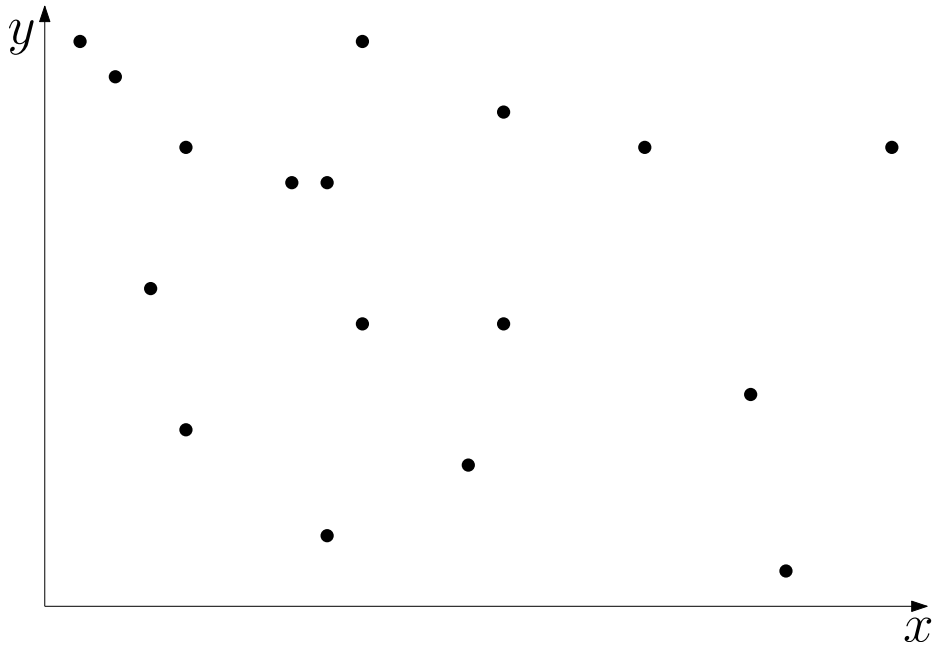
Time-Windowed Closest Pair

Timothy M. Chan,
Simon Pratt

Cheriton School of Computer Science
University of Waterloo

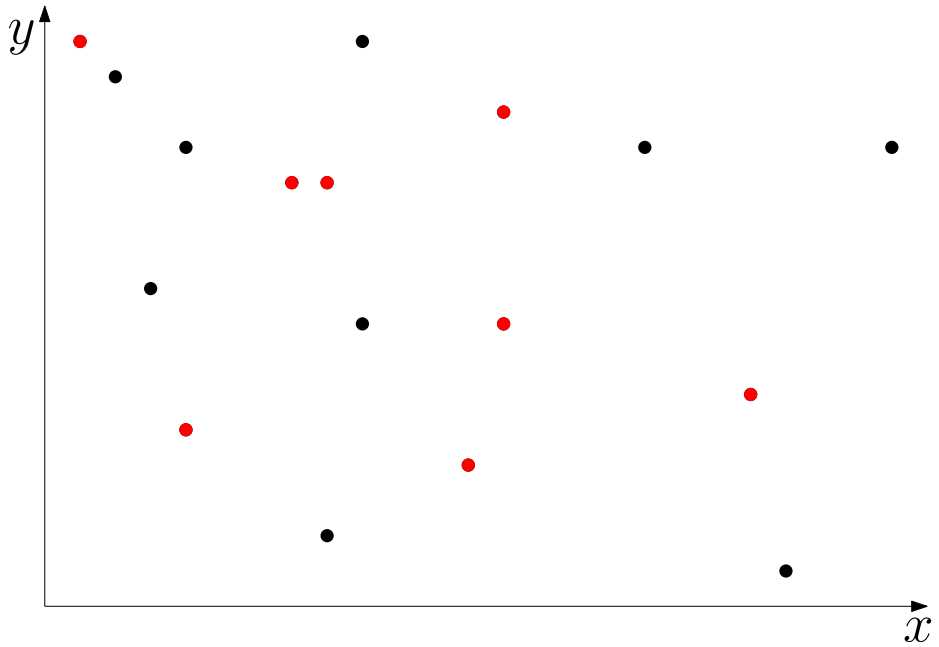
27^{th} Canadian Conference on Computational Geometry, 2015

Problem



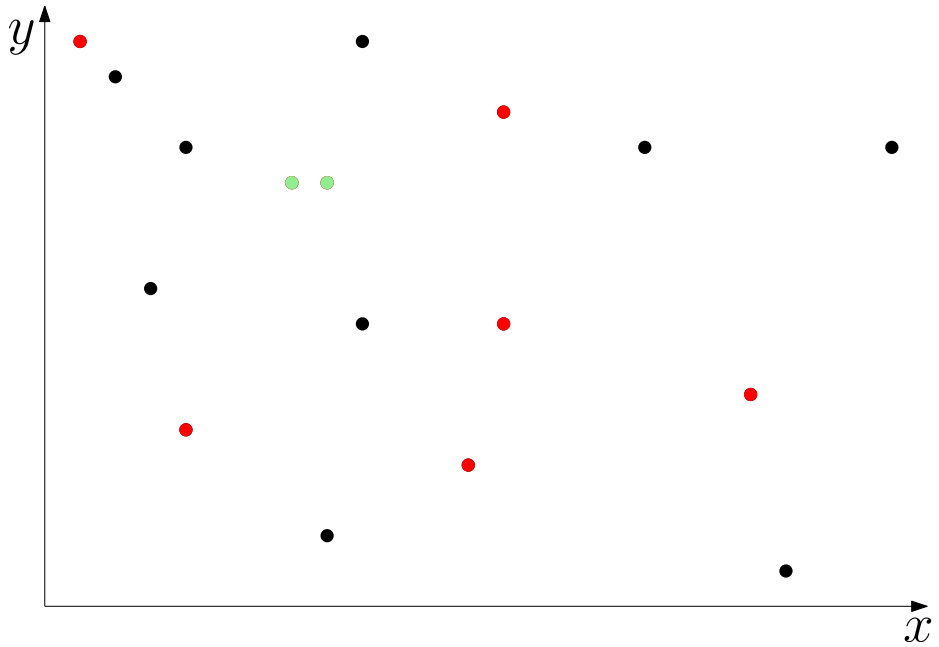
- Point set P
- Each point $p \in P$ associated with a *distinct* time t_p
- Given a query time window $[t_1, t_2]$, find the closest pair of points whose associated times are within that window

Problem



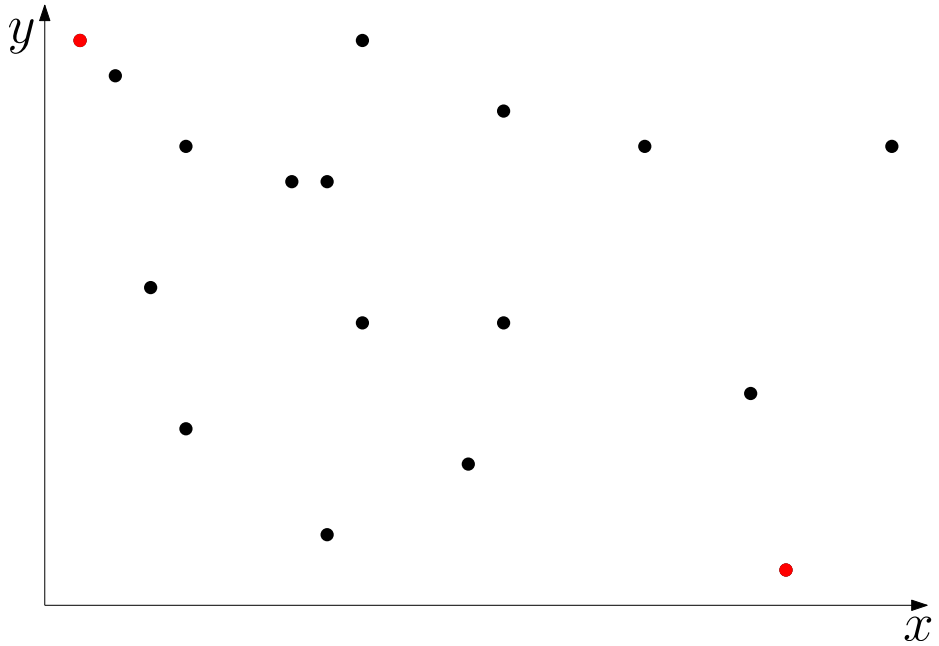
- Point set P
- Each point $p \in P$ associated with a *distinct* time t_p
- Given a query time window $[t_1, t_2]$, find the closest pair of points whose associated times are within that window

Problem



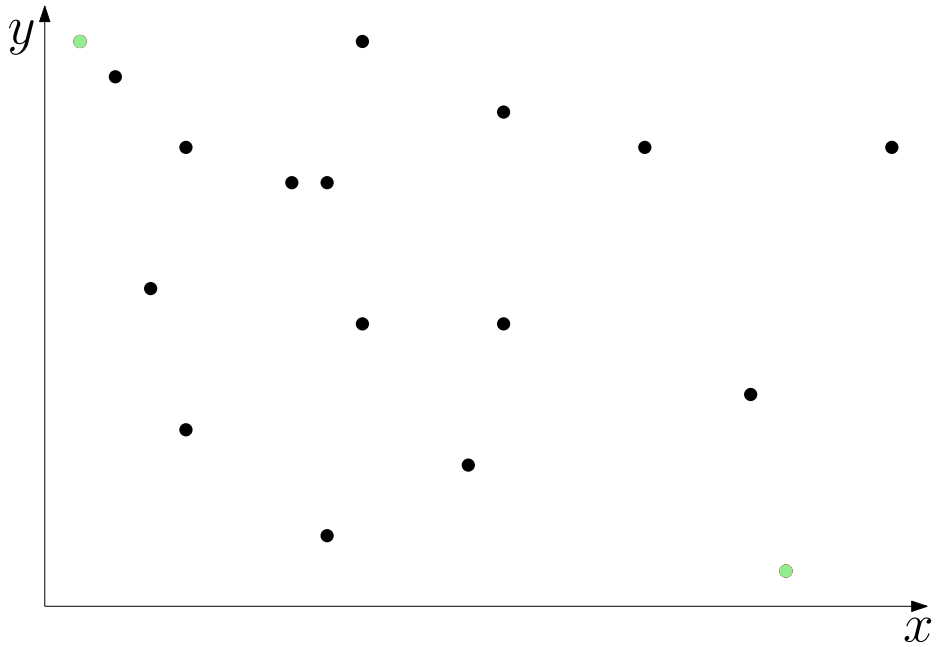
- Point set P
- Each point $p \in P$ associated with a *distinct* time t_p
- Given a query time window $[t_1, t_2]$, find the closest pair of points whose associated times are within that window

Problem



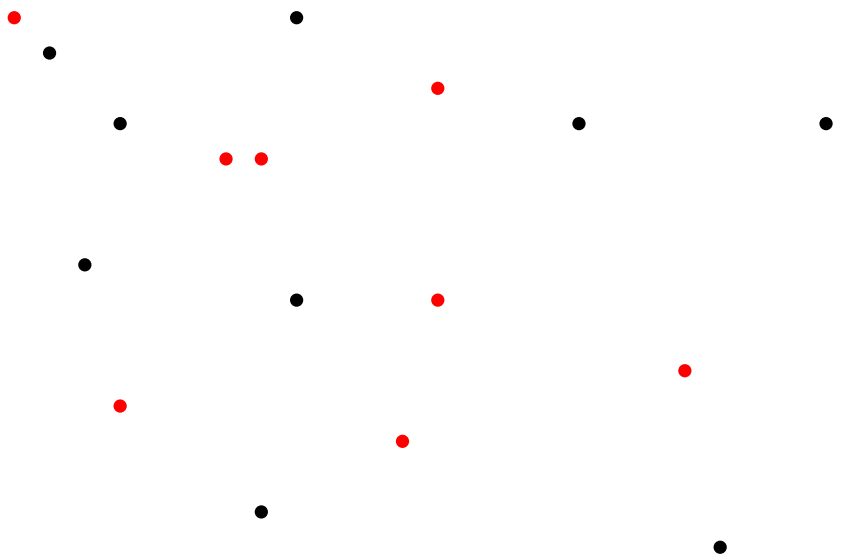
- Point set P
- Each point $p \in P$ associated with a *distinct* time t_p
- Given a query time window $[t_1, t_2]$, find the closest pair of points whose associated times are within that window

Problem



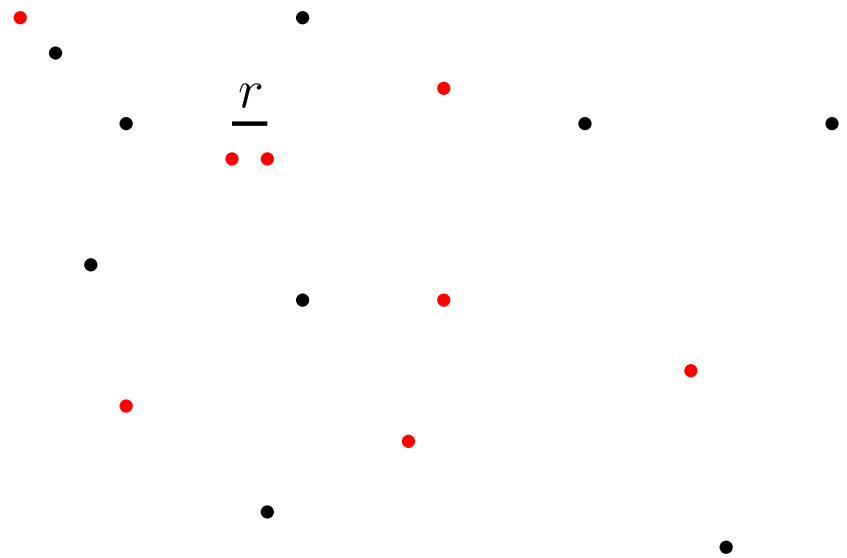
- Point set P
- Each point $p \in P$ associated with a *distinct* time t_p
- Given a query time window $[t_1, t_2]$, find the closest pair of points whose associated times are within that window

Decision Problem



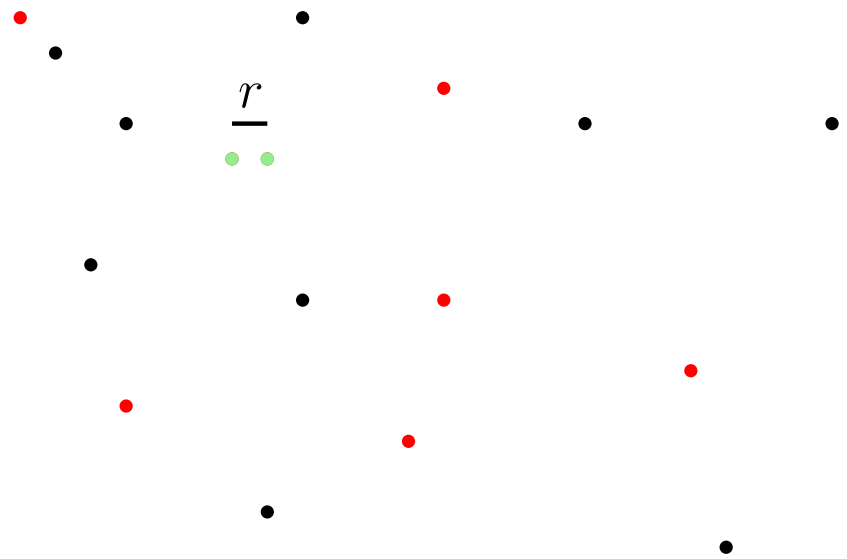
- Point set P
- Each point $p \in P$ associated with a *distinct* time t_p
- Fixed constant r
- Given a query time window $[t_1, t_2]$:
  - Determine if there exists 2 points at most r distance apart

Decision Problem



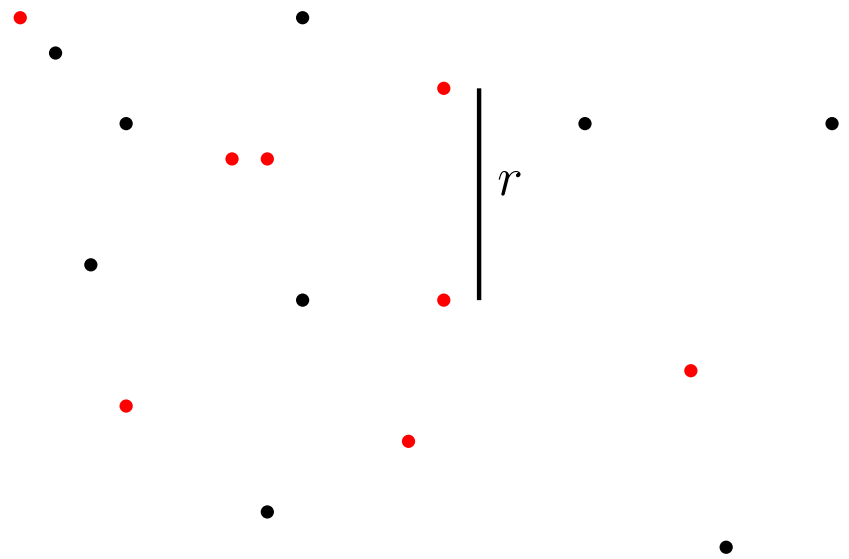
- Point set P
- Each point $p \in P$ associated with a *distinct* time t_p
- Fixed constant r
- Given a query time window $[t_1, t_2]$:
  - Determine if there exists 2 points at most r distance apart

Decision Problem



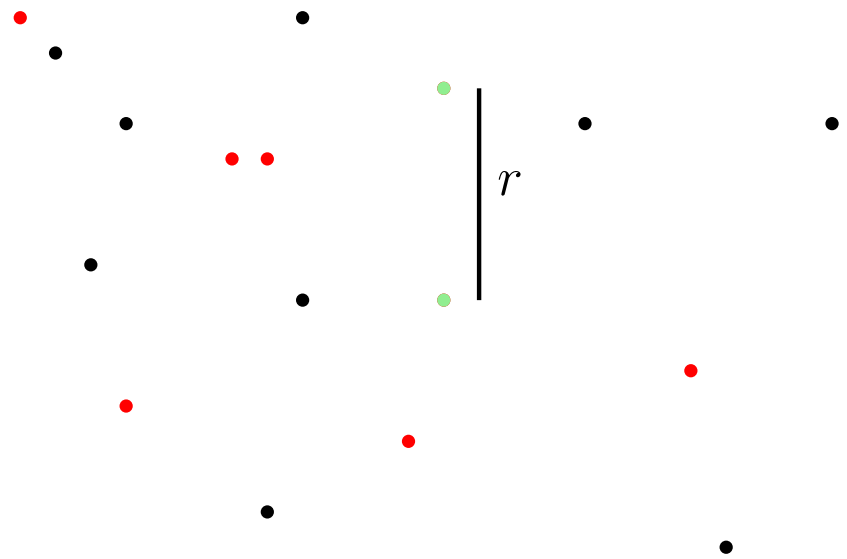
- Point set P
- Each point $p \in P$ associated with a *distinct* time t_p
- Fixed constant r
- Given a query time window $[t_1, t_2]$:
  - Determine if there exists 2 points at most r distance apart

Decision Problem



- Point set P
- Each point $p \in P$ associated with a *distinct* time t_p
- Fixed constant r
- Given a query time window $[t_1, t_2]$:
  - Determine if there exists 2 points at most r distance apart

Decision Problem



- Point set P
- Each point $p \in P$ associated with a *distinct* time t_p
- Fixed constant r
- Given a query time window $[t_1, t_2]$:
  - Determine if there exists 2 points at most r distance apart

Assumptions

- w-bit word-RAM model
- Dimension d is a constant with respect to n
- Times are distinct
- Times are given in rank space $(0 \ldots n-1)$

Assumptions

- w-bit word-RAM model
- Dimension d is a constant with respect to n
- Times are distinct
- Times are given in rank space $(0 \ldots n-1)$
  - If time given as integers, pre-sort and add the cost of a predecessor search to query time:
    $O\left(\min\left\{\log w, \log_w n, \sqrt{\log n / \log\log n}\right\}\right)$

Credit: *Pătraşcu 2008*

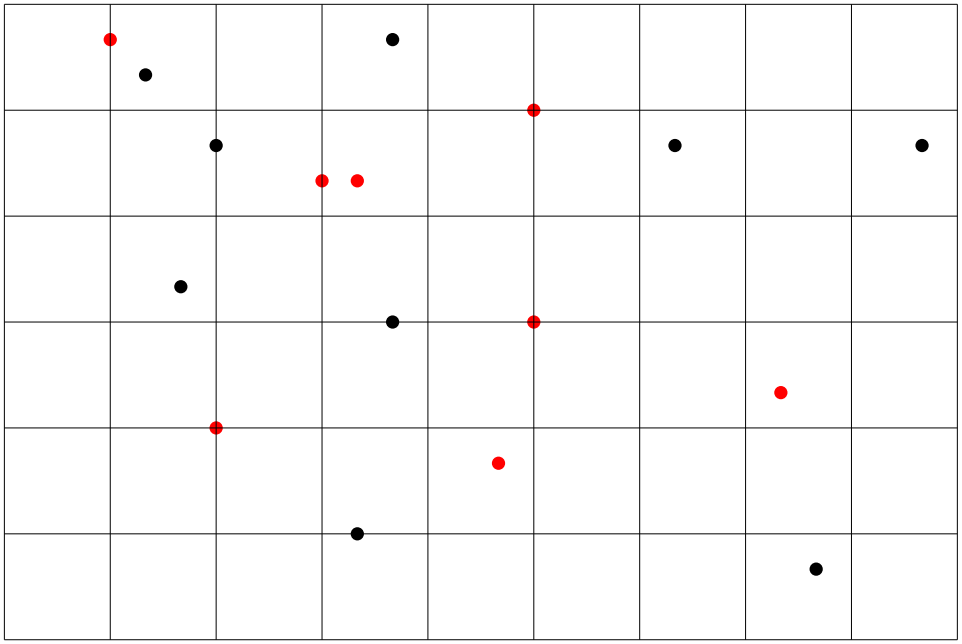
Results and Overview

Problem	Pre-Processing	Space	Query
Decision Closest Pair	$O(n)$ (expected)	$O(n)$ bits	$O(1)$
	Grid	Succinct Rank/Select	
Closest Pair	$O(n \log n \log \log n)$	$O(n \log n)$ words	$O(\log \log n)$
	Quadtree	Orthogonal Point Location	

Decision Problem: first step

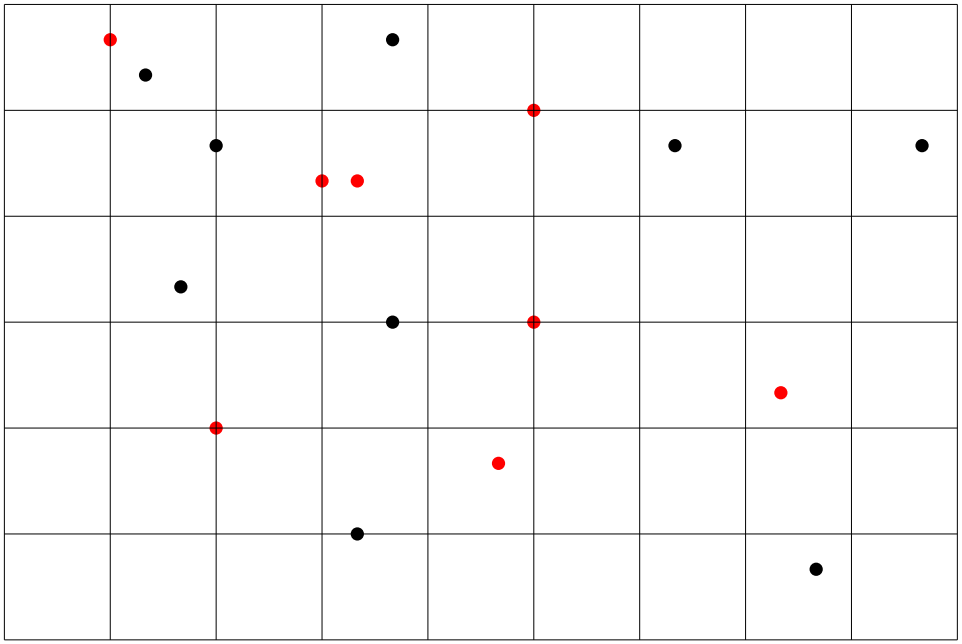
- Aiming for $O(n)$ pre-processing
- Must reduce the number of *candidate satisfying pairs* to $O(n)$

Computing Candidate Satisfying Pairs



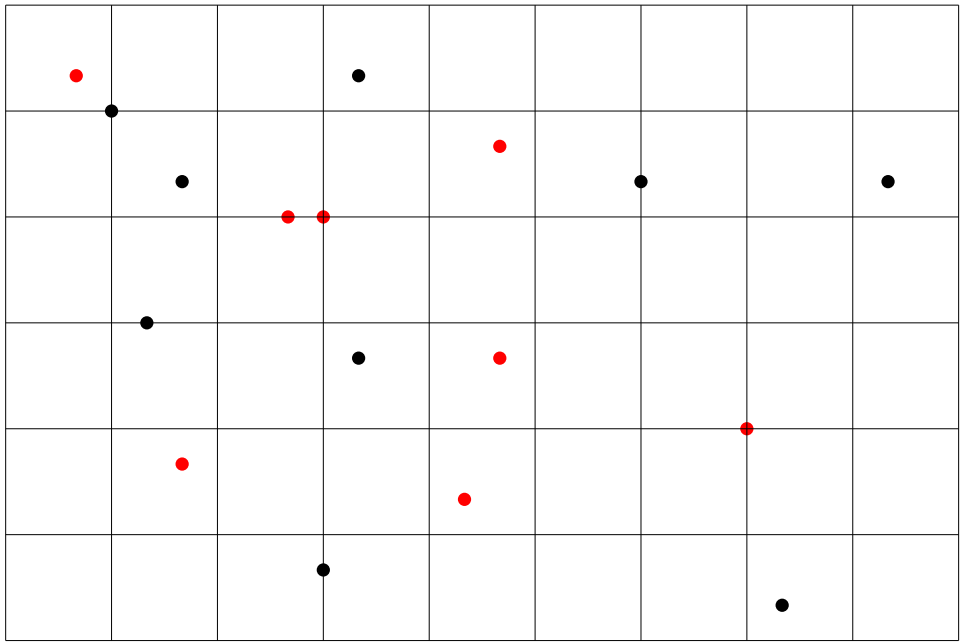
1. Draw a grid over the points
2. Sort points in each grid cell by time
3. For each point:
   1. Consider a constant number of time-order pred. and succ.

What about points that aren't in the same cell?



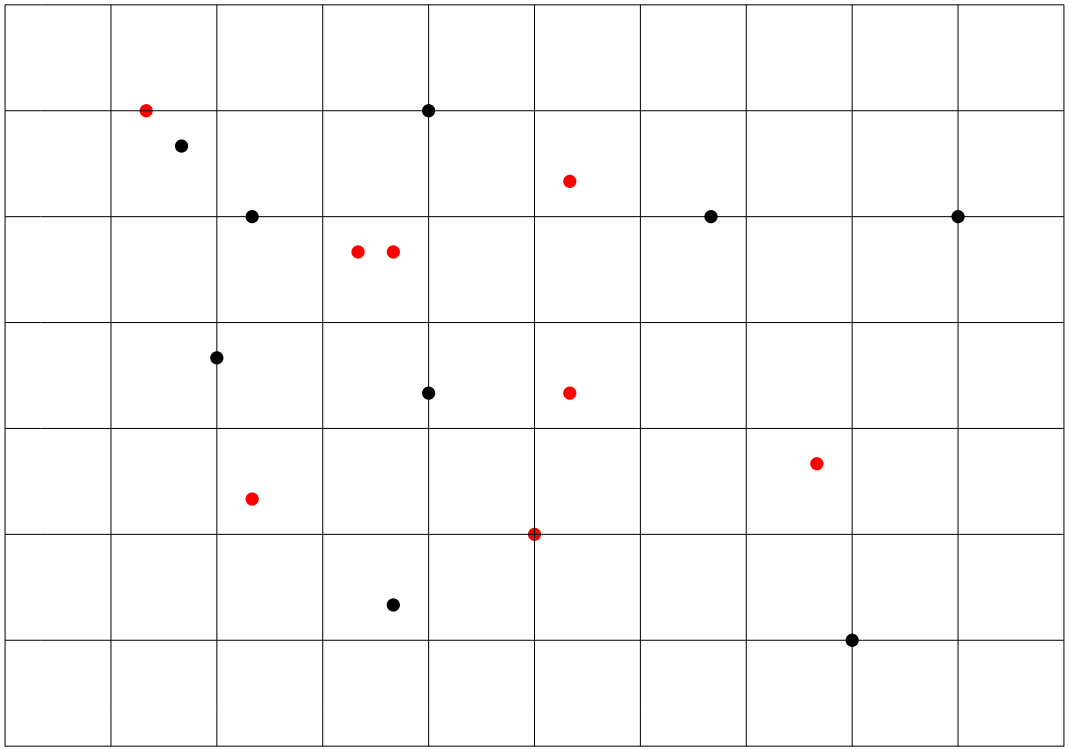
- We draw $d + 1$ grids, each shifted by some constant
- Any pair of points at most r distance apart will be in the same cell in one of the grids

What about points that aren't in the same cell?



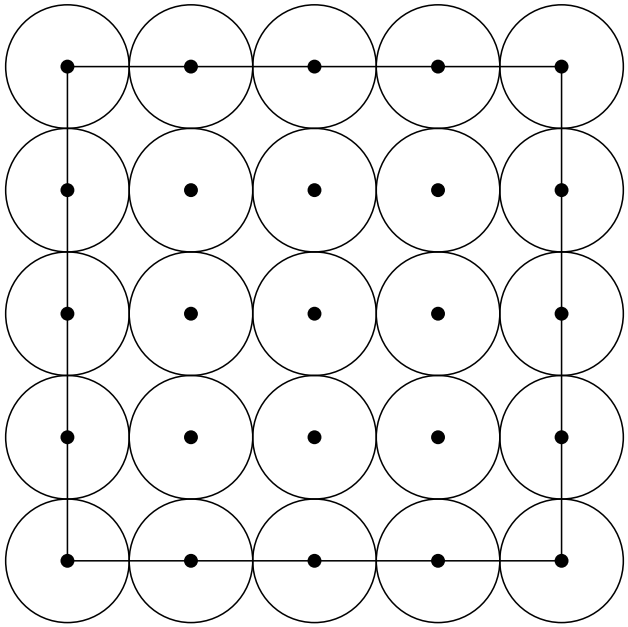
- We draw $d + 1$ grids, each shifted by some constant
- Any pair of points at most r distance apart will be in the same cell in one of the grids

What about points that aren't in the same cell?



- We draw $d + 1$ grids, each shifted by some constant
- Any pair of points at most r distance apart will be in the same cell in one of the grids

How many neighbors do we need to consider?

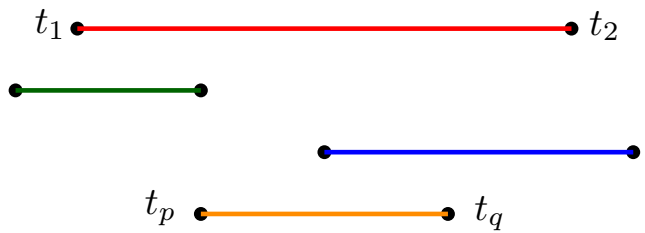


Packing argument $\rightarrow$ constant that depends only on d

Now What?

- $O(n)$ pairs
- How do we find a satisfying pair for a query?

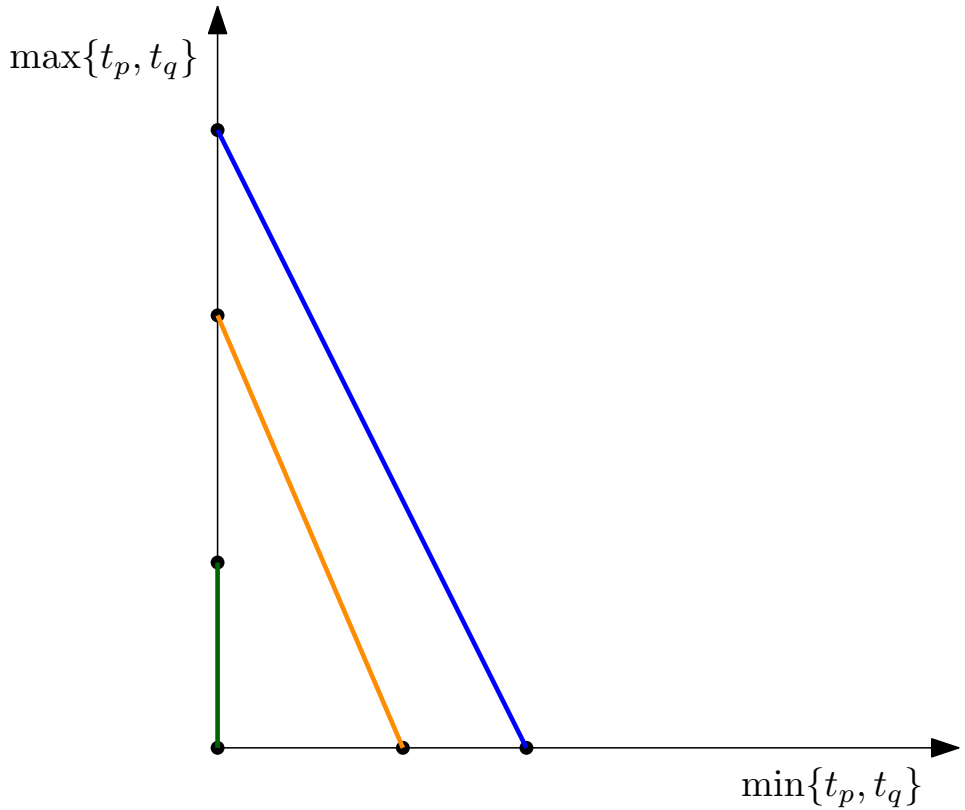
Re-state the Problem



Given a query time window $[t_1, t_2]$, determine if there exists a pair pq such that:

- $t_1 \leq \min\{t_p, t_q\} \leq \max\{t_p, t_q\} \leq t_2$

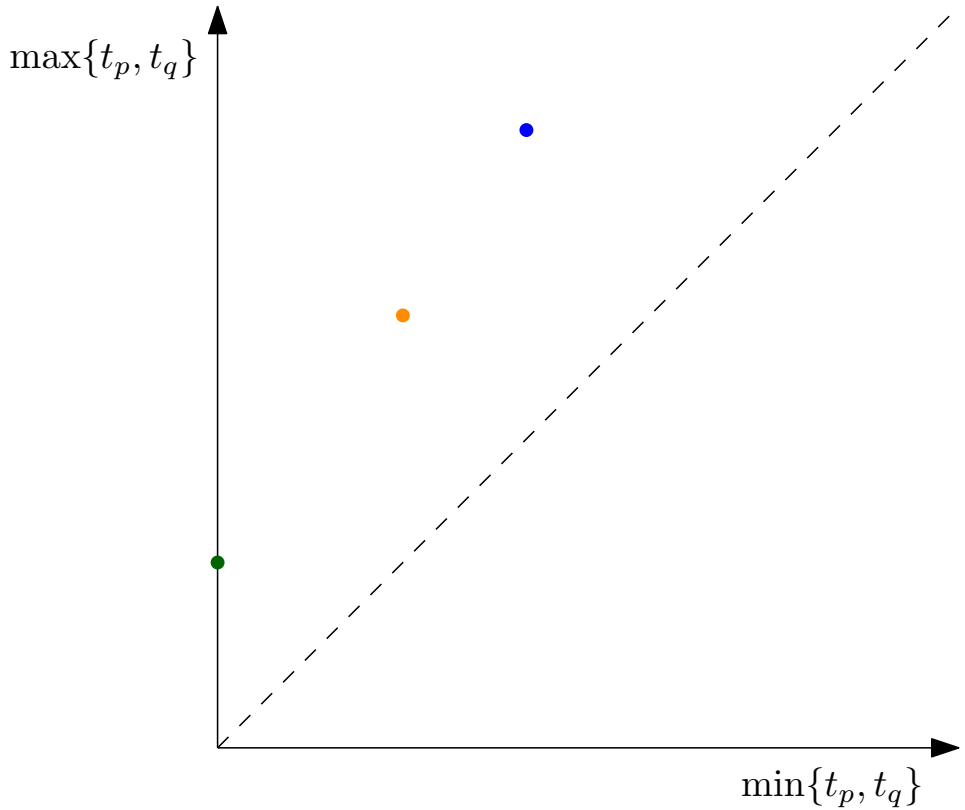
Re-state the Problem



Given a query time window $[t_1, t_2]$, determine if there exists a pair pq such that:

- $t_1 \leq \min\{t_p, t_q\} \leq \max\{t_p, t_q\} \leq t_2$

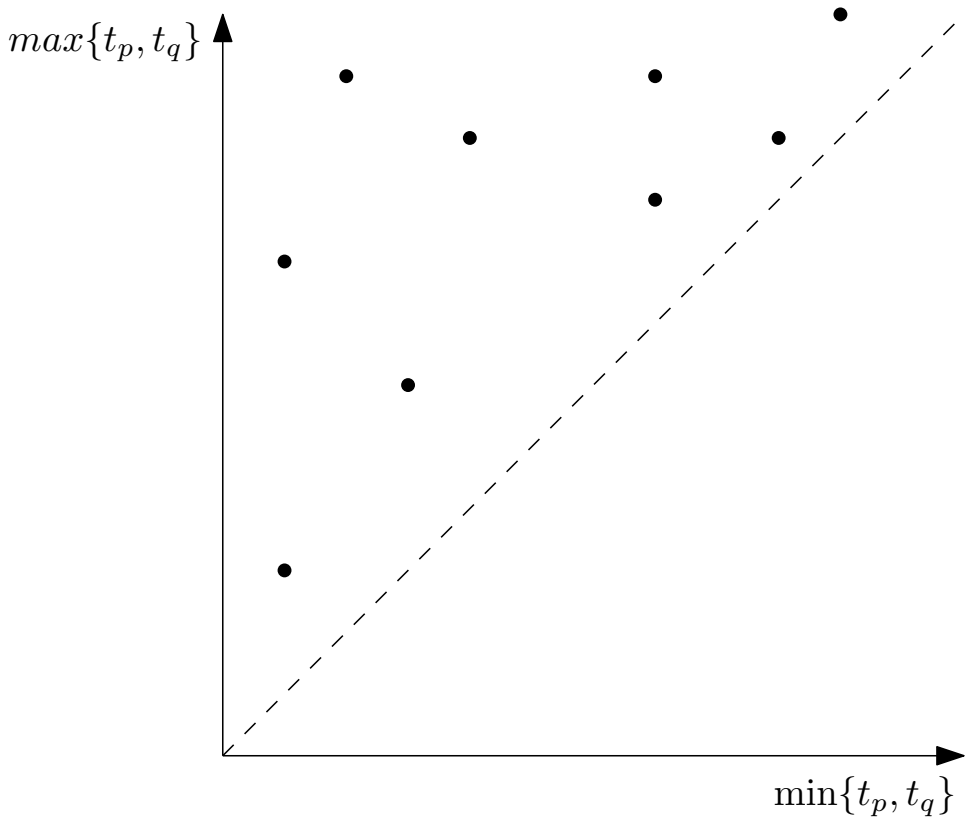
Re-state the Problem



Given a query time window $[t_1, t_2]$, determine if there exists a pair pq such that:

- $t_1 \leq \min\{t_p, t_q\} \leq \max\{t_p, t_q\} \leq t_2$

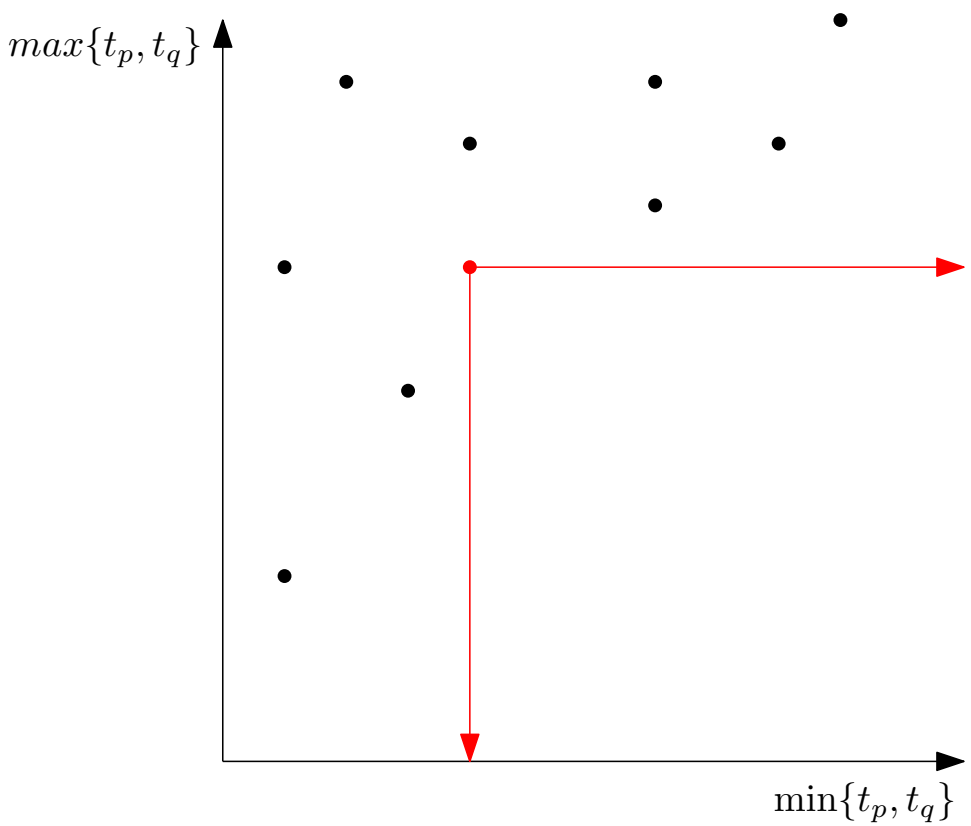
Re-state the Problem



Given a query time window $[t_1, t_2]$, determine if there exists a pair pq such that:

- $t_1 \leq \min\{t_p, t_q\} \leq \max\{t_p, t_q\} \leq t_2$

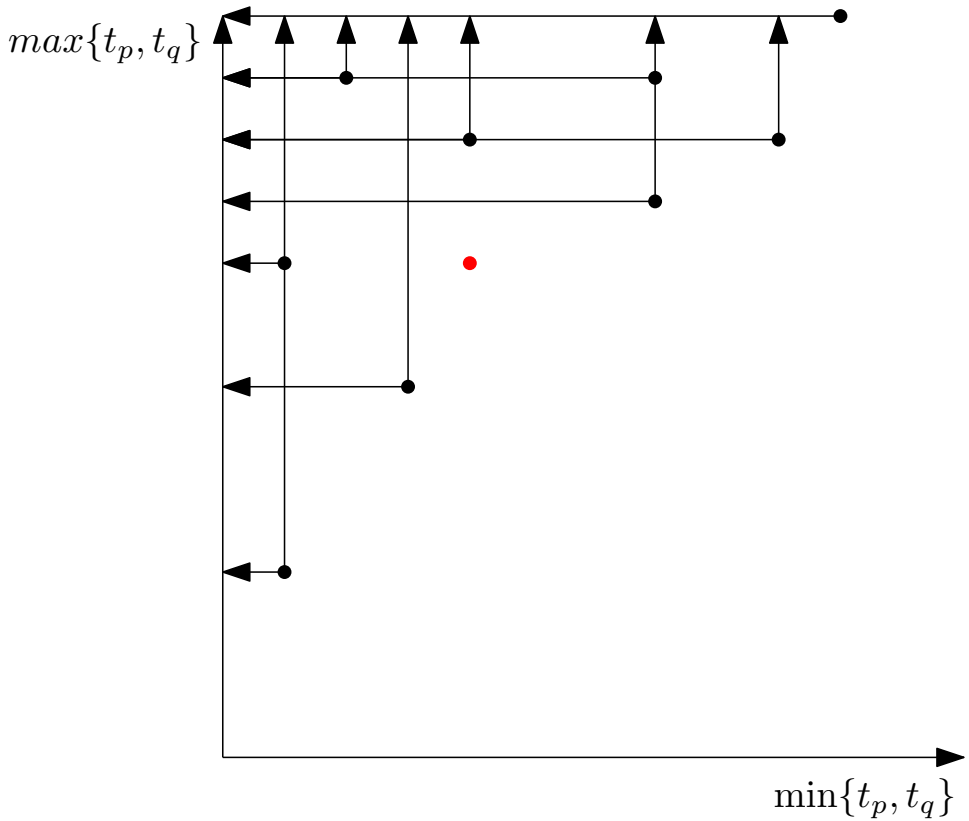
Solution



A query is:

- The corner of a quadrant
- Satisfied if the quadrant is not empty

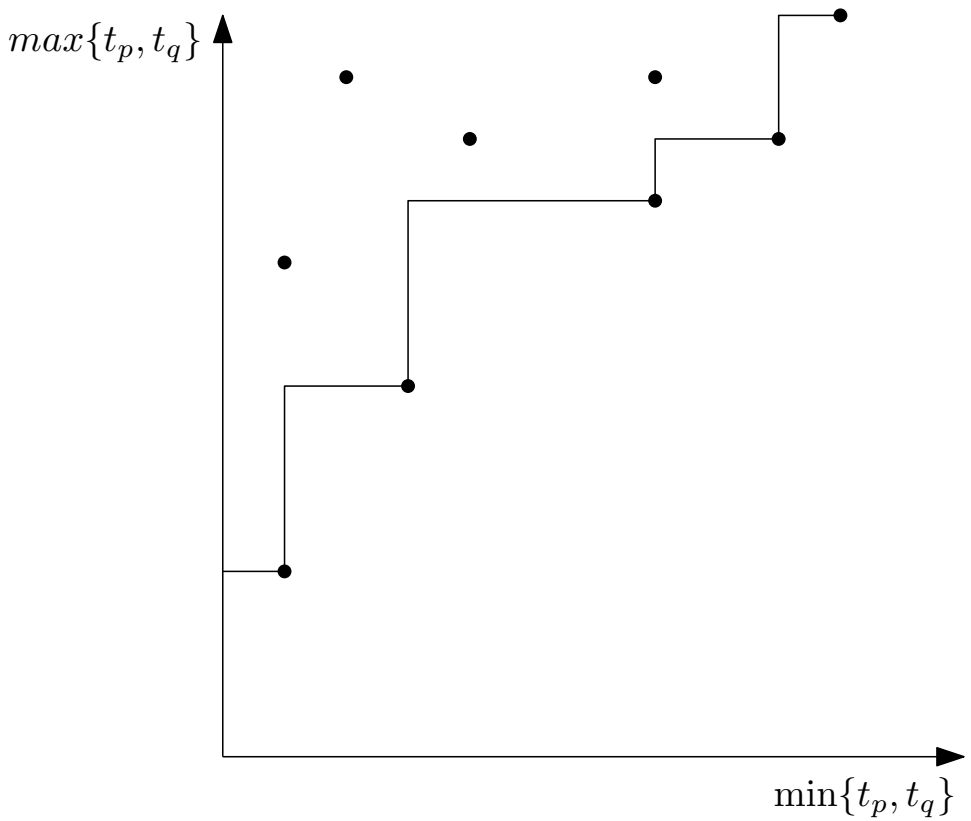
Solution



Points:

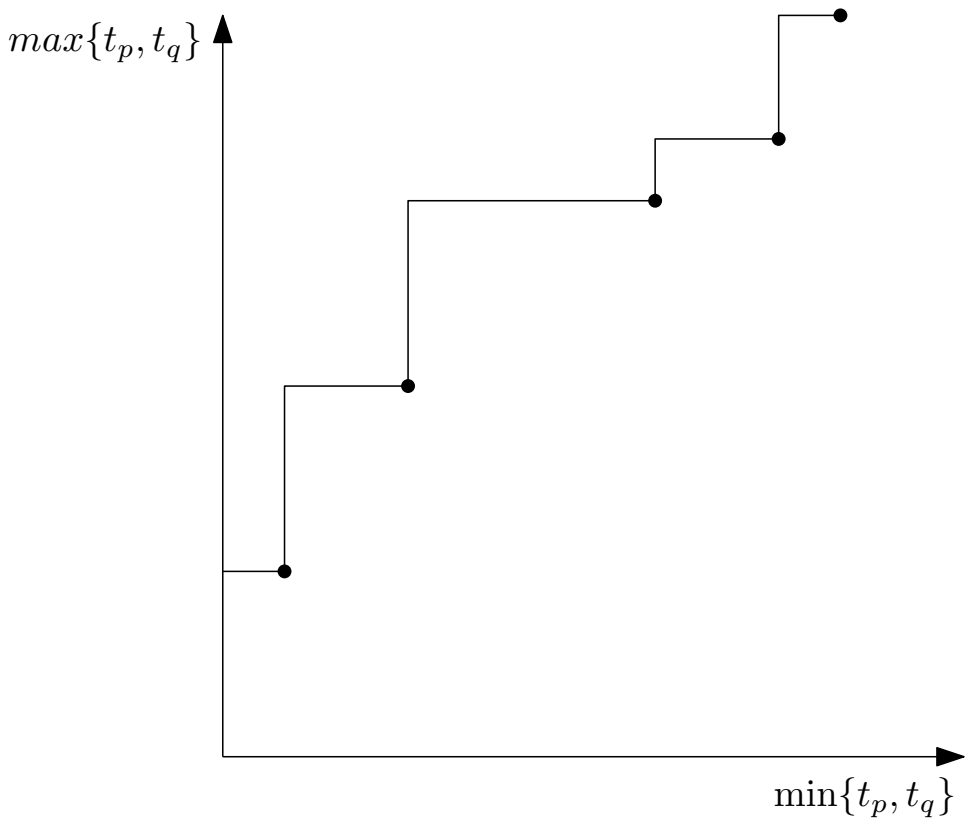
- Are the corner of a quadrant
- Satisfy any query within the boundary of that quadrant

Solution



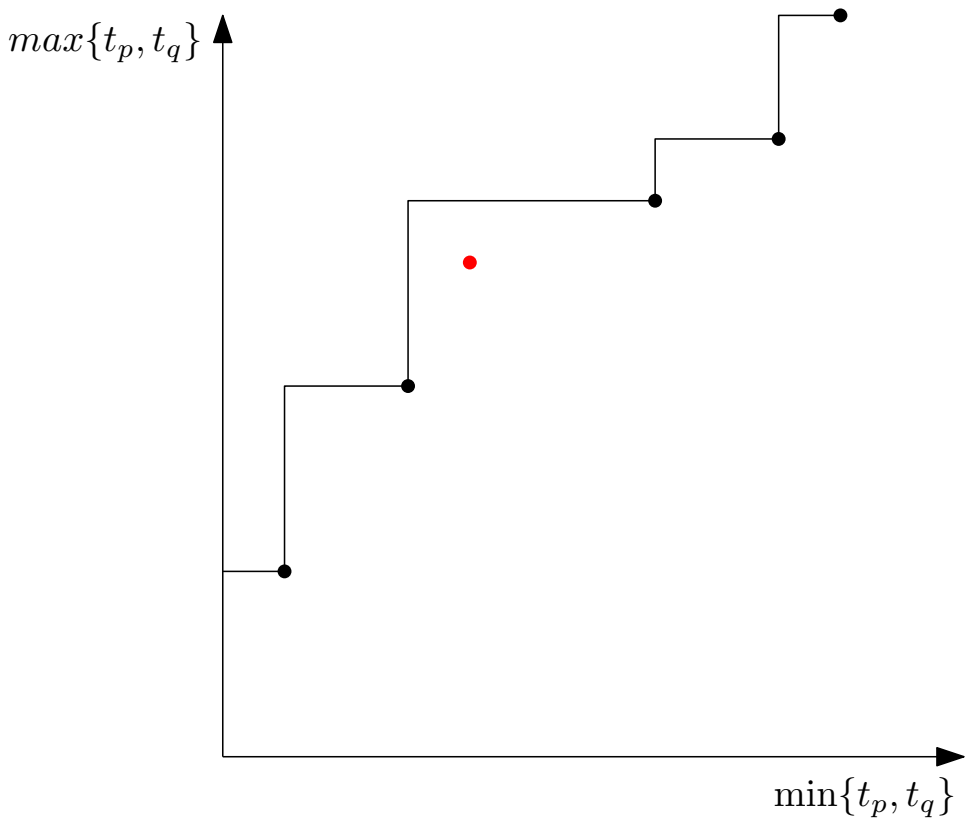
- Compute the staircase or minima of the time points

Solution



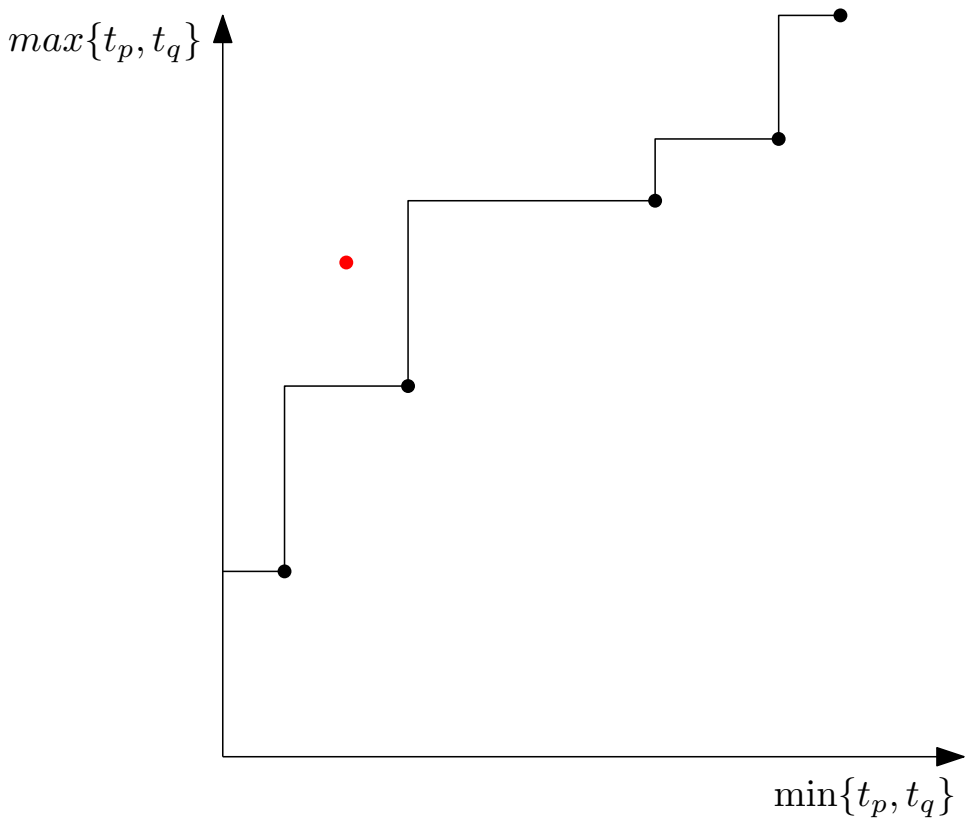
- Compute the staircase or minima of the time points
- We store this staircase in an array

Solution



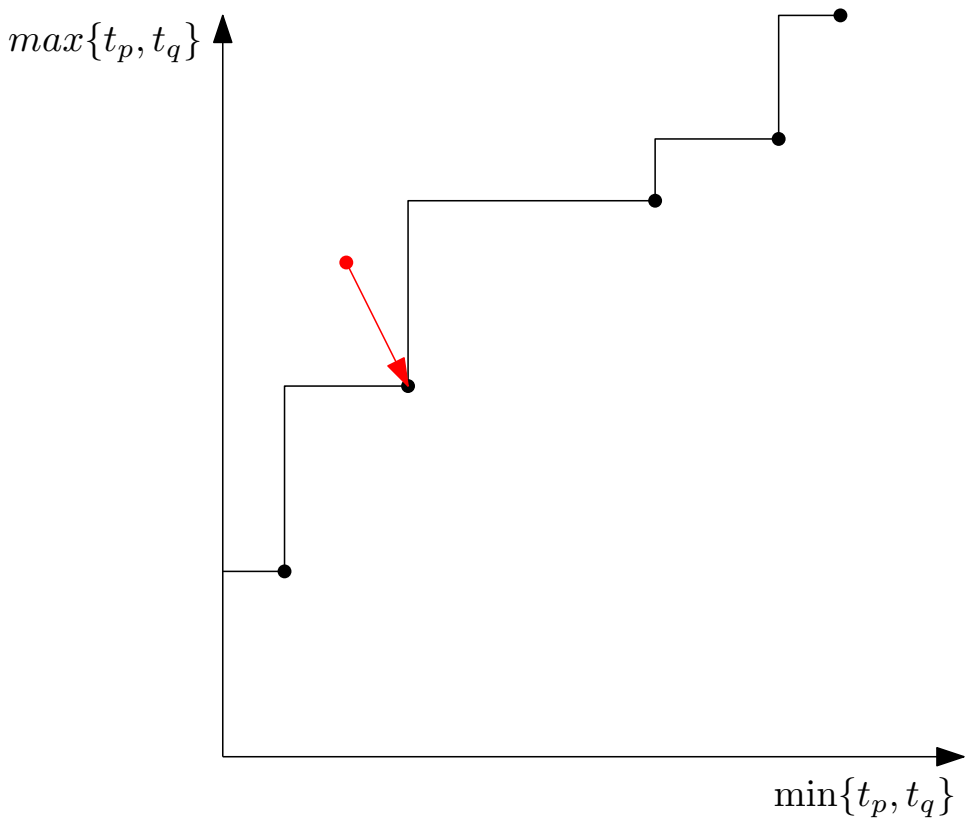
- Compute the staircase or minima of the time points
- We store this staircase in an array
- A query is satisfied iff it is above this staircase

Solution



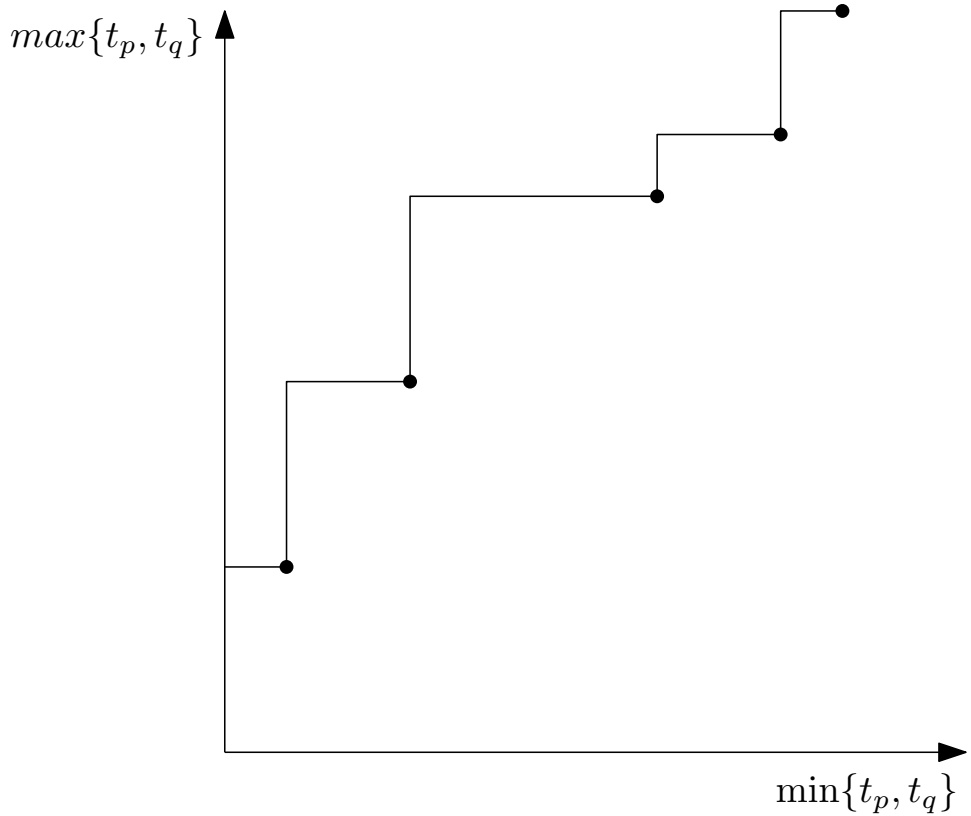
- Compute the staircase or minima of the time points
- We store this staircase in an array
- A query is satisfied iff it is above this staircase

Solution



- Compute the staircase or minima of the time points
- We store this staircase in an array
- A query is satisfied iff it is above this staircase

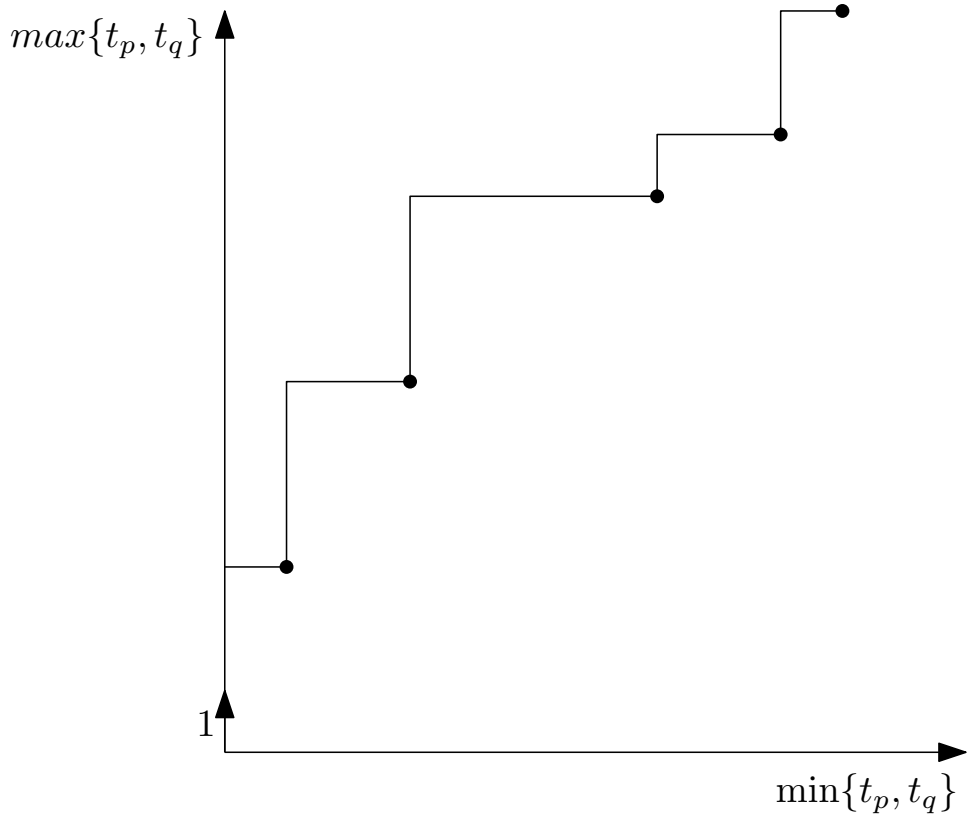
Succinct Solution



- Store the staircase as bits representing a monotone chain
- Succinct rank/select structure
- 0 $\rightarrow$, 1 $\uparrow$

Credit: *Munro 1996*

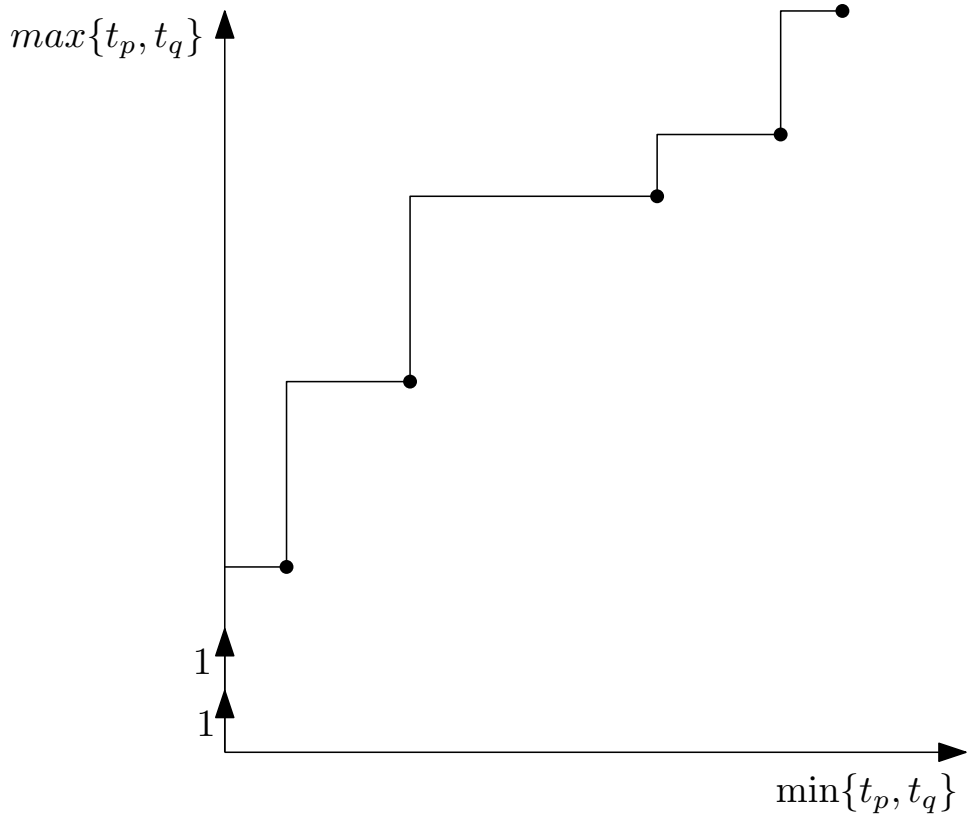
Succinct Solution



- Store the staircase as bits representing a monotone chain
- Succinct rank/select structure
- 0 $\rightarrow$, 1 $\uparrow$

Credit: *Munro 1996*

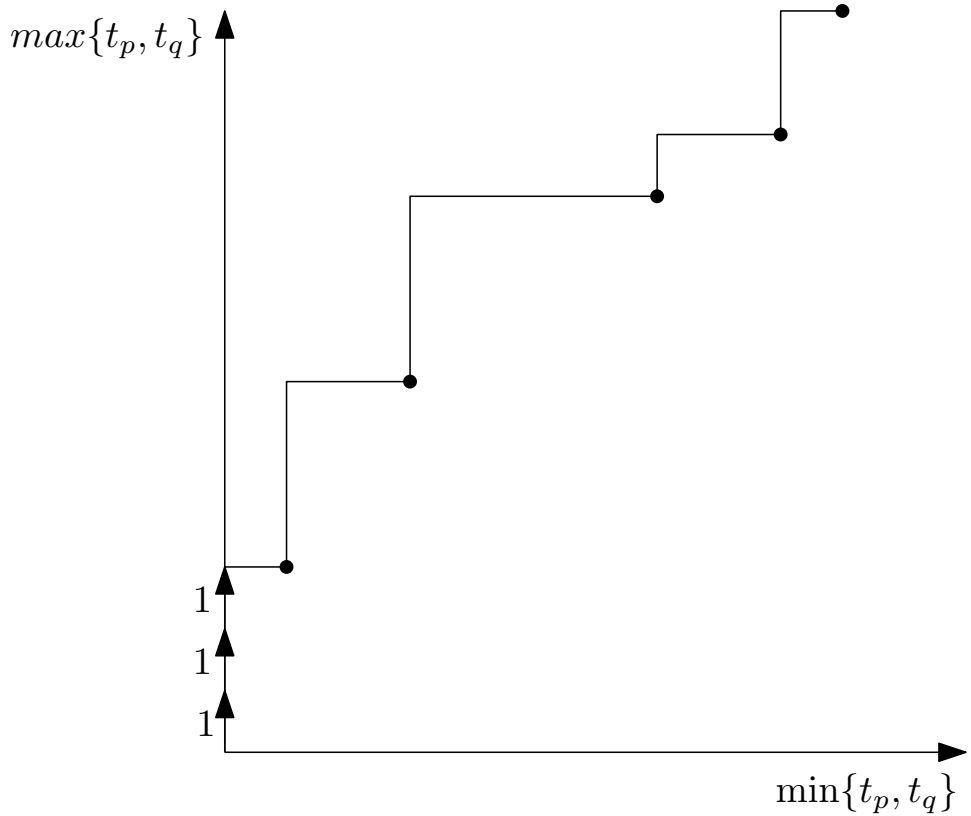
Succinct Solution



- Store the staircase as bits representing a monotone chain
- Succinct rank/select structure
- 0 $\rightarrow$, 1 $\uparrow$

Credit: *Munro 1996*

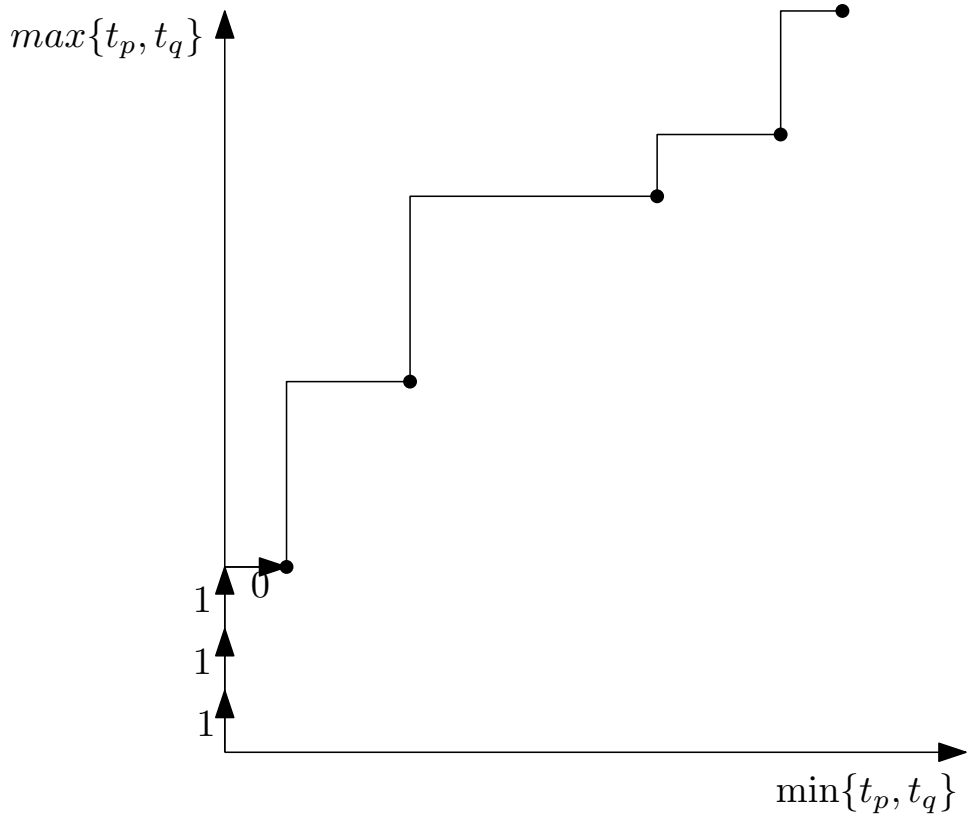
Succinct Solution



- Store the staircase as bits representing a monotone chain
- Succinct rank/select structure
- 0 $\rightarrow$, 1 $\uparrow$

Credit: *Munro 1996*

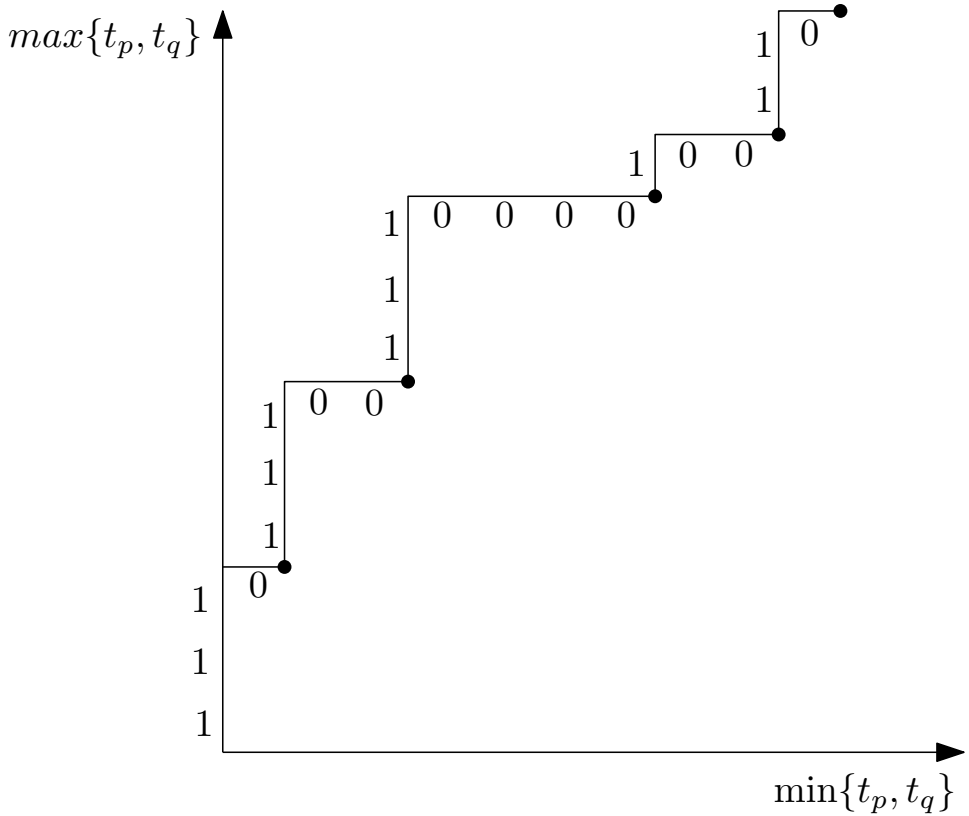
Succinct Solution



- Store the staircase as bits representing a monotone chain
- Succinct rank/select structure
- 0 $\rightarrow$, 1 $\uparrow$

Credit: *Munro 1996*

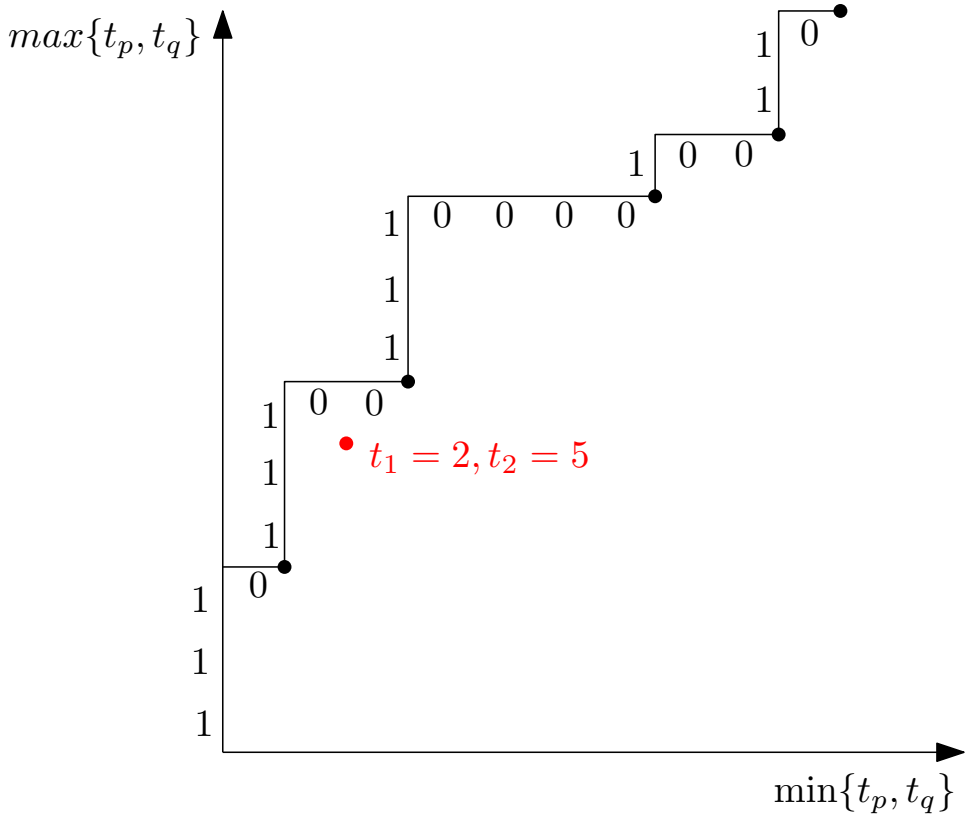
Succinct Solution



- Store the staircase as bits representing a monotone chain
- Succinct rank/select structure
- 0 $\rightarrow$, 1 $\uparrow$

Credit: *Munro 1996*

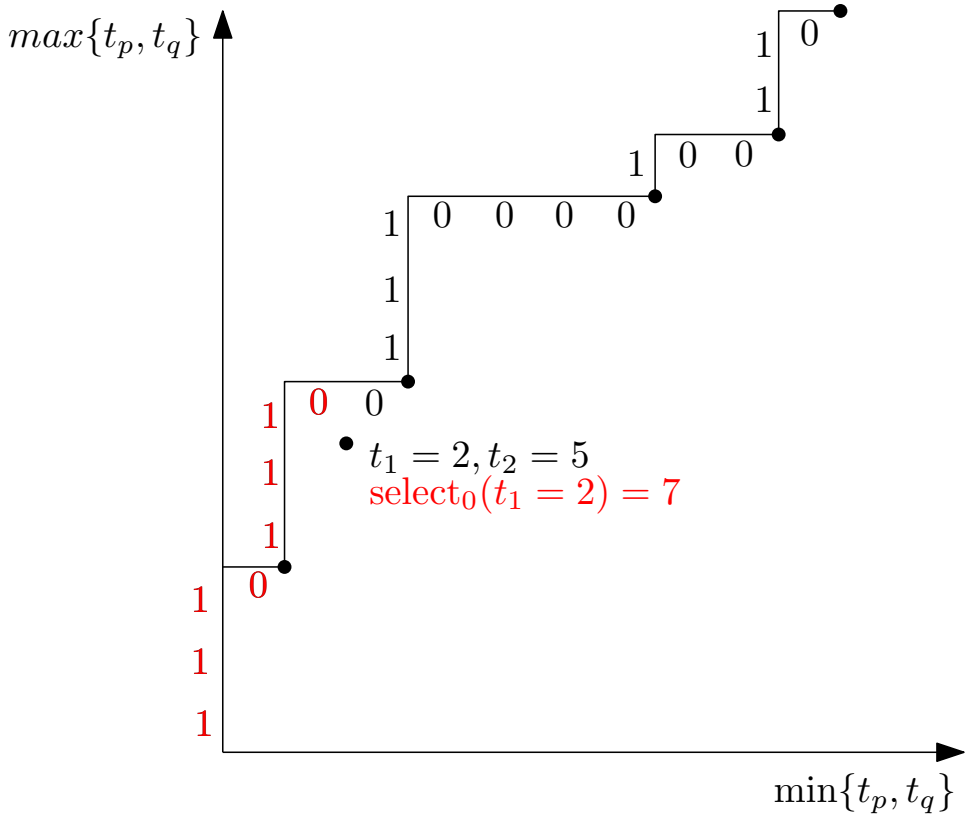
Succinct Solution



- Store the staircase as bits representing a monotone chain
- Succinct rank/select structure
- 0 $\rightarrow$, 1 $\uparrow$

Credit: *Munro 1996*

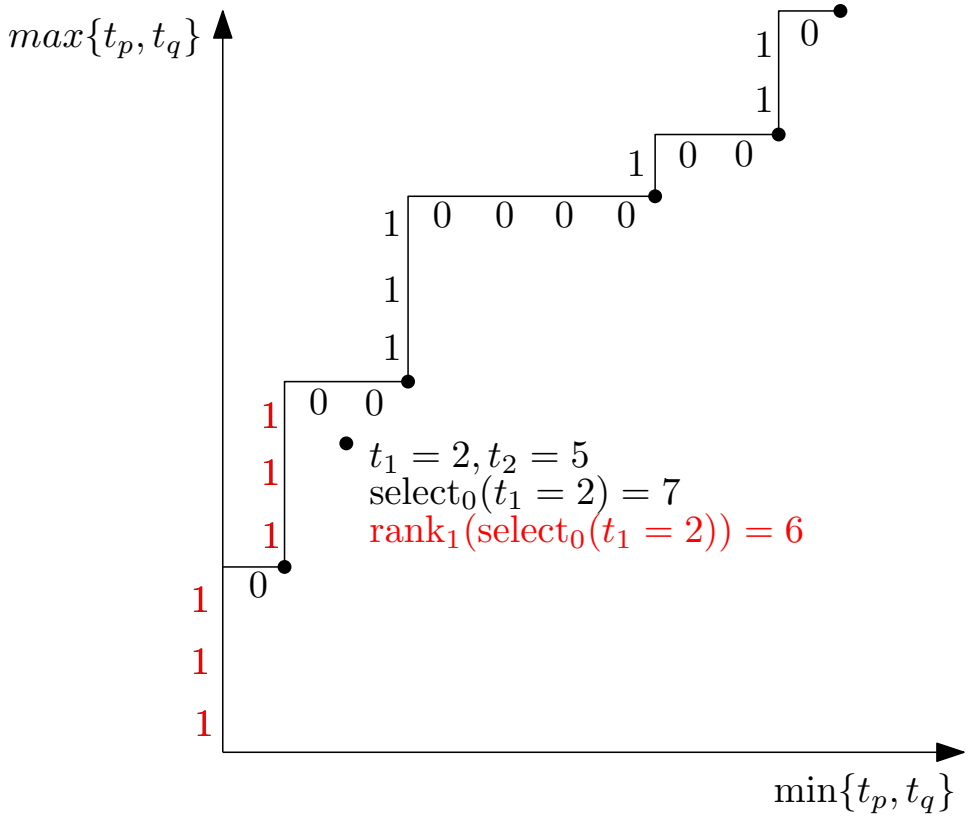
Succinct Solution



- Store the staircase as bits representing a monotone chain
- Succinct rank/select structure
- 0 $\rightarrow$, 1 $\uparrow$

Credit: *Munro 1996*

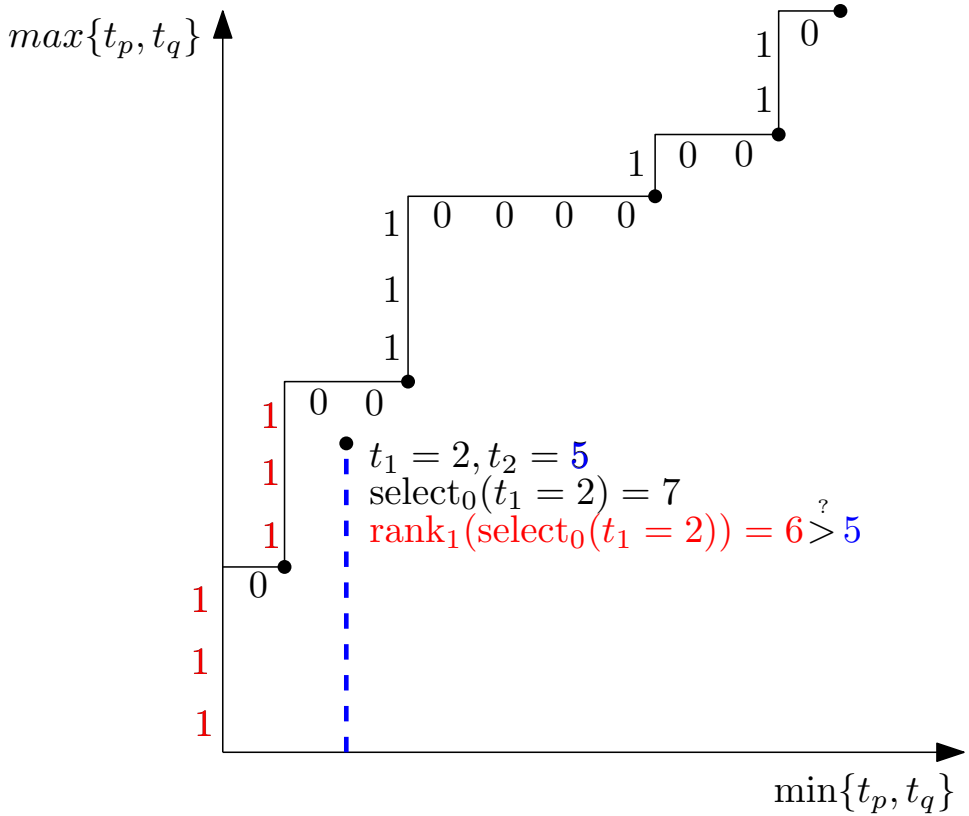
Succinct Solution



- Store the staircase as bits representing a monotone chain
- Succinct rank/select structure
- 0 $\rightarrow$, 1 $\uparrow$

Credit: *Munro 1996*

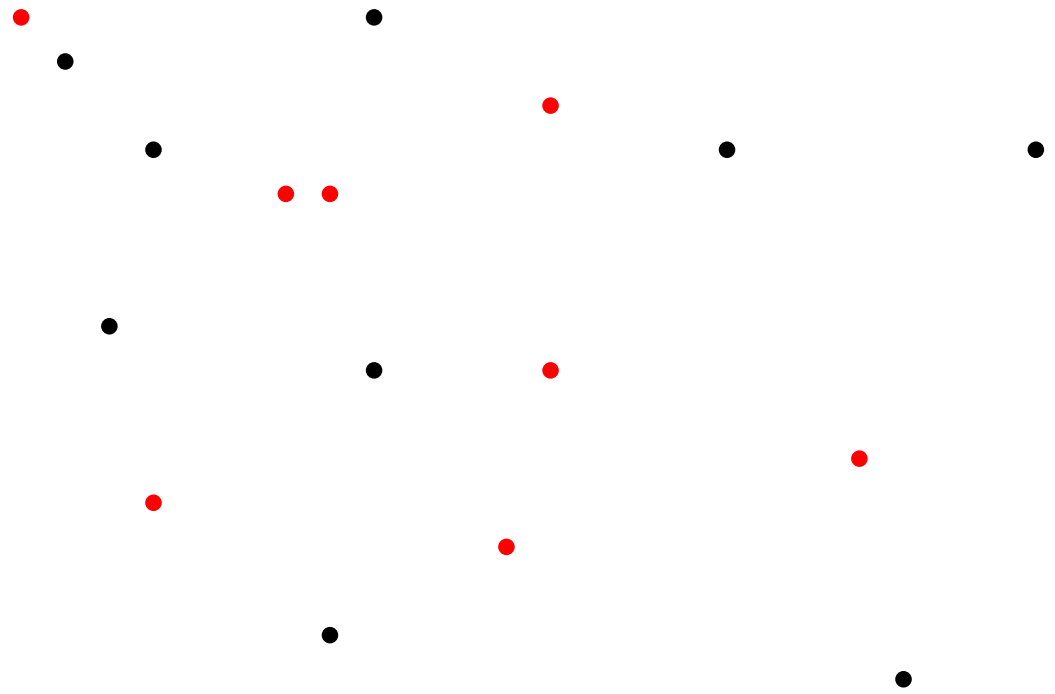
Succinct Solution



- Store the staircase as bits representing a monotone chain
- Succinct rank/select structure
- 0 $\rightarrow$, 1 $\uparrow$

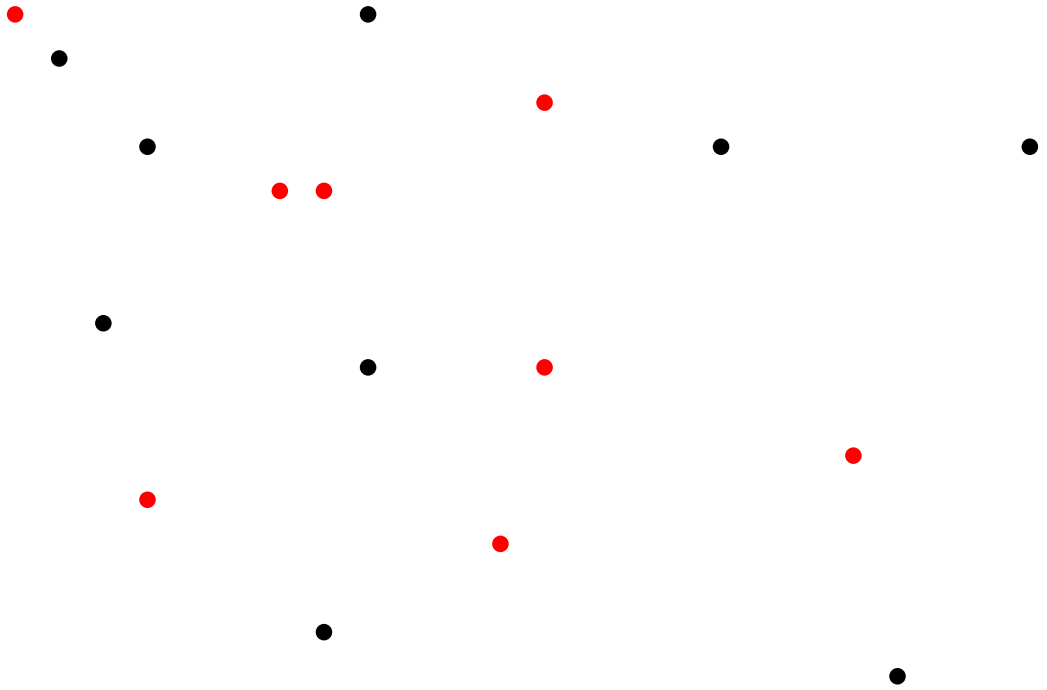
Credit: *Munro 1996*

Problem



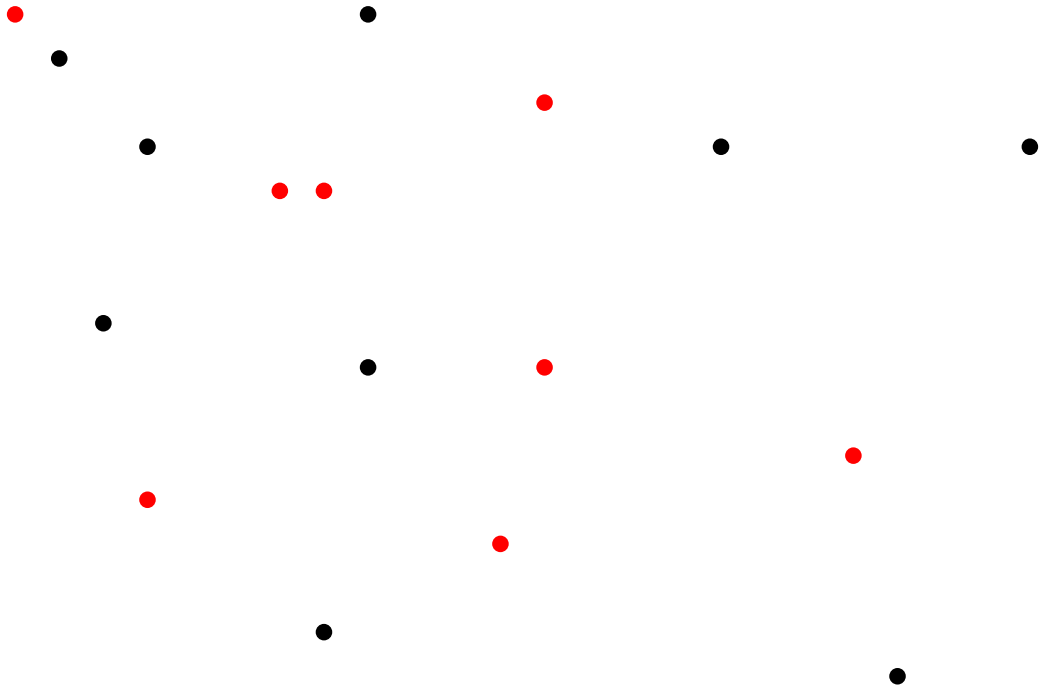
- Point set P
- Each point $p \in P$ associated with a *distinct* time t_p
- Given a query time window t_1, t_2 find the closest pair of points whose associated times are within that window

Computing Candidate Pairs



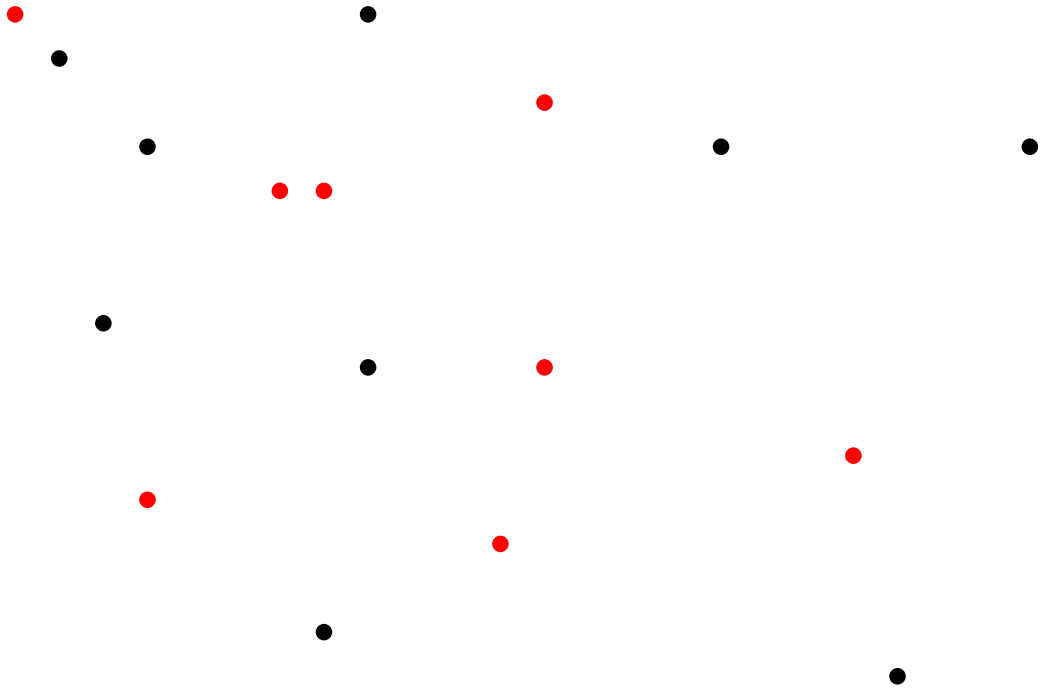
- If we knew the distance between the closest pair, we could use a grid as in the decision problem

Computing Candidate Pairs



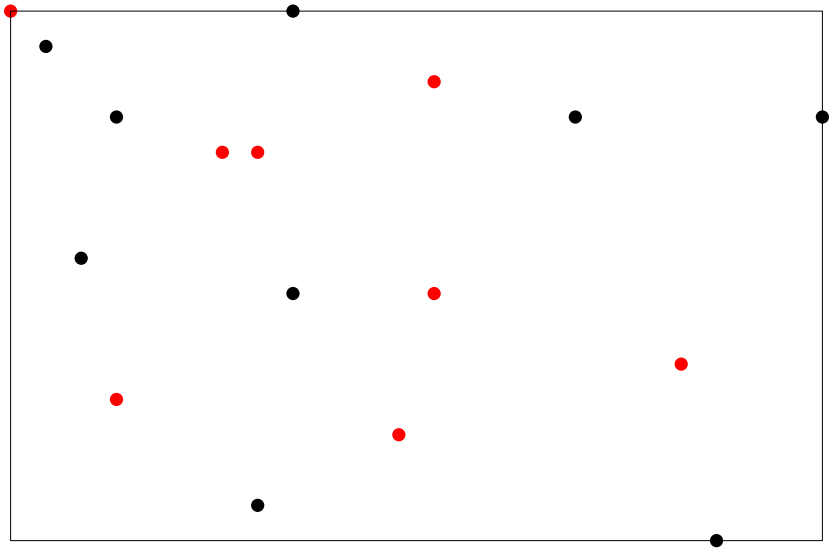
- If we knew the distance between the closest pair, we could use a grid as in the decision problem
- Instead, we need a structure that works for any distance

Computing Candidate Pairs



- If we knew the distance between the closest pair, we could use a grid as in the decision problem
- Instead, we need a structure that works for any distance
- Solution: the quad tree

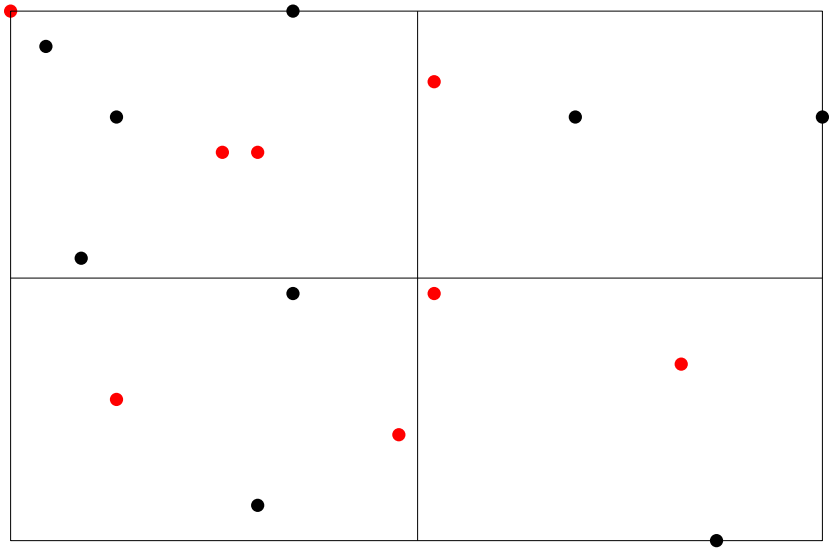
The Quad Tree



- Divides the bounding box of a point set into 4 quadrants we call *cells*
- Recurse on each cell containing more than a single point

Credit: *Finkel & Bentley 1987*

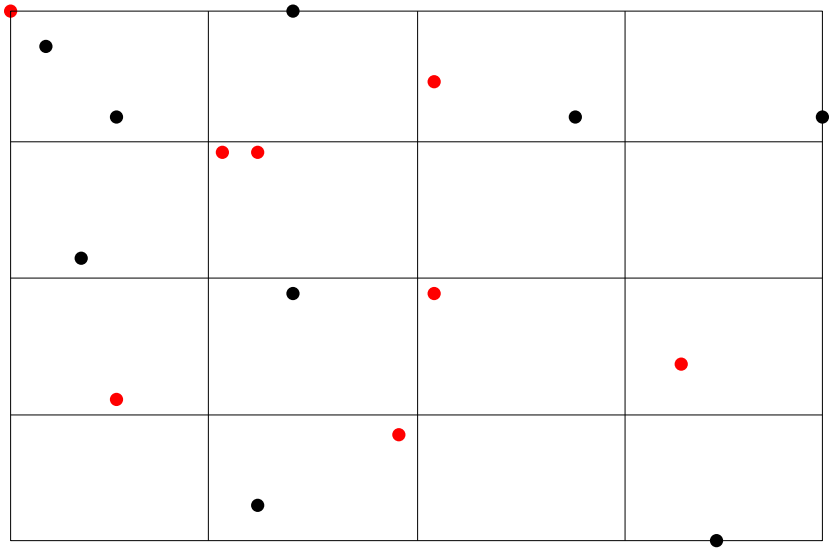
The Quad Tree



- Divides the bounding box of a point set into 4 quadrants we call *cells*
- Recurse on each cell containing more than a single point

Credit: *Finkel & Bentley 1987*

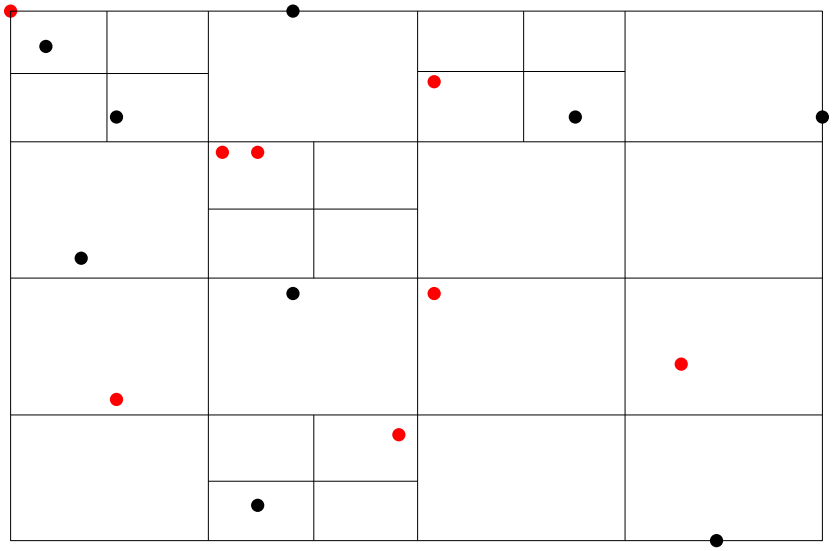
The Quad Tree



- Divides the bounding box of a point set into 4 quadrants we call *cells*
- Recurse on each cell containing more than a single point

Credit: *Finkel & Bentley 1987*

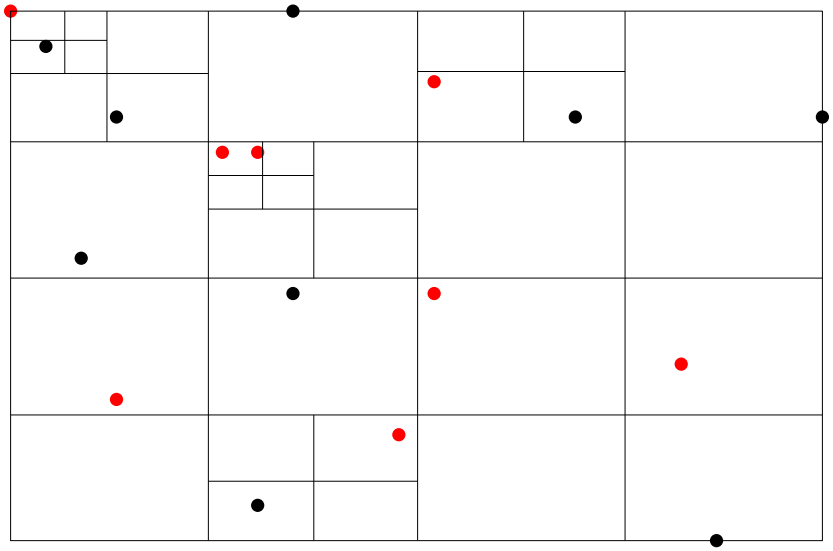
The Quad Tree



- Divides the bounding box of a point set into 4 quadrants we call *cells*
- Recurse on each cell containing more than a single point

Credit: *Finkel & Bentley 1987*

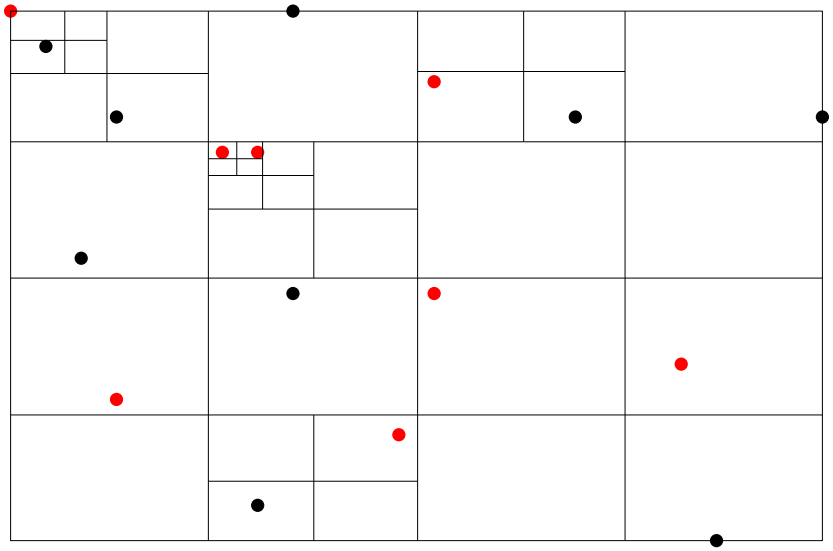
The Quad Tree



- Divides the bounding box of a point set into 4 quadrants we call *cells*
- Recurse on each cell containing more than a single point

Credit: *Finkel & Bentley 1987*

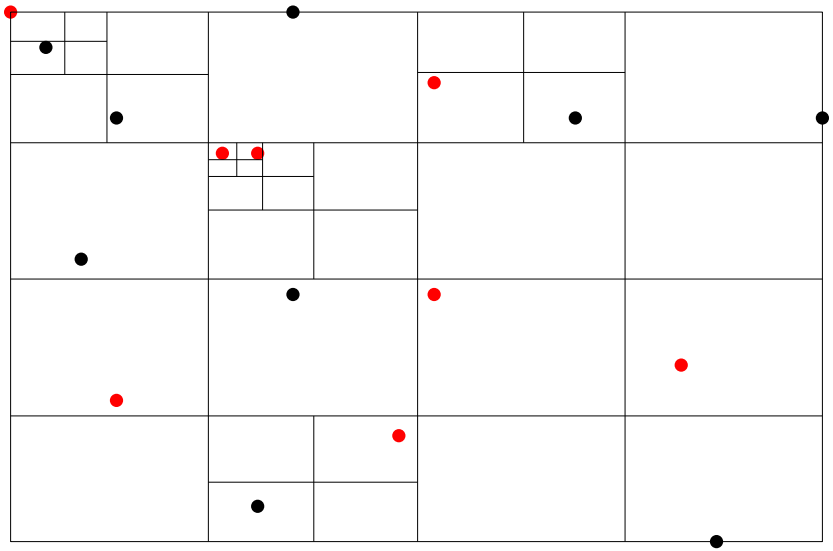
The Quad Tree



- Divides the bounding box of a point set into 4 quadrants we call *cells*
- Recurse on each cell containing more than a single point

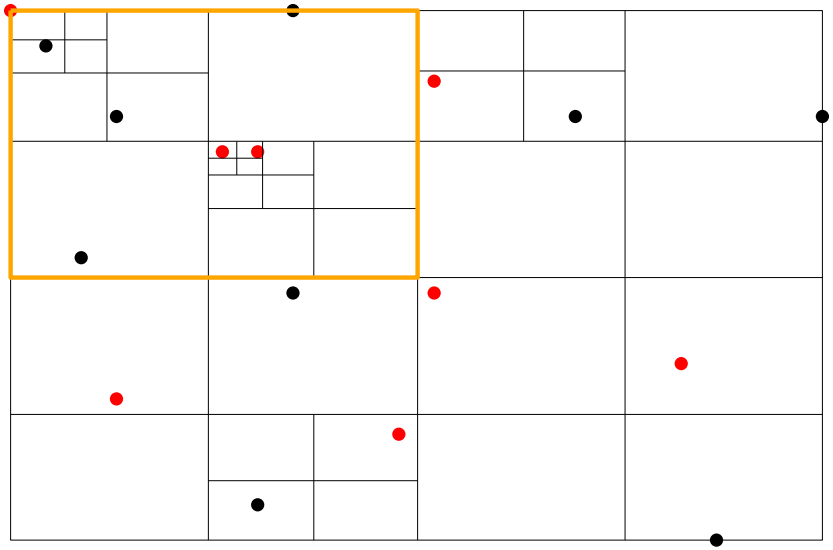
Credit: *Finkel & Bentley 1987*

Quad Tree Issue: Height



Height of quadtree unbounded w.r.t. n

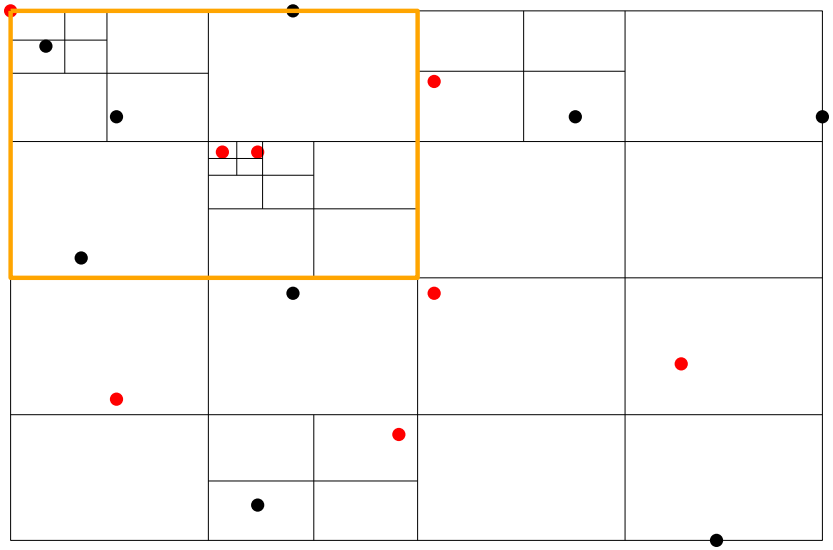
Solution: The Centroid



- There exists a quadtree cell B such that:
  - $|P \cap B| \leq \alpha n$
  - $|P \setminus B| \leq \alpha n$
- We call this a centroid cell

Credit: *Arya et al. 1998*

Computing Candidate Pairs



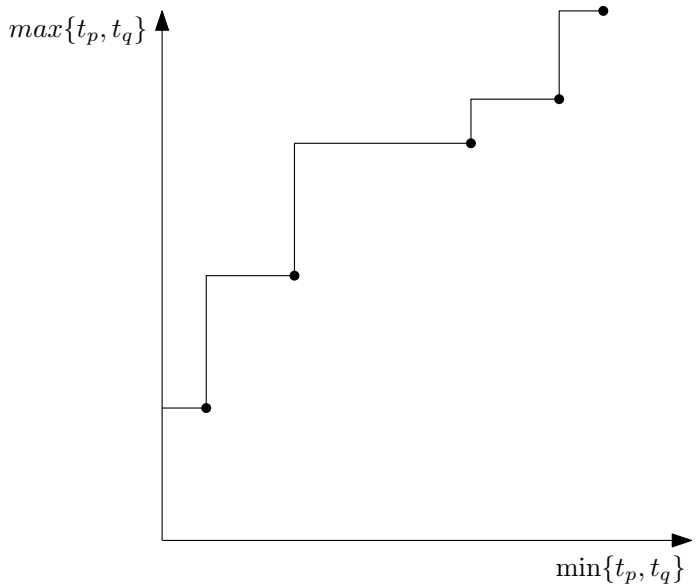
1. Compute the quadtree on P, and the centroid cell B
2. For each point p outside of the centroid ($P \setminus B$):
   1. Consider the pair formed by a constant number of p's time-order neighbors inside the centroid ($P \cap B$)
3. Recurse on $P \cap B$ and $P \setminus B$

Credit: *Chan 1998*

Now What?

- $O(n \log n)$ pairs
- How do we find the closest pair for a query?

Reduction to 2D Dominance Range Minimum

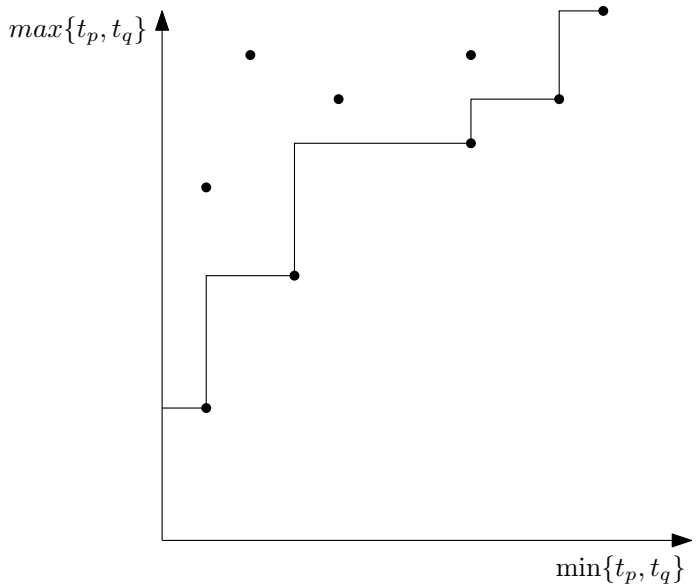


Given a query time window $[t_1, t_2]$, determine if there exists a pair pq such that:

- $t_1 \leq \min\{t_p, t_q\}$
- $t_2 \geq \max\{t_p, t_q\}$
- The distance $d(p, q)$ is minimized

Credit: *Chan 2013*

Reduction to 2D Dominance Range Minimum

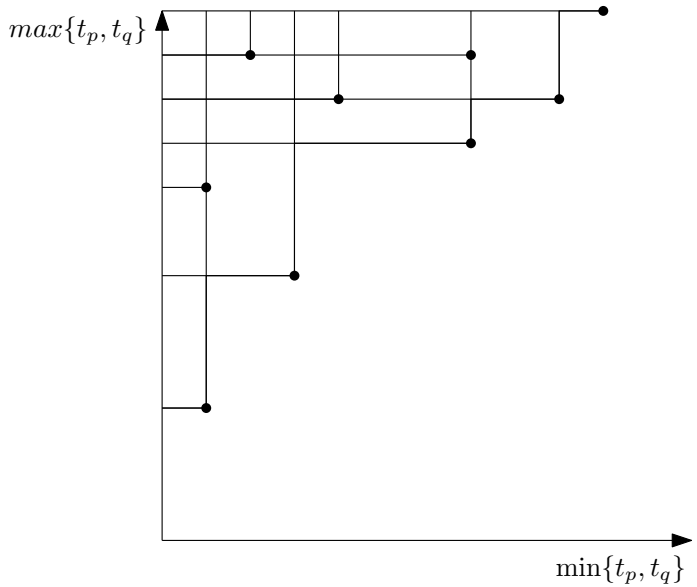


Given a query time window $[t_1, t_2]$, determine if there exists a pair pq such that:

- $t_1 \leq \min\{t_p, t_q\}$
- $t_2 \geq \max\{t_p, t_q\}$
- The distance $d(p, q)$ is minimized

Credit: *Chan 2013*

Reduction to 2D Dominance Range Minimum

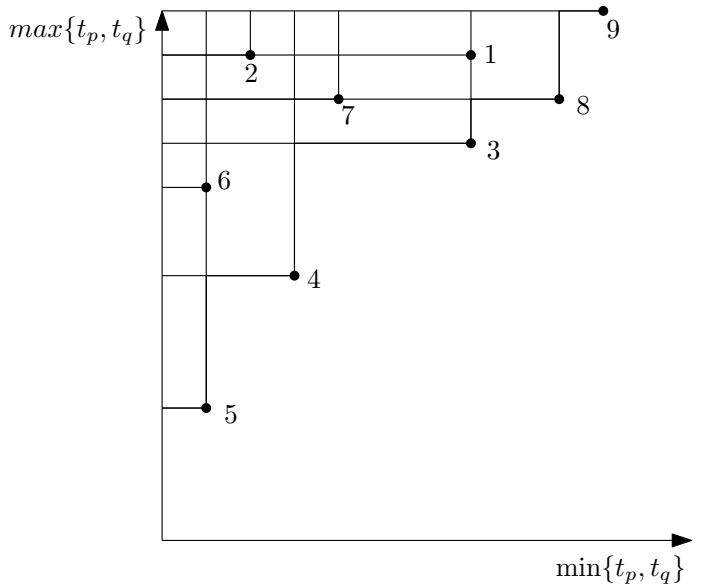


Given a query time window $[t_1, t_2]$, determine if there exists a pair pq such that:

- $t_1 \leq \min\{t_p, t_q\}$
- $t_2 \geq \max\{t_p, t_q\}$
- The distance $d(p, q)$ is minimized

Credit: *Chan 2013*

Reduction to 2D Dominance Range Minimum

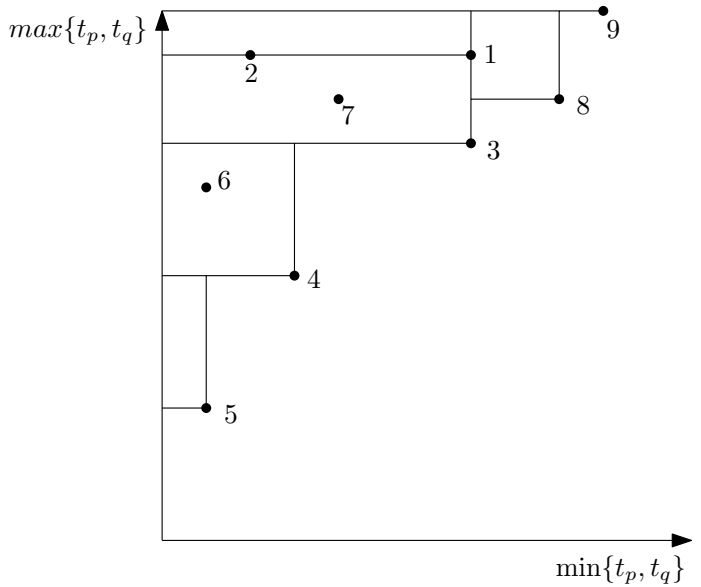


Given a query time window $[t_1, t_2]$, determine if there exists a pair pq such that:

- $t_1 \leq \min\{t_p, t_q\}$
- $t_2 \geq \max\{t_p, t_q\}$
- The distance $d(p, q)$ is minimized

Credit: *Chan 2013*

Reduction to 2D Dominance Range Minimum

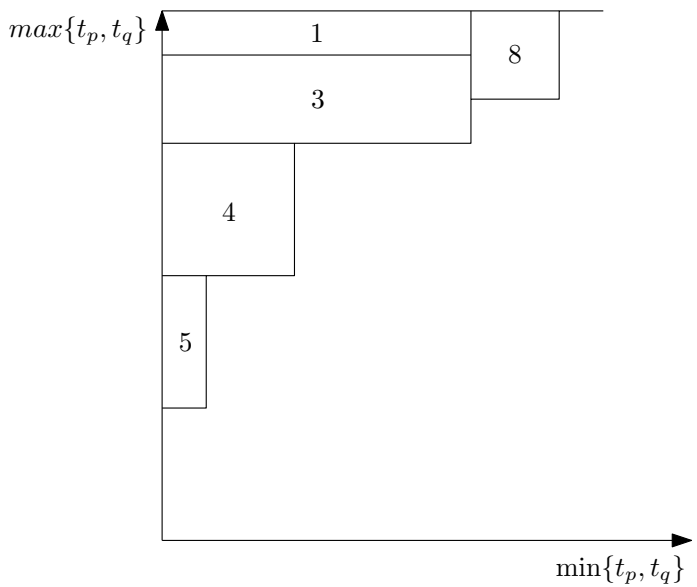


Given a query time window $[t_1, t_2]$, determine if there exists a pair pq such that:

- $t_1 \leq \min\{t_p, t_q\}$
- $t_2 \geq \max\{t_p, t_q\}$
- The distance $d(p, q)$ is minimized

Credit: *Chan 2013*

Reduction to 2D Dominance Range Minimum

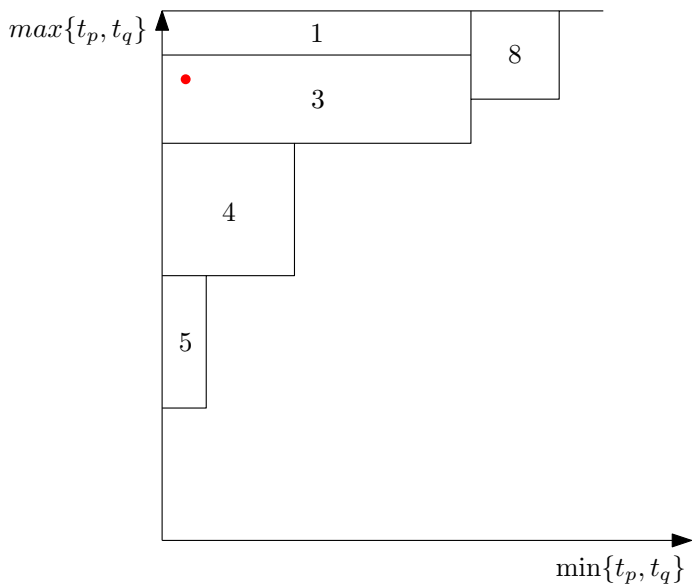


Given a query time window $[t_1, t_2]$, determine if there exists a pair pq such that:

- $t_1 \leq \min\{t_p, t_q\}$
- $t_2 \geq \max\{t_p, t_q\}$
- The distance $d(p, q)$ is minimized

Credit: *Chan 2013*

Reduction to 2D Dominance Range Minimum



Given a query time window $[t_1, t_2]$, determine if there exists a pair pq such that:

- $t_1 \leq \min\{t_p, t_q\}$
- $t_2 \geq \max\{t_p, t_q\}$
- The distance $d(p, q)$ is minimized

Credit: *Chan 2013*

Summary and Questions

Problem	Pre-Processing	Space	Query
Decision Closest Pair	$O(n)$ (expected)	$O(n)$ bits	$O(1)$
	Grid	Succinct Rank/Select	
Closest Pair	$O(n \log n \log \log n)$	$O(n \log n)$ words	$O(\log \log n)$
	Quadtree	Orthogonal Point Location	