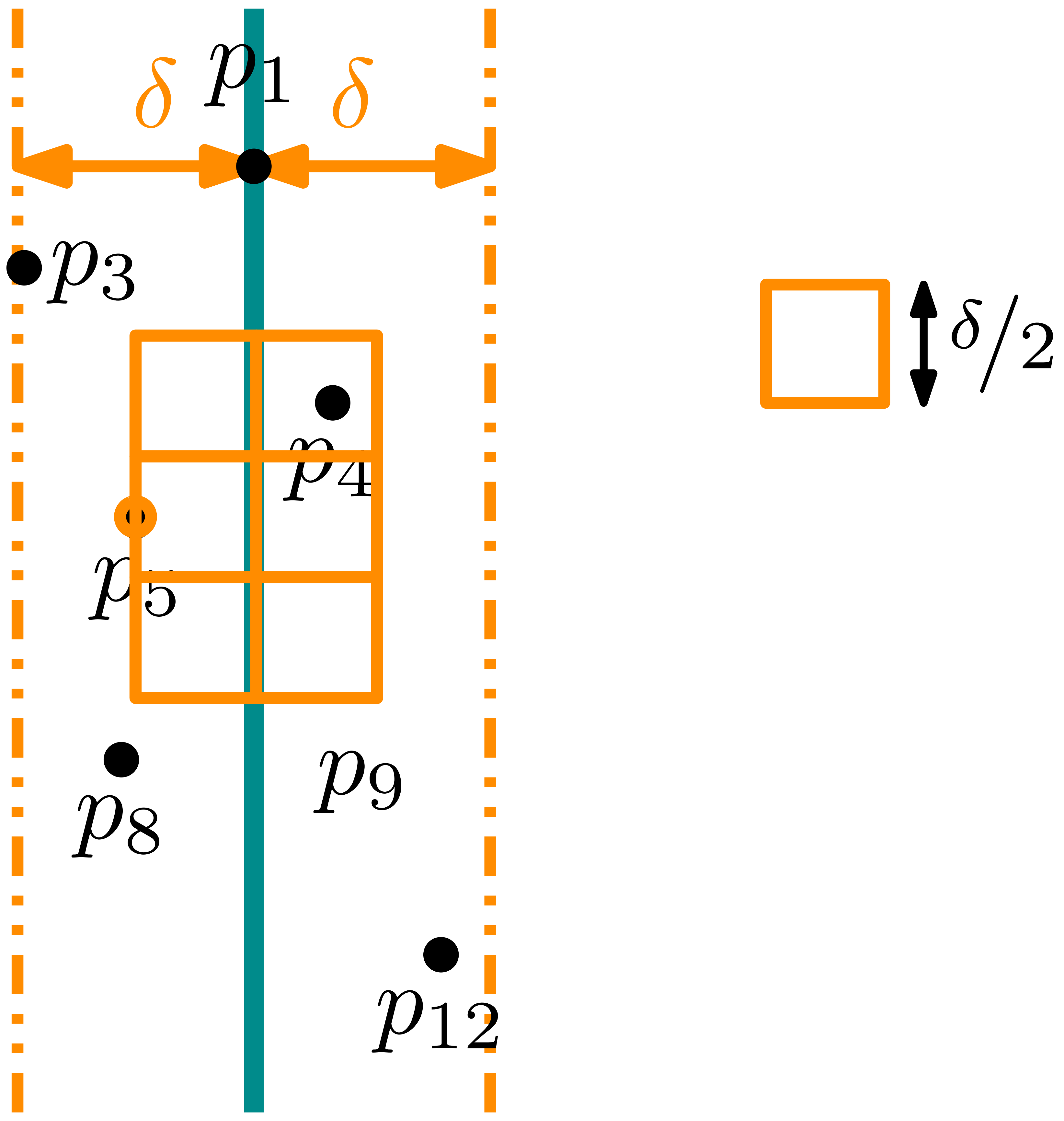


Closest Pair — Divide and conquer
Bentley & Shamos, 1976

- **Theorem 3.8:** A closest pair for n numbers in $\mathbb{R}^2$ can be found in $\mathcal{O}(n \log n)$.

4. … merge by checking pairs in $A \times B$:

- Filter based on distance to median line, remove if farther than $\delta = \min(\delta_A, \delta_B)$.
- Linear scan along one side, y-maximal to y-minimal coordinates.
- Keep track of candidates on the other side of the median line.
- This idea generalizes to higher dimensions!

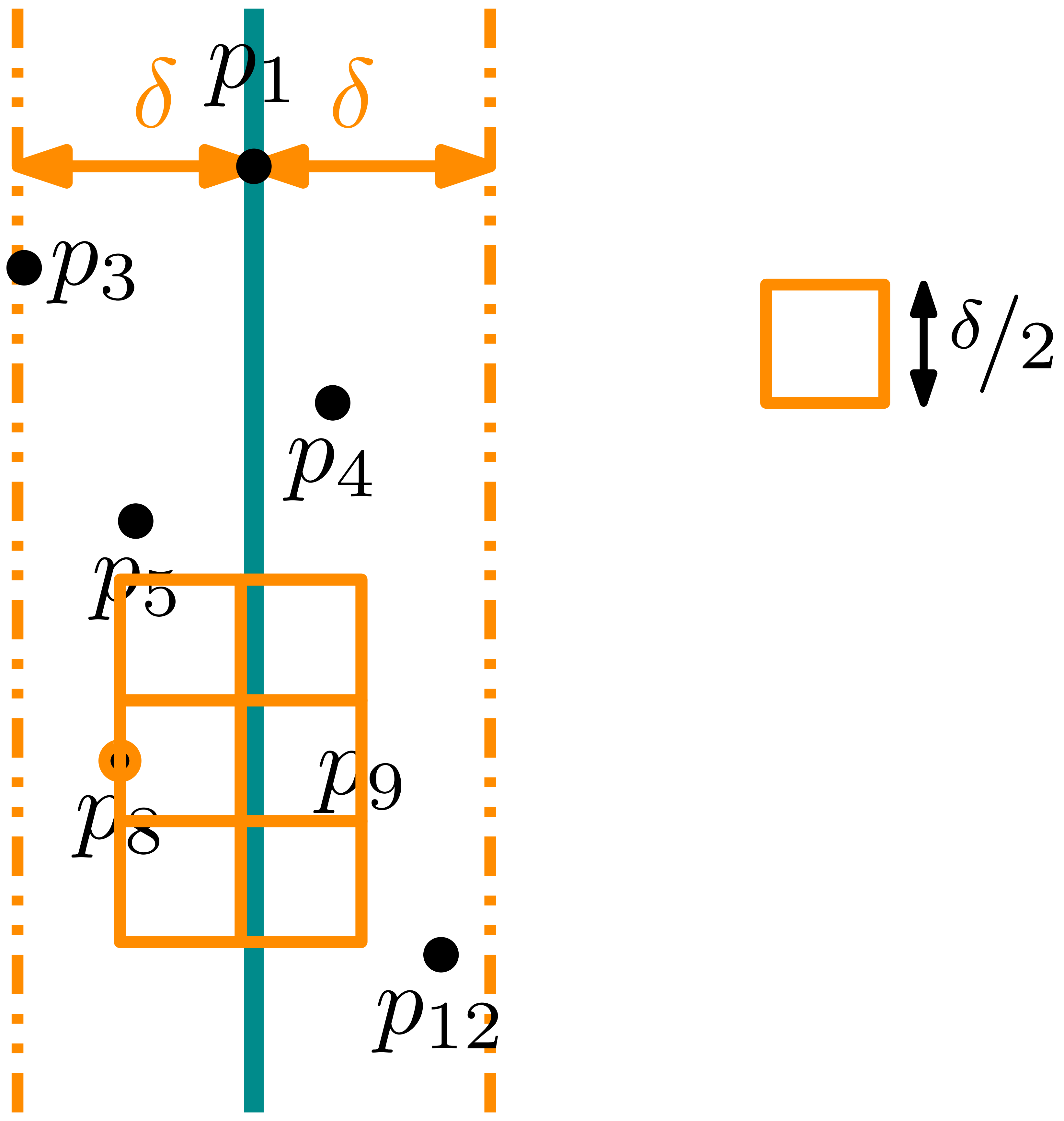


Closest Pair — Divide and conquer Bentley & Shamos, 1976

- **Theorem 3.8:** A closest pair for n numbers in $\mathbb{R}^2$ can be found in $\mathcal{O}(n \log n)$.

4. … merge by checking pairs in $A \times B$:

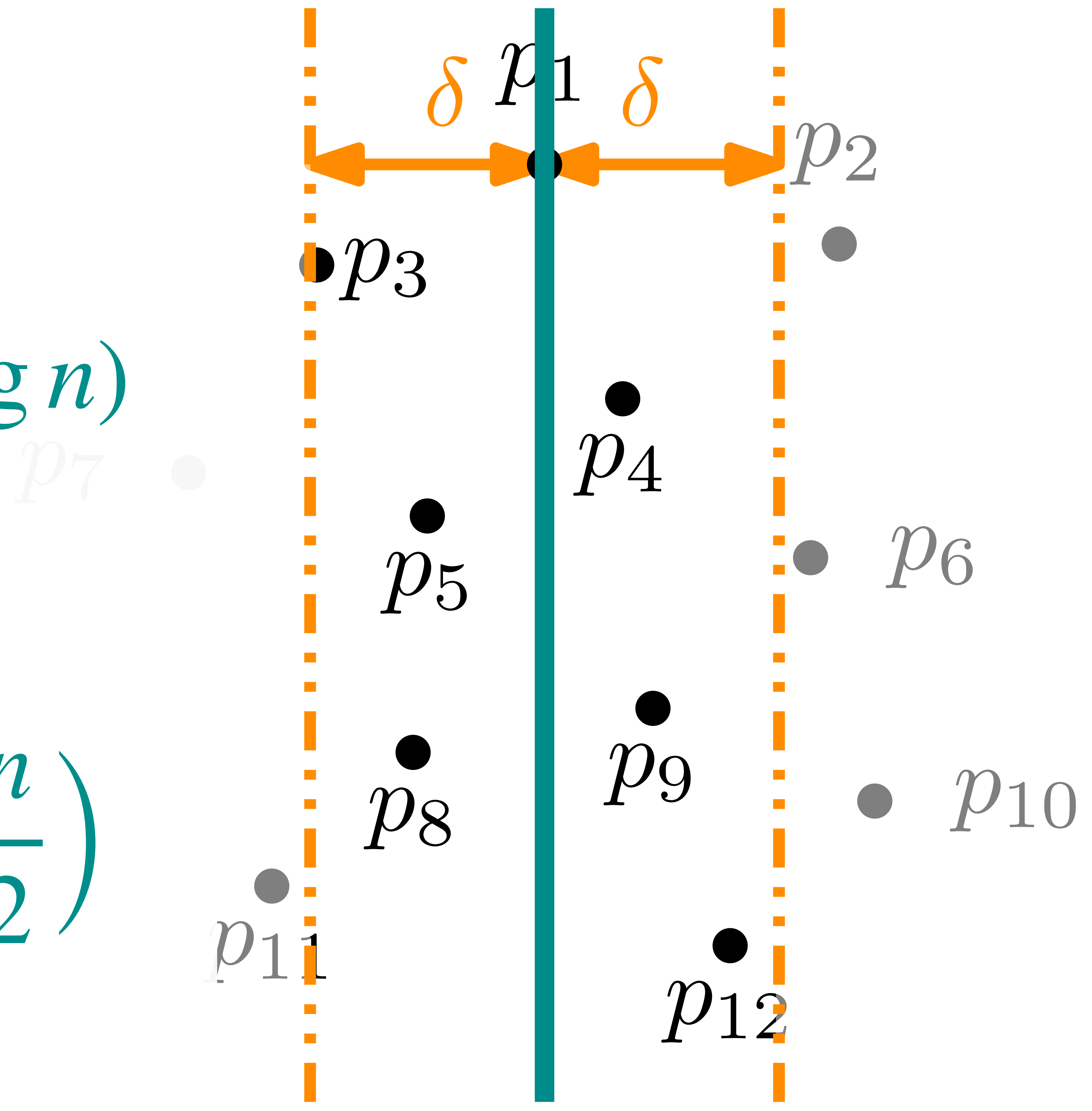
- Filter based on distance to median line, remove if farther than $\delta = \min(\delta_A, \delta_B)$.
- Linear scan along one side, y-maximal to y-minimal coordinates.
- Keep track of candidates on the other side of the median line.
- This idea generalizes to higher dimensions!



Closest Pair — Divide and conquer
Bentley & Shamos, 1976

- **Theorem 3.8:** A closest pair for n numbers in $\mathbb{R}^2$ can be found in $\mathcal{O}(n \log n)$.

1. Sort $\mathscr{P}$ by y-coordinates, i.e., $\geq_y$. $\Theta(n \log n)$
2. Compute an x-median of $\mathscr{P}$, pivot. We get disjoint sets $A \cup B = \mathscr{P}$. $\Theta(n)$
3. Compute δ_A and δ_B recursively. $2 \cdot T\left(\frac{n}{2}\right)$
4. Merge by checking pairs in $A \times B$. $\Theta(n)$

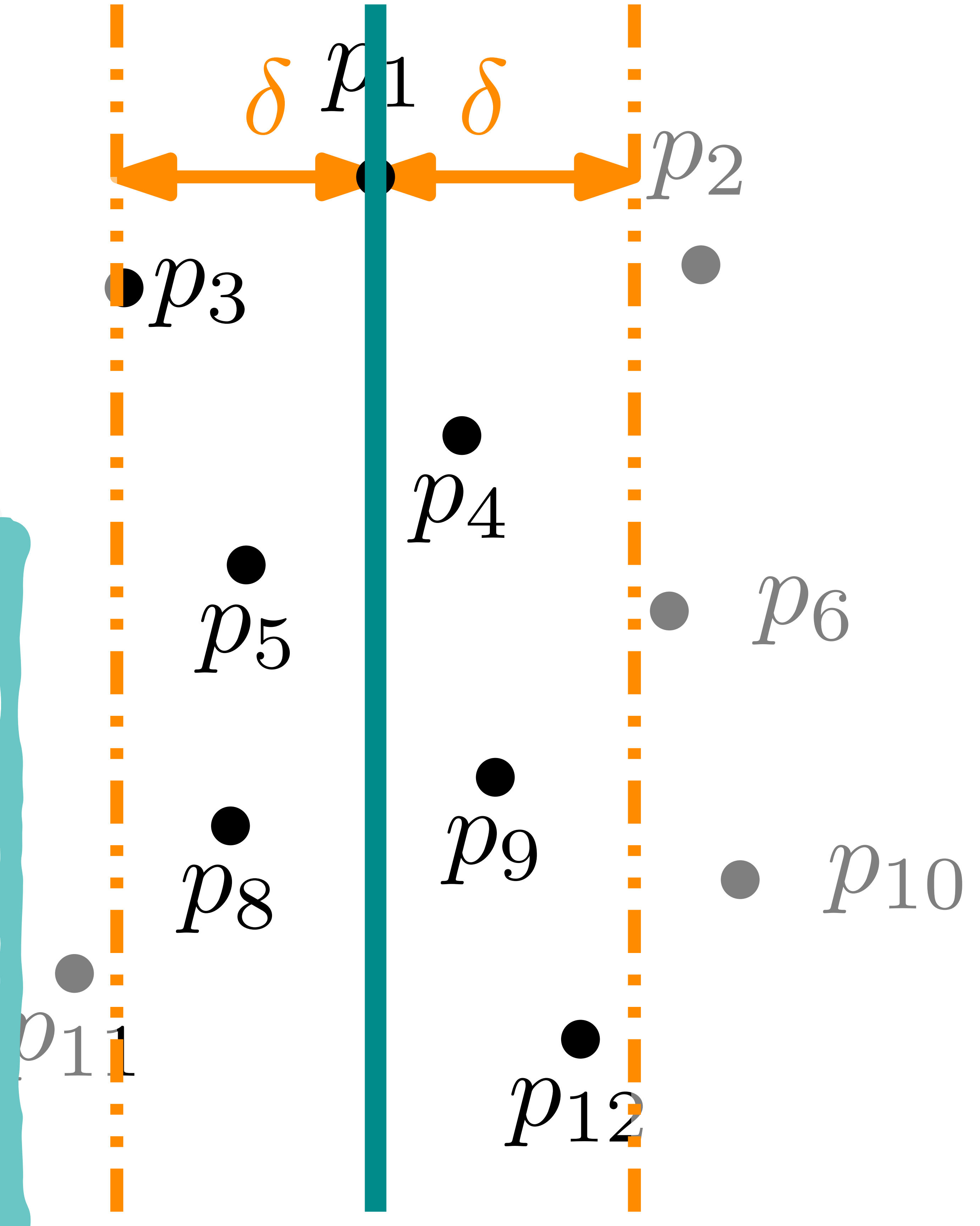


Closest Pair — Divide and conquer
Bentley & Shamos, 1976

- **Theorem 3.8:** A closest pair for n numbers in $\mathbb{R}^2$ can be found in $\mathcal{O}(n \log n)$.

1. Sort $\mathscr{P}$ by y-coordinates, i.e., $\geq_y$. $\Theta(n \log n)$
2. Compute an x-median of $\mathscr{P}$, pivot. We get disjoint sets $A \cup B = \mathscr{P}$. $\Theta(n)$
3. Compute δ_A and δ_B recursively. $2 \cdot T\left(\frac{n}{2}\right)$
4. Merge by checking pairs in $A \times B$. $\Theta(n)$

$T(n)$



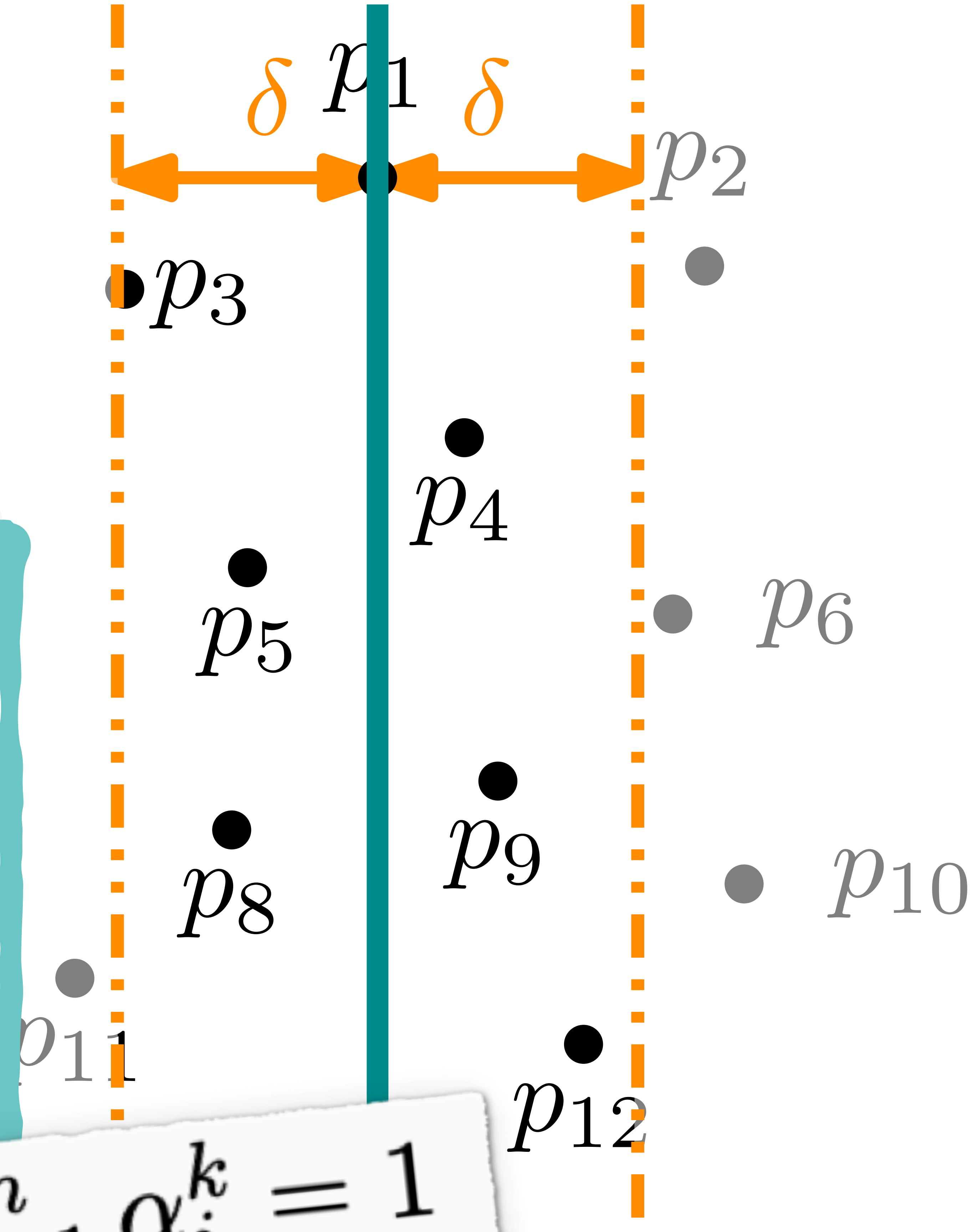
Closest Pair — Divide and conquer
Bentley & Shamos, 1976

- **Theorem 3.8:** A closest pair for n numbers in $\mathbb{R}^2$ can be found in $\mathcal{O}(n \log n)$.

1. Sort $\mathcal{P}$ by y-coordinates, i.e., $\geq_y$. $\Theta(n \log n)$
2. Compute an x-median of $\mathcal{P}$, pivot. We get disjoint sets $A \cup B = \mathcal{P}$. $\Theta(n)$
3. Compute δ_A and δ_B recursively. $2 \cdot T\left(\frac{n}{2}\right)$
4. Merge by checking pairs in $A \times B$. $\Theta(n)$

$T(n)$

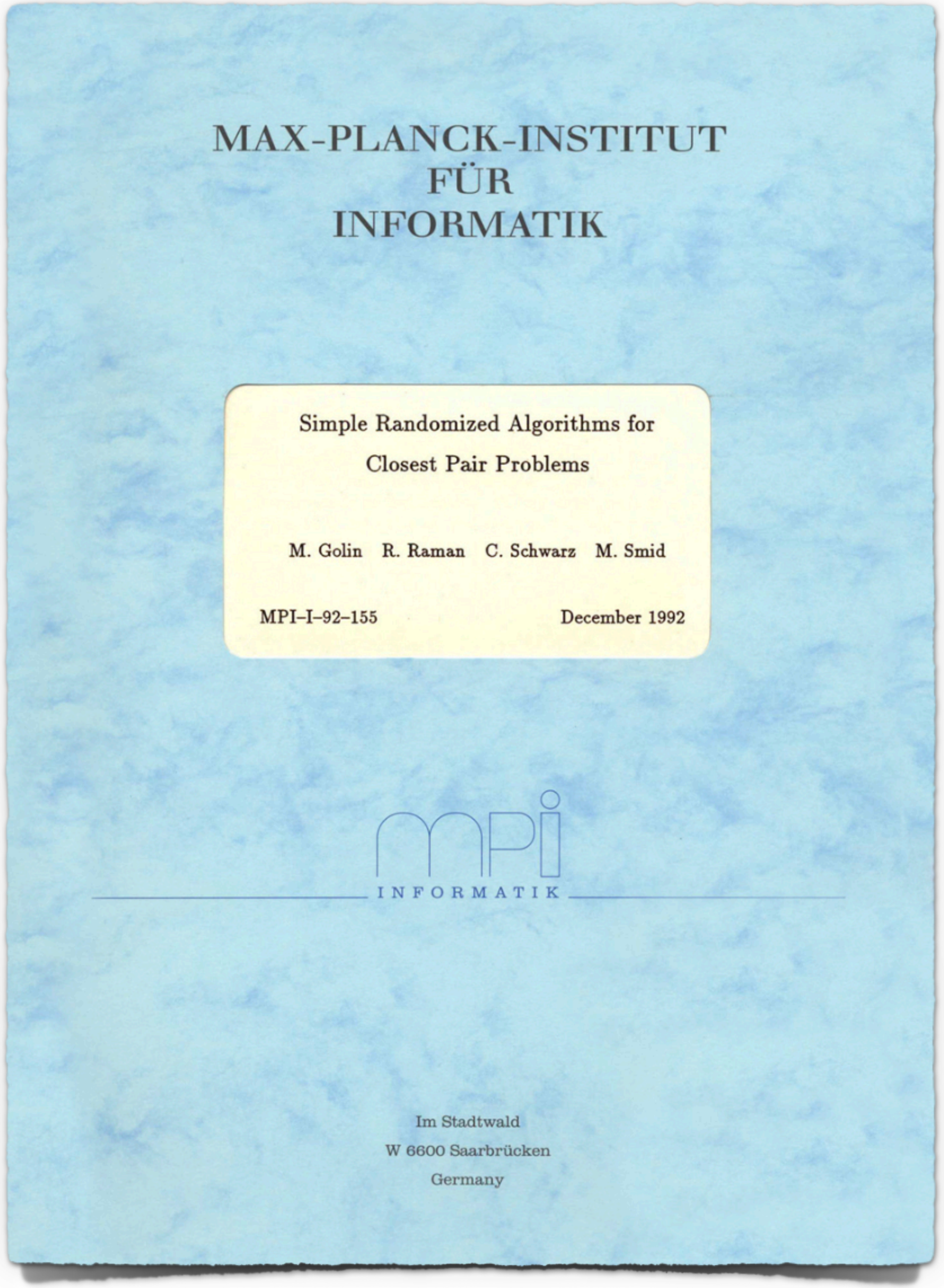
$$\Theta(n^k \log(n)) \quad \textit{for} \ \sum_{i=1}^{m} \alpha_i^k = 1$$



Closest Pair — Randomised Incremental Construction (RIC)

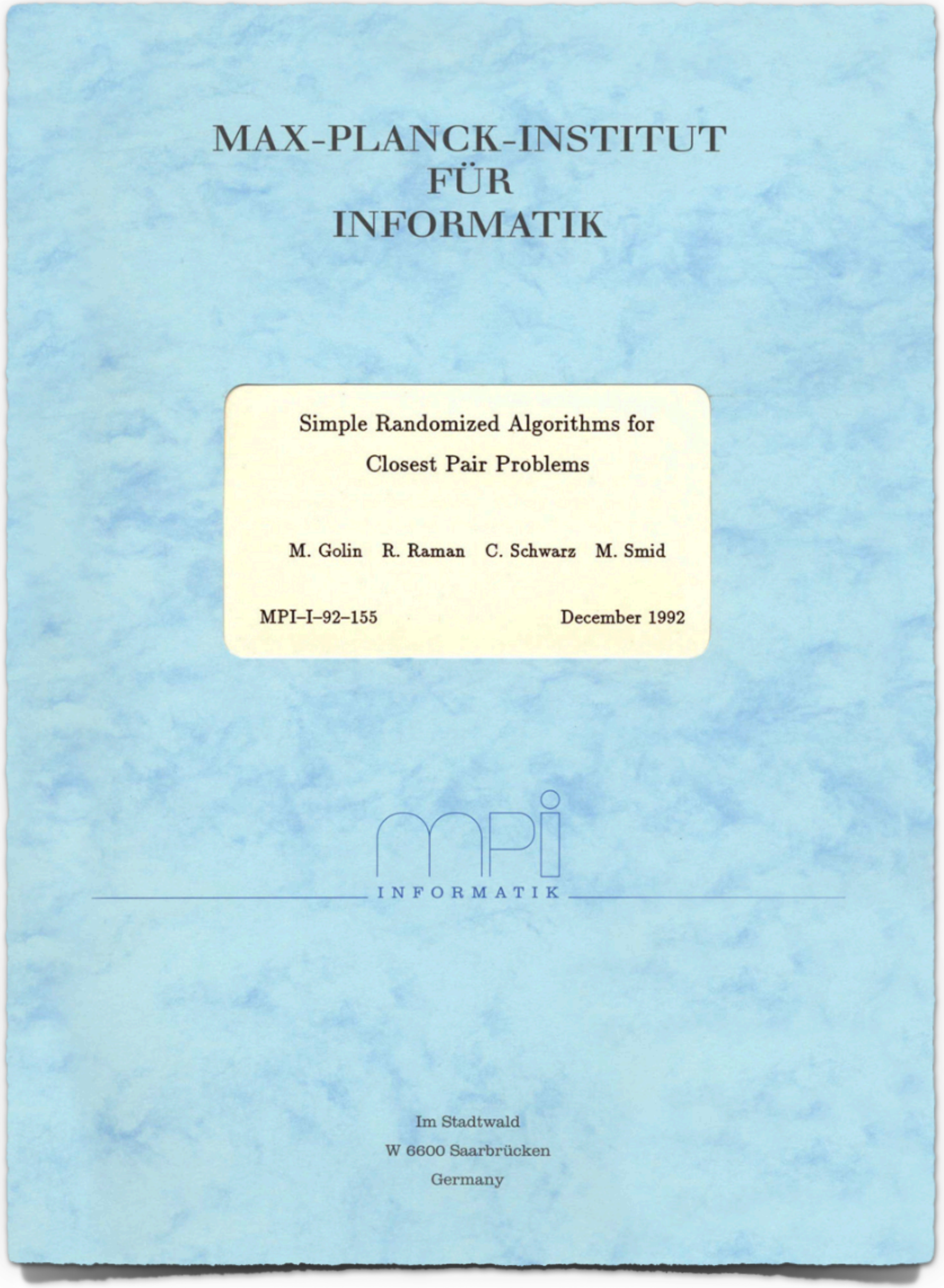
Peter Kramer | November 12th, 2024

Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995



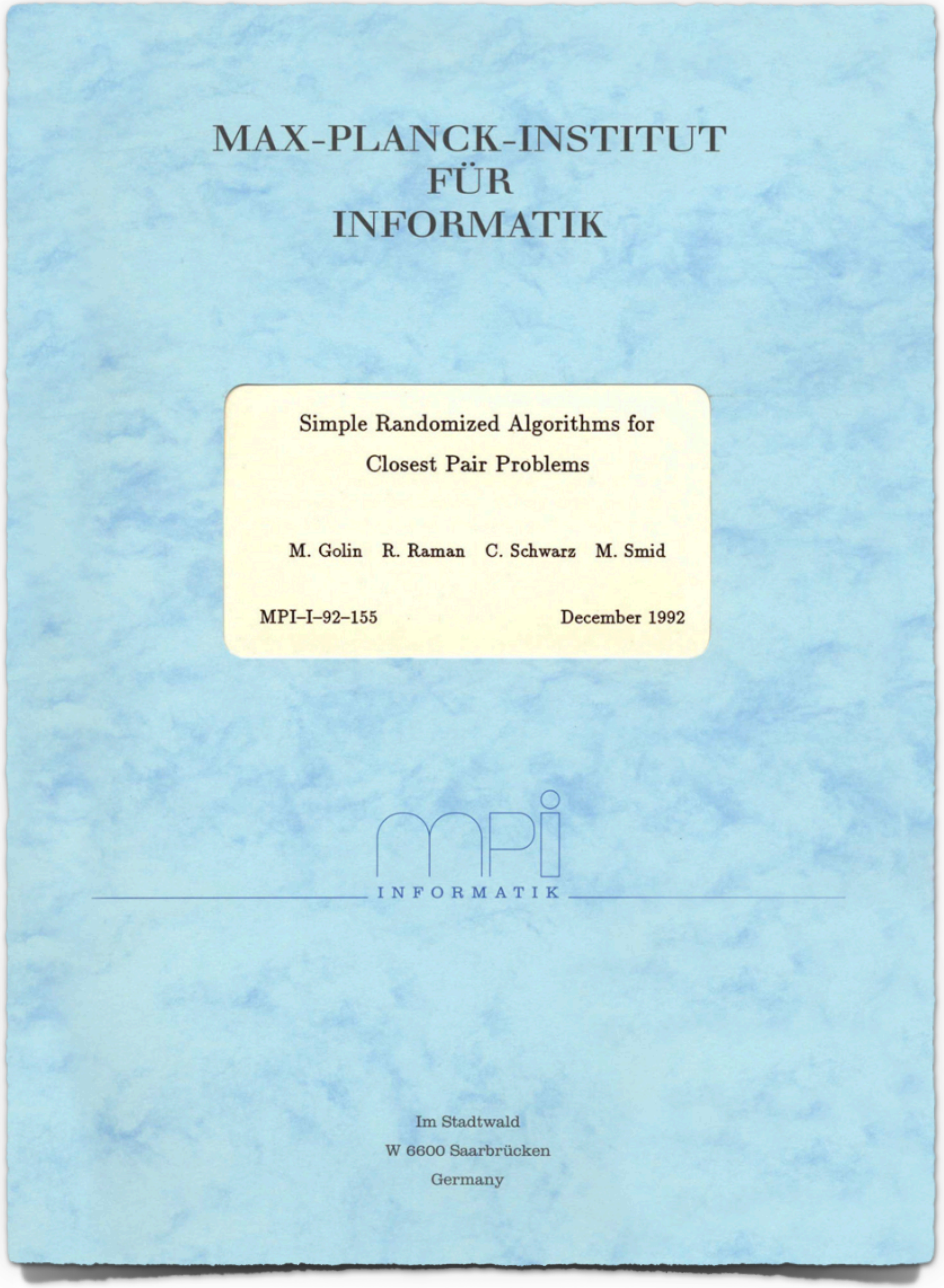
Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

- **Idea:** Incrementally build a solution from a partial solution by refining it:



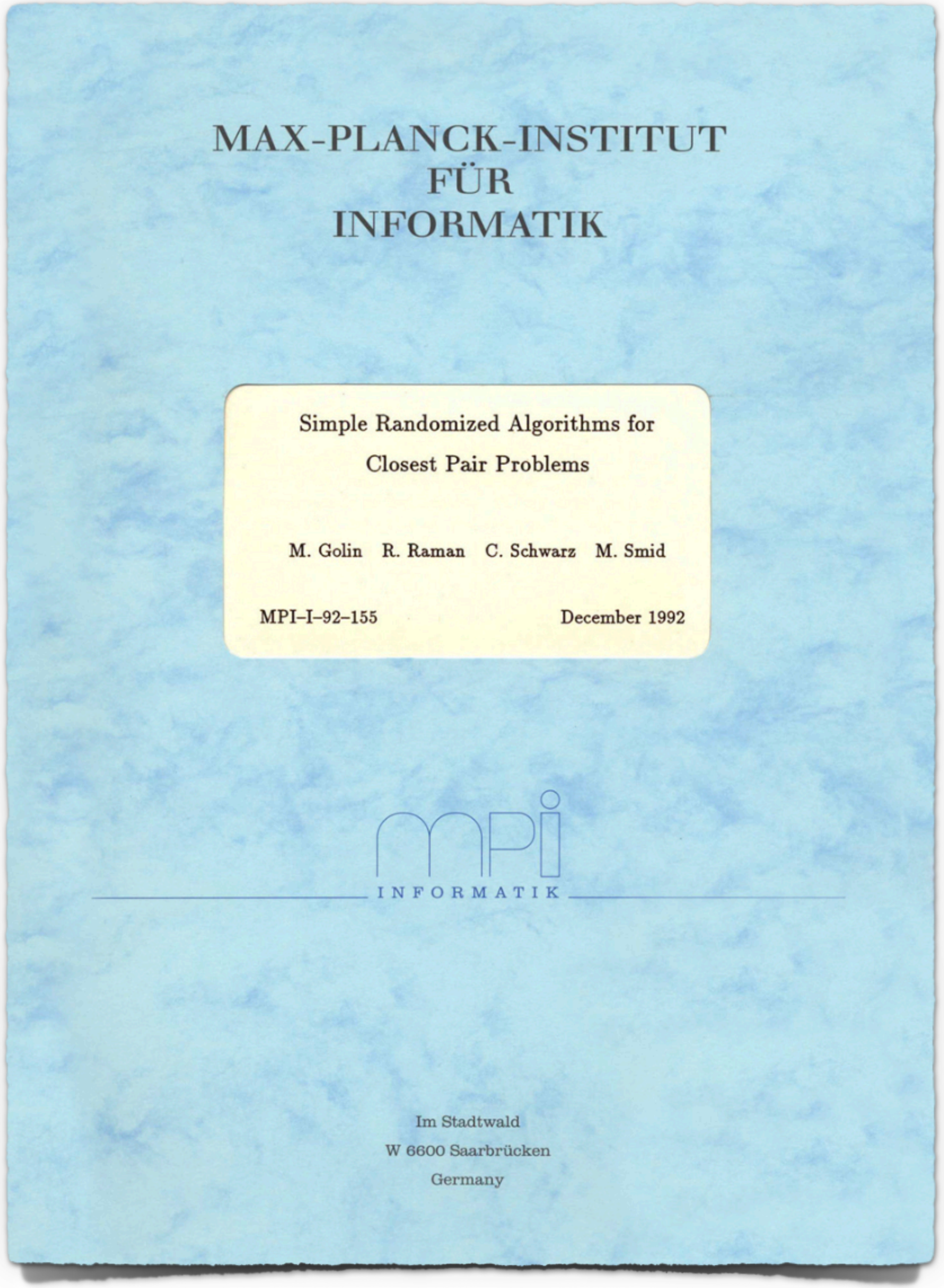
Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

- **Idea:** Incrementally build a solution from a partial solution by refining it:
  - Track current minimal δ.



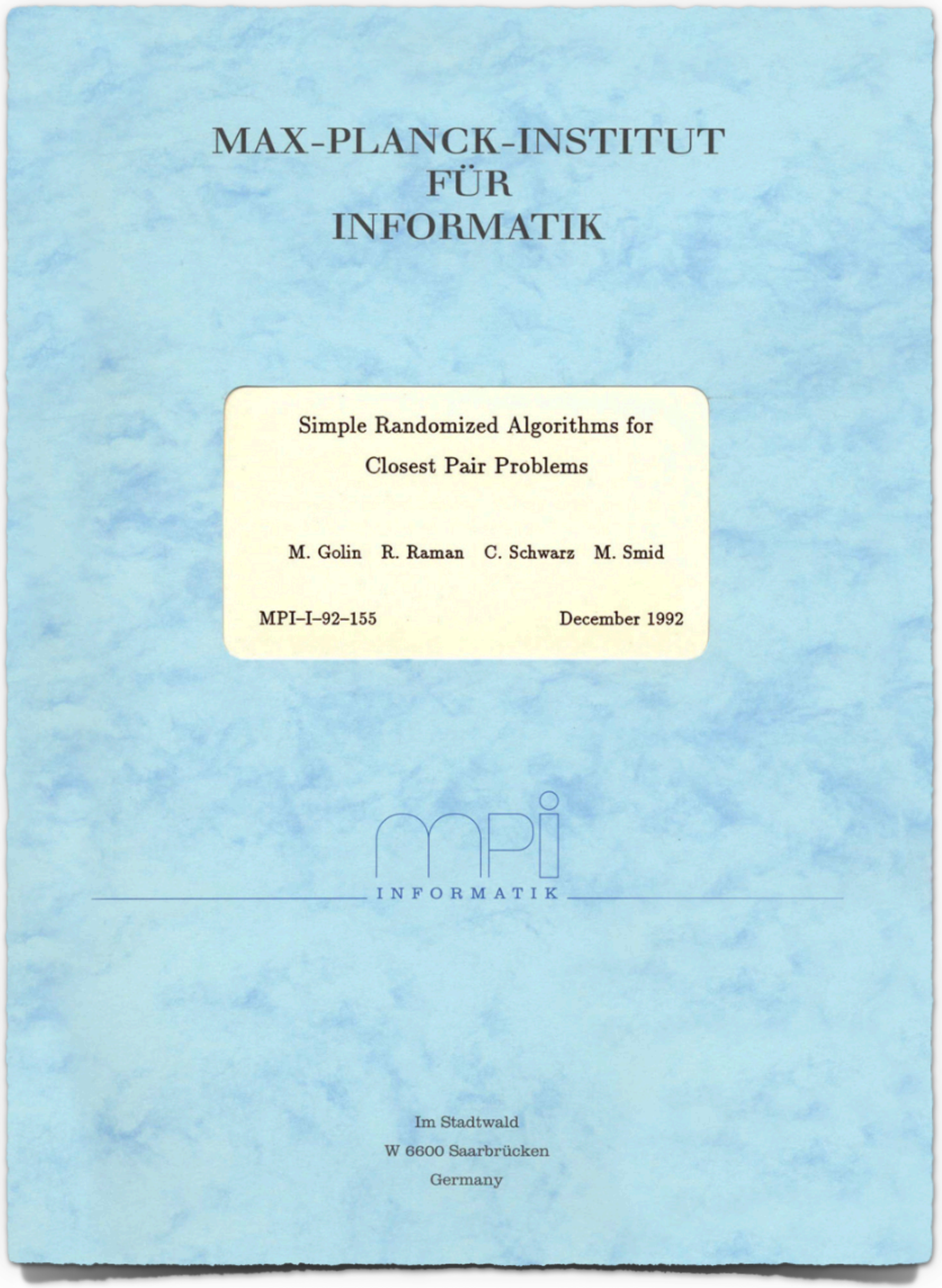
Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

- **Idea:** Incrementally build a solution from a partial solution by refining it:
  - Track current minimal δ.
  - Include next $p \in \mathscr{P}$, update δ, repeat.



Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

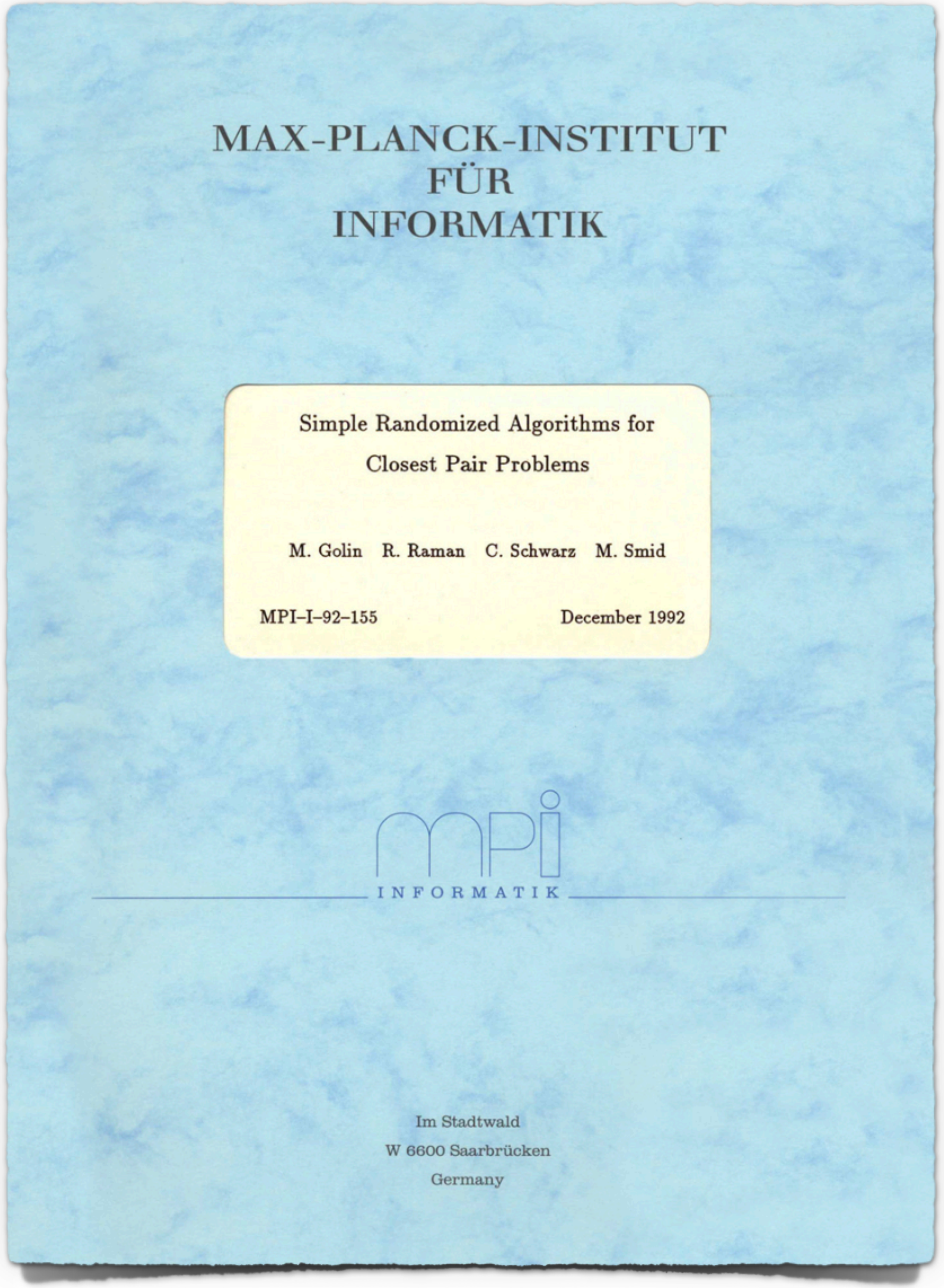
- **Idea:** Incrementally build a solution from a partial solution by refining it:
  - Track current minimal δ.
  - Include next $p \in \mathscr{P}$, update δ, repeat.
  - Crucial: Maintain sparsity, meaning just $\mathcal{O}(1)$ candidates to check!



Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

- **Idea:** Incrementally build a solution from a partial solution by refining it:
  - Track current minimal δ.
  - Include next $p \in \mathscr{P}$, update δ, repeat.
  - Crucial: Maintain sparsity, meaning just $\mathcal{O}(1)$ candidates to check!

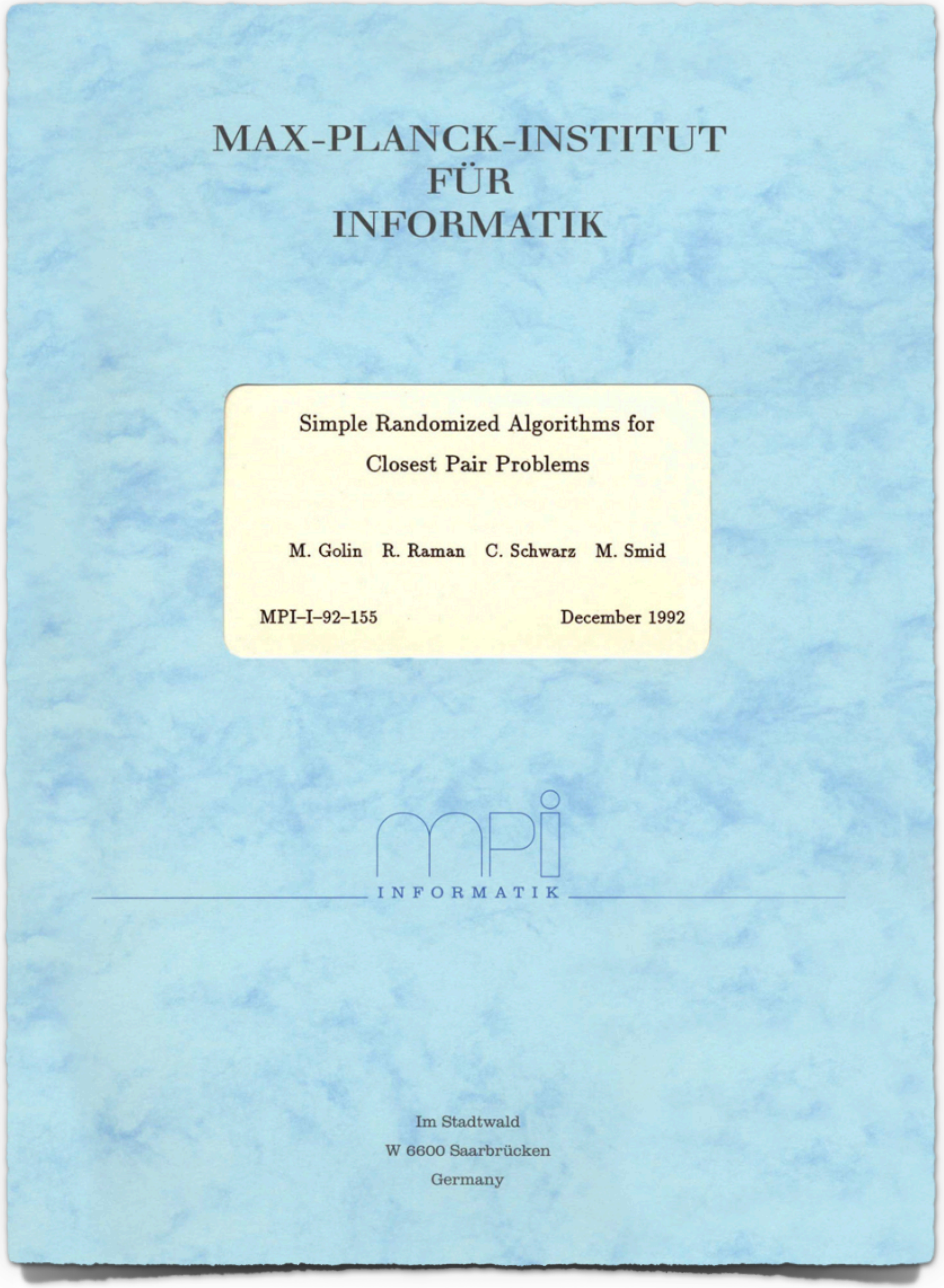
- **Issue:** How do we identify candidates each step — efficiently!?



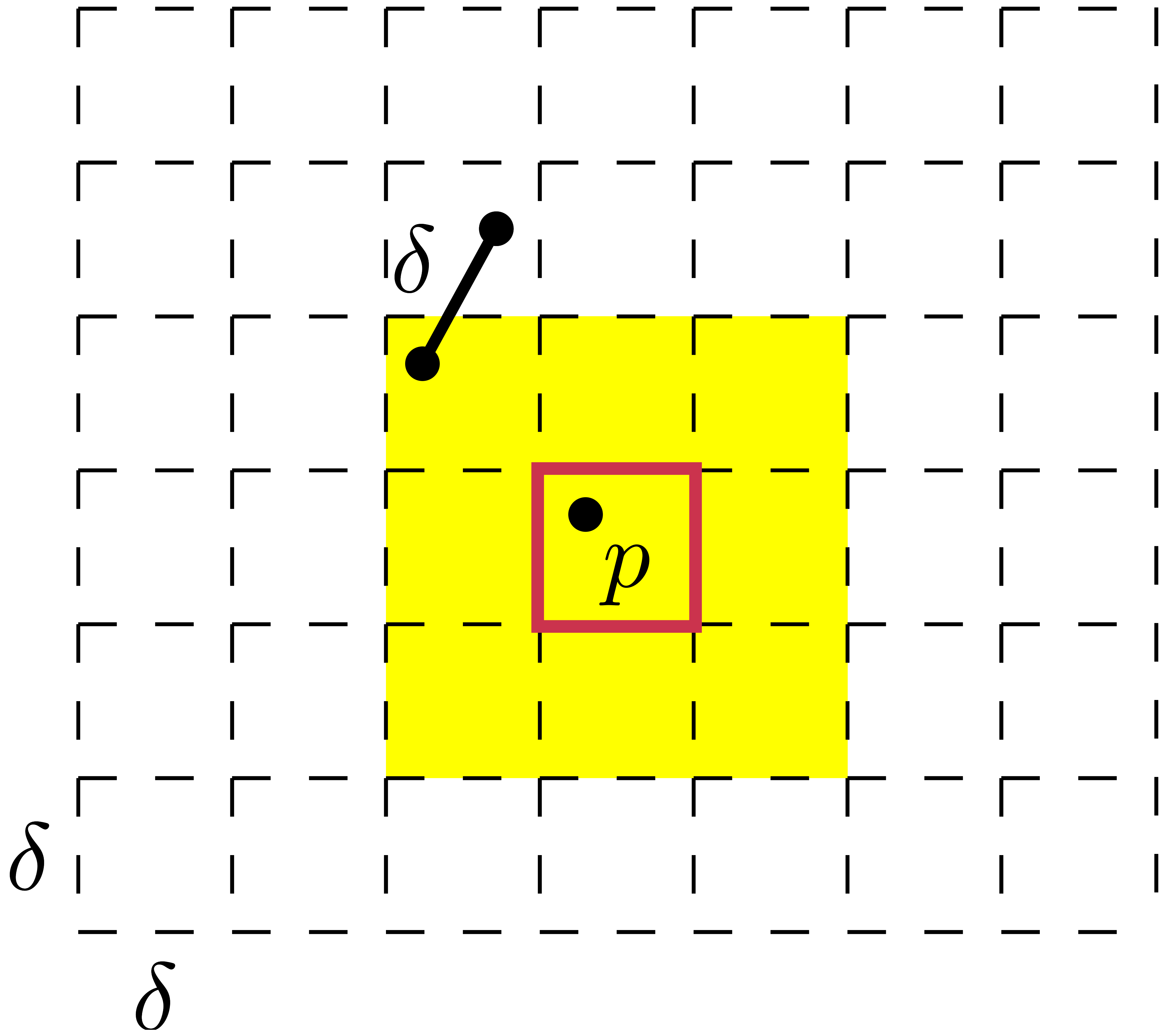
Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

- **Idea:** Incrementally build a solution from a partial solution by refining it:
  - Track current minimal δ.
  - Include next $p \in \mathscr{P}$, update δ, repeat.
  - Crucial: Maintain sparsity, meaning just $\mathcal{O}(1)$ candidates to check!

- **Issue:** How do we identify candidates each step — efficiently!?
- Unsorted sequences $\Rightarrow$ No structure.

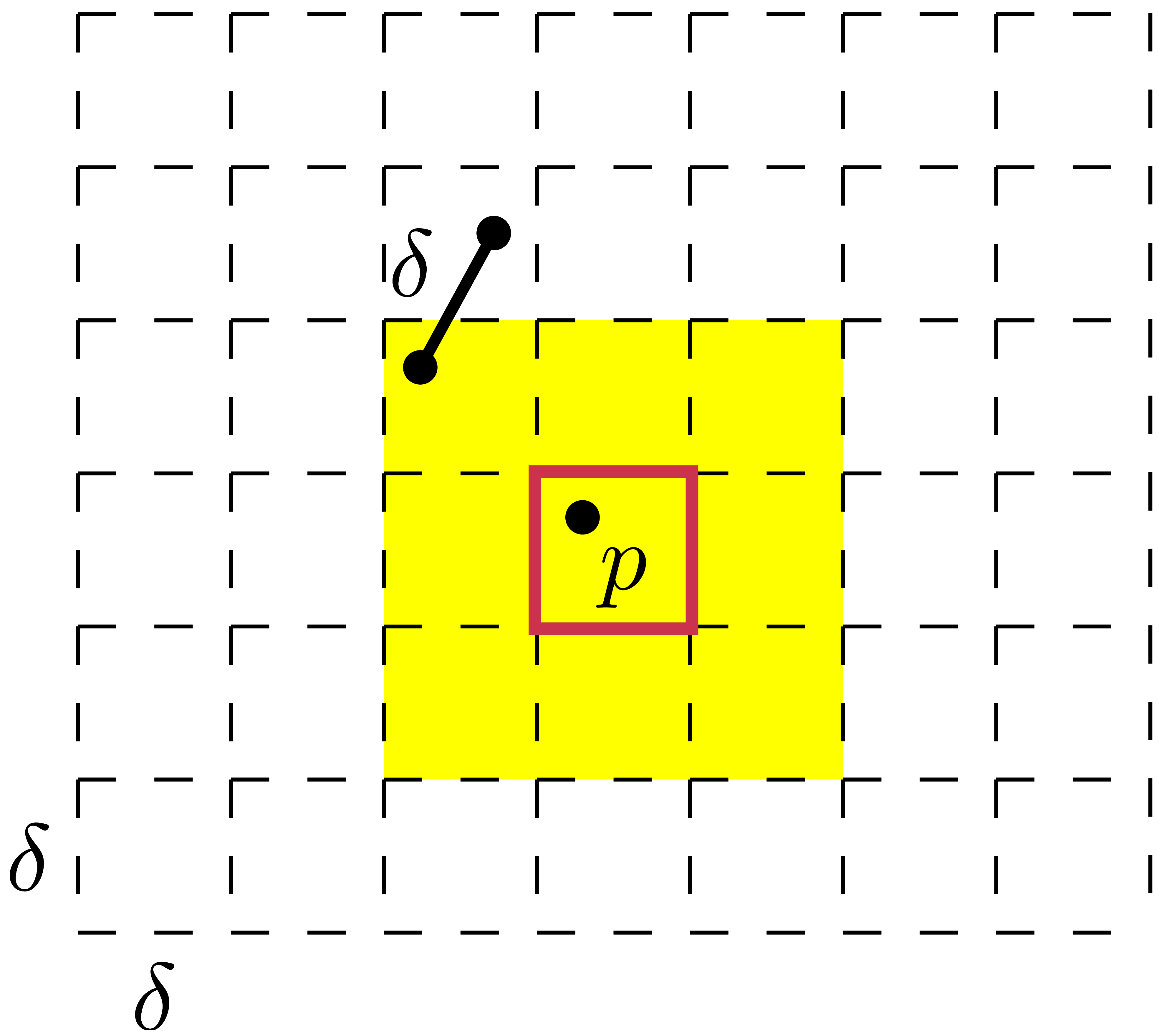


Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995



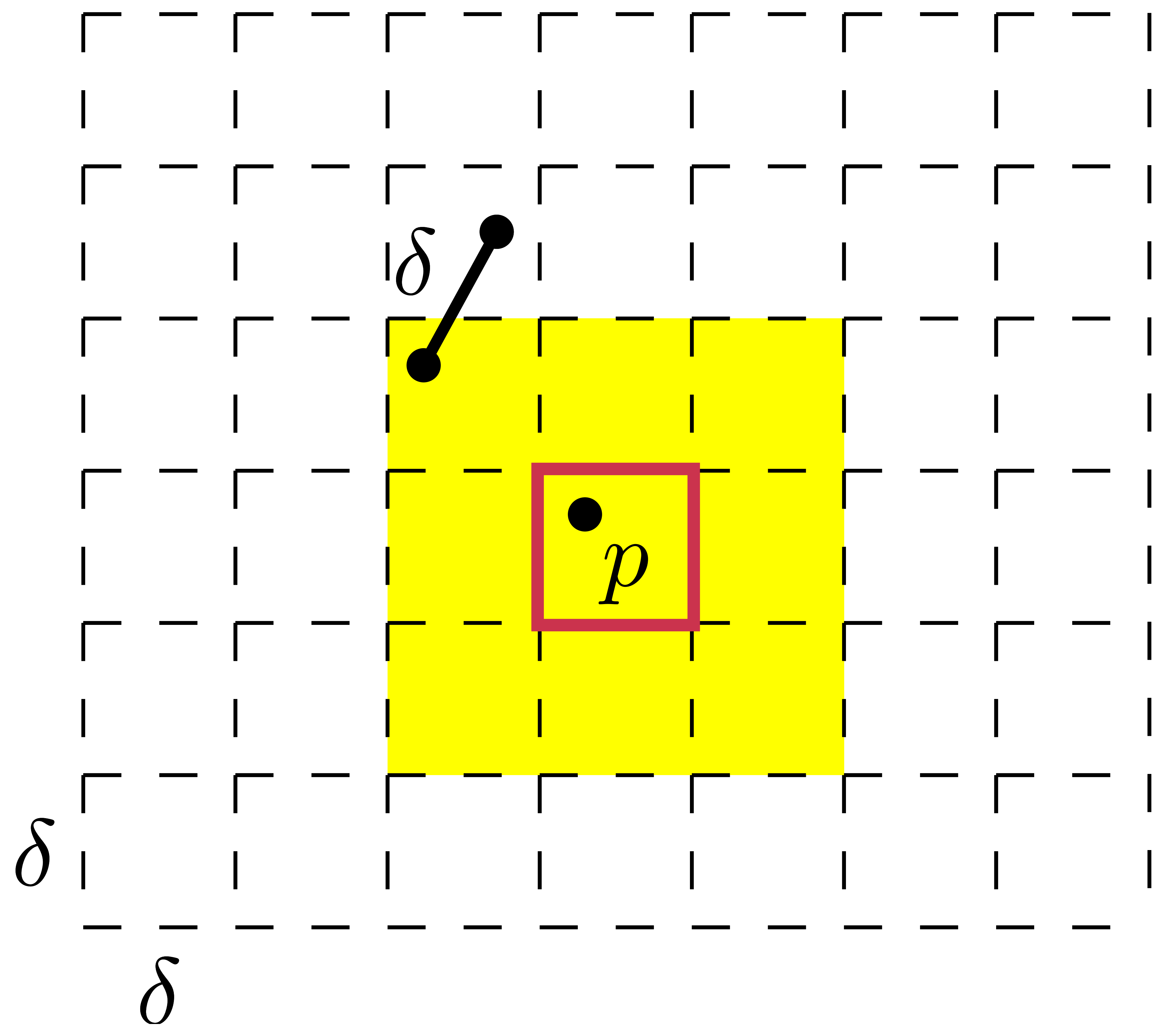
Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

- Cover $\mathbb{R}^2$ with an infinite $\delta \times \delta$ grid G_δ.
- Cells are identified by column x, row y: $G_\delta[x, y]$.
- W.l.o.g., the cell $G_\delta[0,0]$ is located at $(0,0)$.



Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

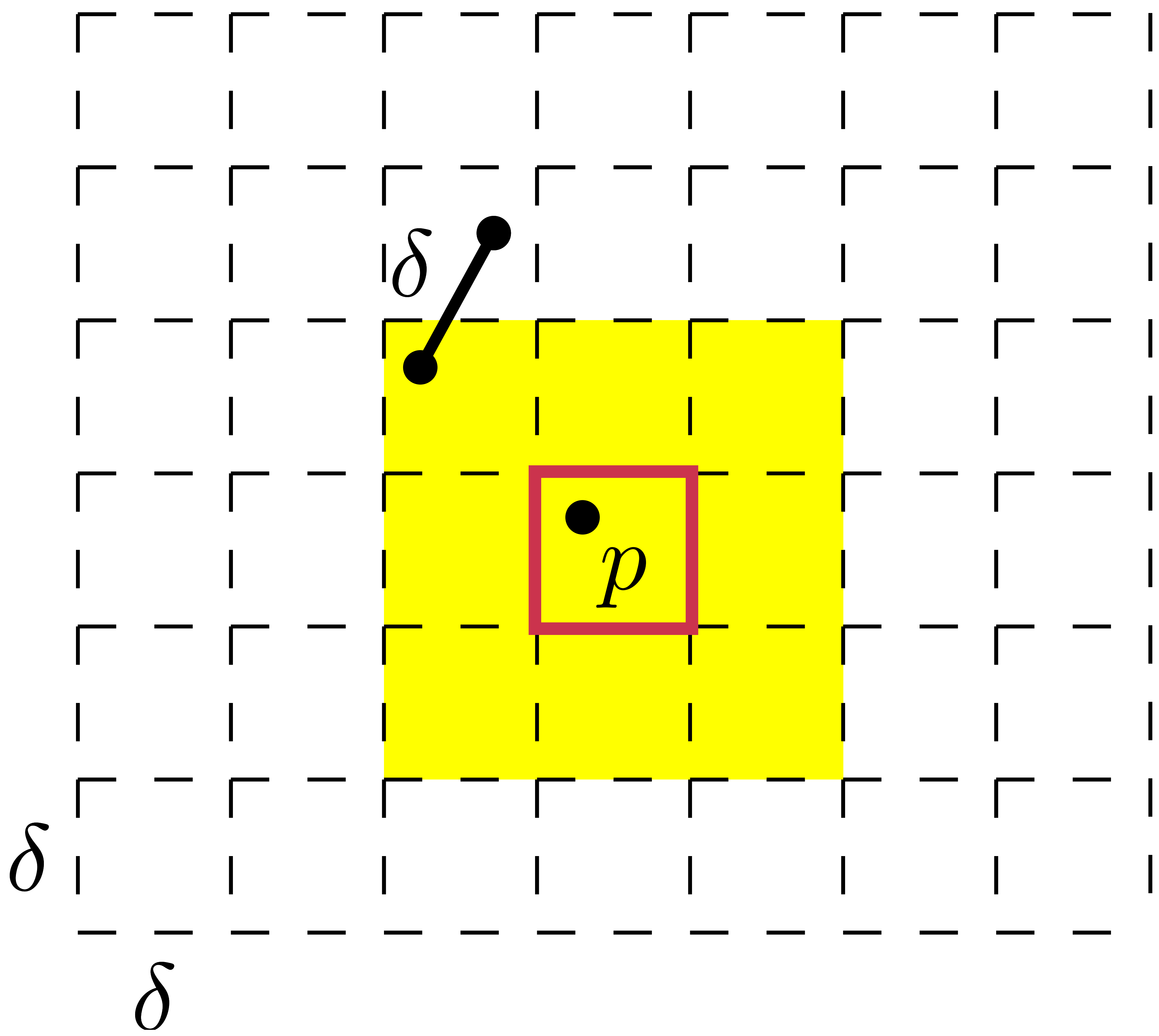
- Cover $\mathbb{R}^2$ with an infinite $\delta \times \delta$ grid G_δ.
- Cells are identified by column x, row y: $G_\delta[x, y]$.
- W.l.o.g., the cell $G_\delta[0,0]$ is located at $(0,0)$.
- Each (half-open) grid cell $G_\delta[x, y]$ stores all previously considered points of $\mathscr{P}$ inside it.



Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

- Cover $\mathbb{R}^2$ with an infinite $\delta \times \delta$ grid G_δ.
- Cells are identified by column x, row y: $G_\delta[x, y]$.
- W.l.o.g., the cell $G_\delta[0,0]$ is located at $(0,0)$.
- Each (half-open) grid cell $G_\delta[x, y]$ stores all previously considered points of $\mathscr{P}$ inside it.
- The neighbourhood of $G_\delta[x, y]$ refers to:

$$N_\delta(x, y) = \bigcup_{u,v \in [-1,1]} G_\delta[x - u, y - v].$$



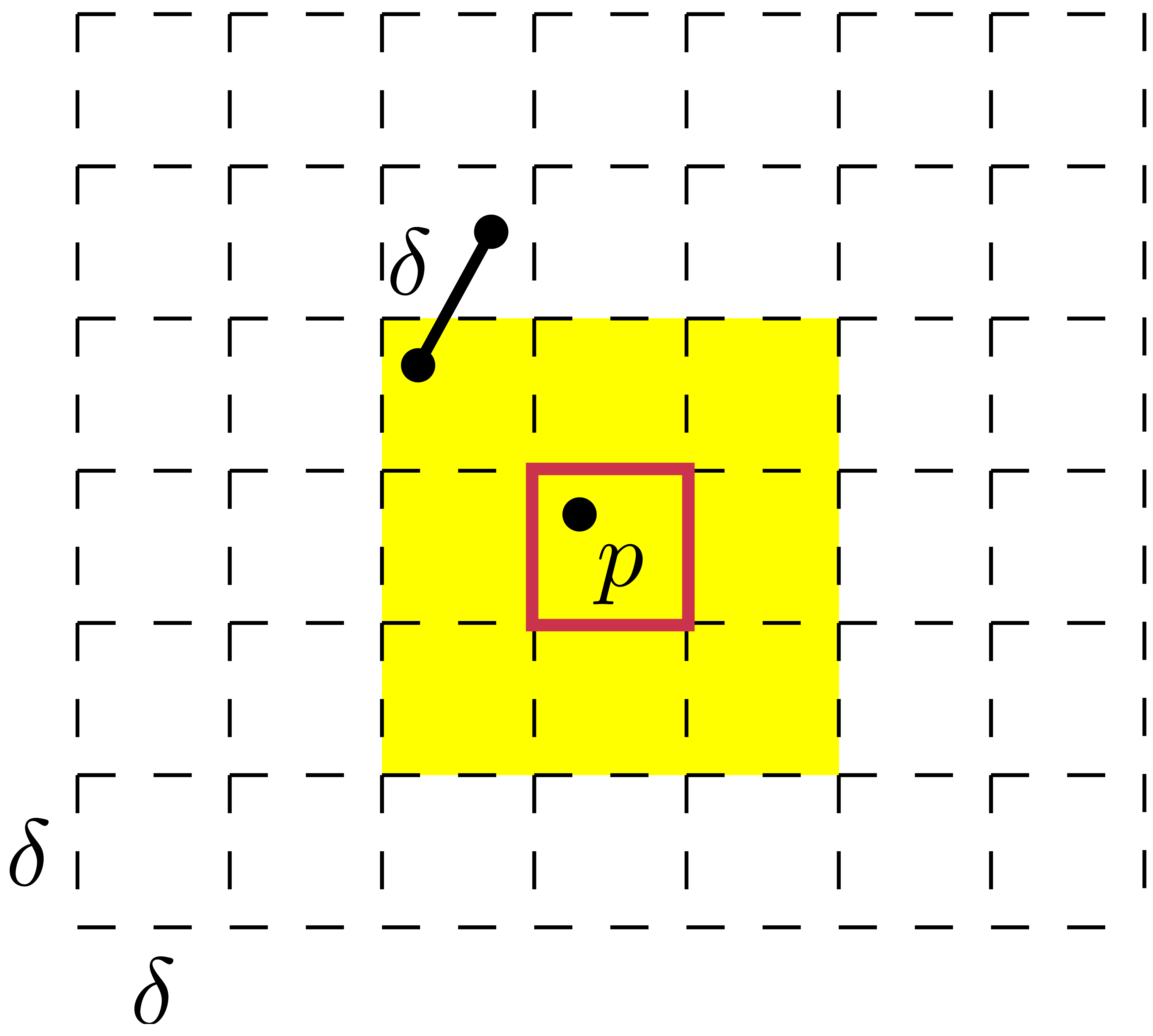
Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

- Cover $\mathbb{R}^2$ with an infinite $\delta \times \delta$ grid G_δ.
- Cells are identified by column x, row y: $G_\delta[x, y]$.
- W.l.o.g., the cell $G_\delta[0,0]$ is located at $(0,0)$.
- Each (half-open) grid cell $G_\delta[x, y]$ stores all previously considered points of $\mathscr{P}$ inside it.
- The neighbourhood of $G_\delta[x, y]$ refers to:

$$N_\delta(x, y) = \bigcup_{u,v \in [-1,1]} G_\delta[x-u, y-v].$$

Lemma. For $p, q \in \mathscr{P}$ such that $p \in G_\delta[x, y]$:

$$d(p, q) \le \delta \implies q \in N_\delta(x, y)$$



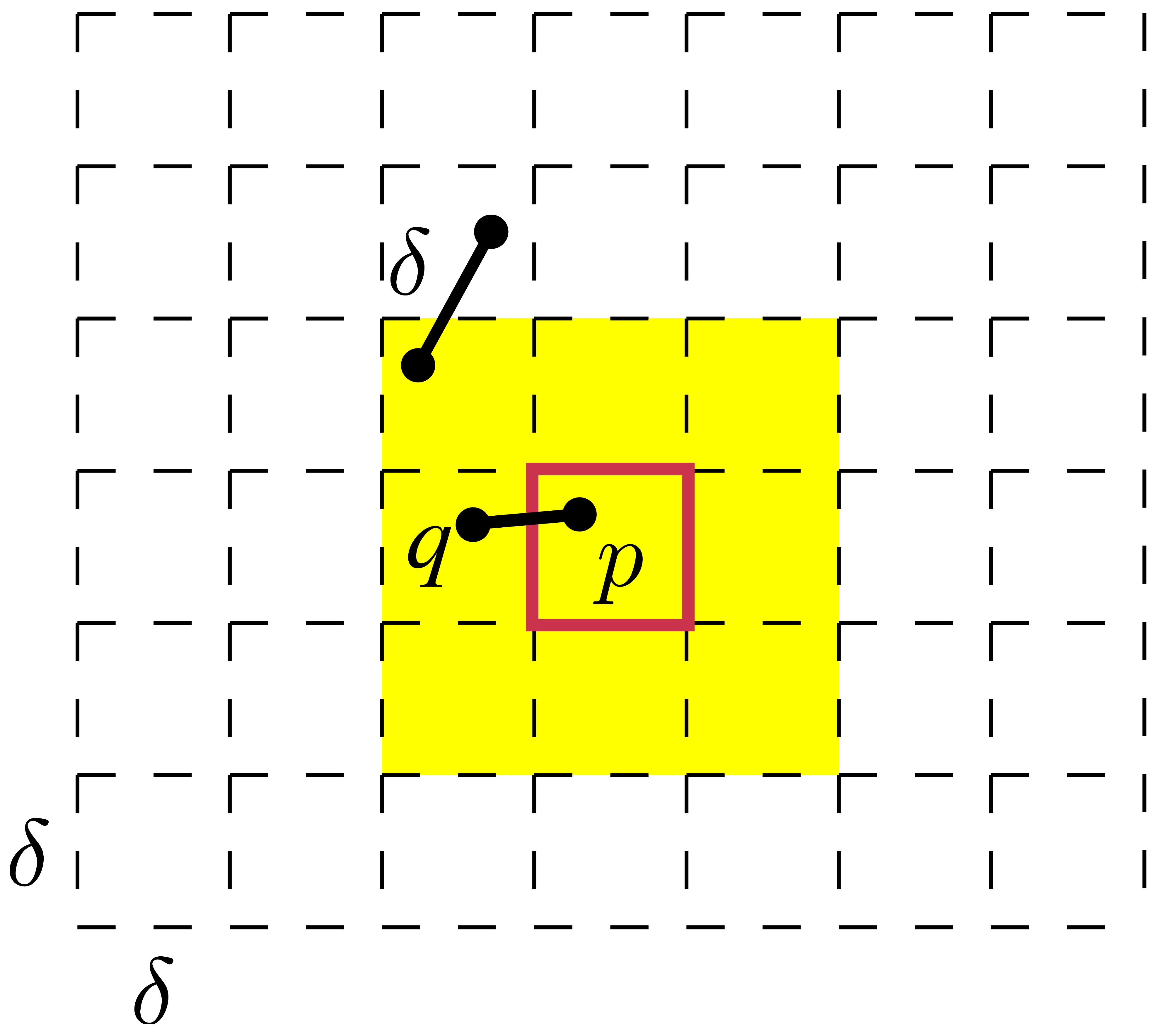
Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

- Cover $\mathbb{R}^2$ with an infinite $\delta \times \delta$ grid G_δ.
- Cells are identified by column x, row y: $G_\delta[x, y]$.
- W.l.o.g., the cell $G_\delta[0,0]$ is located at $(0,0)$.
- Each (half-open) grid cell $G_\delta[x, y]$ stores all previously considered points of $\mathscr{P}$ inside it.
- The neighbourhood of $G_\delta[x, y]$ refers to:

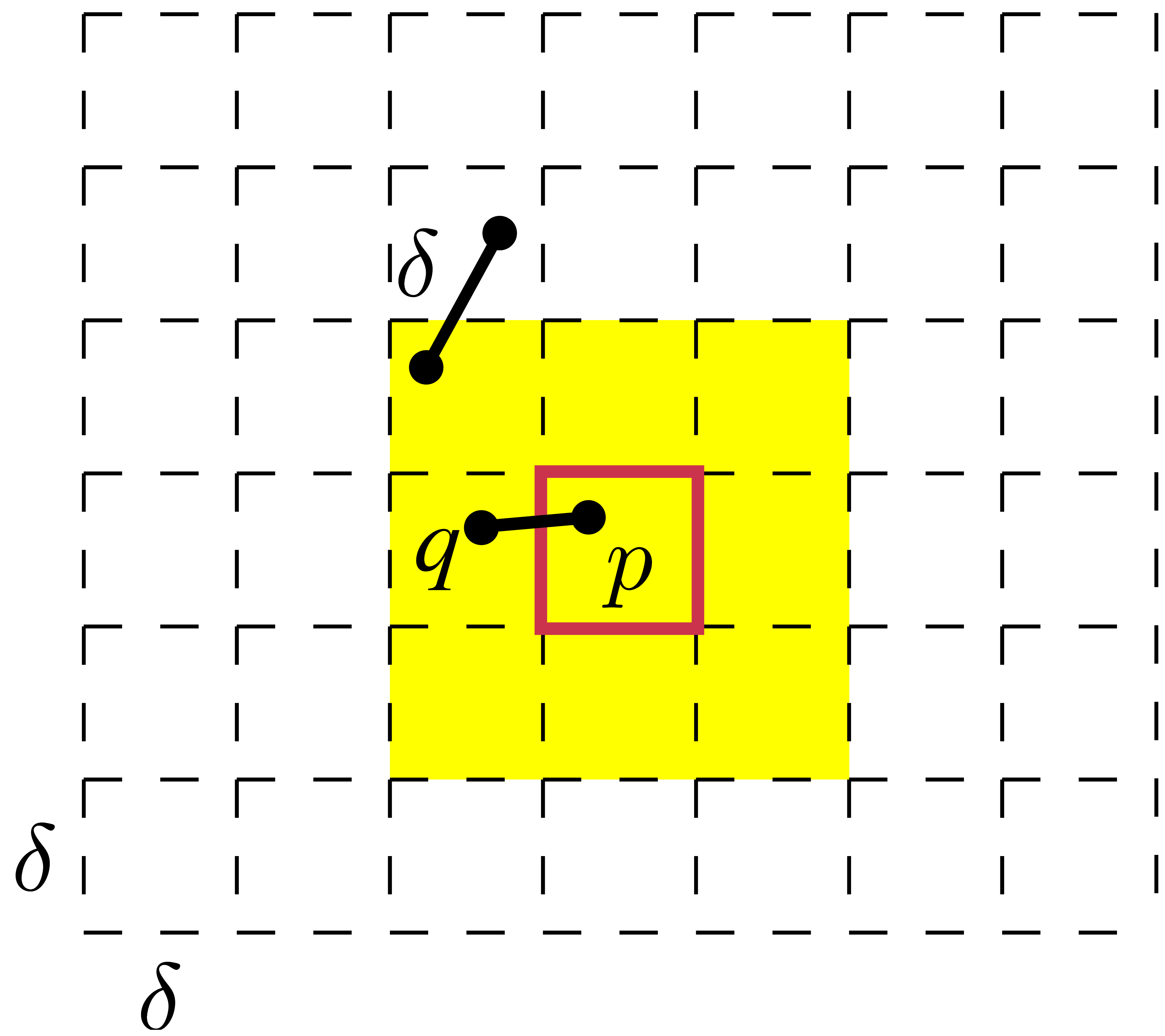
$$N_\delta(x, y) = \bigcup_{u,v \in [-1,1]} G_\delta[x-u, y-v].$$

Lemma. For $p, q \in \mathscr{P}$ such that $p \in G_\delta[x, y]$:

$$d(p, q) \le \delta \implies q \in N_\delta(x, y)$$

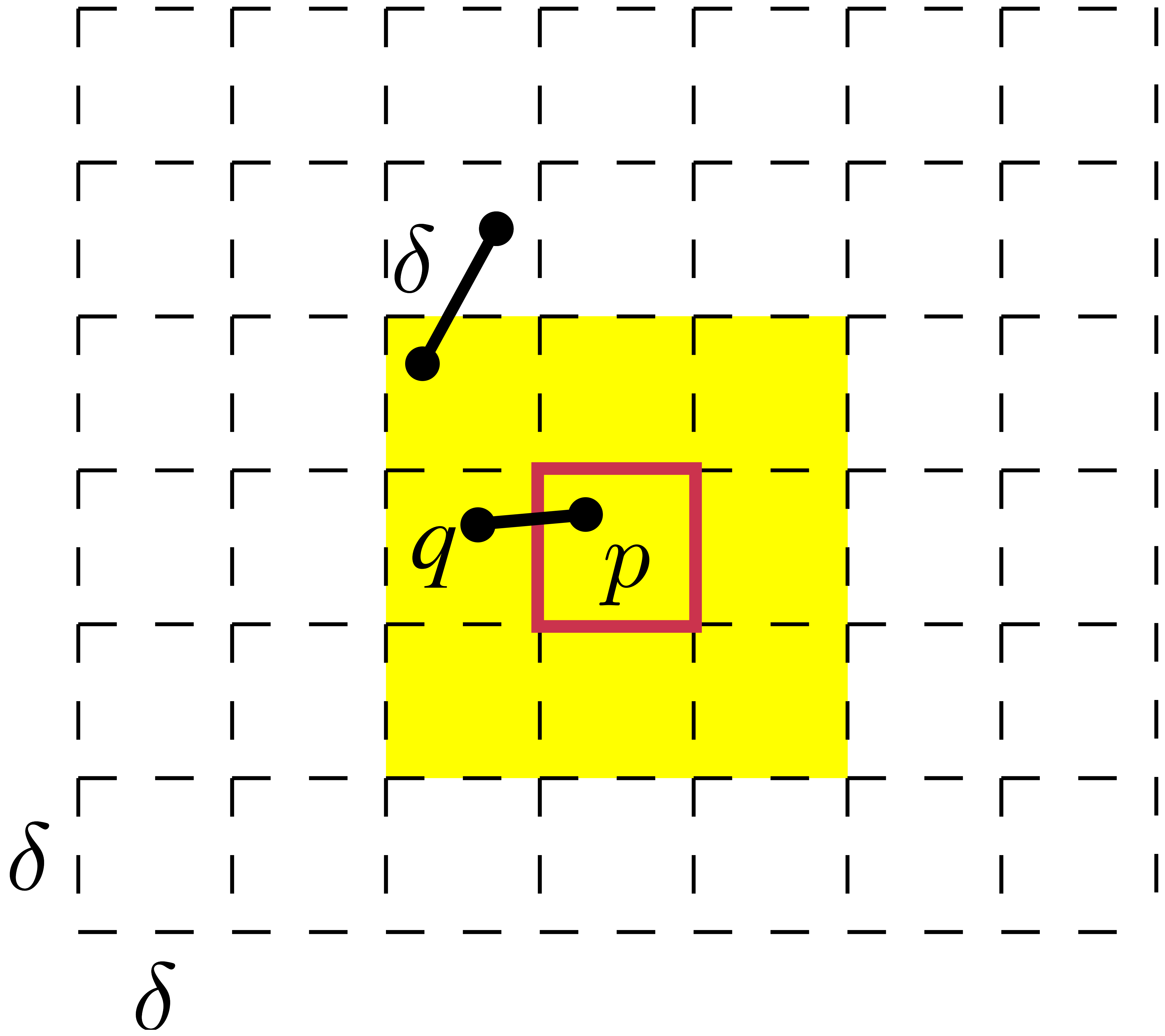


Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995



Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

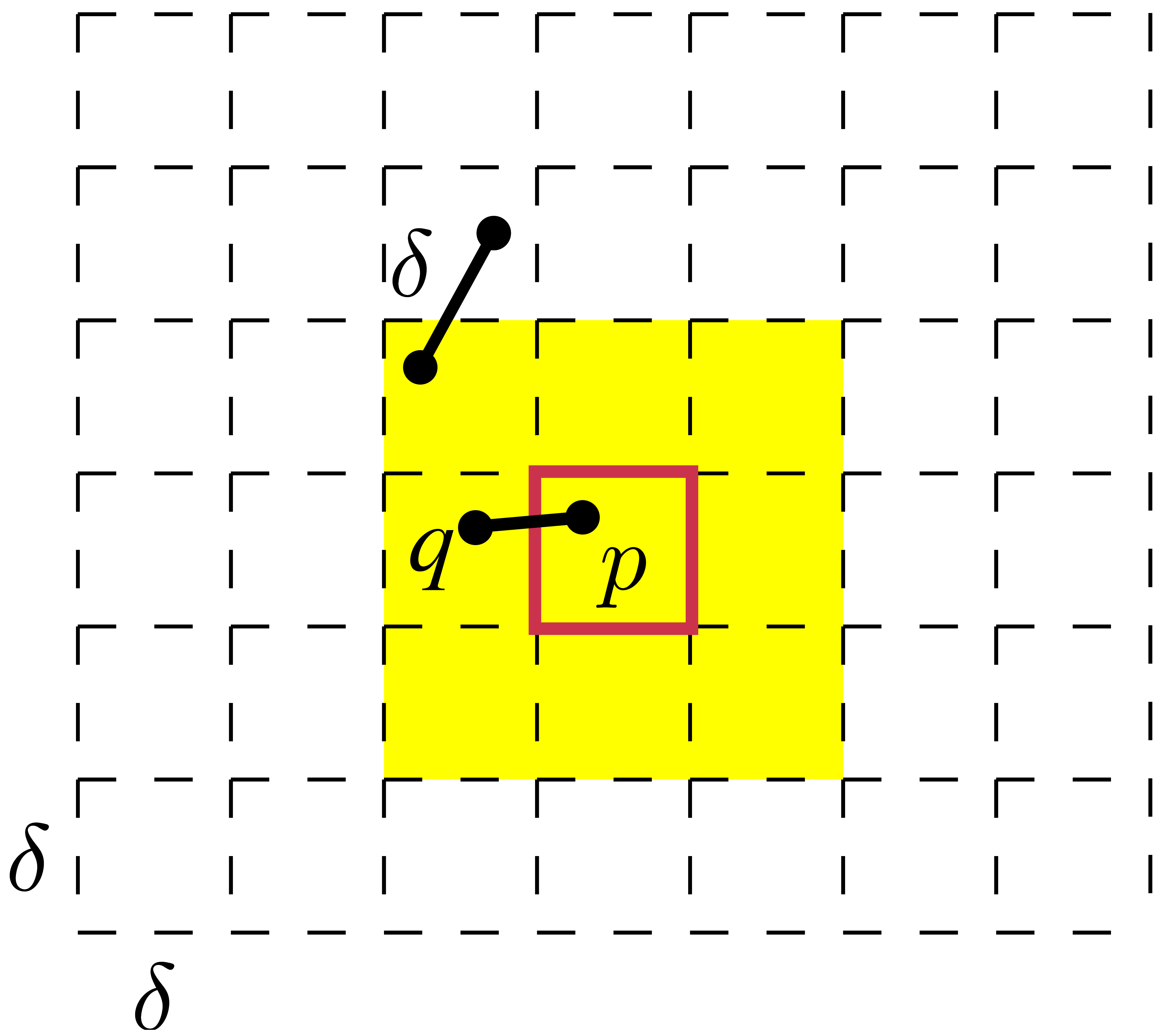
- **Idea:** Sparsity! Keep $|N_\delta(x,y)| \in \mathcal{O}(1)$.
- If each $N_\delta(x,y)$ contains constantly many points of pairwise distance at least δ, …
- … then $\mathcal{O}(1)$ comparisons for next $p \in \mathcal{P}$.



Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

- **Idea:** Sparsity! Keep $|N_\delta(x,y)| \in \mathcal{O}(1)$.
- If each $N_\delta(x,y)$ contains constantly many points of pairwise distance at least δ, …
- … then $\mathcal{O}(1)$ comparisons for next $p \in \mathscr{P}$.

```
For each p ∈ 𝒫:
    Find x, y such that G_δ[x, y] contains p.
    Insert p to G_δ[x, y].
    If p is part of new closest pair {p, q}:
        Update δ = d(p, q).
        Rebuild sparse grid, G_δ'[x, y].
```

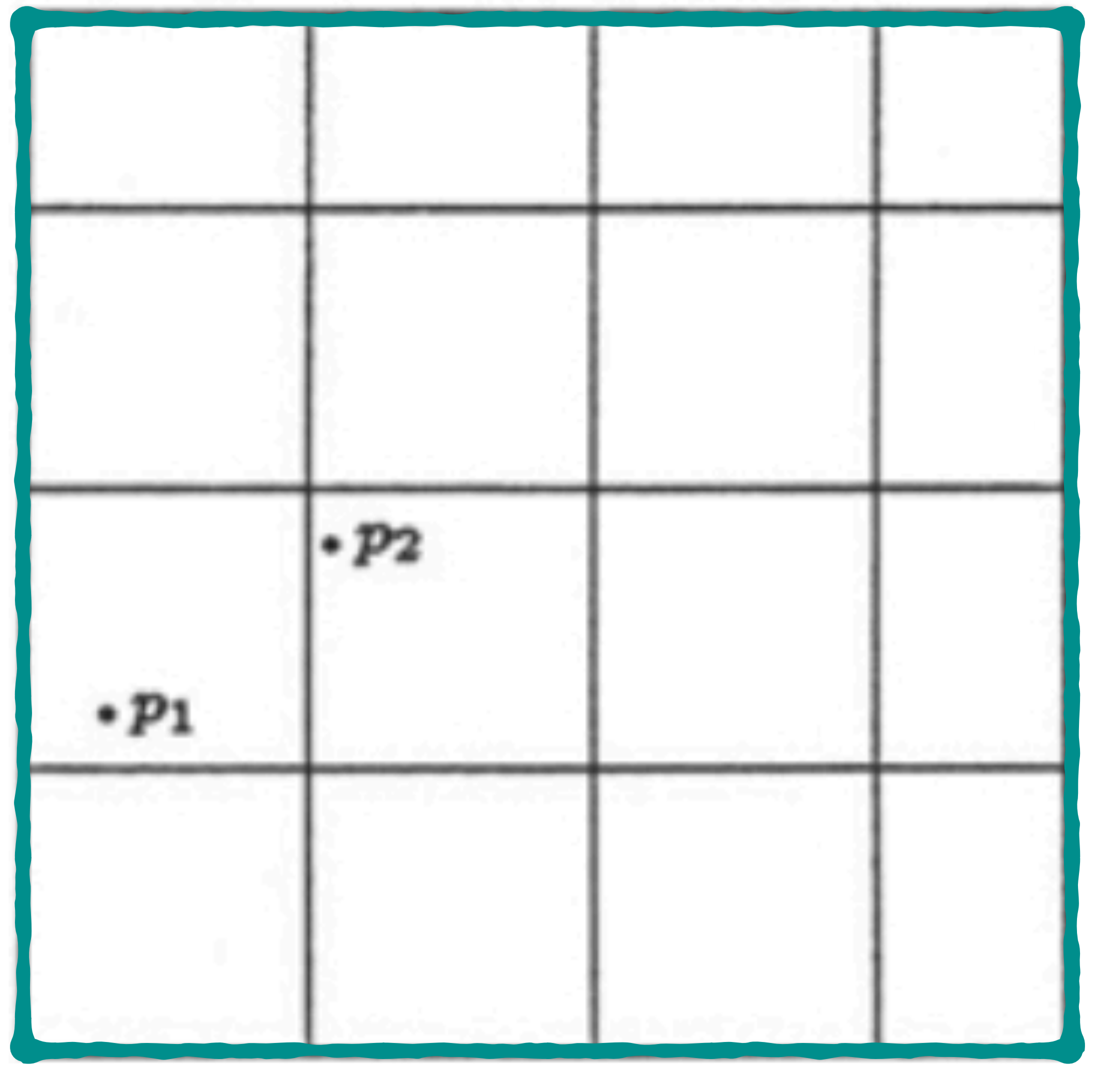


Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

```
Algorithm   CP(p_1, p_2, ..., p_n)

δ := d(p_1, p_2); G := Build(S_2, δ);
for i := 2 to n − 1 do
    begin
    V := {Report(G, b) : b is a neighbor of the box containing p_{i+1}};
    d := min_{q∈V} d(p_{i+1}, q);
    if d ≥ δ then Insert(G, p_{i+1})
             else δ := d; G := Build(S_{i+1}, δ);
    end;
return(δ).
```

$\delta(S_2) = d(p_1, p_2)$

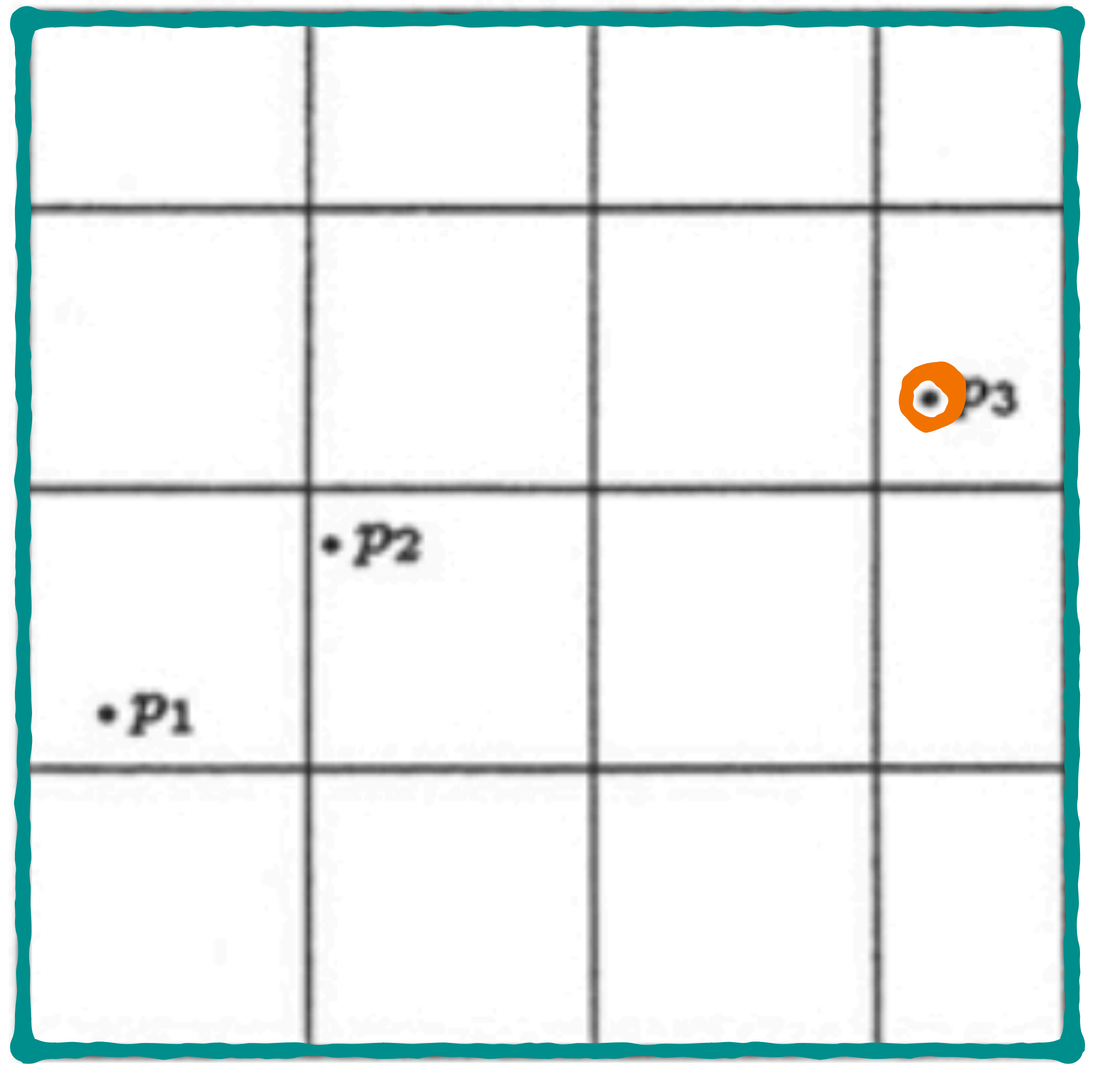


Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Algorithm $CP(p_1, p_2, \ldots, p_n)$

```
δ := d(p_1, p_2); G := Build(S_2, δ);
for i := 2 to n − 1 do
    begin
    V := {Report(G, b) : b is a neighbor of the box containing p_{i+1}};
    d := min_{q∈V} d(p_{i+1}, q);
    if d ≥ δ then Insert(G, p_{i+1})
              else δ := d; G := Build(S_{i+1}, δ);
    end;
return(δ).
```

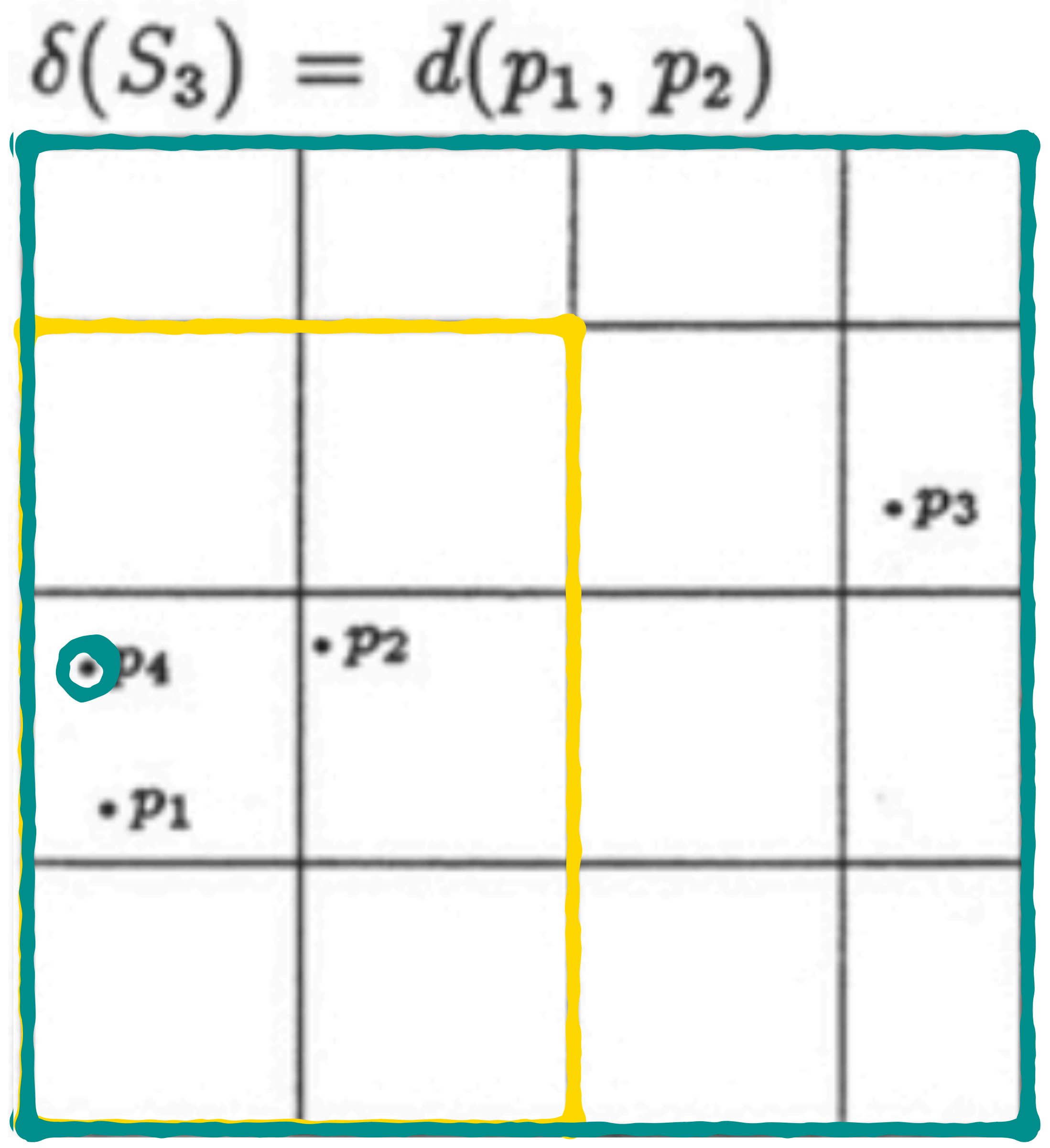
$\delta(S_2) = d(p_1, p_2)$



Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Algorithm $CP(p_1, p_2, \ldots, p_n)$

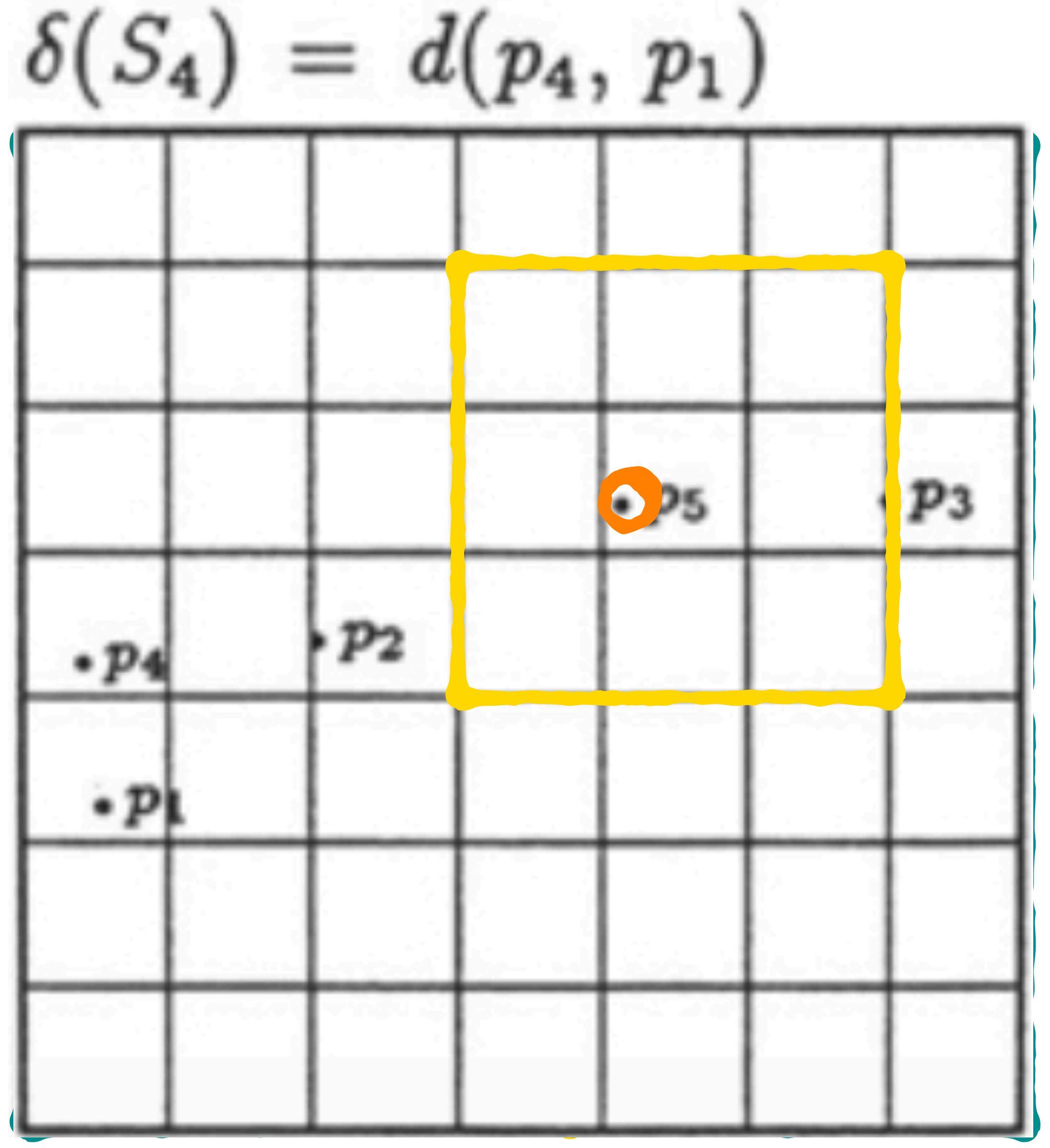
$\delta := d(p_1, p_2)$; $\mathcal{G} := Build(S_2, \delta)$;
for $i := 2$ **to** $n-1$ **do**
  begin
    $V := \{Report(\mathcal{G}, b) : b$ is a neighbor of the box containing $p_{i+1}\}$;
    $d := \min_{q \in V} d(p_{i+1}, q)$;
    if $d \geq \delta$ **then** $Insert(\mathcal{G}, p_{i+1})$
      else $\delta := d$; $\mathcal{G} := Build(S_{i+1}, \delta)$;
  end;
return(δ).



Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Algorithm $CP(p_1, p_2, \ldots, p_n)$

$\delta := d(p_1, p_2)$; $\mathcal{G} := Build(S_2, \delta)$;
for $i := 2$ **to** $n-1$ **do**
  begin
  $V := \{Report(\mathcal{G}, b)$: b is a neighbor of the box containing $p_{i+1}\}$;
  $d := \min_{q \in V} d(p_{i+1}, q)$;
  if $d \geq \delta$ **then** $Insert(\mathcal{G}, p_{i+1})$
  else $\delta := d$; $\mathcal{G} := Build(S_{i+1}, \delta)$;
  end;
return(δ).

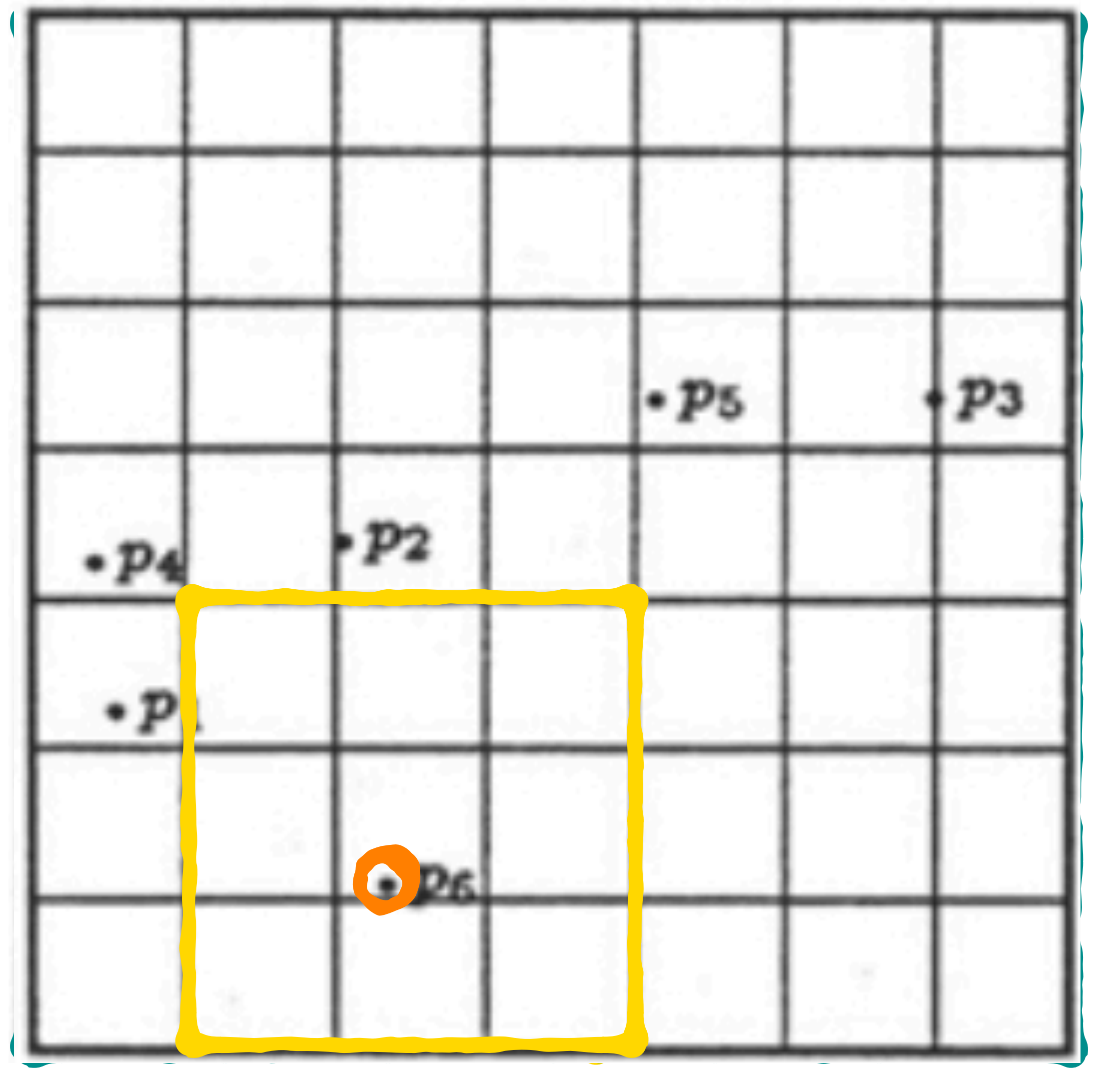


Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Algorithm $CP(p_1, p_2, \ldots, p_n)$

$\delta := d(p_1, p_2)$; $\mathcal{G} := Build(S_2, \delta)$;
for $i := 2$ **to** $n - 1$ **do**
  begin
  $V := \{Report(\mathcal{G}, b) : b$ is a neighbor of the box containing $p_{i+1}\}$;
  $d := \min_{q \in V} d(p_{i+1}, q)$;
  if $d \geq \delta$ **then** $Insert(\mathcal{G}, p_{i+1})$
    else $\delta := d$; $\mathcal{G} := Build(S_{i+1}, \delta)$;
  end;
return(δ).

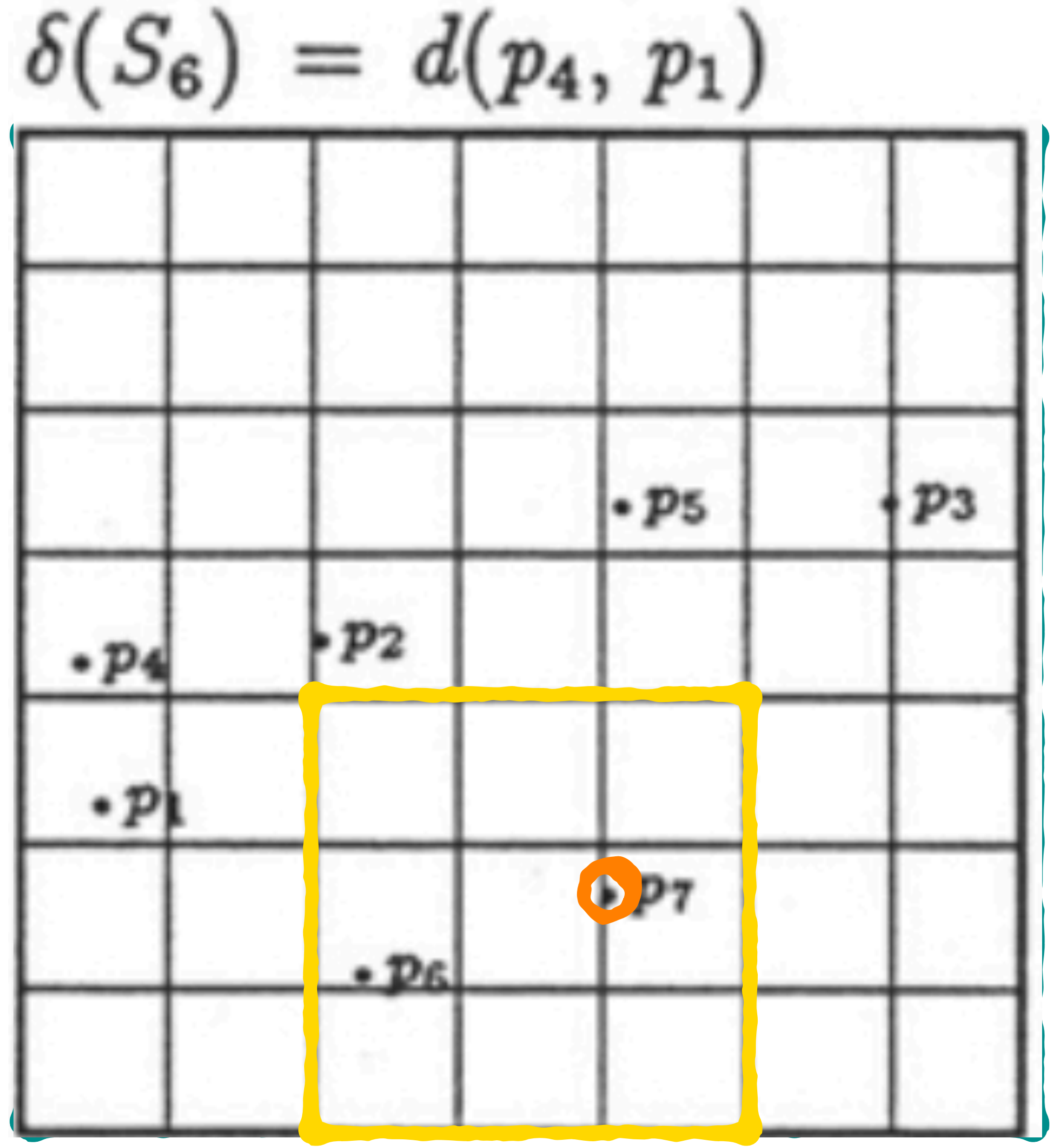
$\delta(S_5) = d(p_4, p_1)$



Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Algorithm $CP(p_1, p_2, \ldots, p_n)$

```
δ := d(p_1, p_2); G := Build(S_2, δ);
for i := 2 to n − 1 do
    begin
    V := {Report(G, b) : b is a neighbor of the box containing p_{i+1}};
    d := min_{q∈V} d(p_{i+1}, q);
    if d ≥ δ then Insert(G, p_{i+1})
              else δ := d; G := Build(S_{i+1}, δ);
    end;
return(δ).
```

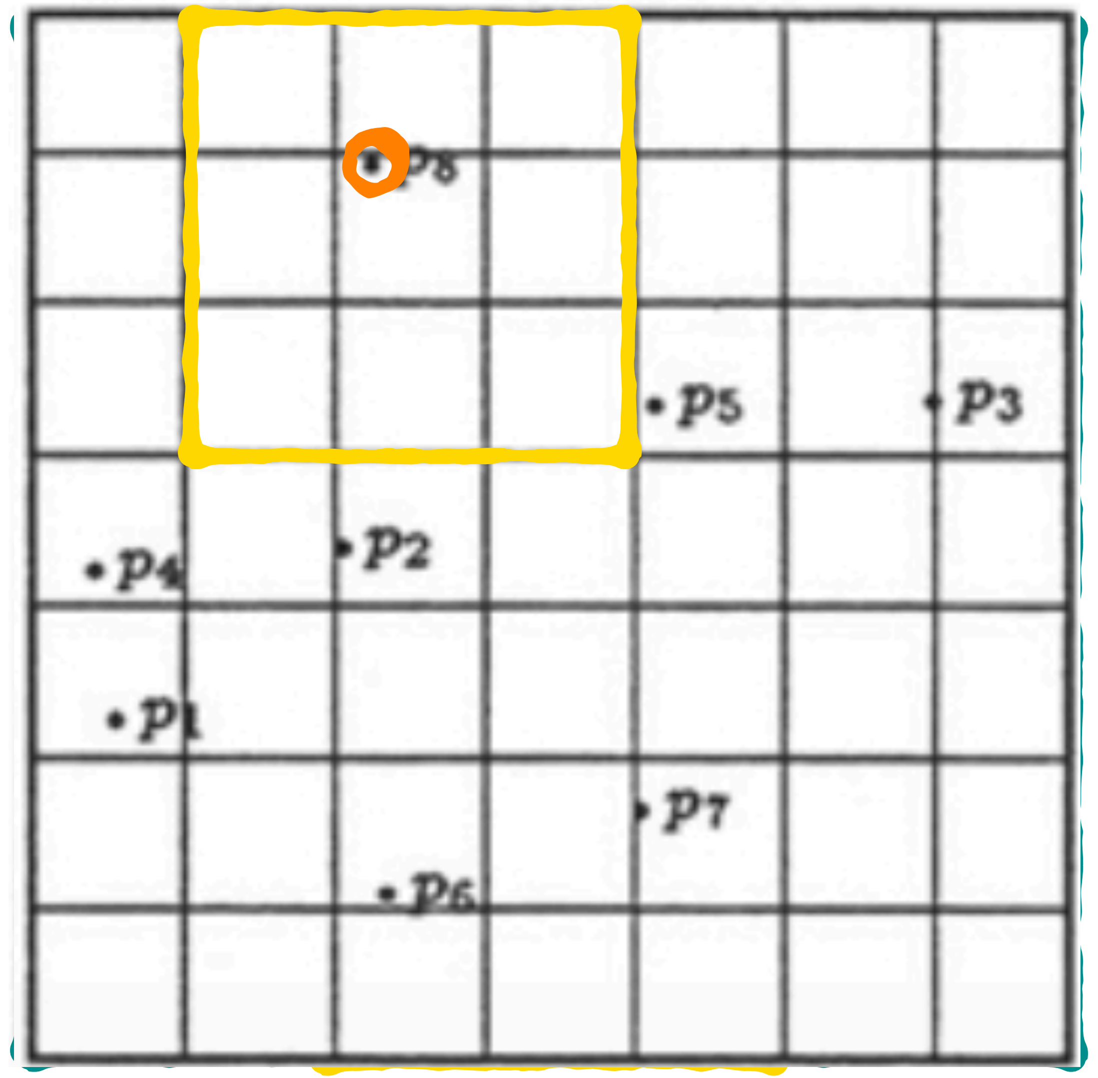


Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Algorithm $CP(p_1, p_2, \ldots, p_n)$

$\delta := d(p_1, p_2)$; $\mathcal{G} := Build(S_2, \delta)$;
for $i := 2$ **to** $n-1$ **do**
  begin
  $V := \{Report(\mathcal{G}, b) : b$ is a neighbor of the box containing $p_{i+1}\}$;
  $d := \min_{q \in V} d(p_{i+1}, q)$;
  if $d \geq \delta$ **then** $Insert(\mathcal{G}, p_{i+1})$
    else $\delta := d$; $\mathcal{G} := Build(S_{i+1}, \delta)$;
  end;
return(δ).

$\delta(S_7) = d(p_4, p_1)$

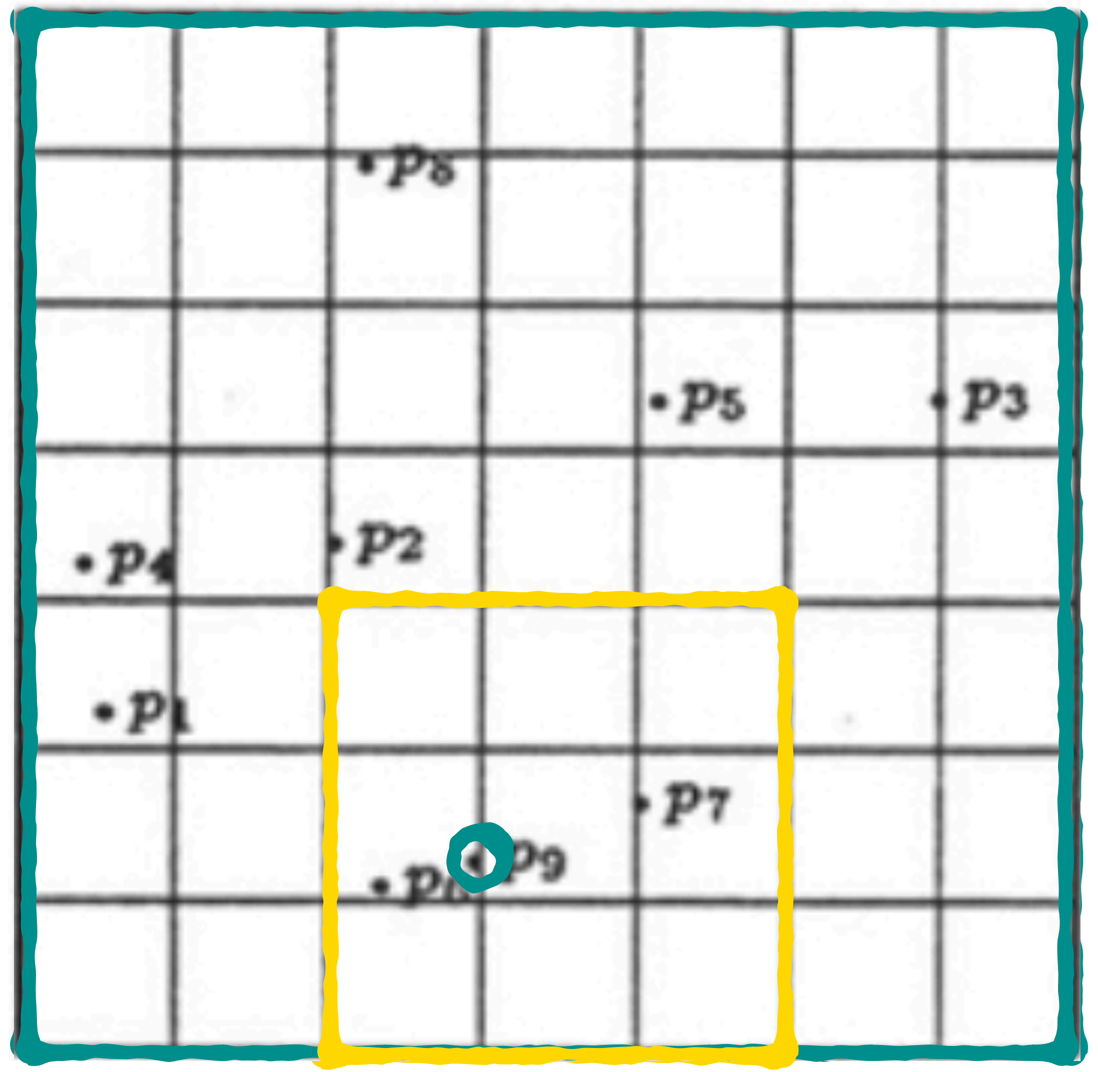


Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Algorithm $CP(p_1, p_2, \ldots, p_n)$

$\delta := d(p_1, p_2)$; $\mathcal{G} := Build(S_2, \delta)$;
for $i := 2$ **to** $n - 1$ **do**
  begin
    $V := \{Report(\mathcal{G}, b)\ :\ b$ is a neighbor of the box containing $p_{i+1}\}$;
    $d := \min_{q \in V} d(p_{i+1}, q)$;
    if $d \geq \delta$ **then** $Insert(\mathcal{G}, p_{i+1})$
      else $\delta := d$; $\mathcal{G} := Build(S_{i+1}, \delta)$;
  end;
return(δ).

$\delta(S_8) = d(p_4, p_1)$

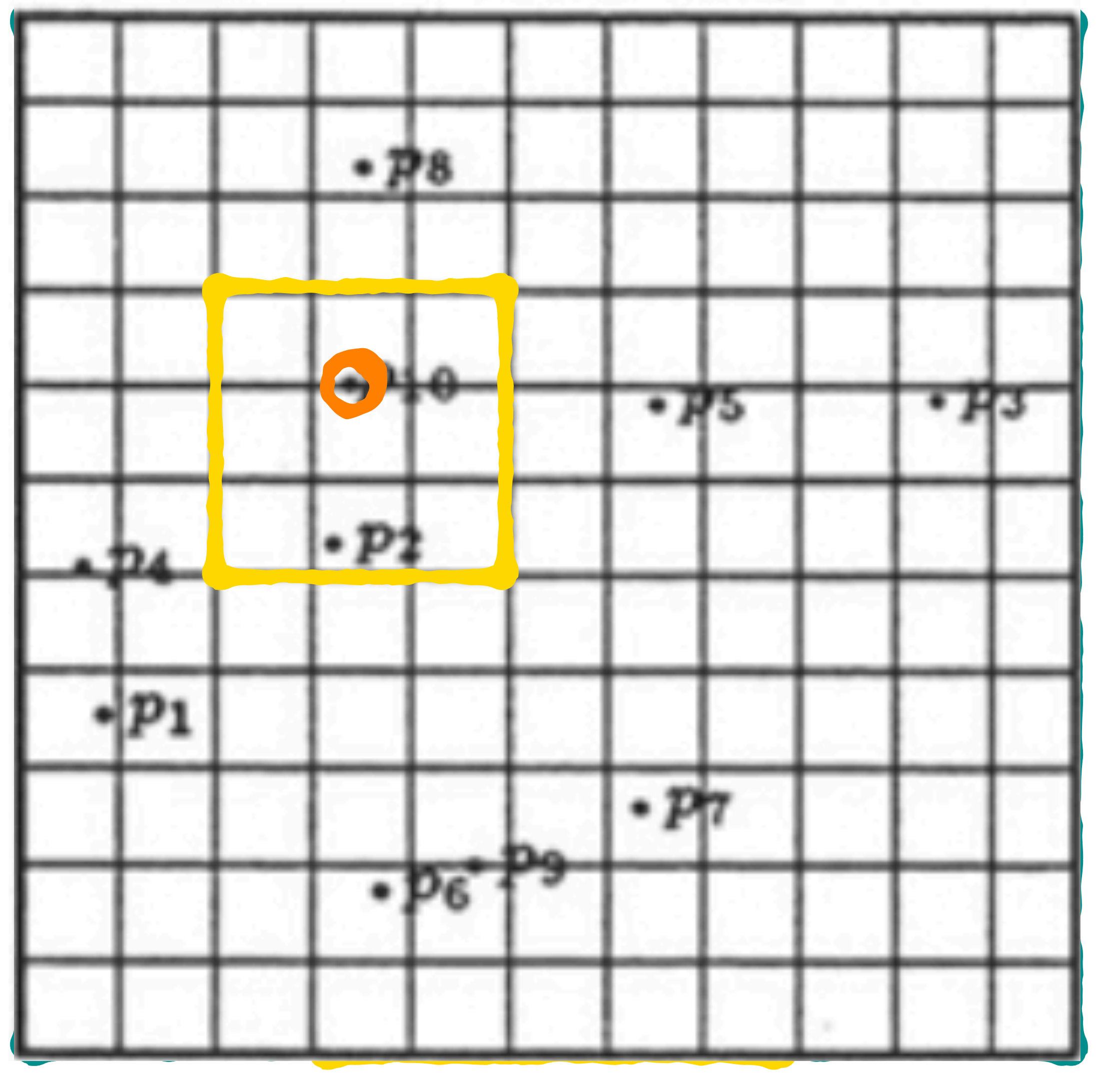


Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Algorithm $CP(p_1, p_2, \ldots, p_n)$

```
δ := d(p_1, p_2); G := Build(S_2, δ);
for i := 2 to n − 1 do
    begin
    V := {Report(G, b) : b is a neighbor of the box containing p_{i+1}};
    d := min_{q∈V} d(p_{i+1}, q);
    if d ≥ δ then Insert(G, p_{i+1})
              else δ := d; G := Build(S_{i+1}, δ);
    end;
return(δ).
```

$\delta(S_9) = d(p_9, p_6)$

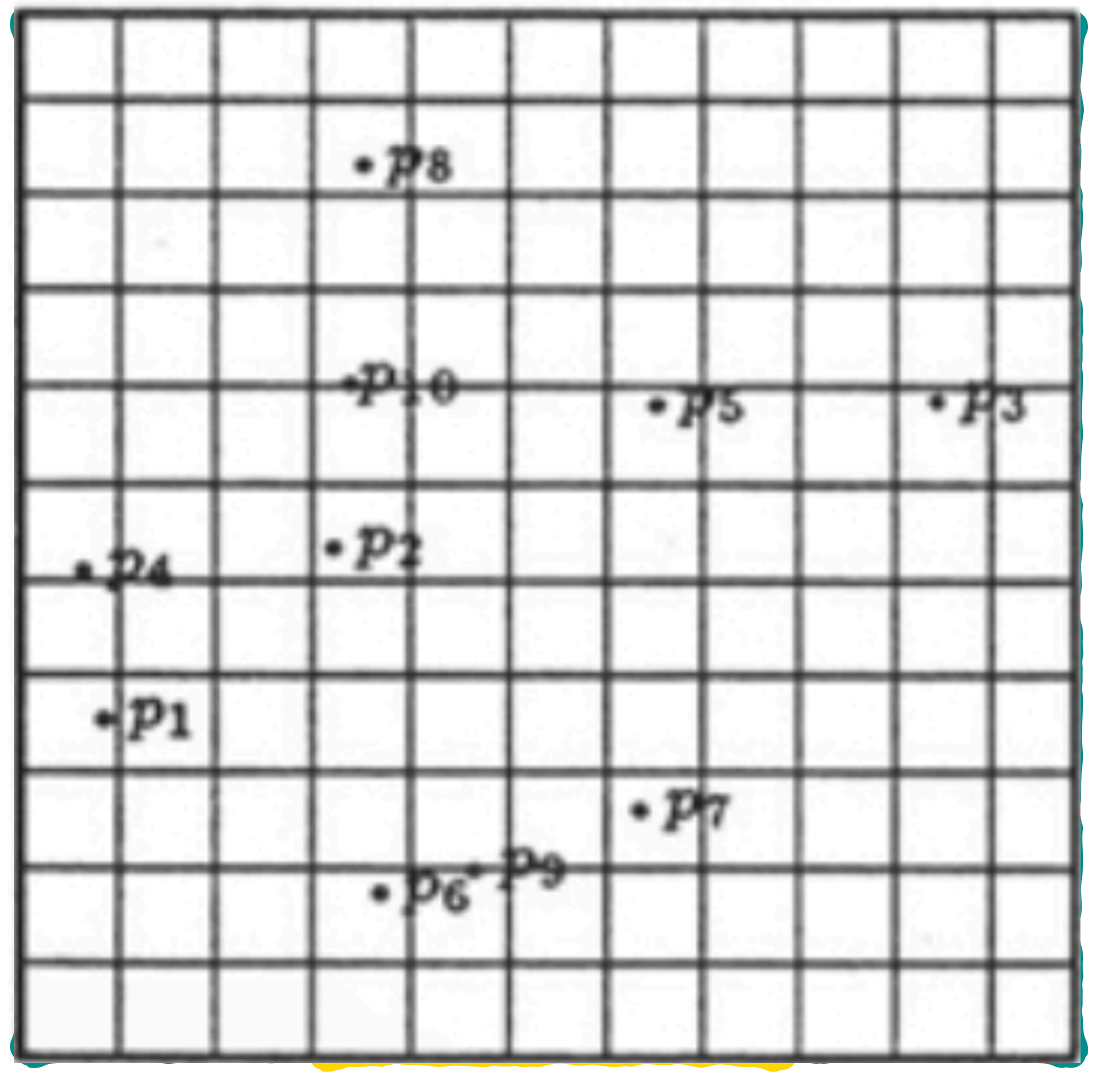


Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Algorithm $CP(p_1, p_2, \ldots, p_n)$

```
δ := d(p_1, p_2); G := Build(S_2, δ);
for i := 2 to n − 1 do
    begin
    V := {Report(G, b) : b is a neighbor of the box containing p_{i+1}};
    d := min_{q∈V} d(p_{i+1}, q);
    if d ≥ δ then Insert(G, p_{i+1})
              else δ := d; G := Build(S_{i+1}, δ);
    end;
return(δ).
```

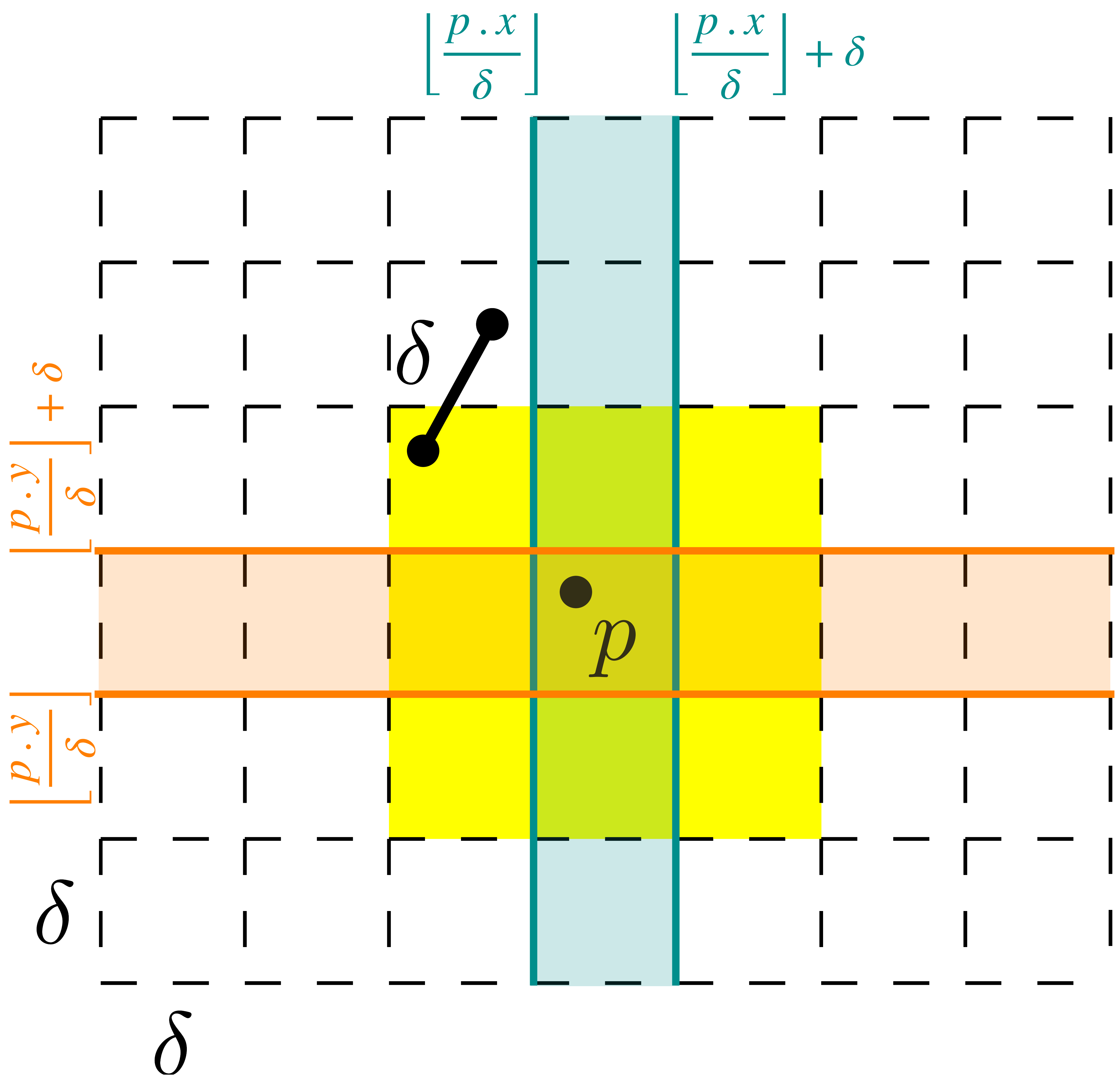
$\delta(S_{10}) = d(p_9, p_6)$



Fairly straightforward algorithm :)

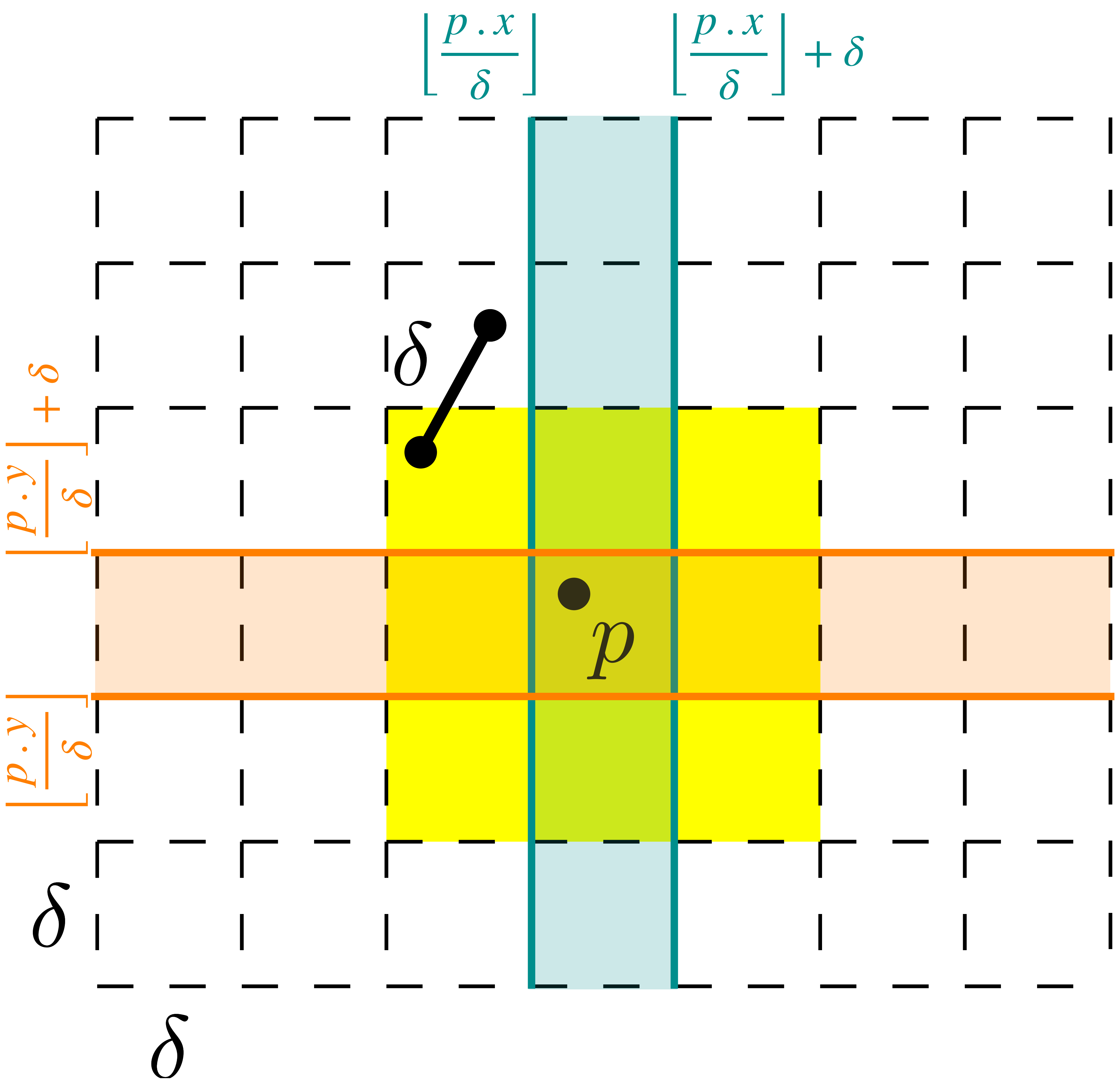
Randomised Incremental Construction

Golin, Raman, Schwarz, Smid 1992/1995



Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Representation

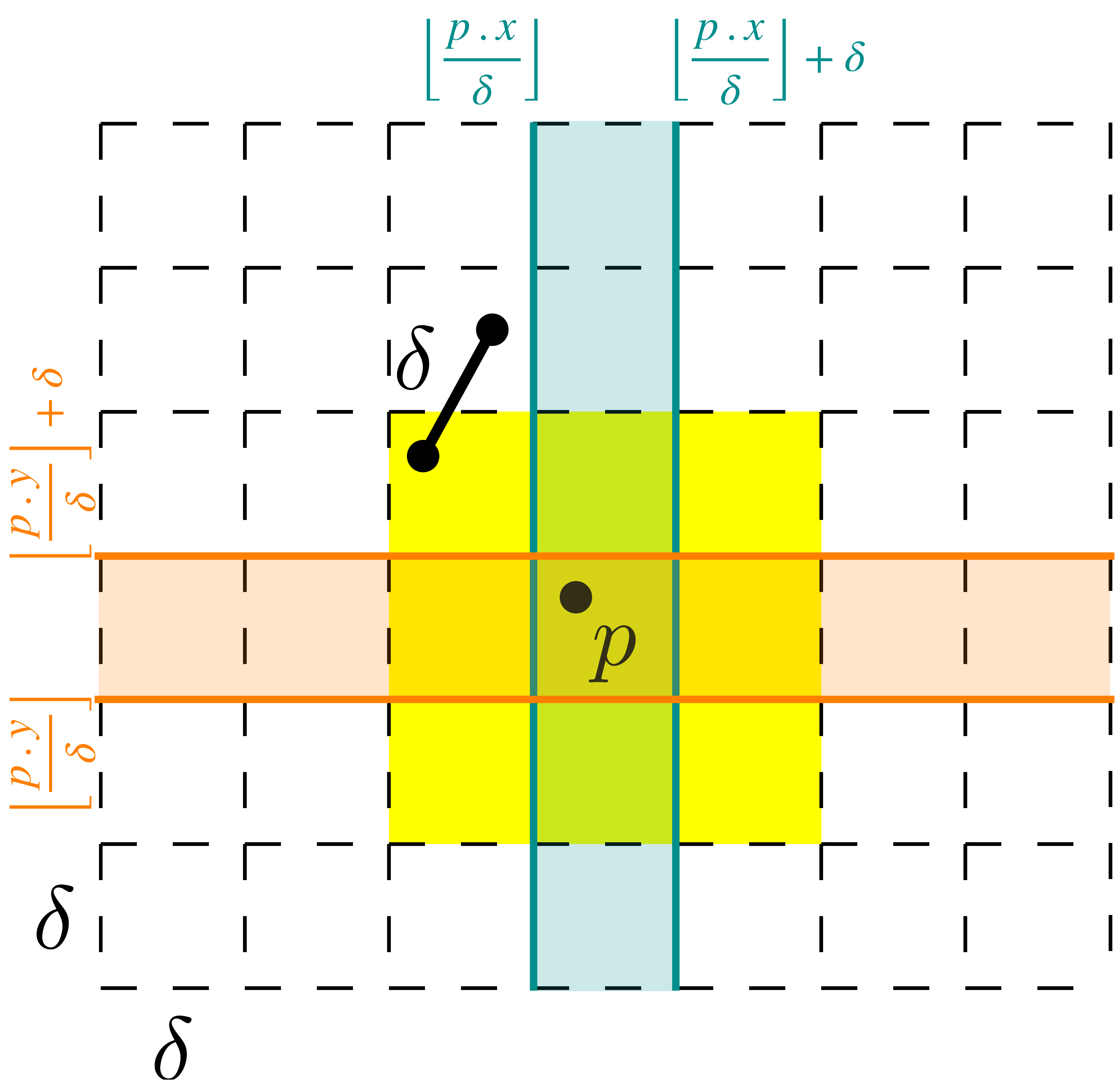


Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Representation

- Grid cell representation

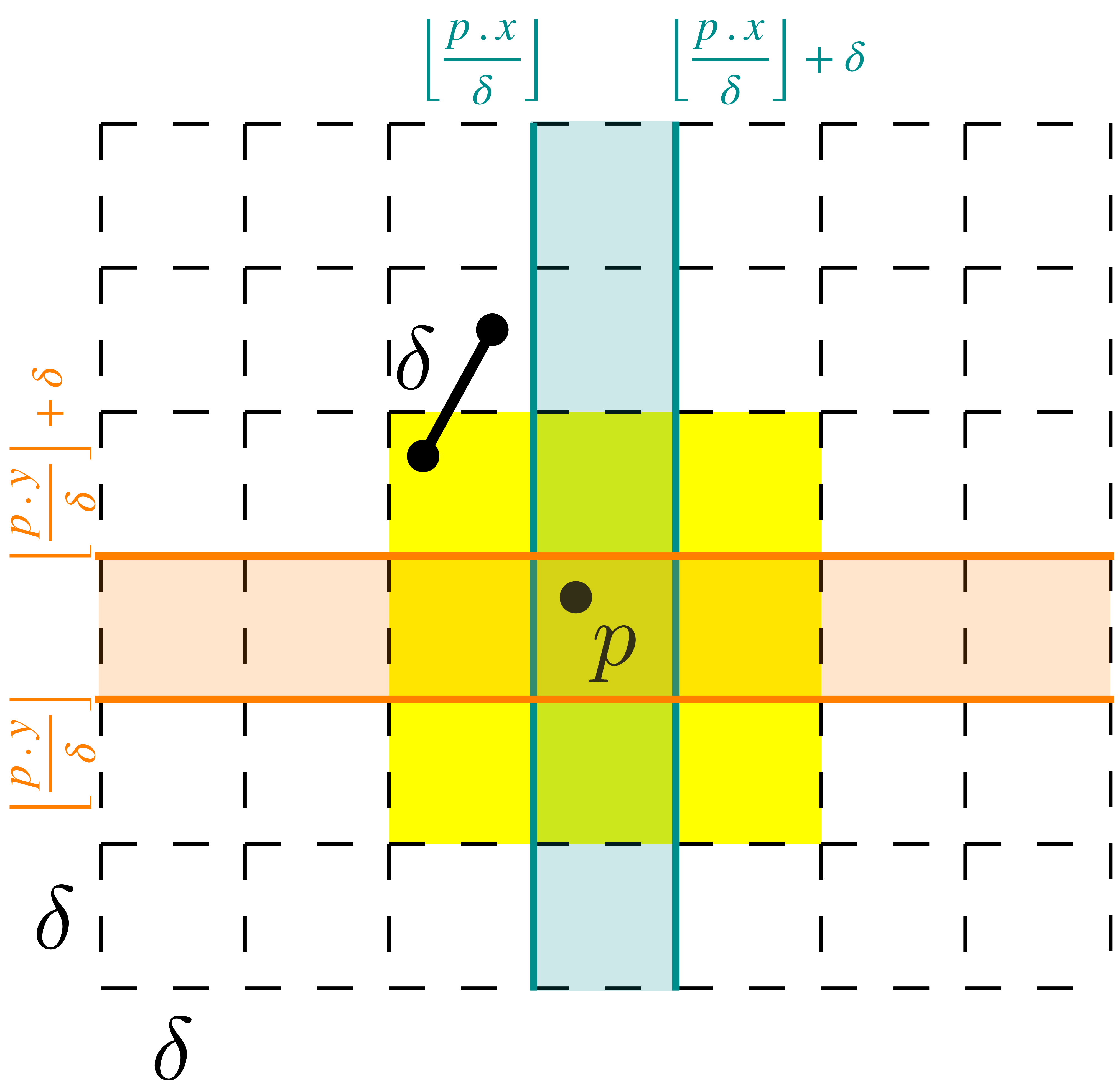
`(x,y)`



Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Representation

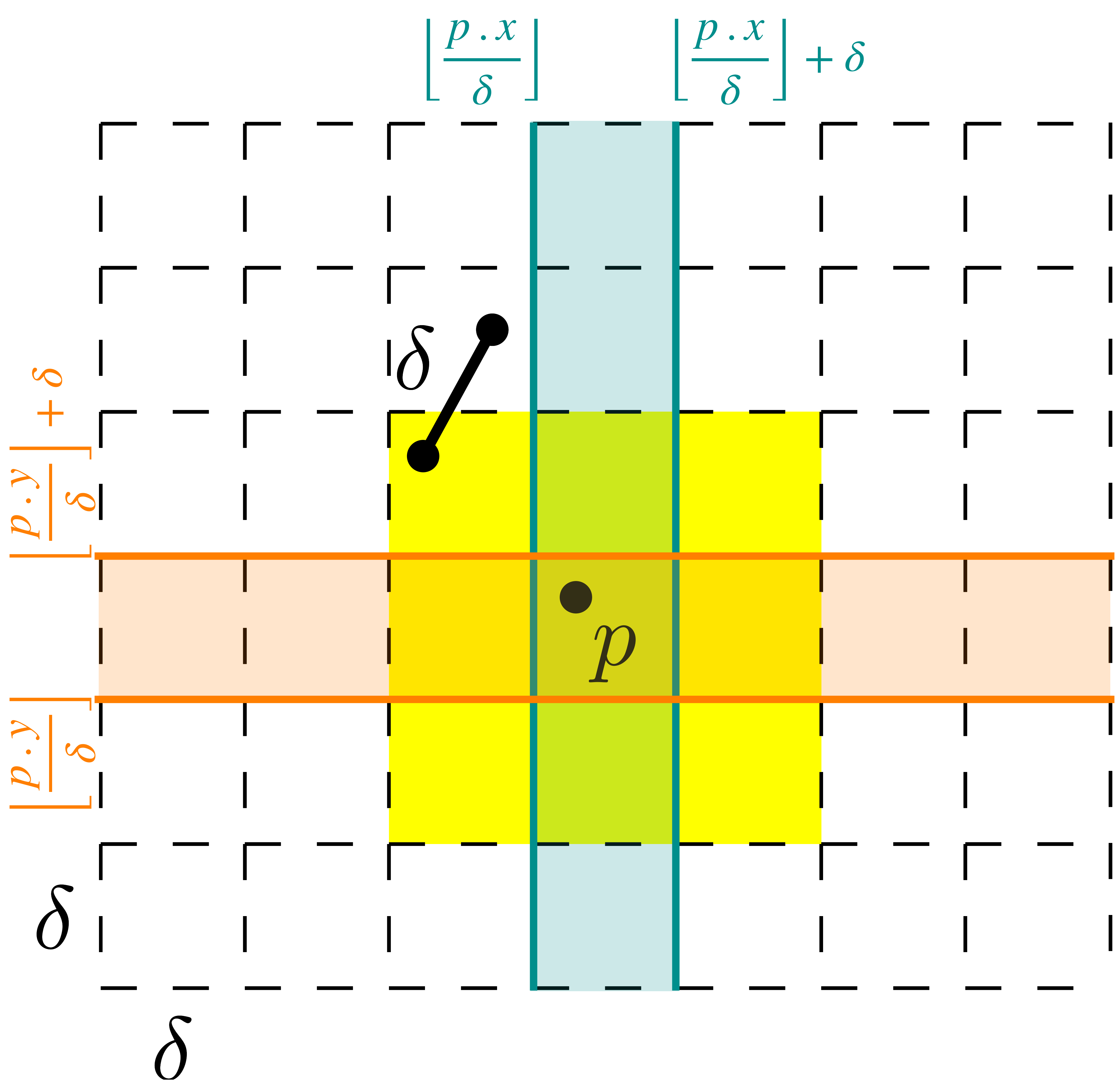
- Grid cell representation **`(x,y)`**
- Grid cell assignment G_δ **`OrderedDictionary`**



Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Representation

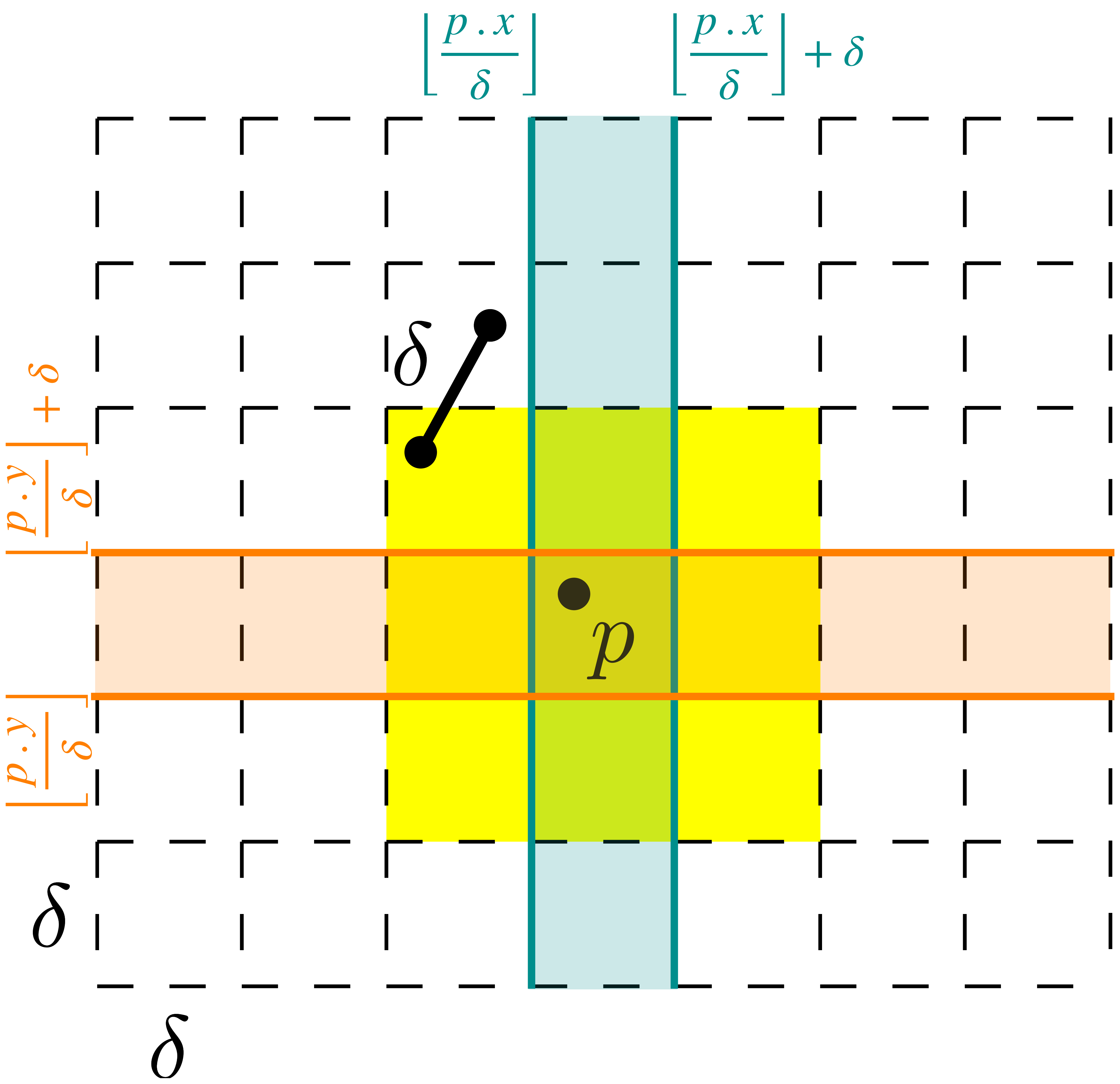
- Grid cell representation `(x,y)`
- Grid cell assignment G_δ `OrderedDictionary`
- Identifier/key for $p \in \mathscr{P}$ `(p.x,p.y)`



Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Representation

- Grid cell representation `(x,y)`
- Grid cell assignment G_δ `OrderedDictionary`
- Identifier/key for $p \in \mathscr{P}$ `(p.x,p.y)`
- $p \in G_\delta[x, y] \quad \Leftrightarrow \quad x = \left\lfloor \frac{p\,.\,x}{\delta} \right\rfloor, y = \left\lfloor \frac{p\,.\,y}{\delta} \right\rfloor$

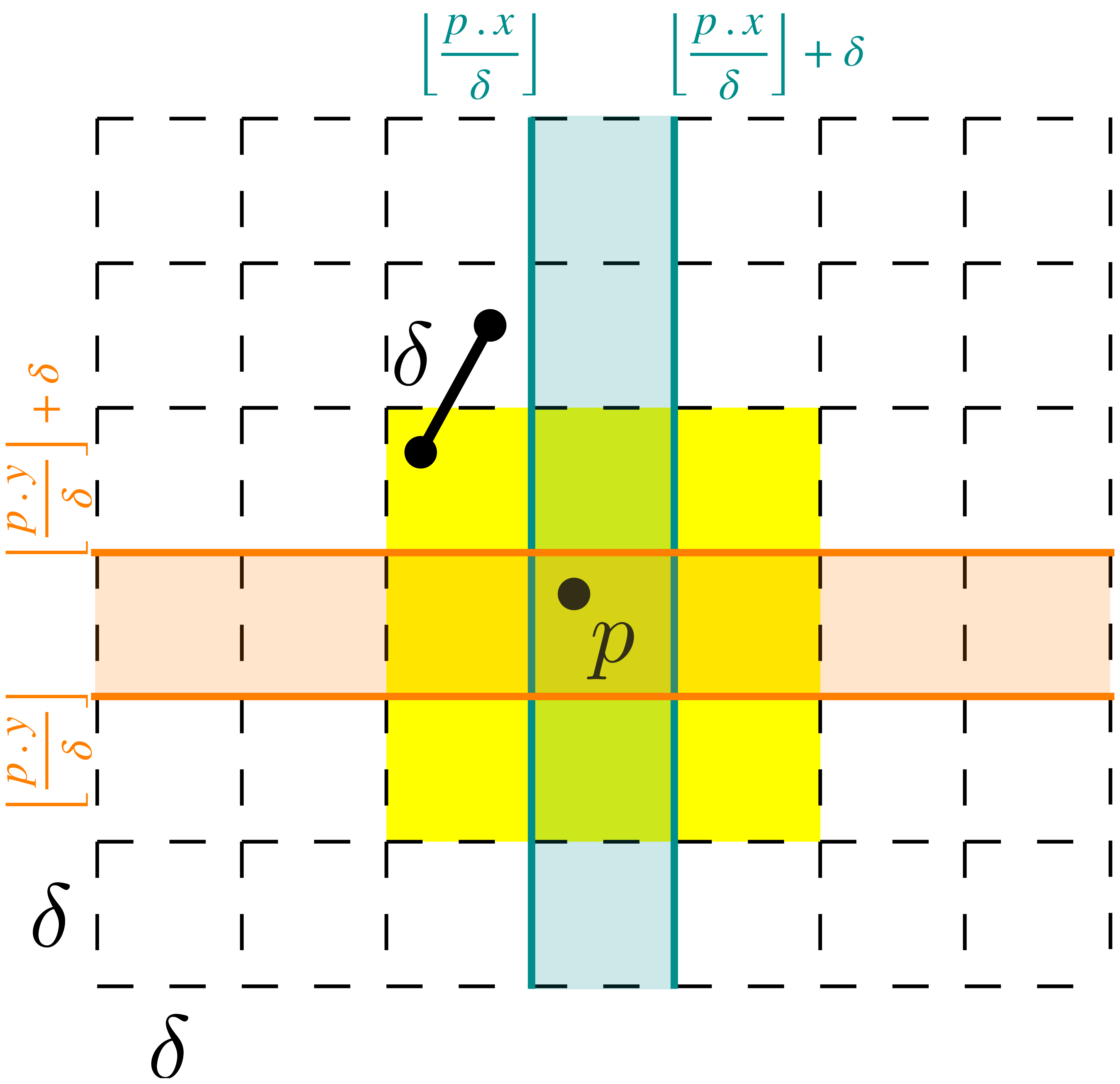


Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Representation

- Grid cell representation `(x,y)`
- Grid cell assignment G_δ `OrderedDictionary`
- Identifier/key for $p \in \mathscr{P}$ `(p.x,p.y)`
- $p \in G_\delta[x, y] \quad \Leftrightarrow \quad x = \left\lfloor \frac{p\,.\,x}{\delta} \right\rfloor, y = \left\lfloor \frac{p\,.\,y}{\delta} \right\rfloor$

Implementation



Randomised Incremental Construction

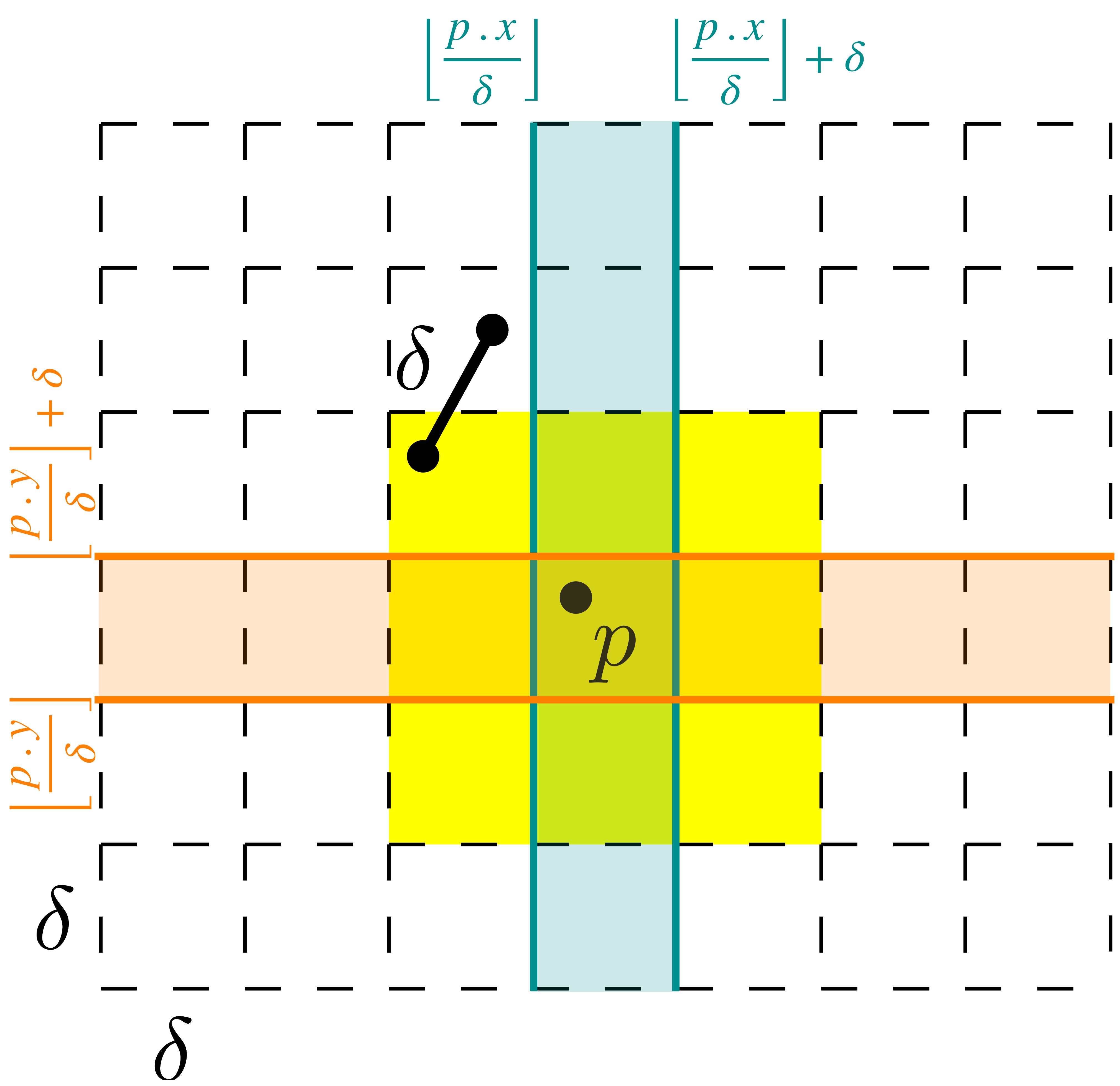
Golin, Raman, Schwarz, Smid 1992/1995

Representation

- Grid cell representation `(x,y)`
- Grid cell assignment G_δ `OrderedDictionary`
- Identifier/key for $p \in \mathscr{P}$ `(p.x,p.y)`
- $p \in G_\delta[x, y] \quad \Leftrightarrow \quad x = \left\lfloor \frac{p\,.\,x}{\delta} \right\rfloor, y = \left\lfloor \frac{p\,.\,y}{\delta} \right\rfloor$

Implementation

- $T_{build}(n)$ Construction cost



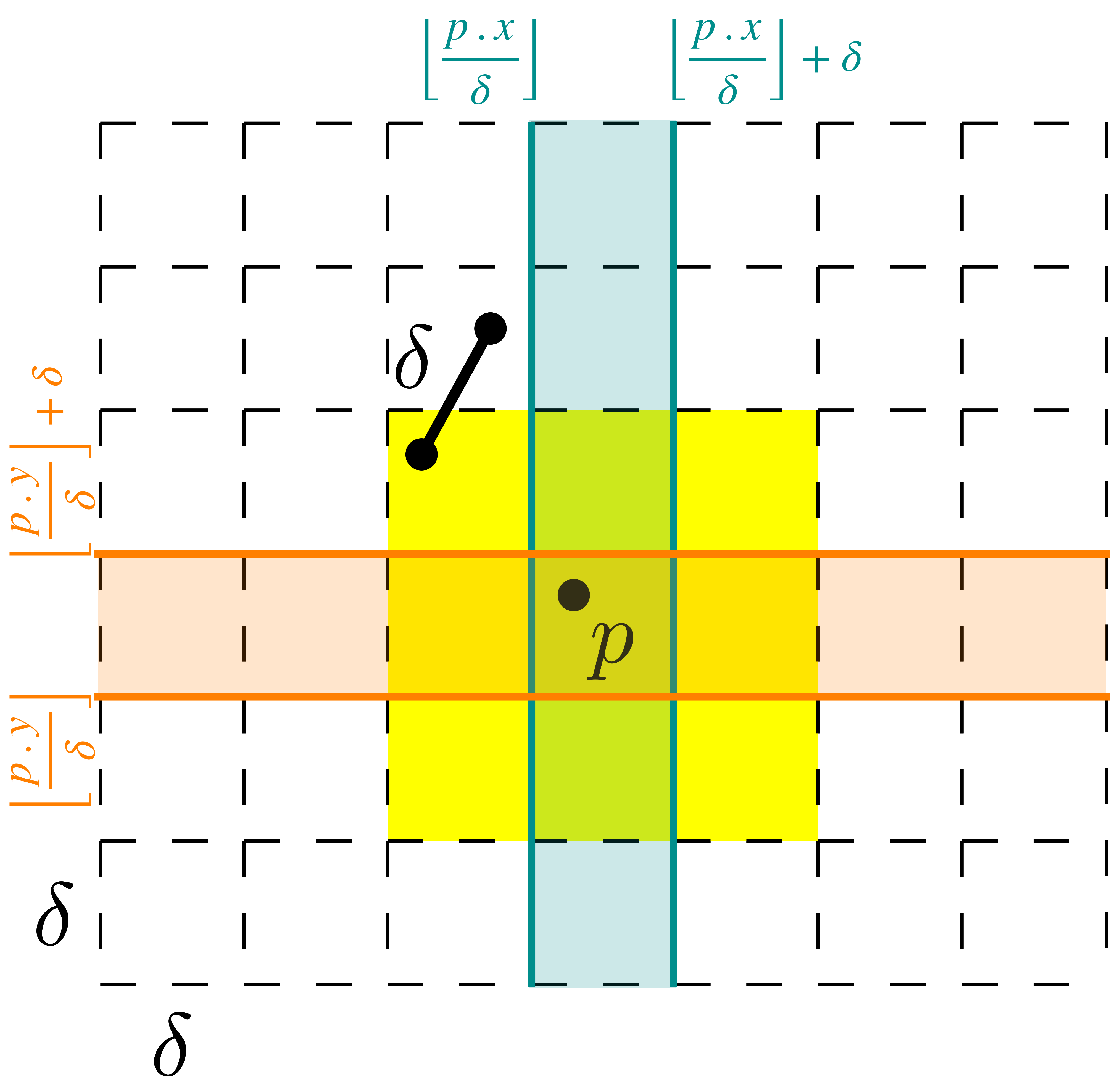
Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Representation

- Grid cell representation `(x,y)`
- Grid cell assignment G_δ `OrderedDictionary`
- Identifier/key for $p \in \mathscr{P}$ `(p.x,p.y)`
- $p \in G_\delta[x, y] \quad \Leftrightarrow \quad x = \left\lfloor \frac{p.x}{\delta} \right\rfloor, y = \left\lfloor \frac{p.y}{\delta} \right\rfloor$

Implementation

- $T_{build}(n)$ Construction cost
- $T_{insert}(n)$ Insertion cost



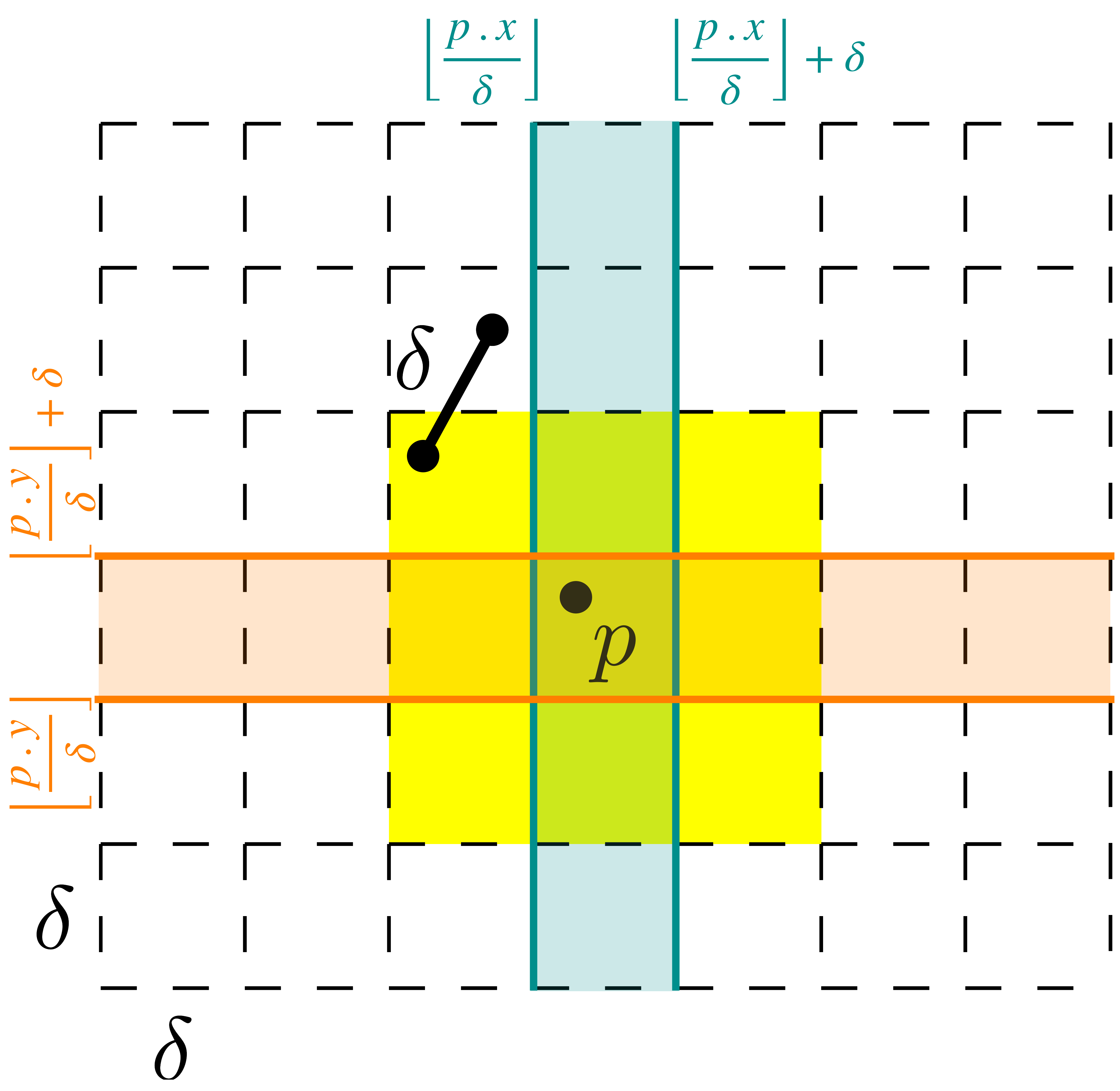
Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Representation

- Grid cell representation `(x,y)`
- Grid cell assignment G_δ `OrderedDictionary`
- Identifier/key for $p \in \mathscr{P}$ `(p.x,p.y)`
- $p \in G_\delta[x, y] \quad \Leftrightarrow \quad x = \left\lfloor \frac{p.x}{\delta} \right\rfloor, y = \left\lfloor \frac{p.y}{\delta} \right\rfloor$

Implementation

- $T_{build}(n)$ Construction cost
- $T_{insert}(n)$ Insertion cost
- $T_{query}(n)$ Query cost



Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Algorithm $CP(p_1, p_2, \ldots, p_n)$

$\delta := d(p_1, p_2)$; $\mathcal{G} := Build(S_2, \delta)$;
for $i := 2$ **to** $n - 1$ **do**
    begin
    $V := \{Report(\mathcal{G}, b) : b$ is a neighbor of the box containing $p_{i+1}\}$;
    $d := \min_{q \in V} d(p_{i+1}, q)$;
    if $d \geq \delta$ **then** $Insert(\mathcal{G}, p_{i+1})$
        else $\delta := d$; $\mathcal{G} := Build(S_{i+1}, \delta)$;
    end;
return(δ).

Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Terminology

Algorithm $CP(p_1, p_2, \ldots, p_n)$

$\delta := d(p_1, p_2)$; $\mathcal{G} := Build(S_2, \delta)$;
for $i := 2$ **to** $n-1$ **do**
  begin
  $V := \{Report(\mathcal{G}, b)$: b is a neighbor of the box containing $p_{i+1}\}$;
  $d := \min_{q \in V} d(p_{i+1}, q)$;
  if $d \geq \delta$ **then** $Insert(\mathcal{G}, p_{i+1})$
    else $\delta := d$; $\mathcal{G} := Build(S_{i+1}, \delta)$;
  end;
return(δ).

Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Terminology

- $(p_1, p_2, \ldots, p_n)$ **Permuted sequence**

```
Algorithm  CP(p_1, p_2, ..., p_n)

δ := d(p_1, p_2); G := Build(S_2, δ);
for i := 2 to n − 1 do
    begin
    V := {Report(G, b) : b is a neighbor of the box containing p_{i+1}};
    d := min_{q∈V} d(p_{i+1}, q);
    if d ≥ δ then Insert(G, p_{i+1})
              else δ := d; G := Build(S_{i+1}, δ);
    end;
return(δ).
```

Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Terminology

- $(p_1, p_2, \ldots, p_n)$ **Permuted sequence**
- $\mathscr{P}_i = \{p_1, p_2, \ldots, p_i\}$ **First i points added**

Algorithm $CP(p_1, p_2, \ldots, p_n)$

$\delta := d(p_1, p_2)$; $\mathcal{G} := Build(S_2, \delta)$;
for $i := 2$ **to** $n - 1$ **do**
  begin
  $V := \{Report(\mathcal{G}, b)$: b is a neighbor of the box containing $p_{i+1}\}$;
  $d := \min_{q \in V} d(p_{i+1}, q)$;
  if $d \geq \delta$ **then** $Insert(\mathcal{G}, p_{i+1})$
    else $\delta := d$; $\mathcal{G} := Build(S_{i+1}, \delta)$;
  end;
return(δ).

Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Terminology

- $(p_1, p_2, \ldots, p_n)$ **Permuted sequence**
- $\mathscr{P}_i = \{p_1, p_2, \ldots, p_i\}$ **First i points added**
- $\delta_i = \min\{\ d(p,q)\ \mid\ p, q \in \mathscr{P}_i\ \}$ **Min. distance**

Algorithm $CP(p_1, p_2, \ldots, p_n)$

$\delta := d(p_1, p_2)$; $\mathcal{G} := Build(S_2, \delta)$;
for $i := 2$ **to** $n-1$ **do**
  begin
    $V := \{Report(\mathcal{G}, b)$: b is a neighbor of the box containing $p_{i+1}\}$;
    $d := \min_{q \in V} d(p_{i+1}, q)$;
    if $d \geq \delta$ **then** $Insert(\mathcal{G}, p_{i+1})$
        else $\delta := d$; $\mathcal{G} := Build(S_{i+1}, \delta)$;
  end;
return(δ).

Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Terminology

- $(p_1, p_2, \ldots, p_n)$ **Permuted sequence**
- $\mathscr{P}_i = \{p_1, p_2, \ldots, p_i\}$ **First i points added**
- $\delta_i = \min\{\ d(p,q)\ \mid\ p,q \in \mathscr{P}_i\ \}$ **Min. distance**

Randomisation — Expected runtime.

Algorithm $CP(p_1, p_2, \ldots, p_n)$

$\delta := d(p_1, p_2)$; $\mathcal{G} := Build(S_2, \delta)$;
for $i := 2$ **to** $n-1$ **do**
  begin
  $V := \{Report(\mathcal{G}, b) : b$ is a neighbor of the box containing $p_{i+1}\}$;
  $d := \min_{q \in V} d(p_{i+1}, q)$;
  if $d \geq \delta$ **then** $Insert(\mathcal{G}, p_{i+1})$
    else $\delta := d$; $\mathcal{G} := Build(S_{i+1}, \delta)$;
  end;
return(δ).

Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Terminology

- $(p_1, p_2, \ldots, p_n)$ **Permuted sequence**
- $\mathscr{P}_i = \{p_1, p_2, \ldots, p_i\}$ **First i points added**
- $\delta_i = \min\{\ d(p,q)\ \mid\ p,q \in \mathscr{P}_i\ \}$ **Min. distance**

Randomisation — Expected runtime.

- Construction cost in $\Omega(n)$, but also $\mathcal{O}(n^2)$…?

```
Algorithm  CP(p_1, p_2, ..., p_n)

δ := d(p_1, p_2); G := Build(S_2, δ);
for i := 2 to n − 1 do
    begin
    V := {Report(G, b) : b is a neighbor of the box containing p_{i+1}};
    d := min_{q∈V} d(p_{i+1}, q);
    if d ≥ δ then Insert(G, p_{i+1})
              else δ := d; G := Build(S_{i+1}, δ);
    end;
return(δ).
```

Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Terminology

- $(p_1, p_2, \ldots, p_n)$ **Permuted sequence**
- $\mathscr{P}_i = \{p_1, p_2, \ldots, p_i\}$ **First i points added**
- $\delta_i = \min\{\ d(p,q)\ \mid\ p, q \in \mathscr{P}_i\ \}$ **Min. distance**

Randomisation — Expected runtime.

- Construction cost in $\Omega(n)$, but also $\mathcal{O}(n^2)$…?
- Depends on the order of points.

```
Algorithm CP(p_1, p_2, ..., p_n)

δ := d(p_1, p_2); G := Build(S_2, δ);
for i := 2 to n − 1 do
    begin
    V := {Report(G, b) : b is a neighbor of the box containing p_{i+1}};
    d := min_{q∈V} d(p_{i+1}, q);
    if d ≥ δ then Insert(G, p_{i+1})
             else δ := d; G := Build(S_{i+1}, δ);
    end;
return(δ).
```

Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Terminology

- $(p_1, p_2, \dots, p_n)$ **Permuted sequence**
- $\mathscr{P}_i = \{p_1, p_2, \dots, p_i\}$ **First i points added**
- $\delta_i = \min\{\ d(p,q)\ \mid\ p, q \in \mathscr{P}_i\ \}$ **Min. distance**

```
Algorithm  CP(p_1, p_2, ..., p_n)

δ := d(p_1, p_2); 𝒢 := Build(S_2, δ);
for i := 2 to n − 1 do
    begin
    V := {Report(𝒢, b) : b is a neighbor of the box containing p_{i+1}};
    d := min_{q∈V} d(p_{i+1}, q);
    if d ≥ δ then Insert(𝒢, p_{i+1})
              else δ := d; 𝒢 := Build(S_{i+1}, δ);
    end;
return(δ).
```

Randomisation — Expected runtime.

- Construction cost in $\Omega(n)$, but also $\mathcal{O}(n^2)$…?
- Depends on the order of points.
- Randomised: $X(p_i, \mathscr{P}_{i-1}) = \begin{cases} 1, \text{ if } \delta_i < \delta_{i-1} \\ 0, \text{ otherwise.} \end{cases}$

Randomised Incremental Construction

Golin, Raman, Schwarz, Smid 1992/1995

Randomised Incremental Construction

Golin, Raman, Schwarz, Smid 1992/1995

Runtime

Randomised Incremental Construction

Golin, Raman, Schwarz, Smid 1992/1995

Runtime

- **Cost of** $\mathscr{P}_i \rightarrow \mathscr{P}_{i+1}$**:**

Randomised Incremental Construction

Golin, Raman, Schwarz, Smid 1992/1995

Runtime

- **Cost of** $\mathscr{P}_i \to \mathscr{P}_{i+1}$**:**

$$T(i) \in \mathcal{O}\Big(T_{insert}(i) \;\; + \;\; T_{query}(i) \;\; + \;\; X(p_{i+1}, \mathscr{P}_i) \cdot T_{build}(i)\Big)$$

Randomised Incremental Construction

Golin, Raman, Schwarz, Smid 1992/1995

Runtime

- **Cost of $\mathscr{P}_i \to \mathscr{P}_{i+1}$:**

$$T(i) \in \mathcal{O}\Big(T_{insert}(i) \;+\; T_{query}(i) \;+\; X(p_{i+1}, \mathscr{P}_i) \cdot T_{build}(i)\Big)$$

- This is non-deterministic — we get a random permutation!

Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Runtime

- **Cost of $\mathscr{P}_i \to \mathscr{P}_{i+1}$:**

$$T(i) \in \mathcal{O}\big(T_{insert}(i) \;+\; T_{query}(i) \;+\; X(p_{i+1}, \mathscr{P}_i) \cdot T_{build}(i)\big)$$

- This is non-deterministic — we get a random permutation!
- **Expected cost of $\mathscr{P}_i \to \mathscr{P}_{i+1}$:**

$$E\big[T(i)\big] \in \mathcal{O}\Big(E\big[T_{insert}(i)\big] \;+\; E\big[T_{query}(i)\big] \;+\; Pr\big[\{\delta_{i+1} < \delta_i\}\big] \cdot E\big[T_{build}(i+1)\big]\Big)$$

Randomised Incremental Construction

Golin, Raman, Schwarz, Smid 1992/1995

Runtime

- **Cost of $\mathscr{P}_i \rightarrow \mathscr{P}_{i+1}$:**

$$T(i) \in \mathcal{O}\left(T_{insert}(i) \;+\; T_{query}(i) \;+\; X(p_{i+1}, \mathscr{P}_i) \cdot T_{build}(i)\right)$$

- This is non-deterministic — we get a random permutation!
- **Expected cost of $\mathscr{P}_i \rightarrow \mathscr{P}_{i+1}$:**

$$E\left[T(i)\right] \in \mathcal{O}\left(E\left[T_{insert}(i)\right] \;+\; E\left[T_{query}(i)\right] \;+\; \underline{Pr\left[\{\delta_{i+1} < \delta_i\}\right]} \cdot E\left[T_{build}(i+1)\right]\right)$$

Randomised Incremental Construction

Golin, Raman, Schwarz, Smid 1992/1995

Runtime

- **Cost of $\mathscr{P}_i \rightarrow \mathscr{P}_{i+1}$:**

$$T(i) \in \mathcal{O}\Big(T_{insert}(i) \ + \ T_{query}(i) \ + \ X(p_{i+1}, \mathscr{P}_i) \cdot T_{build}(i)\Big)$$

- This is non-deterministic — we get a random permutation!
- **Expected cost of $\mathscr{P}_i \rightarrow \mathscr{P}_{i+1}$:**

$$E\big[T(i)\big] \in \mathcal{O}\Big(E\big[T_{insert}(i)\big] \ + \ E\big[T_{query}(i)\big] \ + \ Pr\big[\{\delta_{i+1} < \delta_i\}\big] \cdot E\big[T_{build}(i+1)\big]\Big)$$

Lemma 1 *Let $p_1, p_2, \ldots, p_n$ be a random permutation of the points of S. Let $S_i := \{p_1, p_2, \ldots, p_i\}$. Then* $\Pr[\delta(S_{i+1}) < \delta(S_i)] \leq 2/(i+1)$.

Randomised Incremental Construction

Golin, Raman, Schwarz, Smid 1992/1995

Randomised Incremental Construction

Golin, Raman, Schwarz, Smid 1992/1995

Lemma. $Pr\left[\{\delta_{i+1} < \delta_i\}\right] \le \dfrac{2}{i+1}$

Randomised Incremental Construction

Golin, Raman, Schwarz, Smid 1992/1995

Lemma. $Pr\left[\{\delta_{i+1} < \delta_i\}\right] \leq \dfrac{2}{i+1}$

Proof (Backwards analysis)**.**

Randomised Incremental Construction

Golin, Raman, Schwarz, Smid 1992/1995

Lemma. $Pr\left[\{\delta_{i+1} < \delta_i\}\right] \leq \dfrac{2}{i+1}$

Proof (Backwards analysis)**.**

- Let $\{p\} = \mathscr{P}_{i+1} \setminus \mathscr{P}_i$. **Last point added.**

Randomised Incremental Construction

Golin, Raman, Schwarz, Smid 1992/1995

Lemma. $Pr\left[\{\delta_{i+1} < \delta_i\}\right] \leq \frac{2}{i+1}$

Proof (Backwards analysis)**.**

- Let $\{p\} = \mathscr{P}_{i+1} \setminus \mathscr{P}_i$. **Last point added.**
- $S_{i+1} = \{\, u \in \mathscr{P}_i \mid \exists\, v \in \mathscr{P}_i :\ d(u,v) = \delta_{i+1}\}$. **All points in closest pairs at step $i+1$.**

Randomised Incremental Construction

Golin, Raman, Schwarz, Smid 1992/1995

Lemma. $Pr\left[\{\delta_{i+1} < \delta_i\}\right] \leq \dfrac{2}{i+1}$

Proof (Backwards analysis)**.**

- Let $\{p\} = \mathscr{P}_{i+1} \setminus \mathscr{P}_i$. **Last point added.**
- $S_{i+1} = \{\, u \in \mathscr{P}_i \mid \exists\, v \in \mathscr{P}_i : \; d(u, v) = \delta_{i+1}\}$. **All points in closest pairs at step $i + 1$.**
- Grid rebuilt $\Leftrightarrow \delta_{i+1} < \delta_i$.

Randomised Incremental Construction

Golin, Raman, Schwarz, Smid 1992/1995

Lemma. $Pr\left[\{\delta_{i+1} < \delta_i\}\right] \leq \dfrac{2}{i+1}$

Proof (Backwards analysis)**.**

- Let $\{p\} = \mathscr{P}_{i+1} \backslash \mathscr{P}_i$. — **Last point added.**
- $S_{i+1} = \{\, u \in \mathscr{P}_i \mid \exists\, v \in \mathscr{P}_i : \; d(u,v) = \delta_{i+1}\}$. — **All points in closest pairs at step $i+1$.**
- Grid rebuilt $\Leftrightarrow \delta_{i+1} < \delta_i$.
- If $|S_{i+1}| = 2$: $\quad Pr\left[\{\delta_{i+1} < \delta_i\}\right] = Pr\left[\{p \in S_{i+1}\}\right] = \dfrac{2}{i+1}$ — **Unique closest pair! What are the odds?**

Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

Lemma. $Pr\left[\{\delta_{i+1} < \delta_i\}\right] \leq \dfrac{2}{i+1}$

Proof (Backwards analysis)**.**

- Let $\{p\} = \mathscr{P}_{i+1} \backslash \mathscr{P}_i$. **Last point added.**
- $S_{i+1} = \{ \, u \in \mathscr{P}_i \mid \exists \, v \in \mathscr{P}_i : \; d(u, v) = \delta_{i+1} \}$. **All points in closest pairs at step $i+1$.**
- Grid rebuilt $\Leftrightarrow \delta_{i+1} < \delta_i$.
- If $|S_{i+1}| = 2$: $Pr\left[\{\delta_{i+1} < \delta_i\}\right] = Pr\left[\{p \in S_{i+1}\}\right] = \dfrac{2}{i+1}$ **Unique closest pair! What are the odds?**
- Otherwise ($|S_{i+1}| > 2$):

Randomised Incremental Construction

Golin, Raman, Schwarz, Smid 1992/1995

Lemma. $Pr\left[\{\delta_{i+1} < \delta_i\}\right] \leq \dfrac{2}{i+1}$

Proof (Backwards analysis)**.**

- Let $\{p\} = \mathscr{P}_{i+1} \backslash \mathscr{P}_i$. — **Last point added.**
- $S_{i+1} = \{\ u \in \mathscr{P}_i \mid \exists\ v \in \mathscr{P}_i :\ \ d(u, v) = \delta_{i+1}\}$. — **All points in closest pairs at step $i + 1$.**
- Grid rebuilt $\Leftrightarrow \delta_{i+1} < \delta_i$.
- If $|S_{i+1}| = 2$: $Pr\left[\{\delta_{i+1} < \delta_i\}\right] = Pr\left[\{p \in S_{i+1}\}\right] = \dfrac{2}{i+1}$ — **Unique closest pair! What are the odds?**
- Otherwise ($|S_{i+1}| > 2$):
  - Case 1: All closest pairs in S_i share a unique point. — **We just found the unique "centre" of these pairs!**

    $$Pr\left[\{\delta_{i+1} < \delta_i\}\right] = Pr\left[\{p = s\}\right] = \frac{1}{i+1}$$

Randomised Incremental Construction

Golin, Raman, Schwarz, Smid 1992/1995

Lemma. $Pr\left[\{\delta_{i+1} < \delta_i\}\right] \leq \dfrac{2}{i+1}$

Proof (Backwards analysis)**.**

- Let $\{p\} = \mathscr{P}_{i+1} \backslash \mathscr{P}_i$. — **Last point added.**
- $S_{i+1} = \{\ u \in \mathscr{P}_i \mid \exists\ v \in \mathscr{P}_i :\ \ d(u,v) = \delta_{i+1}\}$. — **All points in closest pairs at step $i+1$.**
- Grid rebuilt $\Leftrightarrow \delta_{i+1} < \delta_i$.
- If $|S_{i+1}| = 2$: $Pr\left[\{\delta_{i+1} < \delta_i\}\right] = Pr\left[\{p \in S_{i+1}\}\right] = \dfrac{2}{i+1}$ — **Unique closest pair! What are the odds?**
- Otherwise ($|S_{i+1}| > 2$):
    - Case 1: All closest pairs in S_i share a unique point. — **We just found the unique “centre" of these pairs!**

      $$Pr\left[\{\delta_{i+1} < \delta_i\}\right] = Pr\left[\{p = s\}\right] = \frac{1}{i+1}$$
    - Case 2: No point is shared among the all pairs. — **Impossible — one such pair must have been known.**

      $$Pr\left[\{\delta_{i+1} < \delta_i\}\right] = 0$$

Randomised Incremental Construction

Golin, Raman, Schwarz, Smid 1992/1995

Randomised Incremental Construction

Golin, Raman, Schwarz, Smid 1992/1995

- **Expected cost of $\mathscr{P}_i \to \mathscr{P}_{i+1}$:**

Randomised Incremental Construction

Golin, Raman, Schwarz, Smid 1992/1995

- **Expected cost of $\mathscr{P}_i \to \mathscr{P}_{i+1}$:**

$$E\left[T(i)\right] \in \mathcal{O}\left(E\left[T_{insert}(i)\right] \; + \; E\left[T_{query}(i)\right] \; + \; \frac{2}{i+1} \cdot E\left[T_{build}(i+1)\right]\right)$$

Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

- **Expected cost of $\mathscr{P}_i \rightarrow \mathscr{P}_{i+1}$:**

$$E\left[T(i)\right] \in \mathcal{O}\left(E\left[T_{insert}(i)\right] \;+\; E\left[T_{query}(i)\right] \;+\; \frac{2}{i+1} \cdot E\left[T_{build}(i+1)\right]\right)$$

Choice of data structure	$T_{build}(n)$	$T_{insert}(n)$	$T_{query}(n)$

Randomised Incremental Construction

Golin, Raman, Schwarz, Smid 1992/1995

- **Expected cost of $\mathscr{P}_i \to \mathscr{P}_{i+1}$:**

$$E\left[T(i)\right] \in \mathcal{O}\left(E\left[T_{insert}(i)\right] \ + \ E\left[T_{query}(i)\right] \ + \ \frac{2}{i+1} \cdot E\left[T_{build}(i+1)\right] \right)$$

Choice of data structure	$T_{build}(n)$	$T_{insert}(n)$	$T_{query}(n)$
AVL Tree	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(n \log n)$

Randomised Incremental Construction

Golin, Raman, Schwarz, Smid 1992/1995

- **Expected cost of $\mathcal{P}_i \to \mathcal{P}_{i+1}$:**

$$E\big[T(i)\big] \in \mathcal{O}\Big(E\big[T_{insert}(i)\big] \ + \ E\big[T_{query}(i)\big] \ + \ \frac{2}{i+1} \cdot E\big[T_{build}(i+1)\big]\Big)$$

Choice of data structure	$T_{build}(n)$	$T_{insert}(n)$	$T_{query}(n)$
AVL Tree	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(n \log n)$
Dynamic perfect hashing	$\mathcal{O}(1)$(exp.)	$\mathcal{O}(1)$(exp.)	$\mathcal{O}(n)$(exp.)

Randomised Incremental Construction
Golin, Raman, Schwarz, Smid 1992/1995

- **Expected cost of $\mathscr{P}_i \to \mathscr{P}_{i+1}$:**

$$E\left[T(i)\right] \in \mathcal{O}\left(E\left[T_{insert}(i)\right] \;+\; E\left[T_{query}(i)\right] \;+\; \frac{2}{i+1} \cdot E\left[T_{build}(i+1)\right]\right)$$

Choice of data structure	$T_{build}(n)$	$T_{insert}(n)$	$T_{query}(n)$
AVL Tree	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(n \log n)$
Dynamic perfect hashing	$\mathcal{O}(1)$(exp.)	$\mathcal{O}(1)$(exp.)	$\mathcal{O}(n)$(exp.)

Randomised Incremental Construction

Golin, Raman, Schwarz, Smid 1992/1995

- **Expected cost of $\mathscr{P}_i \to \mathscr{P}_{i+1}$:**

$$E\left[T(i)\right] \in \mathcal{O}\left(E\left[T_{insert}(i)\right] \;+\; E\left[T_{query}(i)\right] \;+\; \frac{2}{i+1} \cdot E\left[T_{build}(i+1)\right]\right)$$

Choice of data structure	$T_{build}(n)$	$T_{insert}(n)$	$T_{query}(n)$
AVL Tree	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(n \log n)$
Dynamic perfect hashing	$\mathcal{O}(1)$(exp.)	$\mathcal{O}(1)$(exp.)	$\mathcal{O}(n)$(exp.)

- AVL Tree: $\sum_{i=2}^{n-1} E\left[T(i)\right] \in \mathcal{O}\left(\sum_{i=2}^{n-1} \left(\log i + \log i + \frac{(i+1)\log(i+1)}{i+1}\right)\right) = \mathcal{O}\left(n \log n\right)$

Randomised Incremental Construction

Golin, Raman, Schwarz, Smid 1992/1995

- **Expected cost of $\mathcal{P}_i \to \mathcal{P}_{i+1}$:**

$$E\left[T(i)\right] \in \mathcal{O}\left(E\left[T_{insert}(i)\right] \;+\; E\left[T_{query}(i)\right] \;+\; \frac{2}{i+1} \cdot E\left[T_{build}(i+1)\right]\right)$$

Choice of data structure	$T_{build}(n)$	$T_{insert}(n)$	$T_{query}(n)$
AVL Tree	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(n \log n)$
Dynamic perfect hashing	$\mathcal{O}(1)$(exp.)	$\mathcal{O}(1)$(exp.)	$\mathcal{O}(n)$(exp.)

- AVL Tree: $\sum_{i=2}^{n-1} E\left[T(i)\right] \in \mathcal{O}\left(\sum_{i=2}^{n-1} \left(\log i + \log i + \frac{(i+1)\log(i+1)}{i+1}\right)\right) \quad = \mathcal{O}\left(n \log n\right)$
- Hashing: $\sum_{i=2}^{n-1} E\left[T(i)\right] \in \mathcal{O}\left(\sum_{i=2}^{n-1} \left(1 + 1 + \frac{i+1}{i+1}\right)\right) \quad = \mathcal{O}\left(n\right)$

Thank you.

References

[Bentley und Shamos, 1976] Bentley, Jon Louis und Michael Ian Shamos:
Divide-and-Conquer in multidimensional space. In: Proceedings of the Eighth Annual AXM Symposium on the Theory of Computation, S. 220-230, Association for Computing Machinery, 1976.

[Golin et al., 1995] Golin, Mordecai J., Rajeev Raman, Christian Schwarz Michiel Smid:
Simple randomized algorithms for closest pair problems. Nordic Journal of Computing 2(1):3-27, 1995.

[Hinrichs et al., 1988] Hinrichs, Klaus Helmer, Jürg Nievergelt und Peter Schom:
Plane-sweep solves the closest problem elegantly. Information Processing Letters 26(5): 255-261, 1988.

[Preparata und Shamos, 1988] Preparata, Franco P. und Michael Ian Shamos:
Computational Geometry: An Introduction. Springer, Berlin, 2. Edition, 1988.

[Shamos und Hoey, 1976] Shamos, Michael Ian und Daniel J. Hoey:
Geometric intersection problems. In: Proceedings of the 17th Annual Symposium on the Foundation of Computer Science, S. 208-215, IEEE Computer Press, 1976.