Three Approaches to Building Time-Windowed Geometric Data Structures

Simon Pratt

MMath Thesis

Problem Statement

Preprocess a set of n geometric objects each associated with a time value, s.t.:

given a query time window $[t_1, t_2]$, solve a geometric problem on the subset of objects whose associated time values are within that window.

Problem Statement

e.g., points or line segments

Preprocess a set of n geometric objects each associated with a time value, s.t.:

given a query time window $[t_1, t_2]$, solve a geometric problem on the subset of objects whose associated time values are within that window.

Problem Statement

e.g., points or line segments

Preprocess a set of n geometric objects each associated with a time value, s.t.: 1 to n

given a query time window $[t_1, t_2]$, solve a geometric problem on the subset of objects whose associated time values are within that window.

Problem Statement

e.g., points or line segments

Preprocess a set of n geometric objects each associated with a time value, s.t.:

1 to n

given a query time window $[t_1, t_2]$,
solve a geometric problem on the subset of objects whose associated time values are within that window.

e.g. closest pair

Motivation

- *Geographic Information Systems* (GIS) data
  (e.g., latitude, longitude, altitude, time)
- Timestamped data
  (e.g., social network data)

Example: Time-Windowed Closest Pair

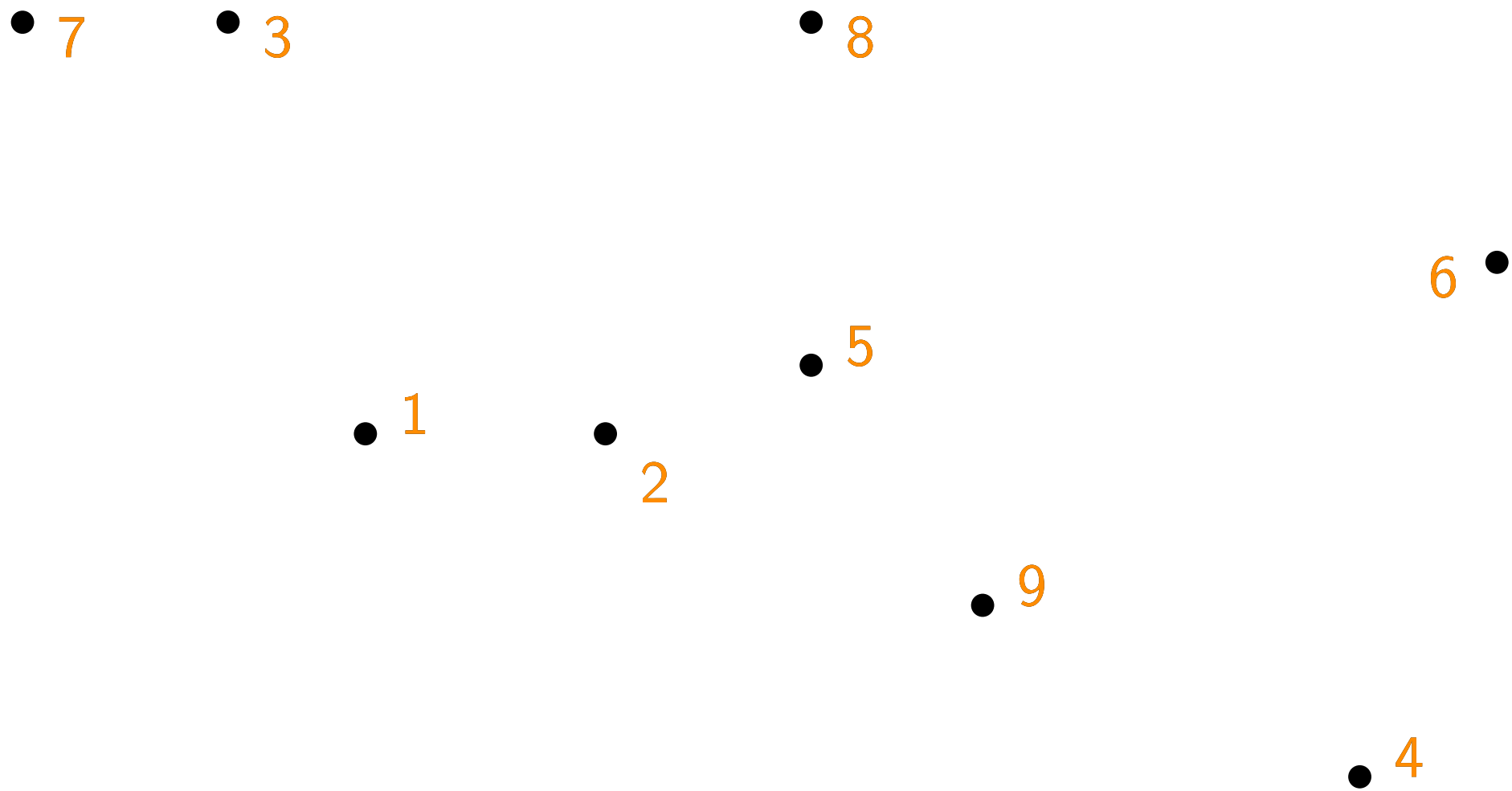
Preprocess a set of n time-labeled points, to efficiently determine closest pair within a time window.

Example: Time-Windowed Closest Pair

Preprocess a set of n time-labeled points, to efficiently determine closest pair within a time window.

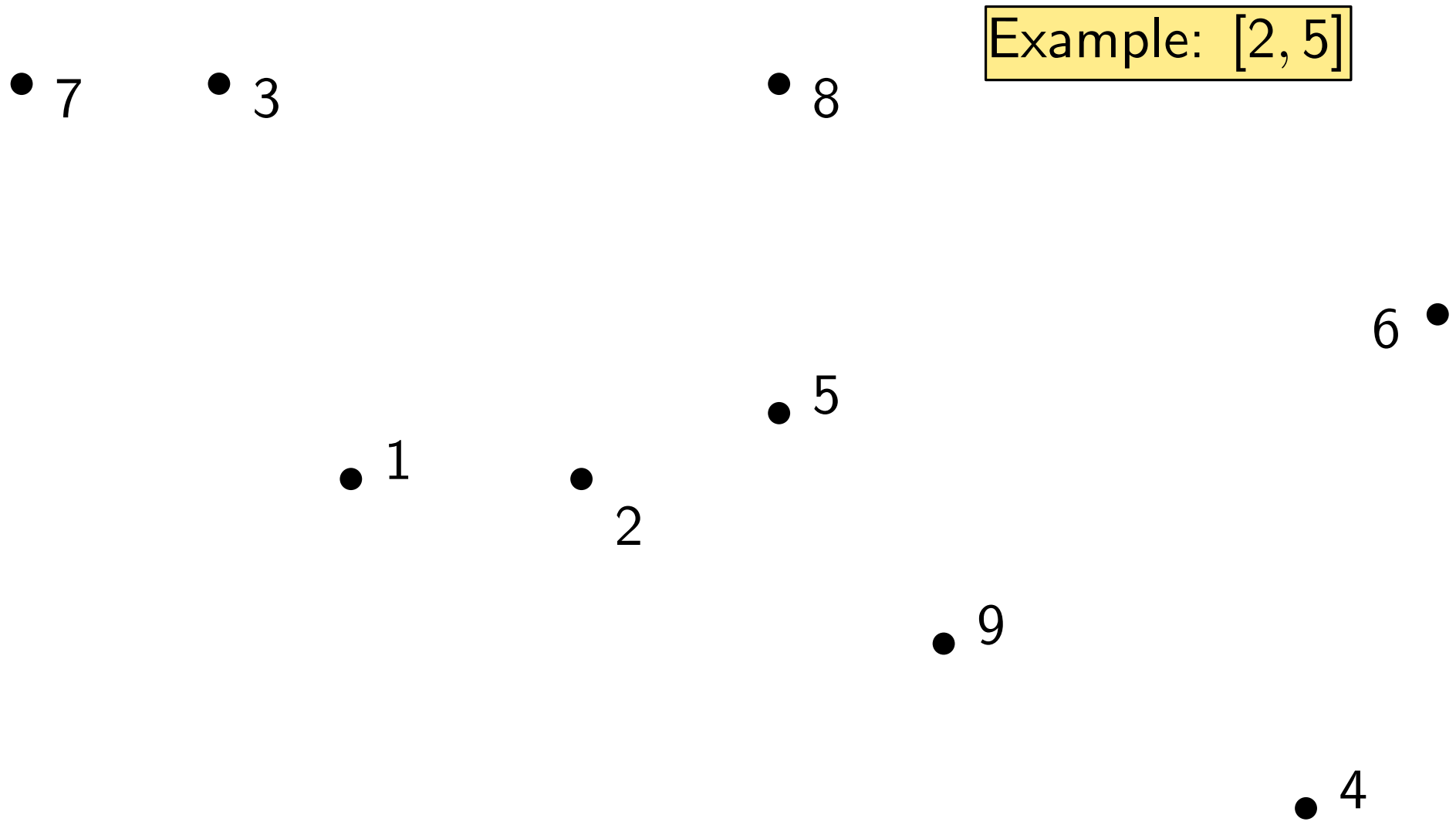
Example: Time-Windowed Closest Pair

Preprocess a set of n time-labeled points, to efficiently determine closest pair within a time window.



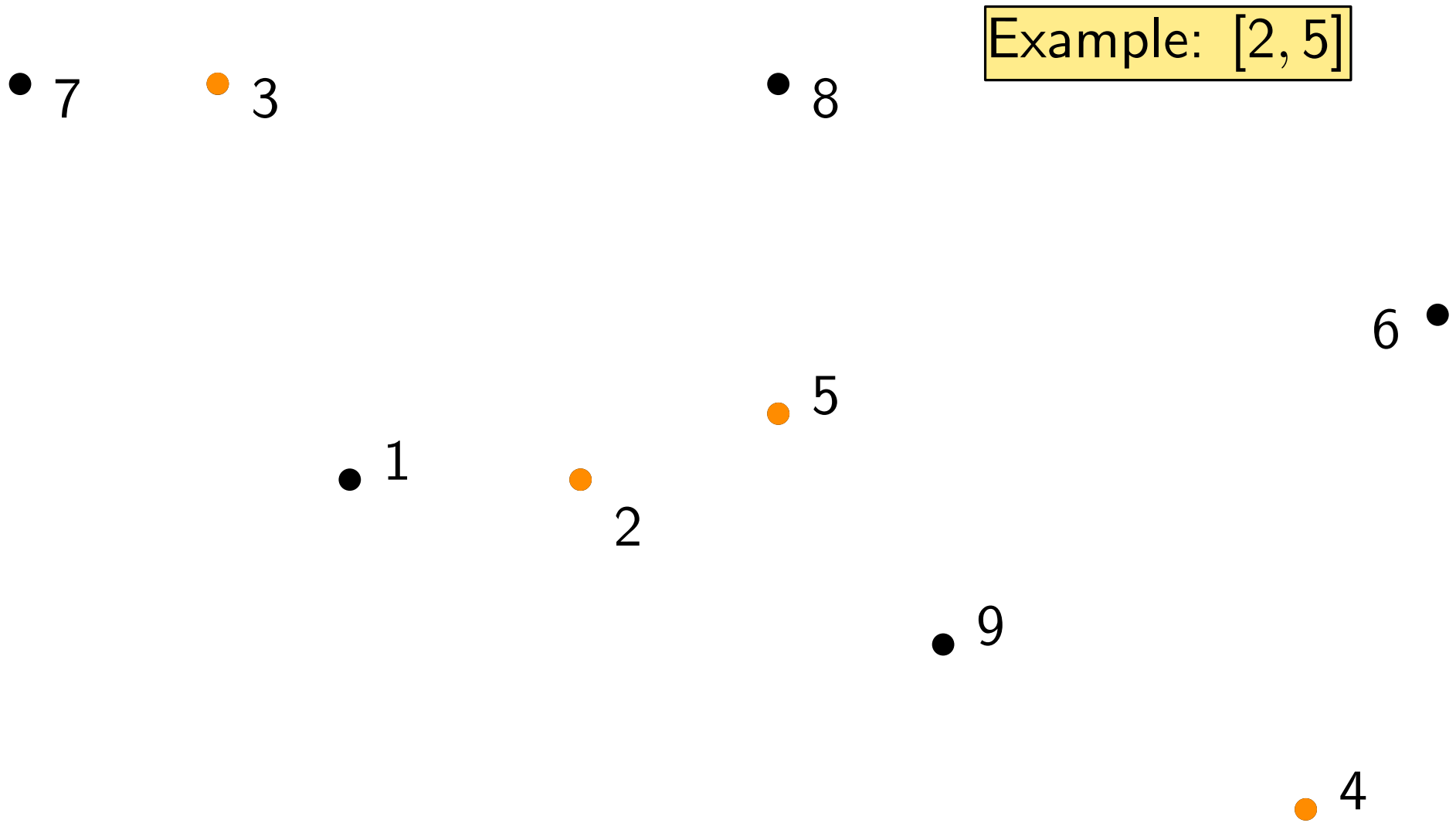
Example: Time-Windowed Closest Pair

Preprocess a set of n time-labeled points, to efficiently determine closest pair within a time window.



Example: Time-Windowed Closest Pair

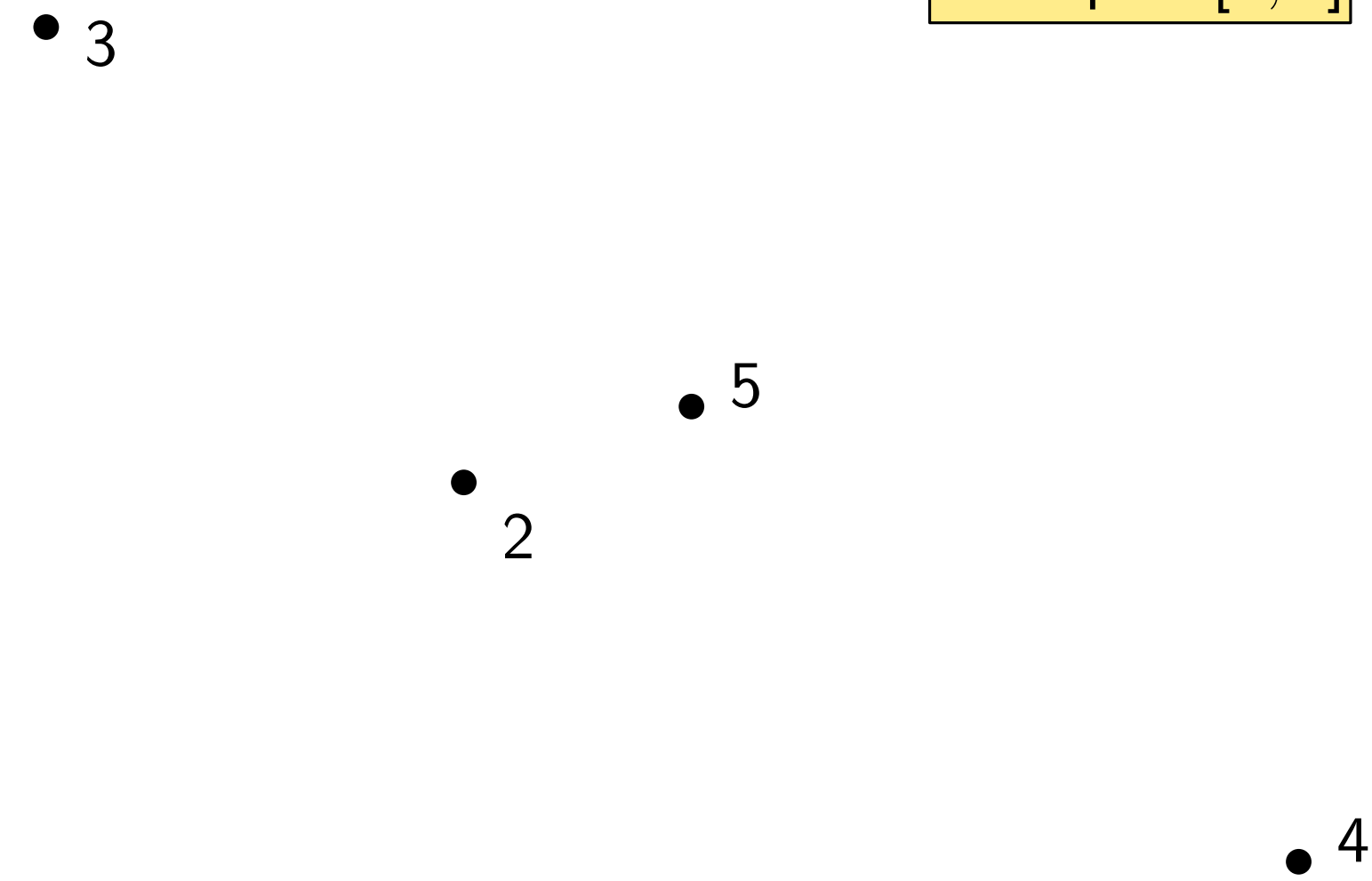
Preprocess a set of n time-labeled points, to efficiently determine closest pair within a time window.



Example: Time-Windowed Closest Pair

Preprocess a set of n time-labeled points, to efficiently determine closest pair within a time window.

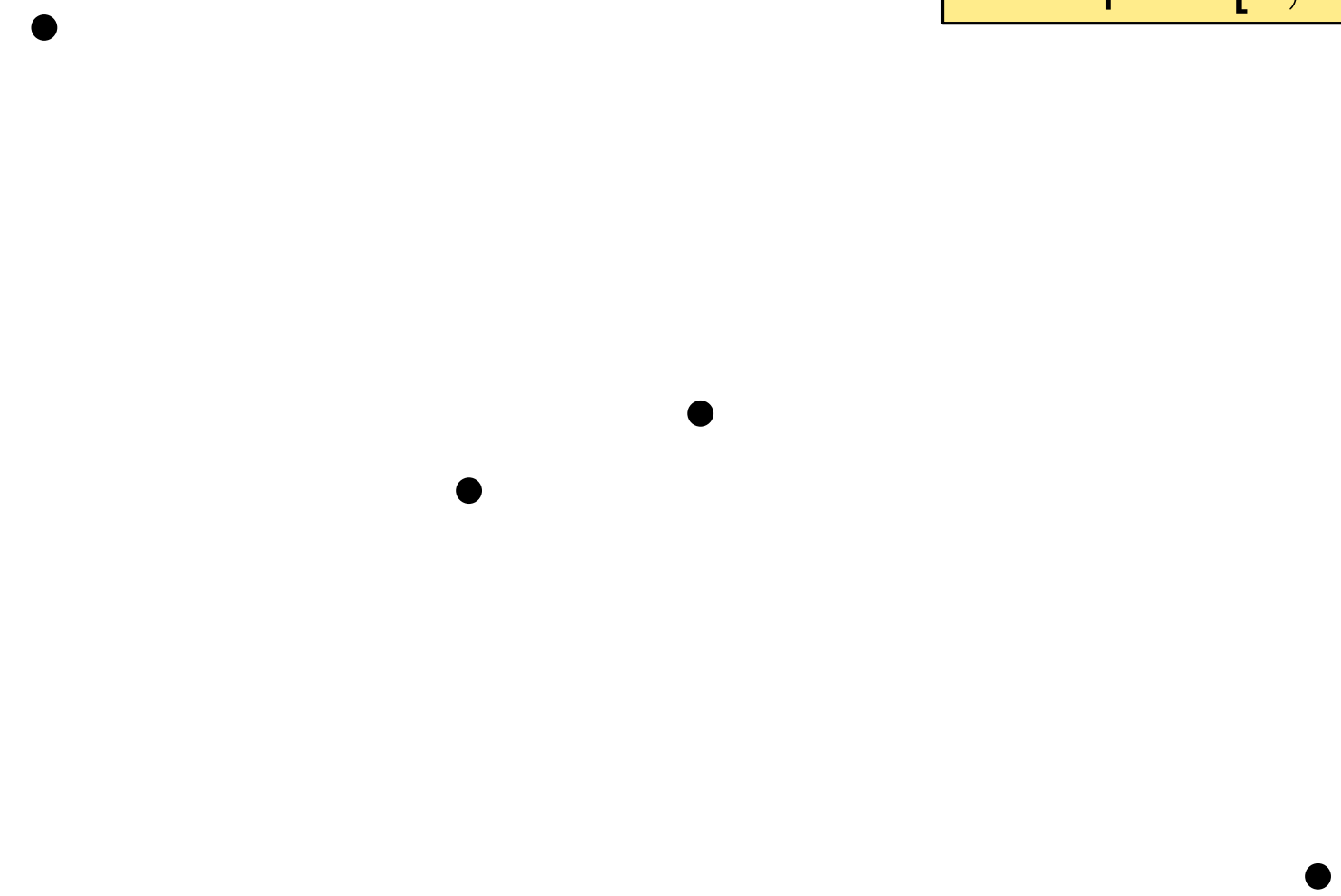
Example: $[2, 5]$



Example: Time-Windowed Closest Pair

Preprocess a set of n time-labeled points, to efficiently determine closest pair within a time window.

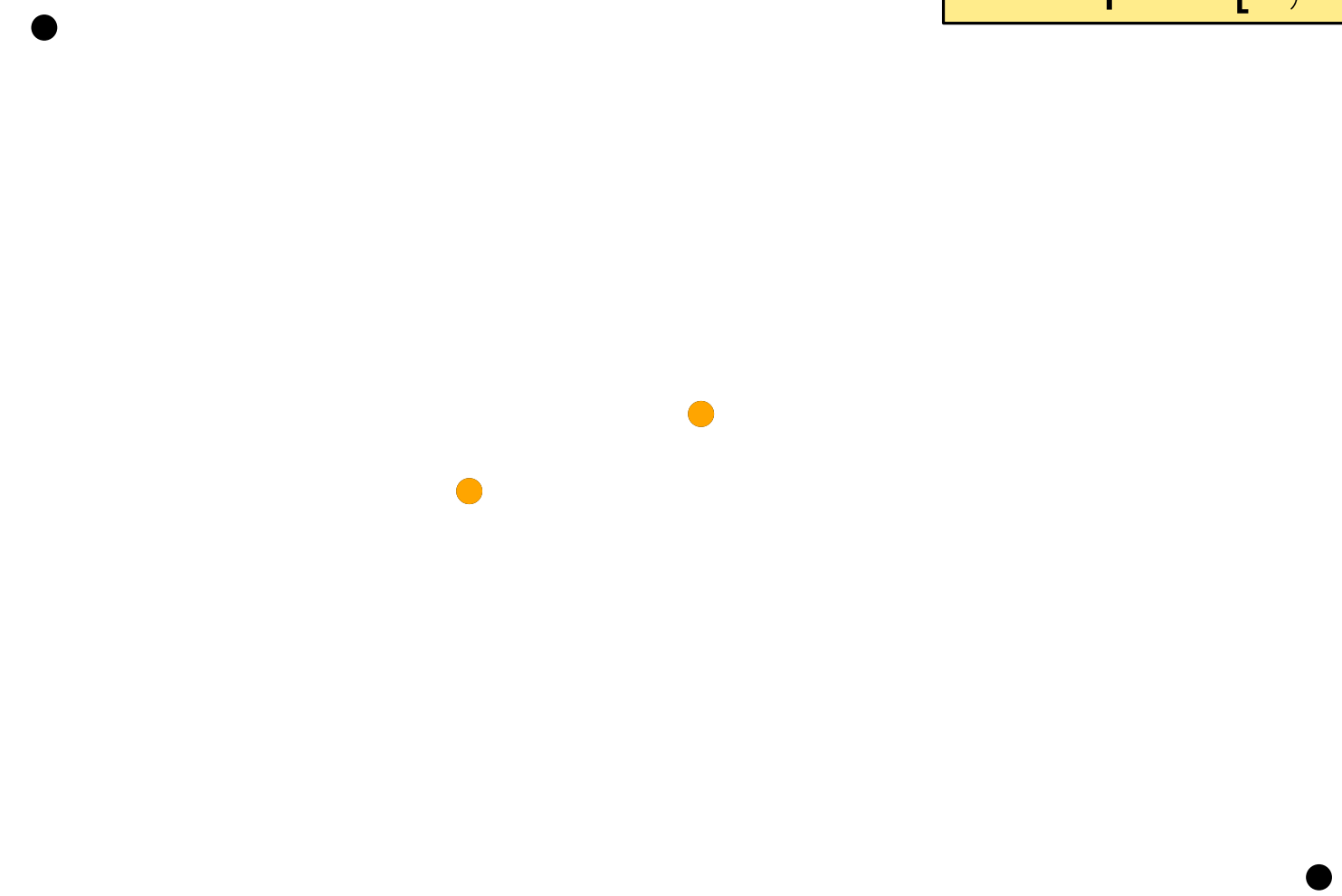
Example: $[2, 5]$



Example: Time-Windowed Closest Pair

Preprocess a set of n time-labeled points, to efficiently determine closest pair within a time window.

Example: $[2, 5]$



Previous Results

Bannister, Devanny, Goodrich, Simons, Trott; *CCCG 2014*:

- Time-windowed convex hull
  - ◇ Gift wrapping & tangent $O(\log^2 \omega)$
  - ◇ Linear programming $O(\log \omega)$
  - ◇ Line decision $O(\log \omega)$
  - ◇ Line stabbing $O(\log^2 \omega)$
- Time-windowed approximate spherical range searching
- Time-windowed approximate nearest neighbor

Previous Results

Bannister, Devanny, Goodrich, Simons, Trott; *CCCG 2014*:

- Time-windowed convex hull
  - ◇ Gift wrapping & tangent $O(\log^2 \boxed{\omega})$
  - ◇ Linear programming $O(\log \boxed{\omega})$
  - ◇ Line decision $O(\log \boxed{\omega})$
  - ◇ Line stabbing $O(\log^2 \boxed{\omega})$
- Time-windowed approximate spherical range searching
- Time-windowed approximate nearest neighbor

$\omega = t_2 - t_1$, the size of the query time window

New Results

New Results

Approach #1: Grids & Quadtrees

Problem	Preprocessing	Space	Query
CP Decision	$O(n)$ exp.	$O(n)$ bits	$O(1)$
Closest Pair	$O(n \lg n \lg \lg n)$	$O(n \lg n)$	$O(\lg \lg n)$

New Results

Approach #1: Grids & Quadtrees

Problem	Preprocessing	Space	Query
CP Decision	$O(n)$ exp.	$O(n)$ bits	$O(1)$
Closest Pair	$O(n \lg n \lg \lg n)$	$O(n \lg n)$	$O(\lg \lg n)$

Approach #2: Generalized Range Successor

Problem	Preprocessing Time
2D diameter decision	$O(n \lg n)$
3D diameter decision	$O(n \lg^2 n)$
2D orth. seg. intersection detection	$O(n \lg n \lg \lg n)$

New Results

Approach #1: Grids & Quadtrees

Problem	Preprocessing	Space	Query
CP Decision	$O(n)$ exp.	$O(n)$ bits	$O(1)$
Closest Pair	$O(n \lg n \lg \lg n)$	$O(n \lg n)$	$O(\lg \lg n)$

Approach #2: Generalized Range Successor

Problem	Preprocessing Time
2D diameter decision	$O(n \lg n)$
3D diameter decision	$O(n \lg^2 n)$
2D orth. seg. intersection detection	$O(n \lg n \lg \lg n)$

Approach #3: FIFO Update Sequence

Problem	Preprocessing Time
2D convex hull area decision	$O(n\alpha(n) \lg n)$
2D width decision	$O(n \lg^7 n)$

New Results

Approach #1: Grids & Quadtrees

Problem	Preprocessing	Space	Query
CP Decision	$O(n)$ exp.	$O(n)$ bits	$O(1)$
Closest Pair	$O(n \lg n \lg \lg n)$	$O(n \lg n)$	$O(\lg \lg n)$

Approach #2: Generalized Range Successor

Problem	Preprocessing Time
2D diameter decision	$O(n \lg n)$
3D diameter decision	$O(n \lg^2 n)$
2D orth. seg. intersection detection	$O(n \lg n \lg \lg n)$

Approach #3: FIFO Update Sequence

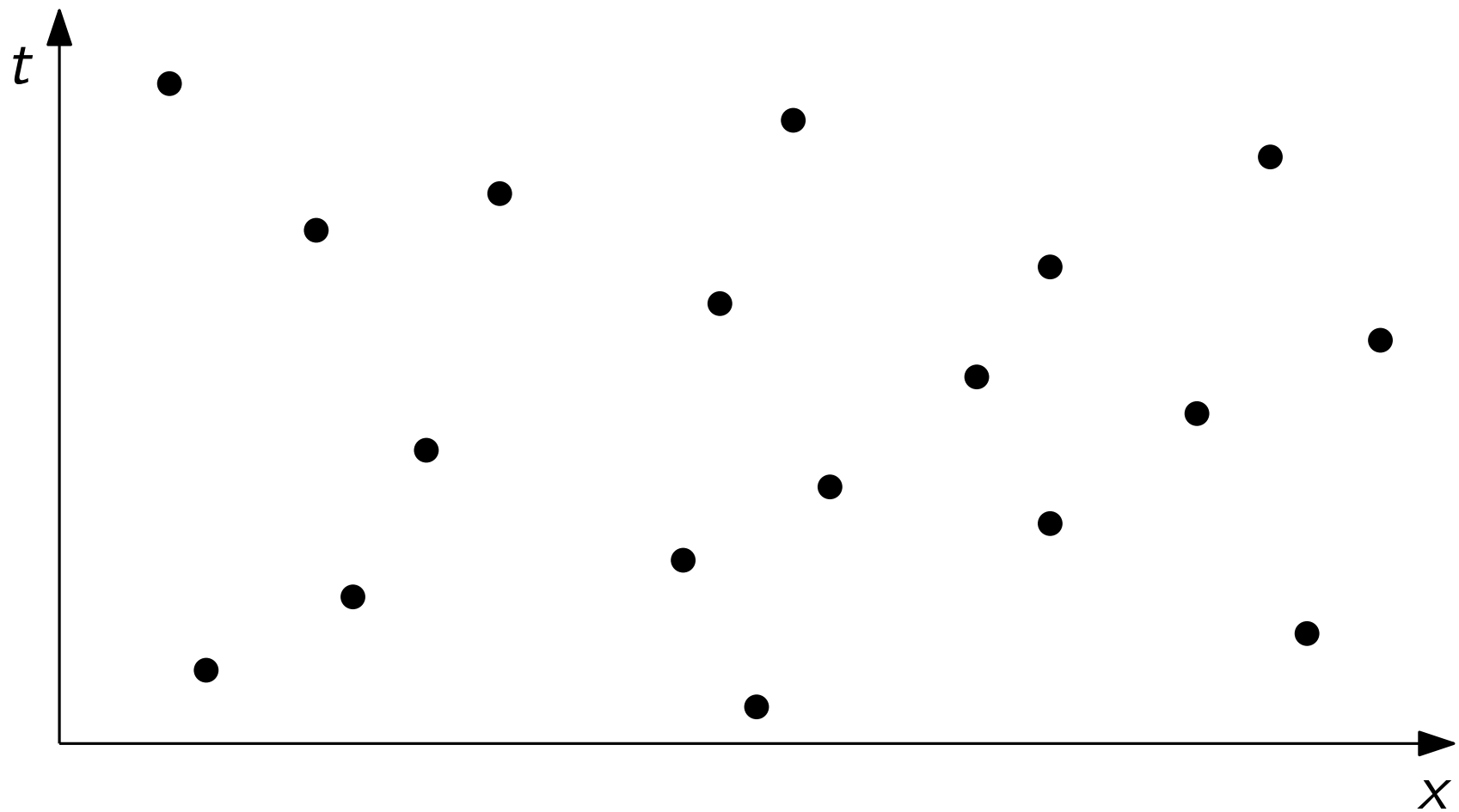
Problem	Preprocessing Time
2D convex hull area decision	$O(n\alpha(n) \lg n)$
2D width decision	$O(n \lg^7 n)$

Time-Windowed Closest Pair Decision Problem

Preprocess n time-labeled points in $\mathbb{R}^d$
s.t. we can determine if $\exists p, q : d(p, q) < 1$.

Time-Windowed Closest Pair Decision Problem

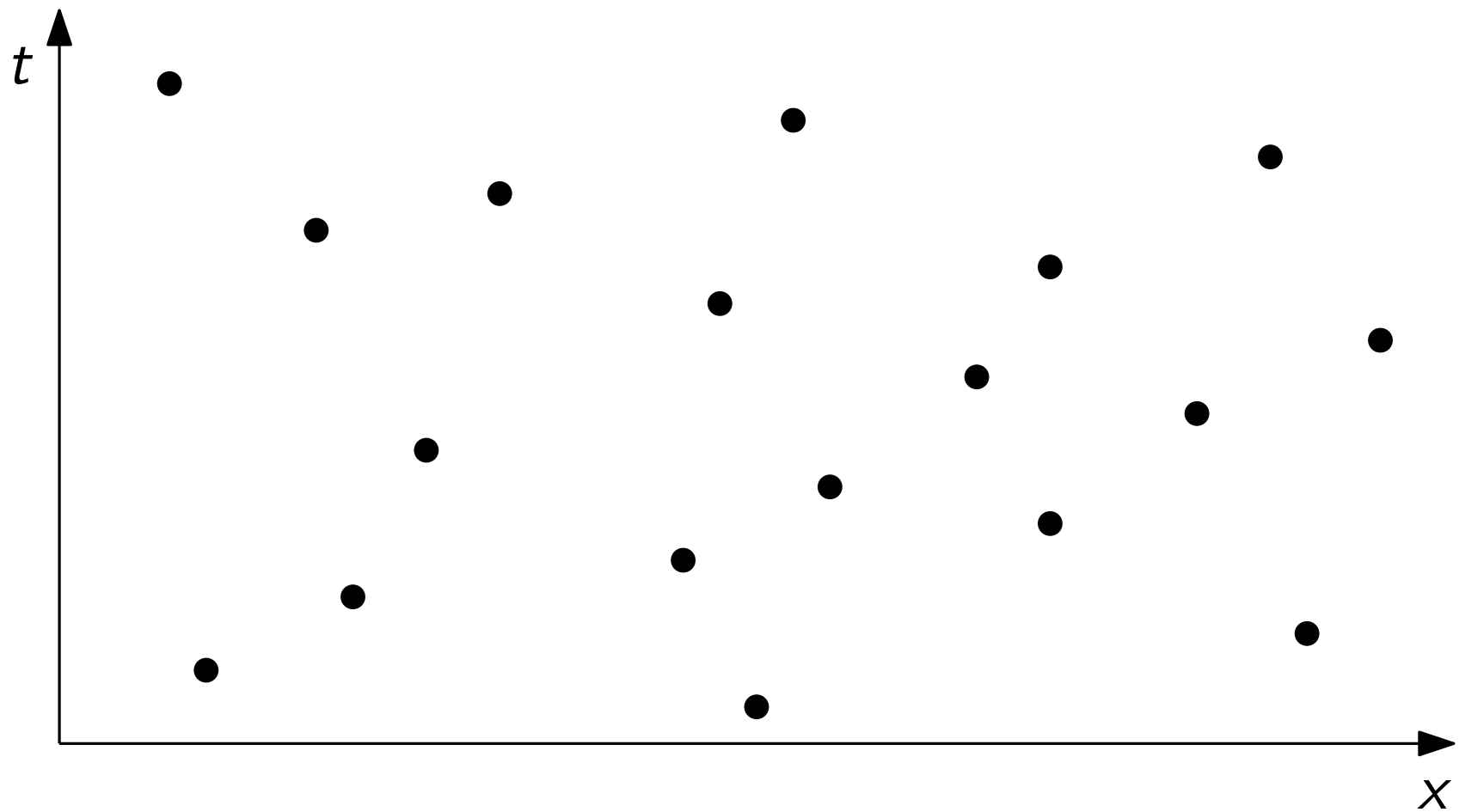
Preprocess n time-labeled points in $\mathbb{R}^d$
s.t. we can determine if $\exists p, q : d(p, q) < 1$.



Time-Windowed Closest Pair Decision Problem

Preprocess n time-labeled points in $\mathbb{R}^d$
s.t. we can determine if $\exists p, q : d(p, q) < 1$.

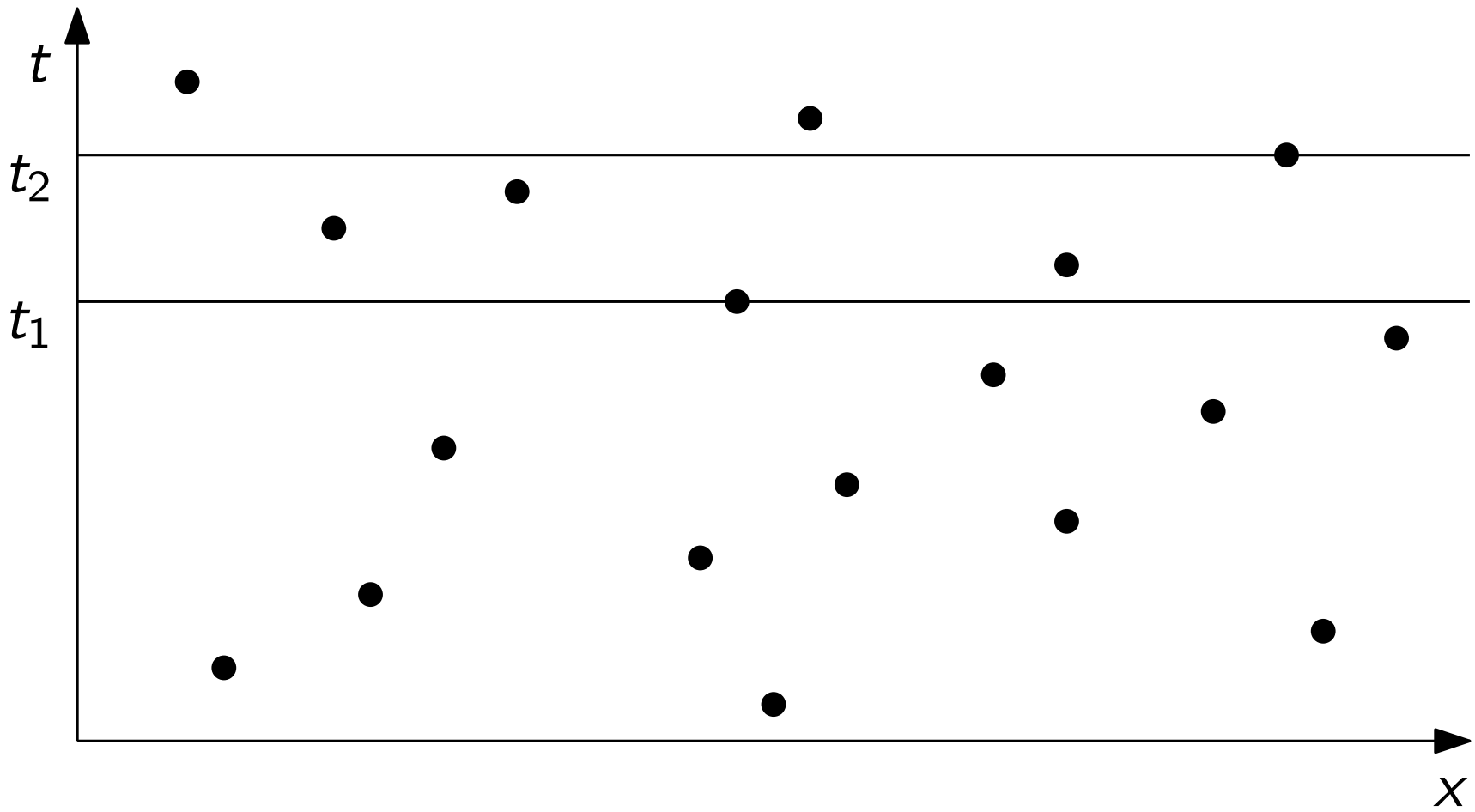
$\overline{\qquad 1 \qquad}$



Time-Windowed Closest Pair Decision Problem

Preprocess n time-labeled points in $\mathbb{R}^d$
s.t. we can determine if $\exists p, q : d(p, q) < 1$.

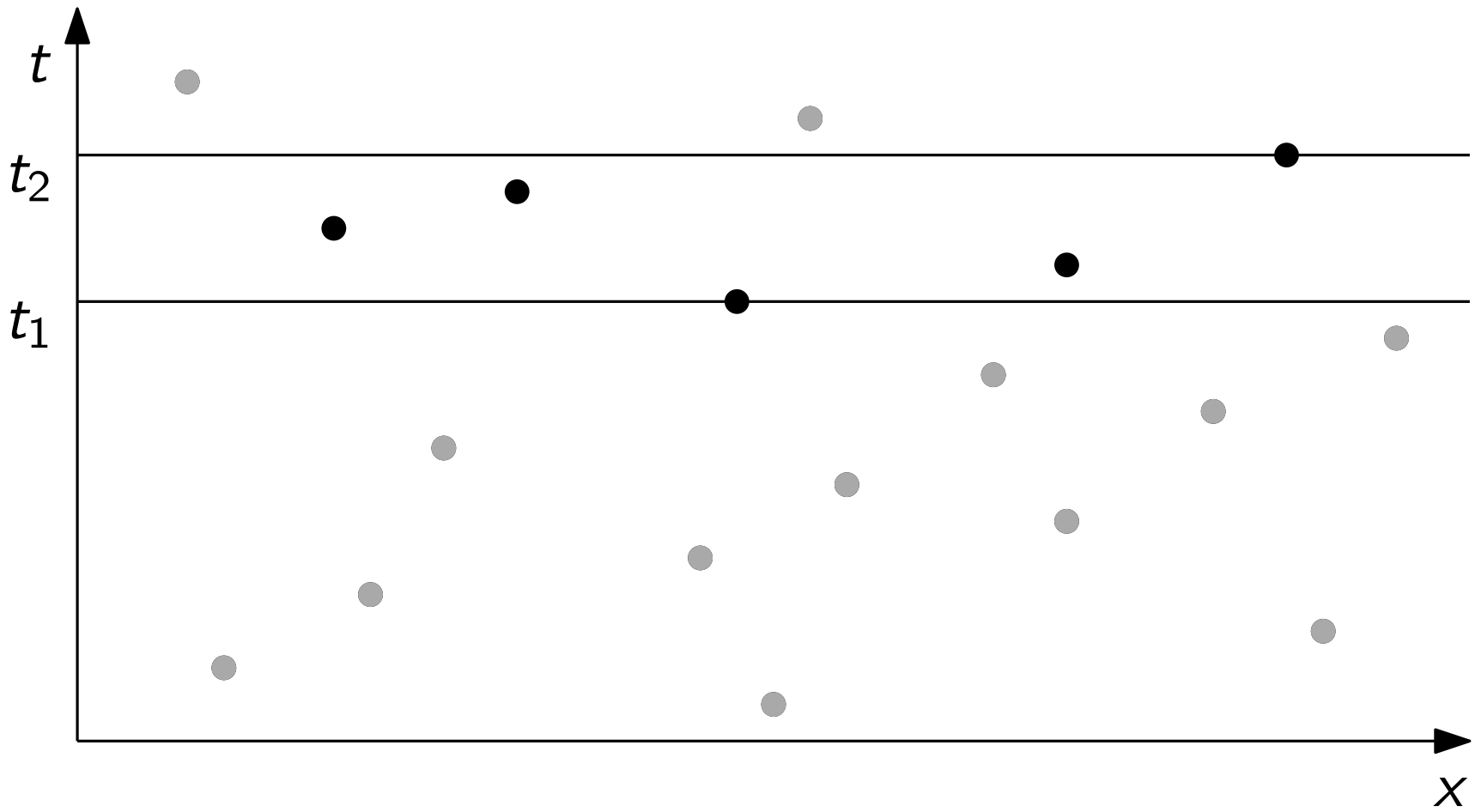
$$\overline{\qquad 1 \qquad}$$



Time-Windowed Closest Pair Decision Problem

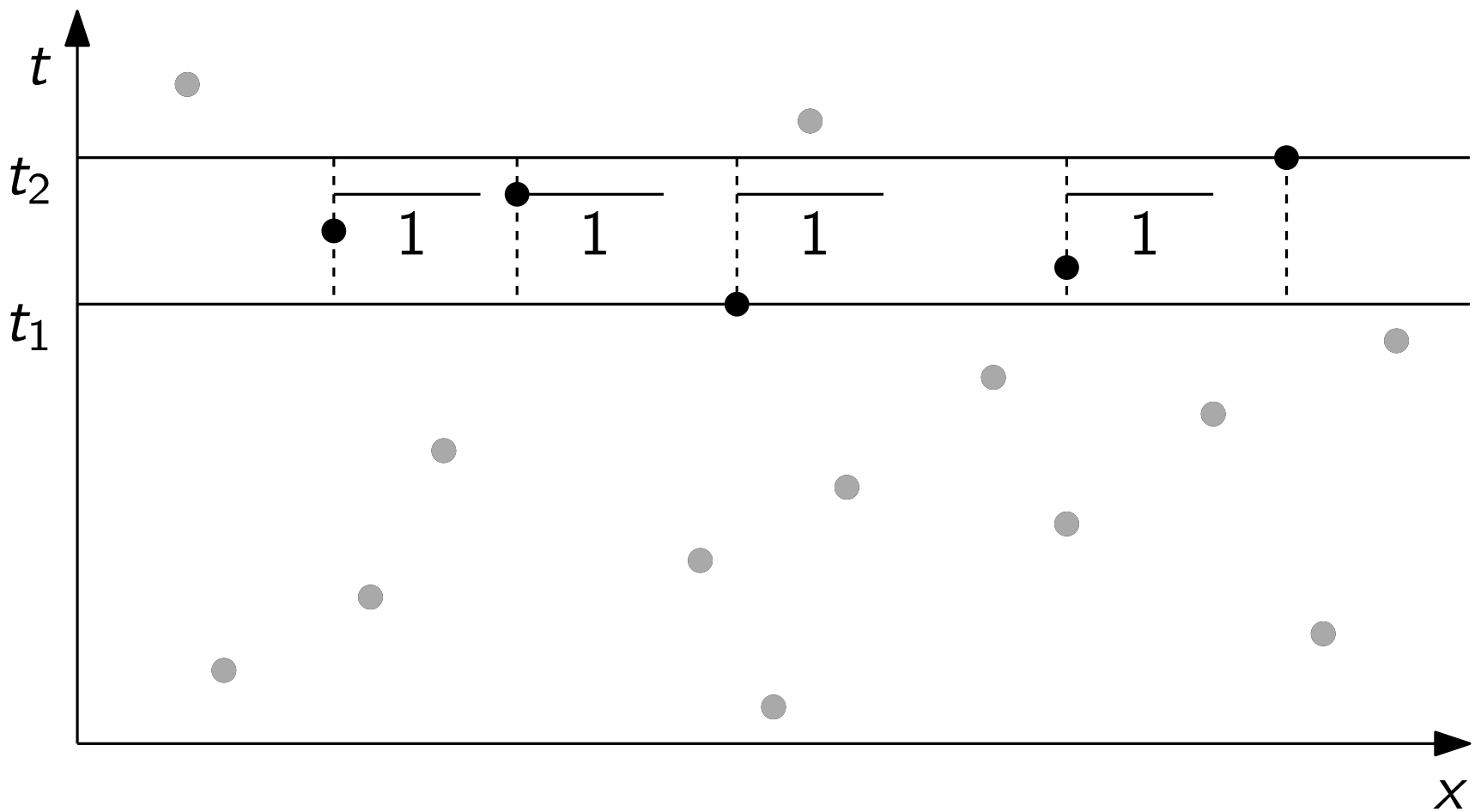
Preprocess n time-labeled points in $\mathbb{R}^d$
s.t. we can determine if $\exists p, q : d(p, q) < 1$.

$\overline{\qquad 1 \qquad}$



Time-Windowed Closest Pair Decision Problem

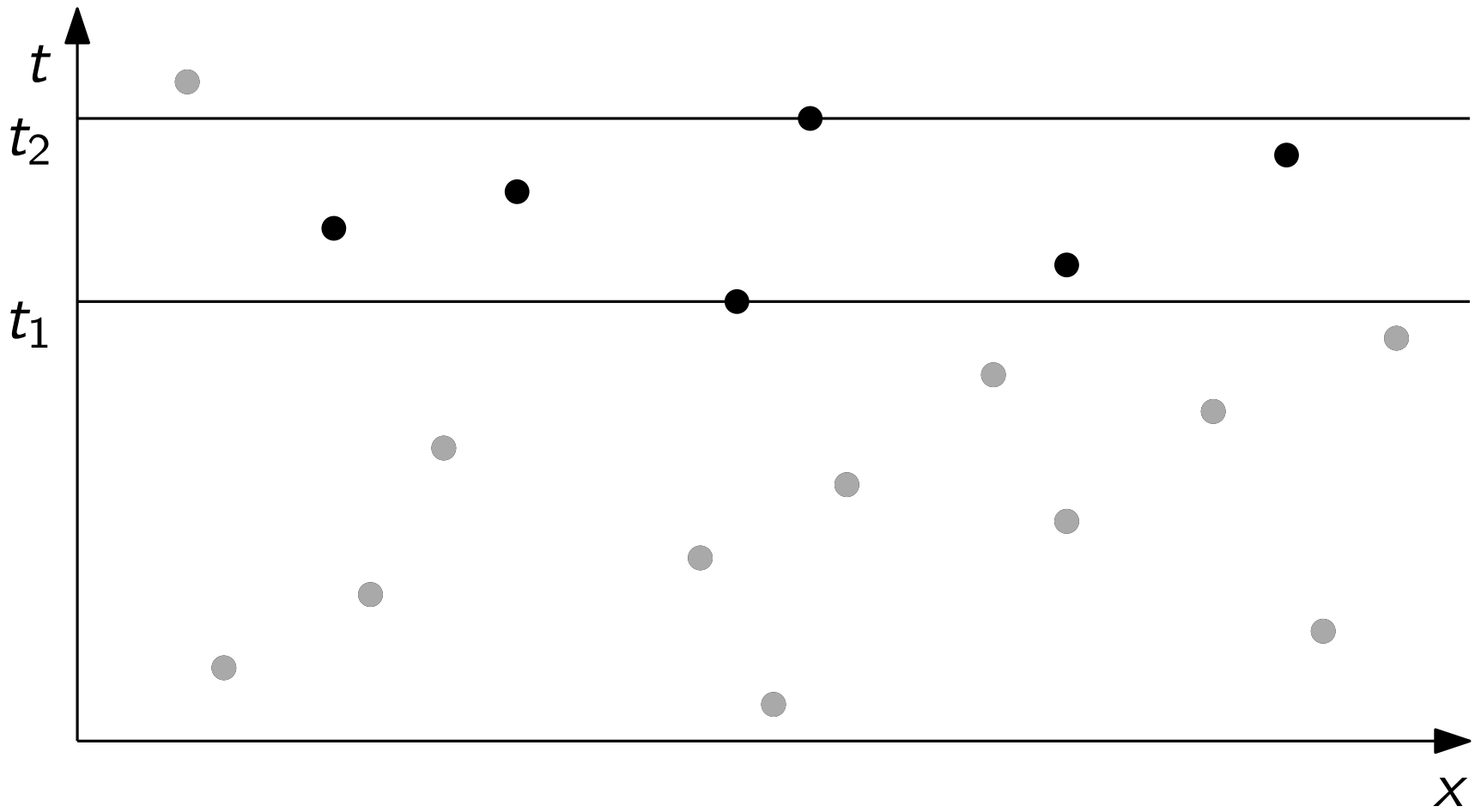
Preprocess n time-labeled points in $\mathbb{R}^d$
s.t. we can determine if $\exists p, q : d(p, q) < 1$.



Time-Windowed Closest Pair Decision Problem

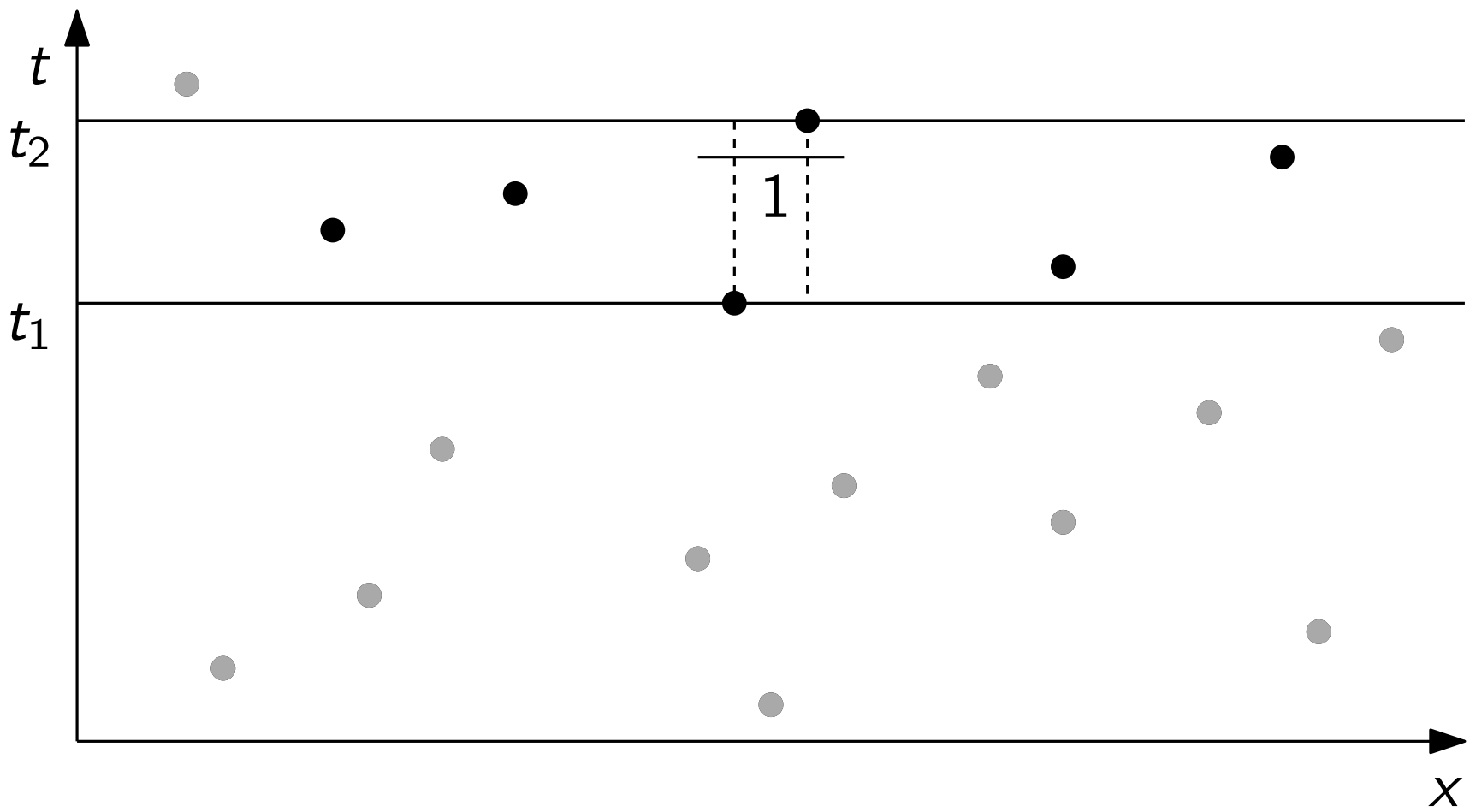
Preprocess n time-labeled points in $\mathbb{R}^d$
s.t. we can determine if $\exists p, q : d(p, q) < 1$.

$\overline{\qquad 1 \qquad}$



Time-Windowed Closest Pair Decision Problem

Preprocess n time-labeled points in $\mathbb{R}^d$
s.t. we can determine if $\exists p, q : d(p, q) < 1$.

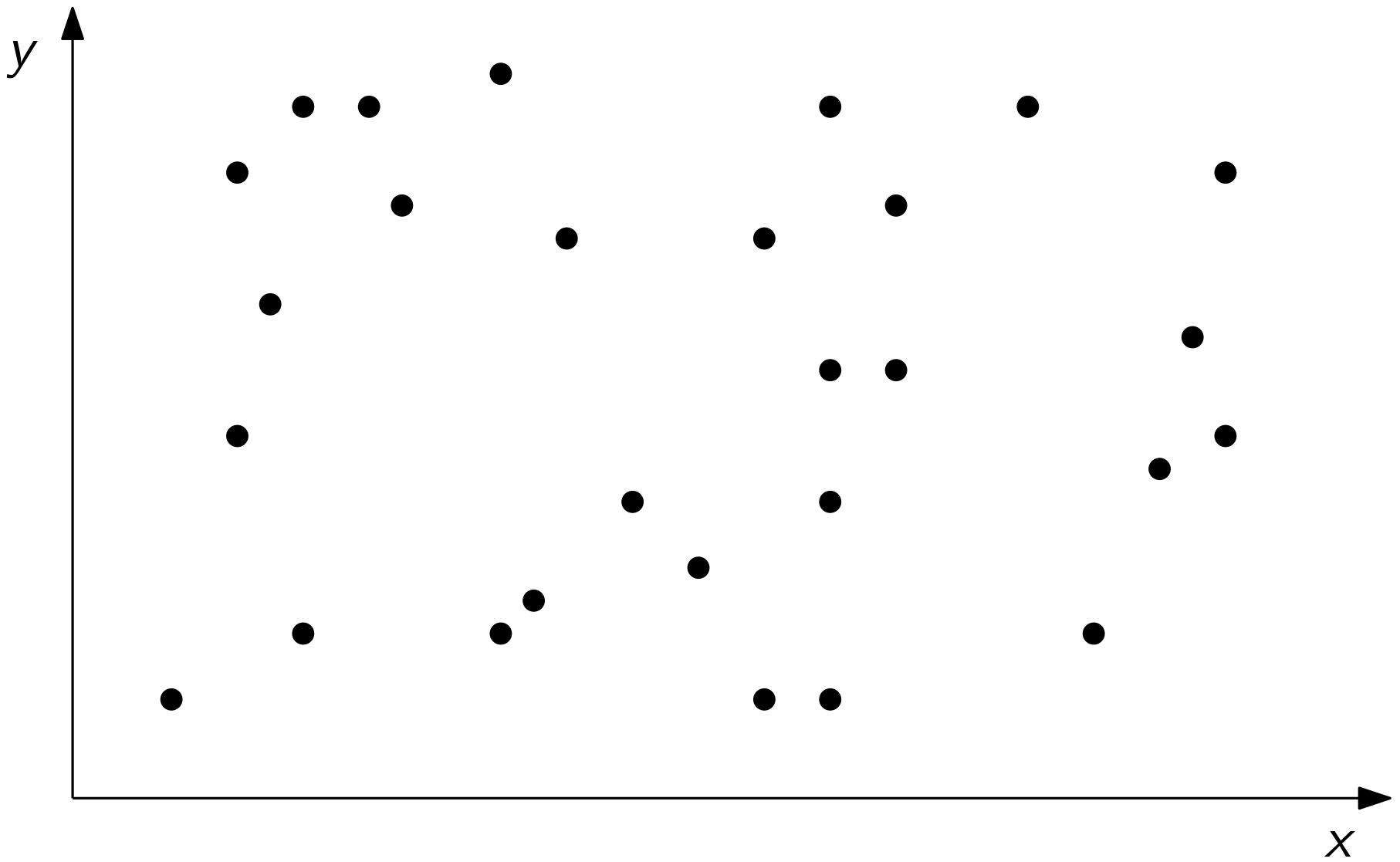


First Idea: Use a Grid

- Naively $O(n^2)$ candidate pairs
- Grid: reduce candidate pairs to $O(n)$

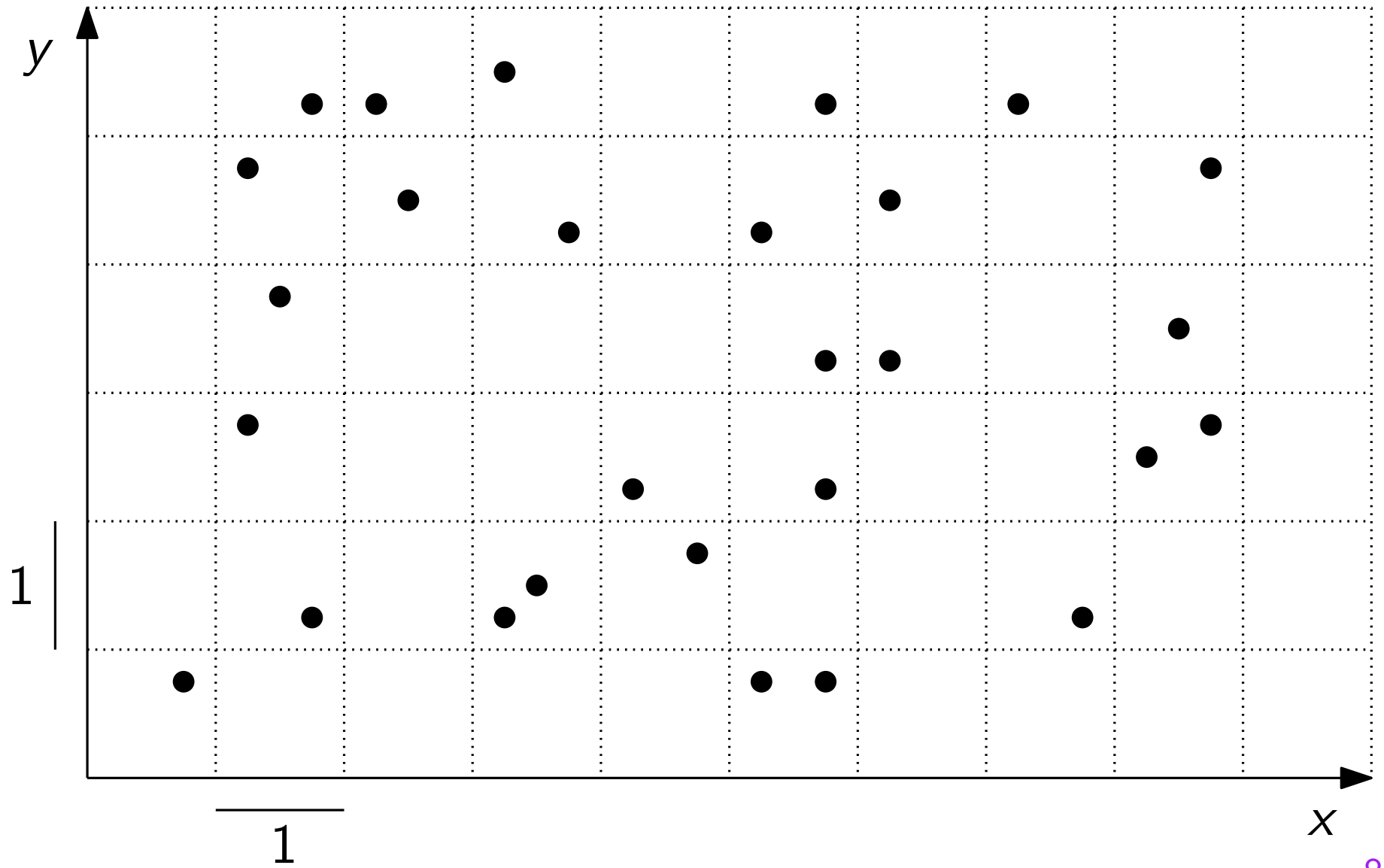
First Idea: Use a Grid

- Naively $O(n^2)$ candidate pairs
- Grid: reduce candidate pairs to $O(n)$



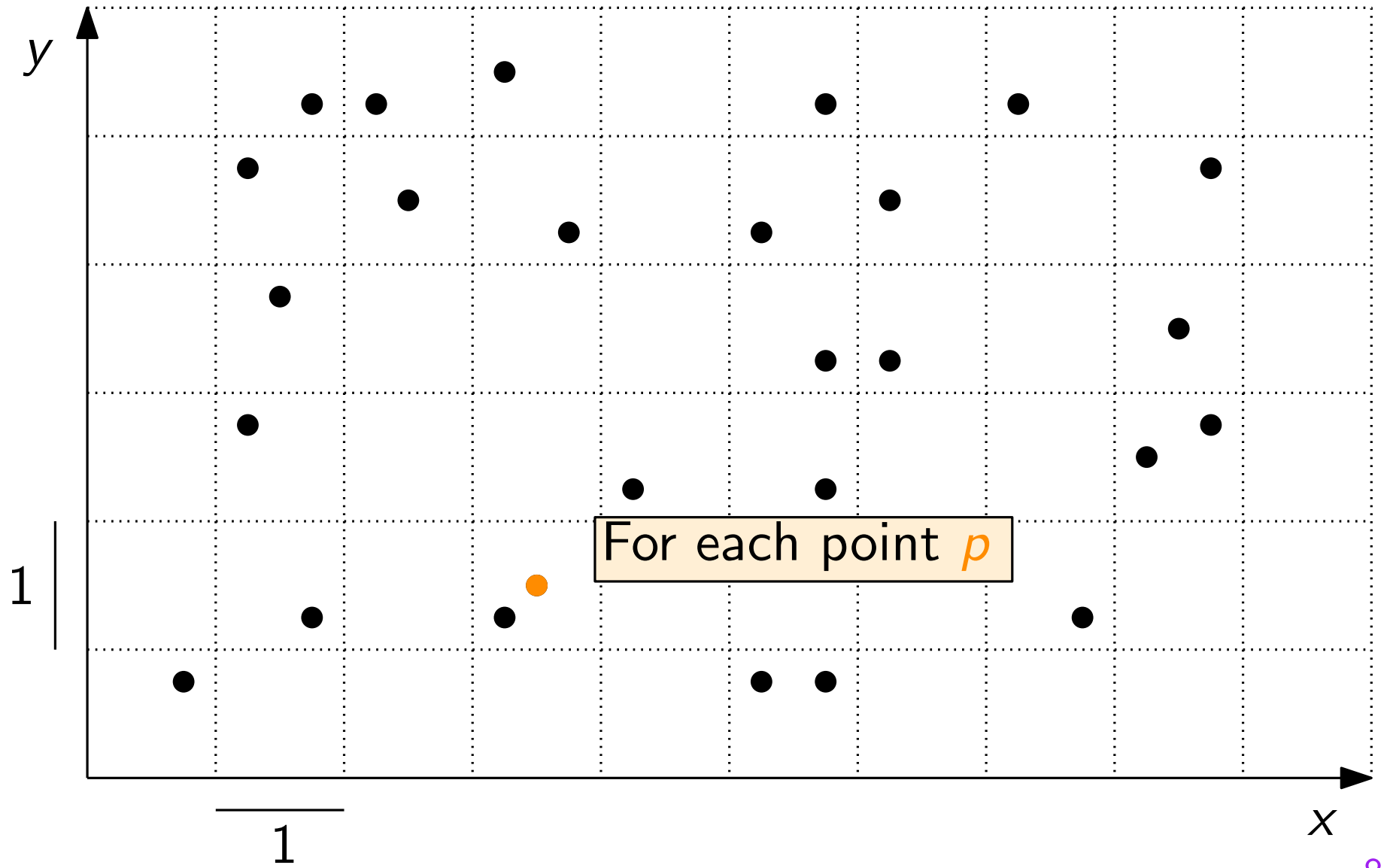
First Idea: Use a Grid

- Naively $O(n^2)$ candidate pairs
- Grid: reduce candidate pairs to $O(n)$



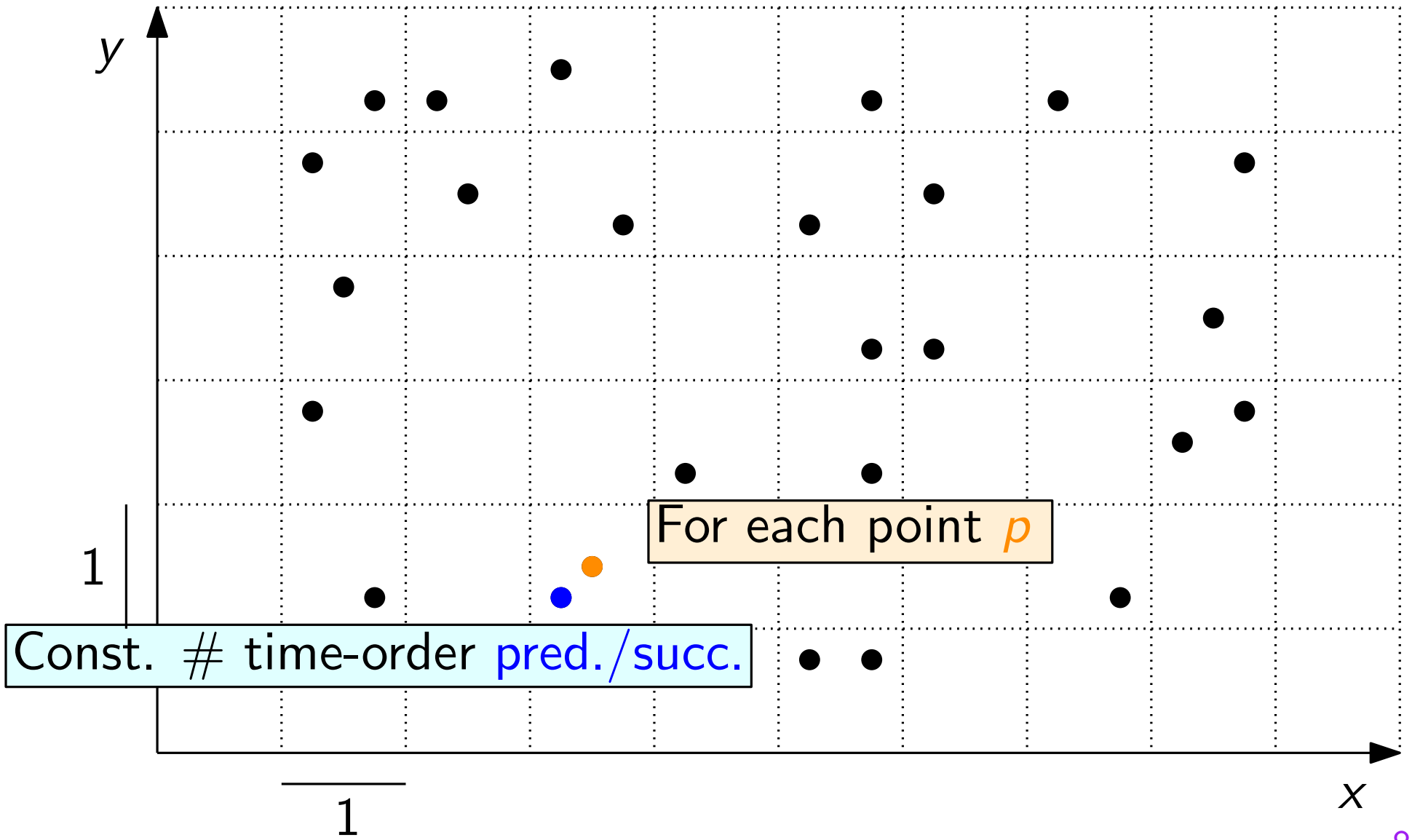
First Idea: Use a Grid

- Naively $O(n^2)$ candidate pairs
- Grid: reduce candidate pairs to $O(n)$



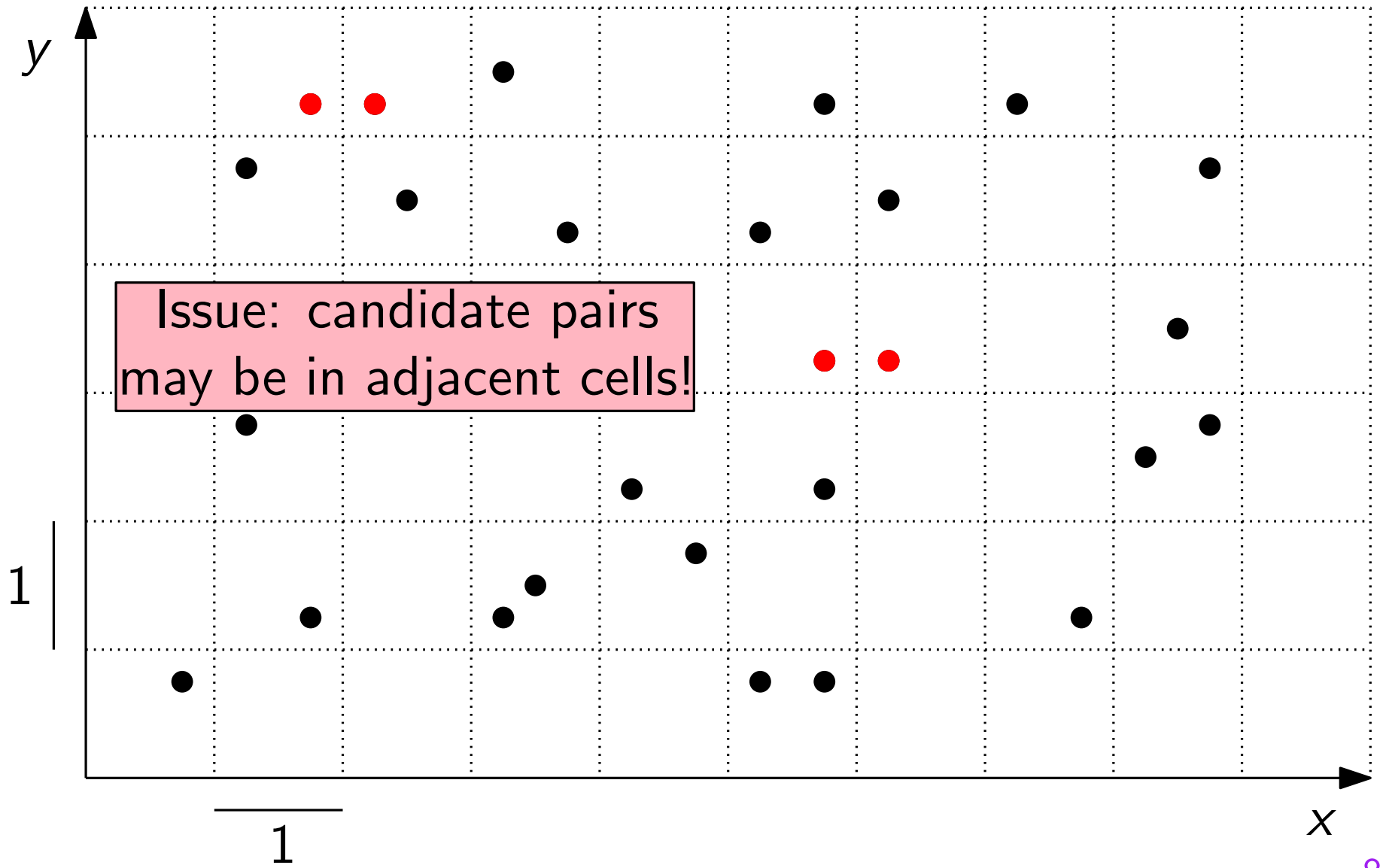
First Idea: Use a Grid

- Naively $O(n^2)$ candidate pairs
- Grid: reduce candidate pairs to $O(n)$



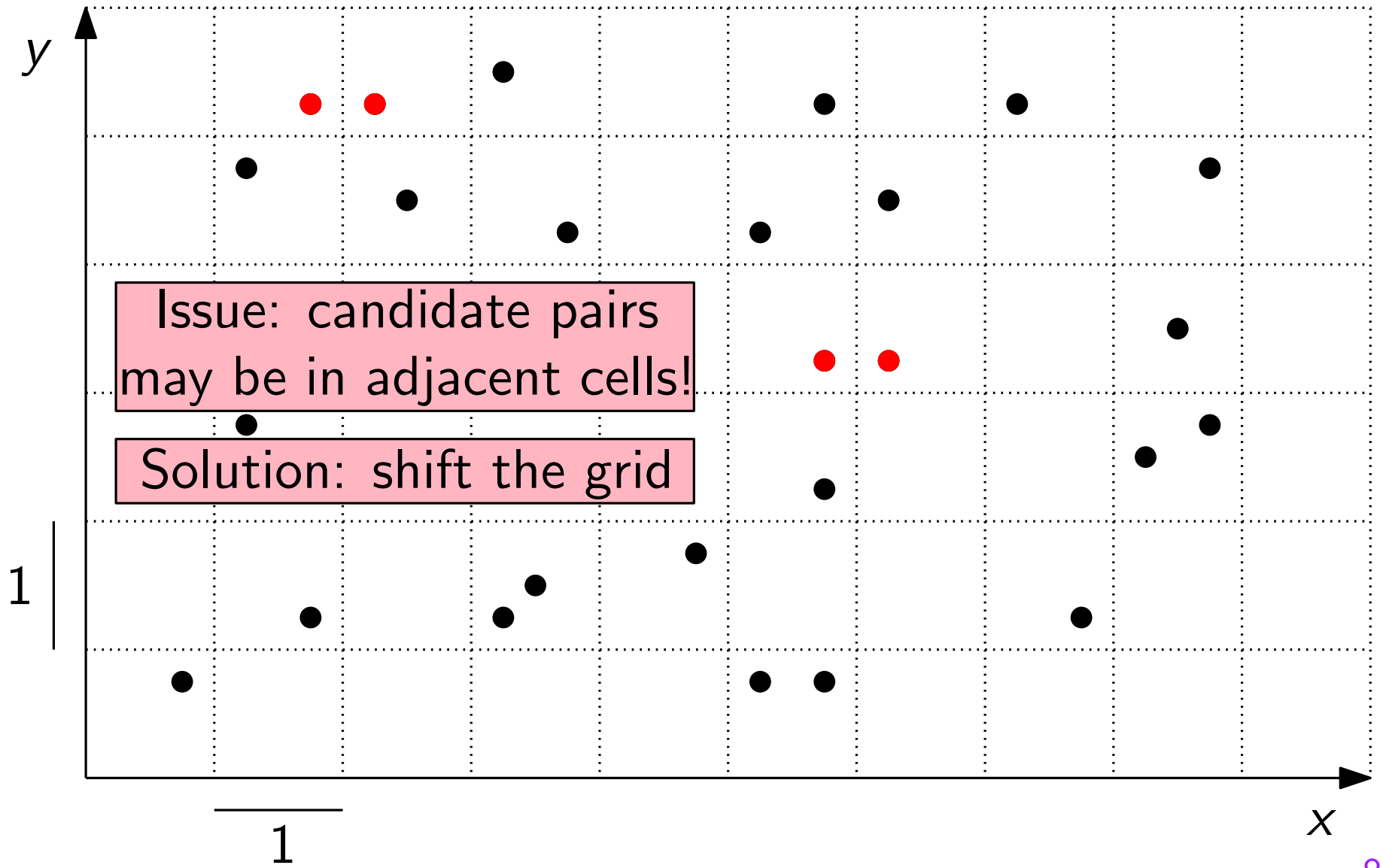
First Idea: Use a Grid

- Naively $O(n^2)$ candidate pairs
- Grid: reduce candidate pairs to $O(n)$



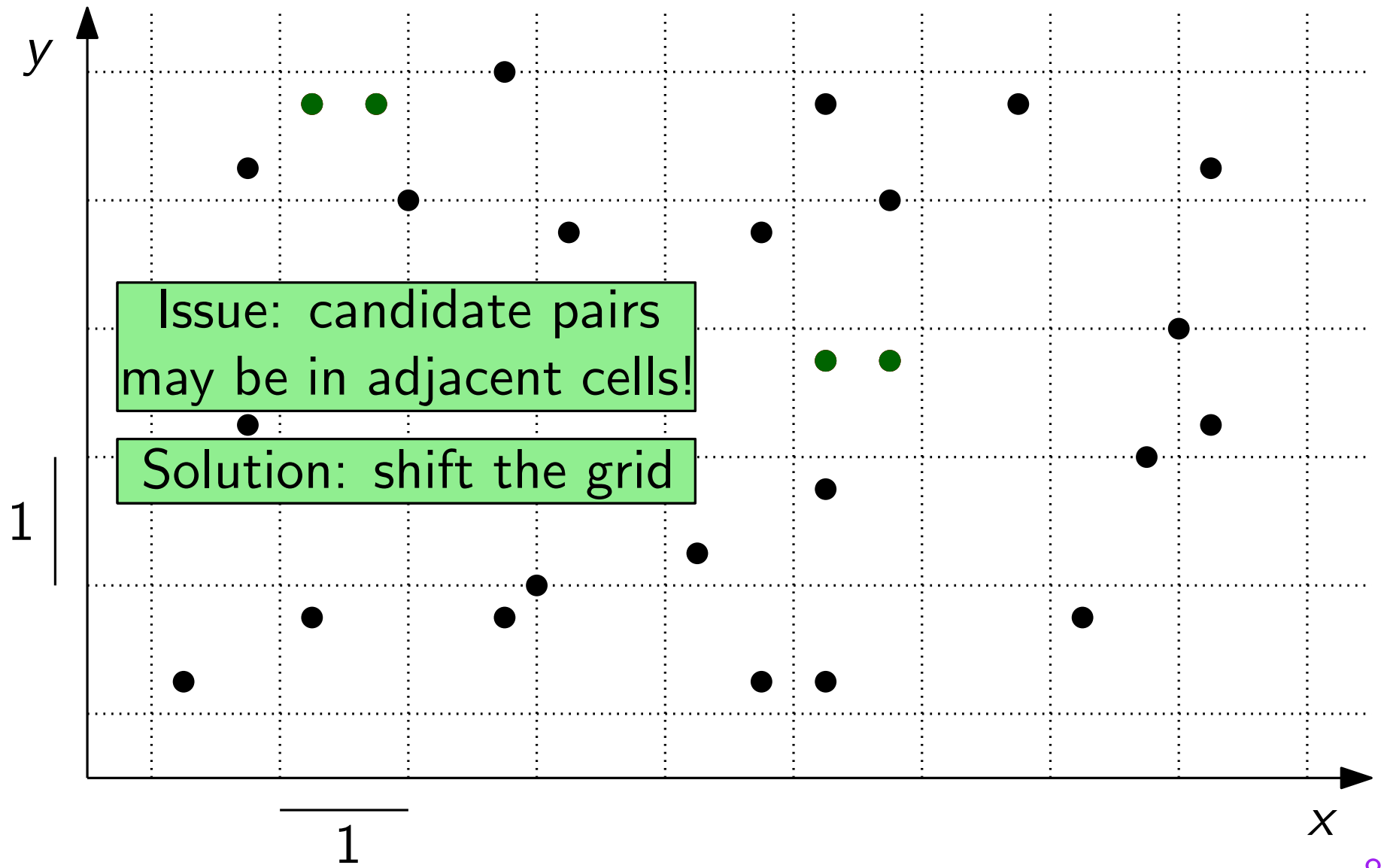
First Idea: Use a Grid

- Naively $O(n^2)$ candidate pairs
- Grid: reduce candidate pairs to $O(n)$



First Idea: Use a Grid

- Naively $O(n^2)$ candidate pairs
- Grid: reduce candidate pairs to $O(n)$



Reduction to 2D Dominance Range Emptiness

Reduction to 2D Dominance Range Emptiness

- Have $O(n)$ *satisfying* pairs (p, q)

Reduction to 2D Dominance Range Emptiness

- Have $O(n)$ *satisfying* pairs (p, q)
- Assume $t(p) \leq t(q)$

Reduction to 2D Dominance Range Emptiness

- Have $O(n)$ *satisfying* pairs (p, q)
- Assume $t(p) \leq t(q)$
- Query time window $[t_1, t_2]$

Reduction to 2D Dominance Range Emptiness

- Have $O(n)$ *satisfying* pairs (p, q)
- Assume $t(p) \leq t(q)$
- Query time window $[t_1, t_2]$
- $t_1 \leq t(p)$ and $t(q) \leq t_2$

Reduction to 2D Dominance Range Emptiness

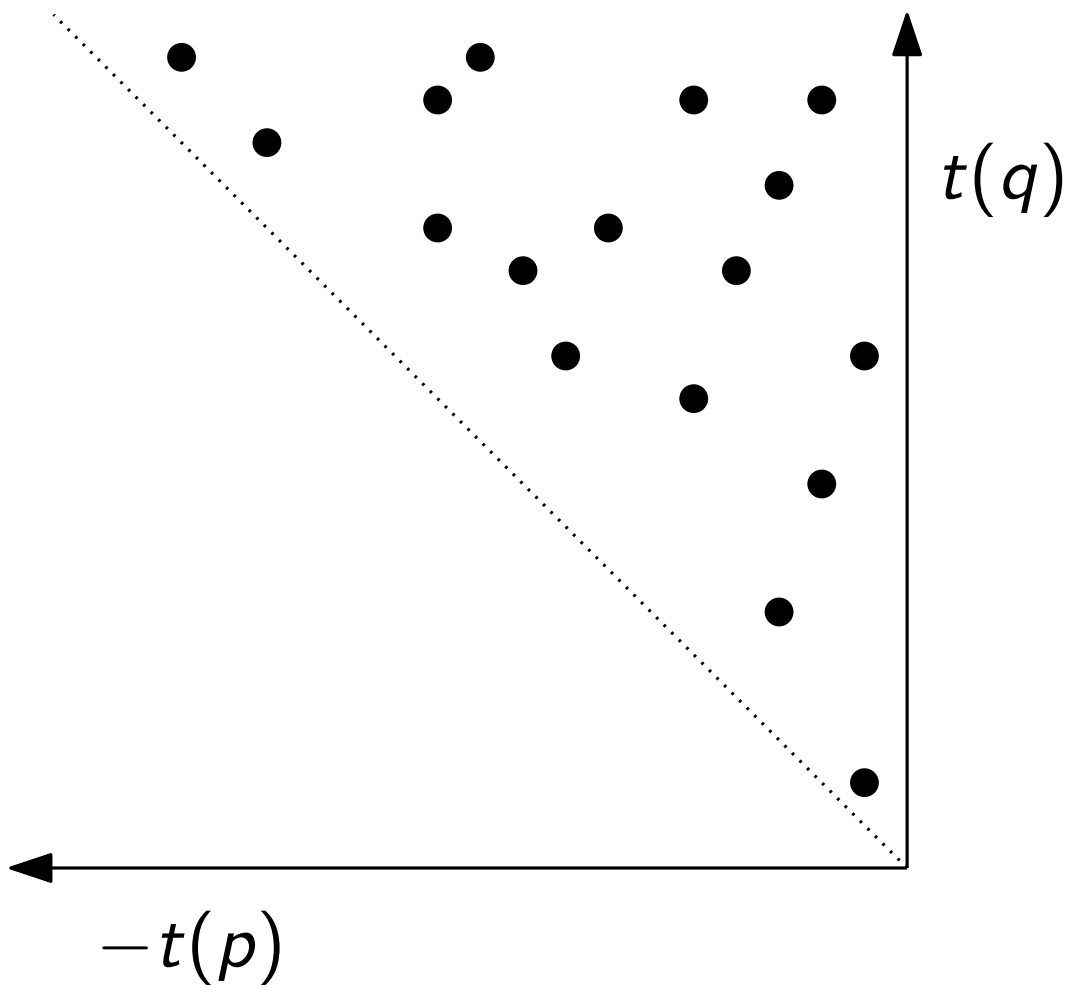
- Have $O(n)$ *satisfying* pairs (p, q)
- Assume $t(p) \leq t(q)$
- Query time window $[t_1, t_2]$
- $-t(p) \leq -t_1$ and $t(q) \leq t_2$

Dominance!

Reduction to 2D Dominance Range Emptiness

- Have $O(n)$ *satisfying* pairs (p, q)
- Assume $t(p) \leq t(q)$
- Query time window $[t_1, t_2]$
- $-t(p) \leq -t_1$ and $t(q) \leq t_2$

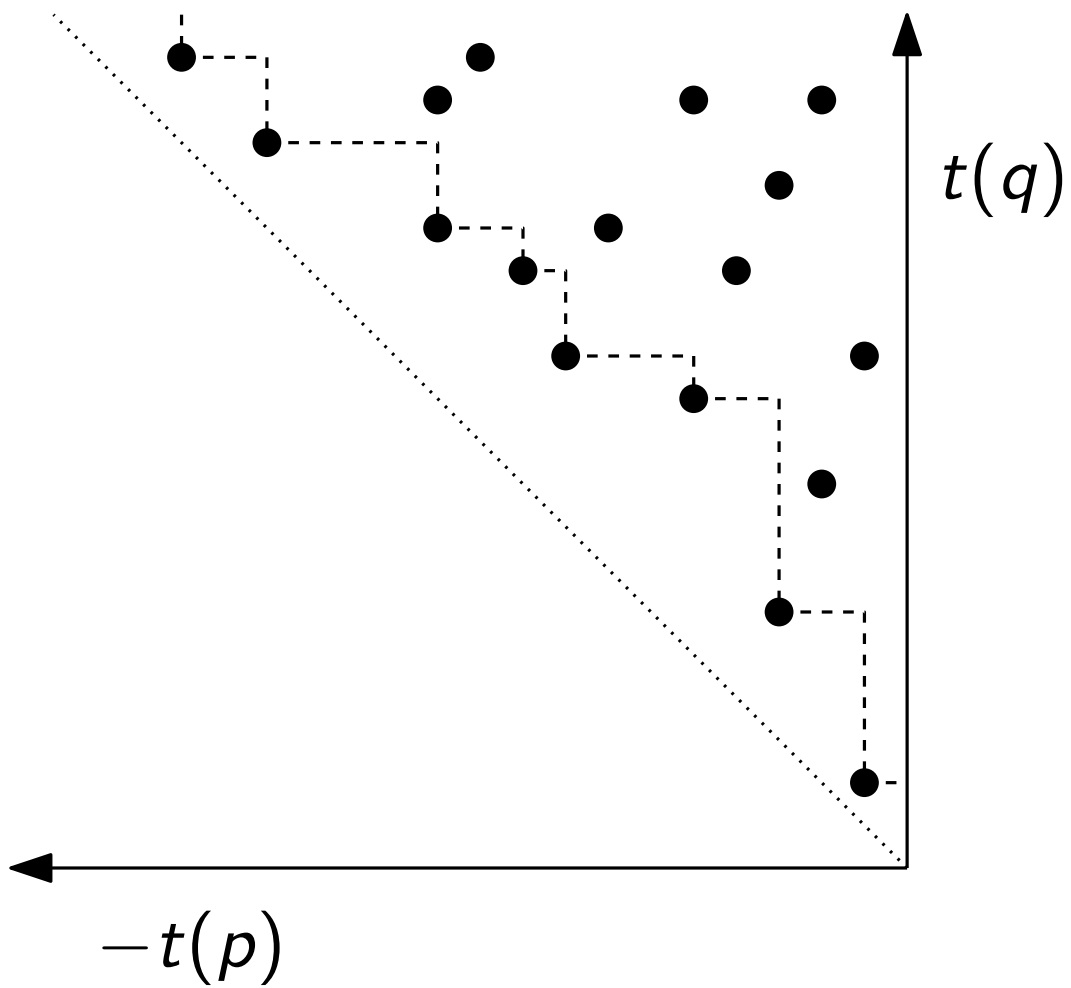
Dominance!



Reduction to 2D Dominance Range Emptiness

- Have $O(n)$ *satisfying* pairs (p, q)
- Assume $t(p) \leq t(q)$
- Query time window $[t_1, t_2]$
- $-t(p) \leq -t_1$ and $t(q) \leq t_2$

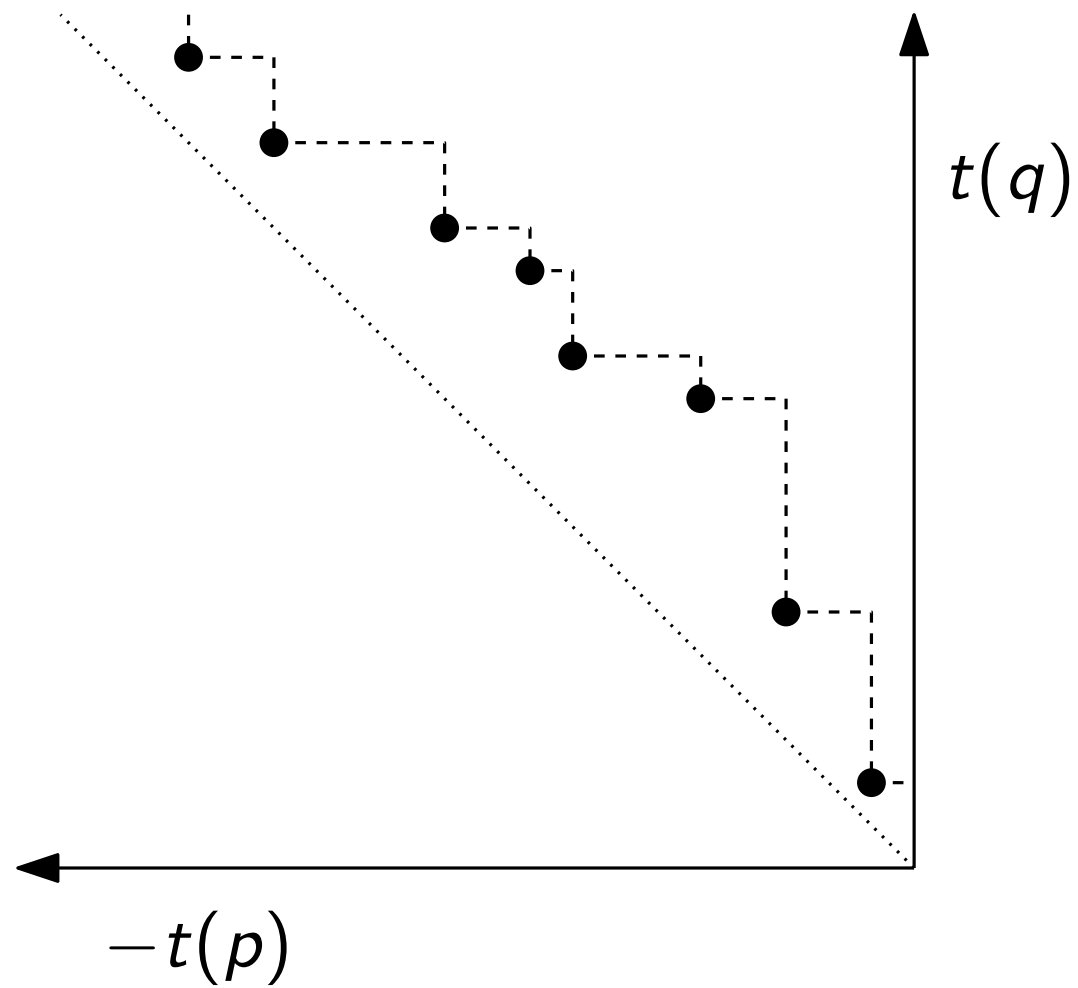
Dominance!



Reduction to 2D Dominance Range Emptiness

- Have $O(n)$ *satisfying* pairs (p, q)
- Assume $t(p) \leq t(q)$
- Query time window $[t_1, t_2]$
- $-t(p) \leq -t_1$ and $t(q) \leq t_2$

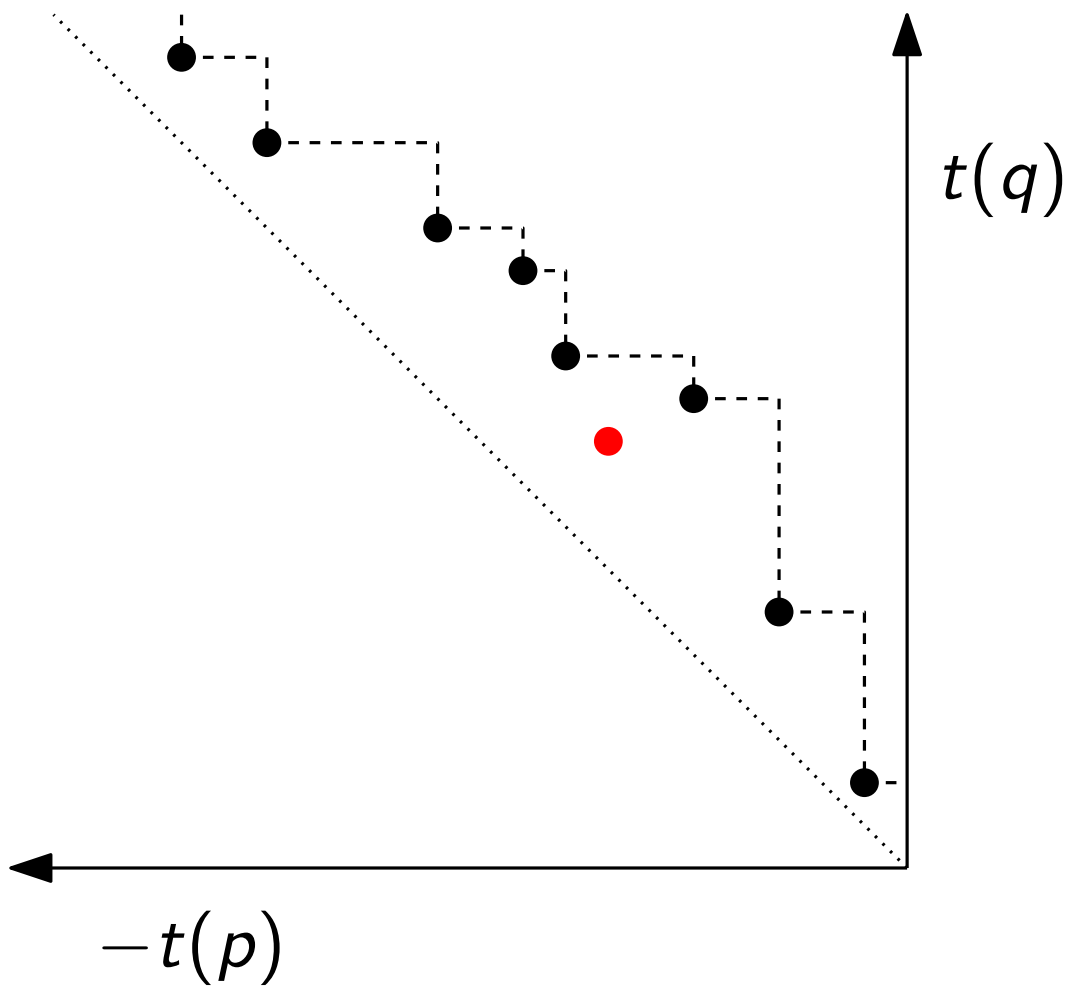
Dominance!



Reduction to 2D Dominance Range Emptiness

- Have $O(n)$ *satisfying* pairs (p, q)
- Assume $t(p) \leq t(q)$
- Query time window $[t_1, t_2]$
- $-t(p) \leq -t_1$ and $t(q) \leq t_2$

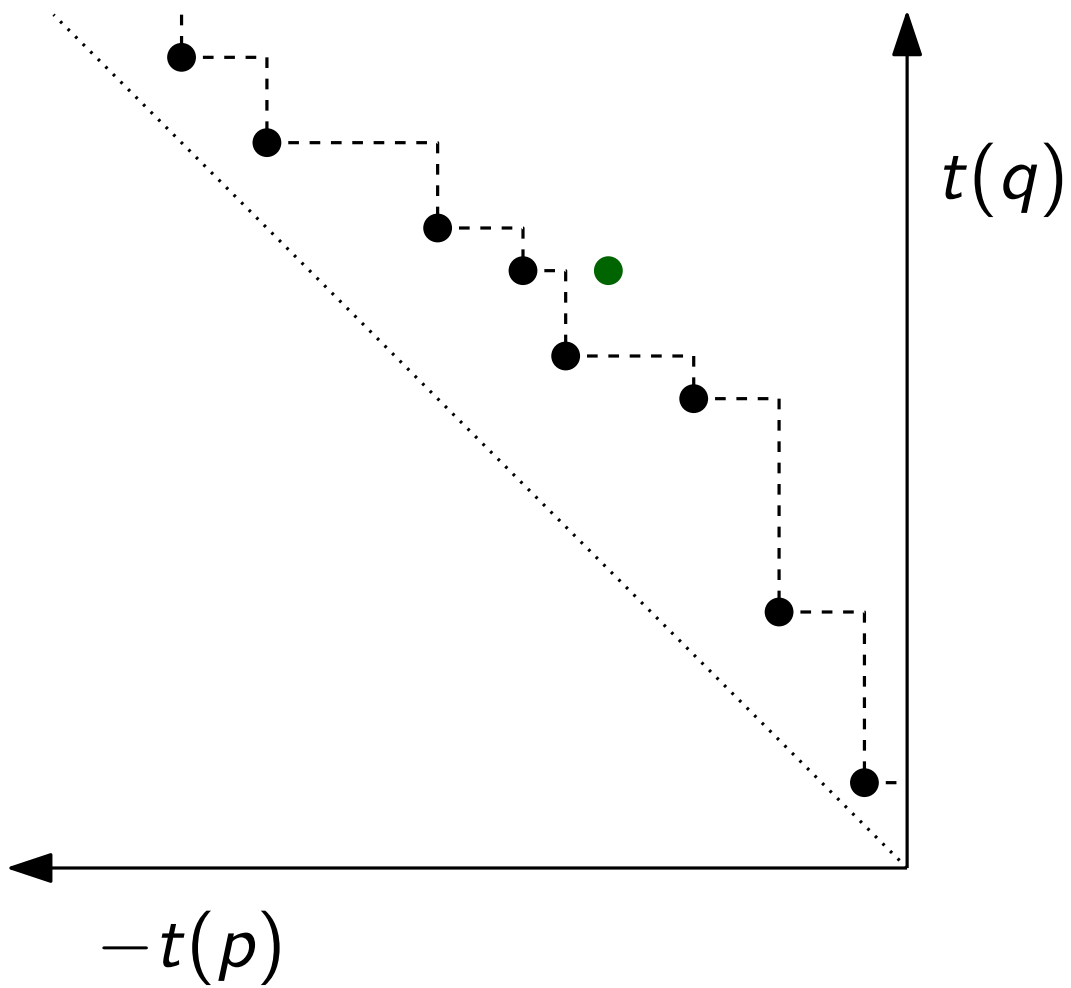
Dominance!



Reduction to 2D Dominance Range Emptiness

- Have $O(n)$ *satisfying* pairs (p, q)
- Assume $t(p) \leq t(q)$
- Query time window $[t_1, t_2]$
- $-t(p) \leq -t_1$ and $t(q) \leq t_2$

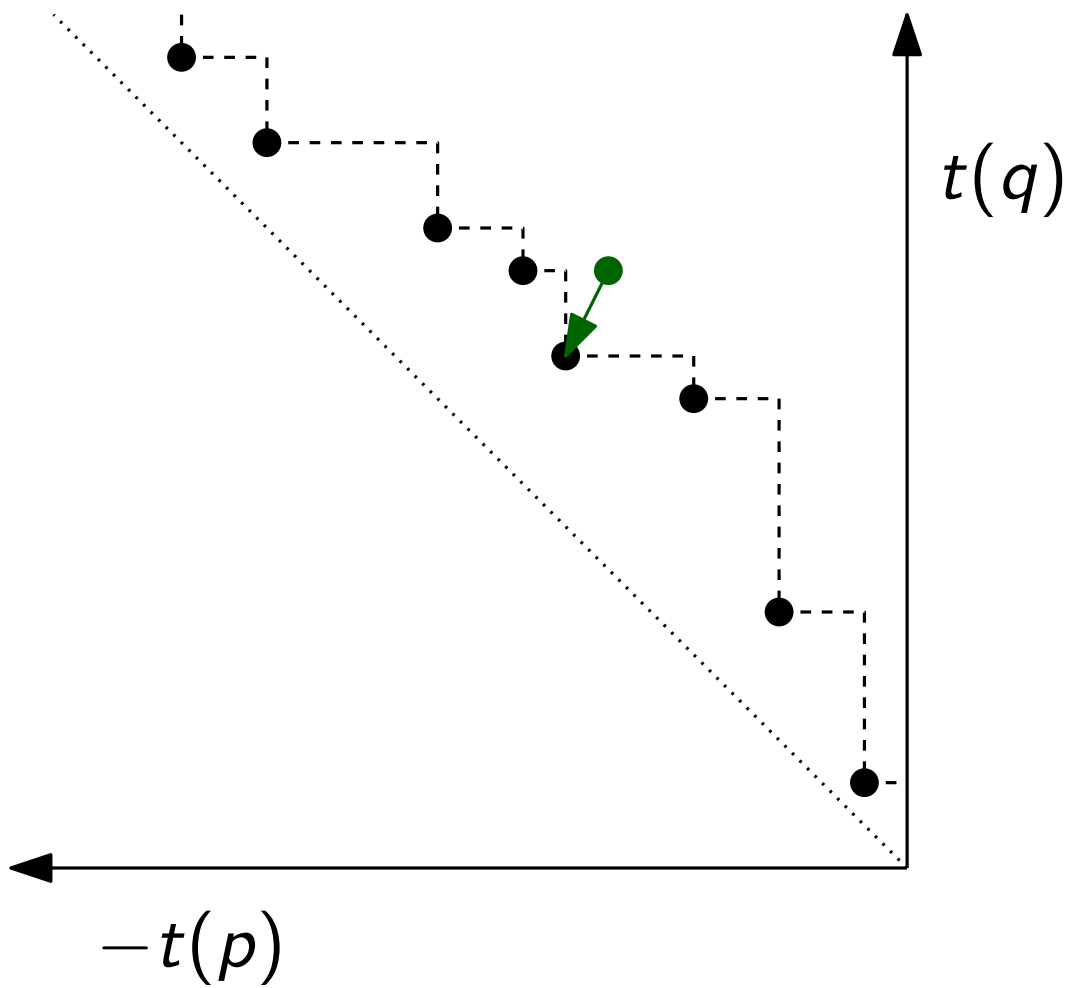
Dominance!



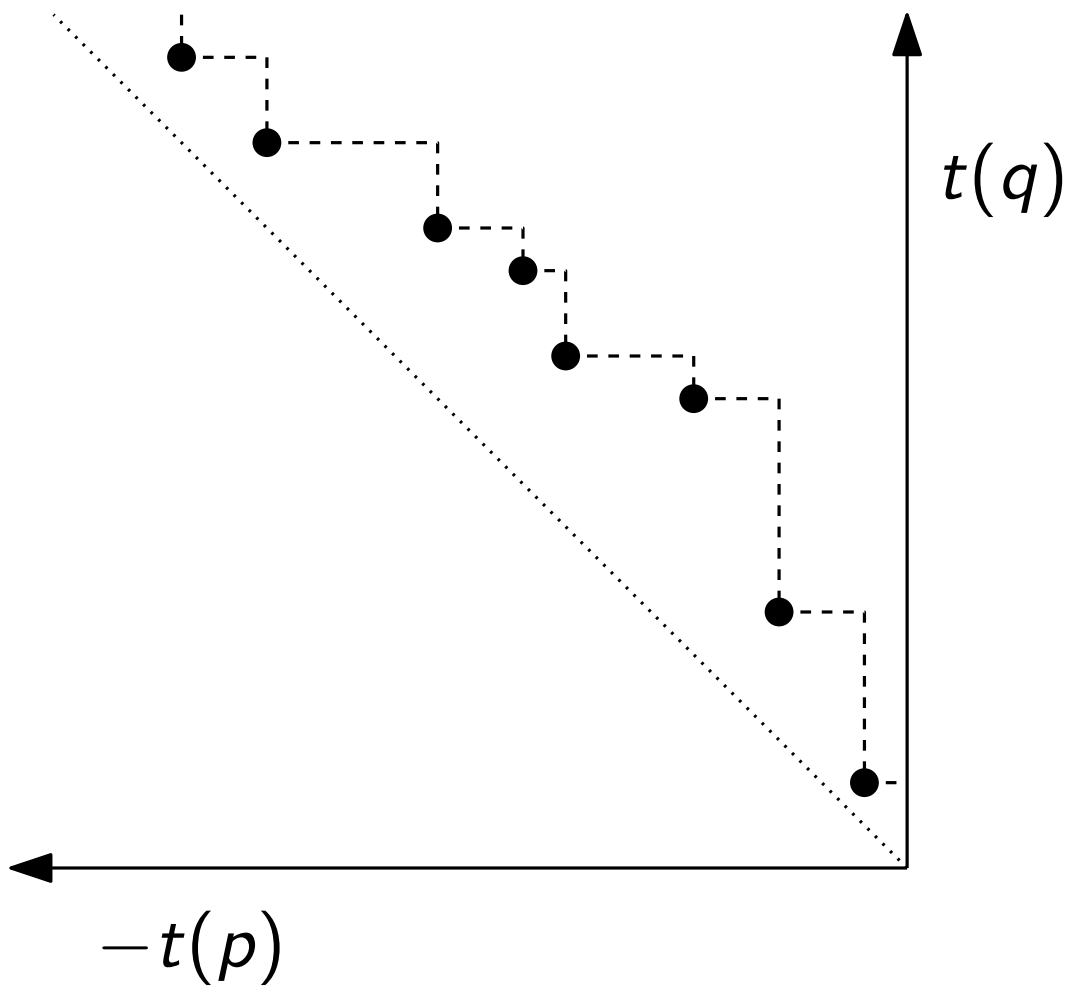
Reduction to 2D Dominance Range Emptiness

- Have $O(n)$ *satisfying* pairs (p, q)
- Assume $t(p) \leq t(q)$
- Query time window $[t_1, t_2]$
- $-t(p) \leq -t_1$ and $t(q) \leq t_2$

Dominance!

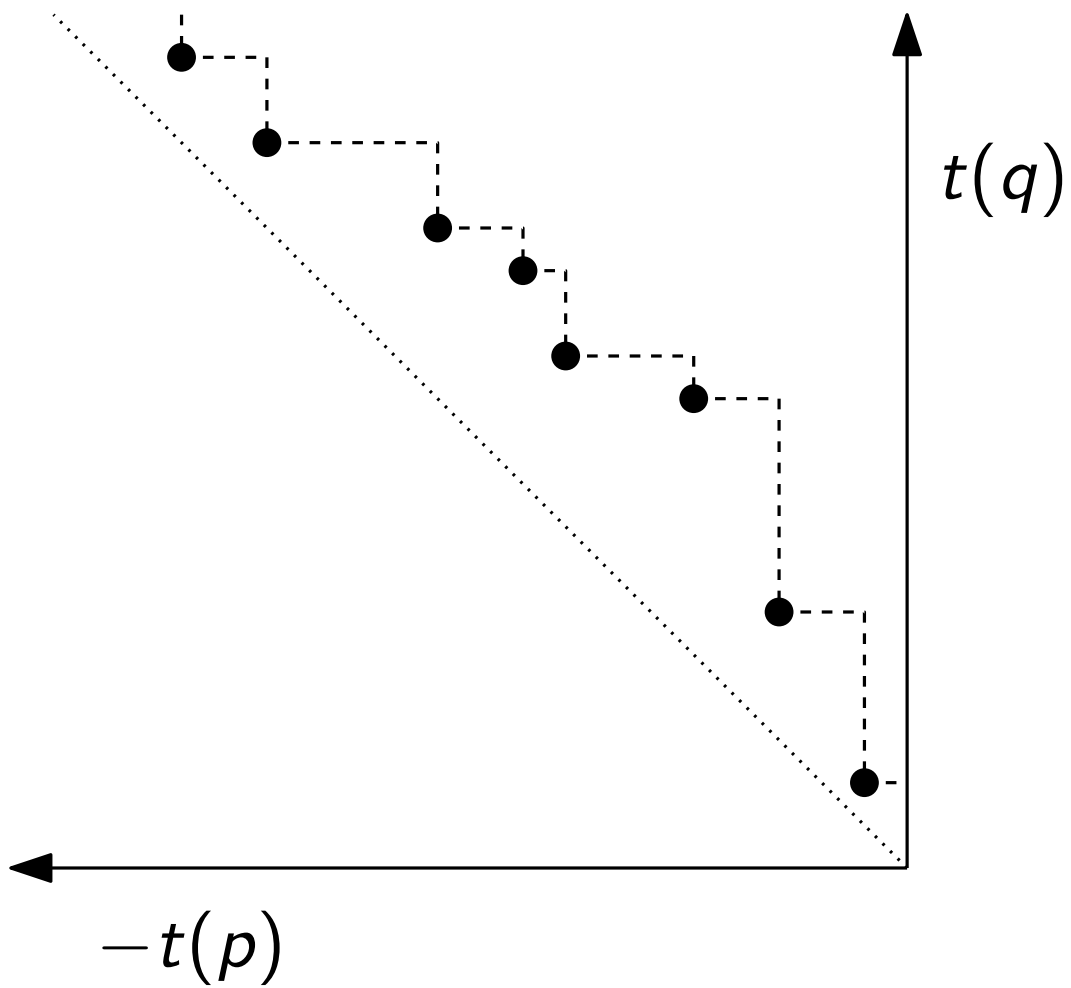


Succinct Representation



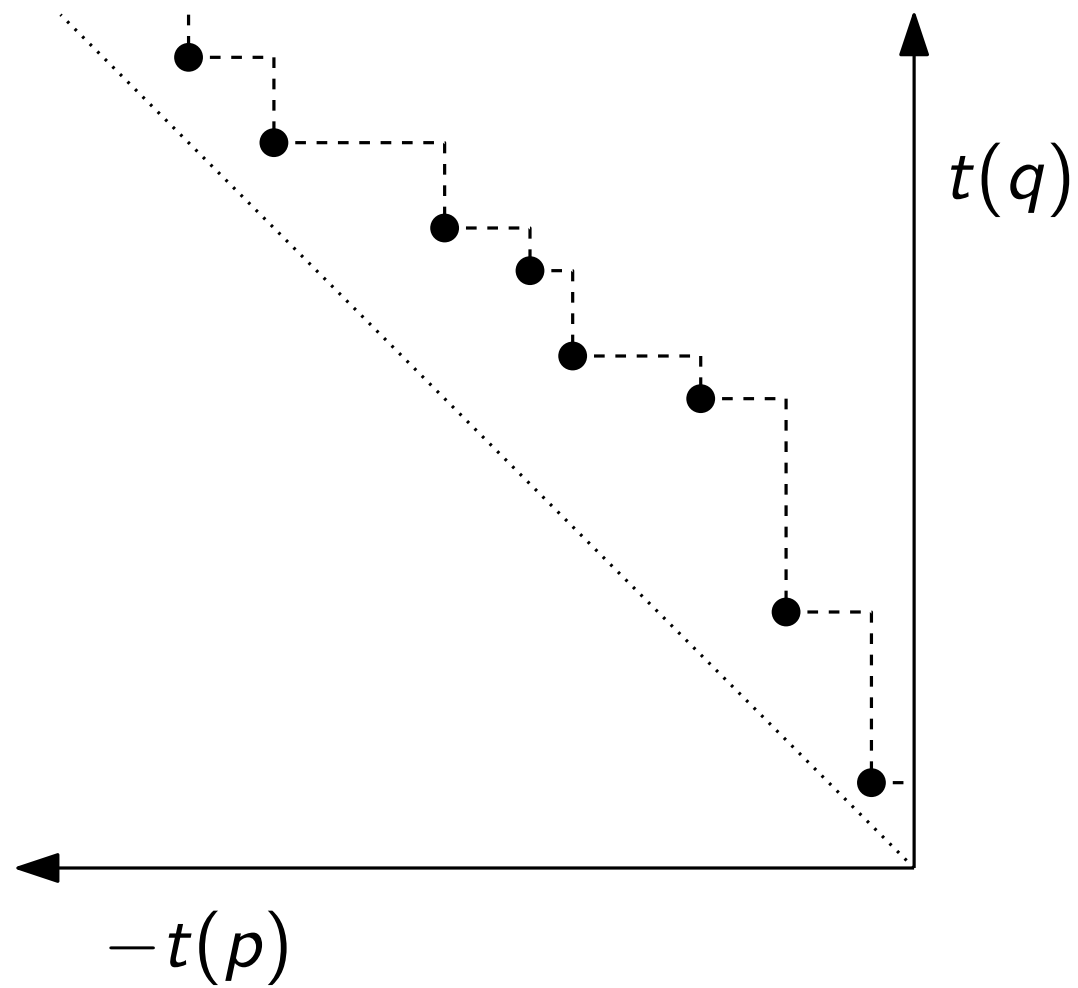
Succinct Representation

- Monotone chain



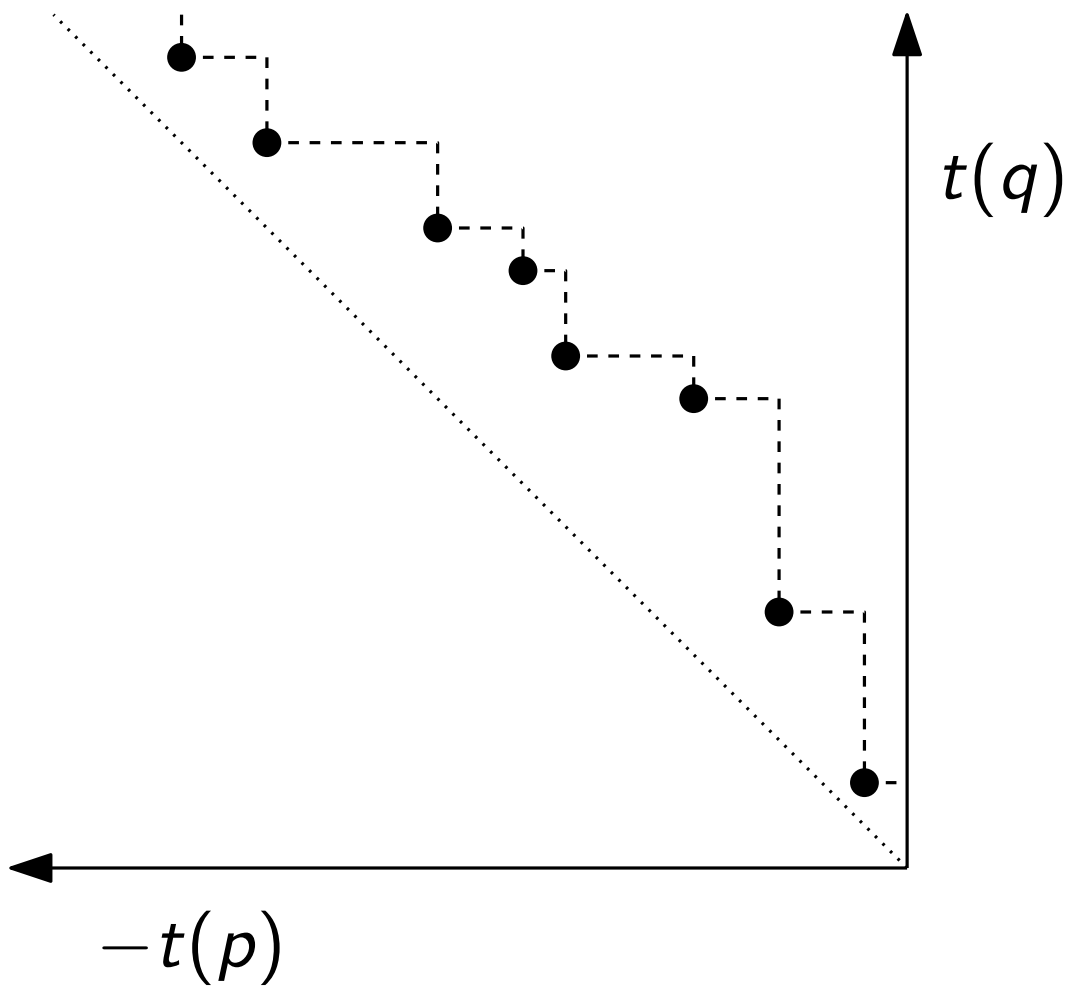
Succinct Representation

- Monotone chain
- Starts at $0, 0$



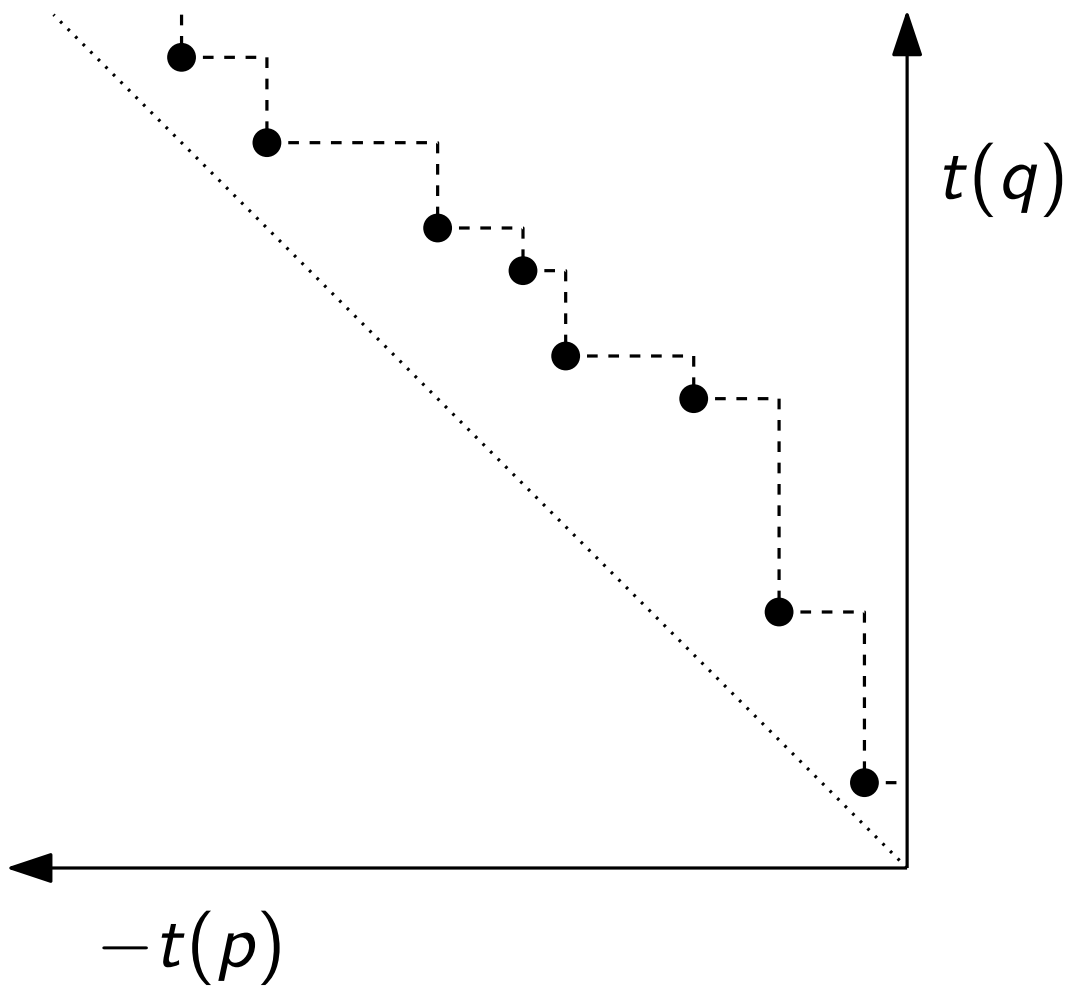
Succinct Representation

- Monotone chain
- Starts at $0, 0$
- Each step moves:



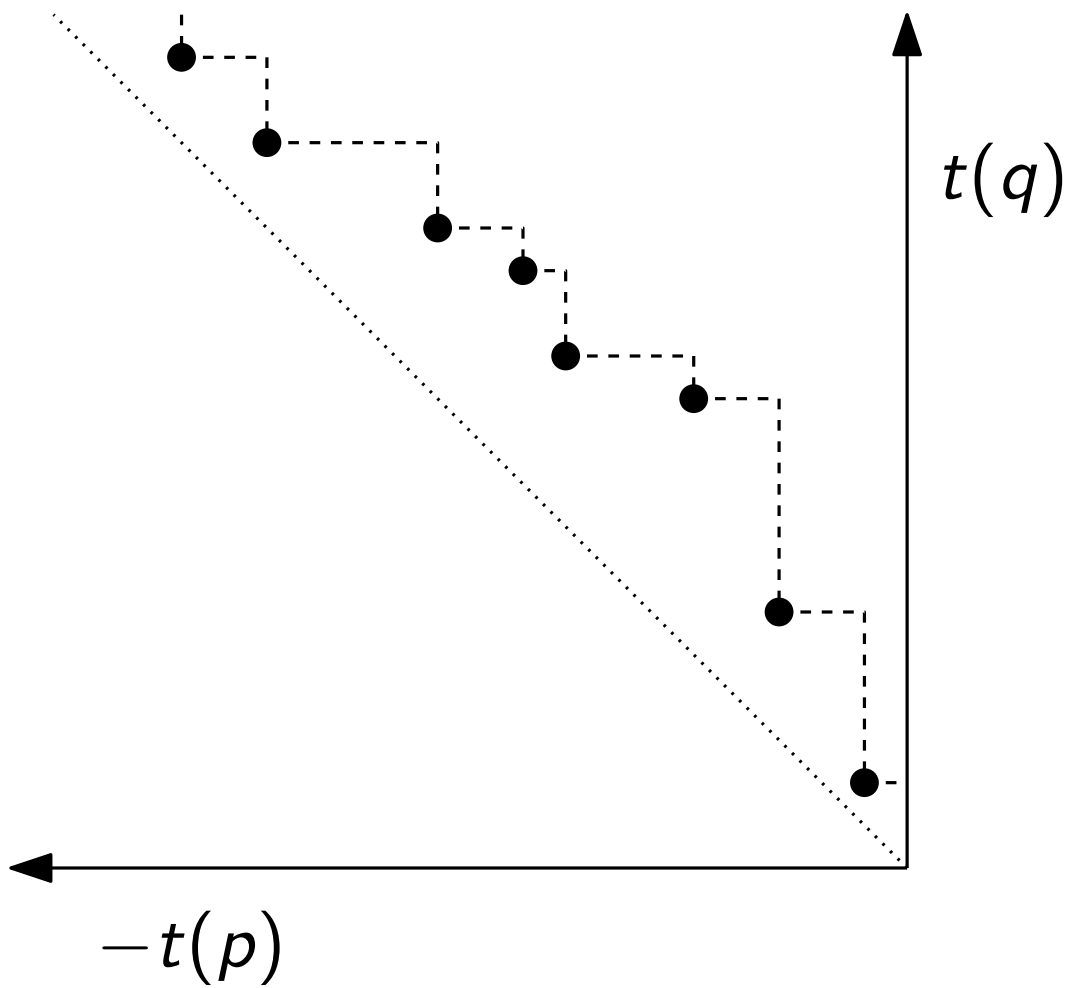
Succinct Representation

- Monotone chain
- Starts at $0, 0$
- Each step moves:
  - ◇ Horizontally



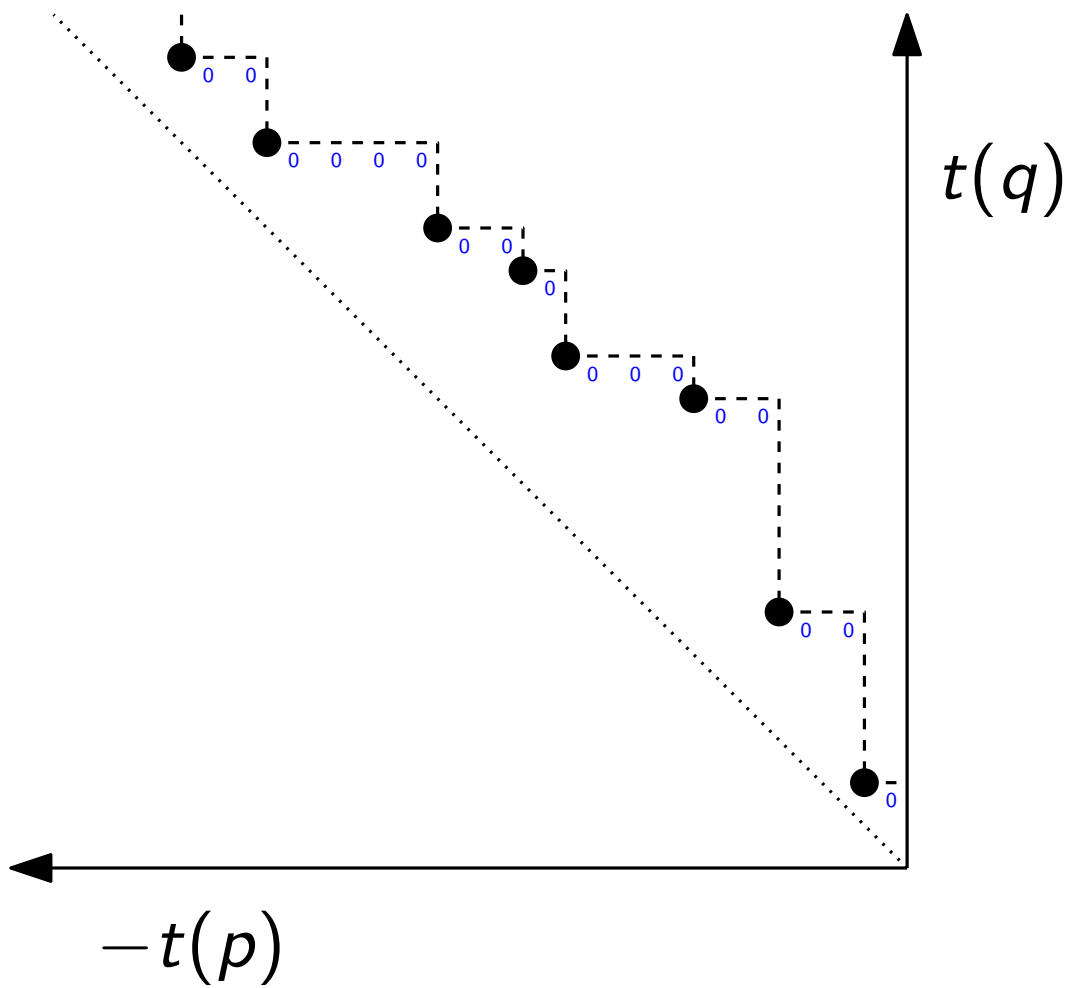
Succinct Representation

- Monotone chain
- Starts at $0, 0$
- Each step moves:
  - ◇ Horizontally
  - ◇ Vertically



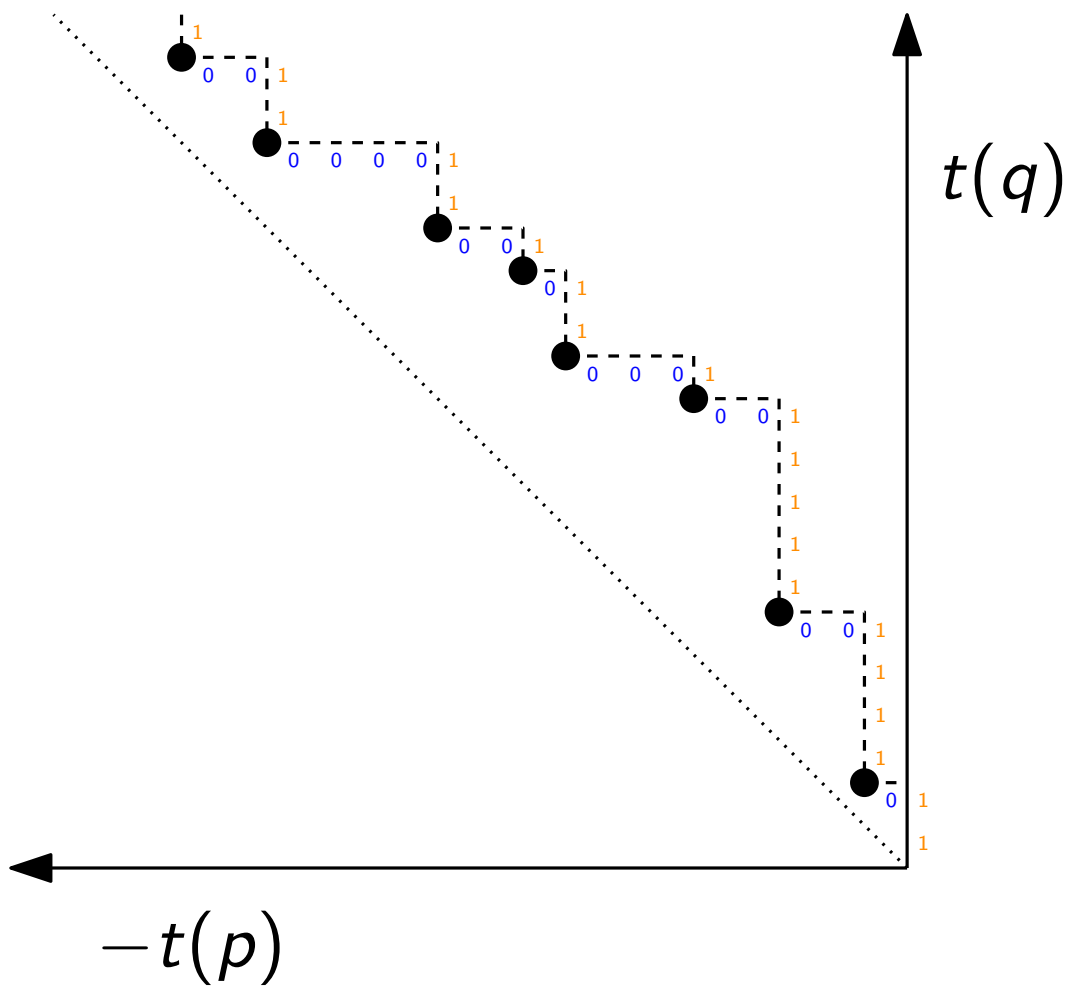
Succinct Representation

- Monotone chain
- Starts at $0, 0$
- Each step moves:
  - ◇ Horizontally 0
  - ◇ Vertically



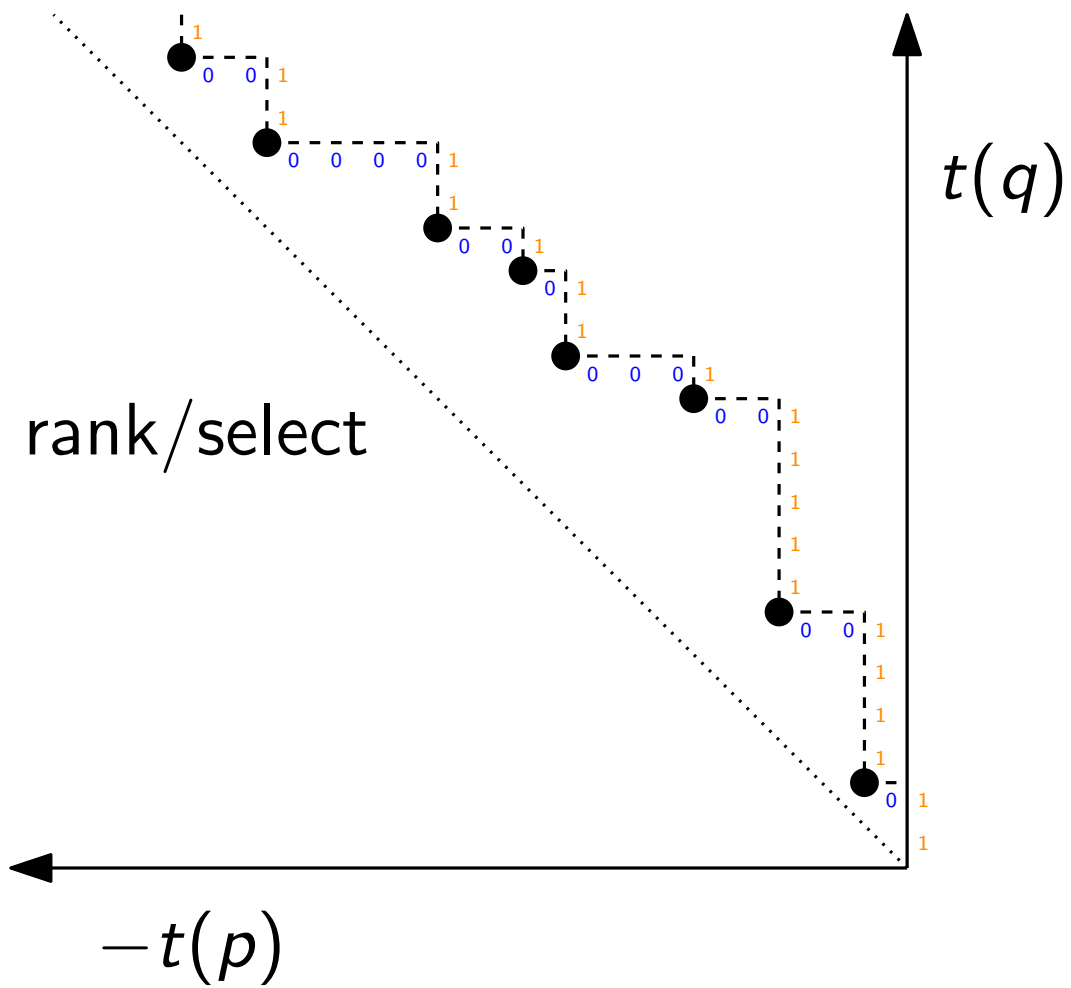
Succinct Representation

- Monotone chain
- Starts at $0, 0$
- Each step moves:
  - ◇ Horizontally 0
  - ◇ Vertically 1



Succinct Representation

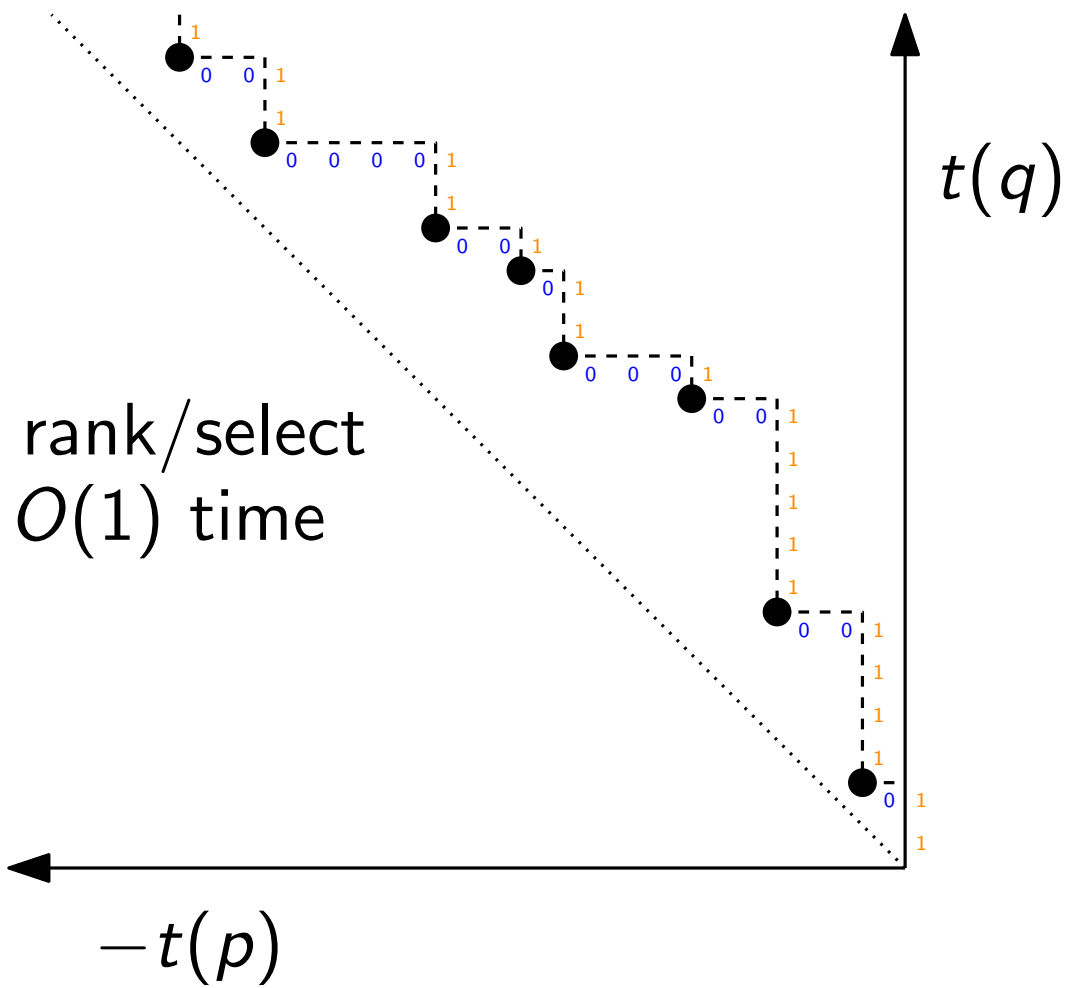
- Monotone chain
- Starts at $0, 0$
- Each step moves:
  - ◇ Horizontally 0
  - ◇ Vertically 1



Solve query $\rightarrow$ rank/select

Succinct Representation

- Monotone chain
- Starts at $0, 0$
- Each step moves:
  - ◇ Horizontally 0
  - ◇ Vertically 1



Solve query $\rightarrow$ rank/select
$O(1)$ time

Time-Windowed Closest Pair

Time-Windowed Closest Pair

- How to extend this solution to find the closest pair?

Time-Windowed Closest Pair

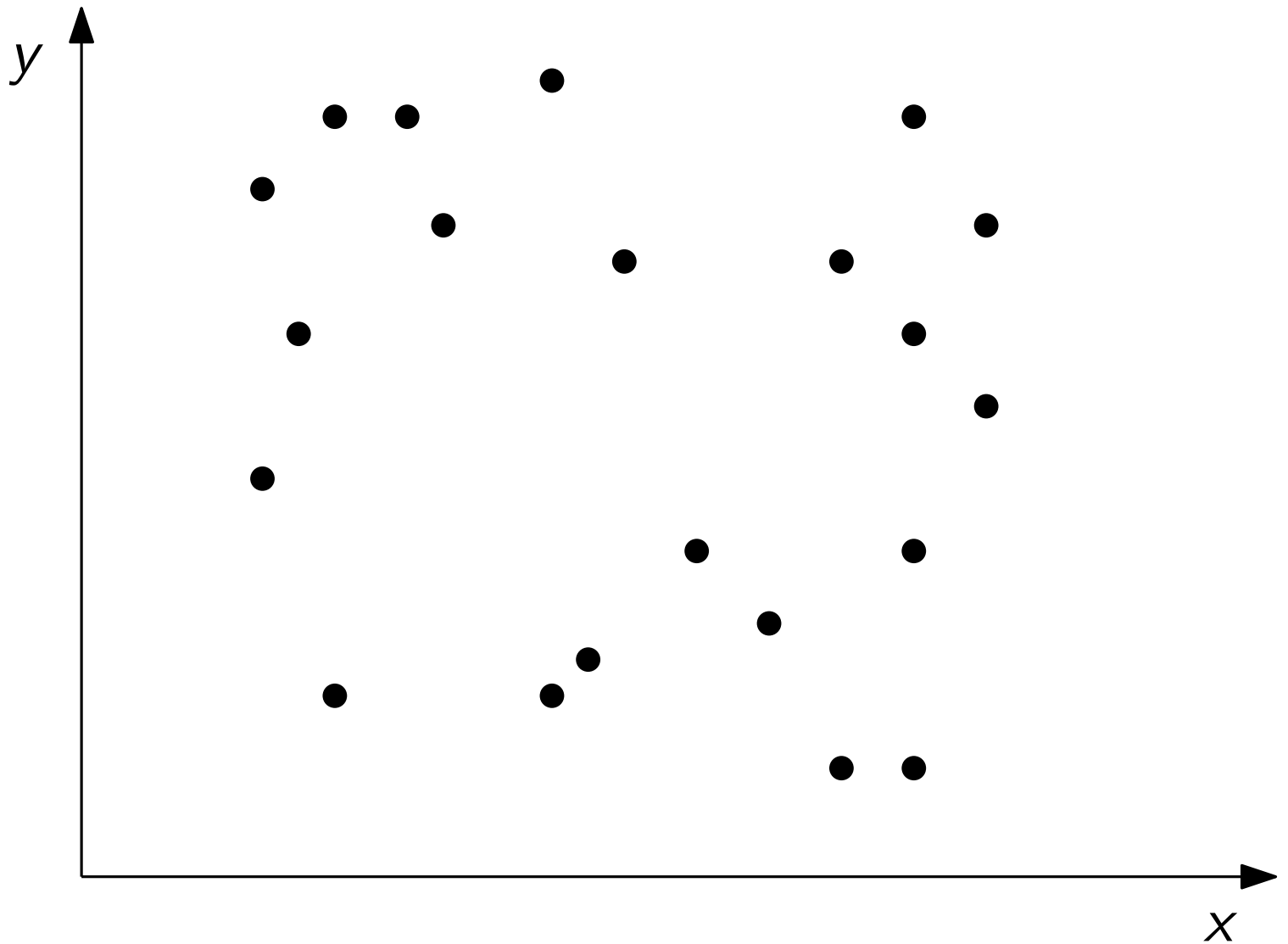
- How to extend this solution to find the closest pair?
- We don't know how big to make the grid cells.

Time-Windowed Closest Pair

- How to extend this solution to find the closest pair?
- We don't know how big to make the grid cells.
- Solution: a recursive grid structure (i.e., a quadtree).

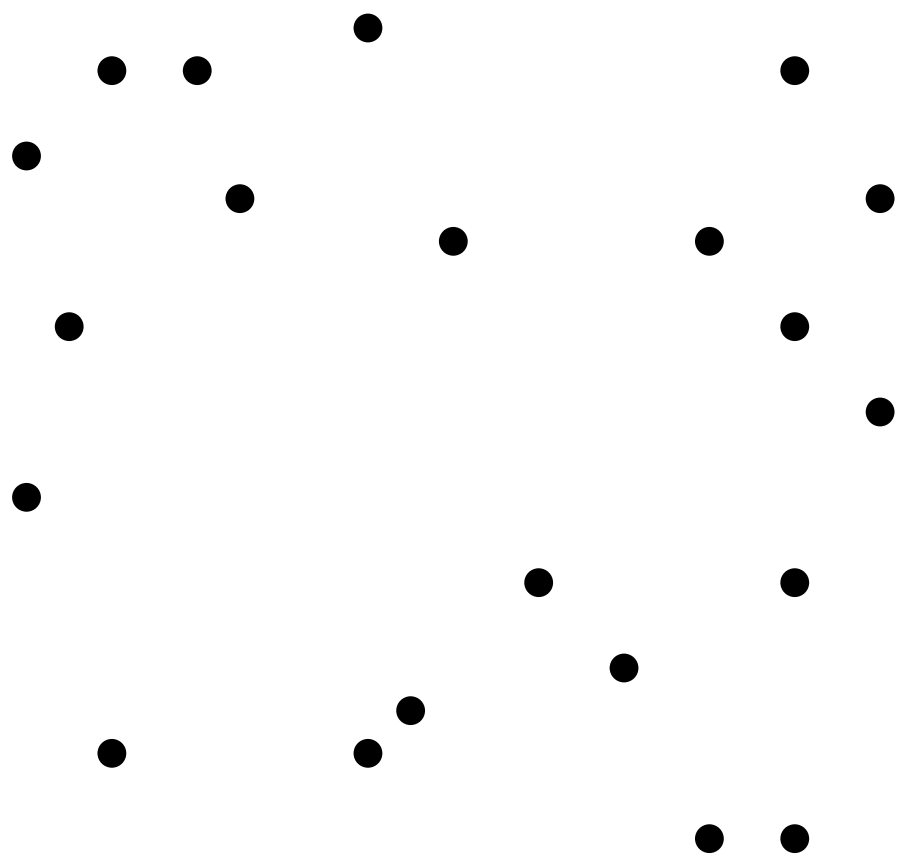
Time-Windowed Closest Pair

- How to extend this solution to find the closest pair?
- We don't know how big to make the grid cells.
- Solution: a recursive grid structure (i.e., a quadtree).



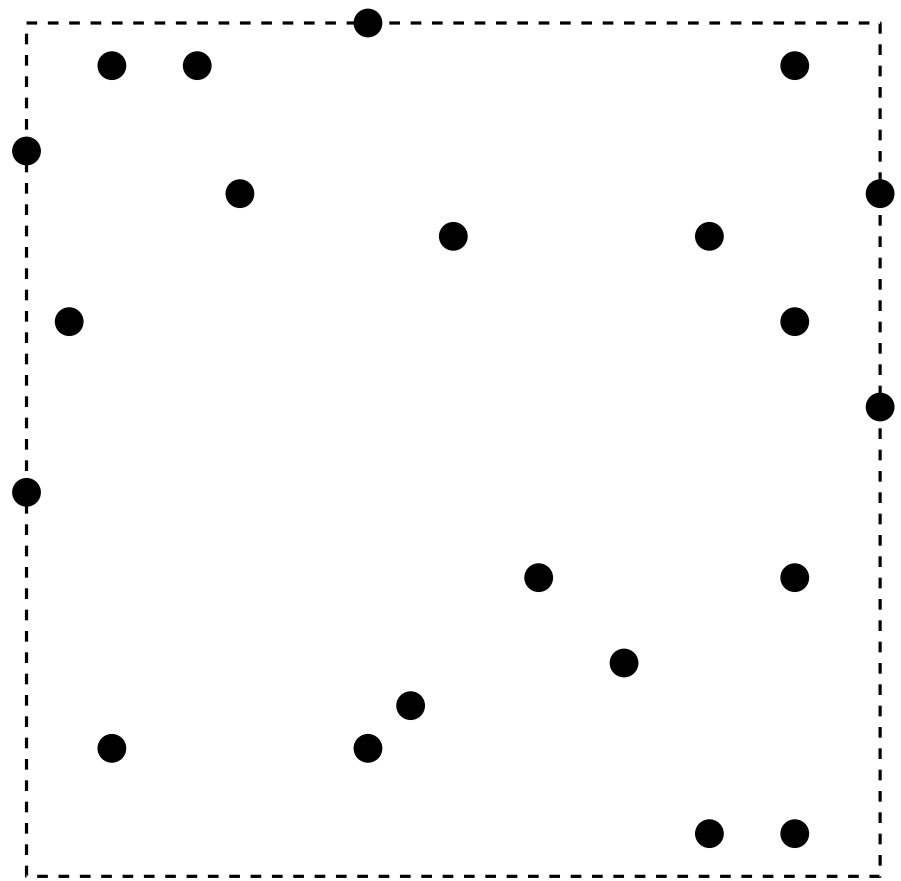
Time-Windowed Closest Pair

- How to extend this solution to find the closest pair?
- We don't know how big to make the grid cells.
- Solution: a recursive grid structure (i.e., a quadtree).



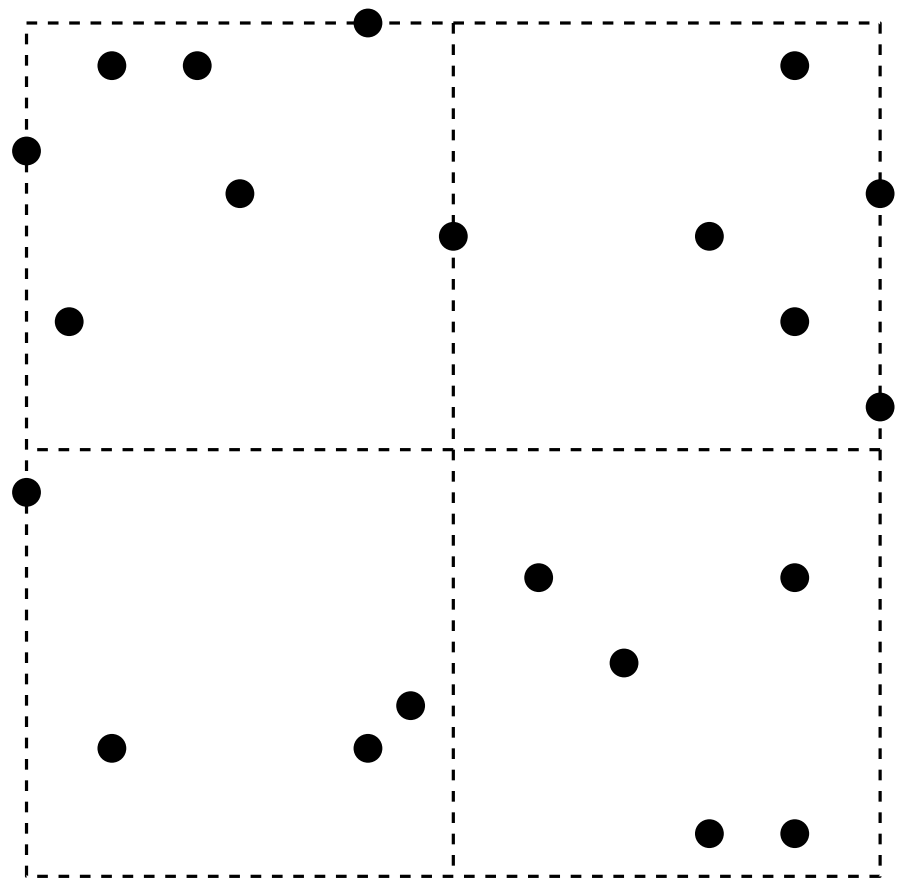
Time-Windowed Closest Pair

- How to extend this solution to find the closest pair?
- We don't know how big to make the grid cells.
- Solution: a recursive grid structure (i.e., a quadtree).



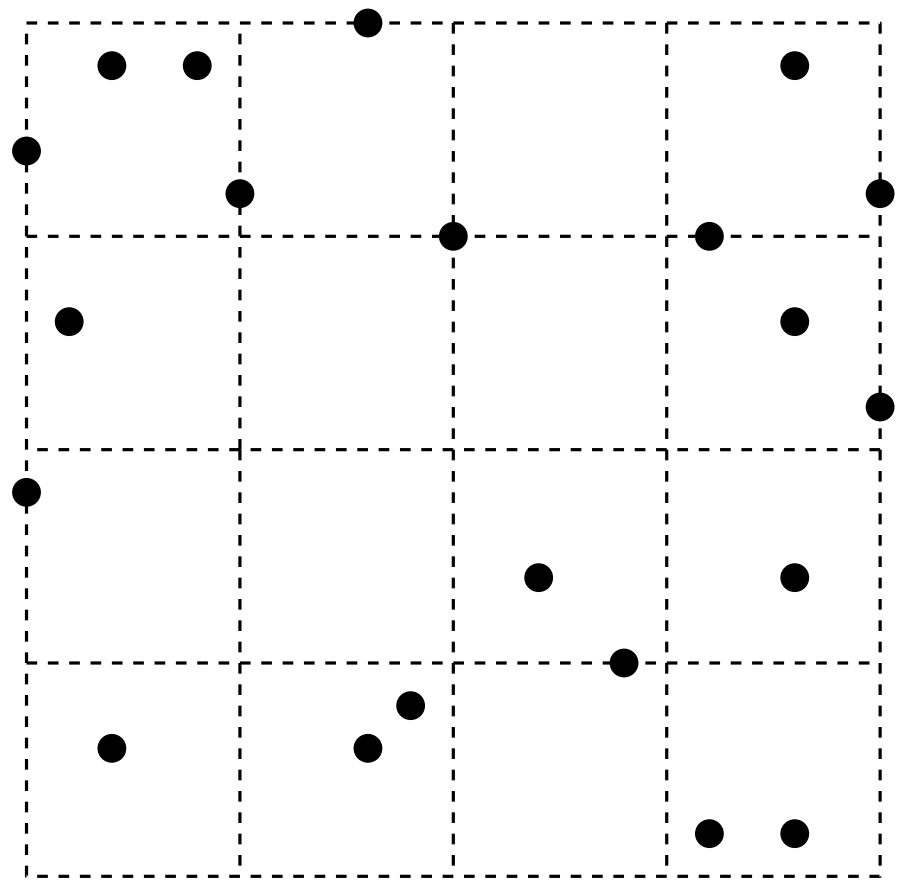
Time-Windowed Closest Pair

- How to extend this solution to find the closest pair?
- We don't know how big to make the grid cells.
- Solution: a recursive grid structure (i.e., a quadtree).



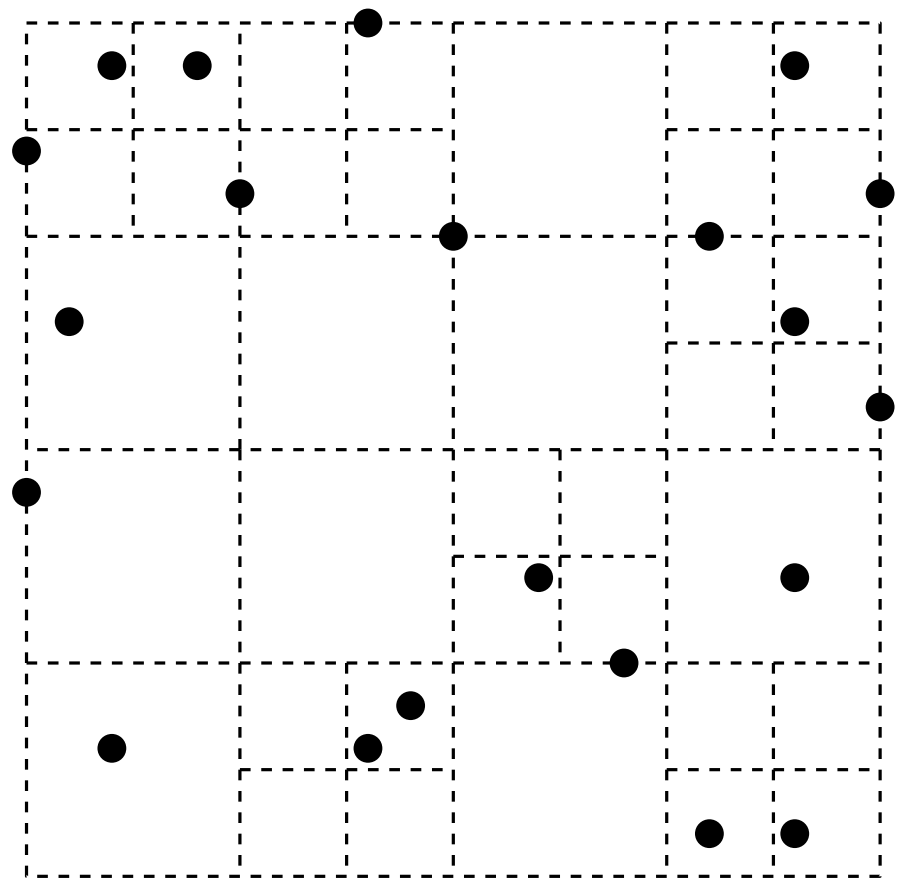
Time-Windowed Closest Pair

- How to extend this solution to find the closest pair?
- We don't know how big to make the grid cells.
- Solution: a recursive grid structure (i.e., a quadtree).



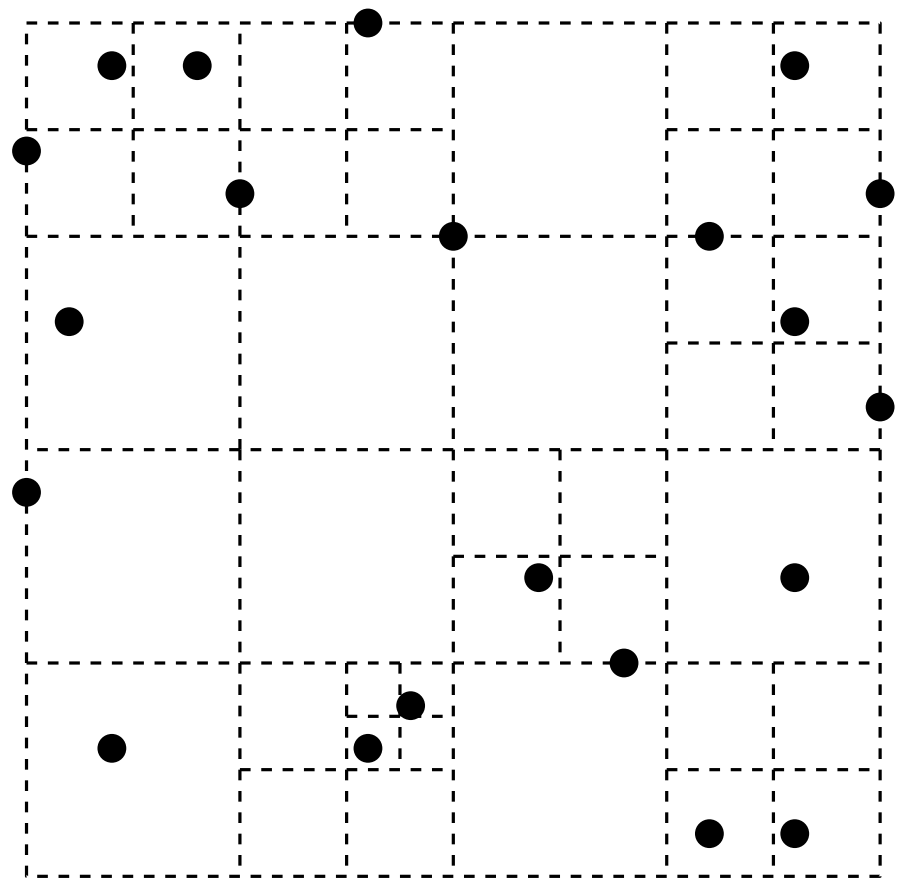
Time-Windowed Closest Pair

- How to extend this solution to find the closest pair?
- We don't know how big to make the grid cells.
- Solution: a recursive grid structure (i.e., a quadtree).



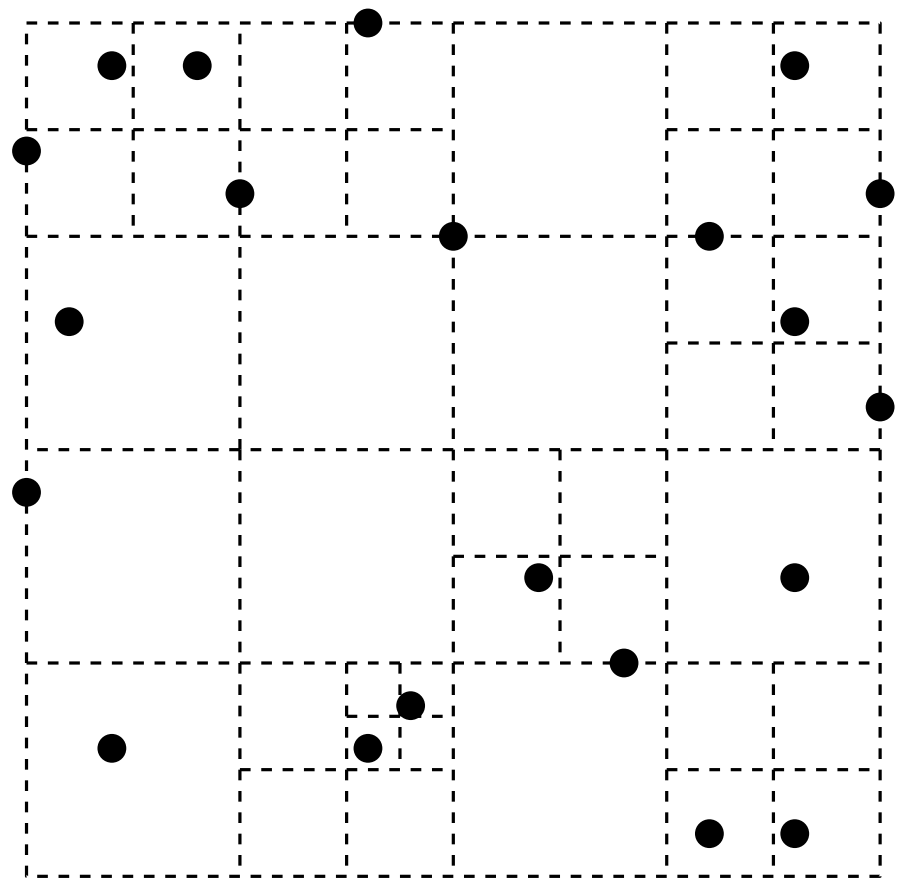
Time-Windowed Closest Pair

- How to extend this solution to find the closest pair?
- We don't know how big to make the grid cells.
- Solution: a recursive grid structure (i.e., a quadtree).



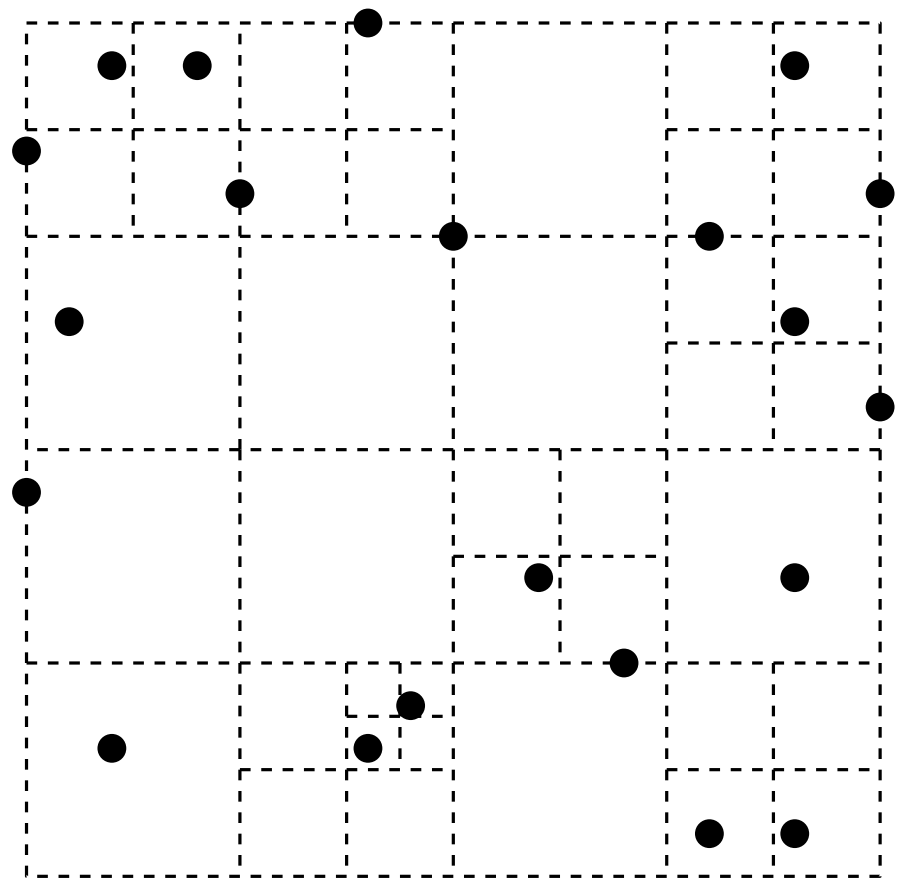
Time-Windowed Closest Pair

- How to extend this solution to find the closest pair?
- We don't know how big to make the grid cells.
- Solution: a recursive grid structure (i.e., a quadtree).
- Problem: quadtree height unbounded w.r.t. #points



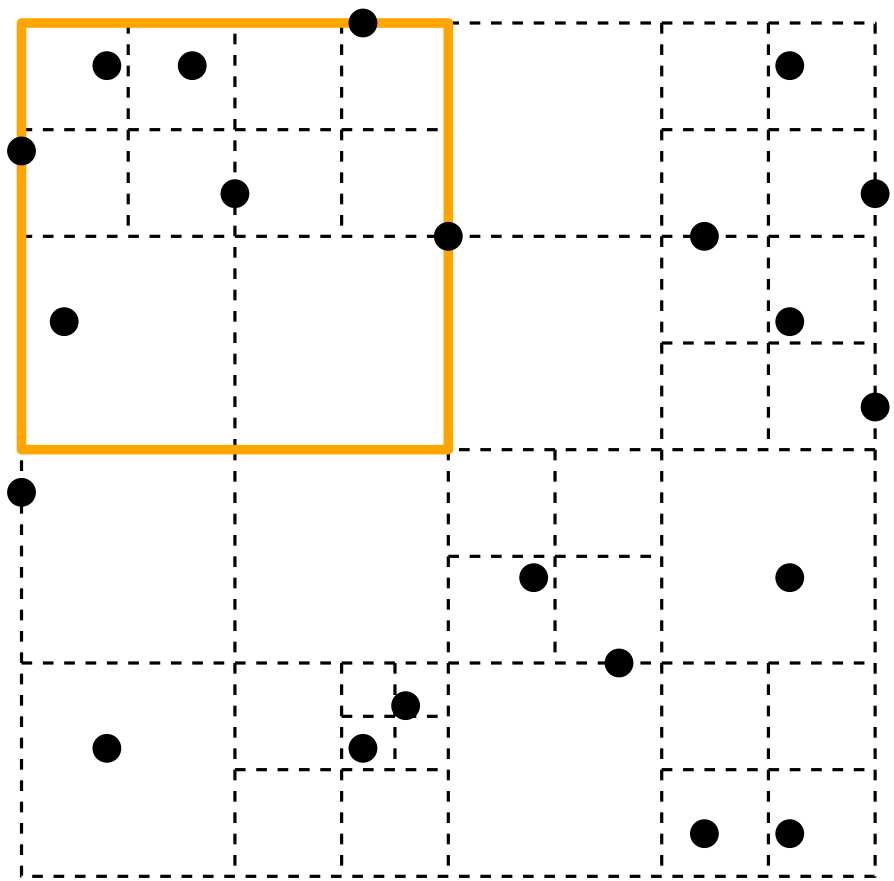
Quadtree Centroid

- For a quadtree on n points P



Quadtree Centroid

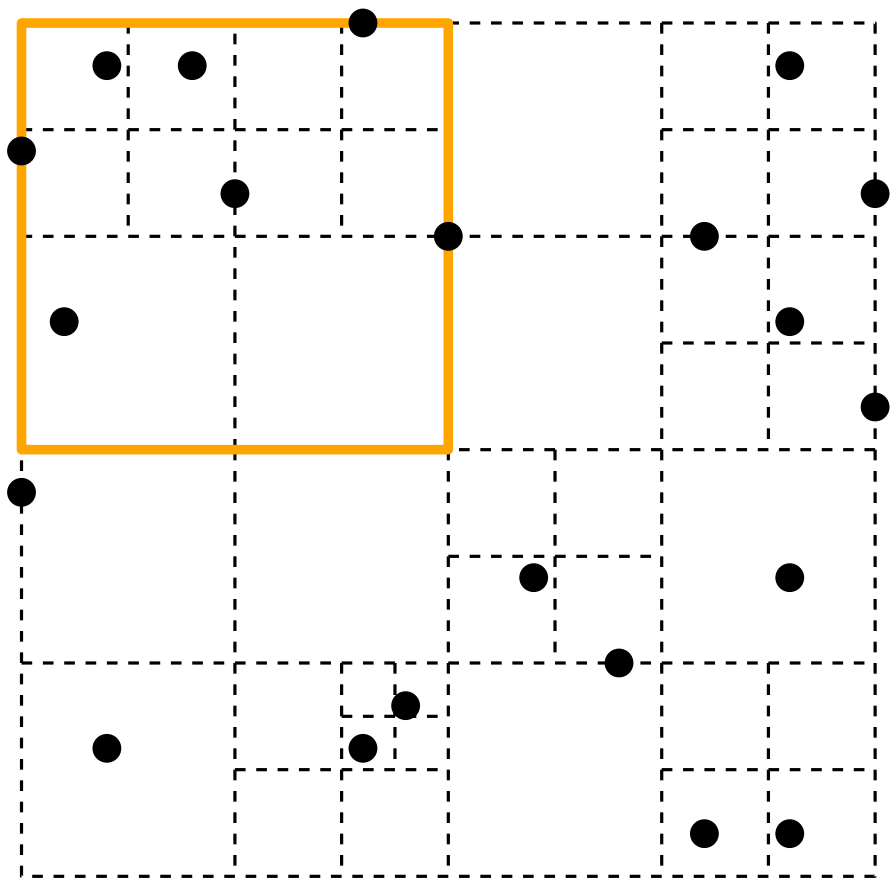
- For a quadtree on n points P
- $\exists$ cell B s.t.
  - $\diamond$ $|P \cap B| \leq \alpha n$
  - $\diamond$ $|P \setminus B| \leq \alpha n$



Quadtree Centroid

- For a quadtree on n points P
- $\exists$ cell B s.t.
  - $|P \cap B| \leq \alpha n$
  - $|P \setminus B| \leq \alpha n$

Depends only on d

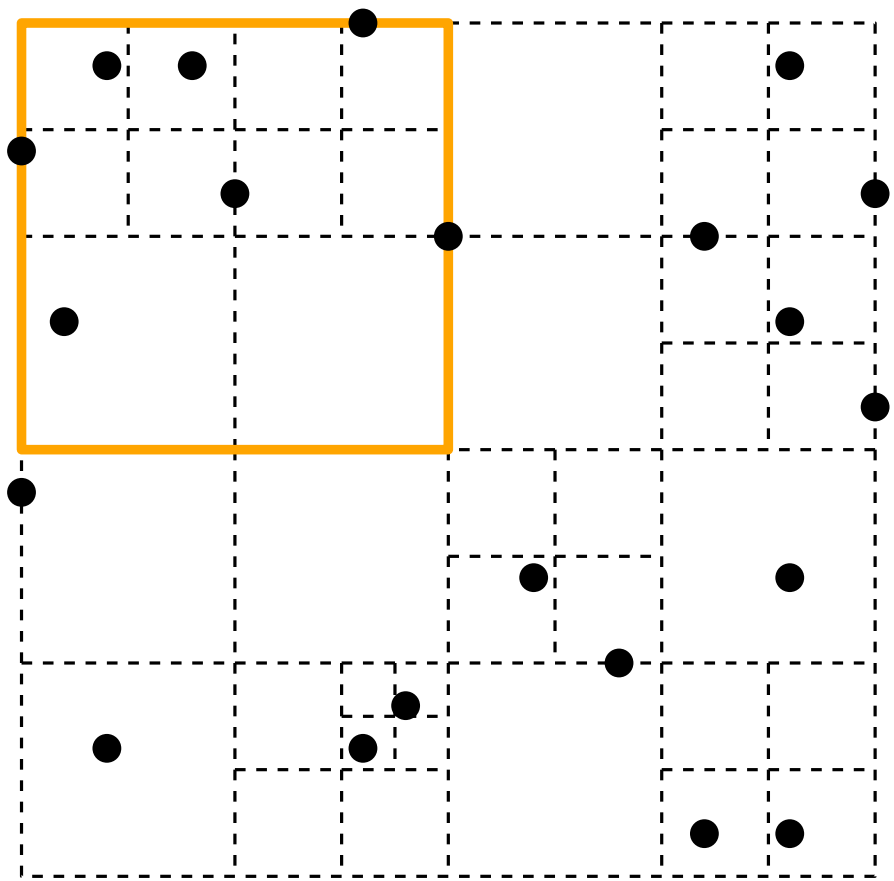


Quadtree Centroid

- For a quadtree on n points P
- $\exists$ cell B s.t.
  - $\diamond$ $|P \cap B| \leq \alpha n$
  - $\diamond$ $|P \setminus B| \leq \alpha n$

Depends only on d

We call B the *centroid*.



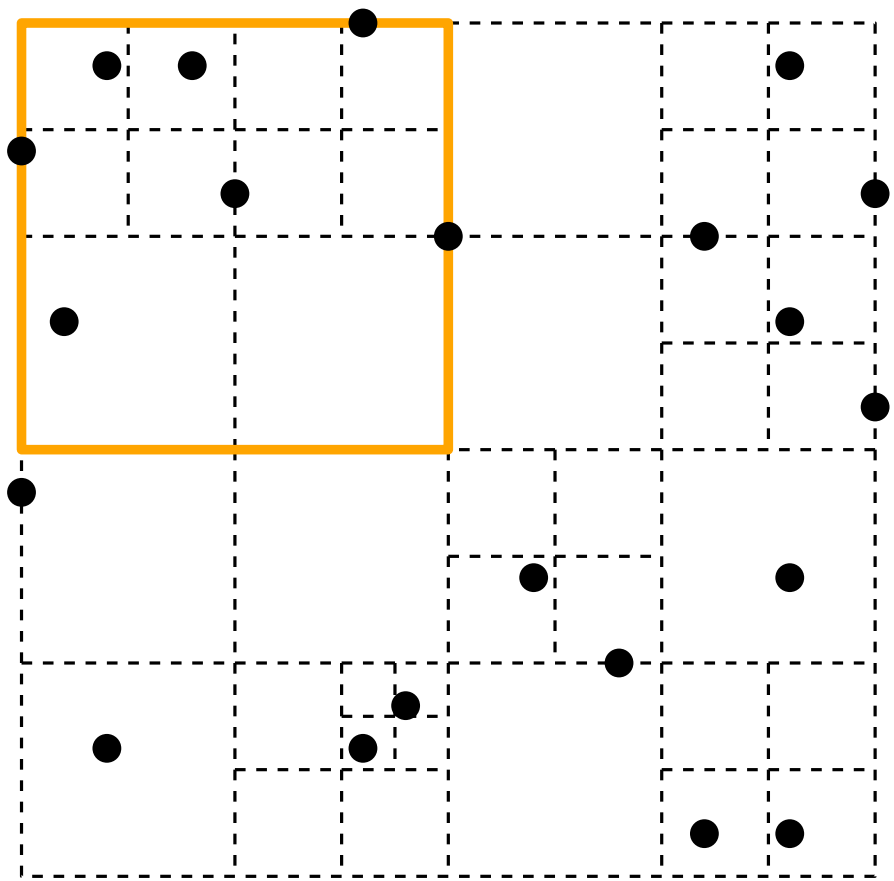
Quadtree Centroid

- For a quadtree on n points P
- $\exists$ cell B s.t.
  - $\diamond$ $|P \cap B| \leq \alpha n$
  - $\diamond$ $|P \setminus B| \leq \alpha n$

Depends only on d

We call B the *centroid*.

Recurse $O(\lg n)$ levels



Quadtree Centroid

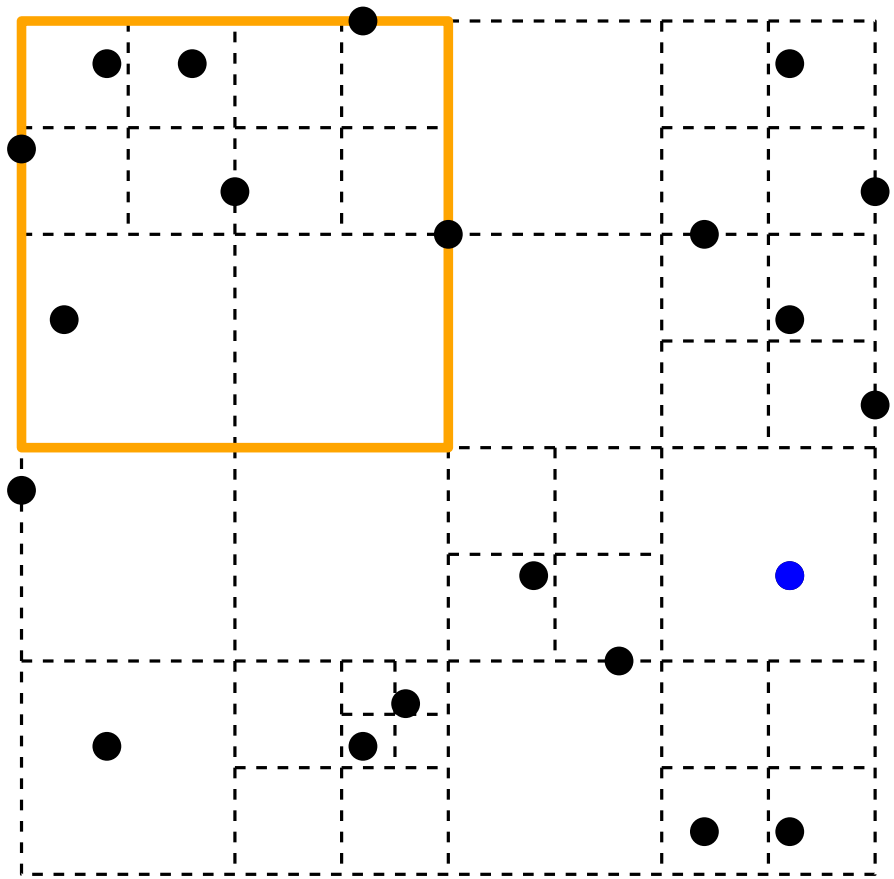
- For a quadtree on n points P
- $\exists$ cell B s.t.
  - $\diamond$ $|P \cap B| \leq \alpha n$
  - $\diamond$ $|P \setminus B| \leq \alpha n$

Depends only on d

- At each level, each $p \notin B$.

We call B the *centroid*.

Recurse $O(\lg n)$ levels



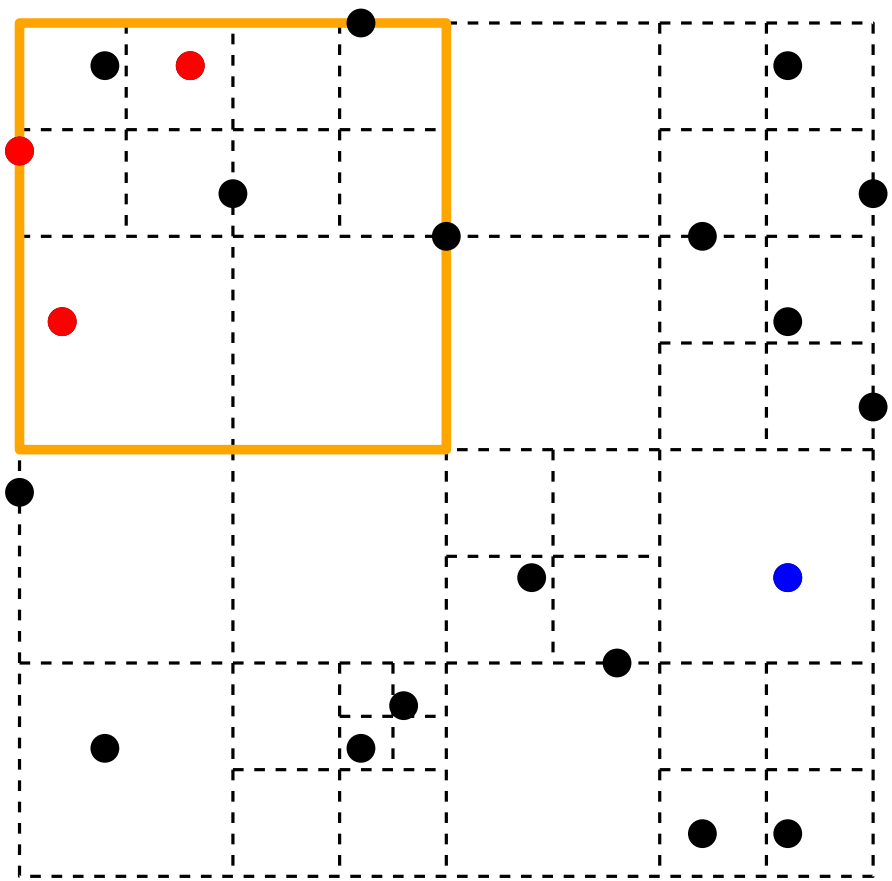
Quadtree Centroid

- For a quadtree on n points P
- $\exists$ cell B s.t.
  - $\diamond$ $|P \cap B| \leq \alpha n$
  - $\diamond$ $|P \setminus B| \leq \alpha n$

Depends only on d

We call B the *centroid*.

Recurse $O(\lg n)$ levels



- At each level, each $p \notin B$.
- Constant # of p's pred./succ. $q \in B$.

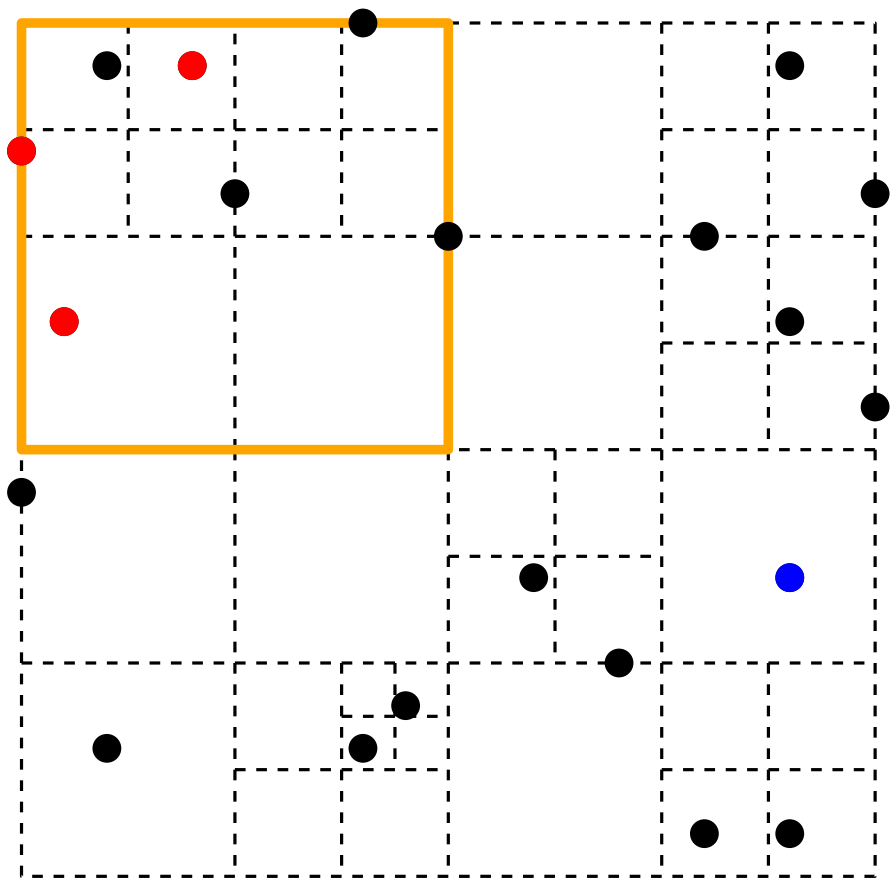
Quadtree Centroid

- For a quadtree on n points P
- $\exists$ cell B s.t.
  - $|P \cap B| \leq \alpha n$
  - $|P \setminus B| \leq \alpha n$

Depends only on d

We call B the *centroid*.

Recurse $O(\lg n)$ levels



- At each level, each $p \notin B$.
- Constant # of p's pred./succ. $q \in B$.
- $O(n \lg n)$ candidate pairs (p,q)

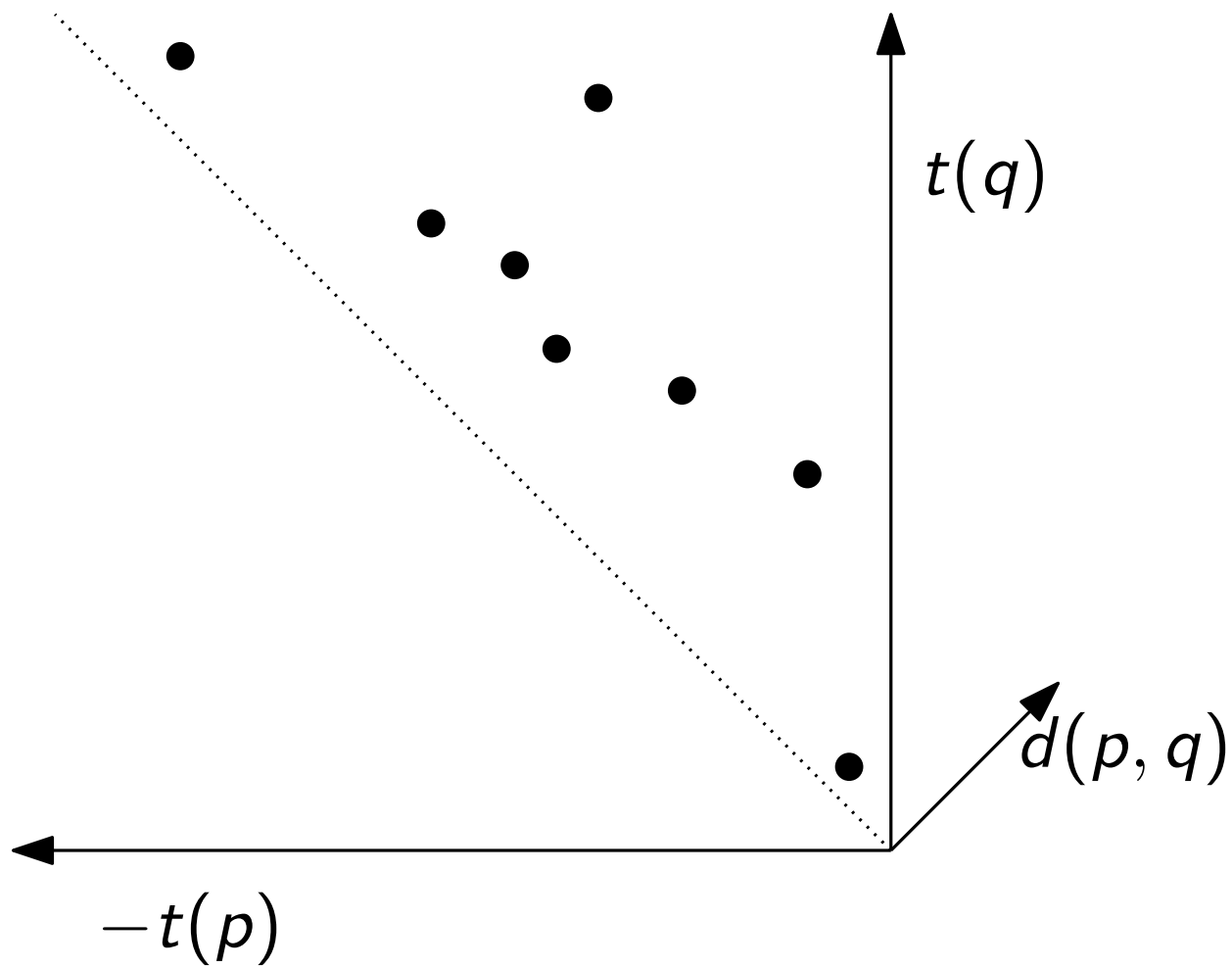
Reduction to 2D Dominance Range Minimum

Reduction to 2D Dominance Range Minimum

- Plot $(t(p), t(q), d(p, q))$ for each candidate pair p, q

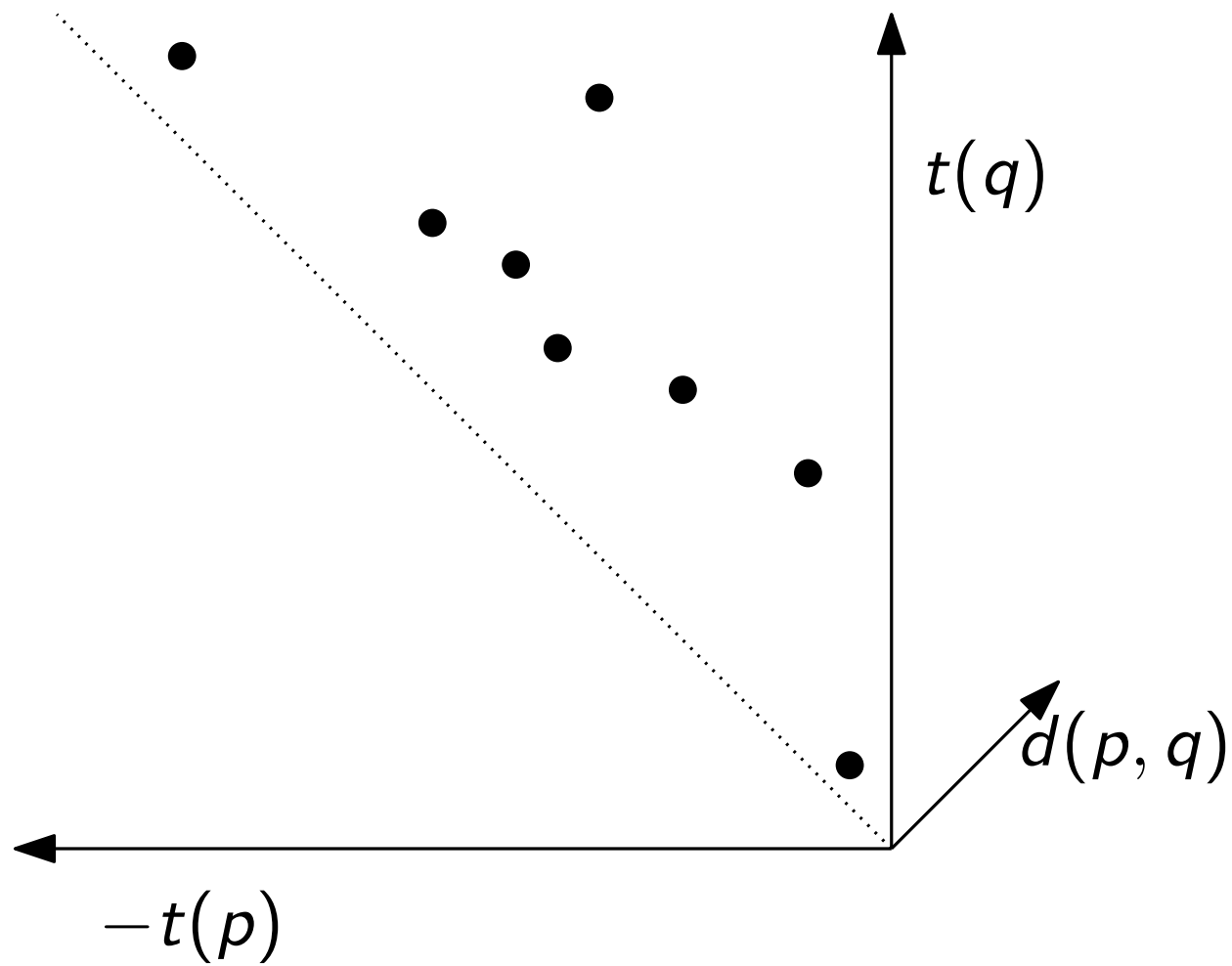
Reduction to 2D Dominance Range Minimum

- Plot $(t(p), t(q), d(p, q))$ for each candidate pair p, q



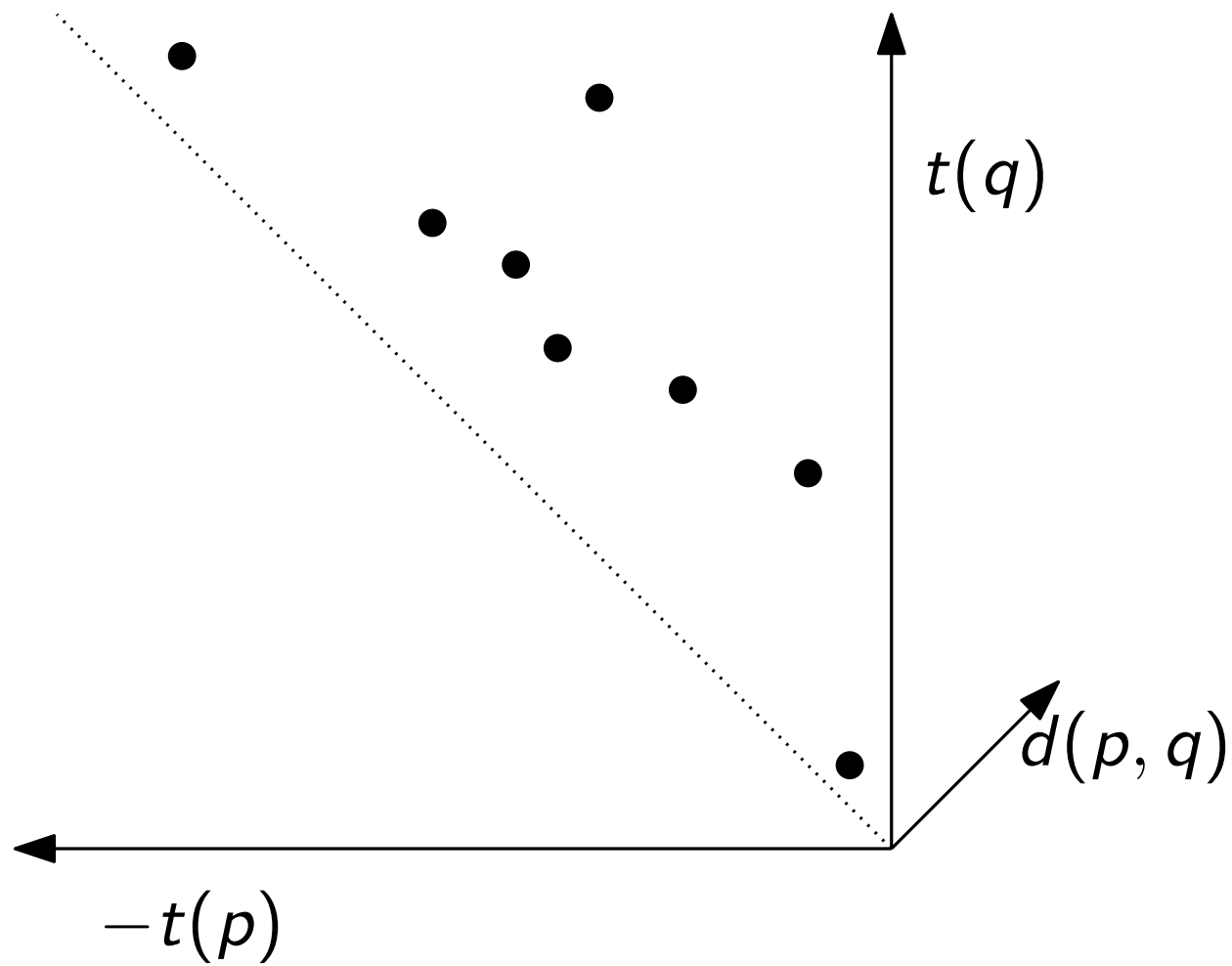
Reduction to 2D Dominance Range Minimum

- Plot $(t(p), t(q), d(p,q))$ for each candidate pair p, q
- Query: $t_1 \leq t(p) \leq t(q) \leq t_2$ with $\min(d(p,q))$



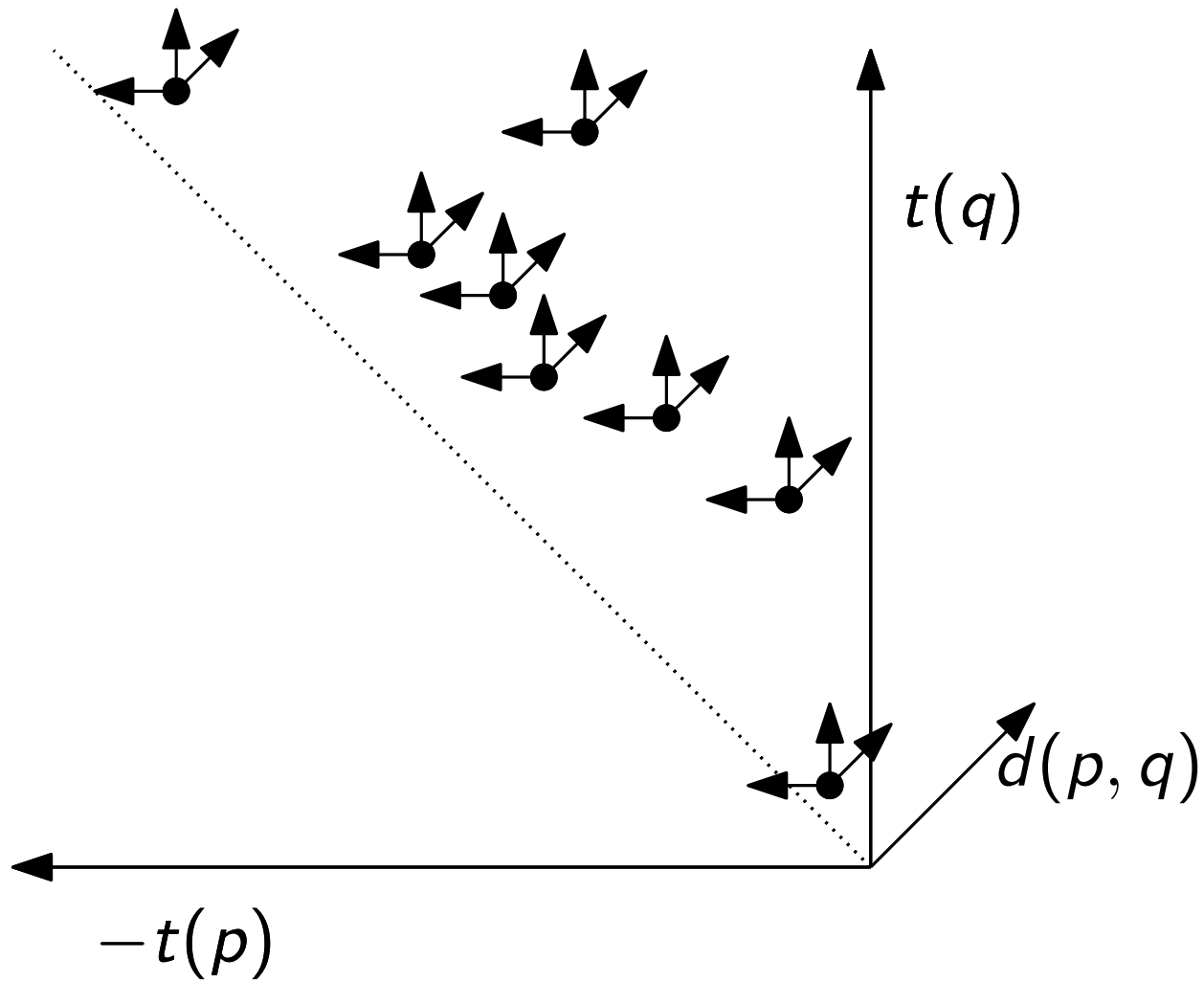
Reduction to 2D Dominance Range Minimum

- Plot $(t(p), t(q), d(p,q))$ for each candidate pair p, q
- Query: $t_1 \leq t(p) \leq t(q) \leq t_2$ with $\min(d(p,q))$
- Find dominated point w/ $\min d(p,q)$



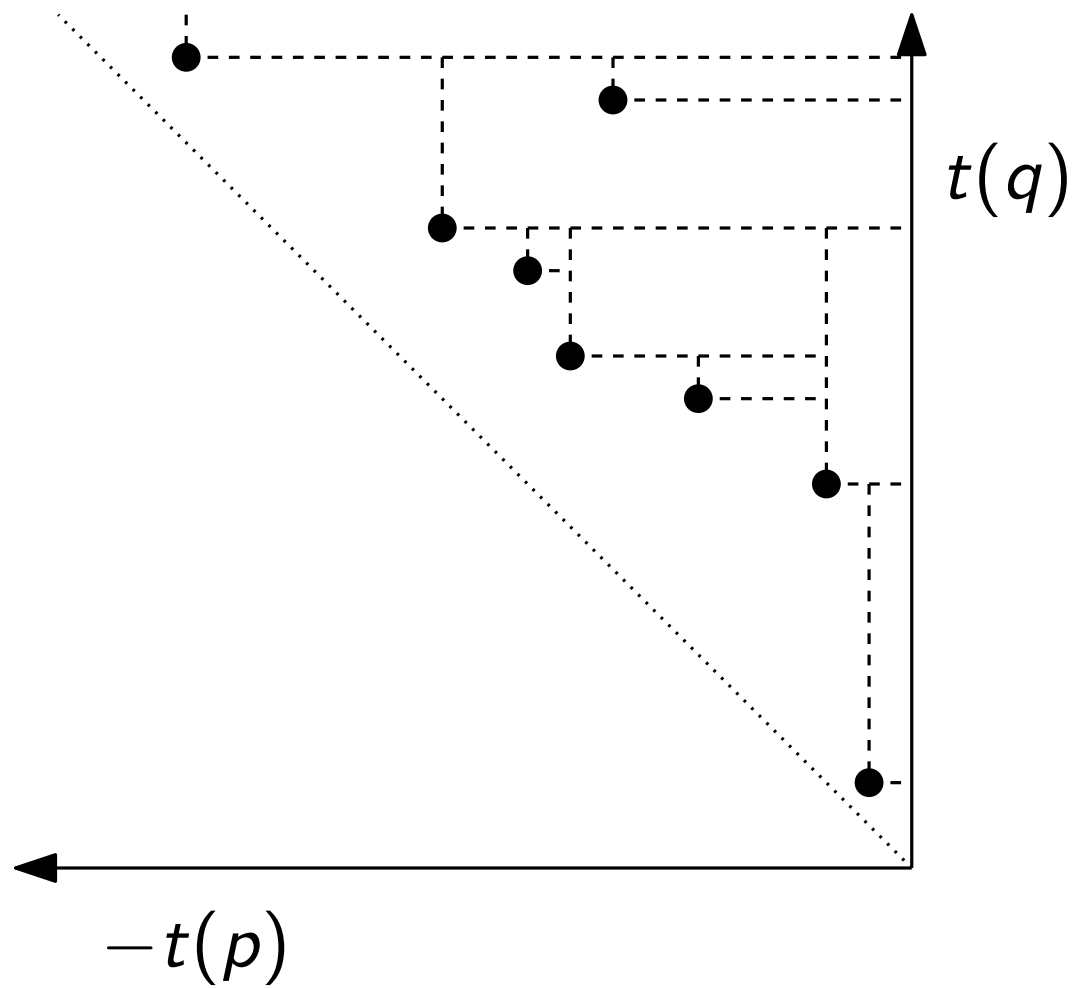
Reduction to 2D Dominance Range Minimum

- Plot $(t(p), t(q), d(p,q))$ for each candidate pair p, q
- Query: $t_1 \leq t(p) \leq t(q) \leq t_2$ with $\min(d(p,q))$
- Find dominated point w/ $\min d(p,q)$
- Staircase equiv.: points as apices of orthants



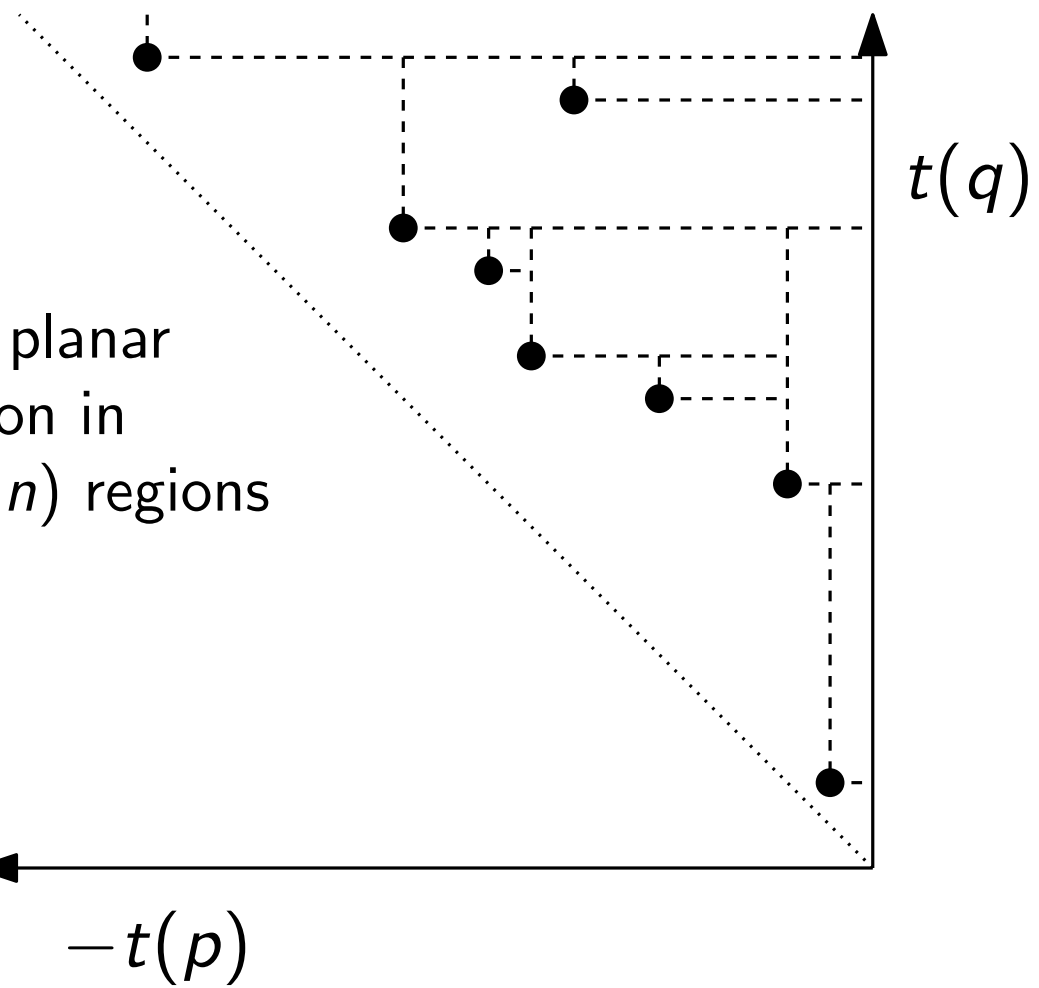
Reduction to 2D Dominance Range Minimum

- Plot $(t(p), t(q), d(p, q))$ for each candidate pair p, q
- Query: $t_1 \leq t(p) \leq t(q) \leq t_2$ with $\min(d(p, q))$
- Find dominated point w/ $\min d(p, q)$
- Staircase equiv.: points as apices of orthants
- Compute lower envelope, project onto plane



Reduction to 2D Dominance Range Minimum

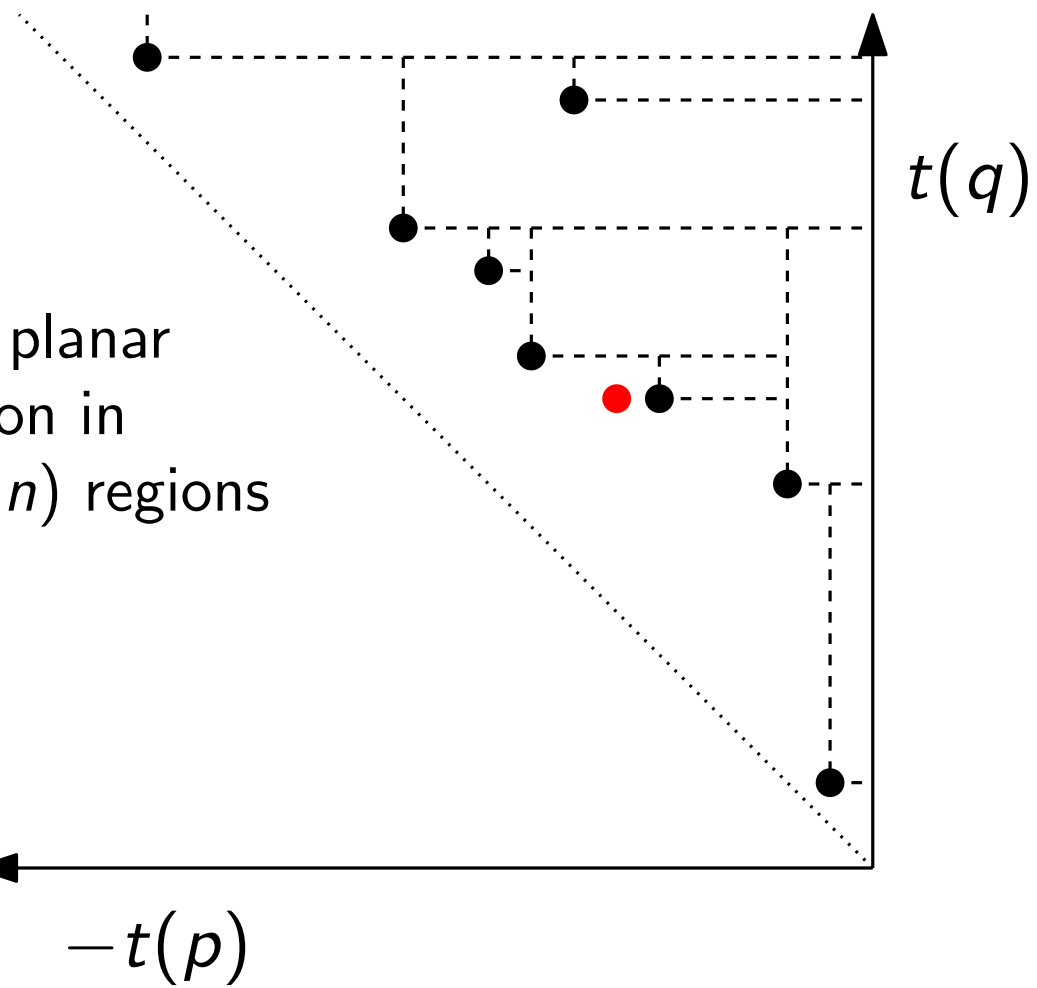
- Plot $(t(p), t(q), d(p, q))$ for each candidate pair p, q
- Query: $t_1 \leq t(p) \leq t(q) \leq t_2$ with $\min(d(p, q))$
- Find dominated point w/ $\min d(p, q)$
- Staircase equiv.: points as apices of orthants
- Compute lower envelope, project onto plane



- Orthogonal planar point location in $N = O(n \lg n)$ regions

Reduction to 2D Dominance Range Minimum

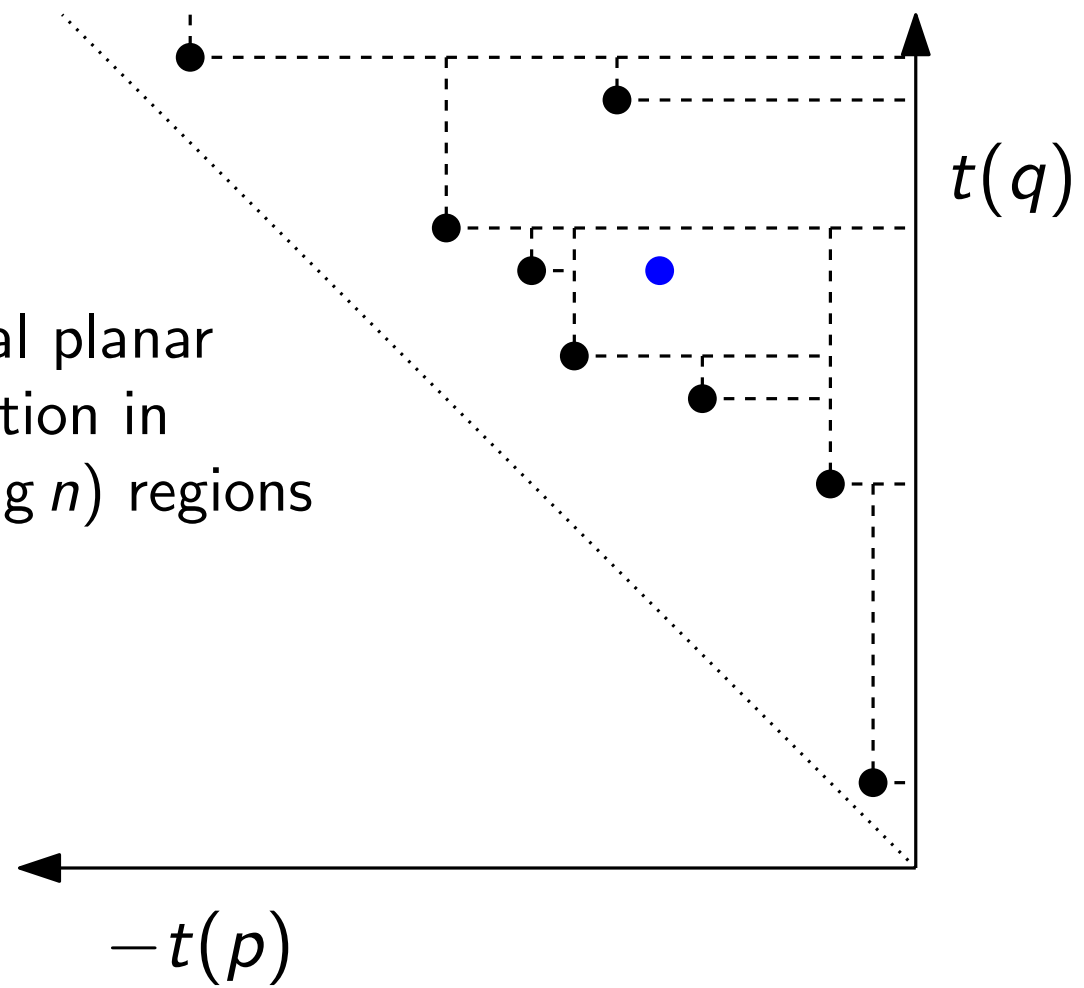
- Plot $(t(p), t(q), d(p,q))$ for each candidate pair p, q
- Query: $t_1 \leq t(p) \leq t(q) \leq t_2$ with $\min(d(p,q))$
- Find dominated point w/ $\min d(p,q)$
- Staircase equiv.: points as apices of orthants
- Compute lower envelope, project onto plane



- Orthogonal planar point location in $N = O(n \lg n)$ regions

Reduction to 2D Dominance Range Minimum

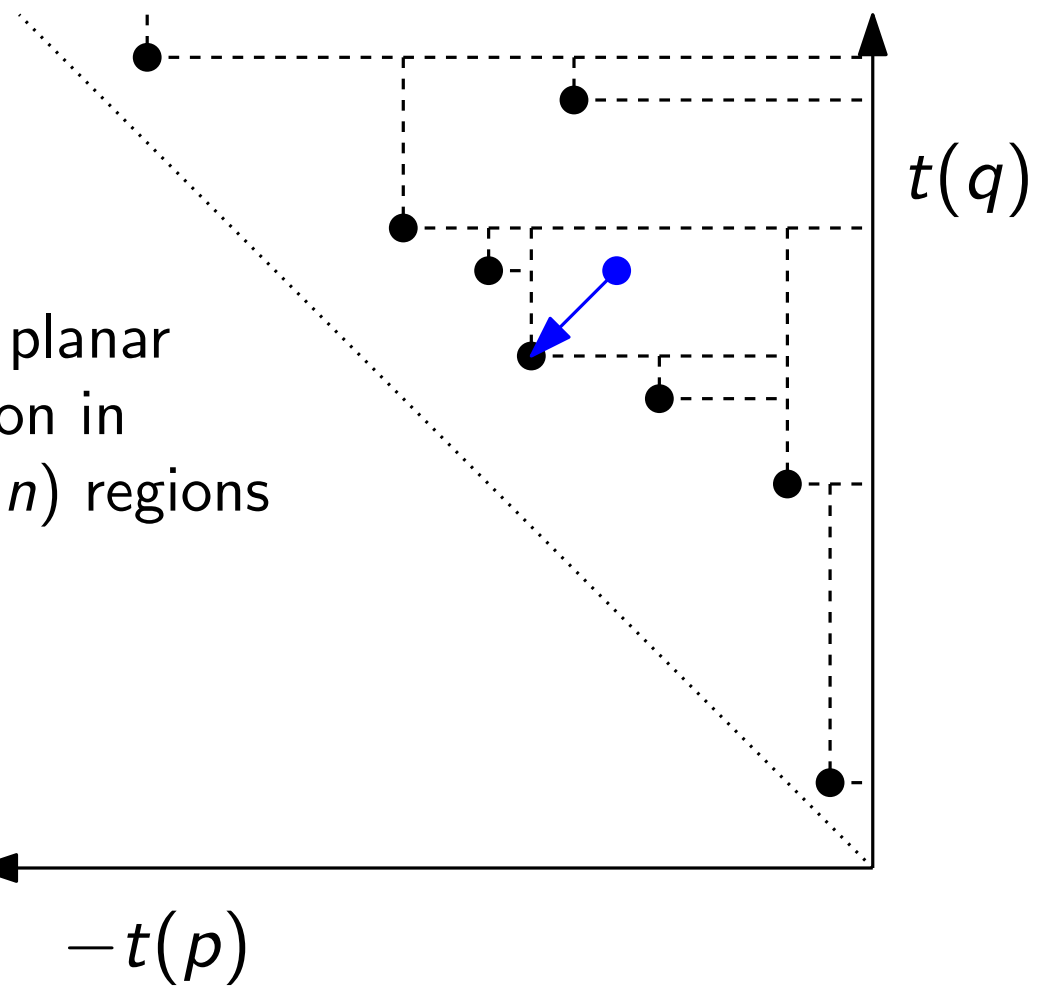
- Plot $(t(p), t(q), d(p, q))$ for each candidate pair p, q
- Query: $t_1 \leq t(p) \leq t(q) \leq t_2$ with $\min(d(p, q))$
- Find dominated point w/ $\min d(p, q)$
- Staircase equiv.: points as apices of orthants
- Compute lower envelope, project onto plane



- Orthogonal planar point location in $N = O(n \lg n)$ regions

Reduction to 2D Dominance Range Minimum

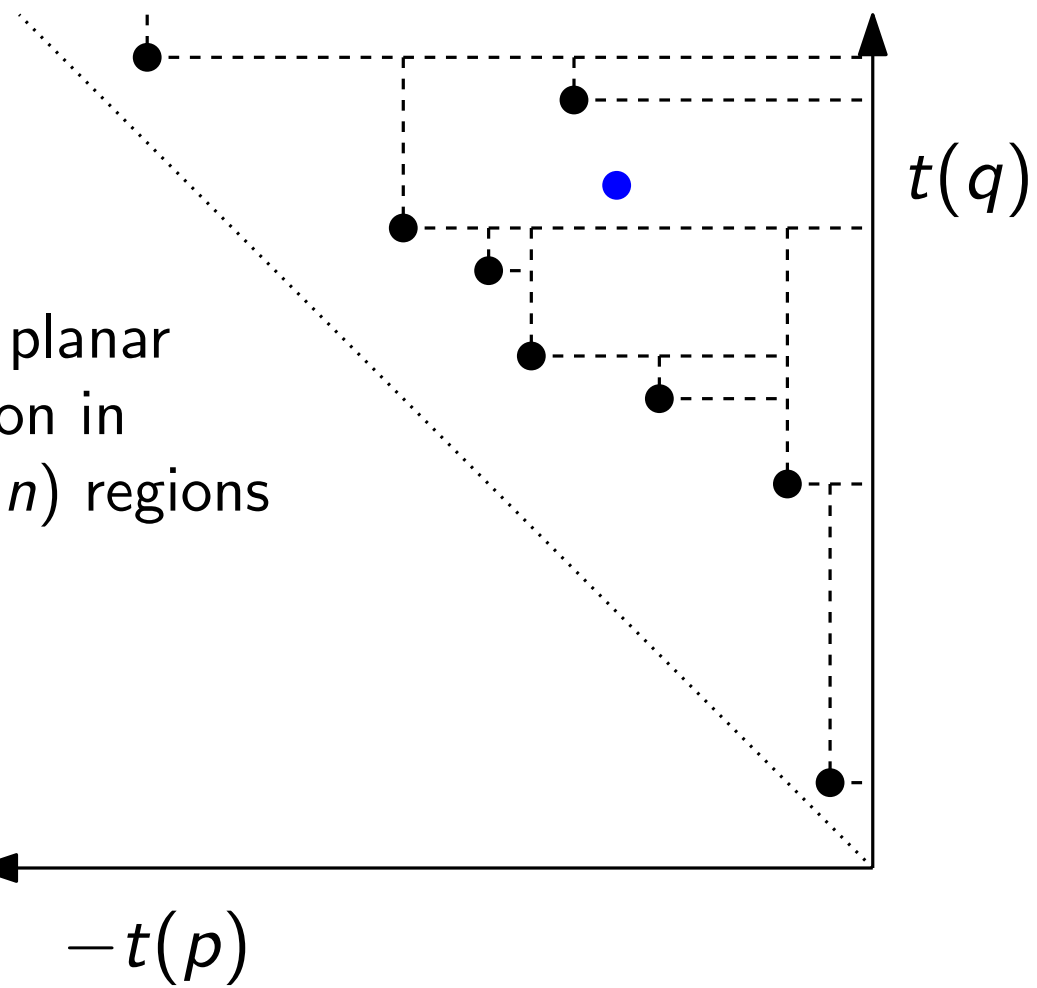
- Plot $(t(p), t(q), d(p, q))$ for each candidate pair p, q
- Query: $t_1 \leq t(p) \leq t(q) \leq t_2$ with $\min(d(p, q))$
- Find dominated point w/ $\min d(p, q)$
- Staircase equiv.: points as apices of orthants
- Compute lower envelope, project onto plane



- Orthogonal planar point location in $N = O(n \lg n)$ regions

Reduction to 2D Dominance Range Minimum

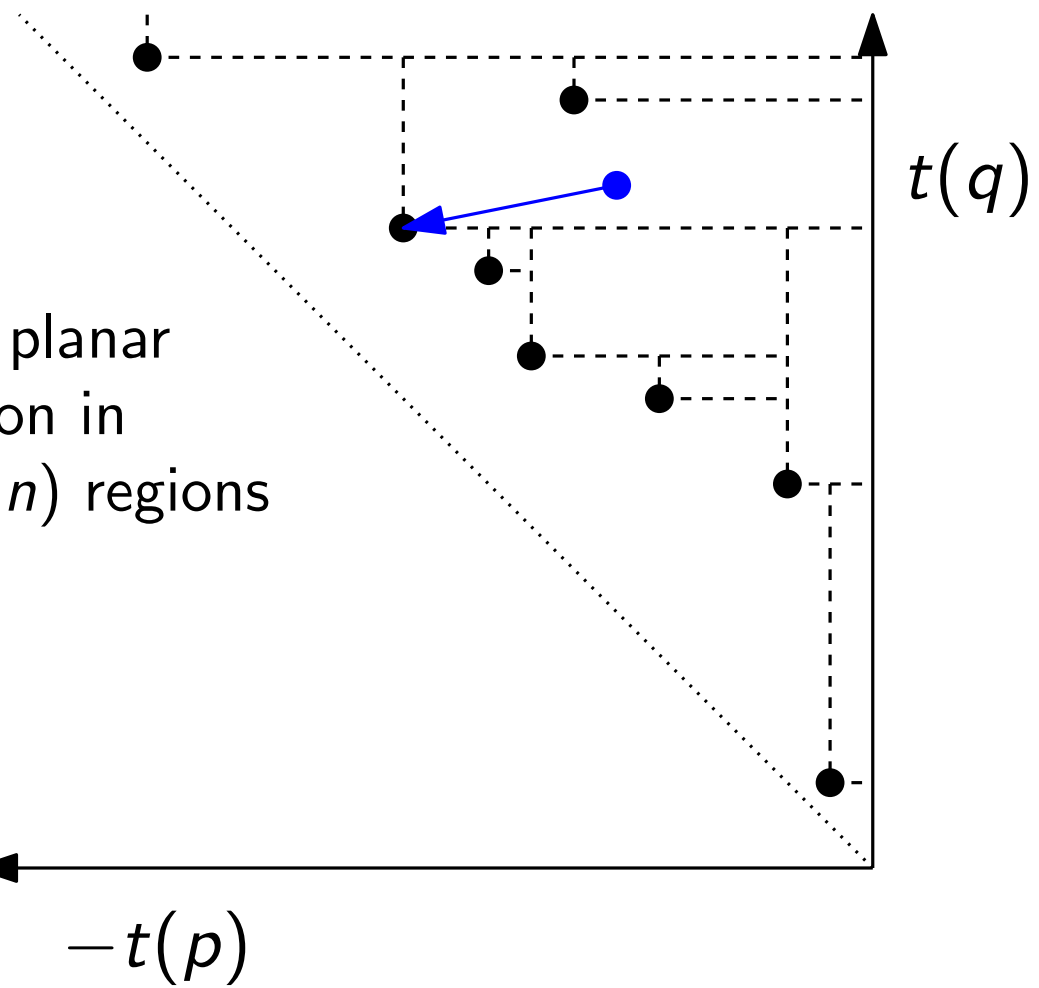
- Plot $(t(p), t(q), d(p,q))$ for each candidate pair p, q
- Query: $t_1 \leq t(p) \leq t(q) \leq t_2$ with $\min(d(p,q))$
- Find dominated point w/ $\min d(p,q)$
- Staircase equiv.: points as apices of orthants
- Compute lower envelope, project onto plane



- Orthogonal planar point location in $N = O(n \lg n)$ regions

Reduction to 2D Dominance Range Minimum

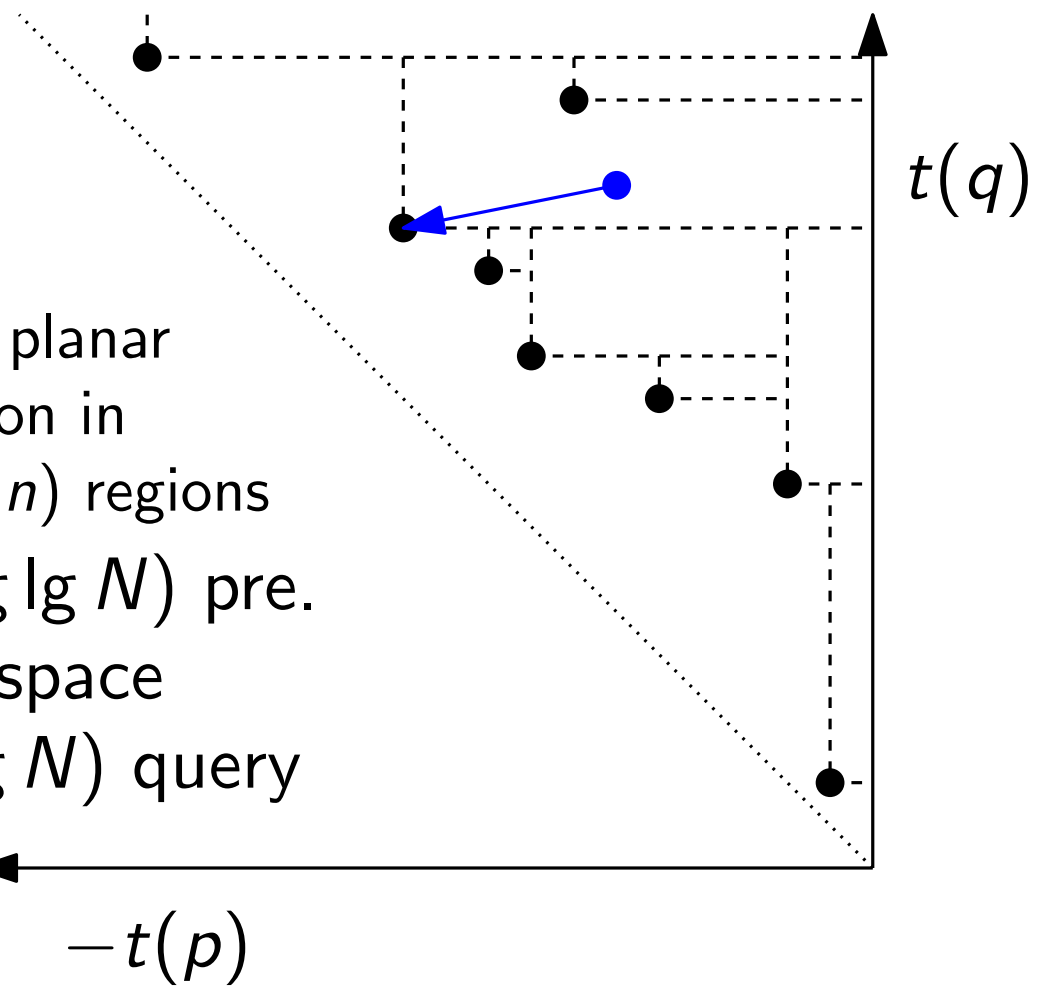
- Plot $(t(p), t(q), d(p,q))$ for each candidate pair p, q
- Query: $t_1 \leq t(p) \leq t(q) \leq t_2$ with $\min(d(p,q))$
- Find dominated point w/ $\min d(p,q)$
- Staircase equiv.: points as apices of orthants
- Compute lower envelope, project onto plane



- Orthogonal planar point location in $N = O(n \lg n)$ regions

Reduction to 2D Dominance Range Minimum

- Plot $(t(p), t(q), d(p,q))$ for each candidate pair p, q
- Query: $t_1 \leq t(p) \leq t(q) \leq t_2$ with $\min(d(p,q))$
- Find dominated point w/ $\min d(p,q)$
- Staircase equiv.: points as apices of orthants
- Compute lower envelope, project onto plane



- Orthogonal planar point location in $N = O(n \lg n)$ regions
  - $O(N \lg \lg N)$ pre.
  - $O(N)$ space
  - $O(\lg \lg N)$ query

Grids & Quadtrees: Conclusion

Grids & Quadtrees: Conclusion

Closest Pair Decision

Grids & Quadtrees: Conclusion

Closest Pair Decision

- Grids with unit side-length, shifting, succinct rank/select
- Reduction to 2D dominance emptiness

Grids & Quadtrees: Conclusion

Closest Pair Decision

- Grids with unit side-length, shifting, succinct rank/select
- Reduction to 2D dominance emptiness
- $O(n)$ expected preprocessing time
- $O(n)$ bits of space
- $O(1)$ query time

Grids & Quadtrees: Conclusion

Closest Pair Decision

- Grids with unit side-length, shifting, succinct rank/select
- Reduction to 2D dominance emptiness
- $O(n)$ expected preprocessing time
- $O(n)$ bits of space
- $O(1)$ query time

Closest Pair

Grids & Quadtrees: Conclusion

Closest Pair Decision

- Grids with unit side-length, shifting, succinct rank/select
- Reduction to 2D dominance emptiness
- $O(n)$ expected preprocessing time
- $O(n)$ bits of space
- $O(1)$ query time

Closest Pair

- Quadtrees + centroid
- Reduction to 2D dominance range min.

Grids & Quadtrees: Conclusion

Closest Pair Decision

- Grids with unit side-length, shifting, succinct rank/select
- Reduction to 2D dominance emptiness
- $O(n)$ expected preprocessing time
- $O(n)$ bits of space
- $O(1)$ query time

Closest Pair

- Quadtrees + centroid
- Reduction to 2D dominance range min.
- $O(n \log n \log \log n)$ preprocessing time
- $O(n \log n)$ space
- $O(\log \log n)$ query time

New Results

Approach #1: Grids & Quadtrees

Problem	Preprocessing	Space	Query
CP Decision	$O(n)$ exp.	$O(n)$ bits	$O(1)$
Closest Pair	$O(n \lg n \lg \lg n)$	$O(n \lg n)$	$O(\lg \lg n)$

Approach #2: Generalized Range Successor

Problem	Preprocessing Time
2D diameter decision	$O(n \lg n)$
3D diameter decision	$O(n \lg^2 n)$
2D orth. seg. intersection detection	$O(n \lg n \lg \lg n)$

Approach #3: FIFO Update Sequence

Problem	Preprocessing Time
2D convex hull area decision	$O(n\alpha(n) \lg n)$
2D width decision	$O(n \lg^7 n)$

New Results

Approach #1: Grids & Quadtrees

Problem	Preprocessing	Space	Query
CP Decision	$O(n)$ exp.	$O(n)$ bits	$O(1)$
Closest Pair	$O(n \lg n \lg \lg n)$	$O(n \lg n)$	$O(\lg \lg n)$

Approach #2: Generalized Range Successor

Problem	Preprocessing Time
2D diameter decision	$O(n \lg n)$
3D diameter decision	$O(n \lg^2 n)$
2D orth. seg. intersection detection	$O(n \lg n \lg \lg n)$

Approach #3: FIFO Update Sequence

Problem	Preprocessing Time
2D convex hull area decision	$O(n\alpha(n) \lg n)$
2D width decision	$O(n \lg^7 n)$

New Results

BUT FIRST!

Approach #1: Grids & Quadtrees

Problem	Preprocessing	Space	Query
CP Decision	$O(n)$ exp.	$O(n)$ bits	$O(1)$
Closest Pair	$O(n \lg n \lg \lg n)$	$O(n \lg n)$	$O(\lg \lg n)$

Approach #2: Generalized Range Successor

Problem	Preprocessing Time
2D diameter decision	$O(n \lg n)$
3D diameter decision	$O(n \lg^2 n)$
2D orth. seg. intersection detection	$O(n \lg n \lg \lg n)$

Approach #3: FIFO Update Sequence

Problem	Preprocessing Time
2D convex hull area decision	$O(n\alpha(n) \lg n)$
2D width decision	$O(n \lg^7 n)$

Time-Windowed Decision Problems

Time-Windowed Decision Problems

From here on,
only time-windowed decision problems!

Time-Windowed Decision Problems

From here on,
only time-windowed decision problems!
. . . with *hereditary* properties!

Hereditary Properties

If a set S has a *hereditary* property $\mathcal{P}$, then any superset $S' \supseteq S$ has $\mathcal{P}$.

Hereditary Properties

If a set S has a *hereditary* property $\mathcal{P}$, then any superset $S' \supseteq S$ has $\mathcal{P}$.

Examples:

Problem	Property
diameter decision	diameter > 1
convex hull area decision	area of convex hull > 1
orthogonal segment inters. detect.	$\exists$ intersecting segments
width decision	width > 1

Hereditary Properties

If a set S has a *hereditary* property $\mathcal{P}$, then any superset $S' \supseteq S$ has $\mathcal{P}$.

Examples:

Problem	Property
diameter decision	diameter > 1
convex hull area decision	area of convex hull > 1
orthogonal segment inters. detect.	$\exists$ intersecting segments
width decision	width > 1

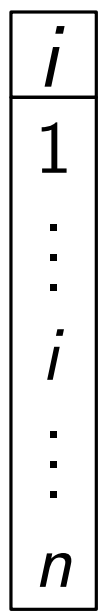
Intuition:

Adding more points won't remove these properties.

Hereditary Properties

If a set S has a *hereditary* property $\mathcal{P}$, then any superset $S' \supseteq S$ has $\mathcal{P}$.

For each time i:



Hereditary Properties

If a set S has a *hereditary* property $\mathcal{P}$, then any superset $S' \supseteq S$ has $\mathcal{P}$.

For each time i: store min time j s.t. $[i, j]$ has $\mathcal{P}$

i	j
1	j_1
$\vdots$	$\vdots$
i	j_i
$\vdots$	$\vdots$
n	j_n

Hereditary Properties

If a set S has a *hereditary* property $\mathcal{P}$, then any superset $S' \supseteq S$ has $\mathcal{P}$.

For each time i: store min time j s.t. $[i, j]$ has $\mathcal{P}$

Query: $[i, k]$

i	j
1	j_1
$\vdots$	$\vdots$
i	j_i
$\vdots$	$\vdots$
n	j_n

Hereditary Properties

If a set S has a *hereditary* property $\mathcal{P}$, then any superset $S' \supseteq S$ has $\mathcal{P}$.

For each time i: store min time j s.t. $[i, j]$ has $\mathcal{P}$

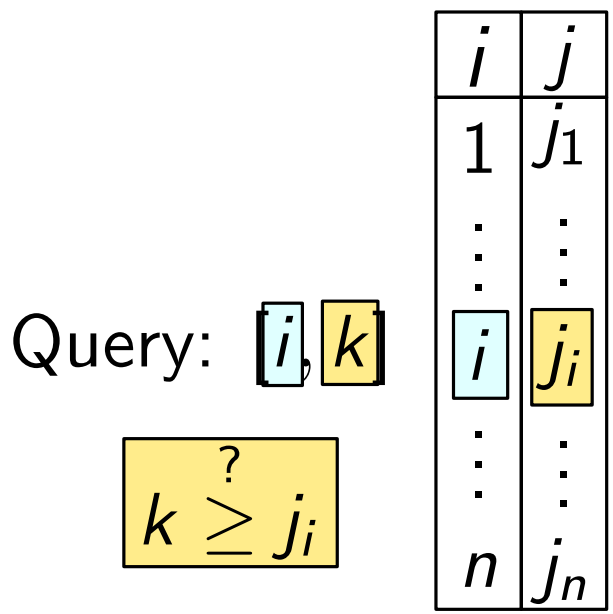
Query: $[i, k]$

i	j
1	j_1
$\vdots$	$\vdots$
i	j_i
$\vdots$	$\vdots$
n	j_n

Hereditary Properties

If a set S has a *hereditary* property $\mathcal{P}$, then any superset $S' \supseteq S$ has $\mathcal{P}$.

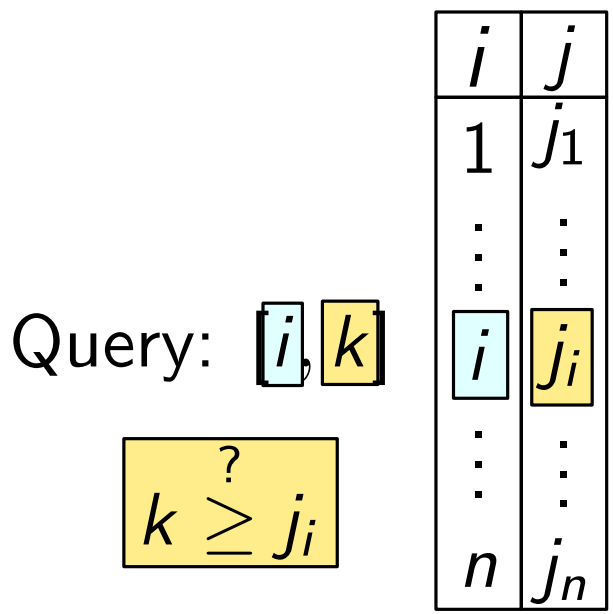
For each time i: store min time j s.t. $[i, j]$ has $\mathcal{P}$



Hereditary Properties

If a set S has a *hereditary* property $\mathcal{P}$, then any superset $S' \supseteq S$ has $\mathcal{P}$.

For each time i: store min time j s.t. $[i, j]$ has $\mathcal{P}$



Monotone chain

Hereditary Properties

If a set S has a *hereditary* property $\mathcal{P}$, then any superset $S' \supseteq S$ has $\mathcal{P}$.

For each time i: store min time j s.t. $[i, j]$ has $\mathcal{P}$

Query: $[i, k]$

$k \overset{?}{\geq} j_i$

i	j
1	j_1
$\vdots$	$\vdots$
i	j_i
$\vdots$	$\vdots$
n	j_n

Monotone chain
$O(n)$ bits
$O(1)$ query time
(From prev. approach)

Hereditary Properties

If a set S has a *hereditary* property $\mathcal{P}$, then any superset $S' \supseteq S$ has $\mathcal{P}$.

For each time i: store min time j s.t. $[i, j]$ has $\mathcal{P}$

Query: $[i, k]$

$k \overset{?}{\geq} j_i$

i	j
1	j_1
$\vdots$	$\vdots$
i	j_i
$\vdots$	$\vdots$
n	j_n

Monotone chain
$O(n)$ bits
$O(1)$ query time
(From prev. approach)

Henceforth, only concerned with preprocessing.

More Related Work

Bokal, Cabello, Eppstein; *SoCG 2015*:

- Time-windowed decision problems w/ hereditary properties
  - ◇ 2D diameter
    - · Preprocessing: $O(n \log^2 n)$
  - ◇ 2D convex hull area
    - · Preprocessing: $O(n \log n \log \log n)$

More Related Work

Bokal, Cabello, Eppstein; *SoCG 2015*:

- Time-windowed decision problems w/ hereditary properties
  - ◇ 2D diameter
    - · Preprocessing: $O(n \log^2 n)$
  - ◇ 2D convex hull area
    - · Preprocessing: $O(n \log n \log \log n)$

New Results

Approach #1: Grids & Quadtrees

Problem	Preprocessing	Space	Query
CP Decision	$O(n)$ exp.	$O(n)$ bits	$O(1)$
Closest Pair	$O(n \lg n \lg\lg n)$	$O(n \lg n)$	$O(\lg\lg n)$

Approach #2: Generalized Range Successor

Problem	Preprocessing Time
2D diameter decision	$O(n \lg^2 n)$
$\rightarrow$ improved to	$O(n \lg n)$
3D diameter decision	$O(n \lg^2 n)$
2D orth. seg. intersection detection	$O(n \lg n \lg\lg n)$

Approach #3: FIFO Update Sequence

Problem	Preprocessing Time
2D convex hull area decision	$O(n \lg n \lg\lg n)$
$\rightarrow$ improved to	$O(n\alpha(n) \lg n)$
2D width decision	$O(n \lg^7 n)$

New Results

Approach #1: Grids & Quadtrees

Problem	Preprocessing	Space	Query
CP Decision	$O(n)$ exp.	$O(n)$ bits	$O(1)$
Closest Pair	$O(n \lg n \lg\lg n)$	$O(n \lg n)$	$O(\lg\lg n)$

Approach #2: Generalized Range Successor

Problem	Preprocessing Time
2D diameter decision	$O(n \lg^2 n)$
$\rightarrow$ improved to	$O(n \lg n)$
3D diameter decision	$O(n \lg^2 n)$
2D orth. seg. intersection detection	$O(n \lg n \lg\lg n)$

Approach #3: FIFO Update Sequence

Problem	Preprocessing Time
2D convex hull area decision	$O(n \lg n \lg\lg n)$
$\rightarrow$ improved to	$O(n\alpha(n) \lg n)$
2D width decision	$O(n \lg^7 n)$

Approach #2: Generalized Range Successor

The *generalized range successor problem* is to preprocess the input to find the *successor* of each object among that object's range.

Approach #2: Generalized Range Successor

The *generalized range successor problem* is to preprocess the input to find the *successor* of each object among that object's range.

i.e. the next object in time order

Approach #2: Generalized Range Successor

The *generalized range successor problem* is to preprocess the input to find the *successor* of each object among that object's range.

i.e. the next object in time order

dependent on the specific problem

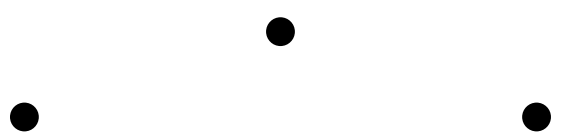
Approach #2: Generalized Range Successor

The *generalized range successor problem* is to preprocess the input to find the *successor* of each object among that object's range.

i.e. the next object in time order

dependent on the specific problem

Example problem: is there a pair of points p, q s.t. $d(p, q) > 1$?



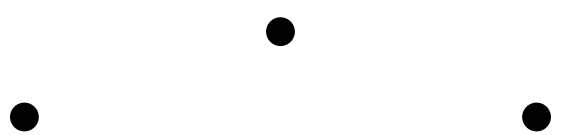
Approach #2: Generalized Range Successor

The *generalized range successor problem* is to preprocess the input to find the *successor* of each object among that object's range.

i.e. the next object in time order

dependent on the specific problem

Example problem: is there a pair of points p, q s.t. $d(p, q) > 1$?



Range of a point p: the complement of a unit disk centered at p

Approach #2: Generalized Range Successor

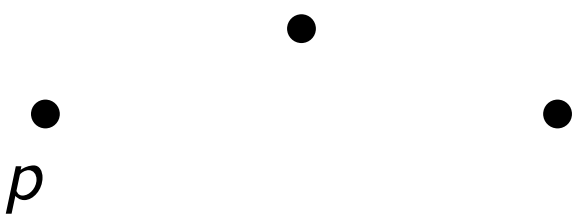
The *generalized range successor problem* is to preprocess the input to find the *successor* of each object among that object's range.

i.e. the next object in time order

dependent on the specific problem

Example problem: is there a pair of points p, q s.t. $d(p, q) > 1$?

Range of a point p: the complement of a unit disk centered at p



Approach #2: Generalized Range Successor

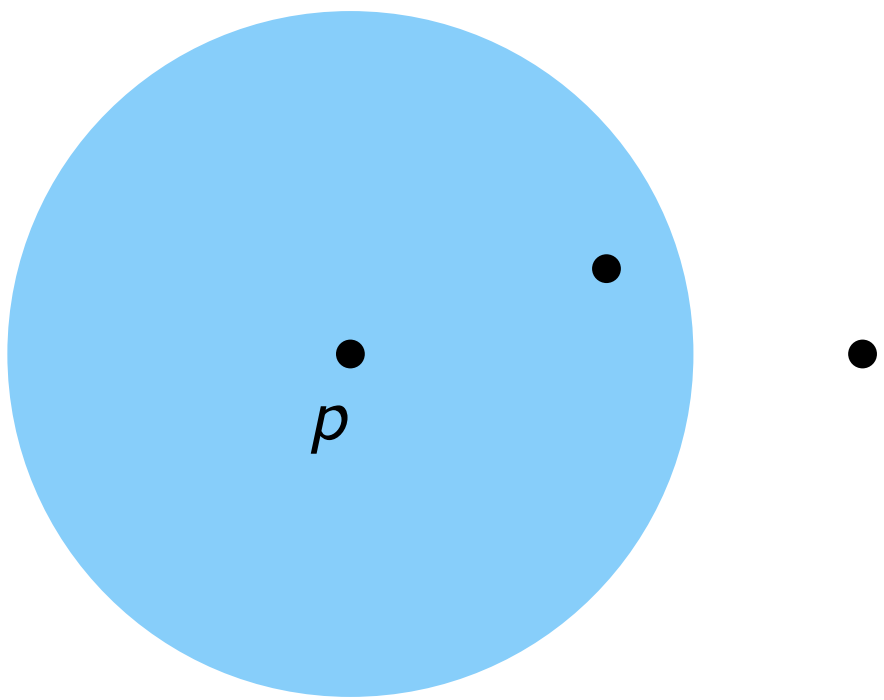
The *generalized range successor problem* is to preprocess the input to find the *successor* of each object among that object's range.

i.e. the next object in time order

dependent on the specific problem

Example problem: is there a pair of points p, q s.t. $d(p, q) > 1$?

Range of a point p: the complement of a unit disk centered at p



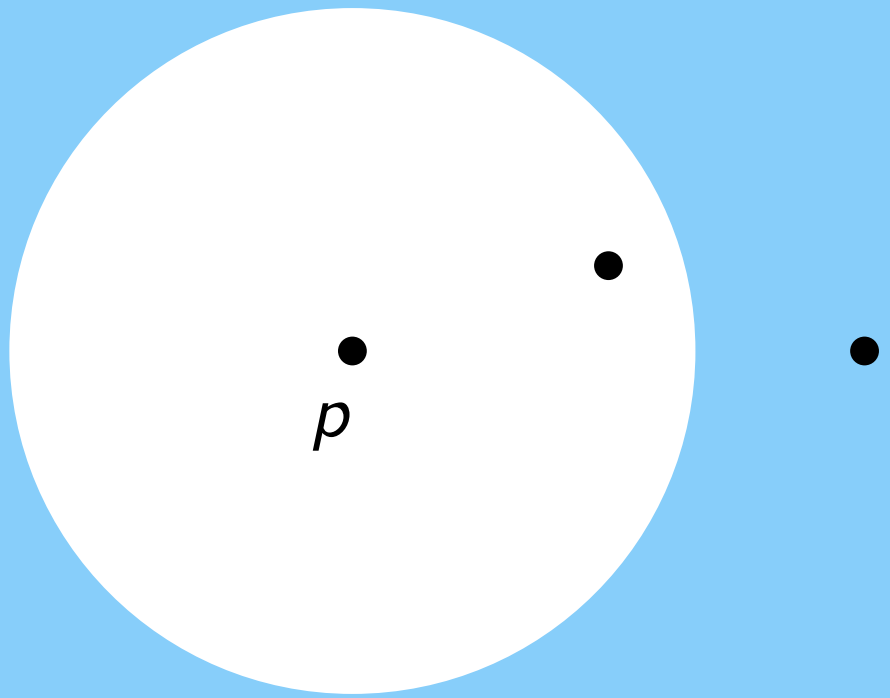
The *generalized range successor problem* is to preprocess the input to find the *successor* of each object among that object's range.

i.e. the next object in time order

dependent on the specific problem

Example problem: is there a pair of points p, q s.t. $d(p, q) > 1$?

Range of a point p: the complement of a unit disk centered at p



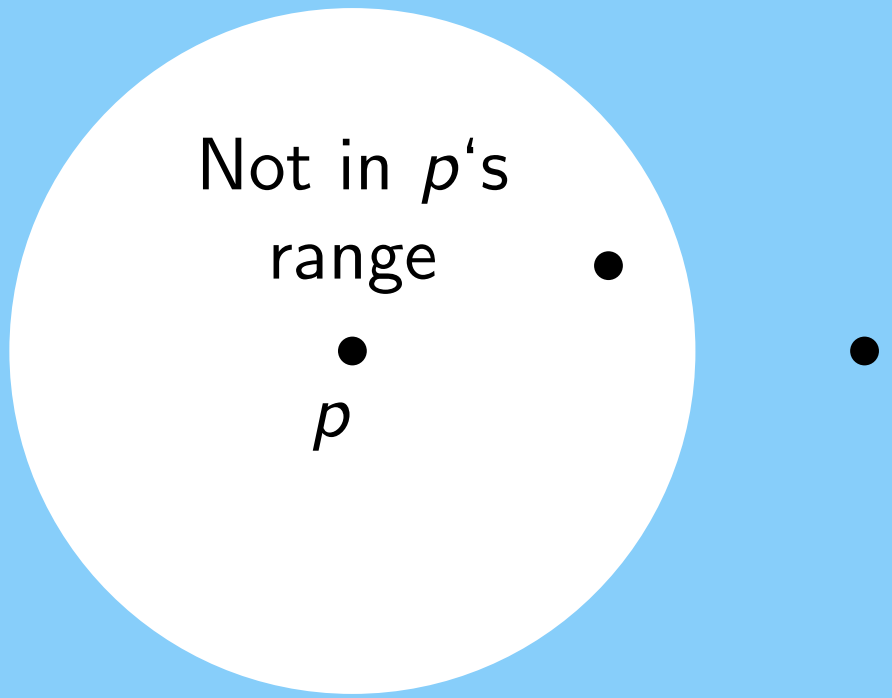
The *generalized range successor problem* is to preprocess the input to find the *successor* of each object among that object's range.

i.e. the next object in time order

dependent on the specific problem

Example problem: is there a pair of points p, q s.t. $d(p, q) > 1$?

Range of a point p: the complement of a unit disk centered at p



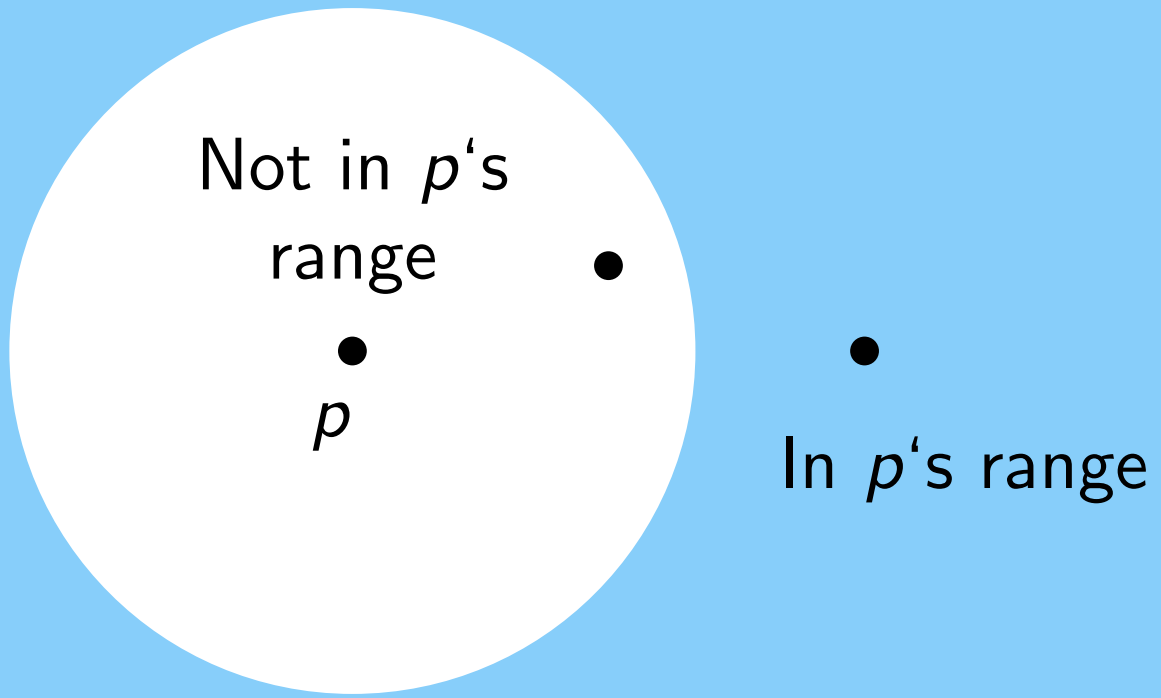
The *generalized range successor problem* is to preprocess the input to find the *successor* of each object among that object's range.

i.e. the next object in time order

dependent on the specific problem

Example problem: is there a pair of points p, q s.t. $d(p, q) > 1$?

Range of a point p: the complement of a unit disk centered at p



Why Generalized Range Successor?

Why Generalized Range Successor?

Recall this table

i	j
1	j_1
$\vdots$	$\vdots$
i	j_i
$\vdots$	$\vdots$
n	j_n

Why Generalized Range Successor?

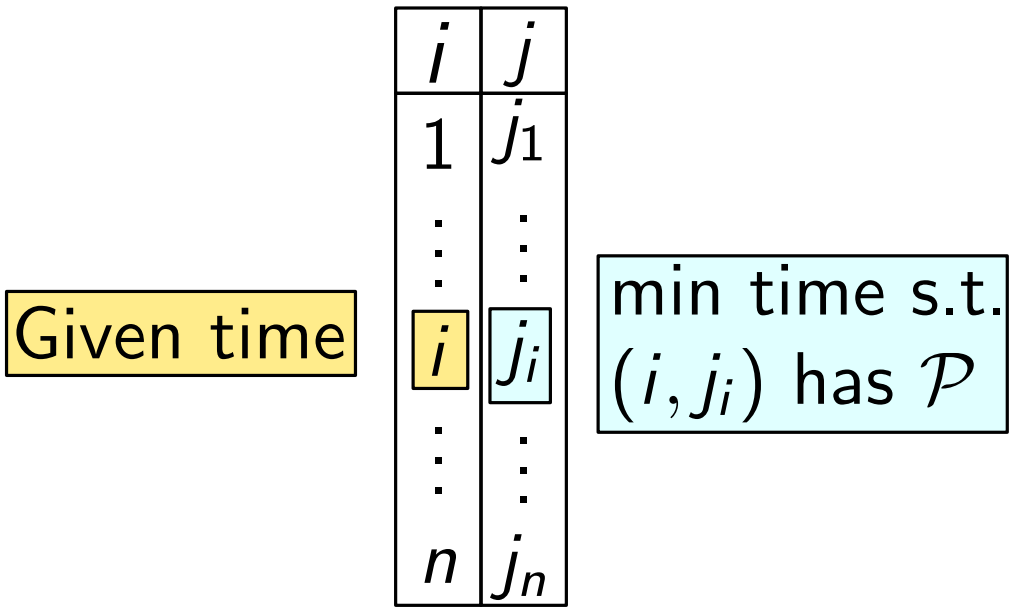
Recall this table

Given time

i	j
1	j_1
⋮	⋮
i	j_i
⋮	⋮
n	j_n

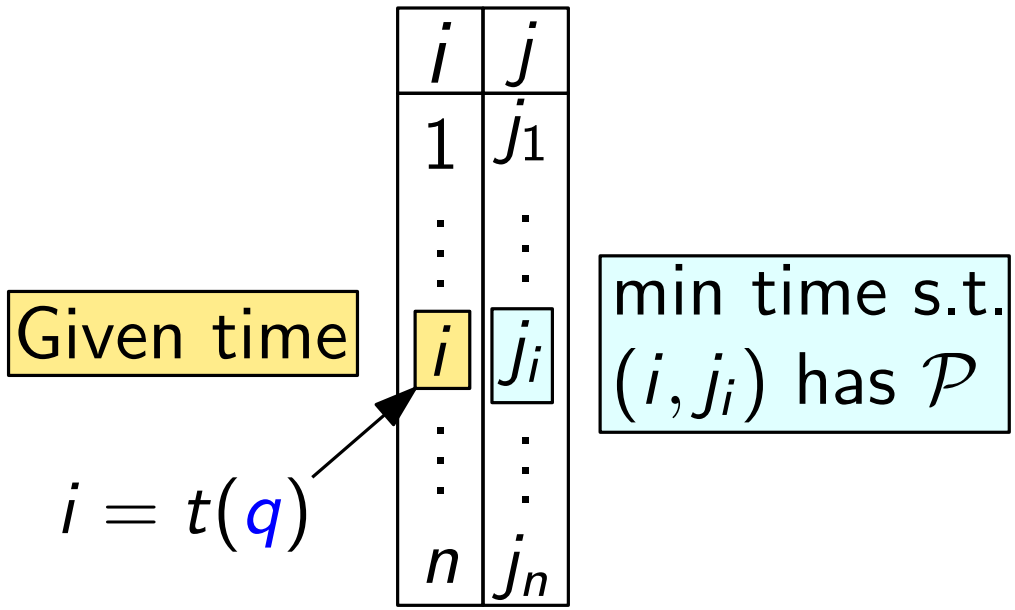
Why Generalized Range Successor?

Recall this table



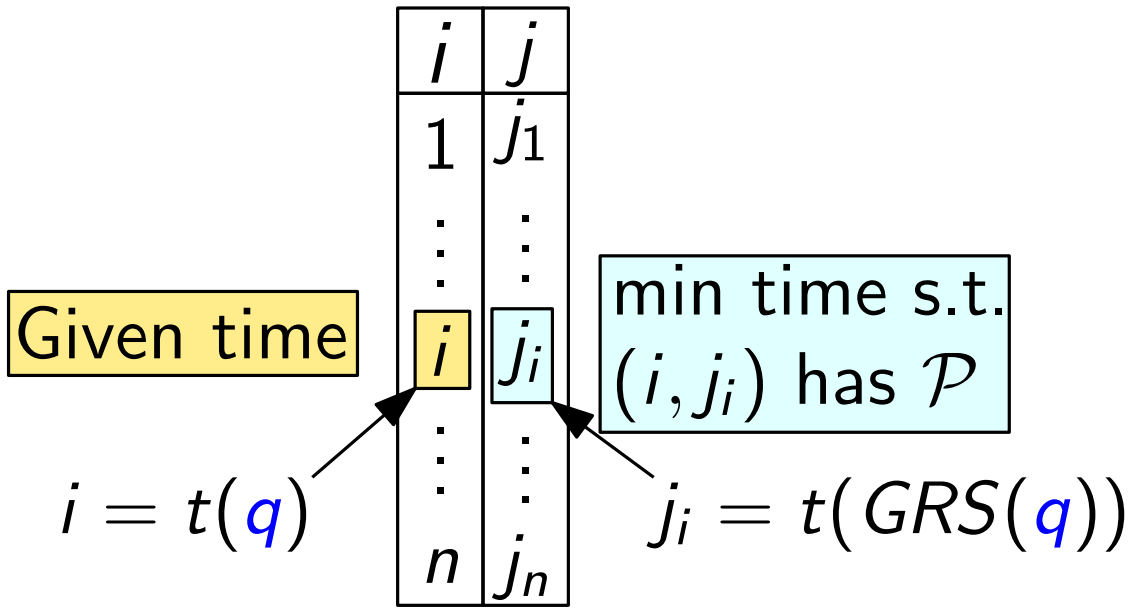
Why Generalized Range Successor?

Recall this table



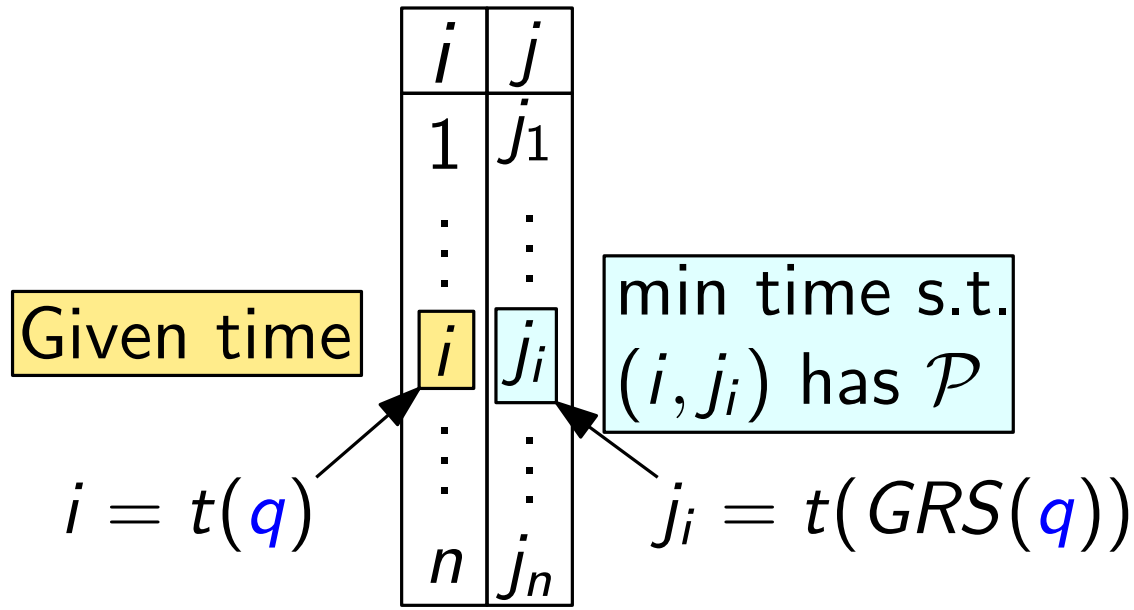
Why Generalized Range Successor?

Recall this table



Why Generalized Range Successor?

Recall this table

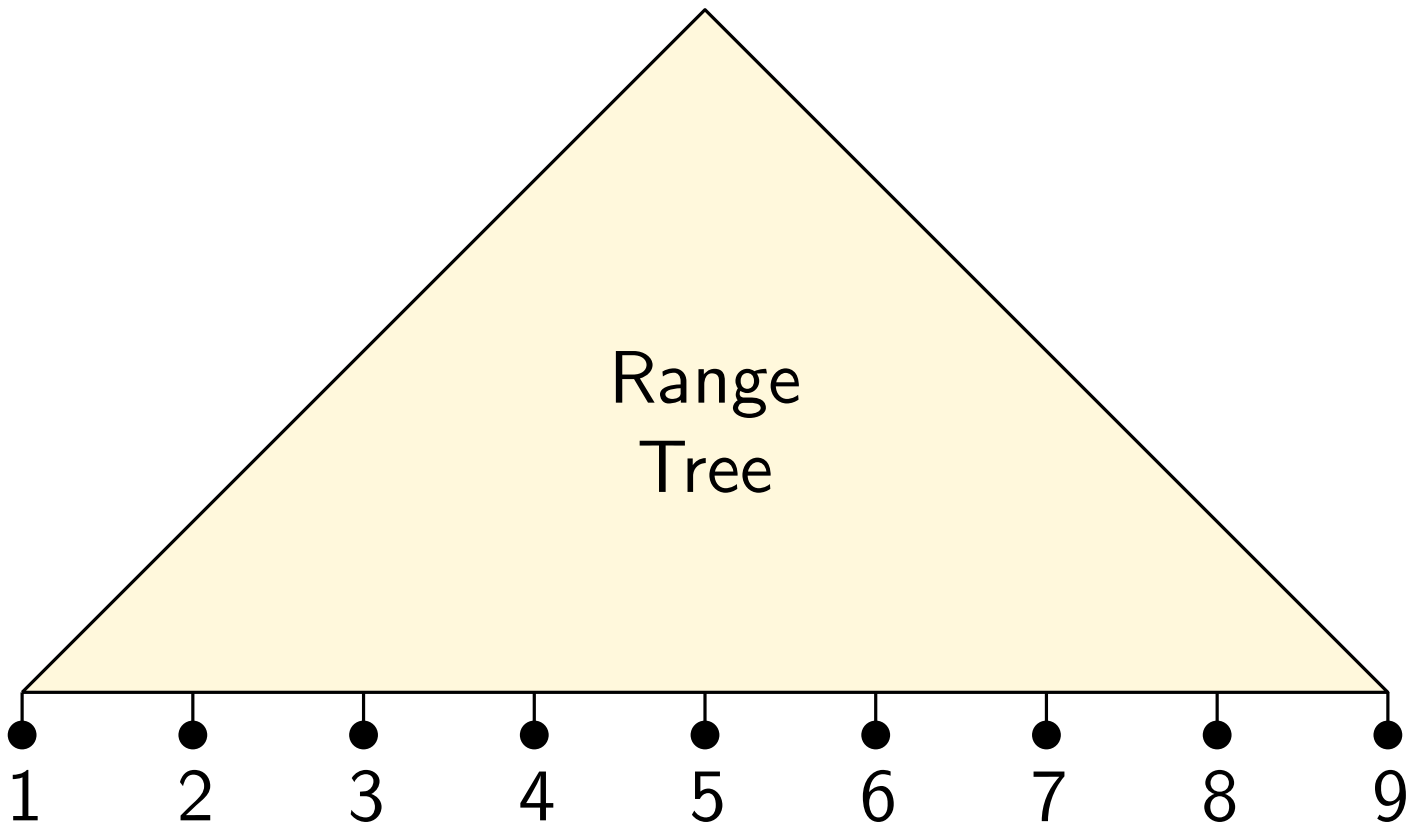


$\therefore$ can build table by finding $GRS(q)$ for all q

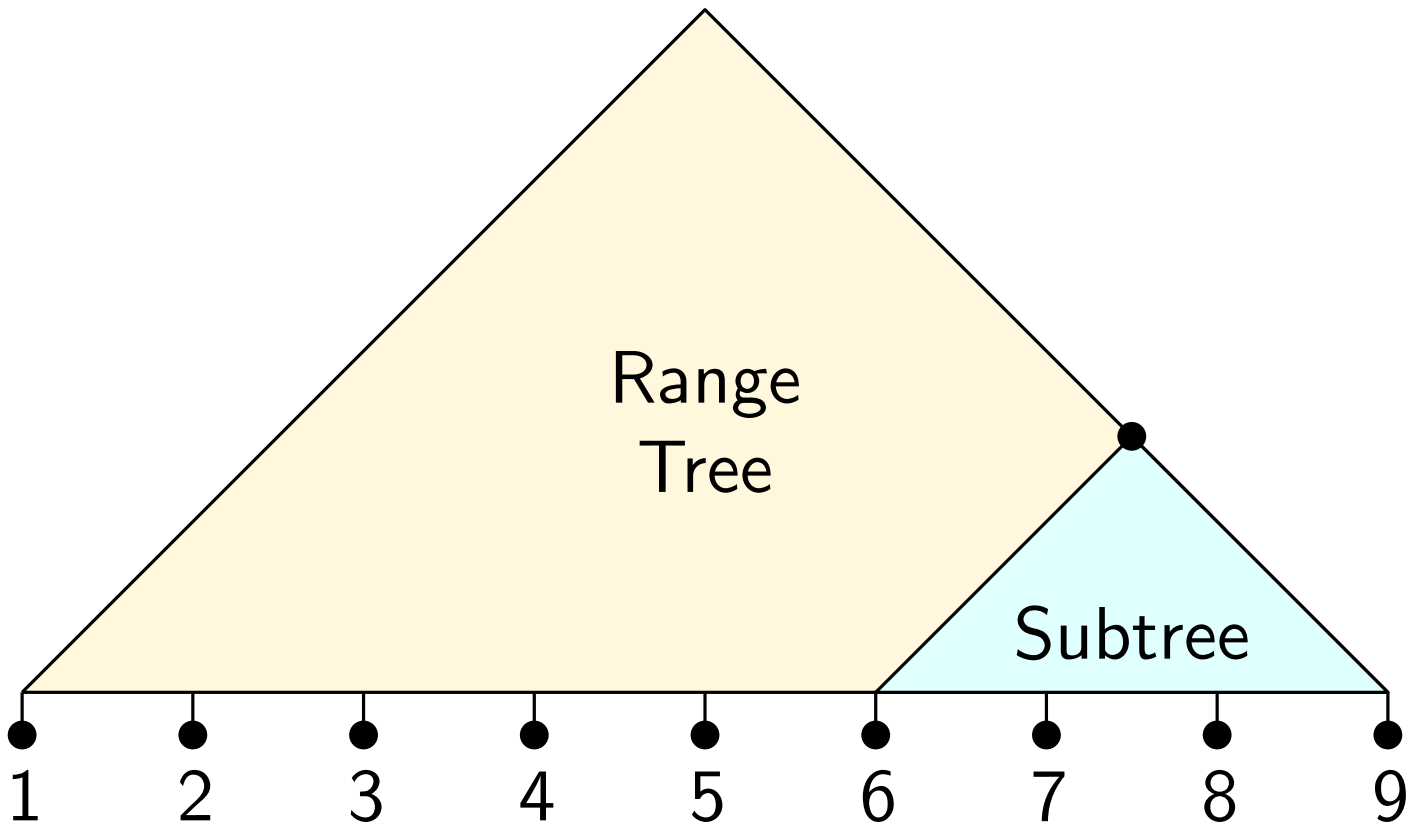
Generalized Range Successor

Consider the objects in time-order

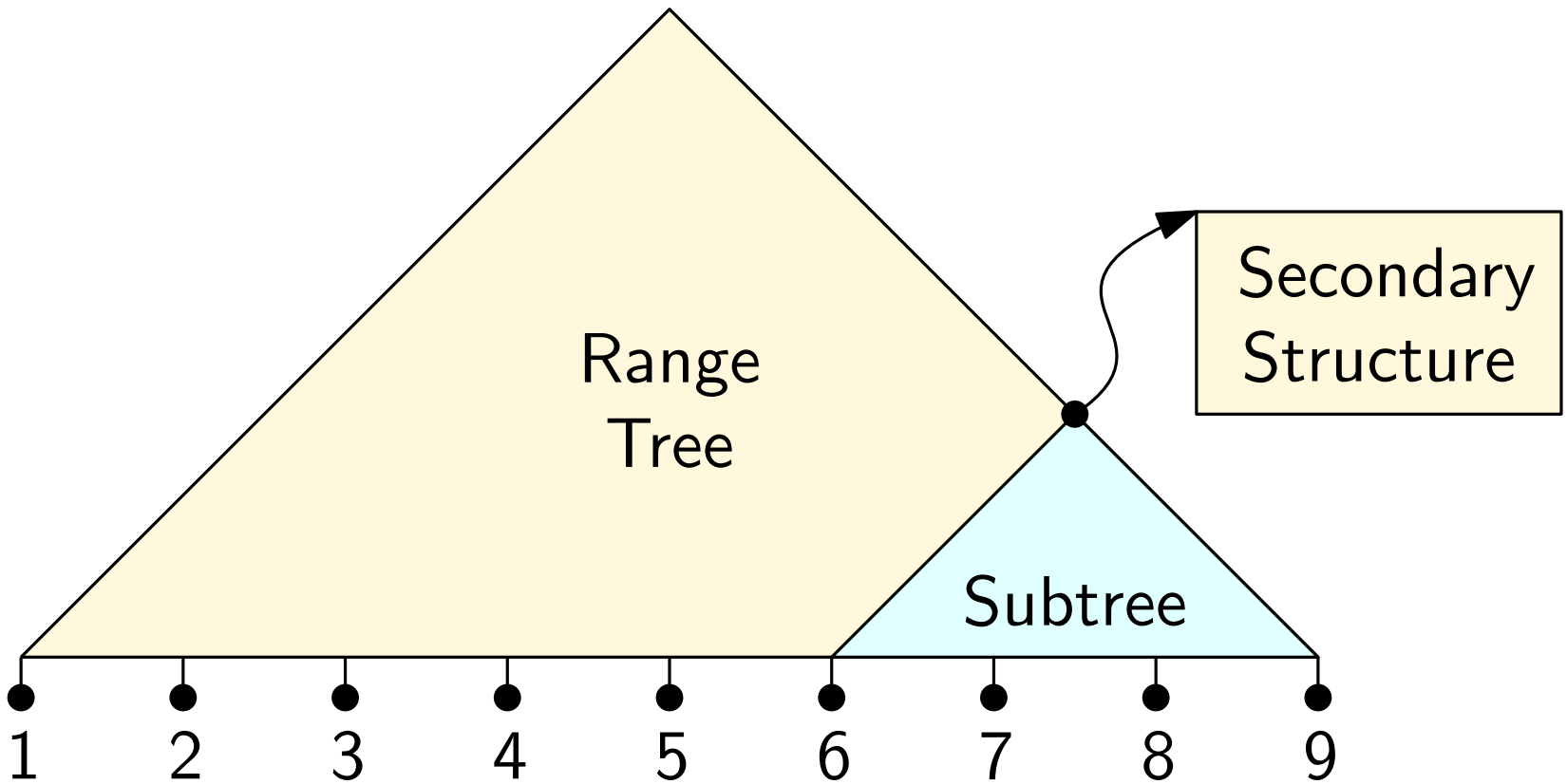
Generalized Range Successor



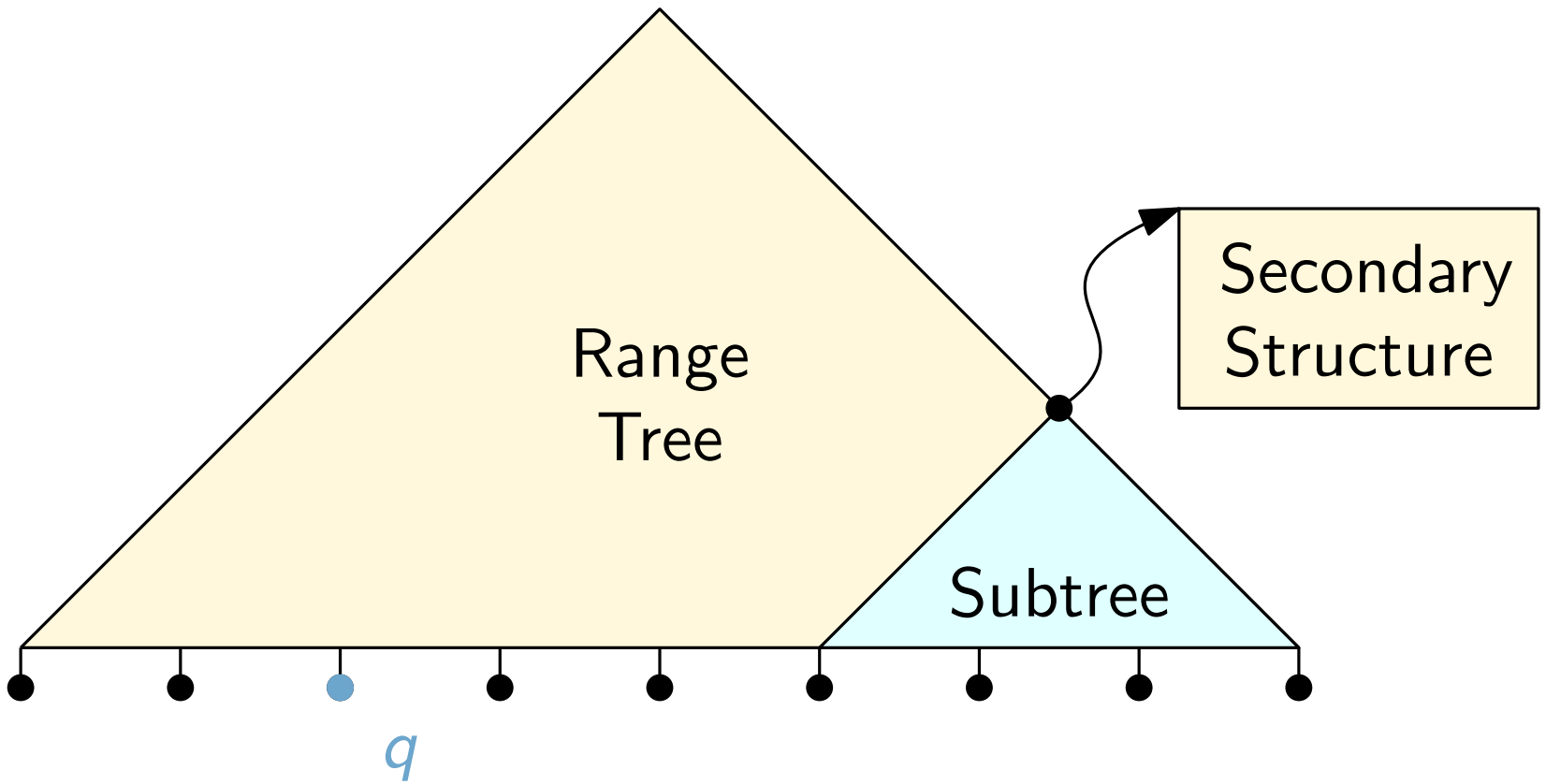
Generalized Range Successor



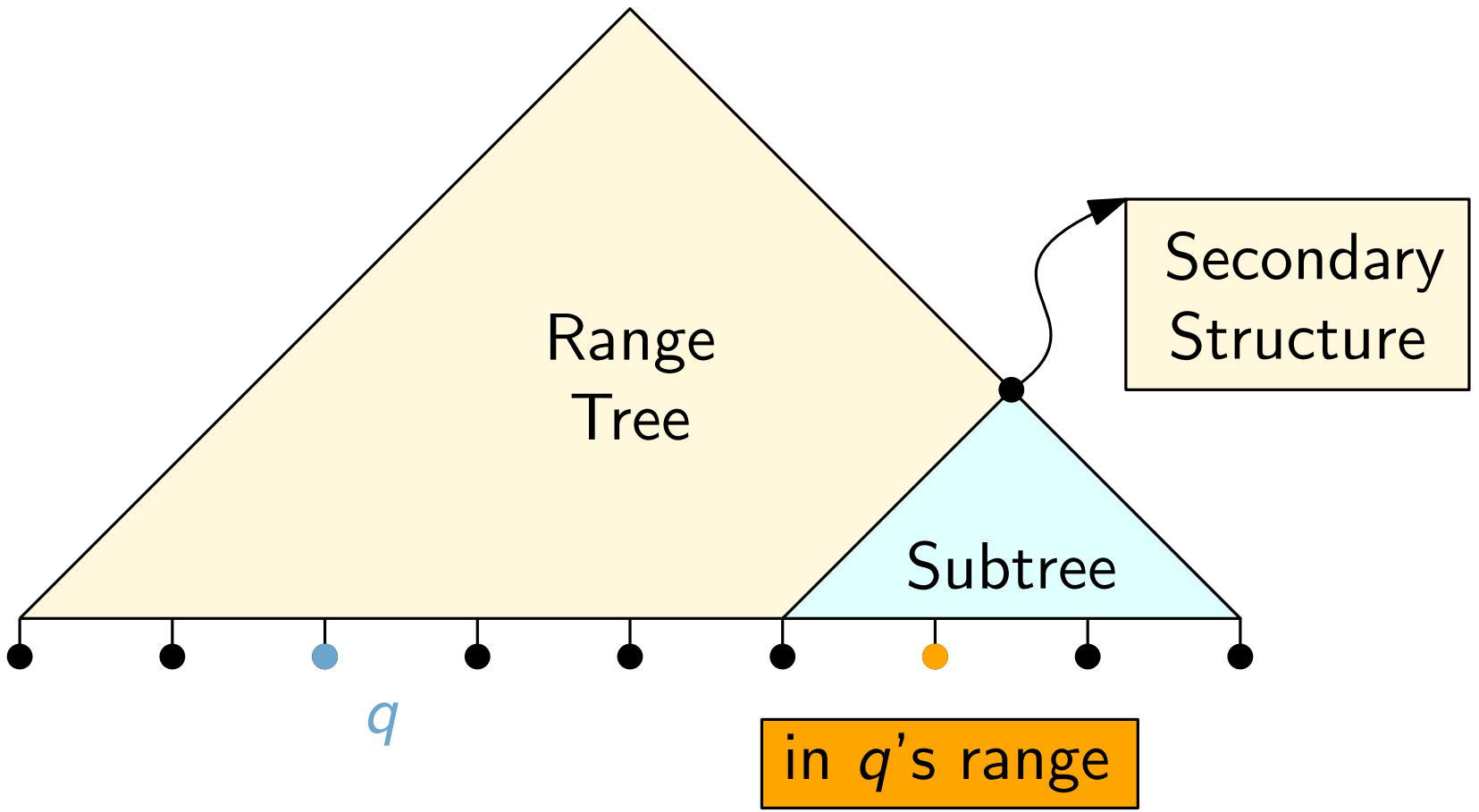
Generalized Range Successor



Generalized Range Successor

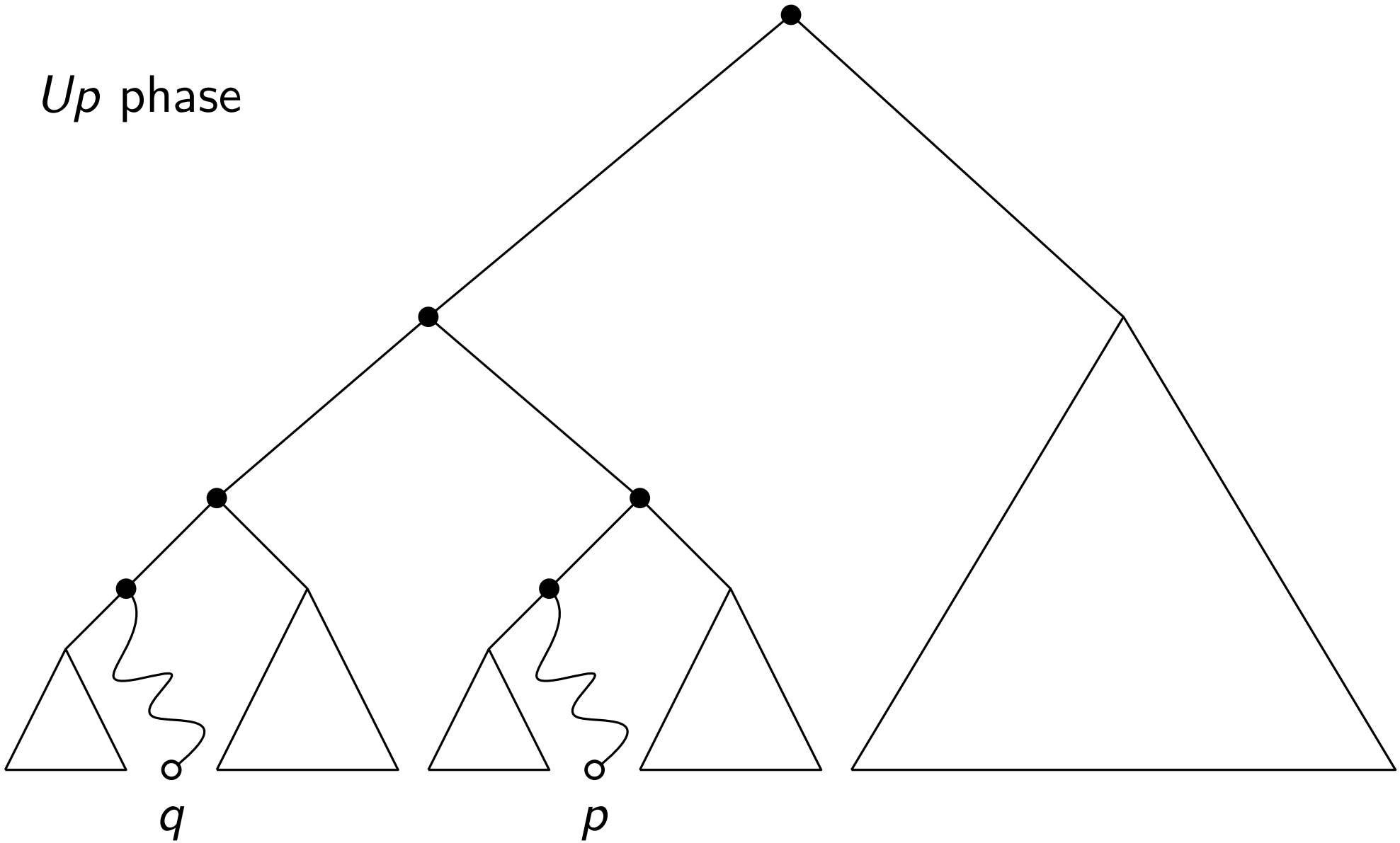


Generalized Range Successor



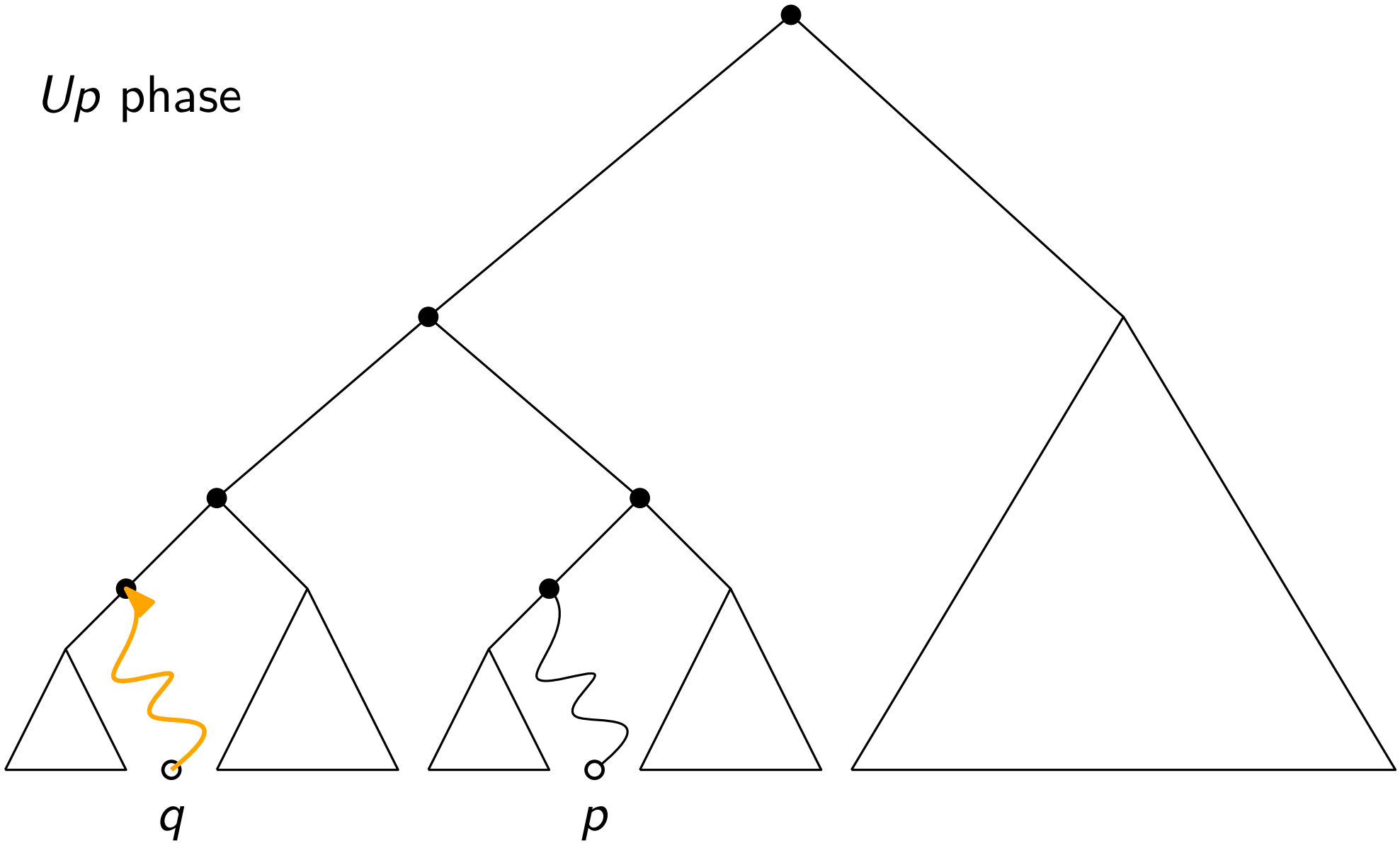
Avoiding Binary Search

Up phase



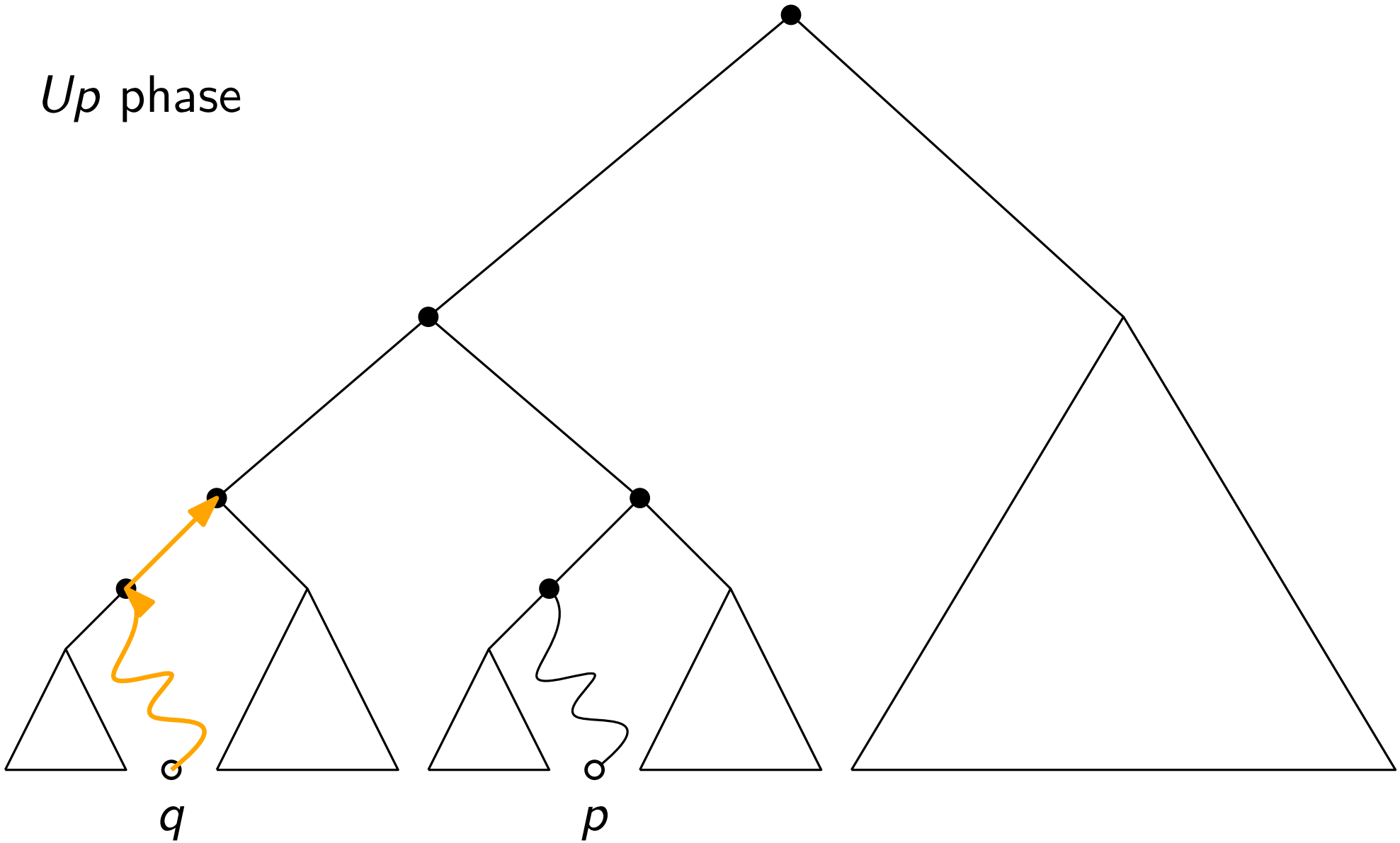
Avoiding Binary Search

Up phase

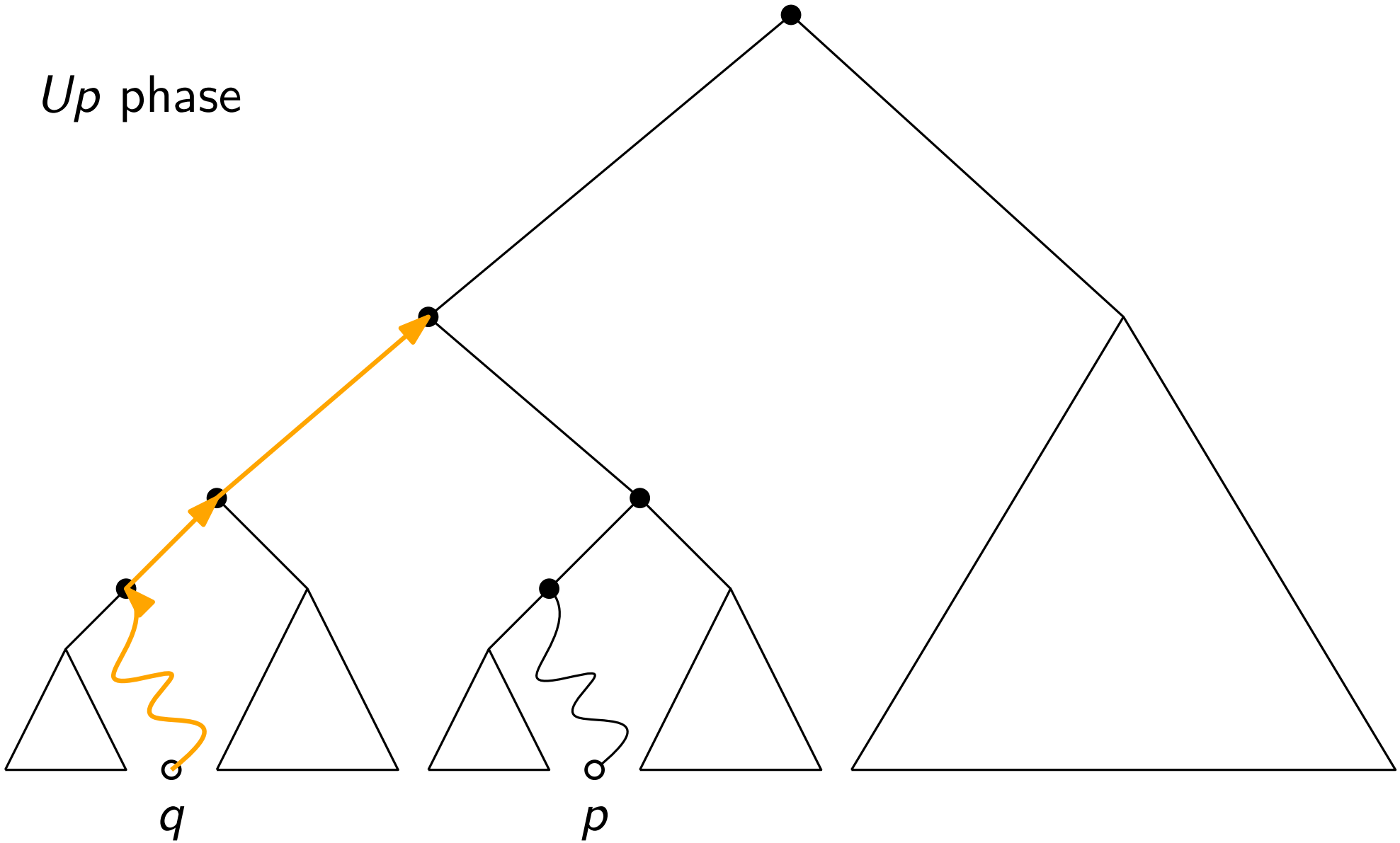


Avoiding Binary Search

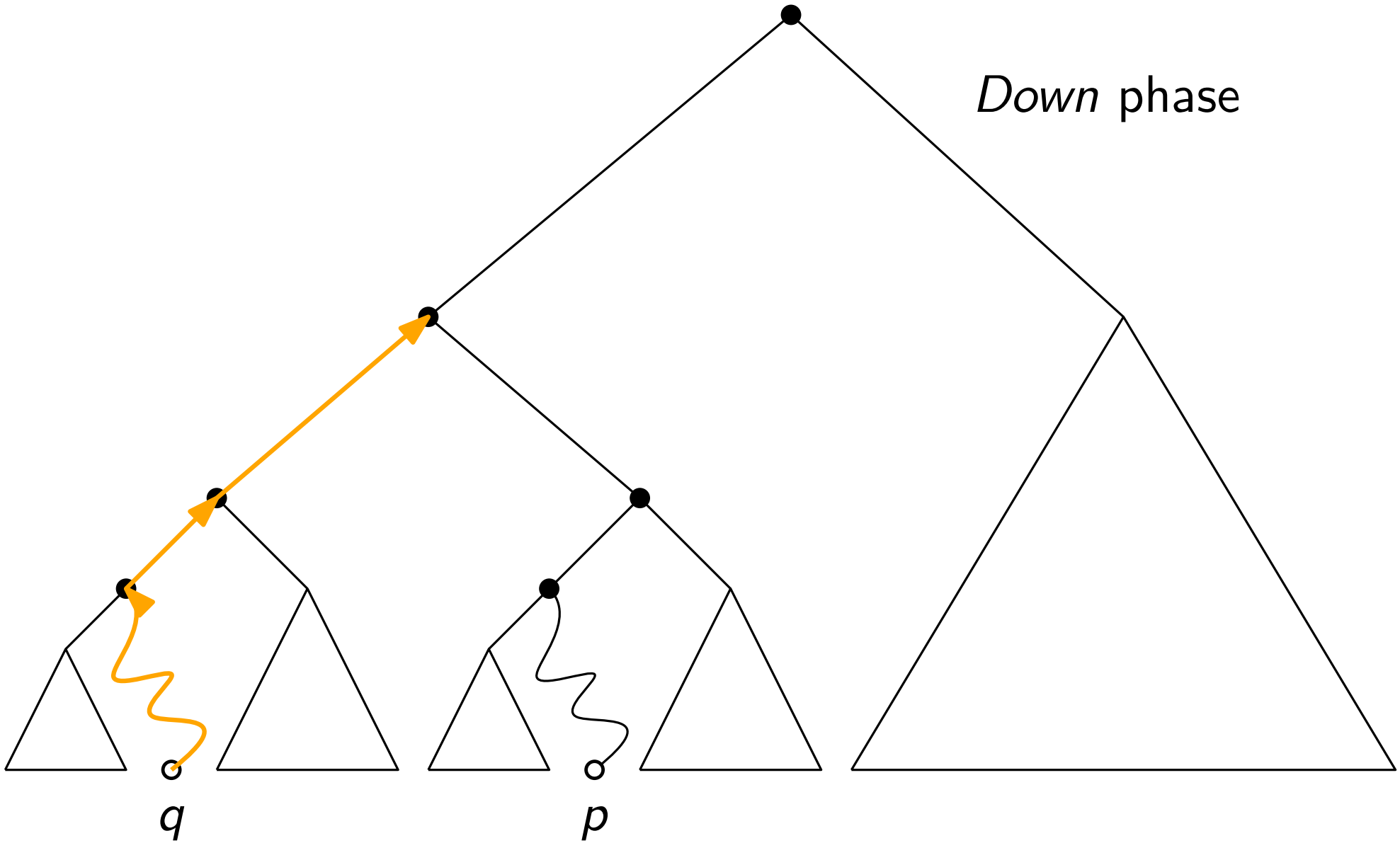
Up phase



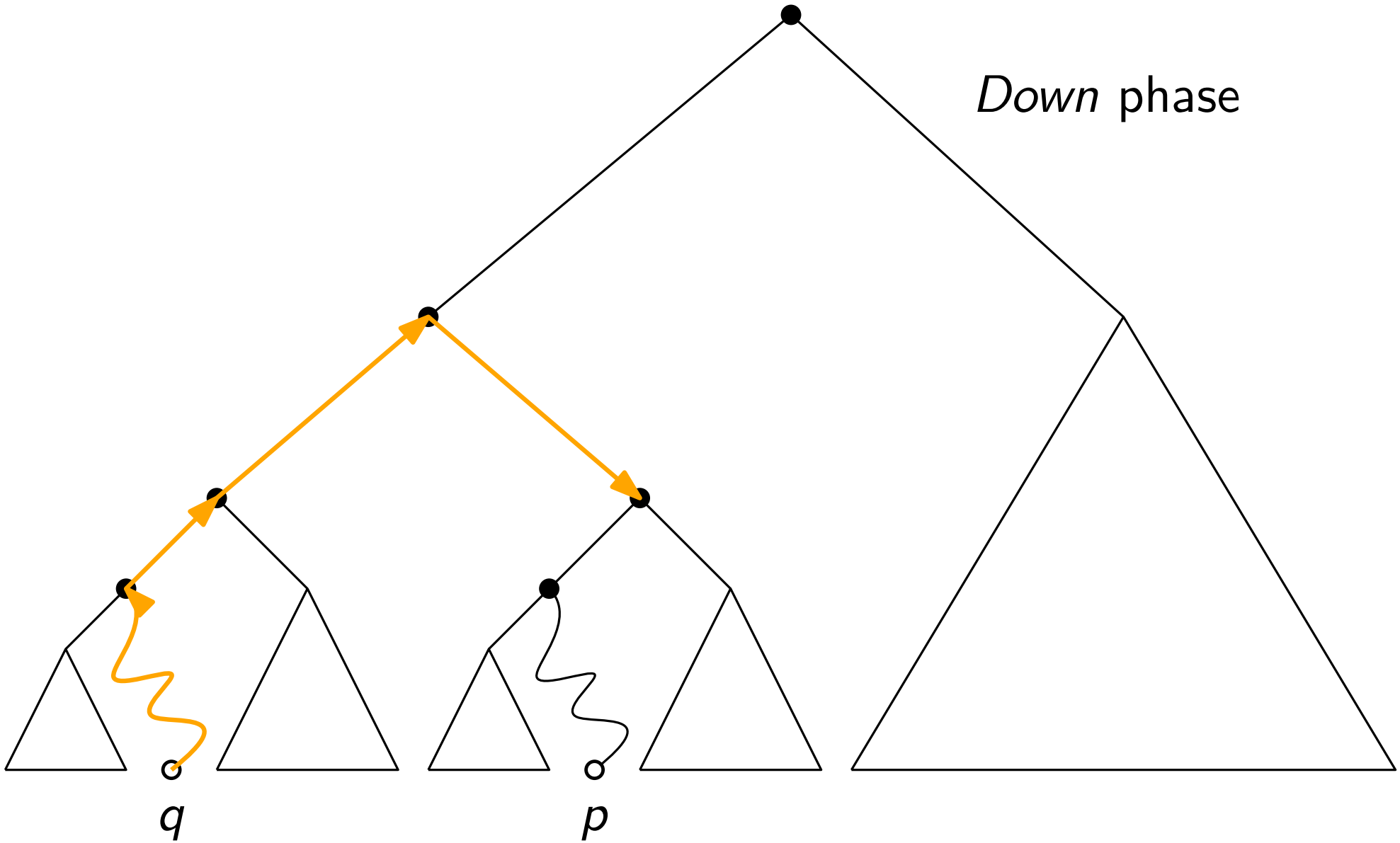
Avoiding Binary Search



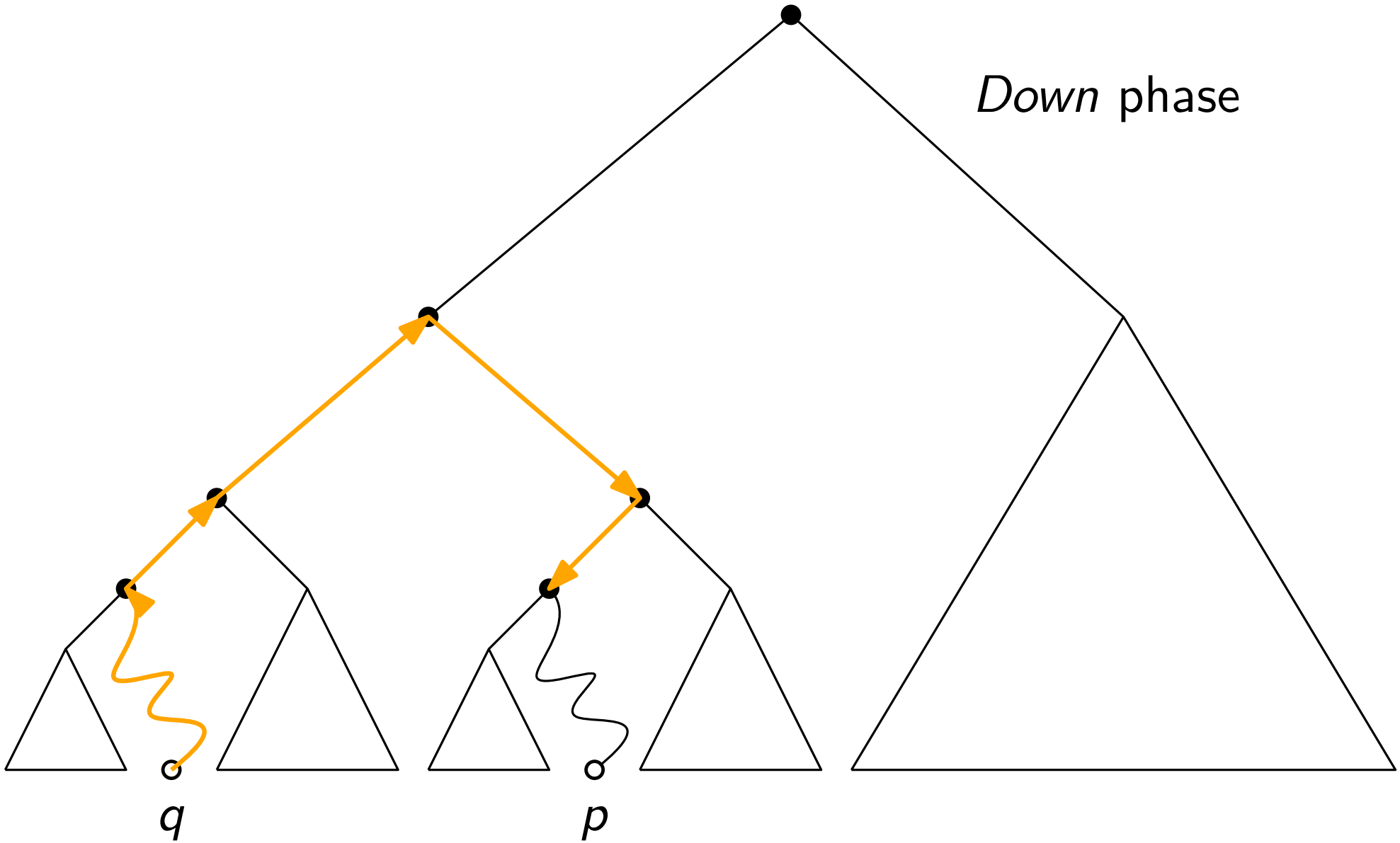
Avoiding Binary Search



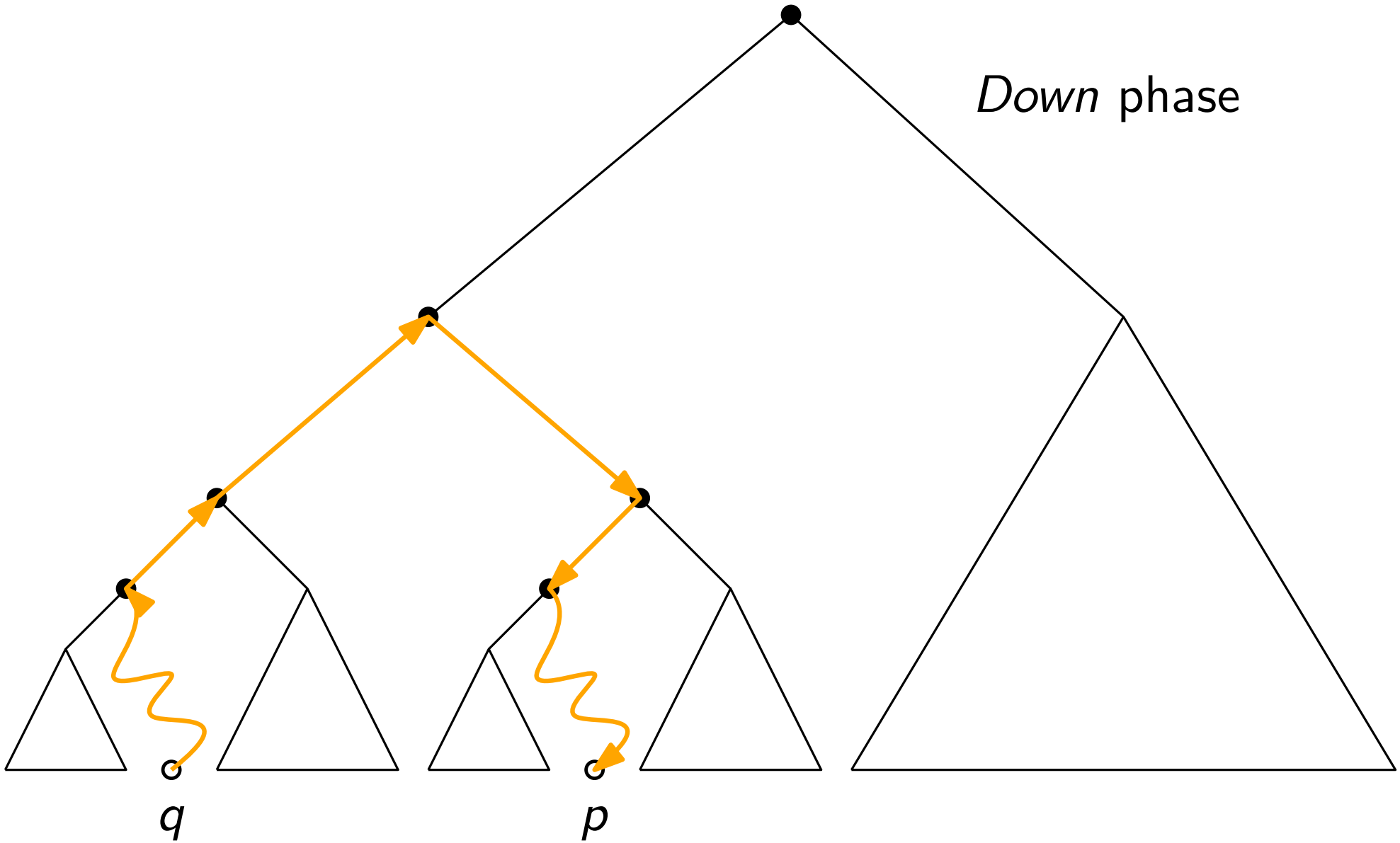
Avoiding Binary Search



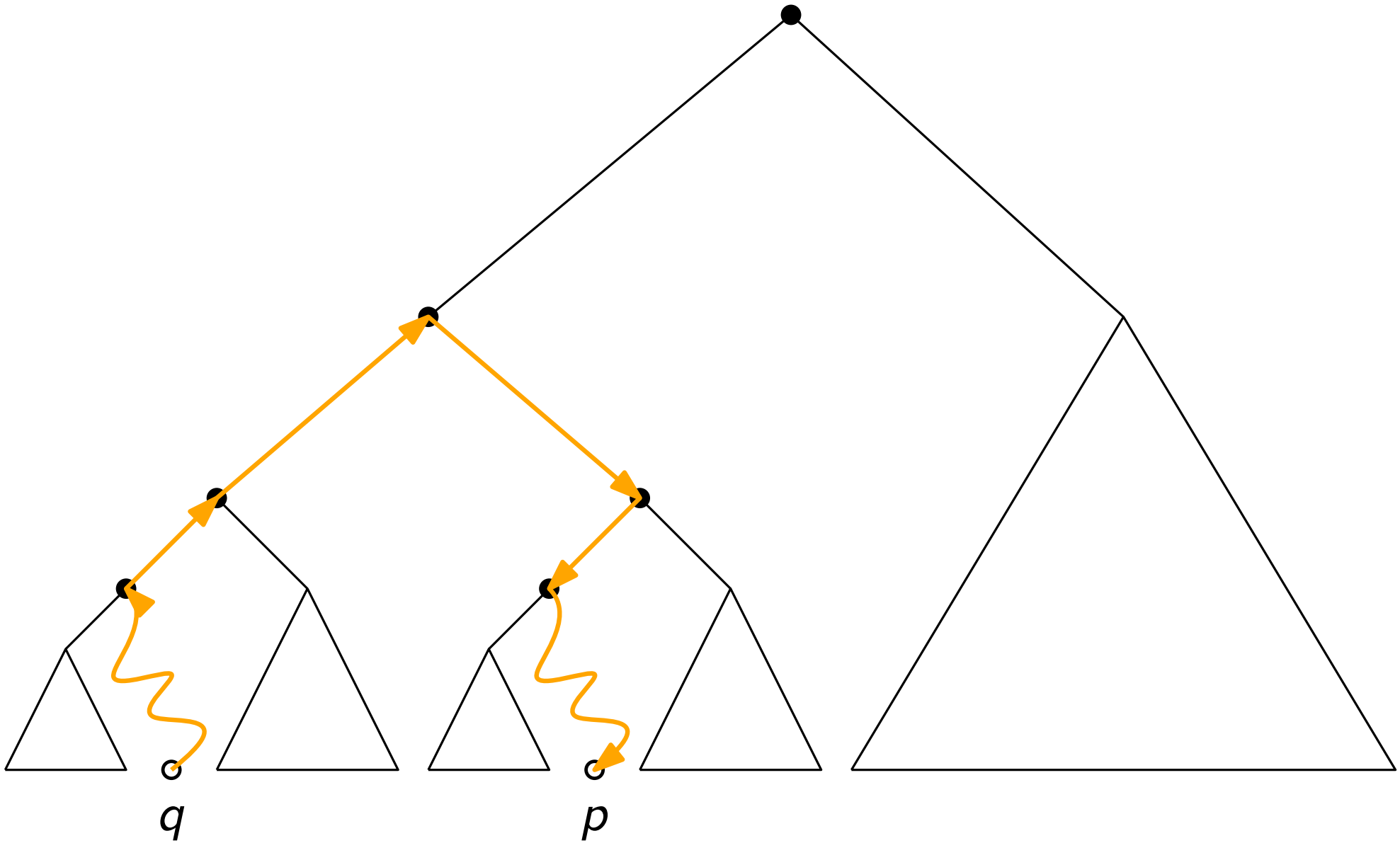
Avoiding Binary Search



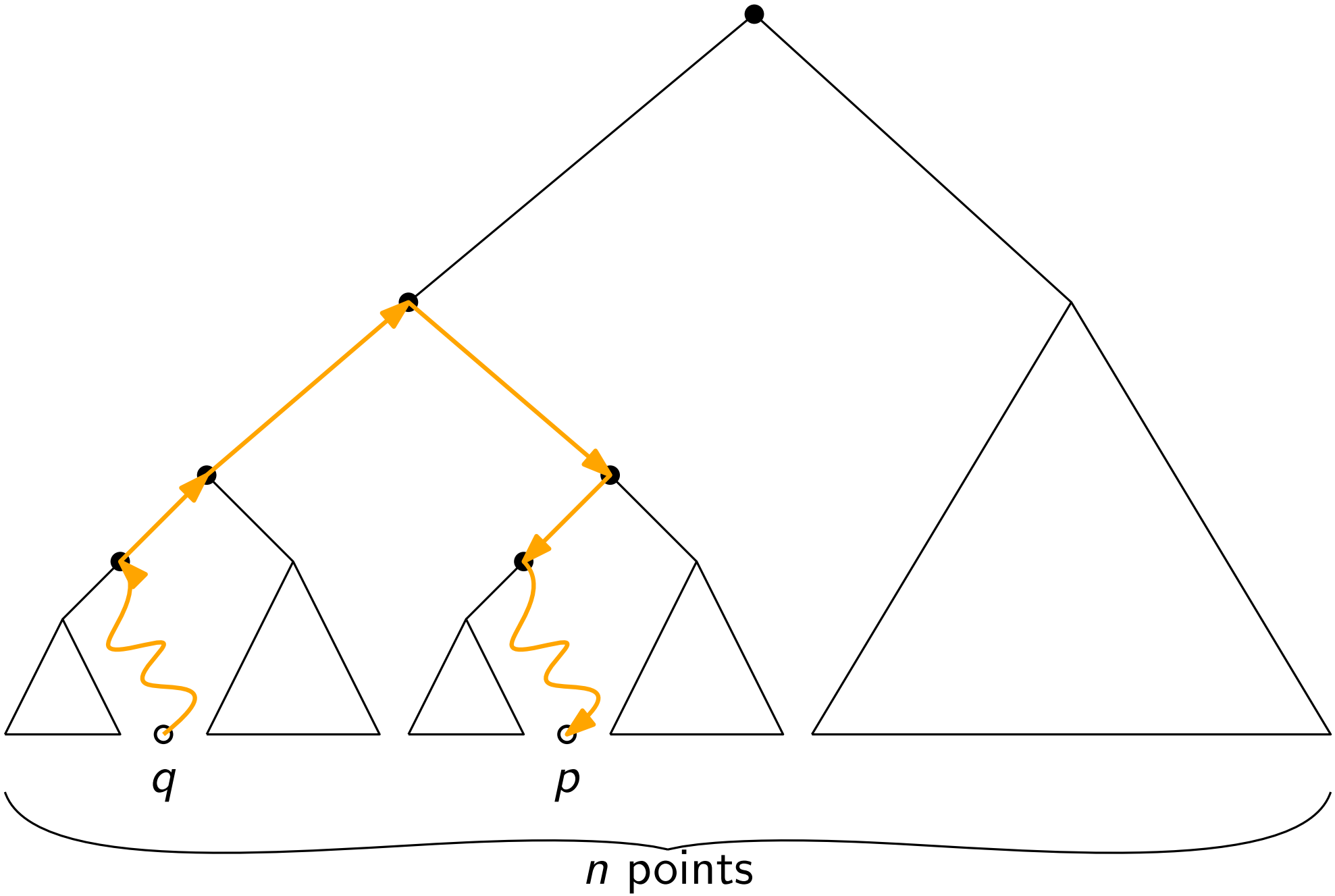
Avoiding Binary Search



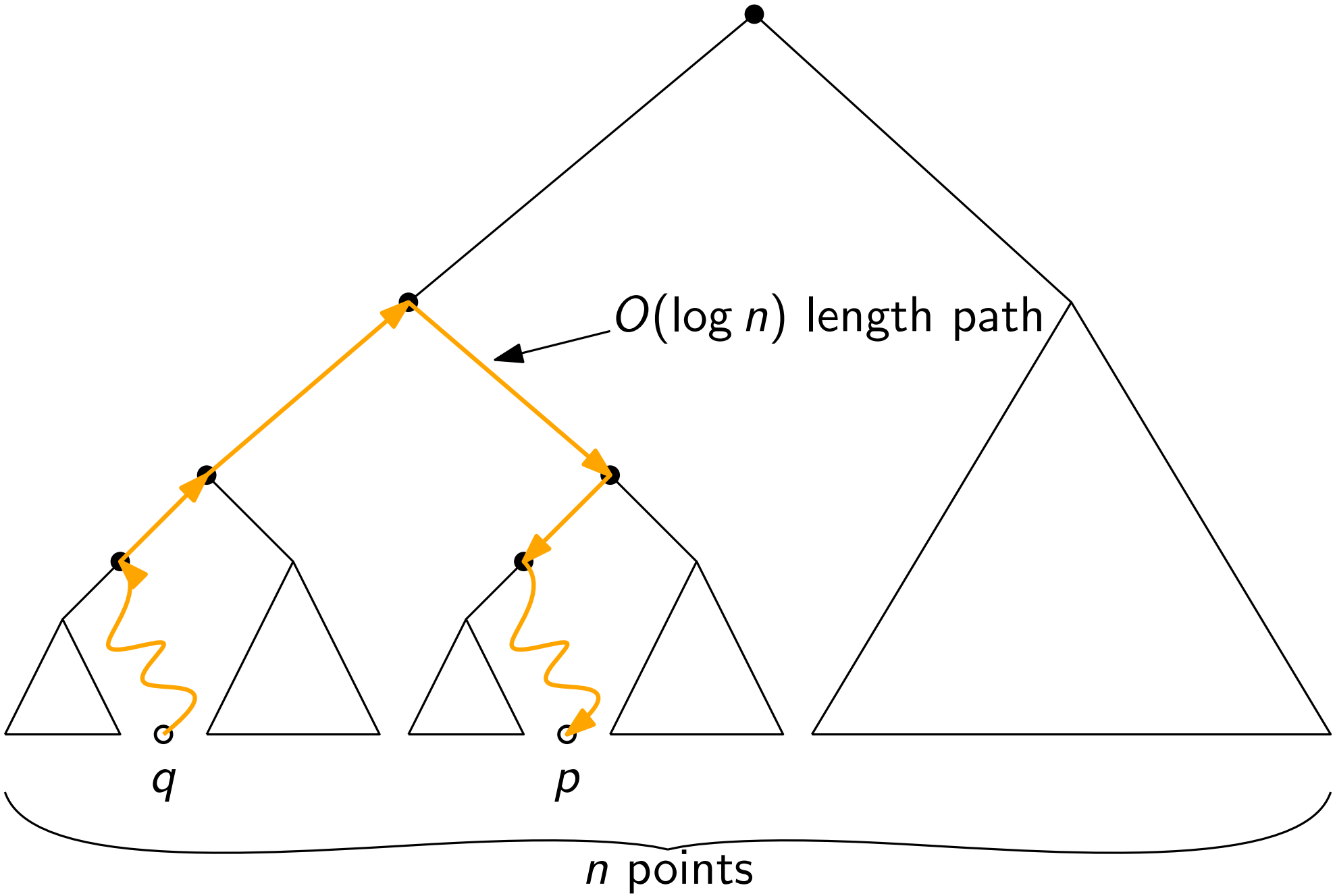
Quick Analysis



Quick Analysis

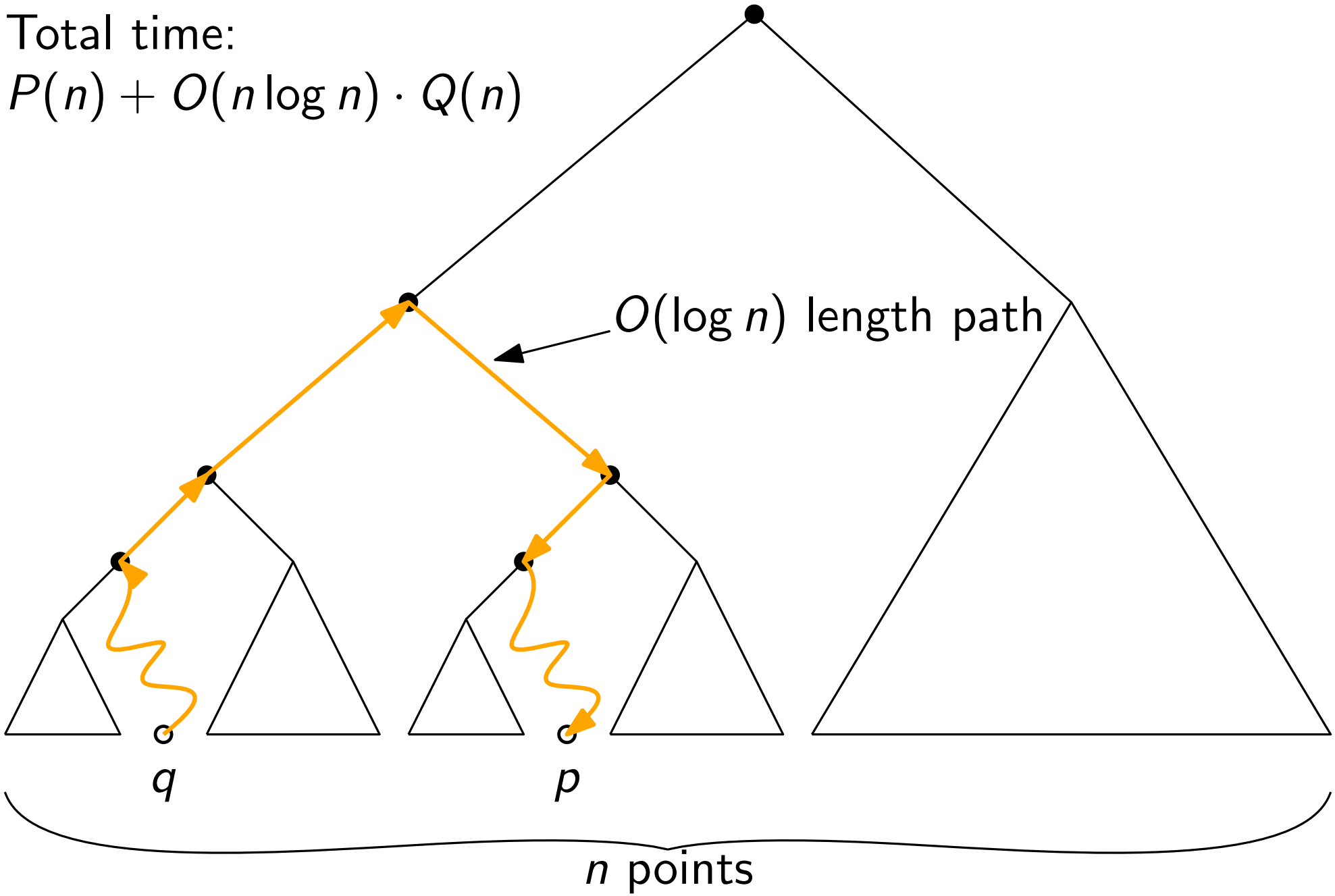


Quick Analysis



Quick Analysis

Total time:
$P(n) + O(n \log n) \cdot Q(n)$



Generalized Range Successor: Conclusion

2D diameter decision

Generalized Range Successor: Conclusion

2D diameter decision

- Point location in unit disk intersection $\rightarrow$ binary search
  $O(\log n) \cdot O(n \log n) = O(n \log^2 n)$

Generalized Range Successor: Conclusion

2D diameter decision

- Point location in unit disk intersection $\rightarrow$ binary search $O(\log n) \cdot O(n \log n) = O(n \log^2 n)$
- Fractional cascading $\rightarrow$ $O(n \log n)$

Generalized Range Successor: Conclusion

2D diameter decision

- Point location in unit disk intersection $\rightarrow$ binary search $O(\log n) \cdot O(n \log n) = O(n \log^2 n)$
- Fractional cascading $\rightarrow$ $O(n \log n)$

3D diameter decision

Generalized Range Successor: Conclusion

2D diameter decision

- Point location in unit disk intersection $\rightarrow$ binary search
  $O(\log n) \cdot O(n \log n) = O(n \log^2 n)$
- Fractional cascading $\rightarrow$ $O(n \log n)$

3D diameter decision

- Point location in unit ball intersection $\rightarrow$ 2D point location
  $O(\log n) \cdot O(n \log n) = O(n \log^2 n)$

Generalized Range Successor: Conclusion

2D diameter decision

- Point location in unit disk intersection $\rightarrow$ binary search
  $O(\log n) \cdot O(n \log n) = O(n \log^2 n)$
- Fractional cascading $\rightarrow$ $O(n \log n)$

3D diameter decision

- Point location in unit ball intersection $\rightarrow$ 2D point location
  $O(\log n) \cdot O(n \log n) = O(n \log^2 n)$

Orthogonal segment intersection detection

Generalized Range Successor: Conclusion

2D diameter decision

- Point location in unit disk intersection $\rightarrow$ binary search
  $O(\log n) \cdot O(n \log n) = O(n \log^2 n)$
- Fractional cascading $\rightarrow$ $O(n \log n)$

3D diameter decision

- Point location in unit ball intersection $\rightarrow$ 2D point location
  $O(\log n) \cdot O(n \log n) = O(n \log^2 n)$

Orthogonal segment intersection detection

- Orthogonal intersection detection $\rightarrow$ point location
  $O(\log \log n) \cdot O(n \log n) = O(n \log n \log \log n)$

New Results

Approach #1: Grids & Quadtrees

Problem	Preprocessing	Space	Query
CP Decision	$O(n)$ exp.	$O(n)$ bits	$O(1)$
Closest Pair	$O(n \lg n \lg\lg n)$	$O(n \lg n)$	$O(\lg\lg n)$

Approach #2: Generalized Range Successor

Problem	Preprocessing Time
2D diameter decision	$O(n \lg^2 n)$
$\rightarrow$ improved to	$O(n \lg n)$
3D diameter decision	$O(n \lg^2 n)$
2D orth. seg. intersection detection	$O(n \lg n \lg\lg n)$

Approach #3: FIFO Update Sequence

Problem	Preprocessing Time
2D convex hull area decision	$O(n \lg n \lg\lg n)$
$\rightarrow$ improved to	$O(n\alpha(n) \lg n)$
2D width decision	$O(n \lg^7 n)$

New Results

Approach #1: Grids & Quadtrees

Problem	Preprocessing	Space	Query
CP Decision	$O(n)$ exp.	$O(n)$ bits	$O(1)$
Closest Pair	$O(n \lg n \lg\lg n)$	$O(n \lg n)$	$O(\lg\lg n)$

Approach #2: Generalized Range Successor

Problem	Preprocessing Time
2D diameter decision	$O(n \lg^2 n)$
$\rightarrow$ improved to	$O(n \lg n)$
3D diameter decision	$O(n \lg^2 n)$
2D orth. seg. intersection detection	$O(n \lg n \lg\lg n)$

Approach #3: FIFO Update Sequence

Problem	Preprocessing Time
2D convex hull area decision	$O(n \lg n \lg\lg n)$
$\rightarrow$ improved to	$O(n\alpha(n) \lg n)$
2D width decision	$O(n \lg^7 n)$

Time-Windowed Decision $\rightarrow$ Dynamic

Time-Windowed Decision $\rightarrow$ Dynamic

Recall this table

i	j
1	j_1
$\vdots$	$\vdots$
i	j_i
$\vdots$	$\vdots$
n	j_n

Time-Windowed Decision $\rightarrow$ Dynamic

One way to build it:

i	j
1	
2	
3	
4	
$\vdots$	
n	

Time-Windowed Decision $\rightarrow$ Dynamic

One way to build it:

- Use a dynamic DS

	Dynamic DS	
	Yes	No
Property Satisfied:	o	✓

i	j
1	
2	
3	
4	
⋮	
n	

Time-Windowed Decision $\rightarrow$ Dynamic

One way to build it:

- Use a dynamic DS
- Insert until satisfied

Dynamic DS		
	Yes	No
Property Satisfied:	o	✓

i	j
1	
2	
3	
4	
⋮	
n	

Time-Windowed Decision $\rightarrow$ Dynamic

One way to build it:

- Use a dynamic DS
- Insert until satisfied

	Dynamic DS	
	Yes	No
Property Satisfied:	o	✓

Insert 1

i	j
1	
2	
3	
4	
⋮	
n	

Time-Windowed Decision $\rightarrow$ Dynamic

One way to build it:

- Use a dynamic DS
- Insert until satisfied

	Dynamic DS	
	Yes	No
Property Satisfied:	o	✓

Insert 1
Insert 2

i	j
1	
2	
3	
4	
⋮	
n	

Time-Windowed Decision $\rightarrow$ Dynamic

One way to build it:

- Use a dynamic DS
- Insert until satisfied

	Dynamic DS	
	Yes	No
Property Satisfied:	✓	o

Insert 1
Insert 2
Insert 3

i	j
1	
2	
3	
4	
⋮	
n	

Time-Windowed Decision $\rightarrow$ Dynamic

One way to build it:

- Use a dynamic DS
- Insert until satisfied

	Dynamic DS	
	Yes	No
Property Satisfied:	✓	o

Insert 1
Insert 2
Insert 3

i	j
1	3
2	
3	
4	
⋮	
n	

Time-Windowed Decision $\rightarrow$ Dynamic

	Dynamic DS	
	Yes	No
Property Satisfied:	✓	o

One way to build it:

- Use a dynamic DS
- Insert until satisfied
- Delete (from beginning) until not satisfied

Insert 1
Insert 2
Insert 3

i	j
1	3
2	
3	
4	
⋮	
n	

Time-Windowed Decision $\rightarrow$ Dynamic

	Dynamic DS	
	Yes	No
Property Satisfied:	✓	o

One way to build it:

- Use a dynamic DS
- Insert until satisfied
- Delete (from beginning) until not satisfied

Insert 1 Delete
Insert 2
Insert 3

i	j
1	3
2	
3	
4	
⋮	
n	

Time-Windowed Decision $\rightarrow$ Dynamic

	Dynamic DS	
	Yes	No
Property Satisfied:	✓	o

One way to build it:

- Use a dynamic DS
- Insert until satisfied
- Delete (from beginning) until not satisfied

Insert 1 Delete
Insert 2
Insert 3

i	j
1	3
2	3
3	
4	
⋮	
n	

Time-Windowed Decision $\rightarrow$ Dynamic

	Dynamic DS	
	Yes	No
Property Satisfied:	o	✓

One way to build it:

- Use a dynamic DS
- Insert until satisfied
- Delete (from beginning) until not satisfied

Insert 1 Delete
Insert 2 Delete
Insert 3

i	j
1	3
2	3
3	
4	
⋮	
n	

Time-Windowed Decision $\rightarrow$ Dynamic

	Dynamic DS	
	Yes	No
Property Satisfied:	o	✓

One way to build it:

- Use a dynamic DS
- Insert until satisfied
- Delete (from beginning) until not satisfied
- Repeat until all points deleted

Insert 1 Delete
Insert 2 Delete
Insert 3

i	j
1	3
2	3
3	
4	
⋮	
n	

Time-Windowed Decision $\rightarrow$ Dynamic

	Dynamic DS	
	Yes	No
Property Satisfied:	✓	o

One way to build it:

- Use a dynamic DS
- Insert until satisfied
- Delete (from beginning) until not satisfied
- Repeat until all points deleted

Insert 1 Delete
Insert 2 Delete
Insert 3
Insert 4

i	j
1	3
2	3
3	
4	
⋮	
n	

Time-Windowed Decision $\rightarrow$ Dynamic

	Dynamic DS	
	Yes	No
Property Satisfied:	✓	o

One way to build it:

- Use a dynamic DS
- Insert until satisfied
- Delete (from beginning) until not satisfied
- Repeat until all points deleted

Insert 1 Delete
Insert 2 Delete
Insert 3
Insert 4

i	j
1	3
2	3
3	4
4	
⋮	
n	

Time-Windowed Decision $\rightarrow$ Dynamic

	Dynamic DS	
	Yes	No
Property Satisfied:	o	✓

One way to build it:

- Use a dynamic DS
- Insert until satisfied
- Delete (from beginning) until not satisfied
- Repeat until all points deleted

Insert 1 Delete
Insert 2 Delete
Insert 3 Delete
Insert 4

i	j
1	3
2	3
3	4
4	
⋮	
n	

Time-Windowed Decision $\rightarrow$ Dynamic

	Dynamic DS	
	Yes	No
Property Satisfied:	o	✓

One way to build it:

- Use a dynamic DS
- Insert until satisfied
- Delete (from beginning) until not satisfied
- Repeat until all points deleted

Insert 1 Delete
Insert 2 Delete
Insert 3 Delete
Insert 4

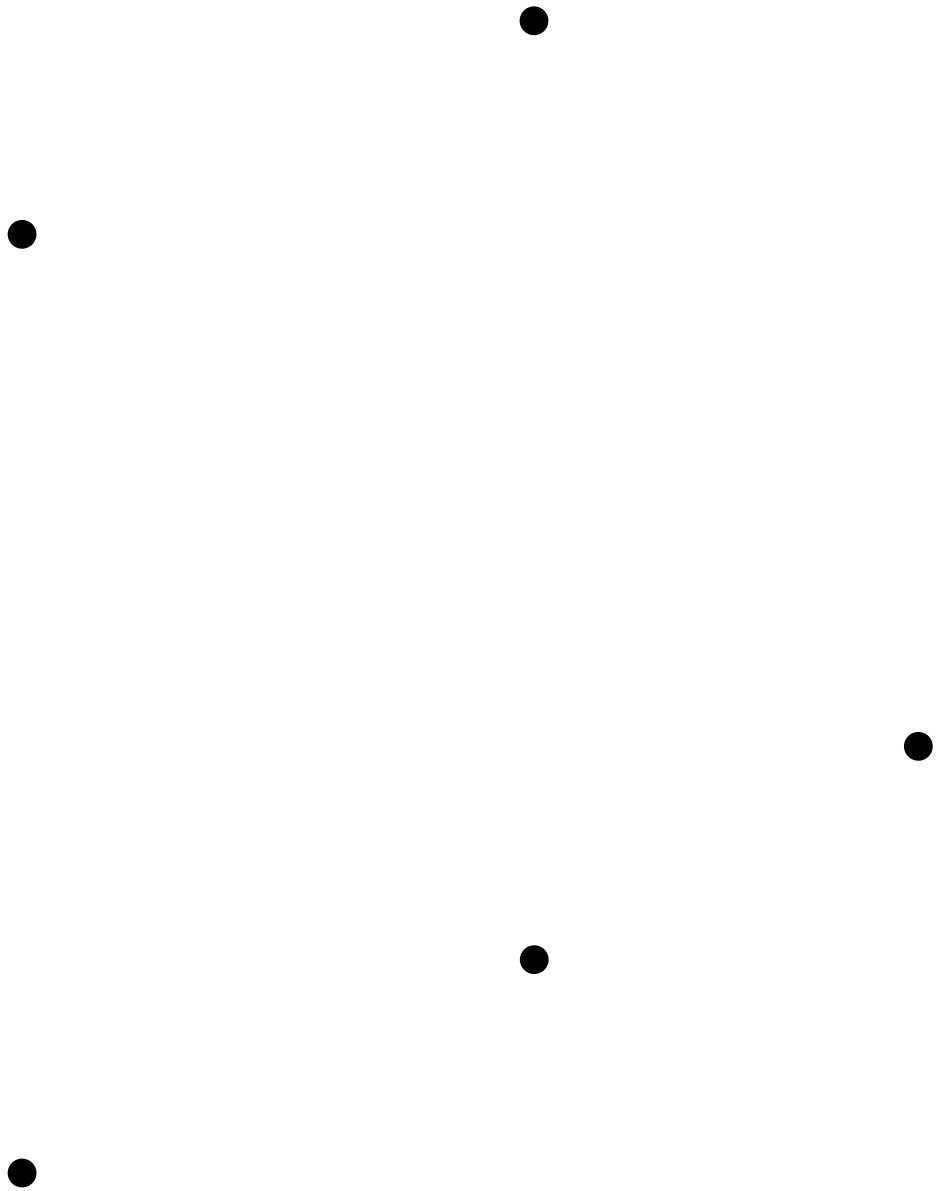
i	j
1	3
2	3
3	4
4	⋮
⋮	
n	

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?

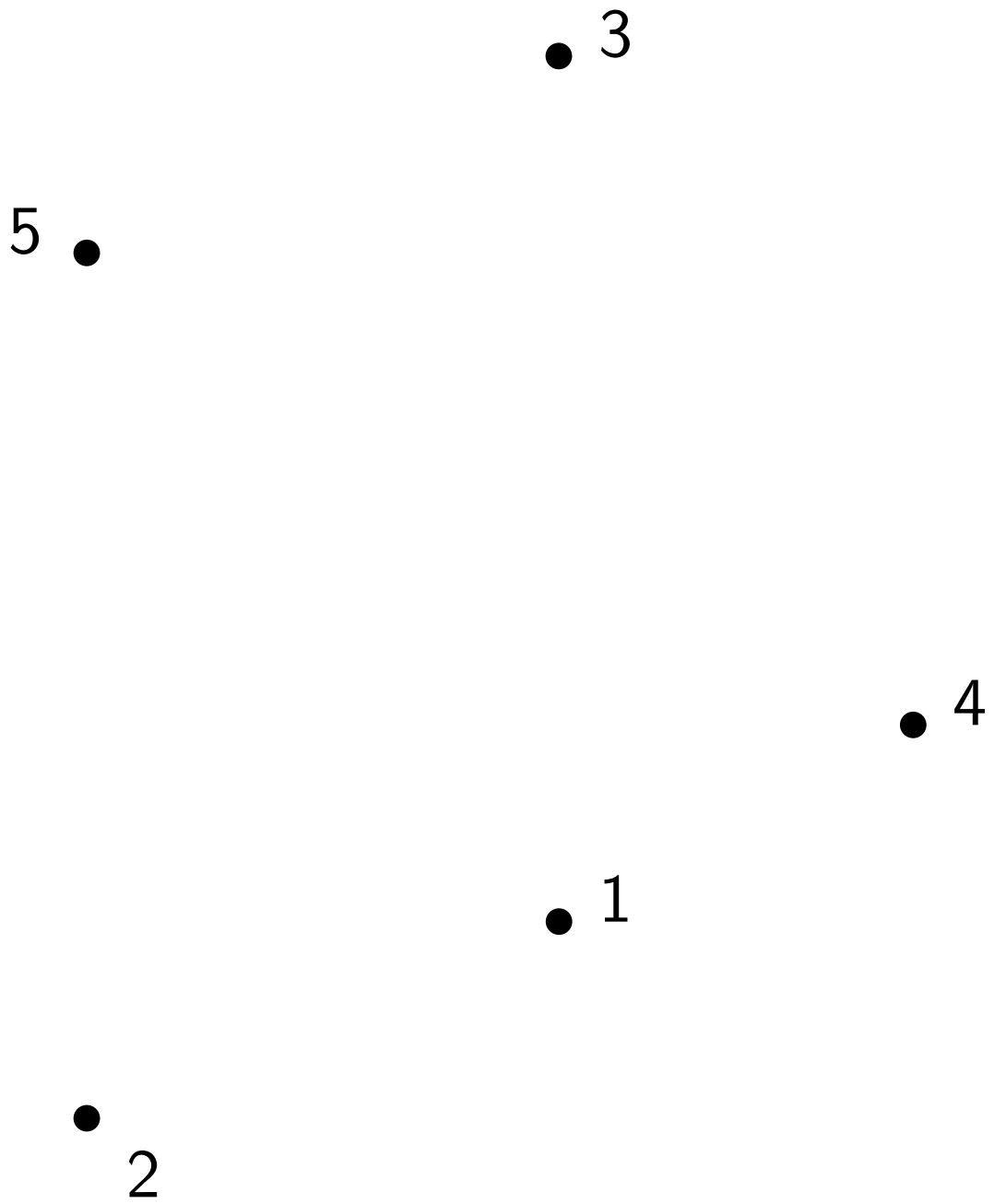
Time-Windowed $\rightarrow$ Dynamic

CH area > 1?



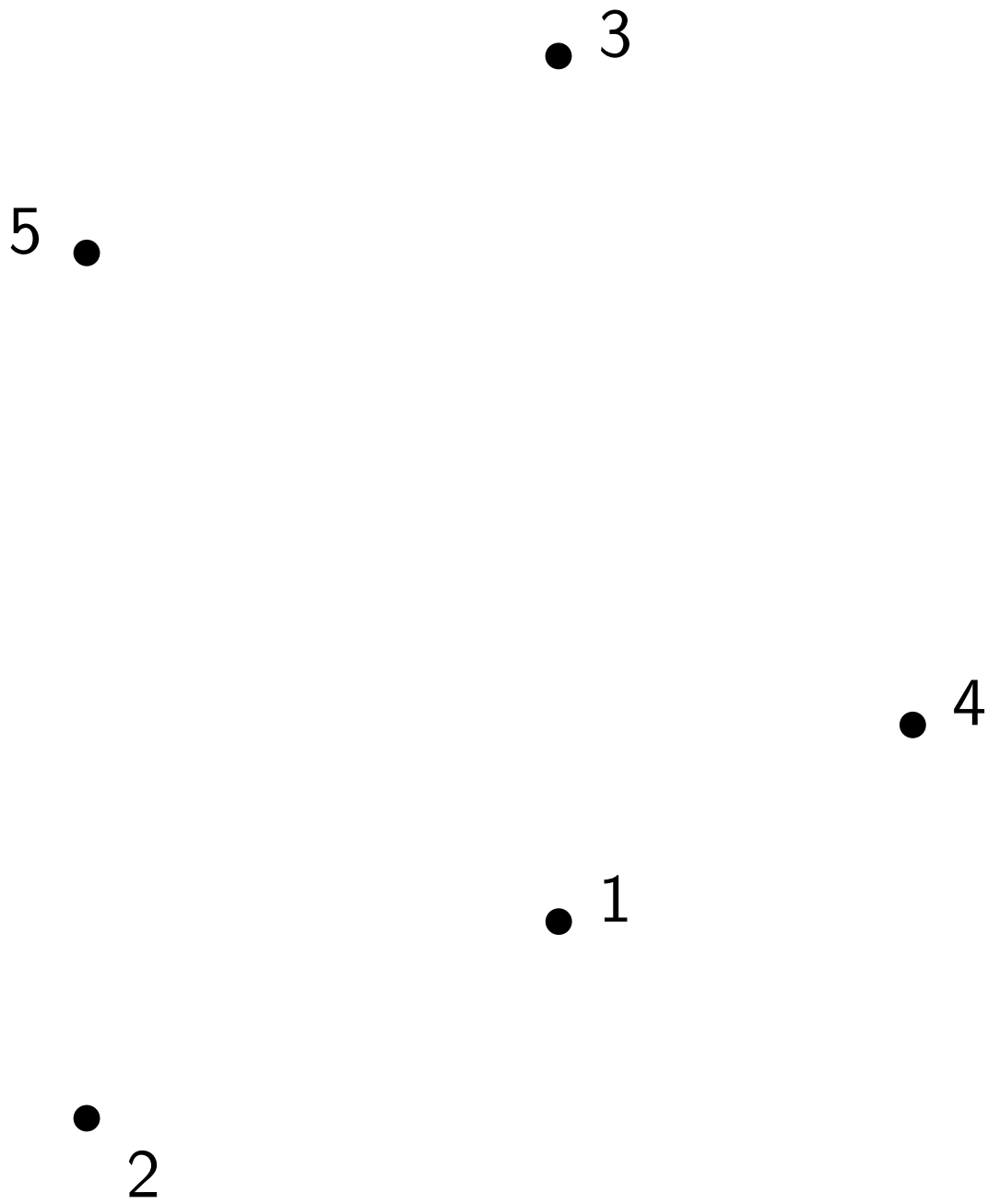
Time-Windowed $\rightarrow$ Dynamic

CH area > 1?



Time-Windowed $\rightarrow$ Dynamic

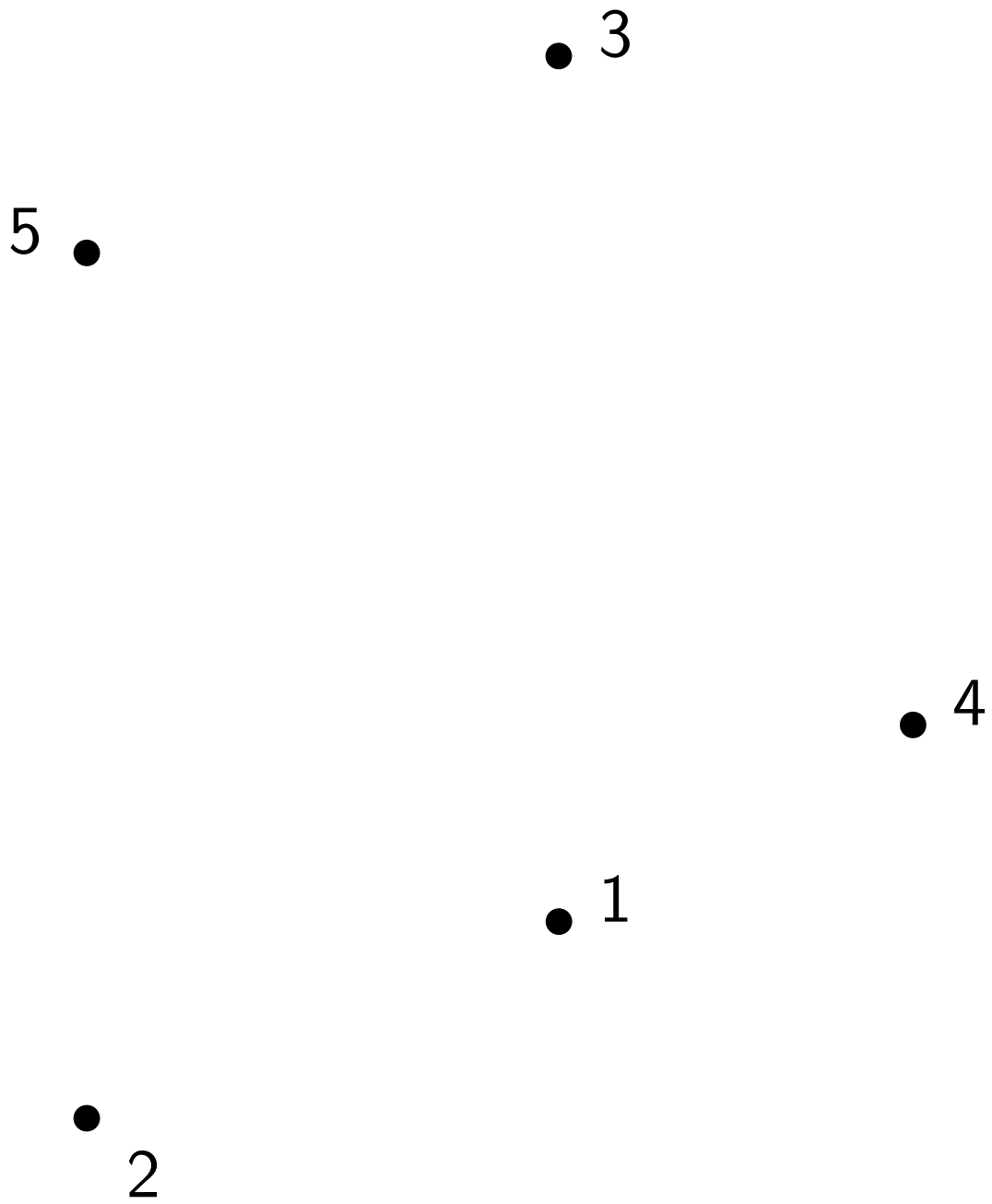
CH area > 1?



i
1
2
3
4
5

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?

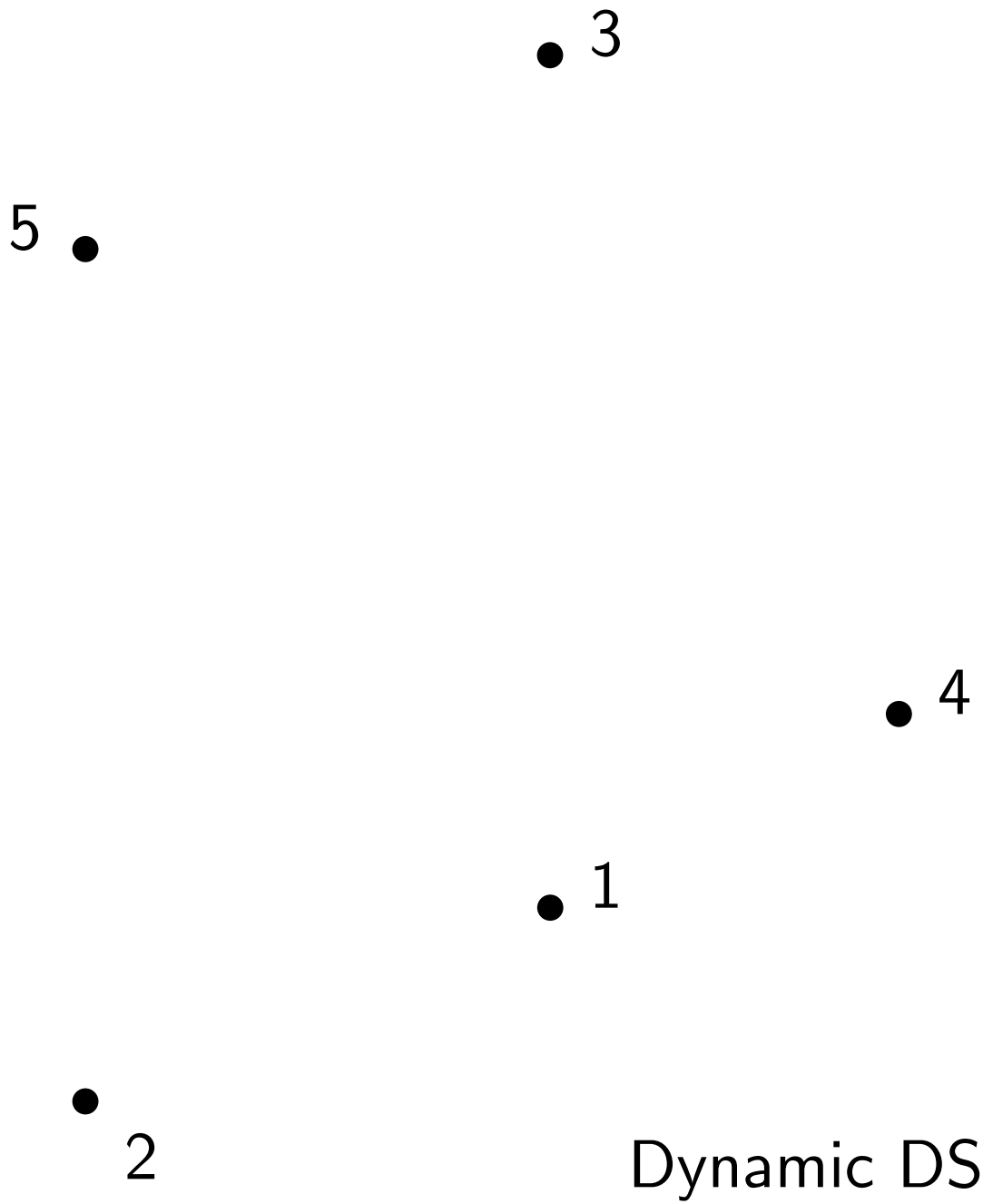


i	j
1	
2	
3	
4	
5	

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?

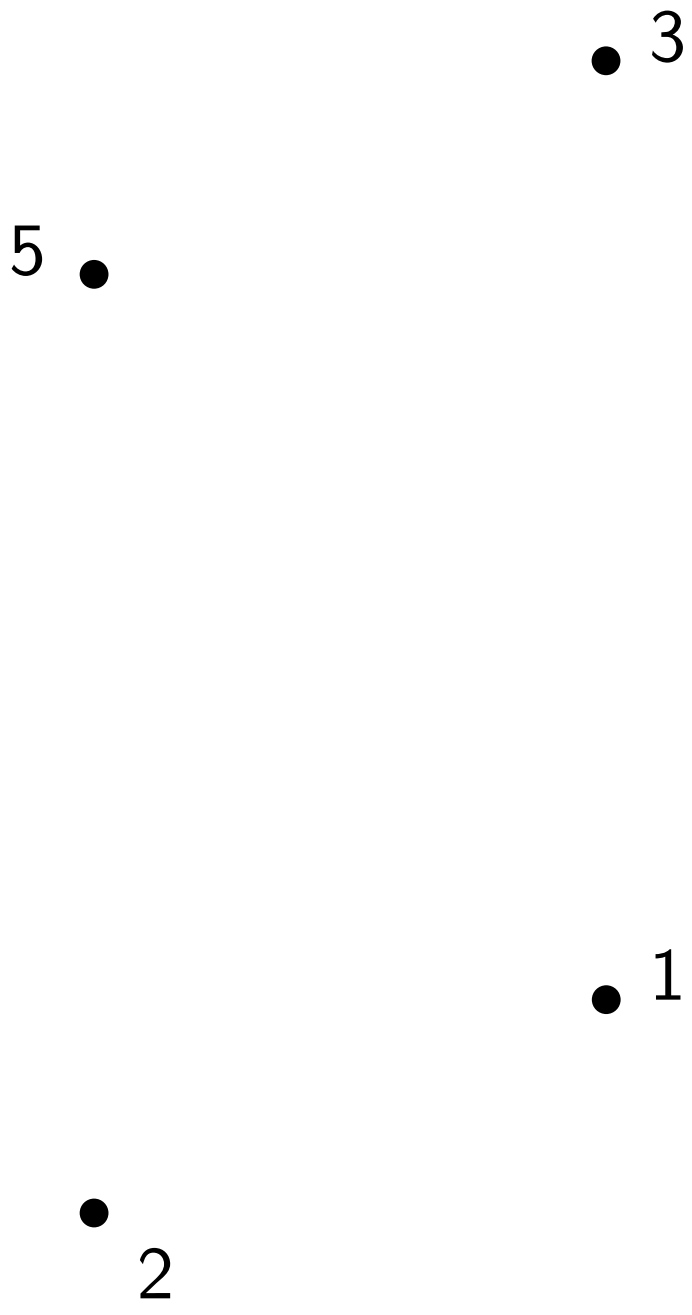
i	j
1	
2	
3	
4	
5	



Dynamic DS

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?



i	j
1	
2	
3	
4	
5	

Dynamic DS : Has property? No

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?

i	j
1	
2	
3	
4	
5	

Dynamic DS : Has property? No

Time-Windowed → Dynamic

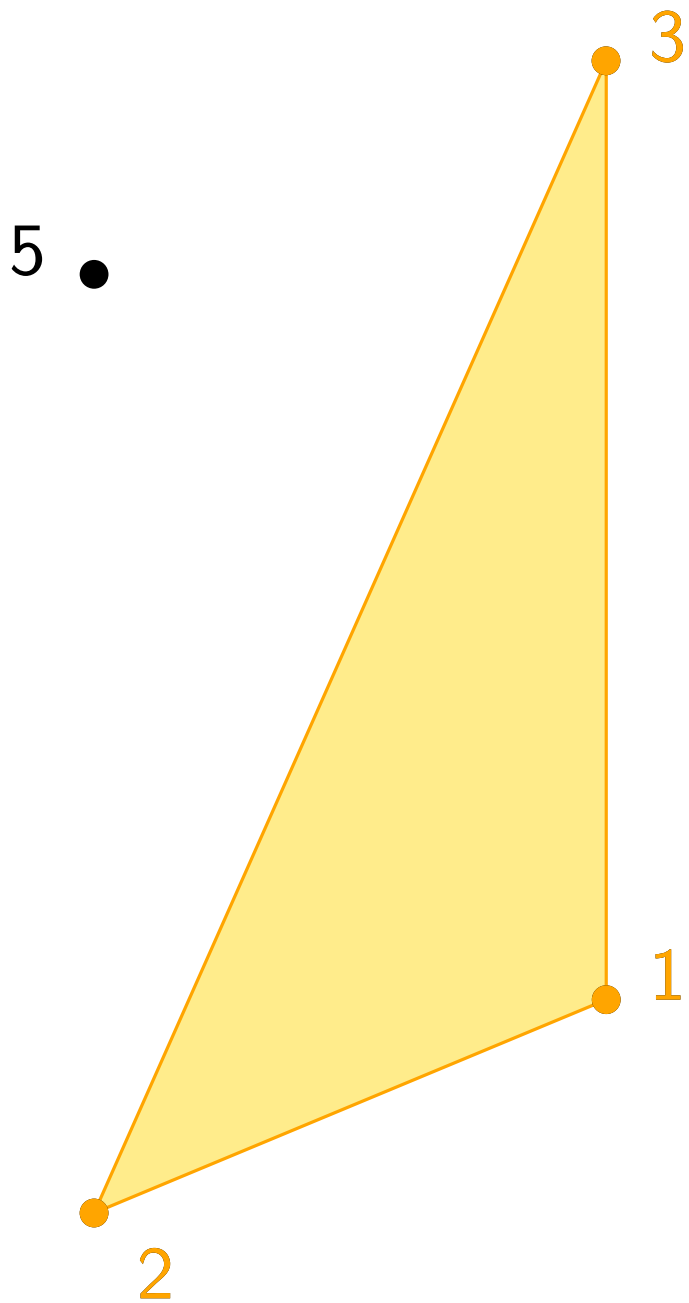
CH area > 1?

i	j
1	
2	
3	
4	
5	

Dynamic DS : Has property? No

Time-Windowed → Dynamic

CH area > 1?

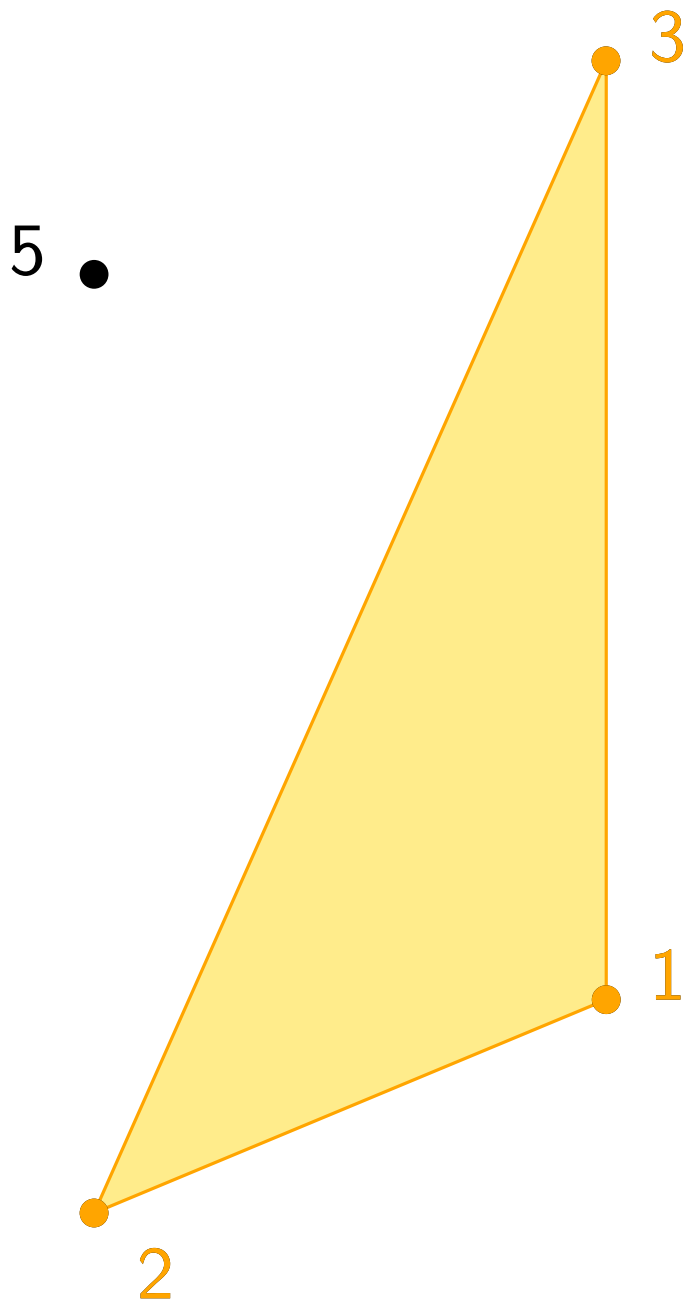


i	j
1	
2	
3	
4	
5	

Dynamic DS : Has property? Yes

Time-Windowed → Dynamic

CH area > 1?

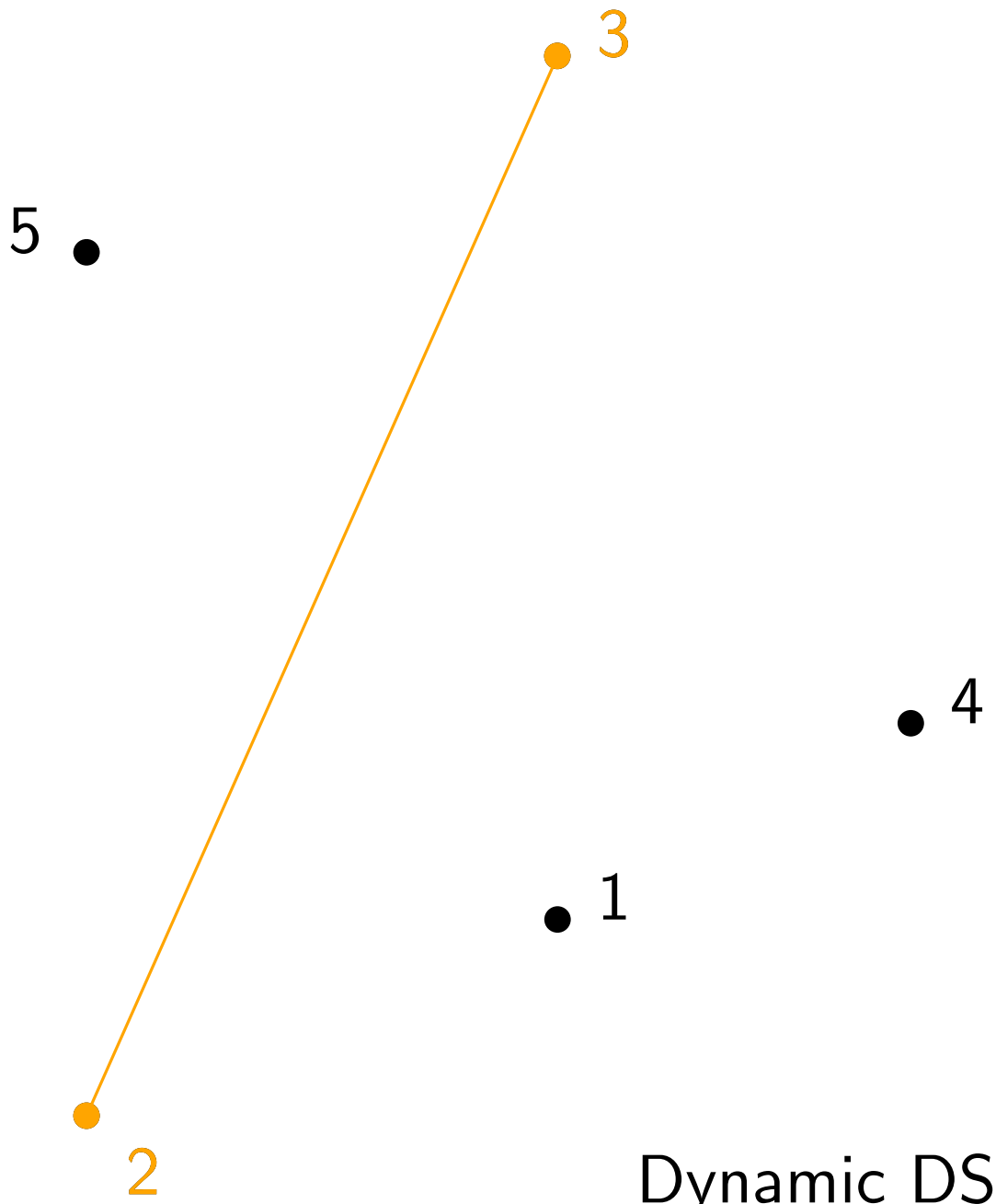


i	j
1	3
2	
3	
4	
5	

Dynamic DS : Has property? Yes

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?

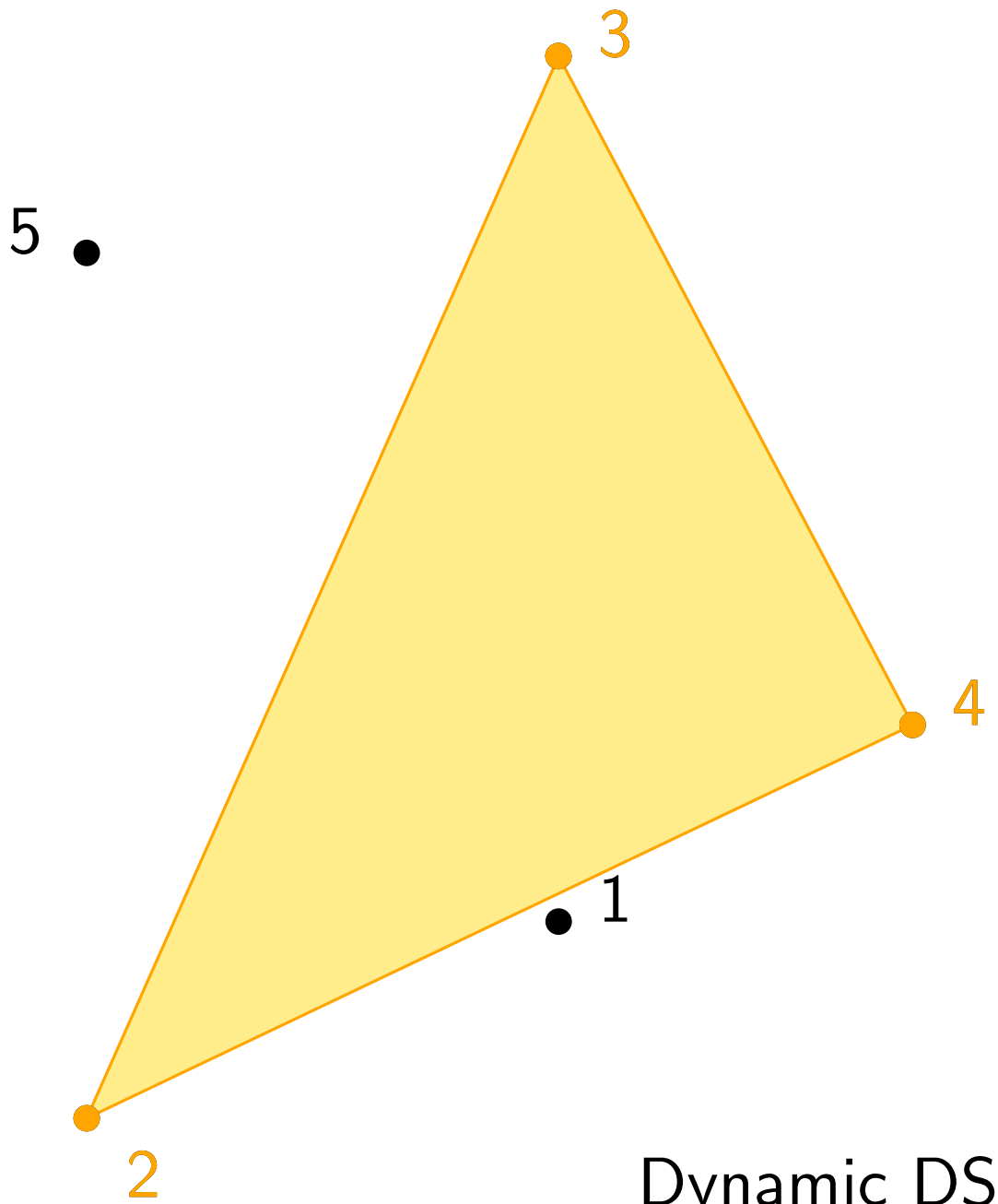


i	j
1	3
2	
3	
4	
5	

Dynamic DS : Has property? No

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?

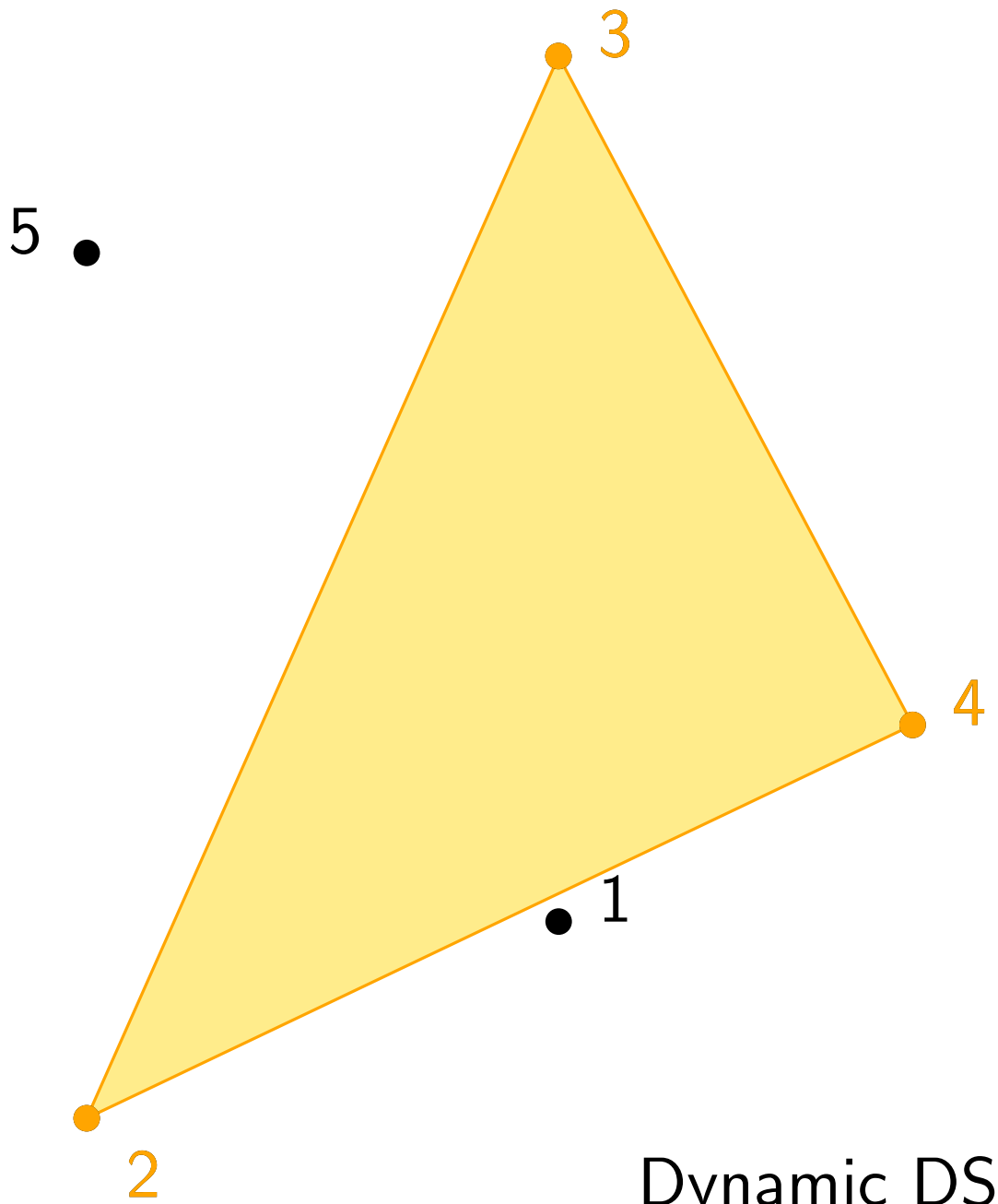


i	j
1	3
2	
3	
4	
5	

Dynamic DS : Has property? Yes

Time-Windowed → Dynamic

CH area > 1?

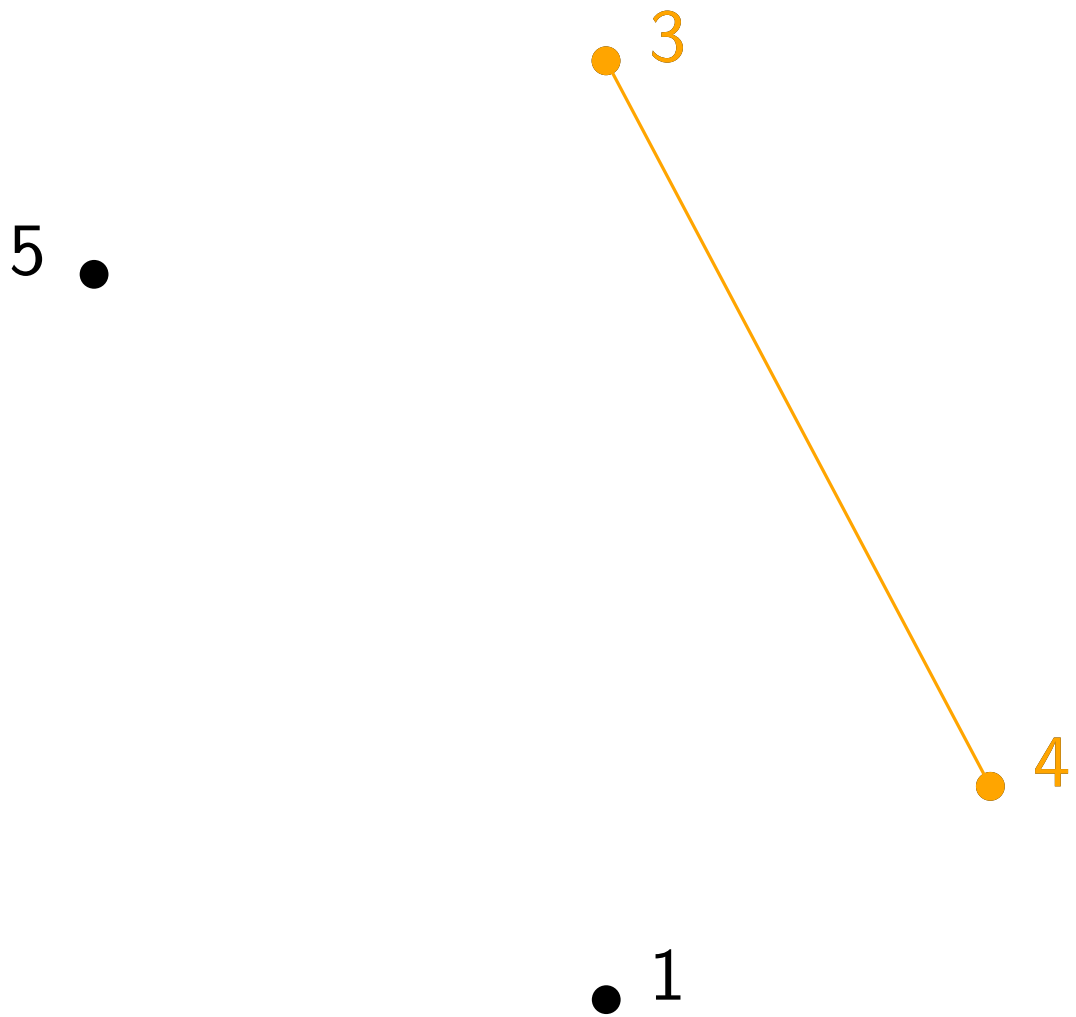


i	j
1	3
2	4
3	
4	
5	

Dynamic DS : Has property? Yes

Time-Windowed → Dynamic

CH area > 1?

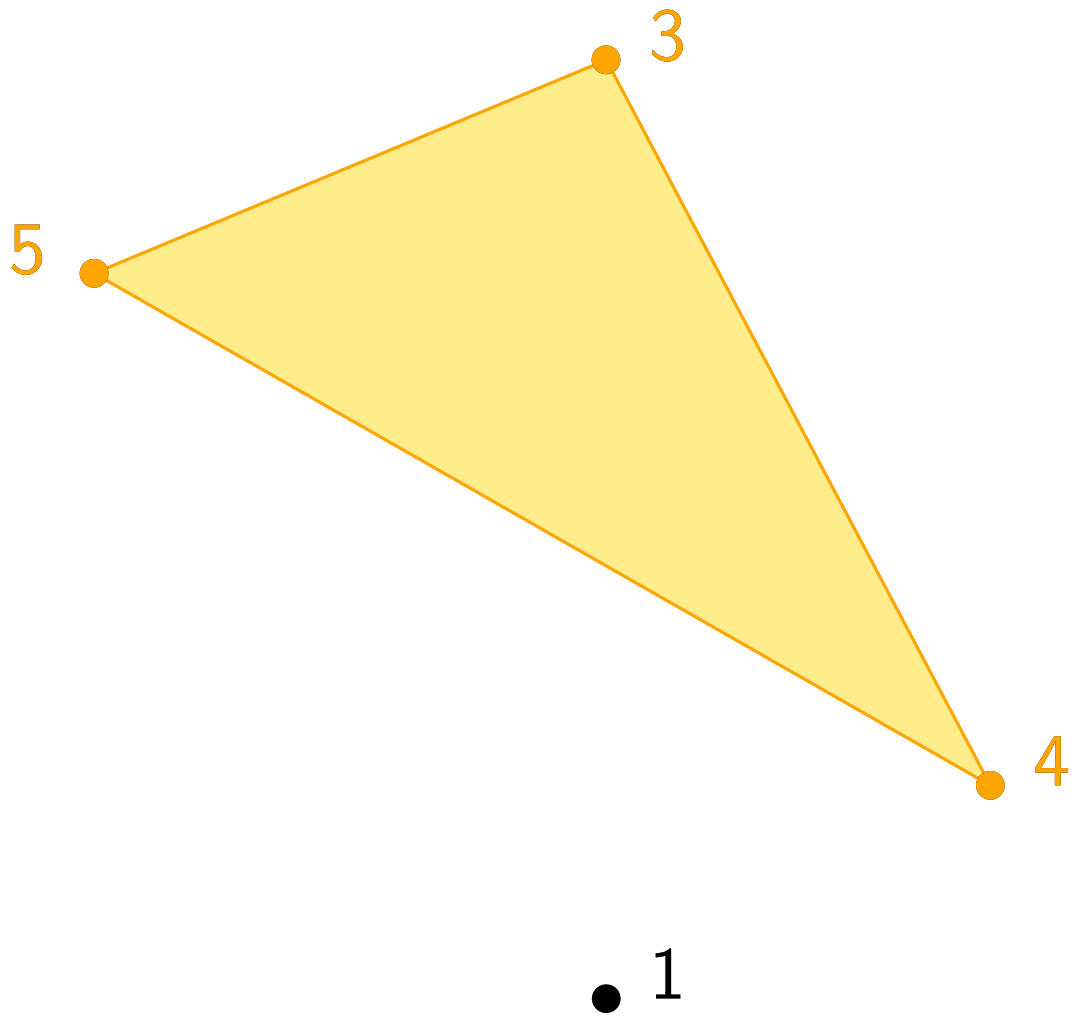


i	j
1	3
2	4
3	
4	
5	

Dynamic DS : Has property? No

Time-Windowed → Dynamic

CH area > 1?

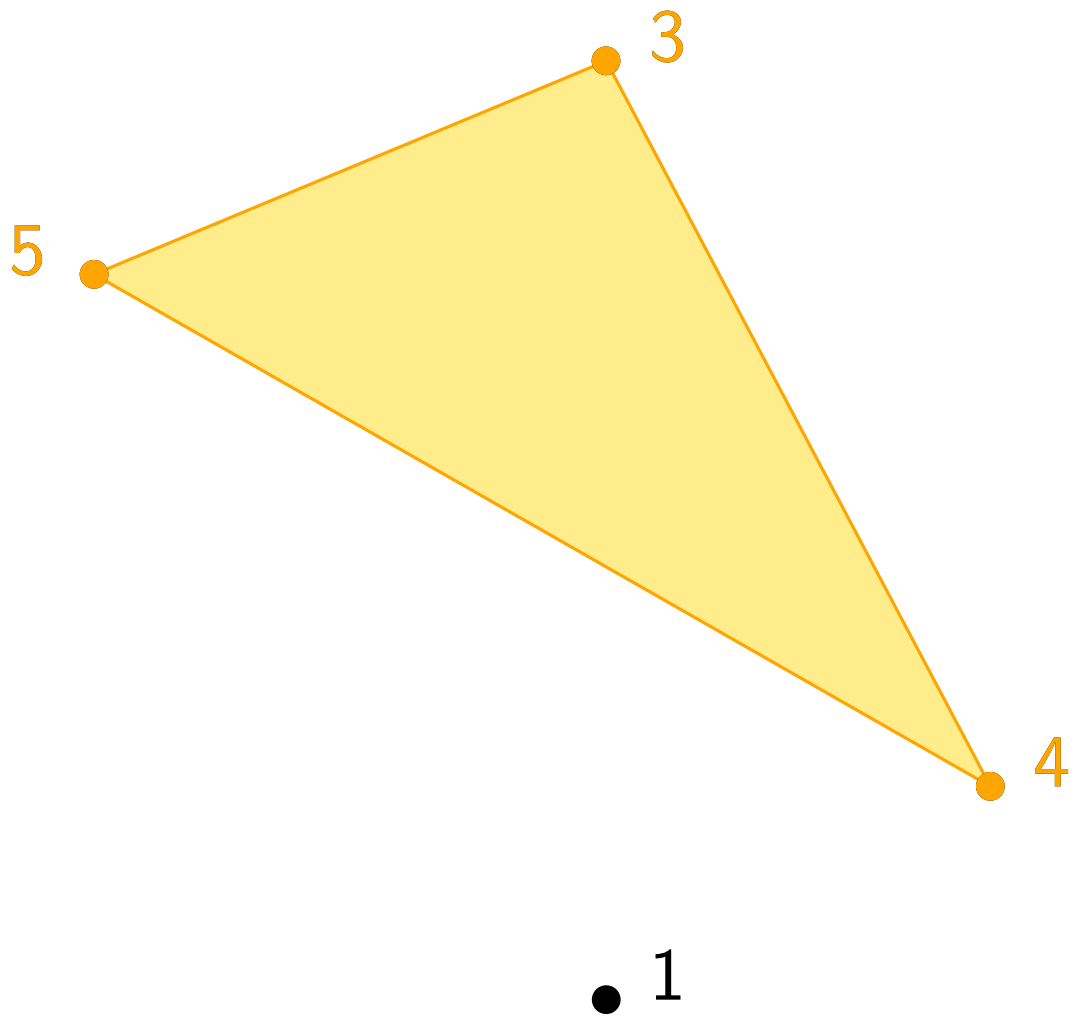


i	j
1	3
2	4
3	
4	
5	

Dynamic DS : Has property? Yes

Time-Windowed → Dynamic

CH area > 1?

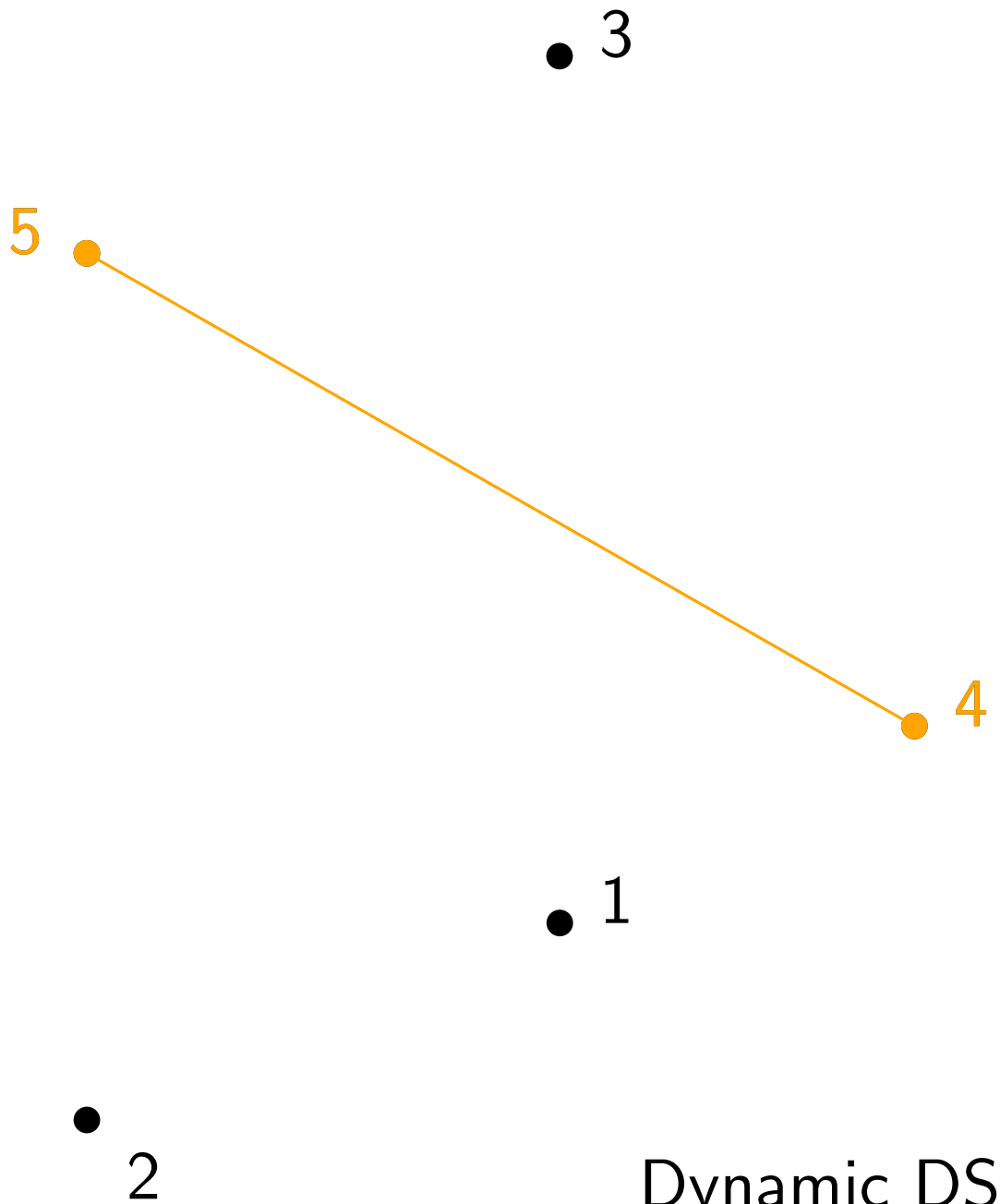


i	j
1	3
2	4
3	5
4	
5	

Dynamic DS : Has property? Yes

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?

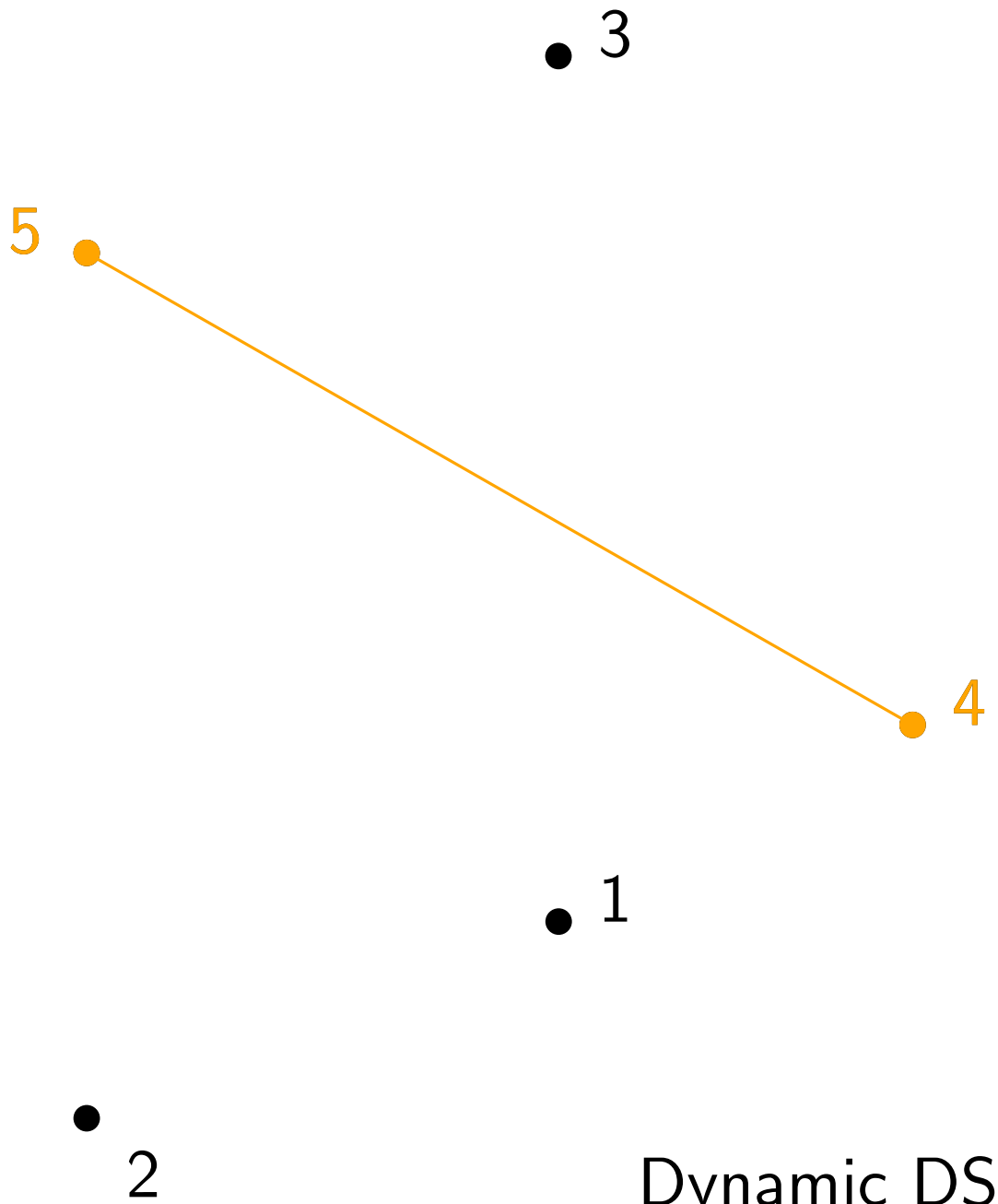


i	j
1	3
2	4
3	5
4	
5	

Dynamic DS : Has property? No

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?

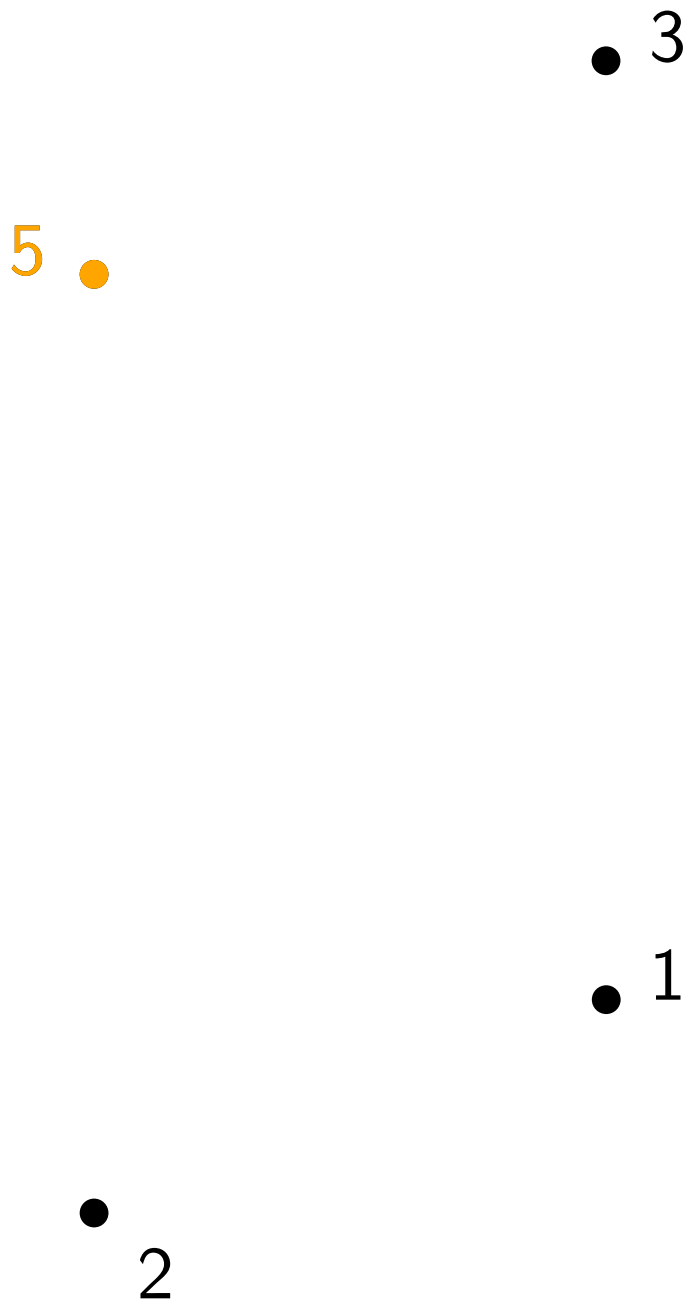


i	j
1	3
2	4
3	5
4	∞
5	

Dynamic DS : Has property? No

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?

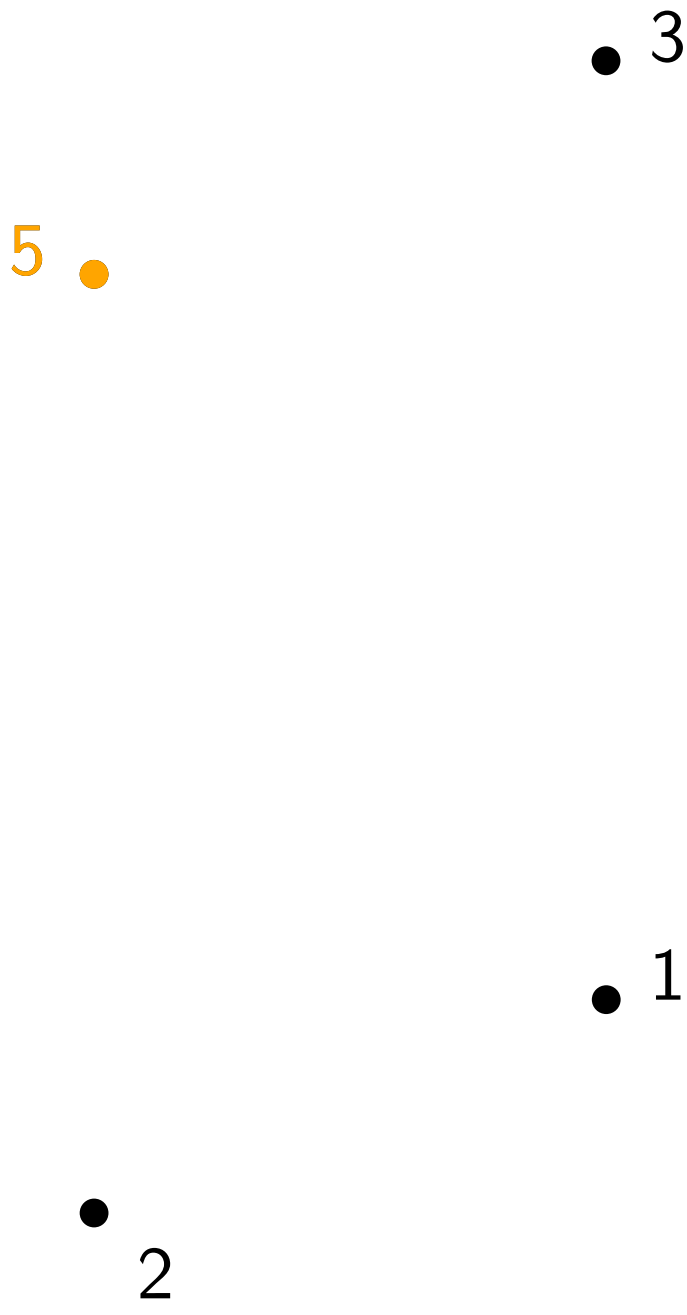


i	j
1	3
2	4
3	5
4	∞
5	

Dynamic DS : Has property? No

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?

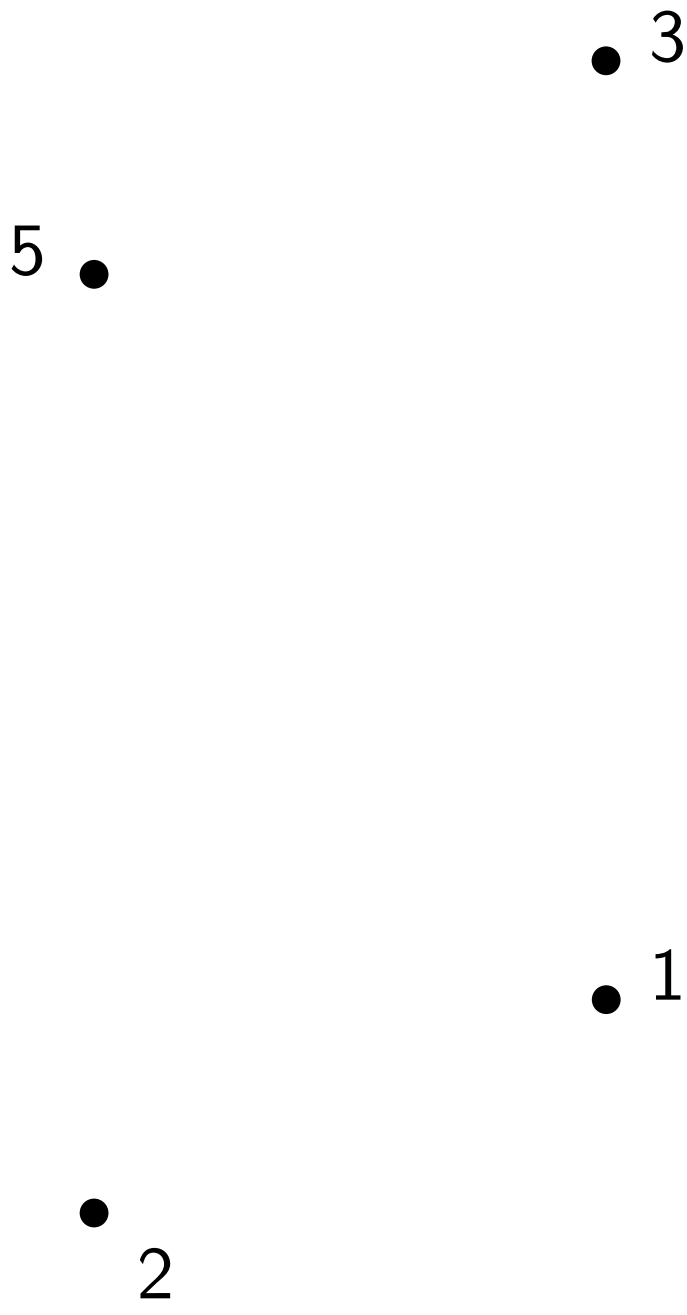


i	j
1	3
2	4
3	5
4	∞
5	∞

Dynamic DS : Has property? No

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?

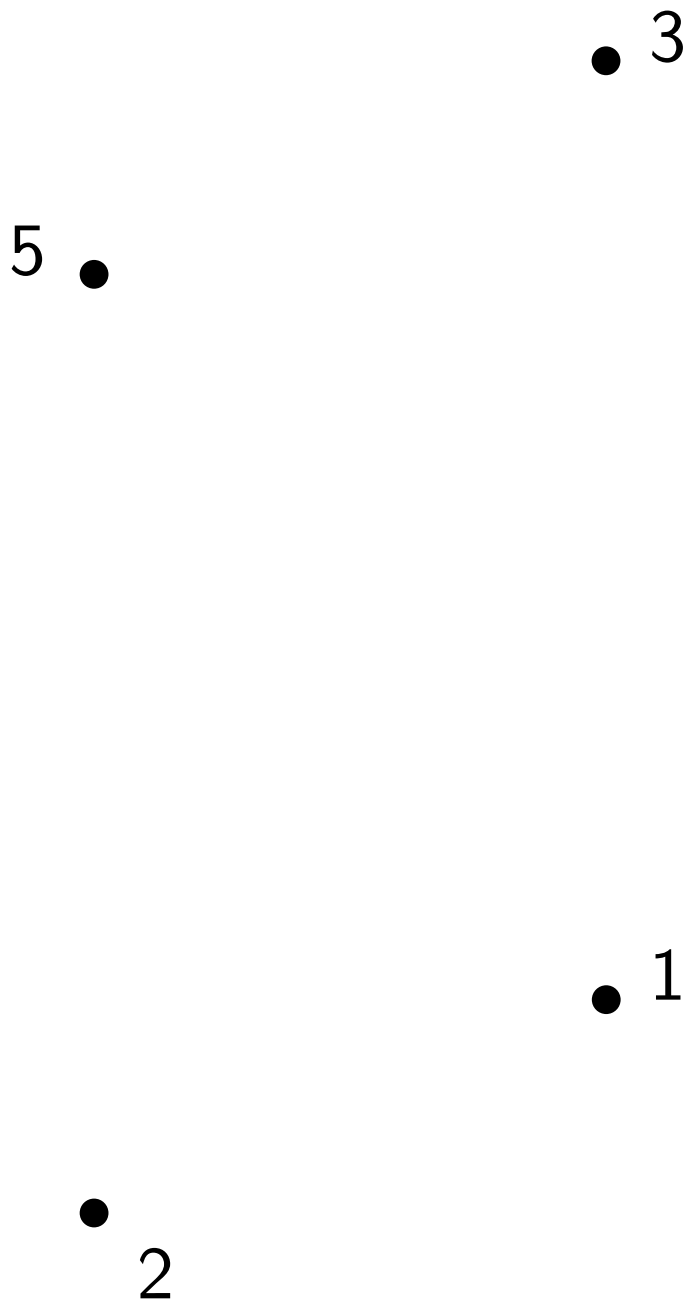


i	j
1	3
2	4
3	5
4	∞
5	∞

Dynamic DS : Has property? No

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?



Notice the order of insertion/deletion

i	j
1	3
2	4
3	5
4	∞
5	∞

Dynamic DS : Has property? No

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?

Notice the order of insertion/deletion

i	j
1	3
2	4
3	5
4	∞
5	∞

Dynamic DS : Has property? No

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?

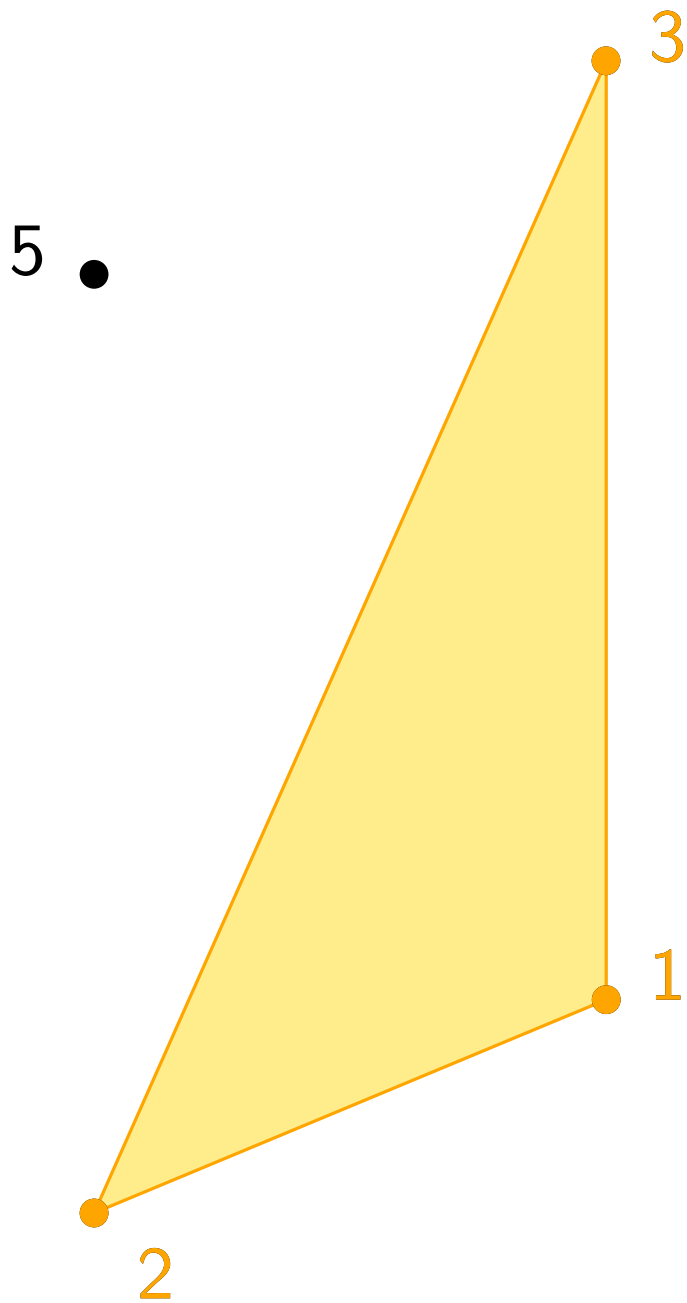
Notice the order of insertion/deletion

i	j
1	3
2	4
3	5
4	∞
5	∞

Dynamic DS : Has property? No

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?



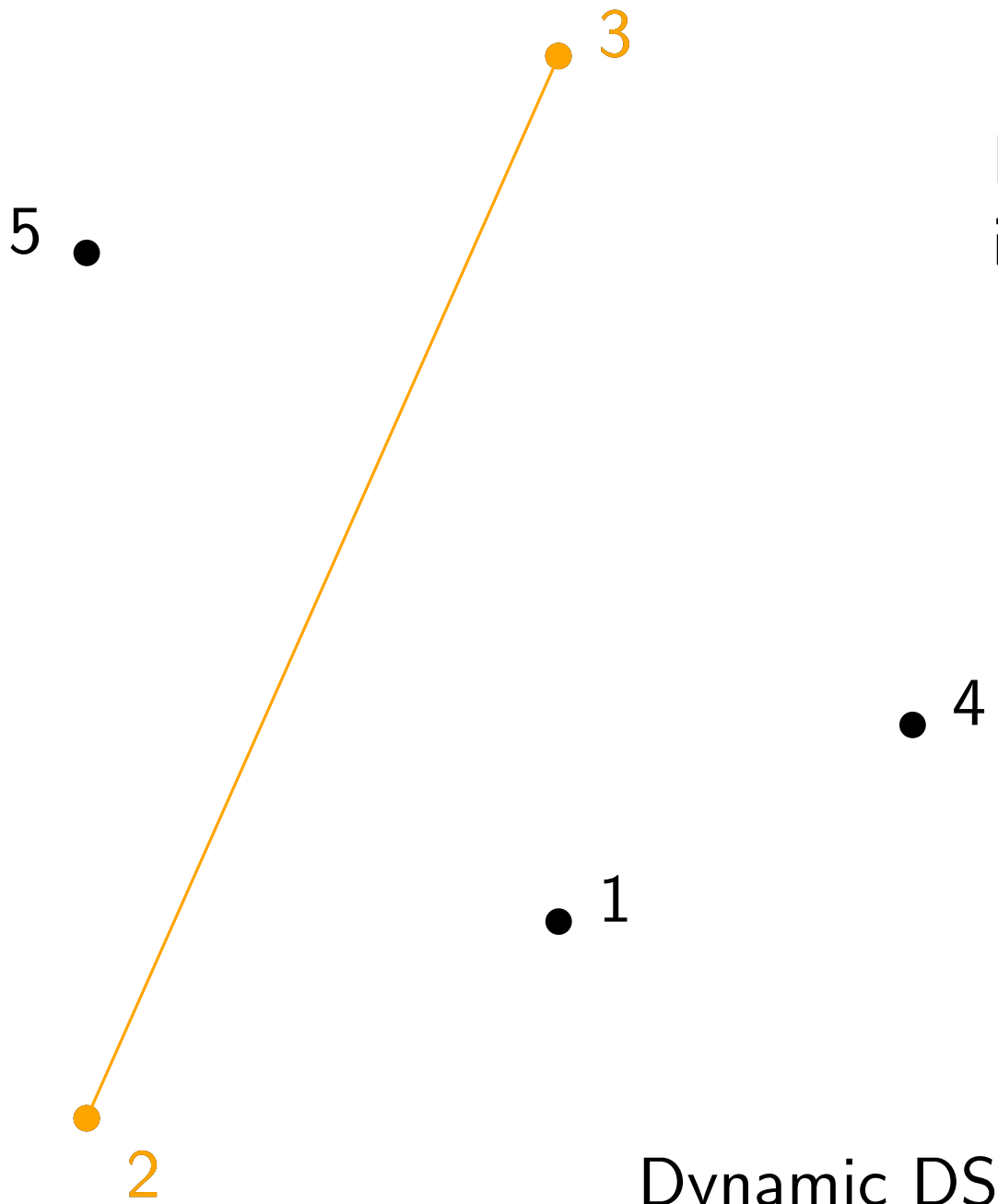
Notice the order of insertion/deletion

i	j
1	3
2	4
3	5
4	∞
5	∞

Dynamic DS : Has property? No

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?



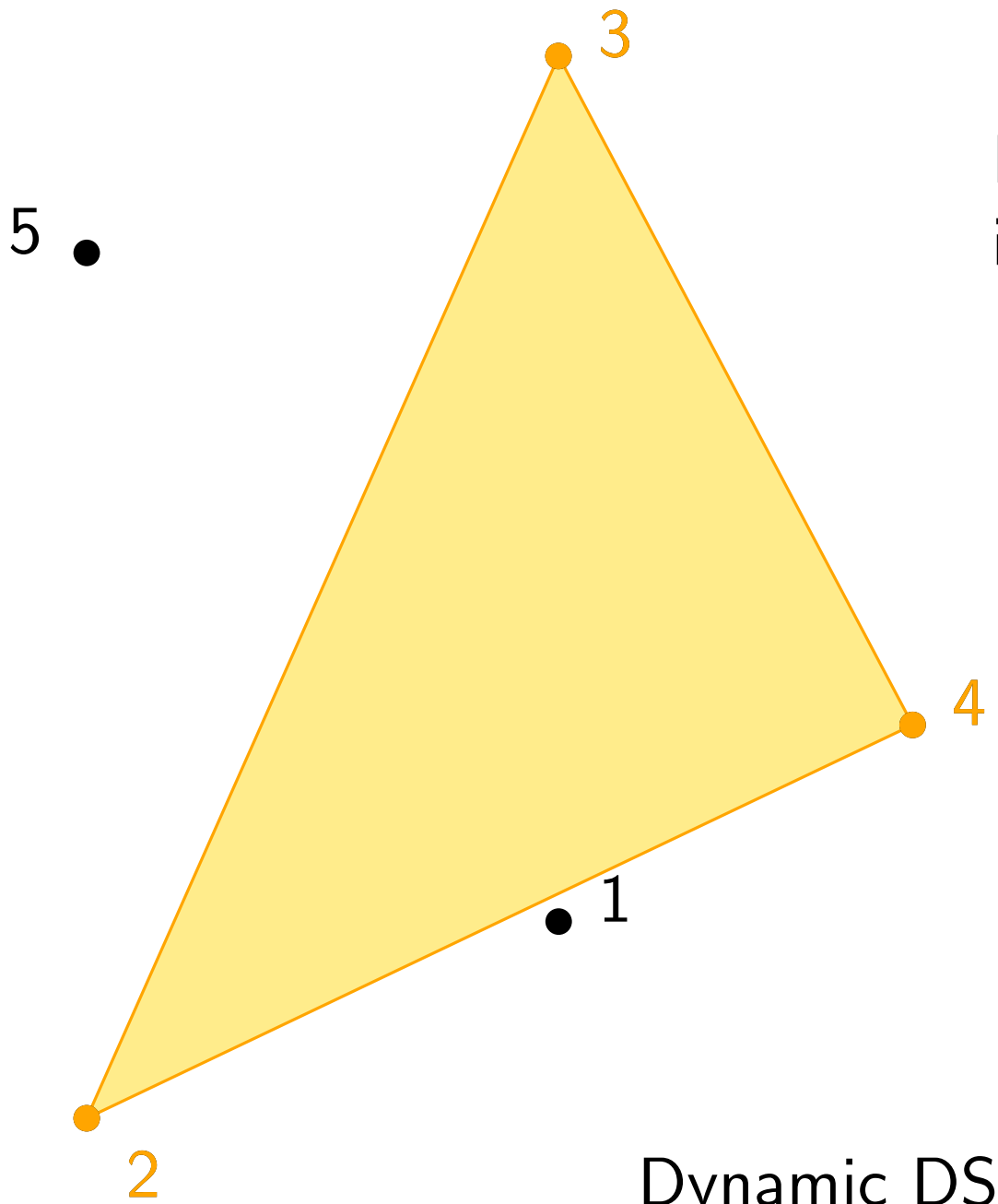
Notice the order of insertion/deletion

i	j
1	3
2	4
3	5
4	∞
5	∞

Dynamic DS : Has property? No

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?



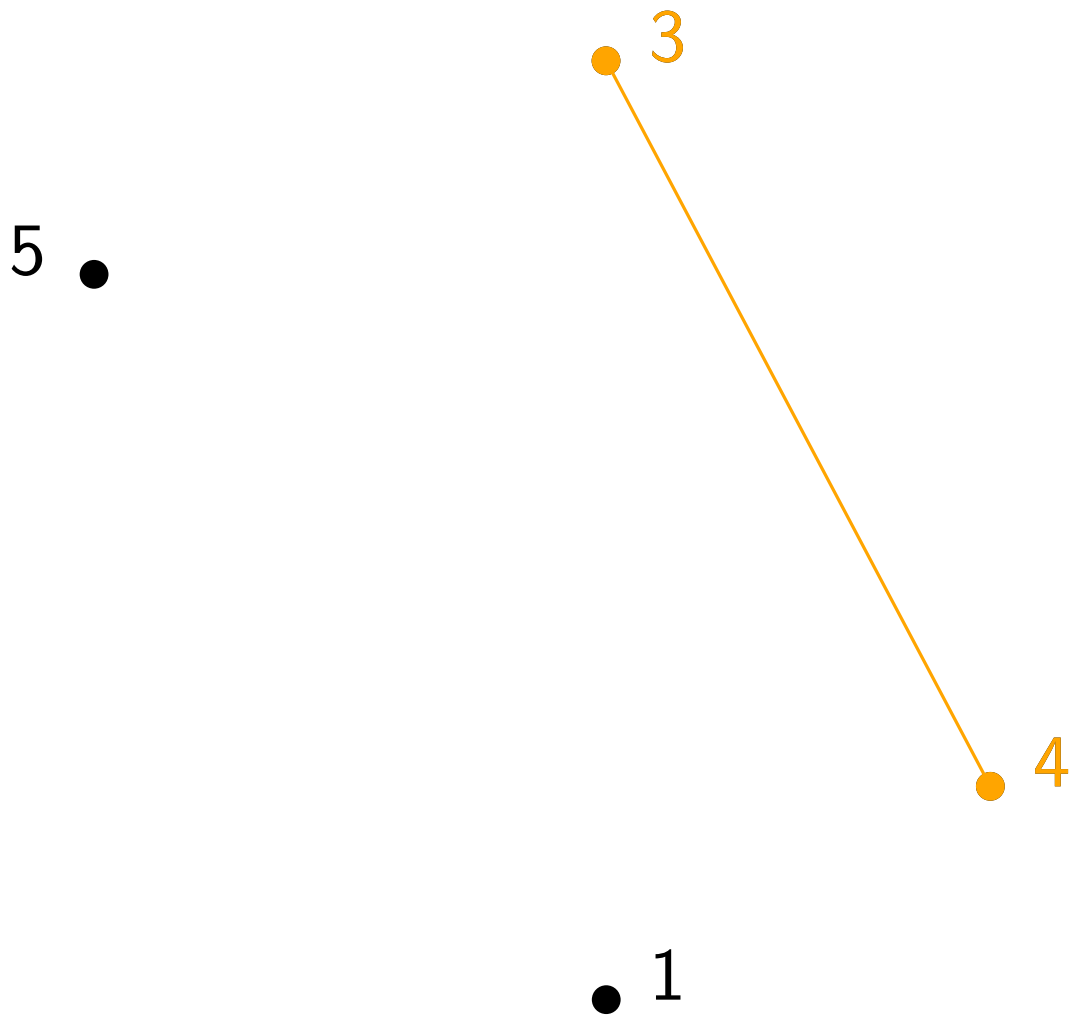
Notice the order of insertion/deletion

i	j
1	3
2	4
3	5
4	∞
5	∞

Dynamic DS : Has property? No

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?



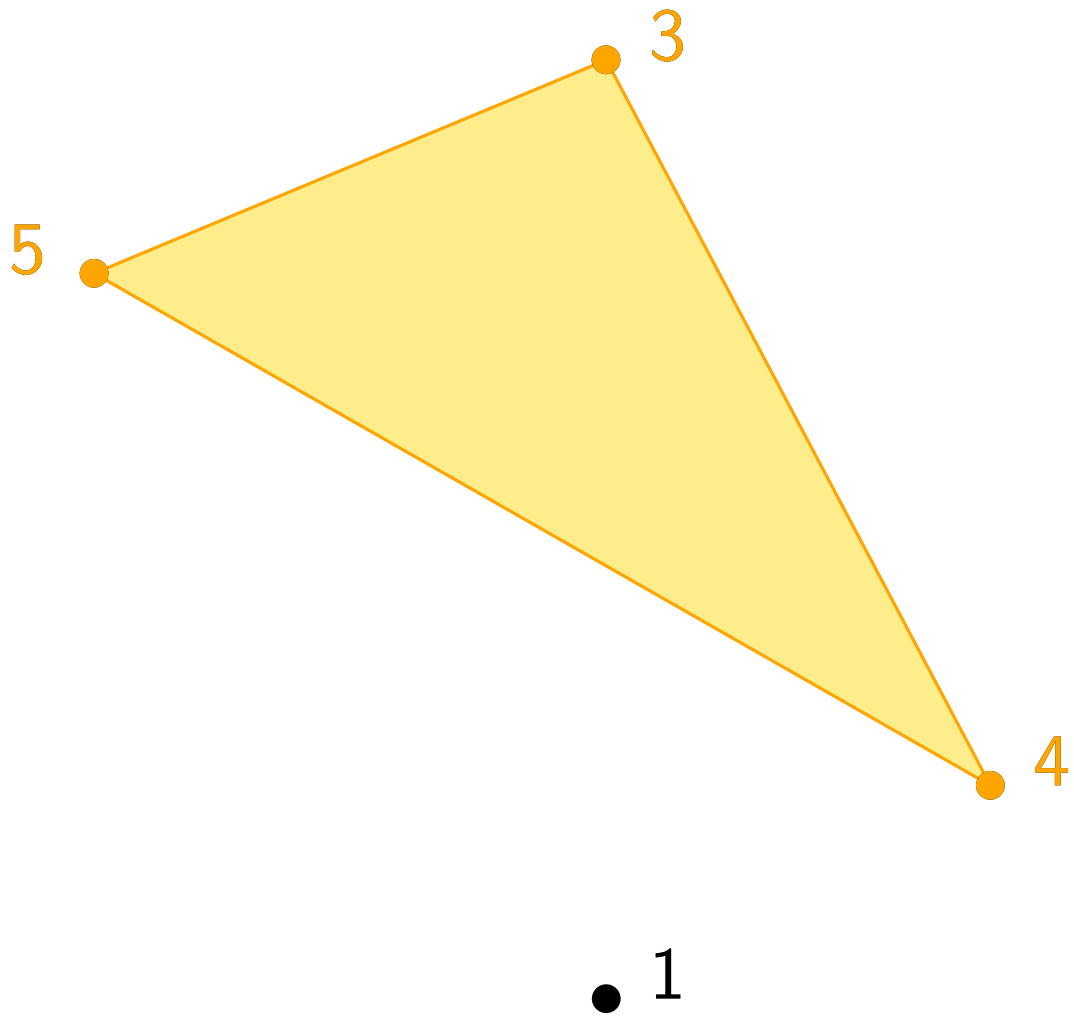
Notice the order of insertion/deletion

i	j
1	3
2	4
3	5
4	∞
5	∞

Dynamic DS : Has property? No

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?



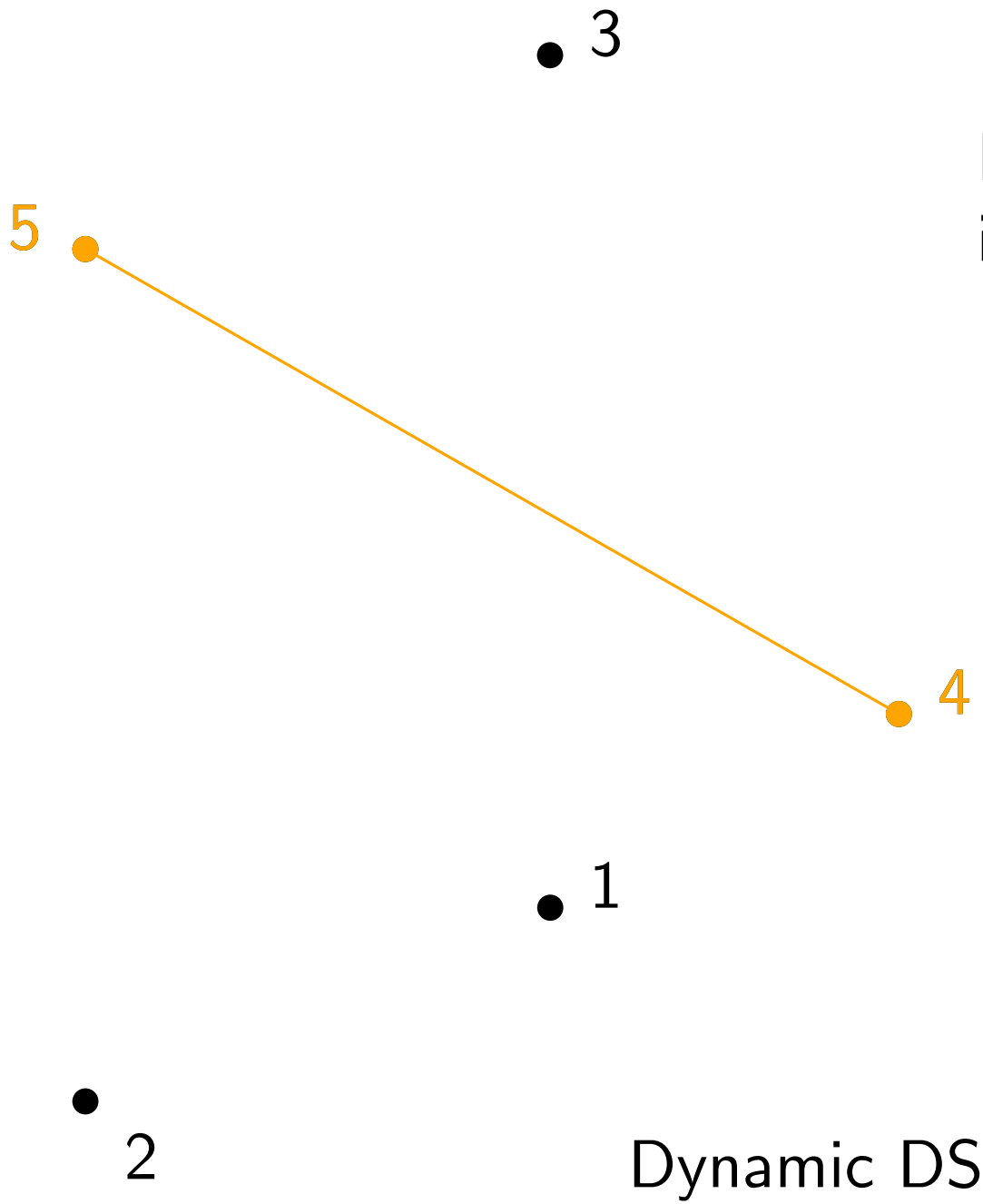
Notice the order of insertion/deletion

i	j
1	3
2	4
3	5
4	∞
5	∞

Dynamic DS : Has property? No

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?



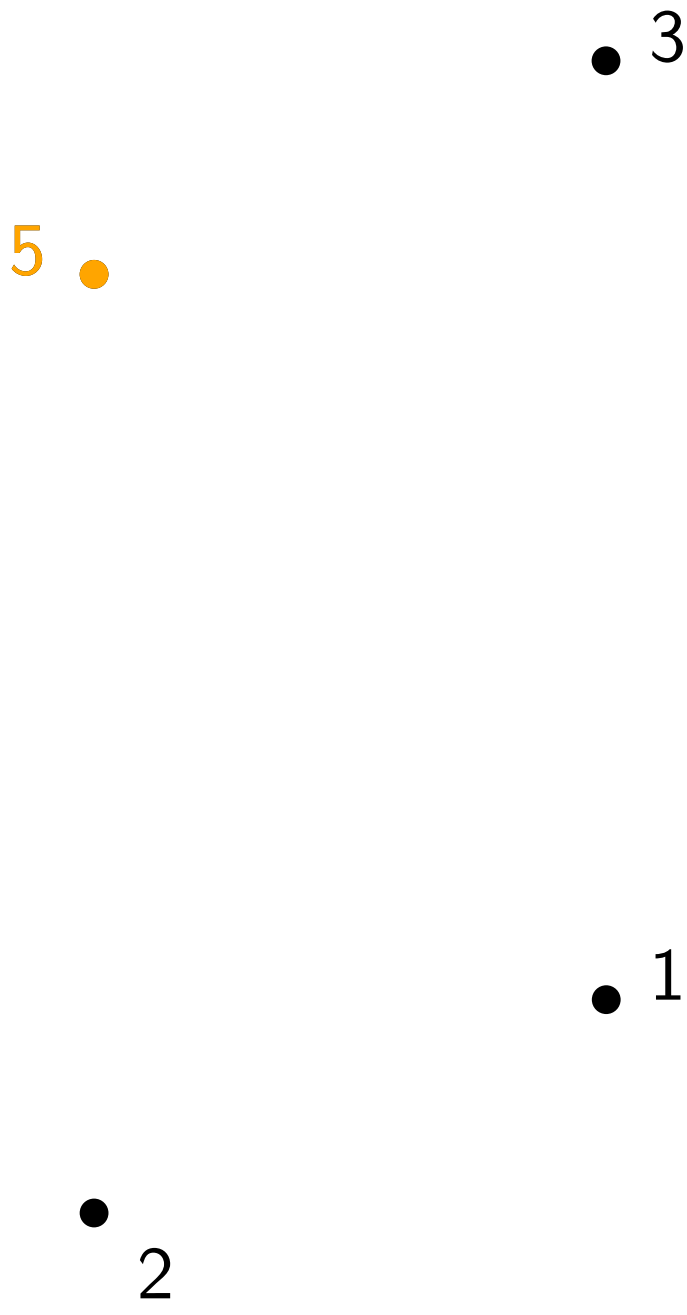
Notice the order of insertion/deletion

i	j
1	3
2	4
3	5
4	∞
5	∞

Dynamic DS : Has property? No

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?



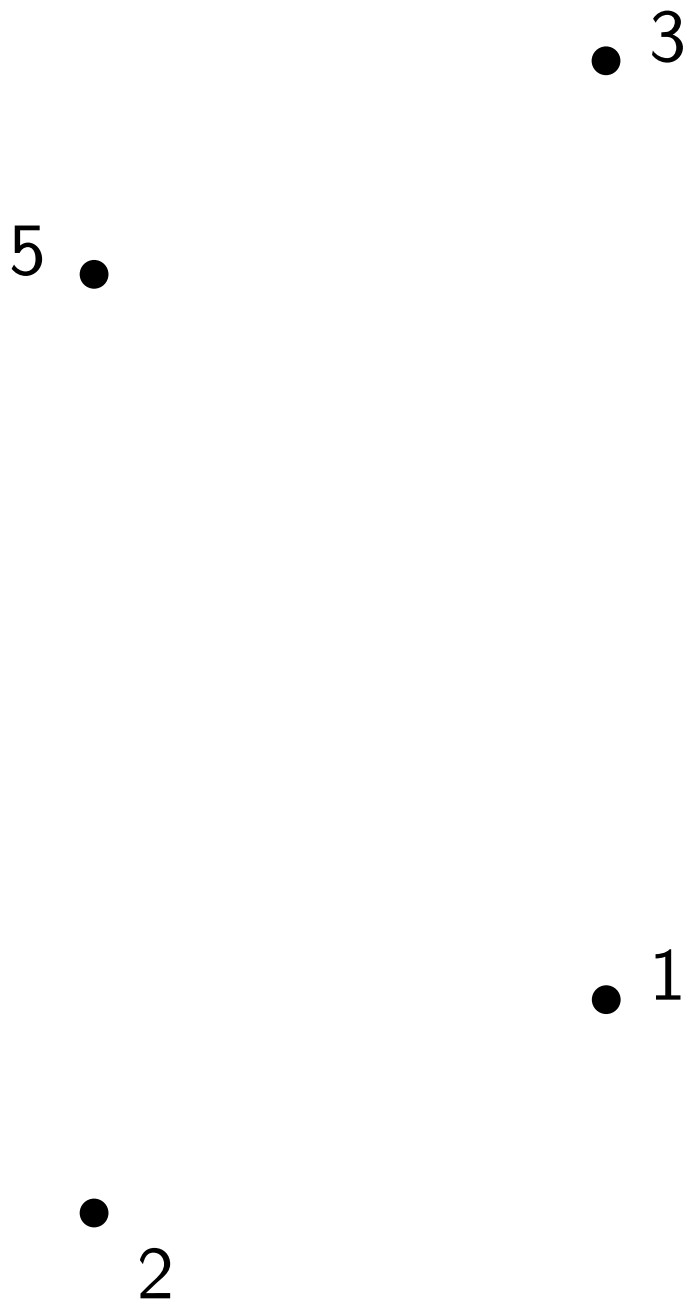
Notice the order of insertion/deletion

i	j
1	3
2	4
3	5
4	∞
5	∞

Dynamic DS : Has property? No

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?



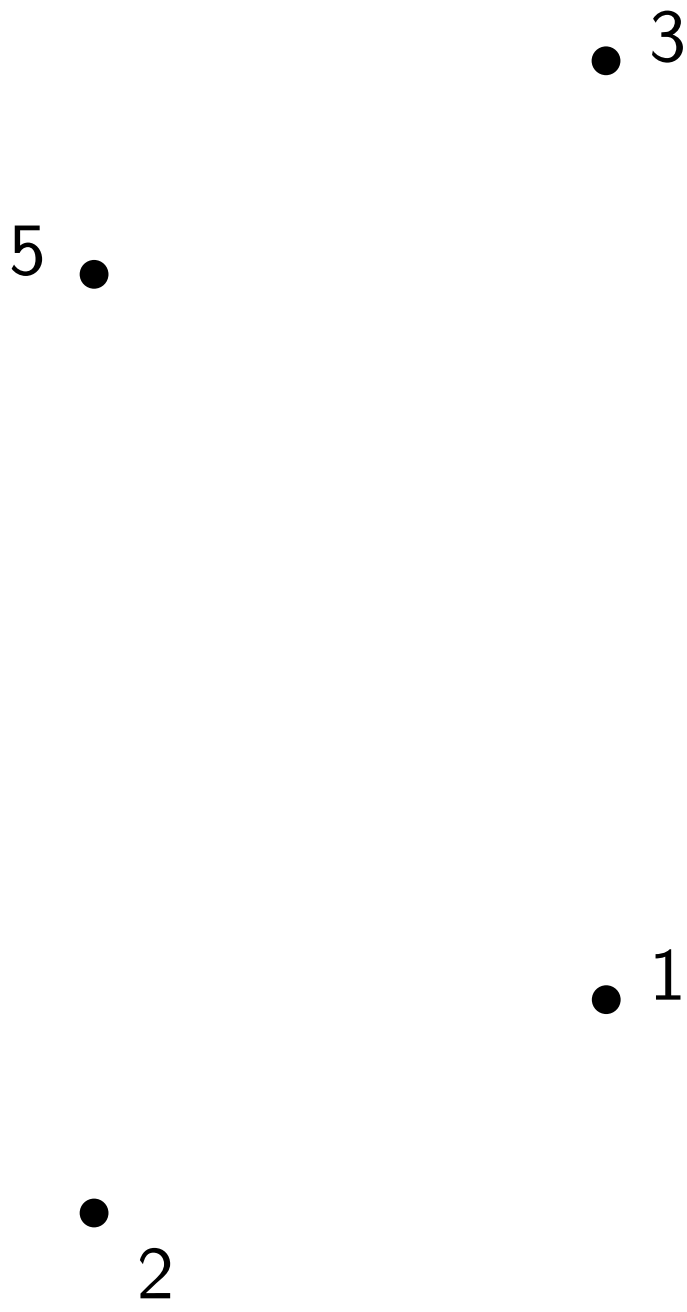
Notice the order of insertion/deletion

i	j
1	3
2	4
3	5
4	∞
5	∞

Dynamic DS : Has property? No

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?



Notice the order of insertion/deletion

First-in, first-out: like a queue

i	j
1	3
2	4
3	5
4	∞
5	∞

Dynamic DS : Has property? No

Time-Windowed → Dynamic

CH area > 1?

Notice the order of insertion/deletion

First-in, first-out: like a queue

FIFO update sequence

i	j
1	3
2	4
3	5
4	∞
5	∞

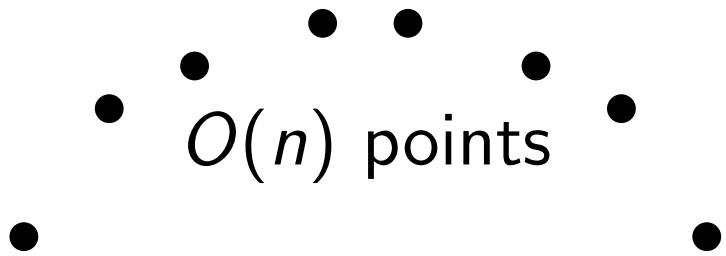
Dynamic DS : Has property? No

Dynamic Convex Hull Complexity

Generally, $O(n^2)$ changes over $O(n)$ updates

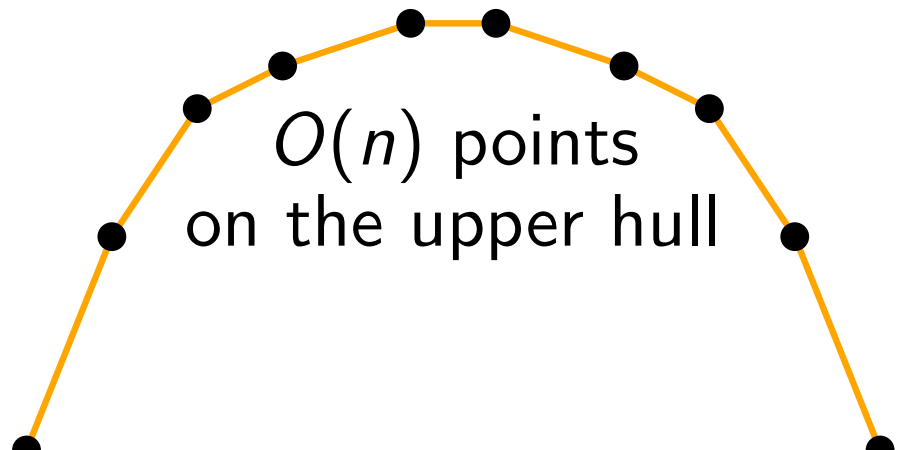
Dynamic Convex Hull Complexity

Generally, $O(n^2)$ changes over $O(n)$ updates



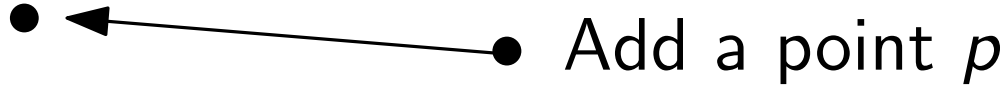
Dynamic Convex Hull Complexity

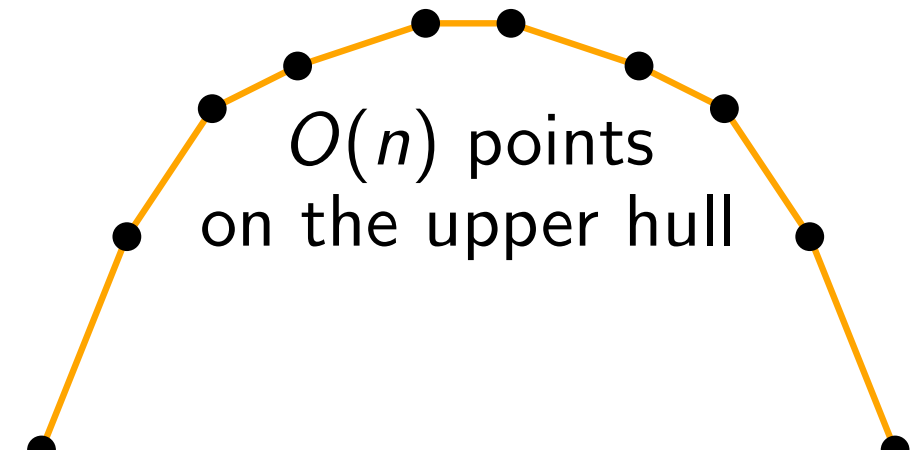
Generally, $O(n^2)$ changes over $O(n)$ updates



Dynamic Convex Hull Complexity

Generally, $O(n^2)$ changes over $O(n)$ updates

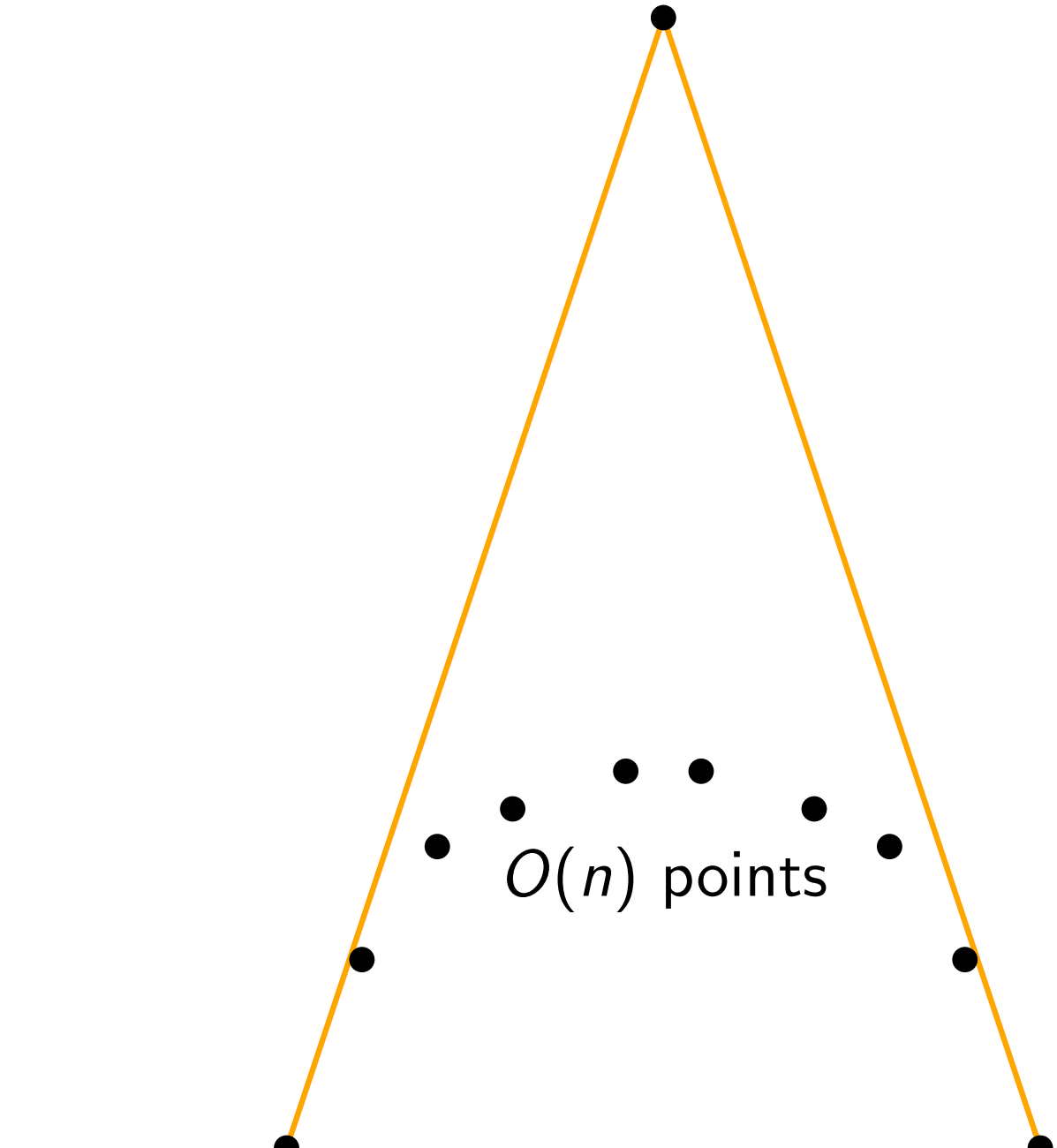




Dynamic Convex Hull Complexity

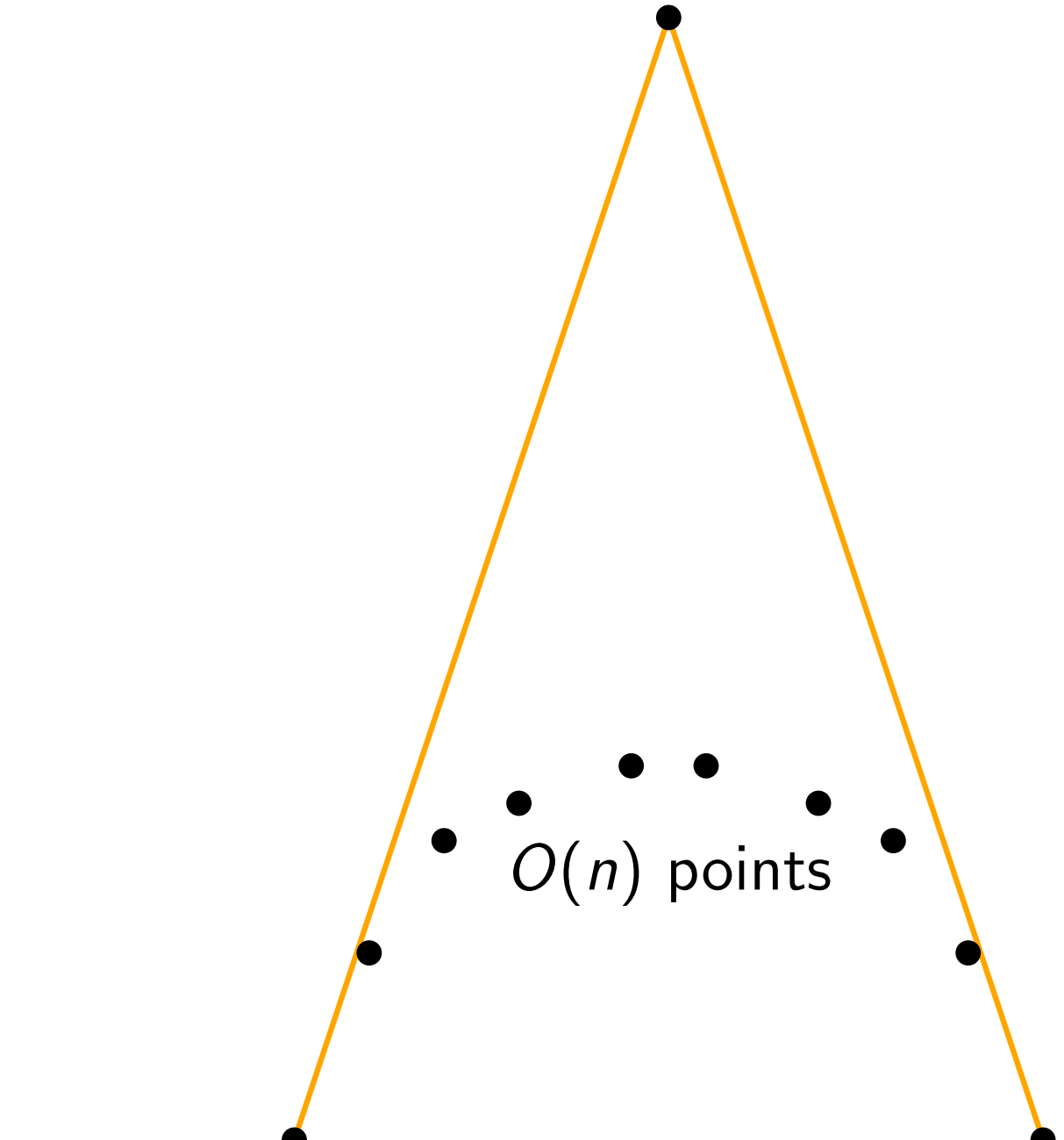
Generally, $O(n^2)$ changes over $O(n)$ updates

- Add a point p
- $O(n)$ changes



Dynamic Convex Hull Complexity

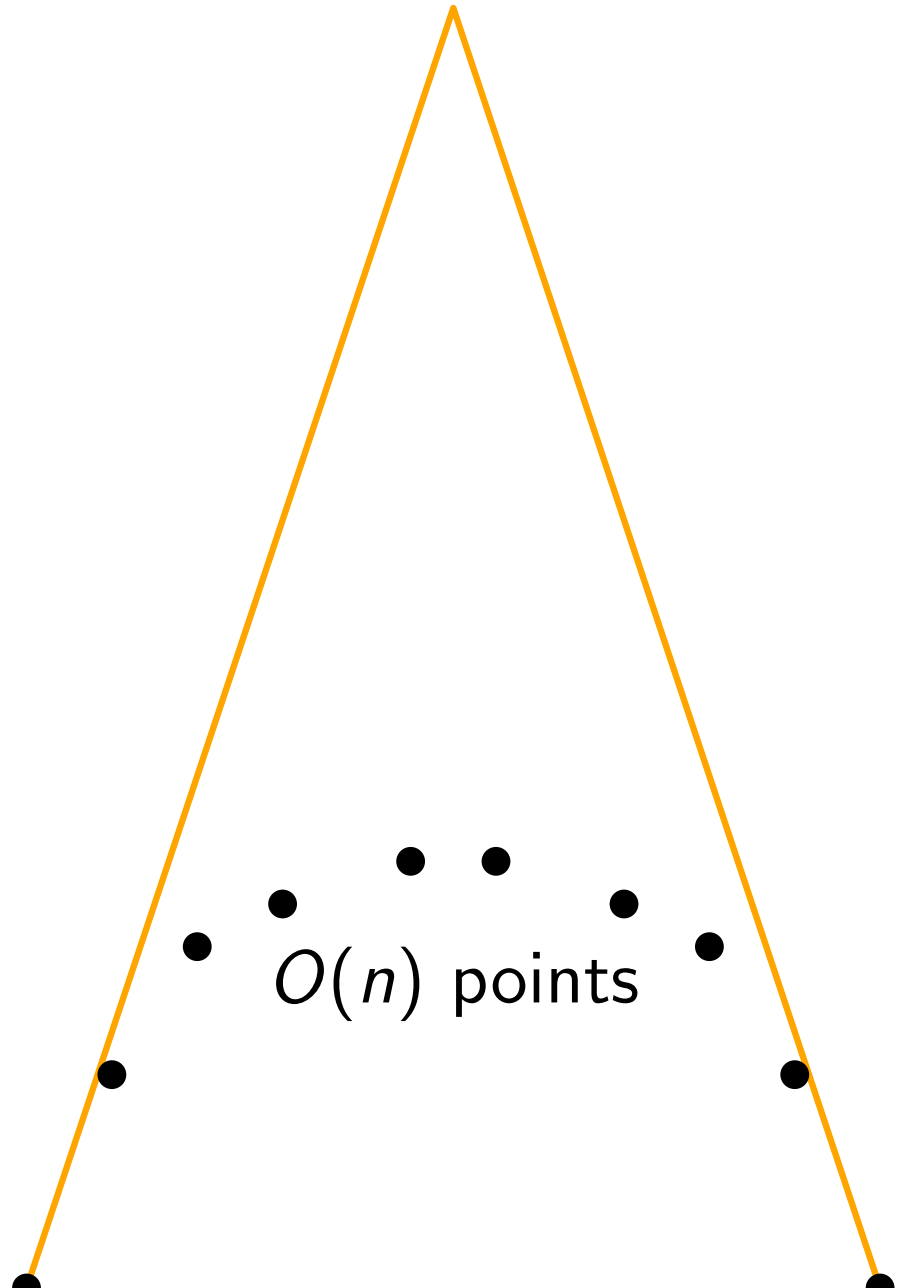
Generally, $O(n^2)$ changes over $O(n)$ updates



- Add a point p
- $O(n)$ changes
- Add/remove p O(n) times

Dynamic Convex Hull Complexity

Generally, $O(n^2)$ changes over $O(n)$ updates

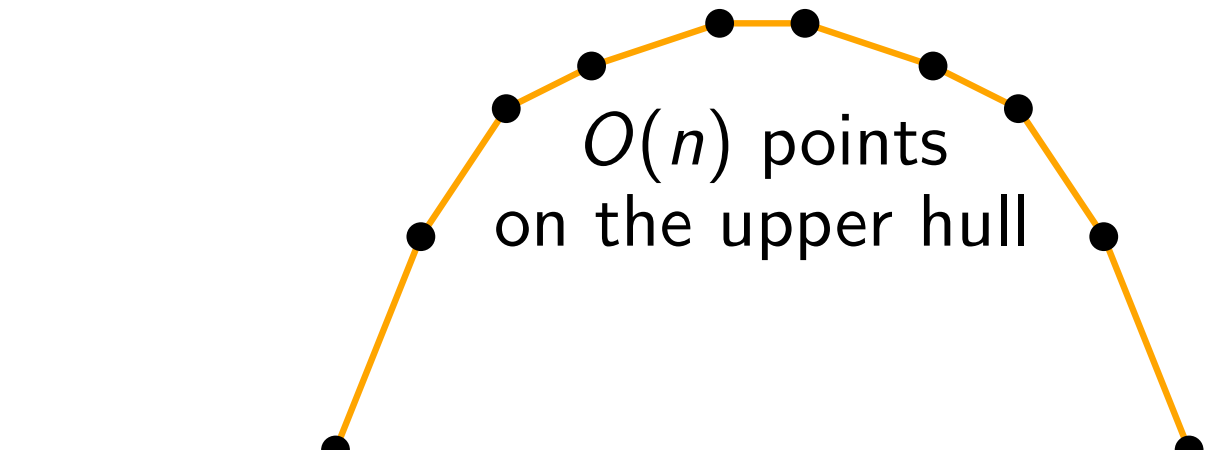


- Add a point p
- $O(n)$ changes
- Add/remove p O(n) times

Dynamic Convex Hull Complexity

Generally, $O(n^2)$ changes over $O(n)$ updates

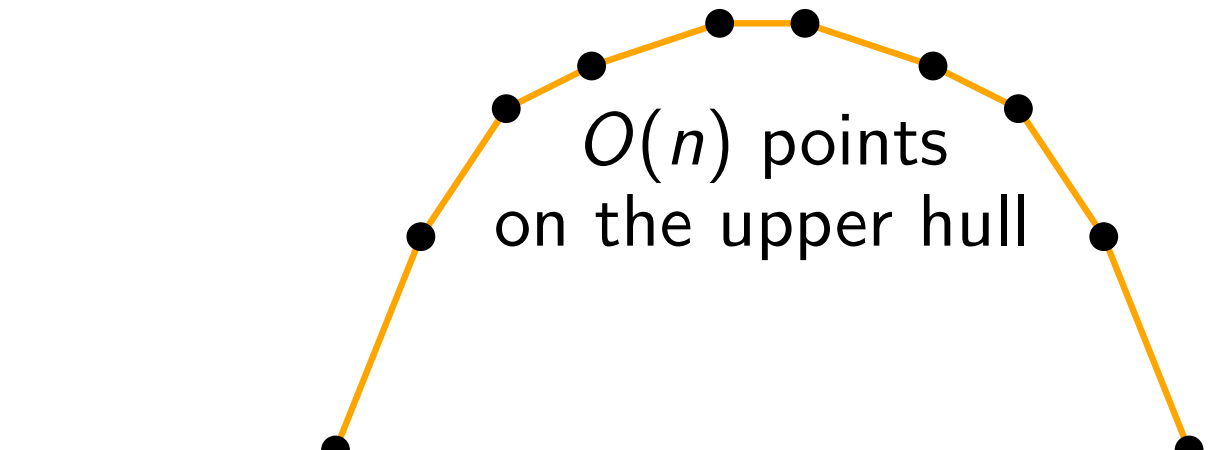
- Add a point p
- $O(n)$ changes
- Add/remove p O(n) times



Dynamic Convex Hull Complexity

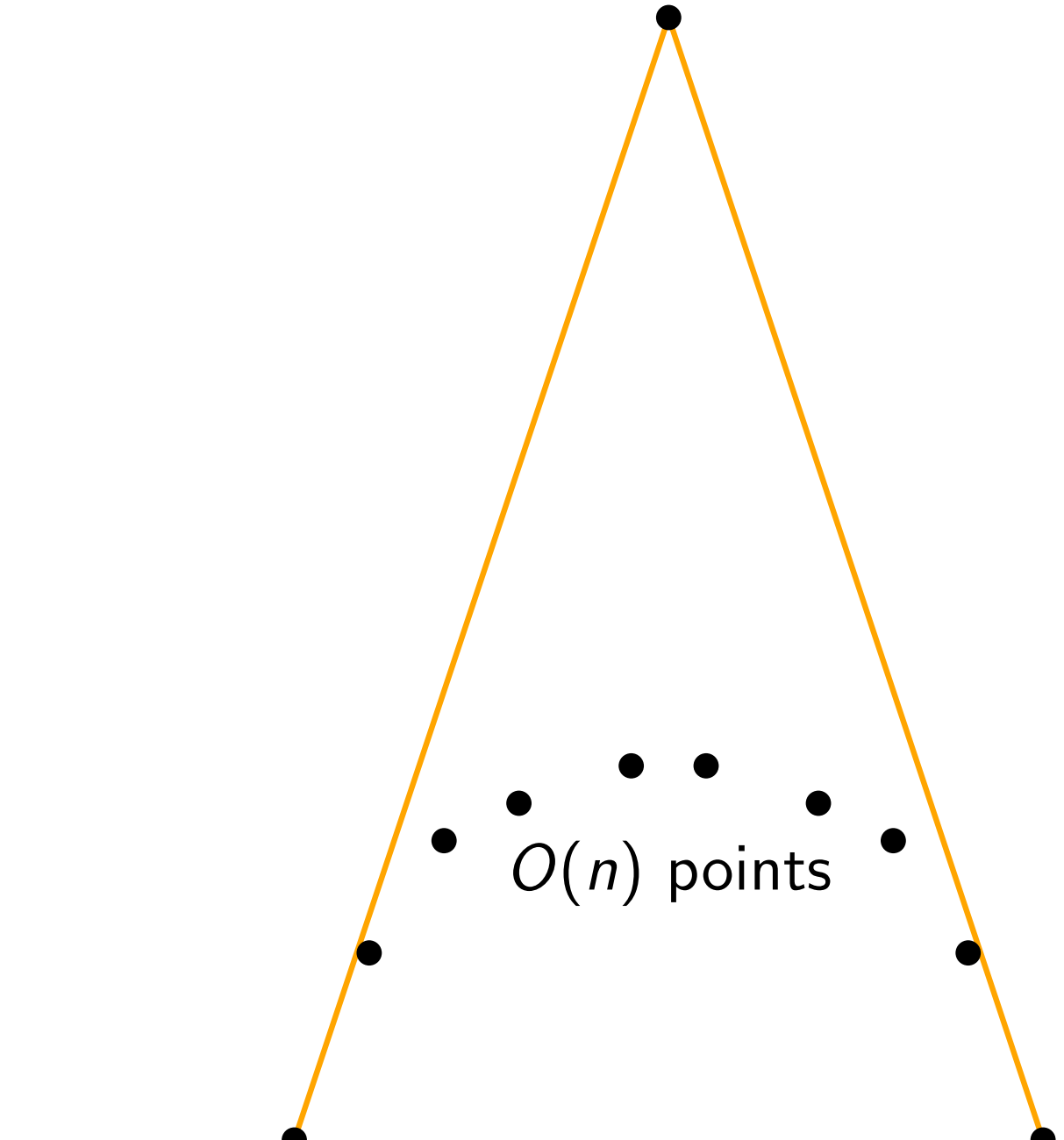
Generally, $O(n^2)$ changes over $O(n)$ updates

- Add a point p
- $O(n)$ changes
- Add/remove p O(n) times



Dynamic Convex Hull Complexity

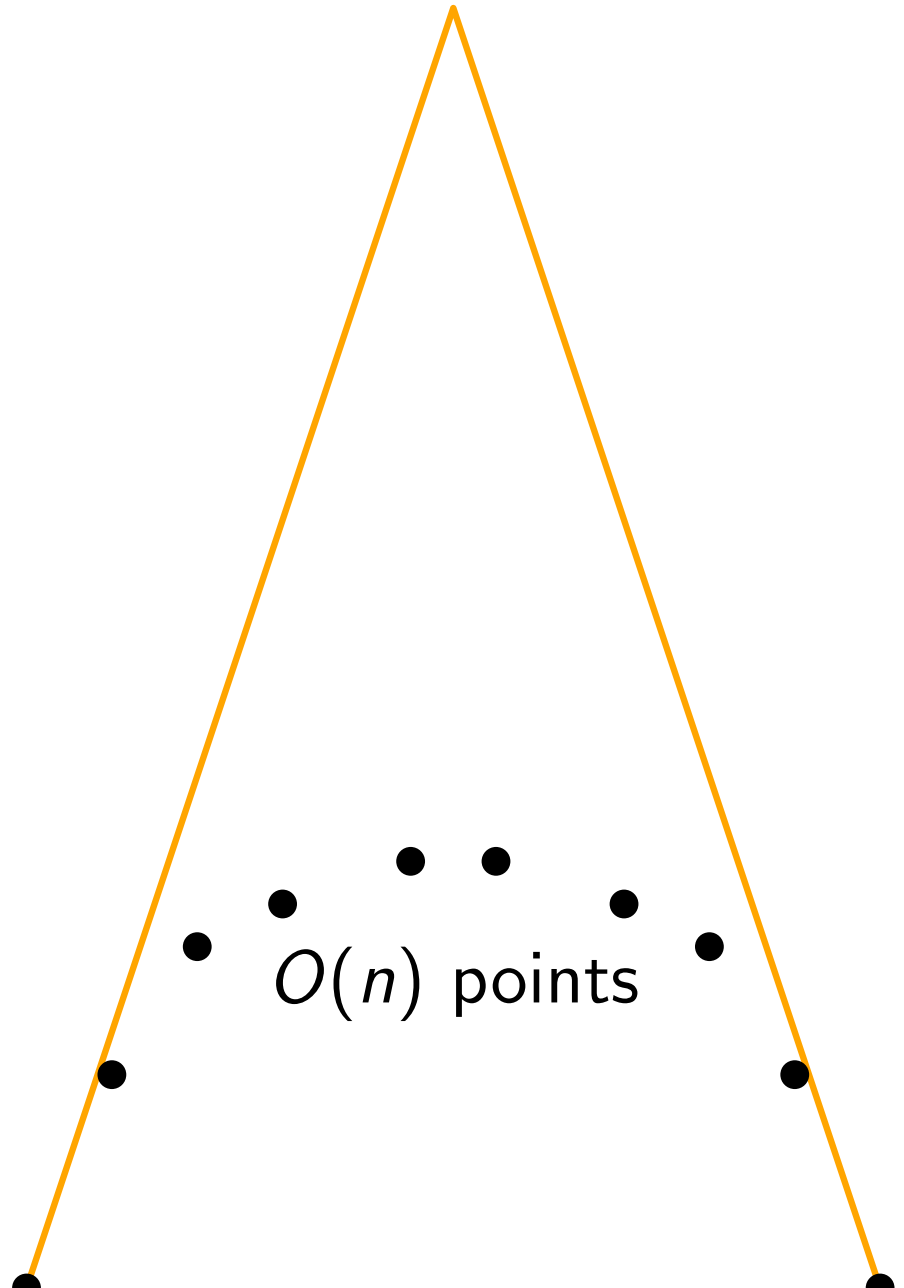
Generally, $O(n^2)$ changes over $O(n)$ updates



- Add a point p
- $O(n)$ changes
- Add/remove p O(n) times

Dynamic Convex Hull Complexity

Generally, $O(n^2)$ changes over $O(n)$ updates

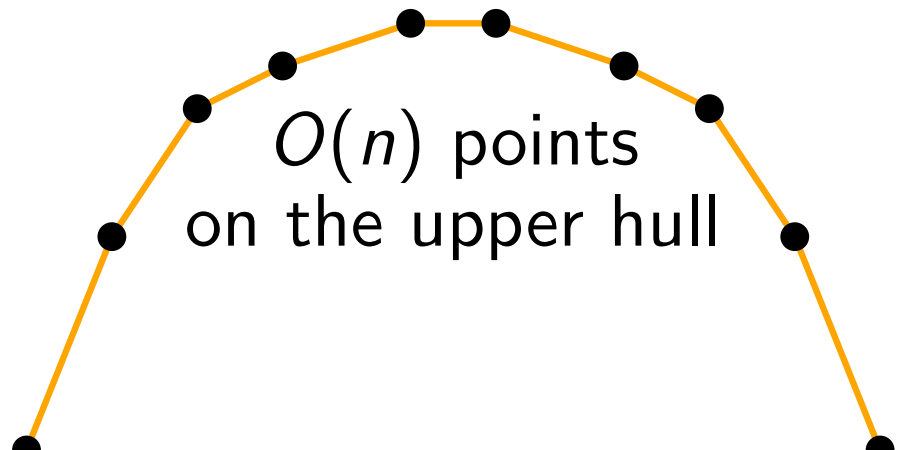


- Add a point p
- $O(n)$ changes
- Add/remove p O(n) times

Dynamic Convex Hull Complexity

Generally, $O(n^2)$ changes over $O(n)$ updates

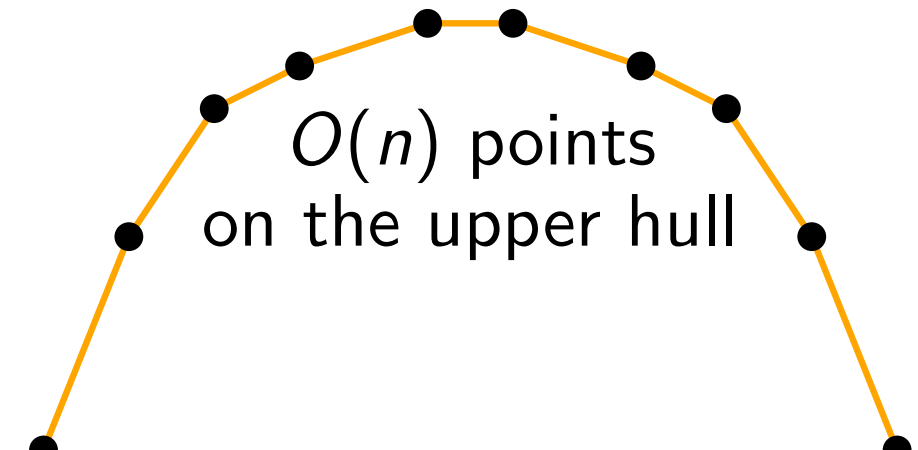
- Add a point p
- $O(n)$ changes
- Add/remove p O(n) times



Dynamic Convex Hull Complexity

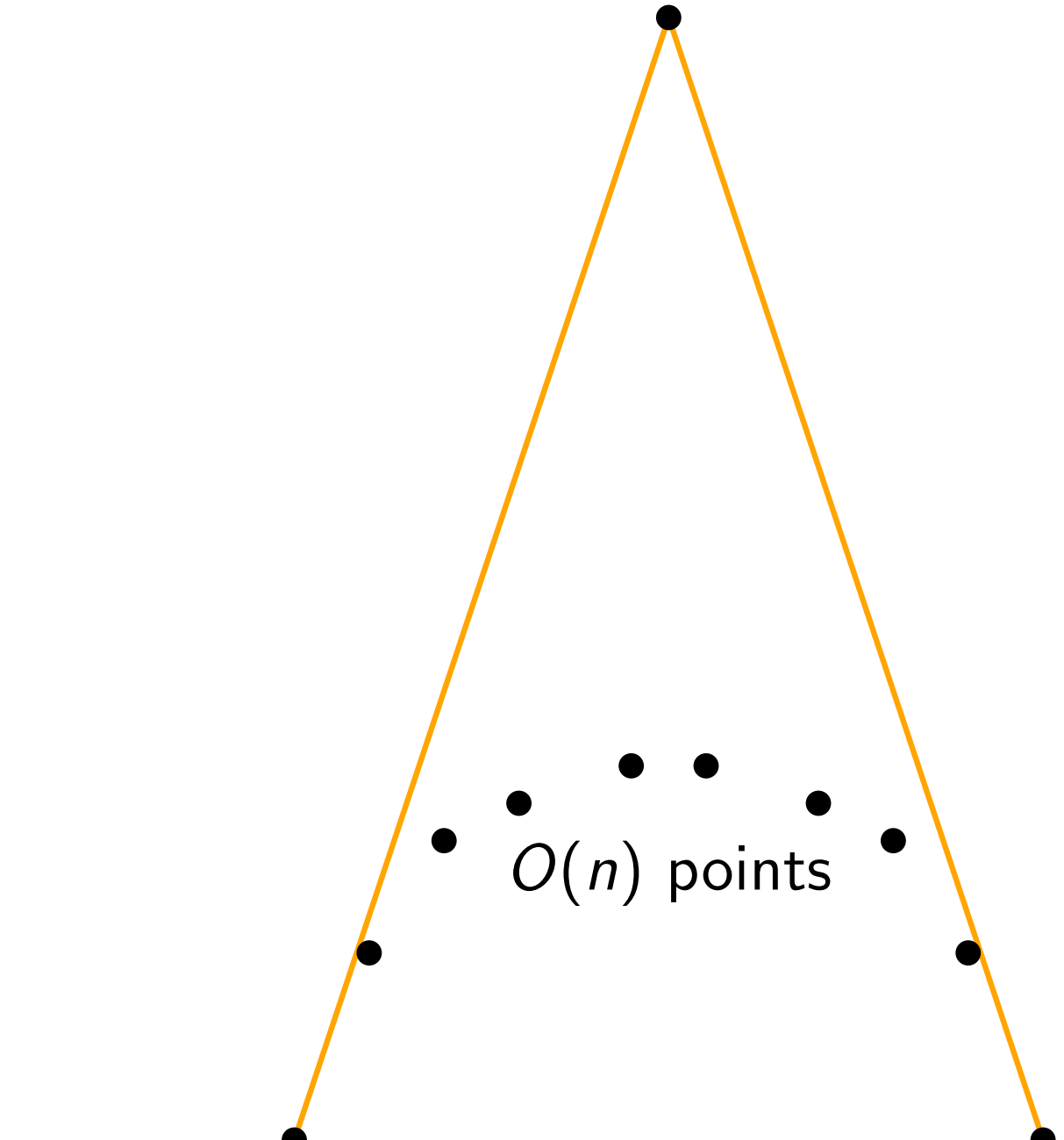
Generally, $O(n^2)$ changes over $O(n)$ updates

- Add a point p
- $O(n)$ changes
- Add/remove p O(n) times



Dynamic Convex Hull Complexity

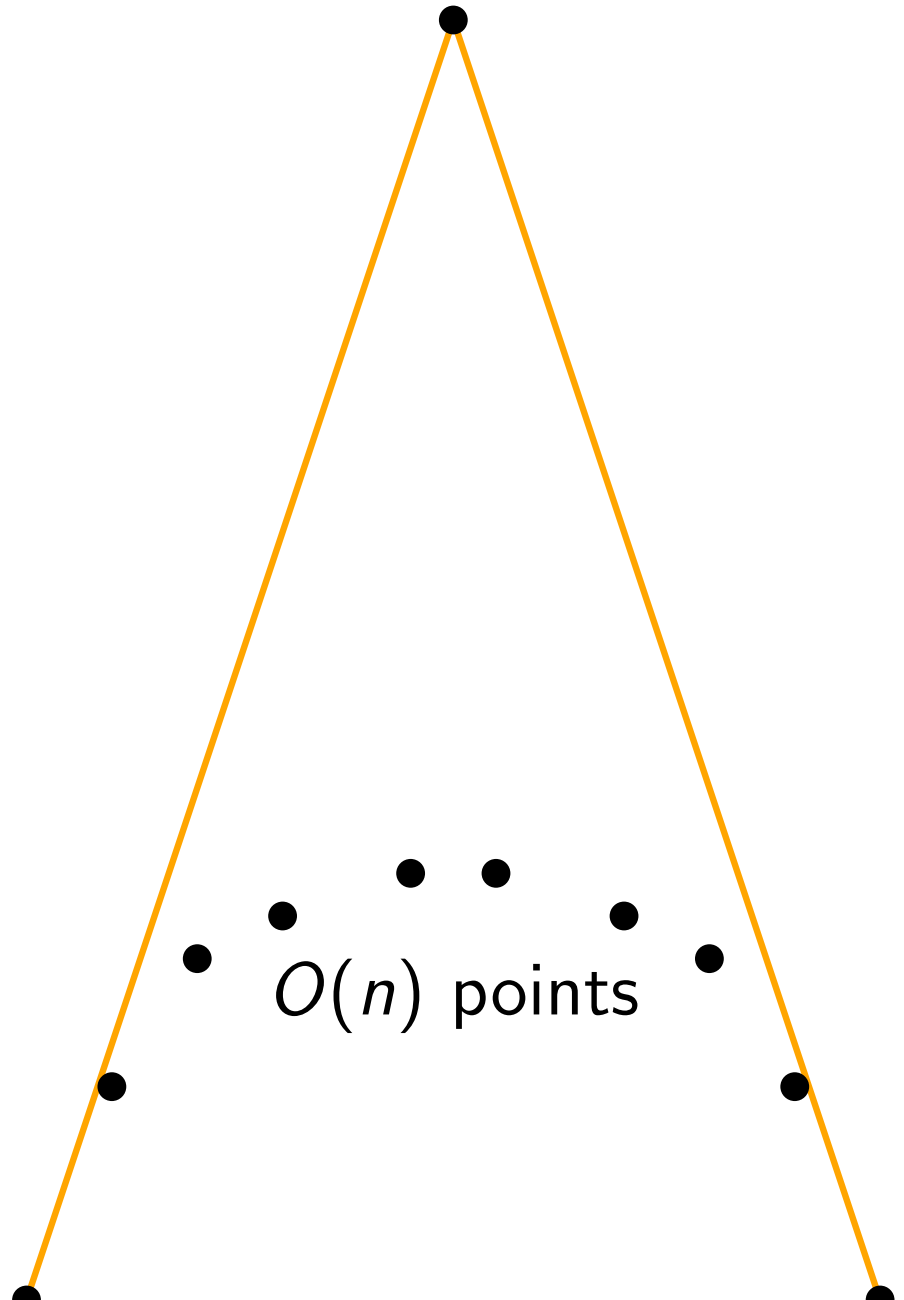
Generally, $O(n^2)$ changes over $O(n)$ updates



- Add a point p
- $O(n)$ changes
- Add/remove p O(n) times

Dynamic Convex Hull Complexity

Generally, $O(n^2)$ changes over $O(n)$ updates



- Add a point p
- $O(n)$ changes
- Add/remove p O(n) times
- $O(n^2)$ changes

FIFO Convex Hull Complexity

Lemma: # of structural changes to upper hull after n FIFO updates is $O(n \log n)$.

FIFO Convex Hull Complexity

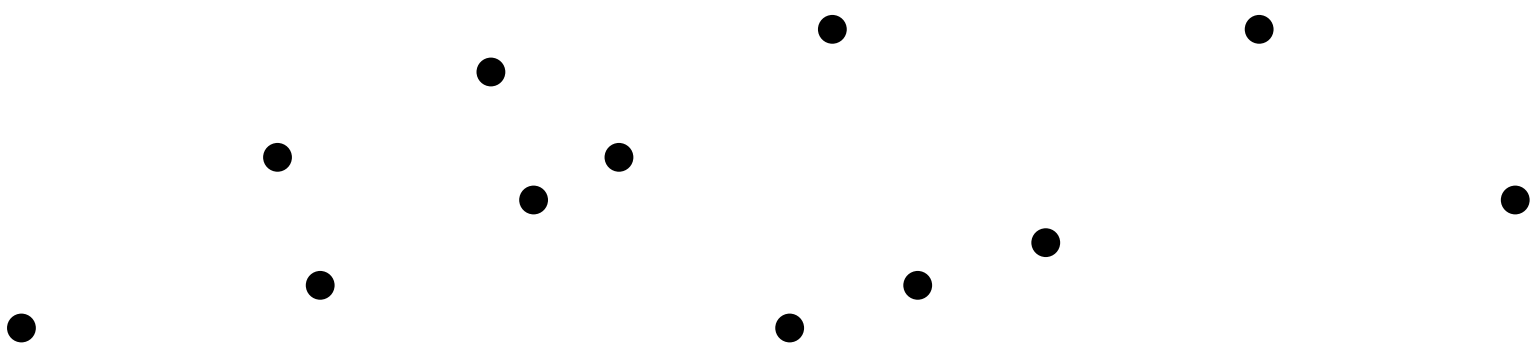
Lemma: # of structural changes to upper hull after n FIFO updates is $O(n \log n)$.

distinct upper hull edges over n updates

FIFO Convex Hull Complexity

Lemma: # of structural changes to upper hull after n FIFO updates is $O(n \log n)$.

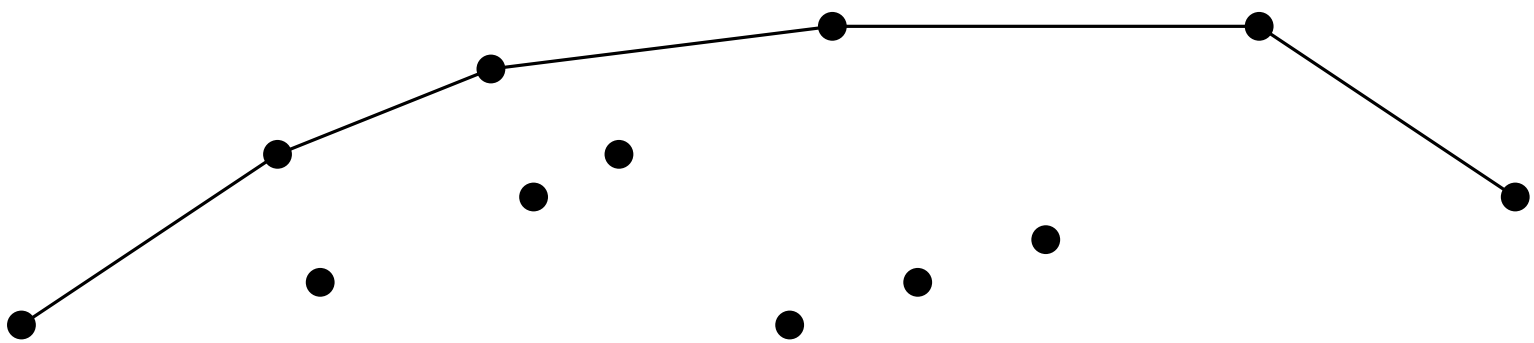
distinct upper hull edges over n updates



FIFO Convex Hull Complexity

Lemma: # of structural changes to upper hull after n FIFO updates is $O(n \log n)$.

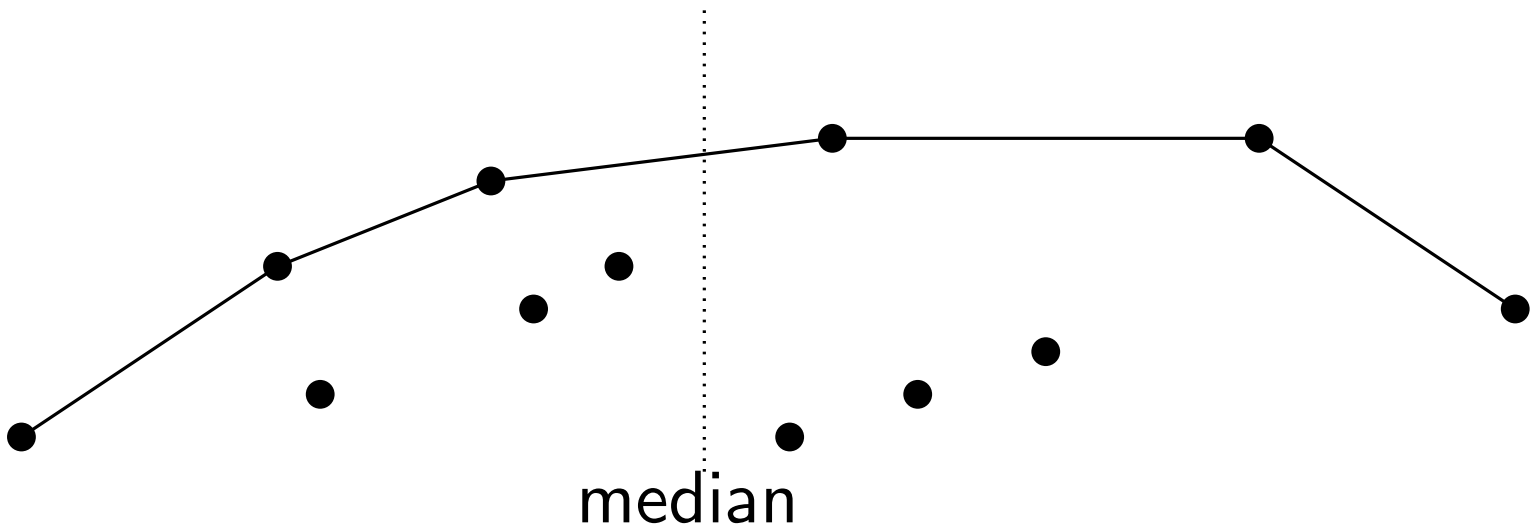
distinct upper hull edges over n updates



FIFO Convex Hull Complexity

Lemma: # of structural changes to upper hull after n FIFO updates is $O(n \log n)$.

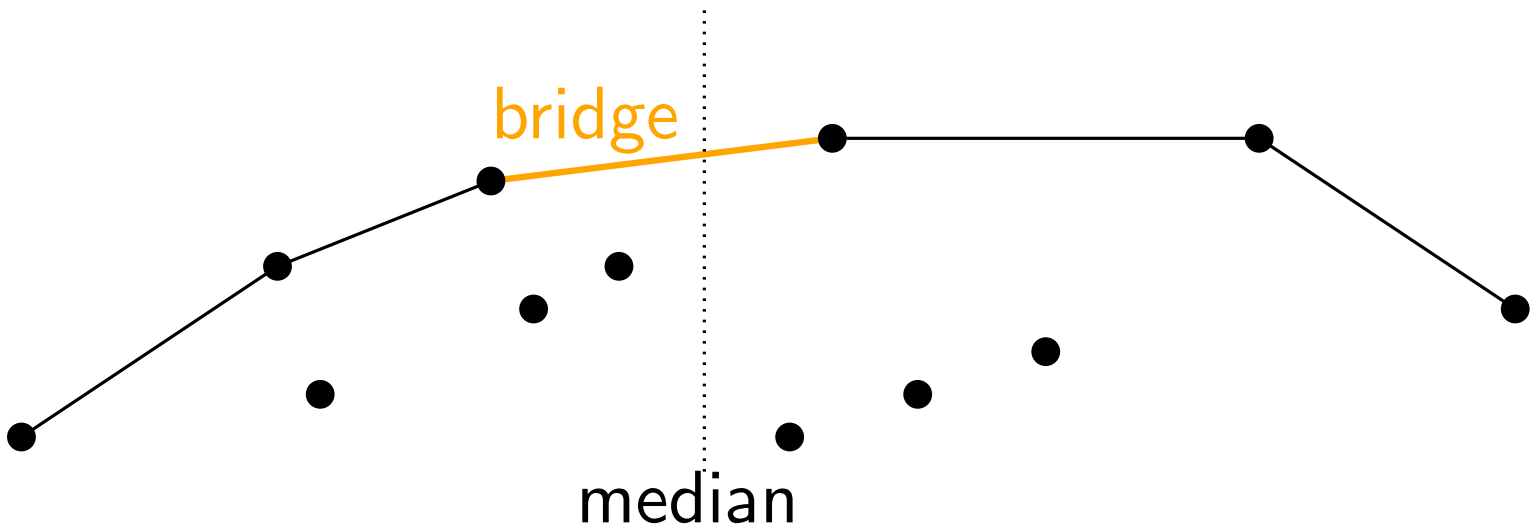
distinct upper hull edges over n updates



FIFO Convex Hull Complexity

Lemma: # of structural changes to upper hull after n FIFO updates is $O(n \log n)$.

distinct upper hull edges over n updates

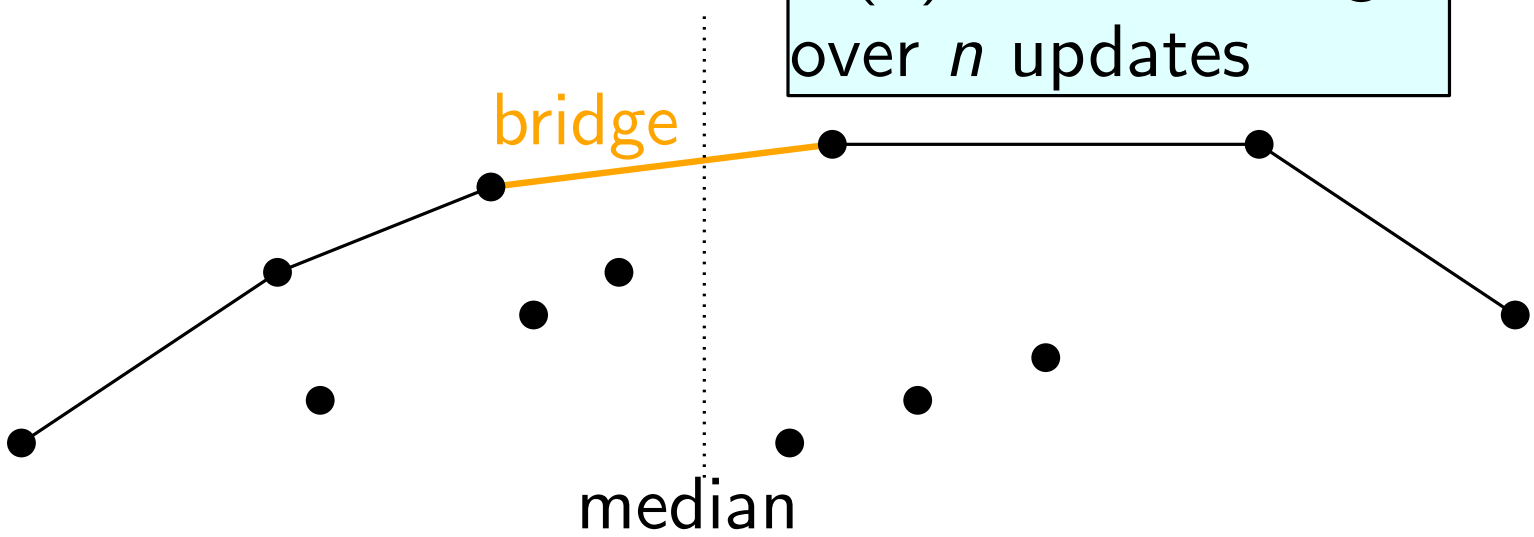


FIFO Convex Hull Complexity

Lemma: # of structural changes to upper hull after n FIFO updates is $O(n \log n)$.

distinct upper hull edges over n updates

$O(n)$ distinct bridges over n updates



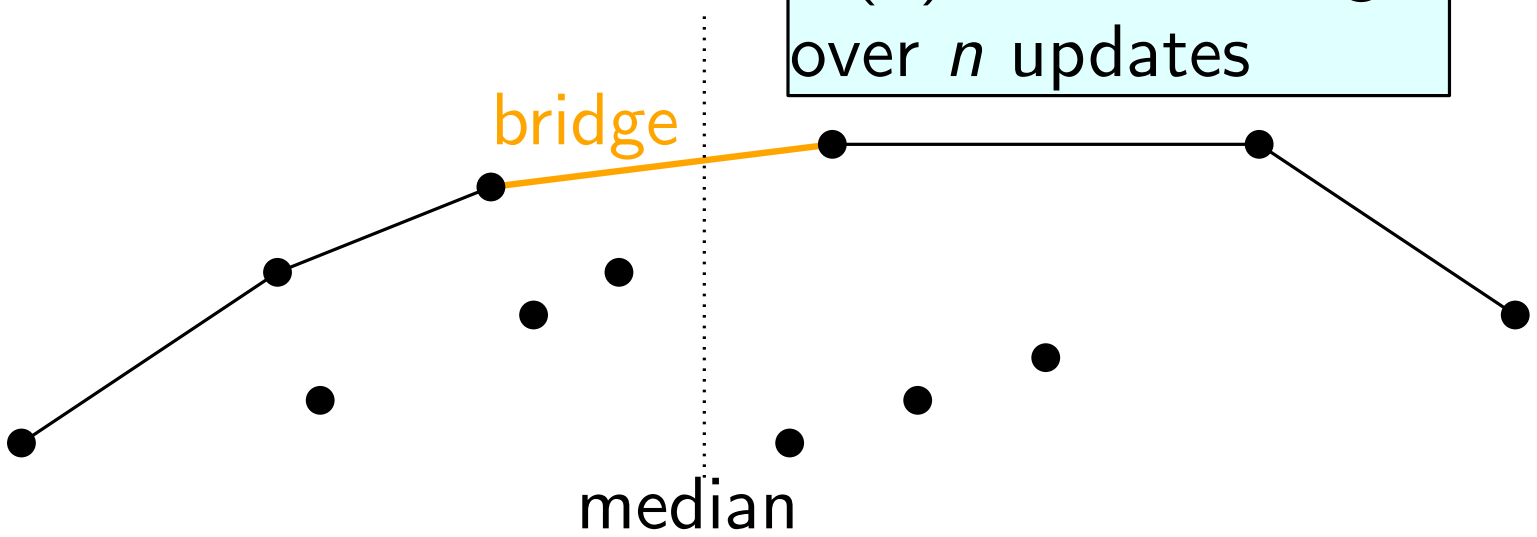
FIFO Convex Hull Complexity

Lemma: # of structural changes to upper hull after n FIFO updates is $O(n \log n)$.

distinct upper hull edges over n updates

$$E(n) = 2 \cdot E(n) + O(n)$$

$O(n)$ distinct bridges over n updates



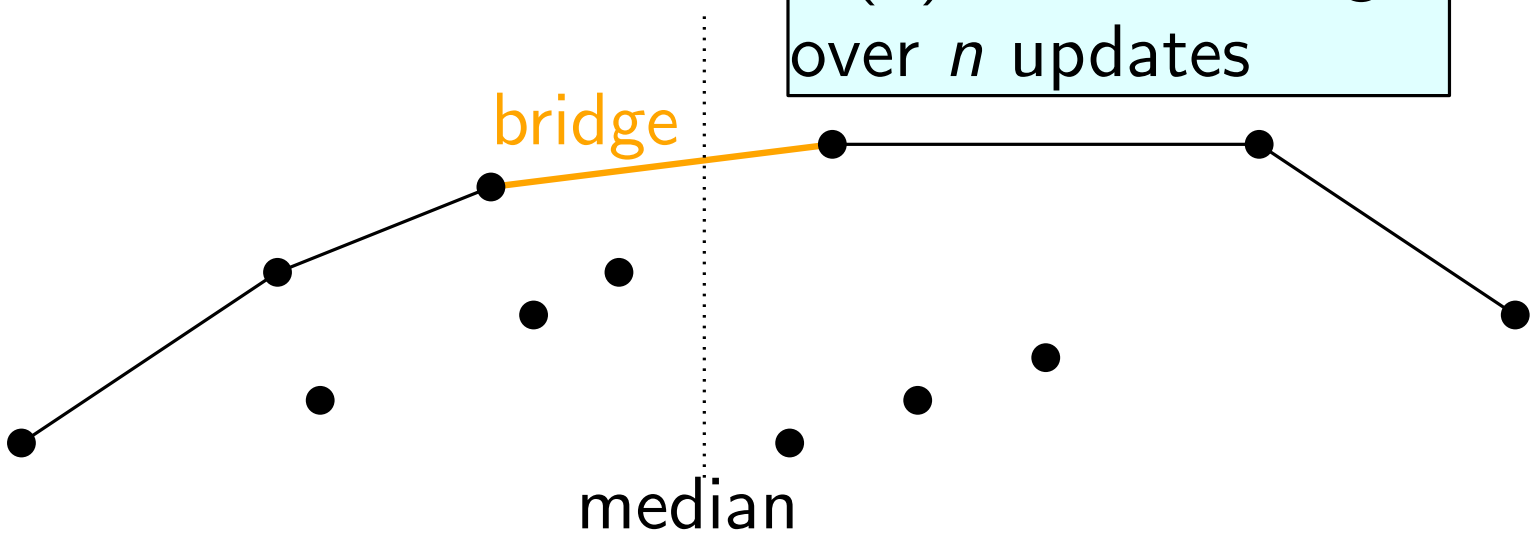
FIFO Convex Hull Complexity

Lemma: # of structural changes to upper hull after n FIFO updates is $O(n \log n)$.

distinct upper hull edges over n updates

$$E(n) = 2 \cdot E(n) + O(n) = O(n \log n)$$

$O(n)$ distinct bridges over n updates



FIFO Convex Hull Complexity

Lemma: # of structural changes to upper hull after n FIFO updates is $O(n \log n)$.

distinct upper hull edges over n updates

$O(n \log n)$

FIFO Convex Hull Complexity

Lemma: # of structural changes to upper hull after n FIFO updates is $O(n \log n)$.

distinct upper hull edges over n updates

$O(n \log n)$

FIFO order $\rightarrow$ edge never re-added to upper hull

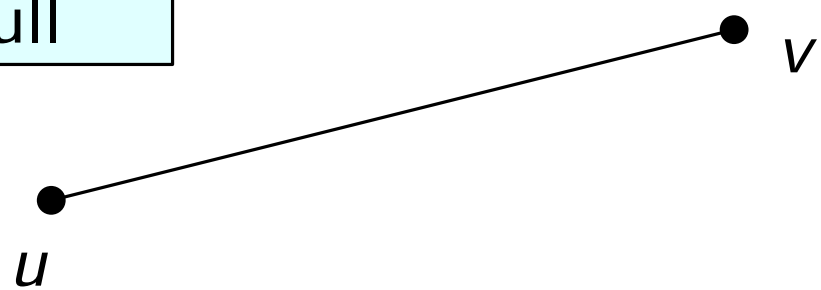
FIFO Convex Hull Complexity

Lemma: # of structural changes to upper hull after n FIFO updates is $O(n \log n)$.

distinct upper hull edges over n updates

$O(n \log n)$

FIFO order $\rightarrow$ edge never re-added to upper hull



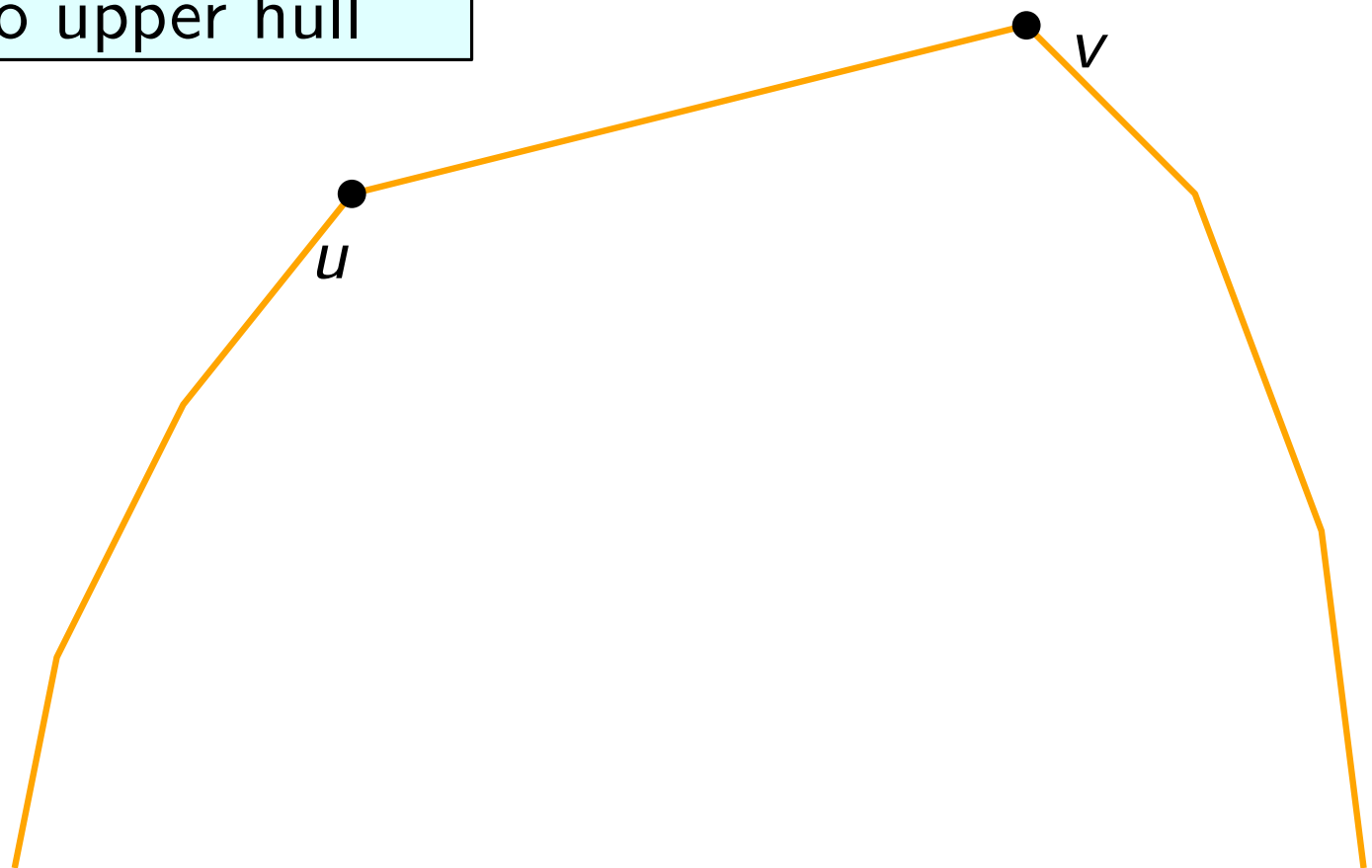
FIFO Convex Hull Complexity

Lemma: # of structural changes to upper hull after n FIFO updates is $O(n \log n)$.

distinct upper hull edges over n updates

$O(n \log n)$

FIFO order $\rightarrow$ edge never re-added to upper hull



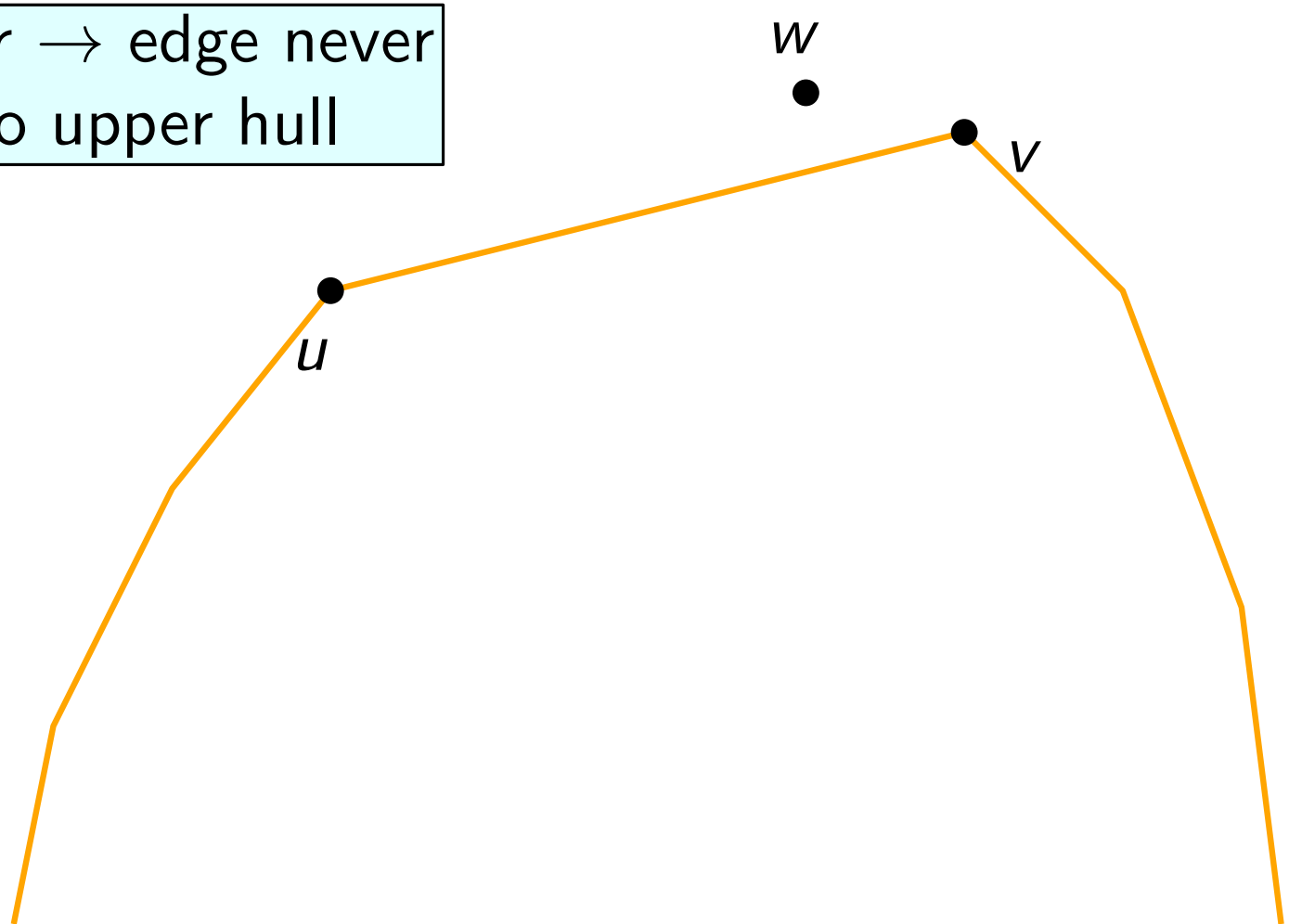
FIFO Convex Hull Complexity

Lemma: # of structural changes to upper hull after n FIFO updates is $O(n \log n)$.

distinct upper hull edges over n updates

$O(n \log n)$

FIFO order $\rightarrow$ edge never re-added to upper hull



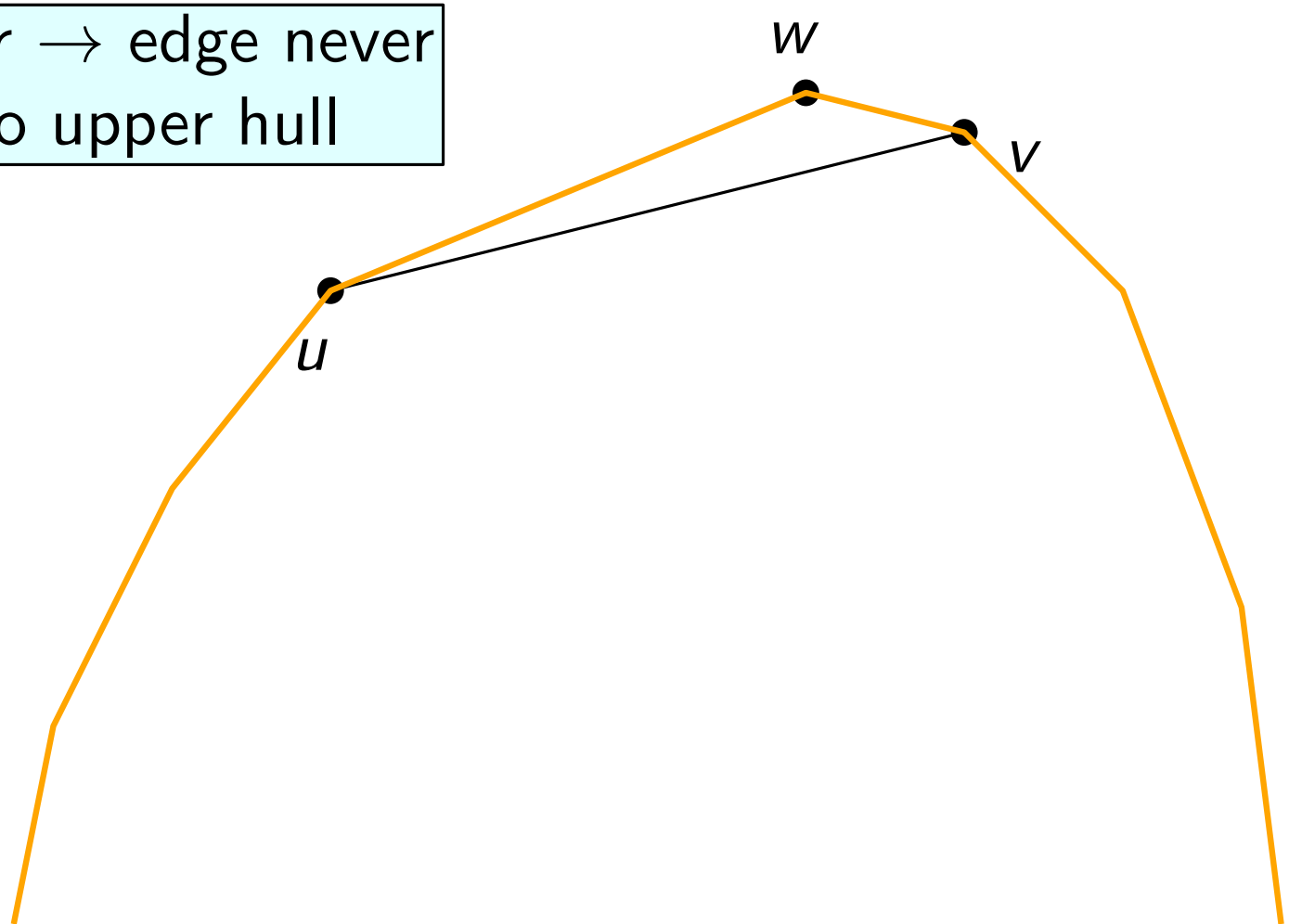
FIFO Convex Hull Complexity

Lemma: # of structural changes to upper hull after n FIFO updates is $O(n \log n)$.

distinct upper hull edges over n updates

$O(n \log n)$

FIFO order $\rightarrow$ edge never re-added to upper hull



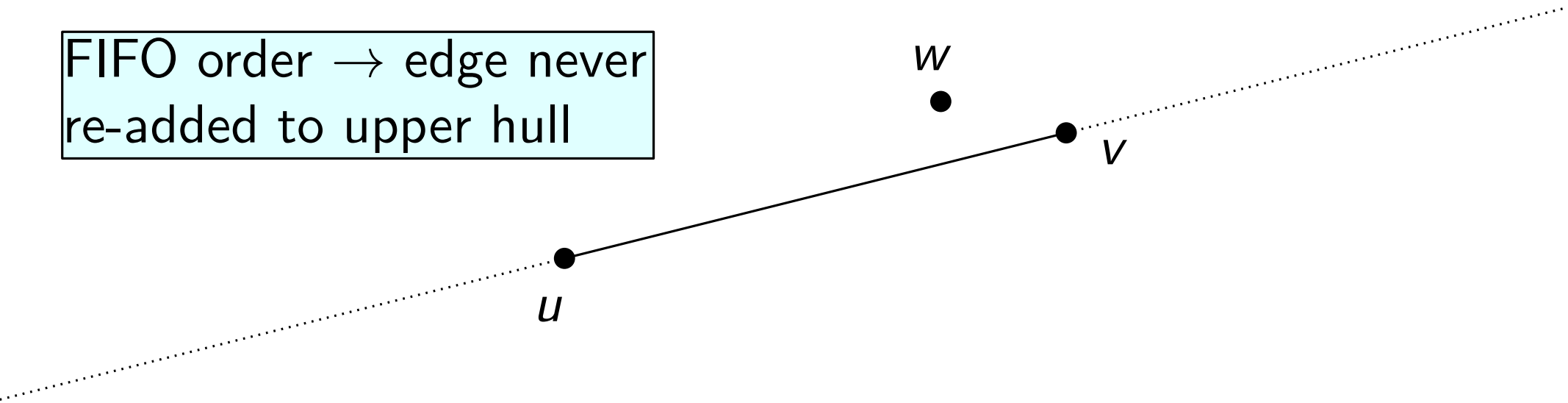
FIFO Convex Hull Complexity

Lemma: # of structural changes to upper hull after n FIFO updates is $O(n \log n)$.

distinct upper hull edges over n updates

$O(n \log n)$

FIFO order $\rightarrow$ edge never re-added to upper hull



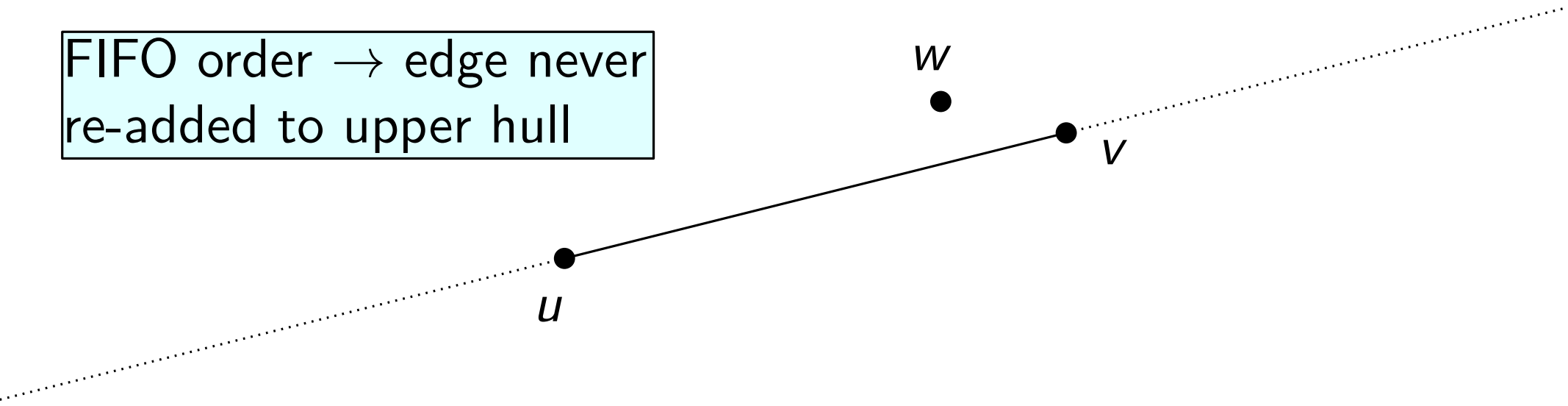
FIFO Convex Hull Complexity

Lemma: # of structural changes to upper hull after n FIFO updates is $O(n \log n)$.

distinct upper hull edges over n updates

$O(n \log n)$

FIFO order $\rightarrow$ edge never re-added to upper hull



FIFO: Conclusion

Combinatorial Result

FIFO: Conclusion

Combinatorial Result

- At most $O(n \log n)$ changes over n FIFO updates.

FIFO: Conclusion

Combinatorial Result

- At most $O(n \log n)$ changes over n FIFO updates.

2D CH area decision

FIFO: Conclusion

Combinatorial Result

- At most $O(n \log n)$ changes over n FIFO updates.

2D CH area decision

- Brodal and Jacob
  $O(n \log^2 n)$

FIFO: Conclusion

Combinatorial Result

- At most $O(n \log n)$ changes over n FIFO updates.

2D CH area decision

- Brodal and Jacob
  $O(n \log^2 n)$
- Hull trees + Davenport-Schinzel sequences
  $O(n\alpha(n) \log n)$, where α is inverse Ackermann function

FIFO: Conclusion

Combinatorial Result

- At most $O(n \log n)$ changes over n FIFO updates.

2D CH area decision

- Brodal and Jacob
  $O(n \log^2 n)$
- Hull trees + Davenport-Schinzel sequences
  $O(n\alpha(n) \log n)$, where α is inverse Ackermann function

2D width decision

FIFO: Conclusion

Combinatorial Result

- At most $O(n \log n)$ changes over n FIFO updates.

2D CH area decision

- Brodal and Jacob
  $O(n \log^2 n)$
- Hull trees + Davenport-Schinzel sequences
  $O(n\alpha(n) \log n)$, where α is inverse Ackermann function

2D width decision

- Eppstein's dynamic 2D data structure
  $O(n \log^7 n)$

FIFO: Conclusion

Combinatorial Result

- At most $O(n \log n)$ changes over n FIFO updates.

Can we do better?

2D CH area decision

- Brodal and Jacob
  $O(n \log^2 n)$
- Hull trees + Davenport-Schinzel sequences
  $O(n\alpha(n) \log n)$, where α is inverse Ackermann function

2D width decision

- Eppstein's dynamic 2D data structure
  $O(n \log^7 n)$

FIFO: Conclusion

Combinatorial Result

- At most $O(n \log n)$ changes over n FIFO updates.

Can we do better? Hershberger says yes!

2D CH area decision

- Brodal and Jacob
  $O(n \log^2 n)$
- Hull trees + Davenport-Schinzel sequences
  $O(n\alpha(n) \log n)$, where α is inverse Ackermann function

2D width decision

- Eppstein's dynamic 2D data structure
  $O(n \log^7 n)$

New Results

Approach #1: Grids & Quadtrees

Problem	Preprocessing	Space	Query
CP Decision	$O(n)$ exp.	$O(n)$ bits	$O(1)$
Closest Pair	$O(n \lg n \lg \lg n)$	$O(n \lg n)$	$O(\lg \lg n)$

Approach #2: Generalized Range Successor

Problem	Preprocessing Time
2D diameter decision	$O(n \lg^2 n)$
$\rightarrow$ improved to	$O(n \lg n)$
3D diameter decision	$O(n \lg^2 n)$
2D orth. seg. intersection detection	$O(n \lg n \lg \lg n)$

Approach #3: FIFO Update Sequence

Problem	Preprocessing Time
2D convex hull area decision	$O(n \lg n \lg \lg n)$
$\rightarrow$ improved to	$O(n\alpha(n) \lg n)$
2D width decision	$O(n \lg^7 n)$

New Results

Approach #1: Grids & Quadtrees

Problem	Preprocessing	Space	Query
CP Decision	$O(n)$ exp.	$O(n)$ bits	$O(1)$
Closest Pair	$O(n \lg n \lg \lg n)$	$O(n \lg n)$	$O(\lg \lg n)$

Approach #2: Generalized Range Successor

Problem	Preprocessing Time
2D diameter decision	$O(n \lg^2 n)$
$\rightarrow$ improved to	$O(n \lg n)$
3D diameter decision	$O(n \lg^2 n)$
2D orth. seg. intersection detection	$O(n \lg n \lg \lg n)$

Approach #3: FIFO Update Sequence

Problem	Preprocessing Time
2D convex hull area decision	$O(n \lg n \lg \lg n)$
$\rightarrow$ improved to	$O(n \lg n)$
2D width decision	$O(n \lg^7 n)$

New Results

Approach #1: Grids & Quadtrees

Problem	Preprocessing	Space	Query
CP Decision	$O(n)$ exp.	$O(n)$ bits	$O(1)$
Closest Pair	$O(n \lg n \lg\lg n)$	$O(n \lg n)$	$O(\lg\lg n)$

Approach #2: Generalized Range Successor

Problem	Preprocessing Time
2D diameter decision	$O(n \lg^2 n)$
$\rightarrow$ improved to	$O(n \lg n)$
3D diameter decision	$O(n \lg^2 n)$
2D orth. seg. intersection detection	$O(n \lg n \lg\lg n)$

Approach #3: FIFO Update Sequence

Problem	Preprocessing Time
2D convex hull area decision	$O(n \lg n \lg\lg n)$
$\rightarrow$ improved to	$O(n \lg n)$
2D width decision	$O(n \lg^6 n)$

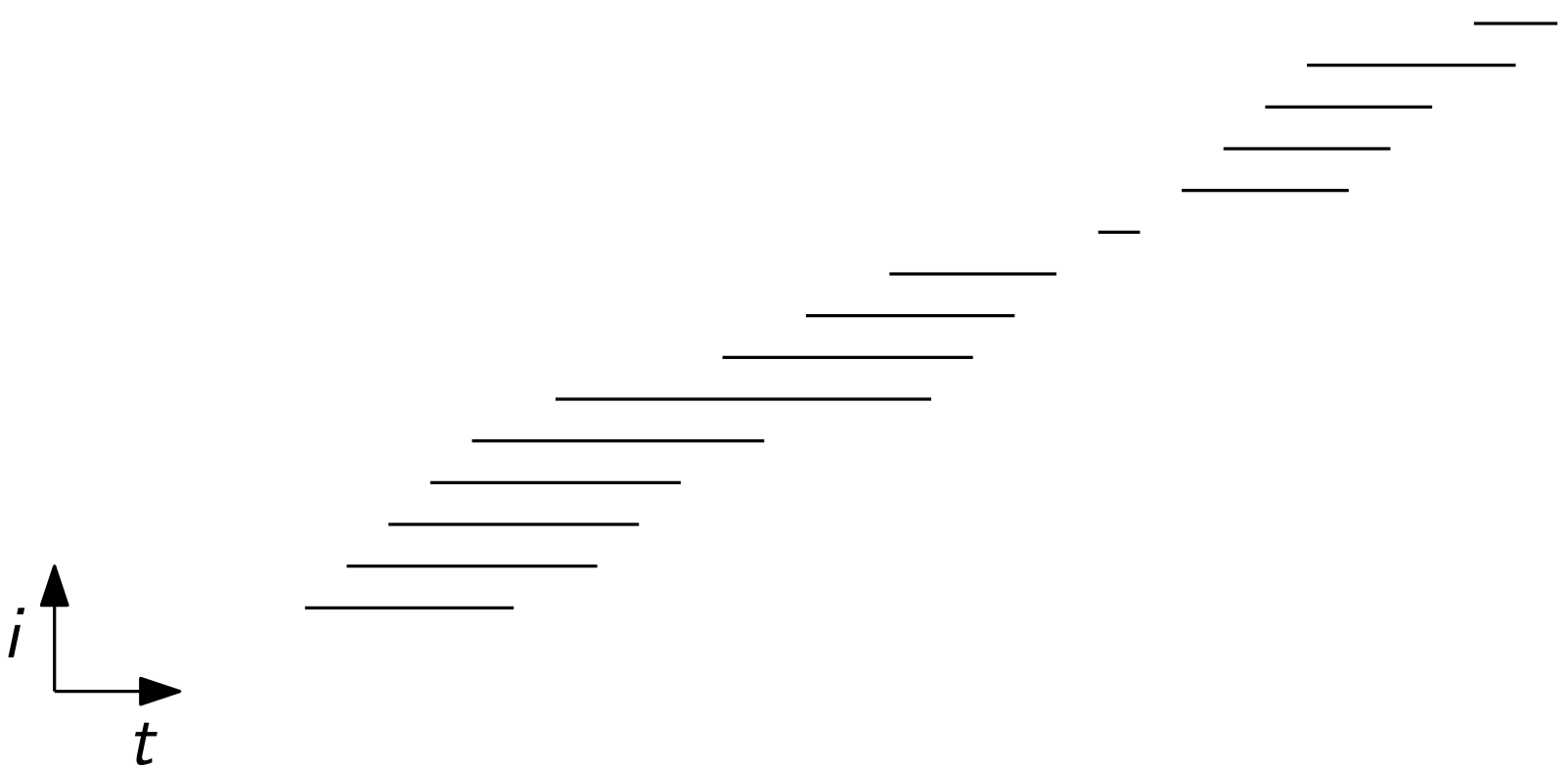
Hershberger's Combinatorial Result

Hershberger's Combinatorial Result

- Plot each point p_i as segment $[(i(p_i), i), (d(p_i), i)]$

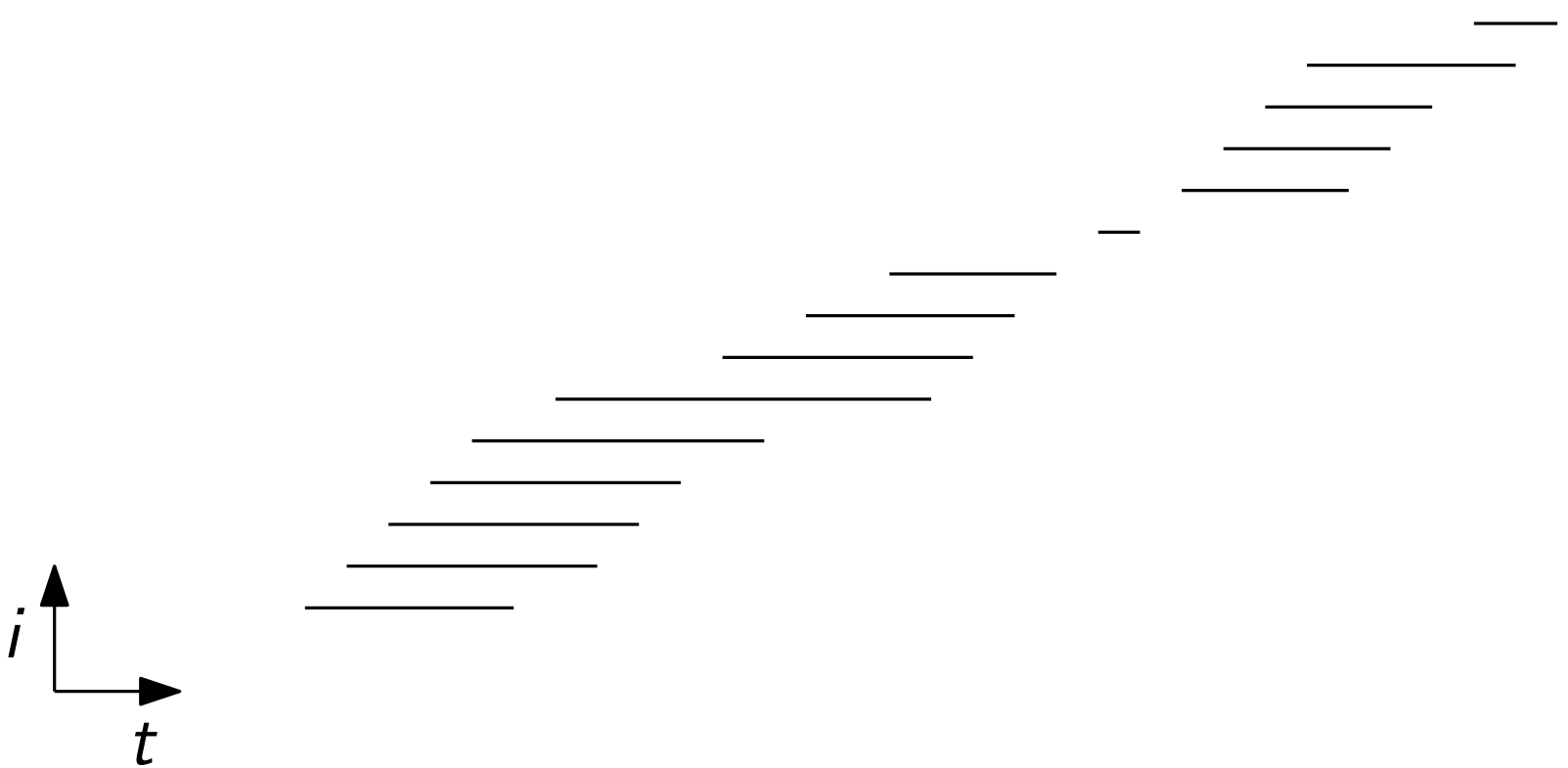
Hershberger's Combinatorial Result

- Plot each point p_i as segment $[(i(p_i), i), (d(p_i), i)]$



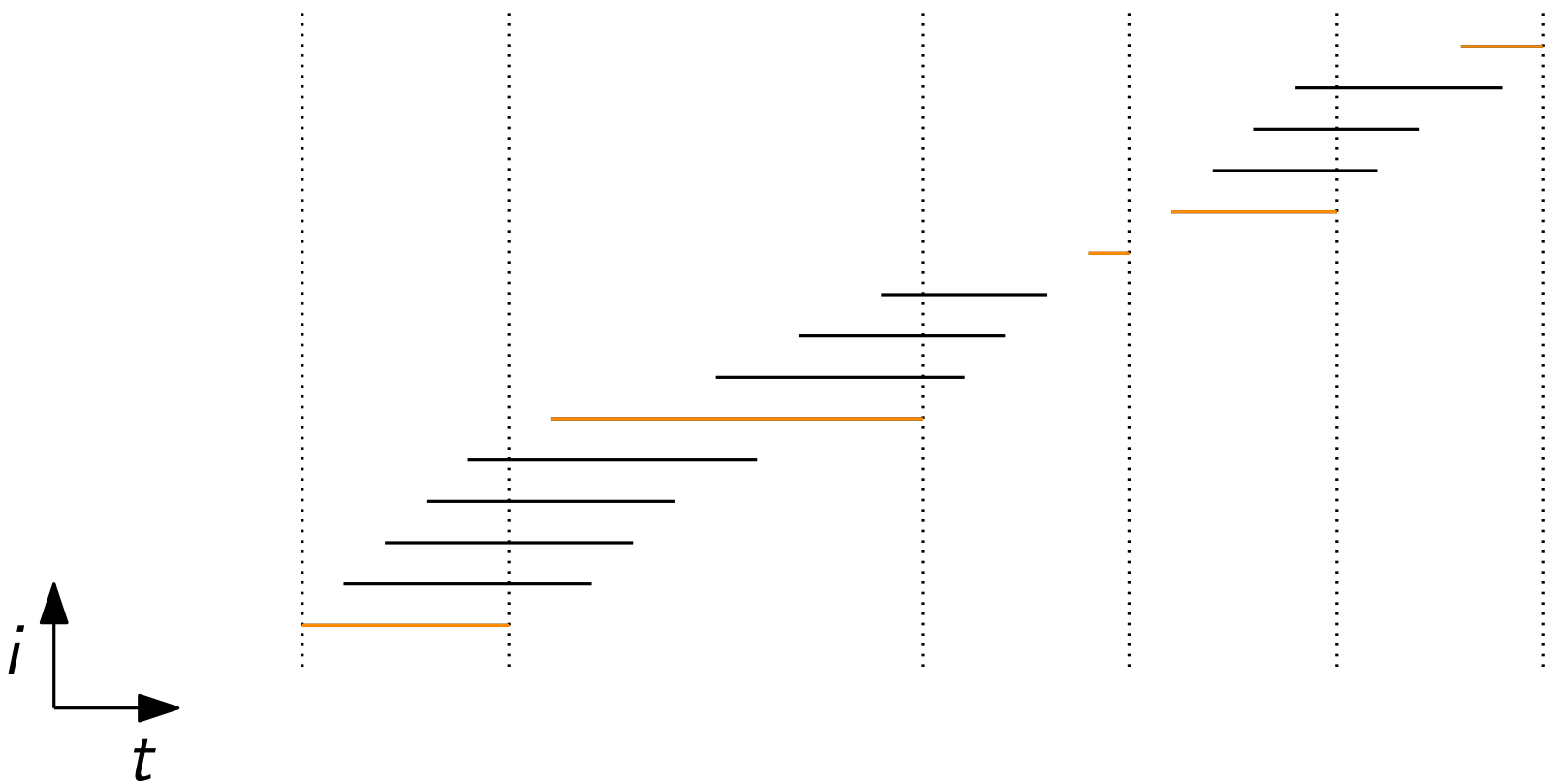
Hershberger's Combinatorial Result

- Plot each point p_i as segment $[(i(p_i), i), (d(p_i), i)]$
- Divide into regions s.t. no point is both inserted & deleted



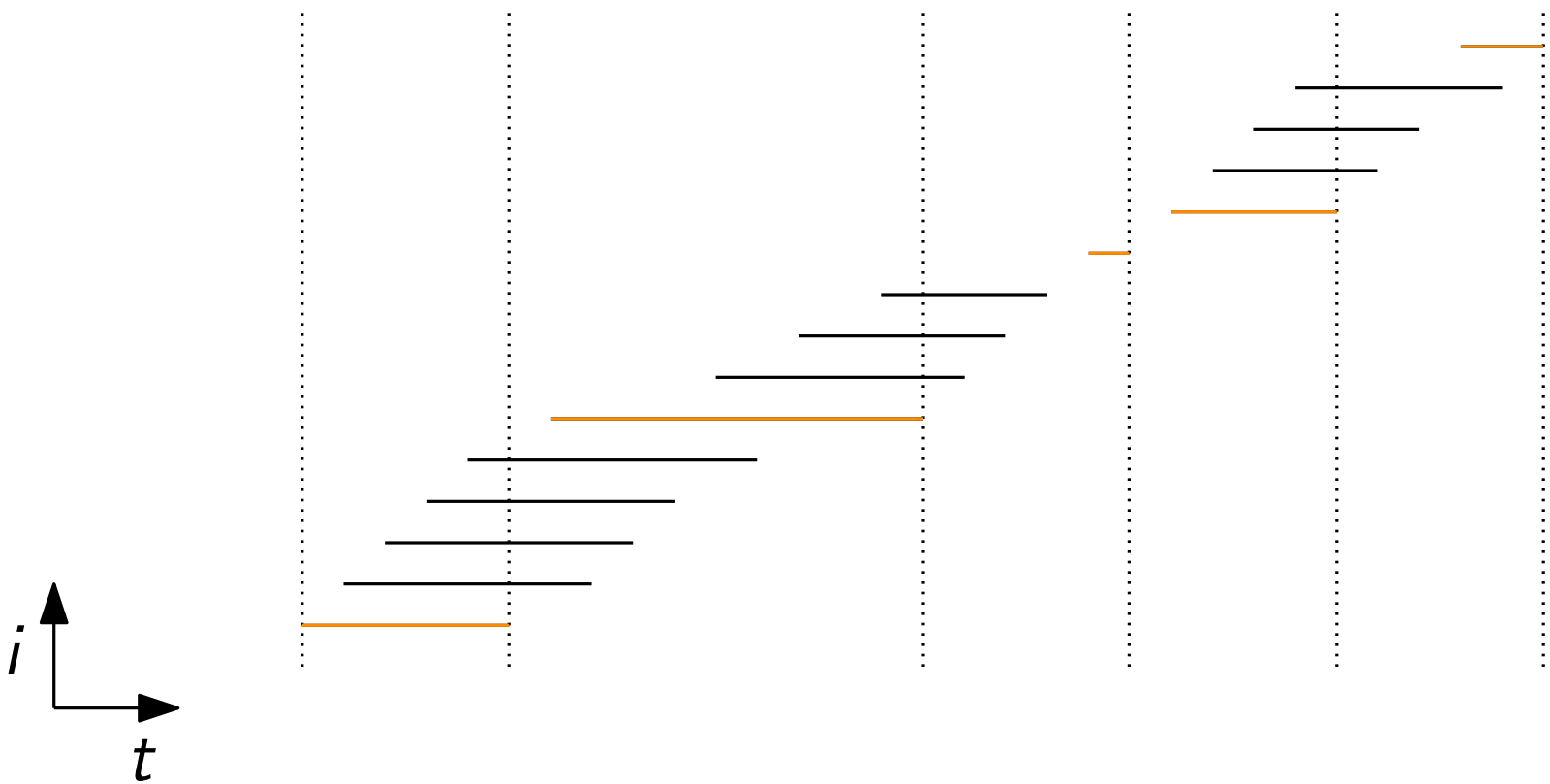
Hershberger's Combinatorial Result

- Plot each point p_i as segment $[(i(p_i), i), (d(p_i), i)]$
- Divide into regions s.t. no point is both inserted & deleted



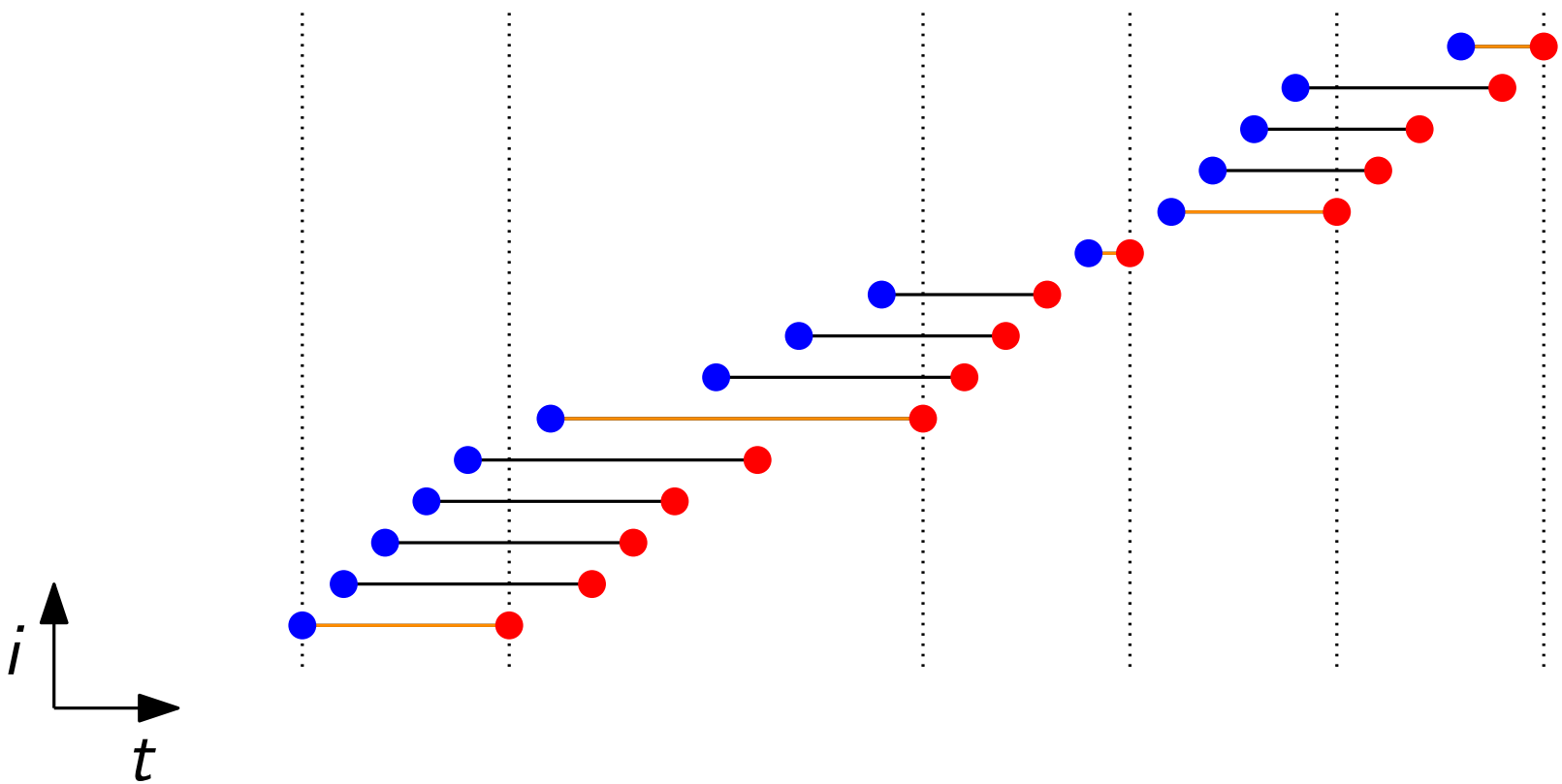
Hershberger's Combinatorial Result

- Plot each point p_i as segment $[(i(p_i), i), (d(p_i), i)]$
- Divide into regions s.t. no point is both inserted & deleted
- In each region, each point is either inserted or deleted



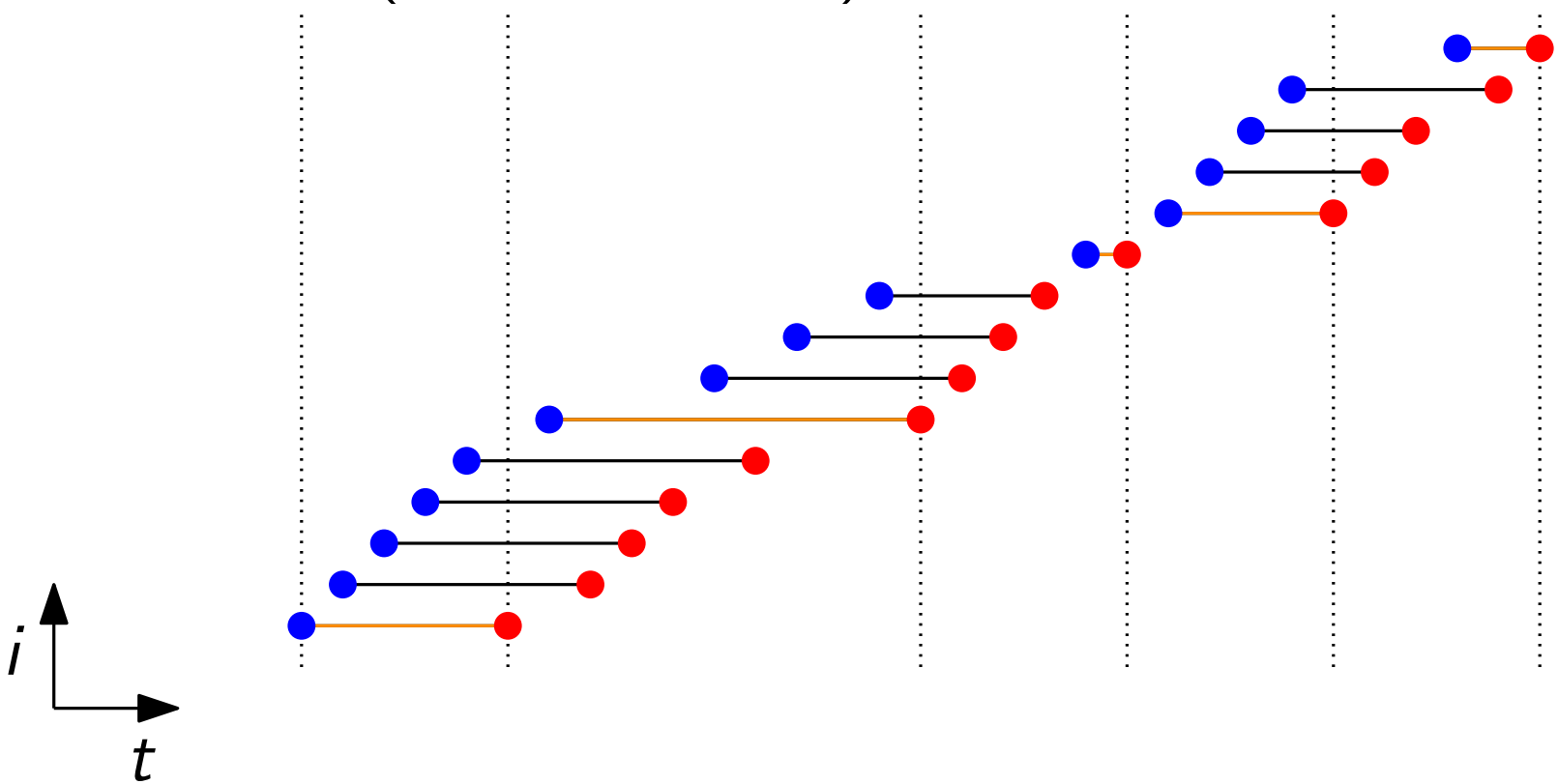
Hershberger's Combinatorial Result

- Plot each point p_i as segment $[(i(p_i), i), (d(p_i), i)]$
- Divide into regions s.t. no point is both inserted & deleted
- In each region, each point is either inserted or deleted



Hershberger's Combinatorial Result

- Plot each point p_i as segment $[(i(p_i), i), (d(p_i), i)]$
- Divide into regions s.t. no point is both inserted & deleted
- In each region, each point is either inserted or deleted
- 3 types of edges:
  - ◇ Insert-only
  - ◇ Delete-only
  - ◇ Mixed (1 ins. & 1 del.)



Hershberger: bounding insert-only

Hershberger: bounding insert-only

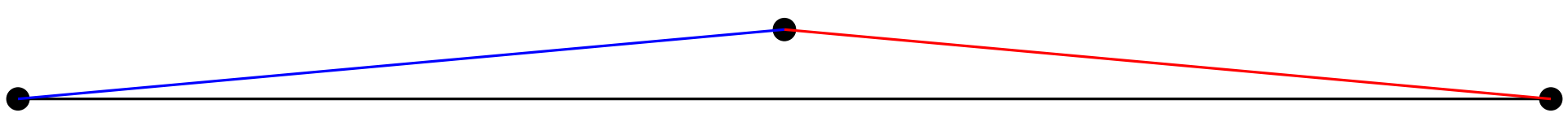
- Insertion-only: each point creates at most 2 edges
- Set I of points, #edges bounded by $2|I|$

Hershberger: bounding insert-only

- Insertion-only: each point creates at most 2 edges
- Set I of points, #edges bounded by $2|I|$

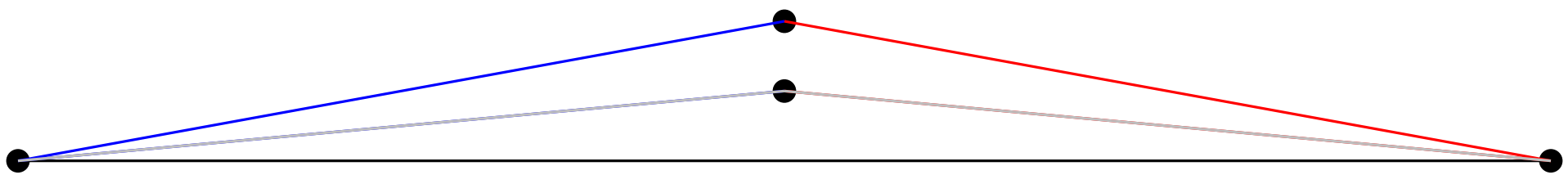
Hershberger: bounding insert-only

- Insertion-only: each point creates at most 2 edges
- Set I of points, #edges bounded by $2|I|$



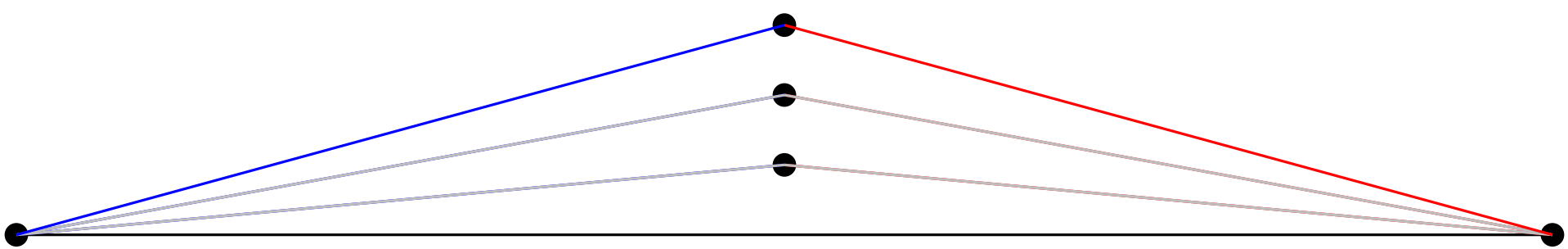
Hershberger: bounding insert-only

- Insertion-only: each point creates at most 2 edges
- Set I of points, #edges bounded by $2|I|$



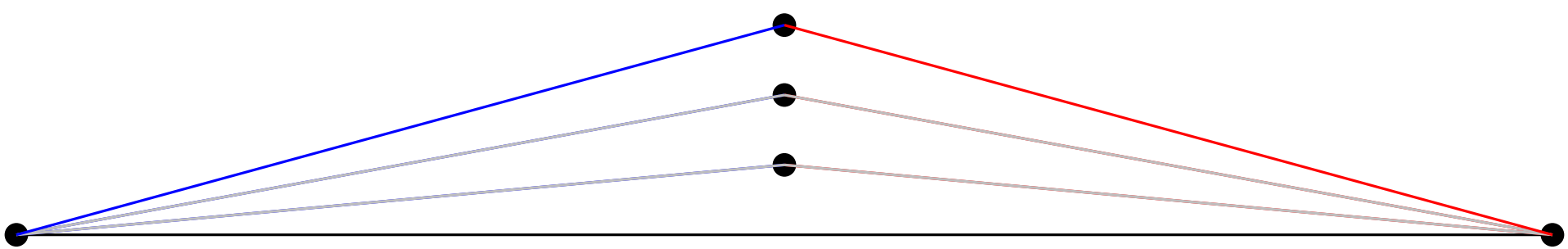
Hershberger: bounding insert-only

- Insertion-only: each point creates at most 2 edges
- Set I of points, #edges bounded by $2|I|$



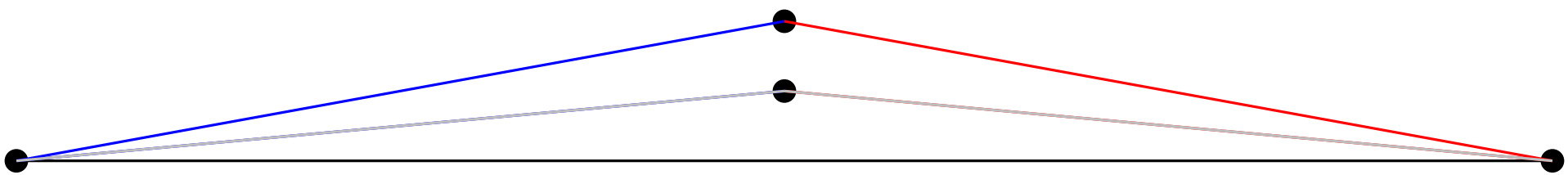
Hershberger: bounding insert-only

- Insertion-only: each point creates at most 2 edges
- Set I of points, #edges bounded by $2|I|$
- Deletion-only: same in reverse



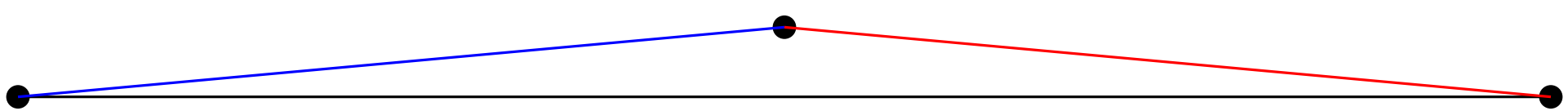
Hershberger: bounding insert-only

- Insertion-only: each point creates at most 2 edges
- Set I of points, #edges bounded by $2|I|$
- Deletion-only: same in reverse



Hershberger: bounding insert-only

- Insertion-only: each point creates at most 2 edges
- Set I of points, #edges bounded by $2|I|$
- Deletion-only: same in reverse



Hershberger: bounding insert-only

- Insertion-only: each point creates at most 2 edges
- Set I of points, #edges bounded by $2|I|$
- Deletion-only: same in reverse

Hershberger: bounding mixed edges

Hershberger: bounding mixed edges

- Given mixed edge (u, v), $u \in D$, $v \in I$

Hershberger: bounding mixed edges

- Given mixed edge (u, v), $u \in D$, $v \in I$
- Charge mixed edge's del. to an ins. point or a del. point

Hershberger: bounding mixed edges

- Given mixed edge (u, v), $u \in D$, $v \in I$
- Charge mixed edge's del. to an ins. point or a del. point
- Deletion of (u, v) can be due to insertion or deletion

Hershberger: bounding mixed edges

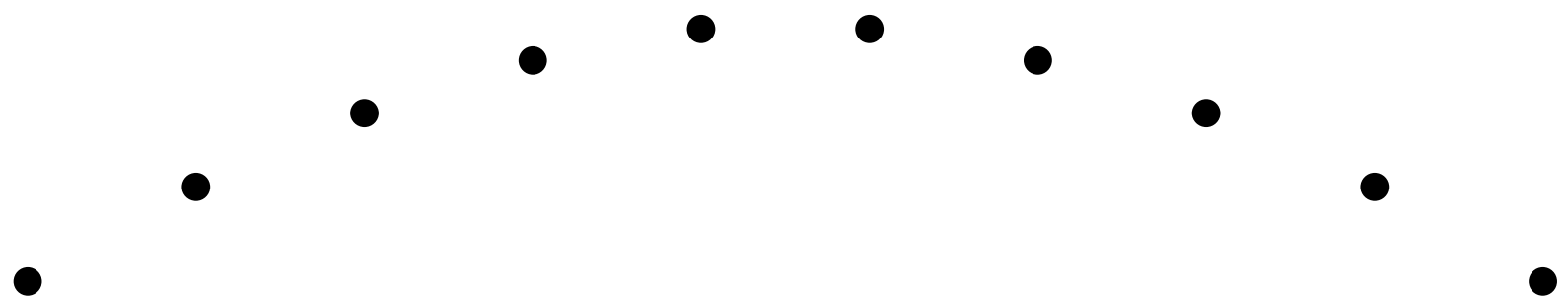
- Given mixed edge (u, v), $u \in D$, $v \in I$
- Charge mixed edge's del. to an ins. point or a del. point
- Deletion of (u, v) can be due to insertion or deletion
- If due to deletion, must be deletion of u (charge to u)

Hershberger: bounding mixed edges

- Given mixed edge (u, v), $u \in D$, $v \in I$
- Charge mixed edge's del. to an ins. point or a del. point
- Deletion of (u, v) can be due to insertion or deletion
- If due to deletion, must be deletion of u (charge to u)
- Otherwise, must be due to insertion of point x

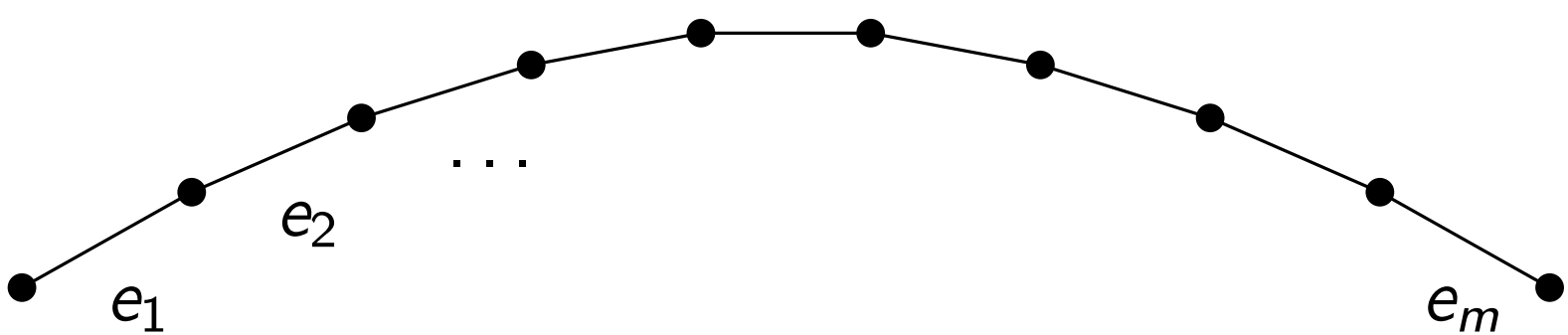
Hershberger: bounding mixed edges

- Given mixed edge (u, v), $u \in D$, $v \in I$
- Charge mixed edge's del. to an ins. point or a del. point
- Deletion of (u, v) can be due to insertion or deletion
- If due to deletion, must be deletion of u (charge to u)
- Otherwise, must be due to insertion of point x



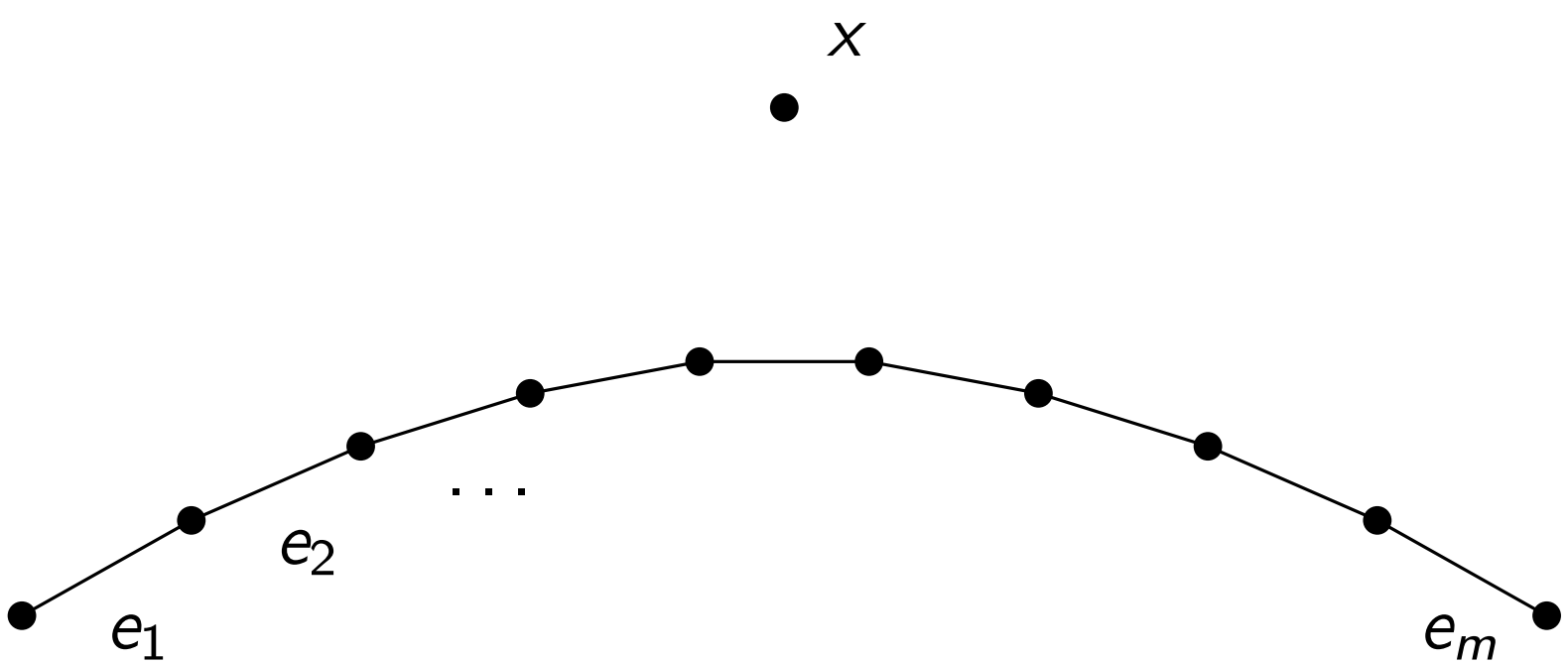
Hershberger: bounding mixed edges

- Given mixed edge (u, v), $u \in D$, $v \in I$
- Charge mixed edge's del. to an ins. point or a del. point
- Deletion of (u, v) can be due to insertion or deletion
- If due to deletion, must be deletion of u (charge to u)
- Otherwise, must be due to insertion of point x



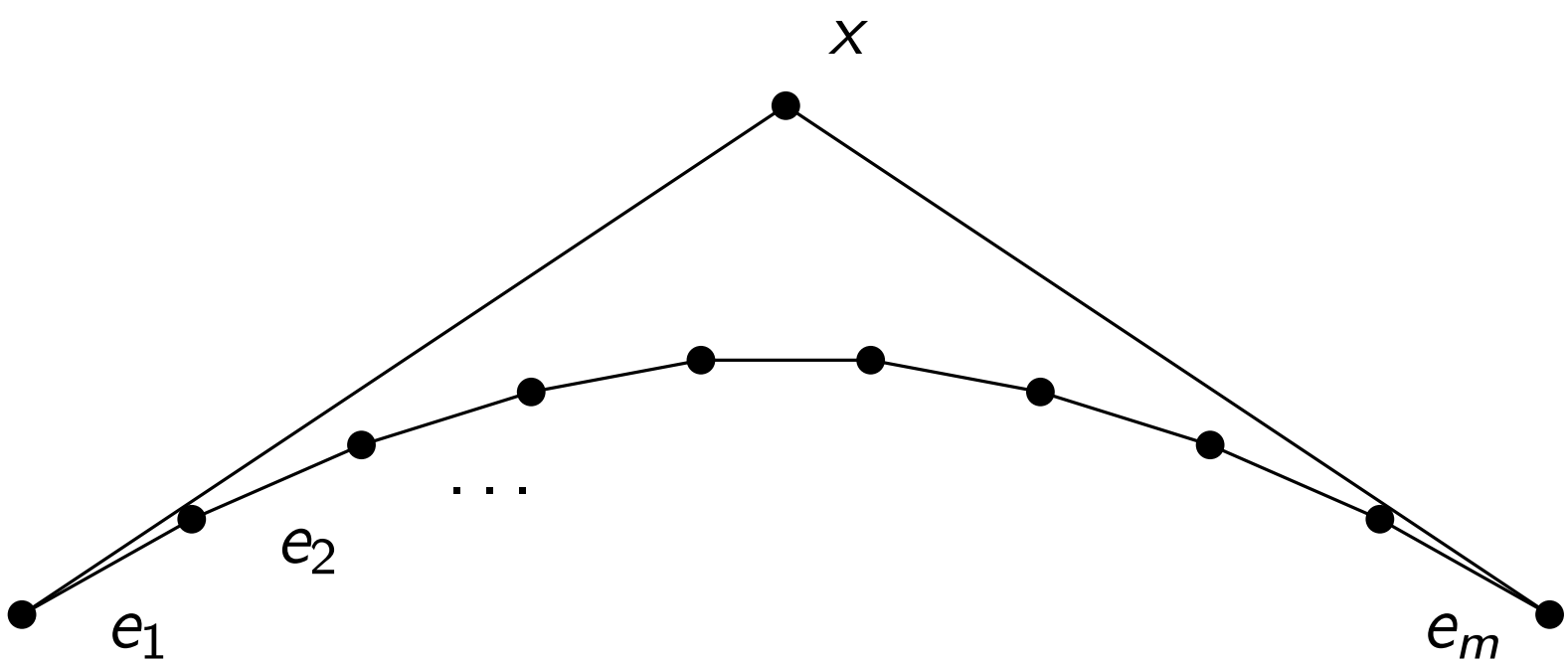
Hershberger: bounding mixed edges

- Given mixed edge (u, v), $u \in D$, $v \in I$
- Charge mixed edge's del. to an ins. point or a del. point
- Deletion of (u, v) can be due to insertion or deletion
- If due to deletion, must be deletion of u (charge to u)
- Otherwise, must be due to insertion of point x



Hershberger: bounding mixed edges

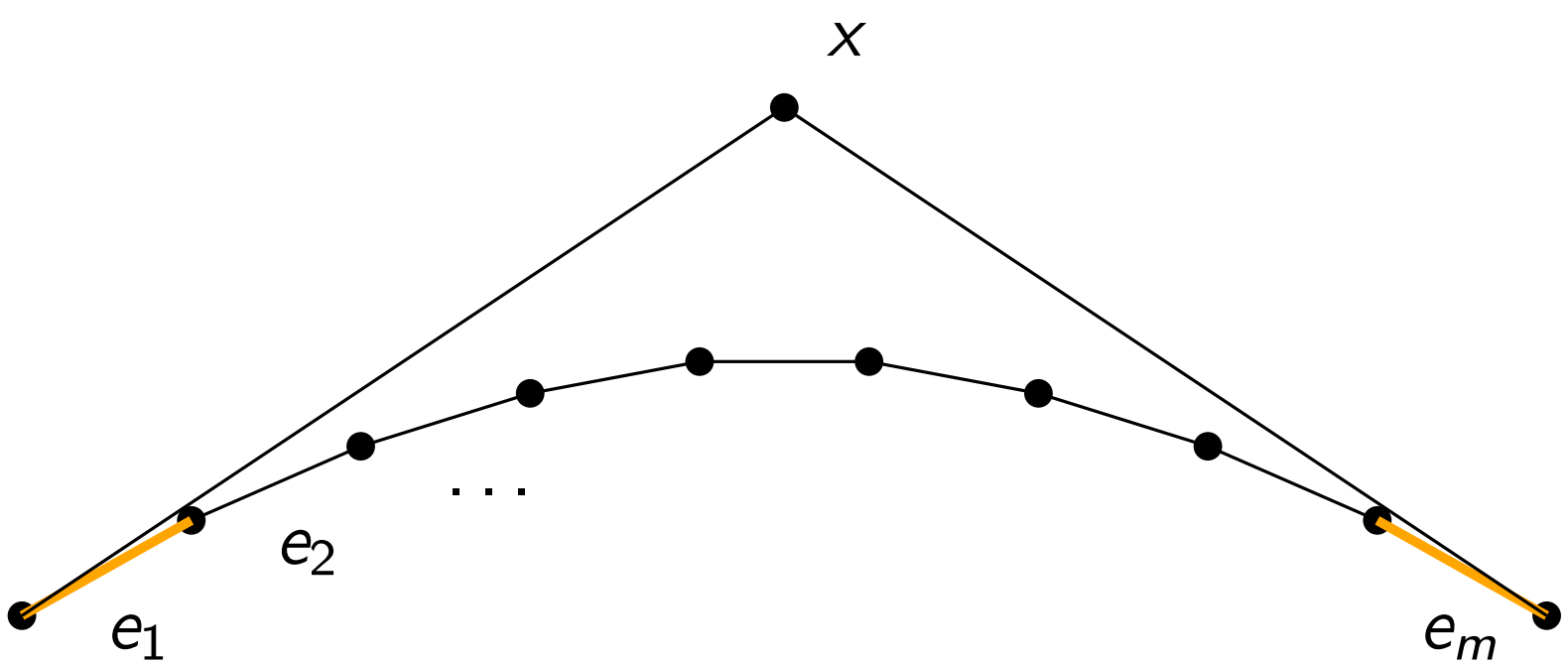
- Given mixed edge (u, v), $u \in D$, $v \in I$
- Charge mixed edge's del. to an ins. point or a del. point
- Deletion of (u, v) can be due to insertion or deletion
- If due to deletion, must be deletion of u (charge to u)
- Otherwise, must be due to insertion of point x



Hershberger: bounding mixed edges

- Given mixed edge (u, v), $u \in D$, $v \in I$
- Charge mixed edge's del. to an ins. point or a del. point
- Deletion of (u, v) can be due to insertion or deletion
- If due to deletion, must be deletion of u (charge to u)
- Otherwise, must be due to insertion of point x

— charge to x

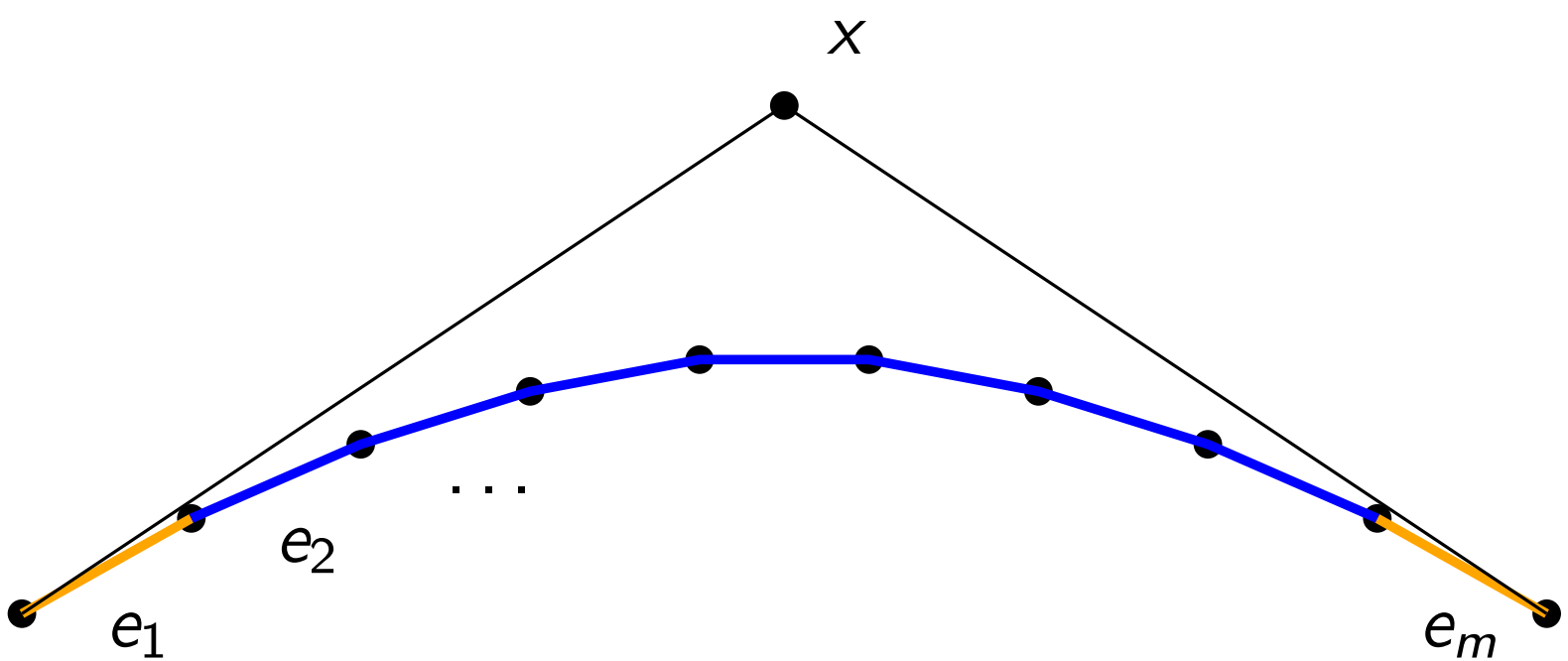


Hershberger: bounding mixed edges

- Given mixed edge (u, v), $u \in D$, $v \in I$
- Charge mixed edge's del. to an ins. point or a del. point
- Deletion of (u, v) can be due to insertion or deletion
- If due to deletion, must be deletion of u (charge to u)
- Otherwise, must be due to insertion of point x

— charge to x

— charge (u', v') to u'



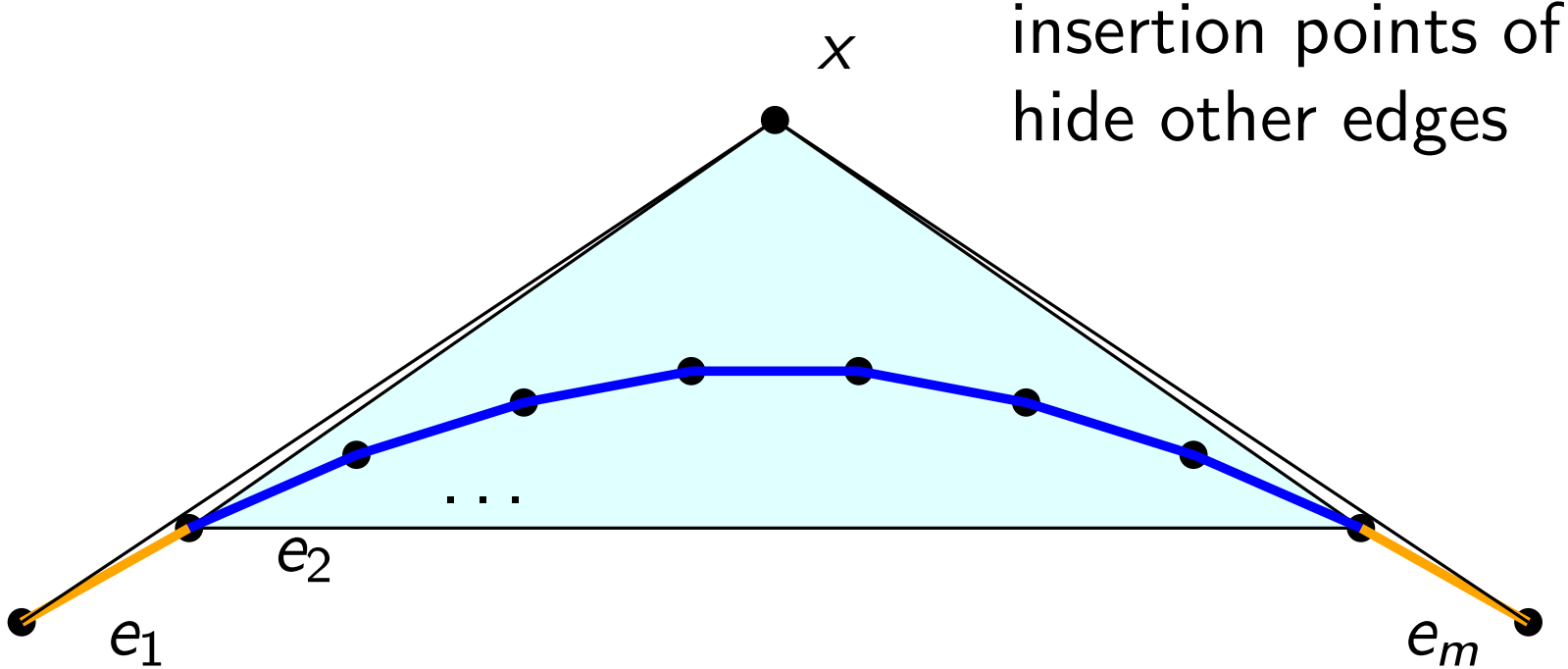
Hershberger: bounding mixed edges

- Given mixed edge (u, v), $u \in D$, $v \in I$
- Charge mixed edge's del. to an ins. point or a del. point
- Deletion of (u, v) can be due to insertion or deletion
- If due to deletion, must be deletion of u (charge to u)
- Otherwise, must be due to insertion of point x

— charge to x

— charge (u', v') to u'

- No double-counting since convex hull of x and insertion points of e_1, e_m hide other edges



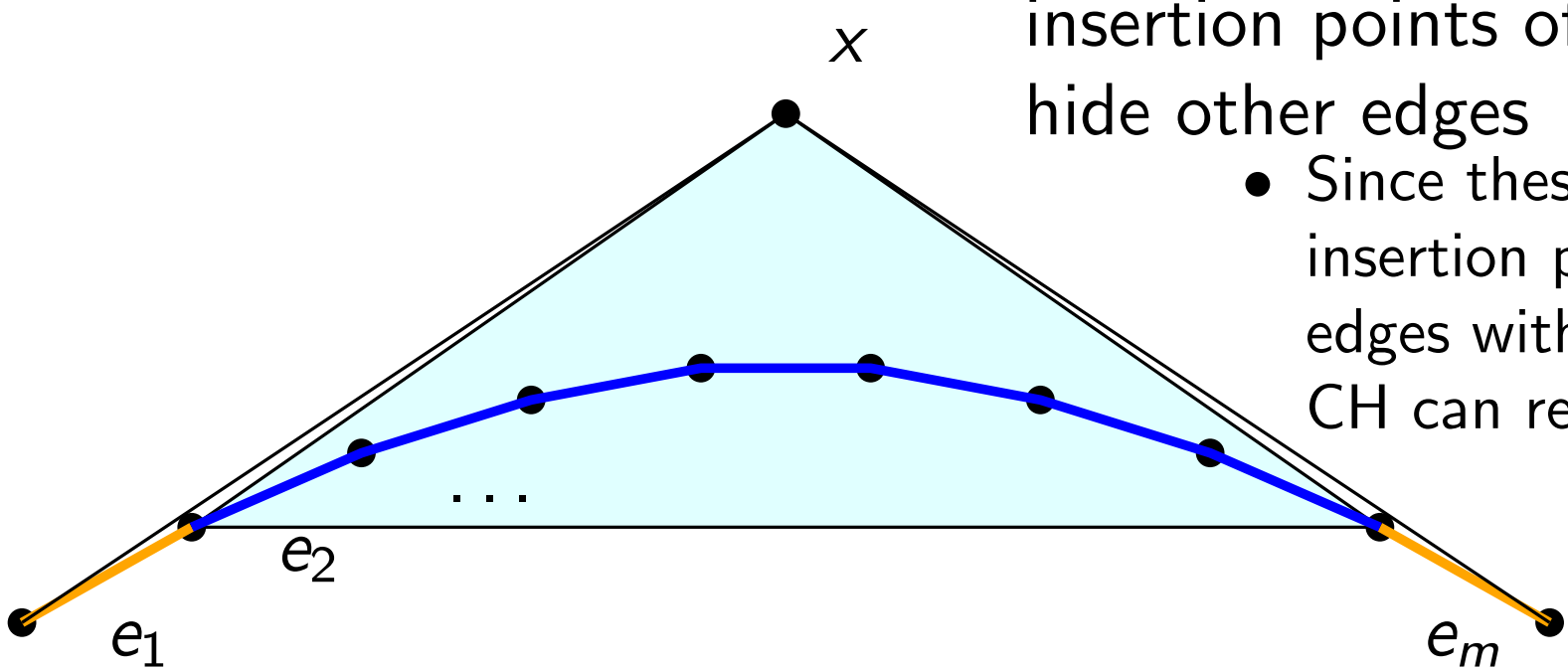
Hershberger: bounding mixed edges

- Given mixed edge (u, v), $u \in D$, $v \in I$
- Charge mixed edge's del. to an ins. point or a del. point
- Deletion of (u, v) can be due to insertion or deletion
- If due to deletion, must be deletion of u (charge to u)
- Otherwise, must be due to insertion of point x

— charge to x

— charge (u', v') to u'

- No double-counting since convex hull of x and insertion points of e_1, e_m hide other edges
  - Since these are insertion points, no edges within their CH can re-appear.



New Results

Approach #1: Grids & Quadtrees

Problem	Preprocessing	Space	Query
CP Decision	$O(n)$ exp.	$O(n)$ bits	$O(1)$
Closest Pair	$O(n \lg n \lg \lg n)$	$O(n \lg n)$	$O(\lg \lg n)$

Approach #2: Generalized Range Successor

Problem	Preprocessing Time
2D diameter decision	$O(n \lg^2 n)$
$\rightarrow$ improved to	$O(n \lg n)$
3D diameter decision	$O(n \lg^2 n)$
2D orth. seg. intersection detection	$O(n \lg n \lg \lg n)$

Approach #3: FIFO Update Sequence

Problem	Preprocessing Time
2D convex hull area decision	$O(n \lg n \lg \lg n)$
$\rightarrow$ improved to	$O(n\alpha(n) \lg n)$
2D width decision	$O(n \lg^7 n)$

New Results

Approach #1: Grids & Quadtrees

Problem	Preprocessing	Space	Query
CP Decision	$O(n)$ exp.	$O(n)$ bits	$O(1)$
Closest Pair	$O(n \lg n \lg \lg n)$	$O(n \lg n)$	$O(\lg \lg n)$

Approach #2: Generalized Range Successor

Problem	Preprocessing Time
2D diameter decision	$O(n \lg^2 n)$
$\rightarrow$ improved to	$O(n \lg n)$
3D diameter decision	$O(n \lg^2 n)$
2D orth. seg. intersection detection	$O(n \lg n \lg \lg n)$

Approach #3: FIFO Update Sequence

Problem	Preprocessing Time
2D convex hull area decision	$O(n \lg n \lg \lg n)$
$\rightarrow$ improved to	$O(n \lg n)$
2D width decision	$O(n \lg^7 n)$

New Results

Approach #1: Grids & Quadtrees

Problem	Preprocessing	Space	Query
CP Decision	$O(n)$ exp.	$O(n)$ bits	$O(1)$
Closest Pair	$O(n \lg n \lg \lg n)$	$O(n \lg n)$	$O(\lg \lg n)$

Approach #2: Generalized Range Successor

Problem	Preprocessing Time
2D diameter decision	$O(n \lg^2 n)$
$\rightarrow$ improved to	$O(n \lg n)$
3D diameter decision	$O(n \lg^2 n)$
2D orth. seg. intersection detection	$O(n \lg n \lg \lg n)$

Approach #3: FIFO Update Sequence

Problem	Preprocessing Time
2D convex hull area decision	$O(n \lg n \lg \lg n)$
$\rightarrow$ improved to	$O(n \lg n)$
2D width decision	$O(n \lg^6 n)$