Two Approaches to Building Time-Windowed Geometric Data Structures

Timothy M. Chan and Simon Pratt

Cheriton School of Computer Science
University of Waterloo, Canada

SoCG 2016

Problem Statement

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

given a query window $[t_1, t_2]$, determine whether the subset of objects whose time values are within that window have a given property.

Problem Statement

e.g., points or segments

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

given a query window $[t_1, t_2]$, determine whether the subset of objects whose time values are within that window have a given property.

Problem Statement

e.g., points or segments

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

given a query window $[t_1, t_2]$, determine whether the subset of objects whose time values are within that window have a given property.

Problem Statement

e.g., points or segments

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

given a query window $[t_1, t_2]$, determine whether the subset of objects whose time values are within that window have a given property.

e.g. diameter > 1

Motivation

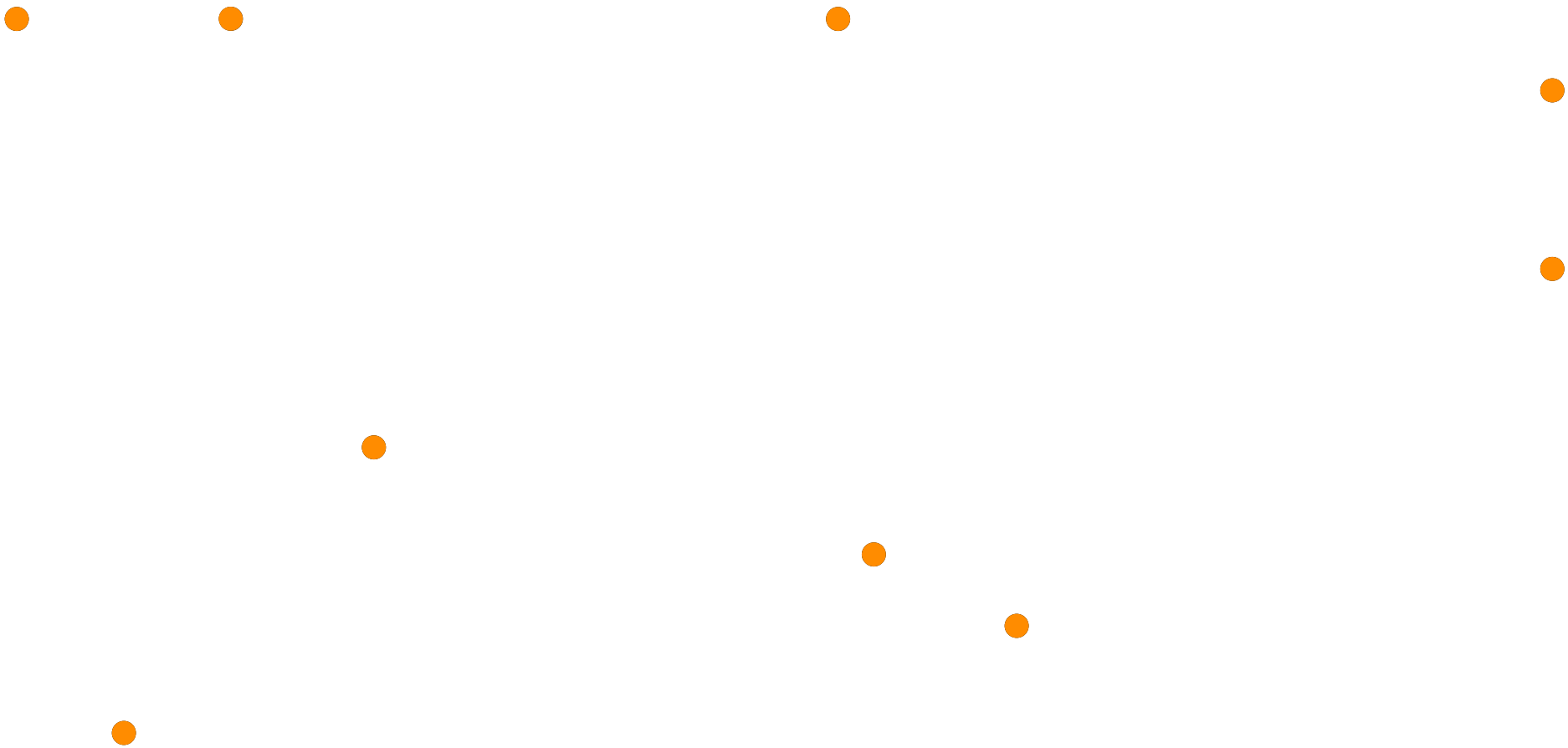
- *Geographic Information Systems* (GIS) data may consist not only of latitude, longitude, and altitude coordinates but also time
- Timestamped data (for example, social network data)

Example: Time-Windowed Diameter Decision

Preprocess a set of n time-labeled points, to efficiently determine if the points within the query time window have diameter > 1.

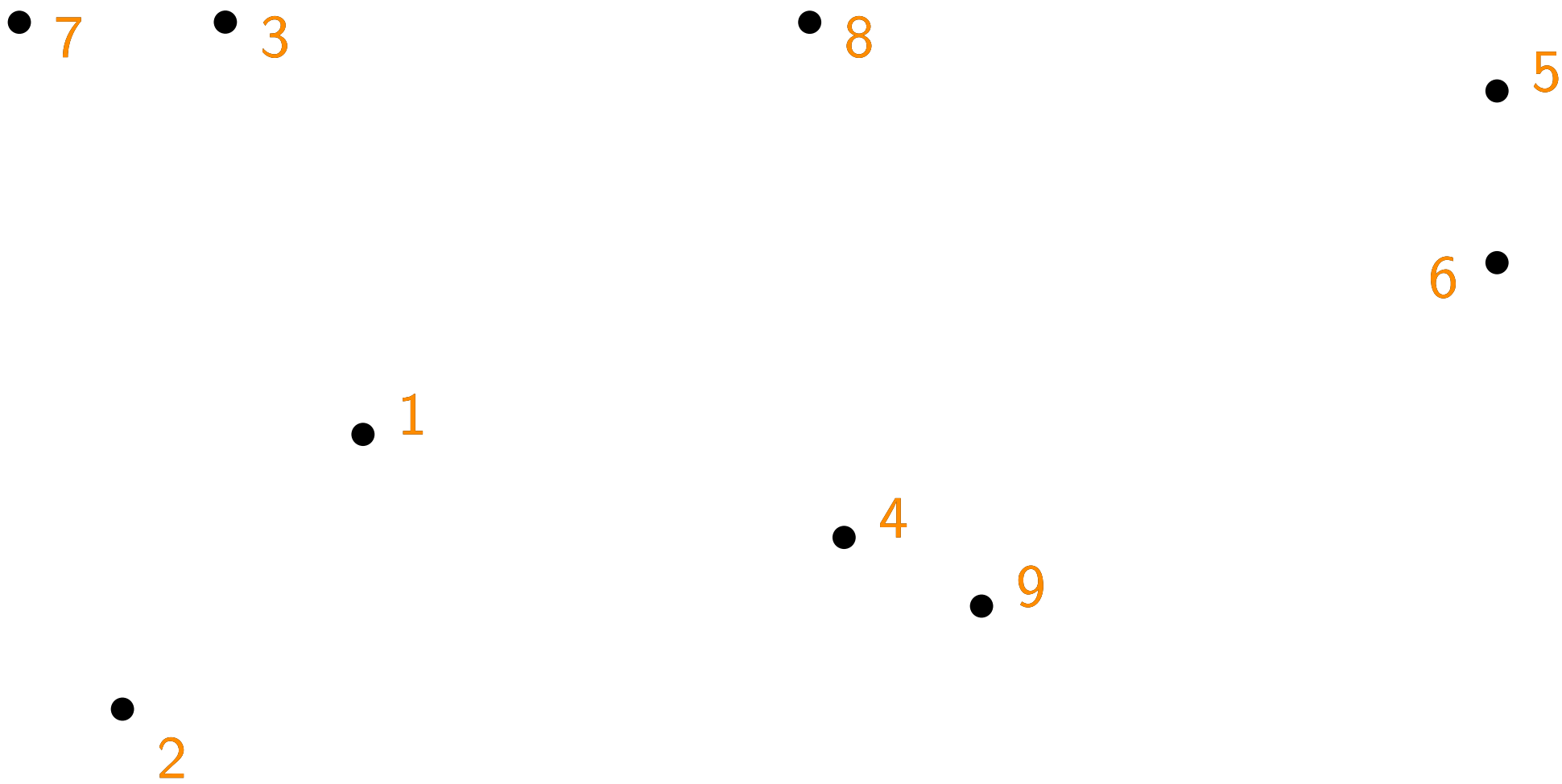
Example: Time-Windowed Diameter Decision

Preprocess a set of n time-labeled points, to efficiently determine if the points within the query time window have diameter > 1.



Example: Time-Windowed Diameter Decision

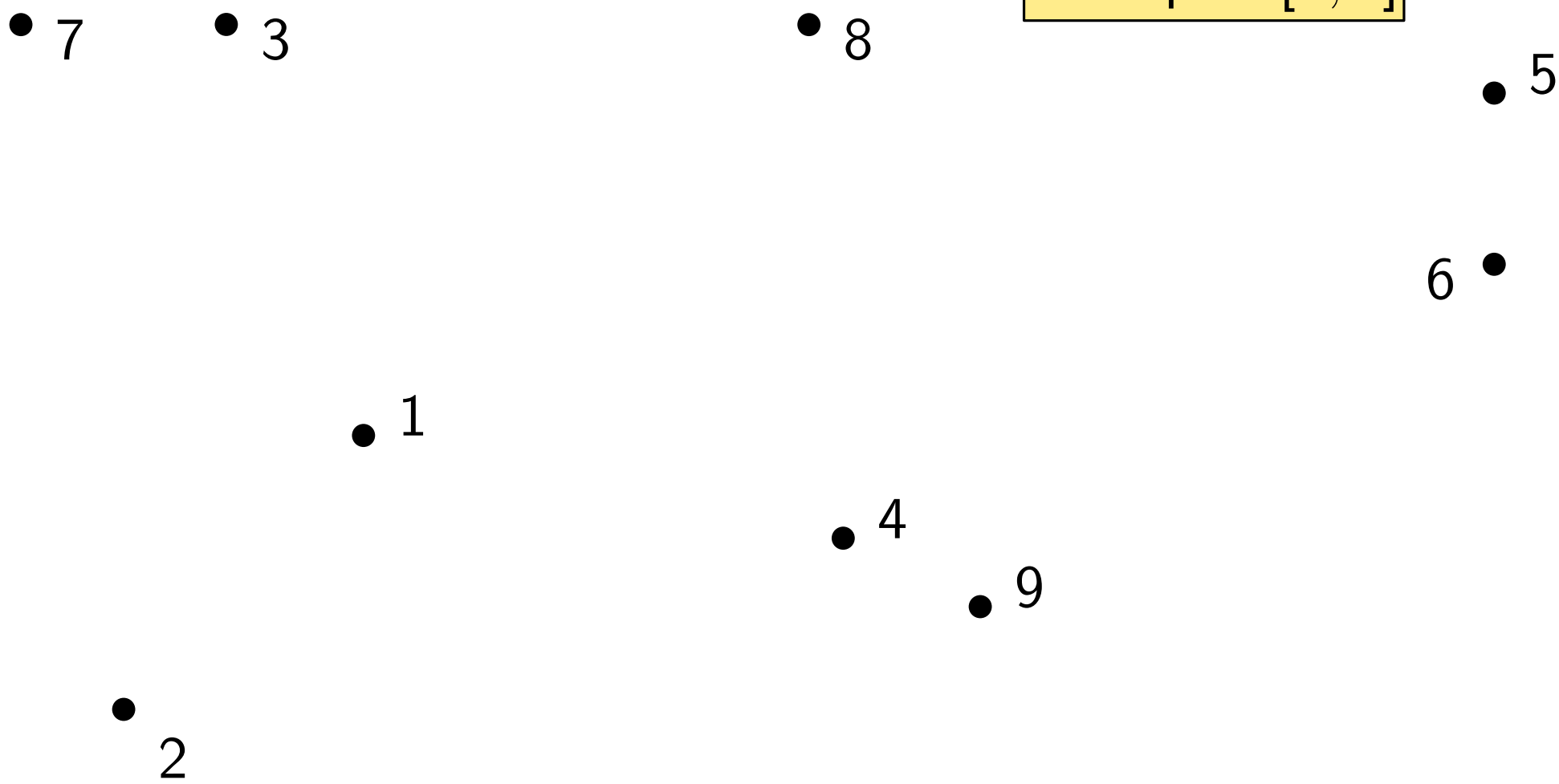
Preprocess a set of n time-labeled points, to efficiently determine if the points within the query time window have diameter > 1.



Example: Time-Windowed Diameter Decision

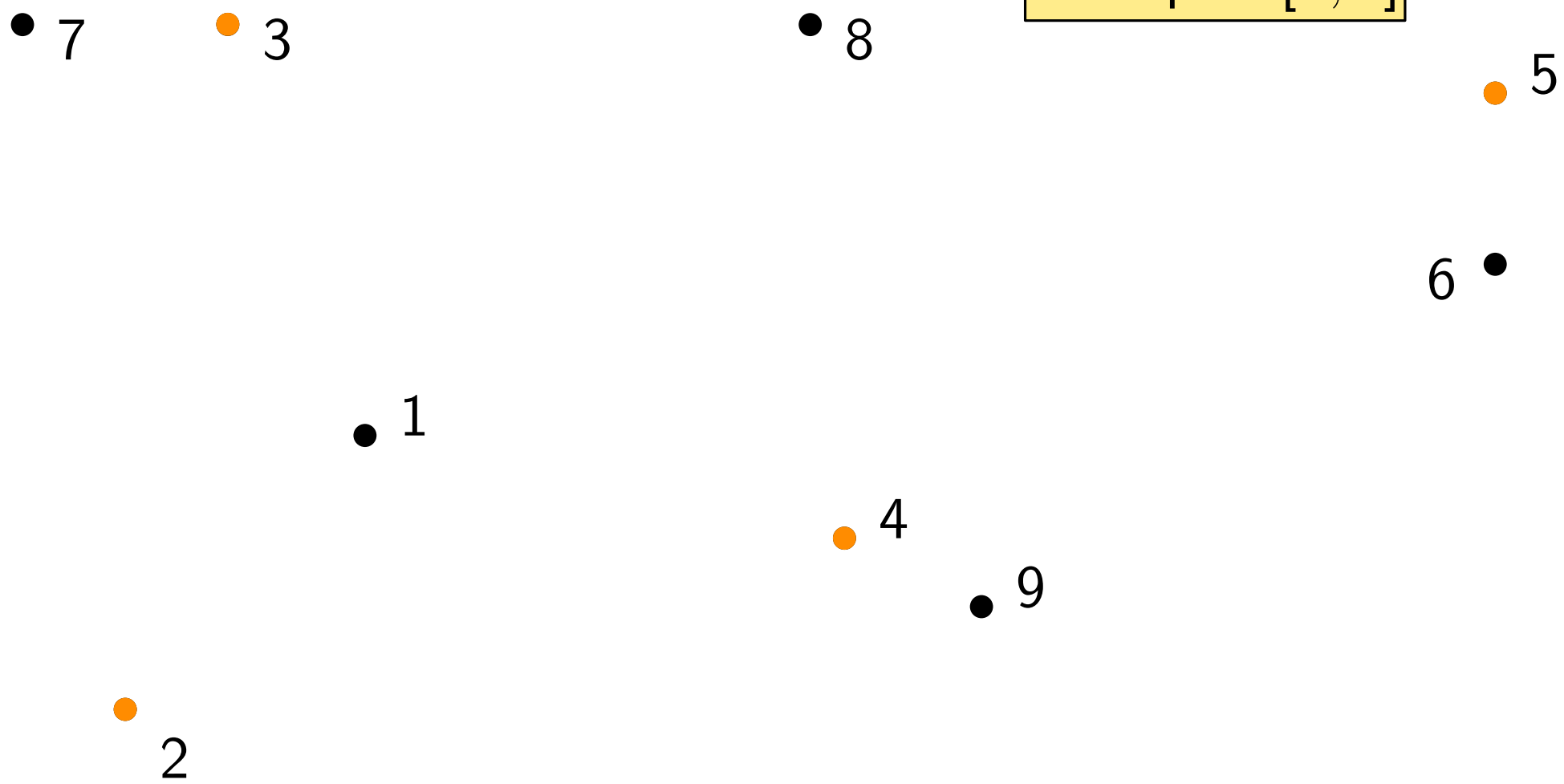
Preprocess a set of n time-labeled points, to efficiently determine if the points within the query time window have diameter > 1.

Example: $[2, 5]$



Example: Time-Windowed Diameter Decision

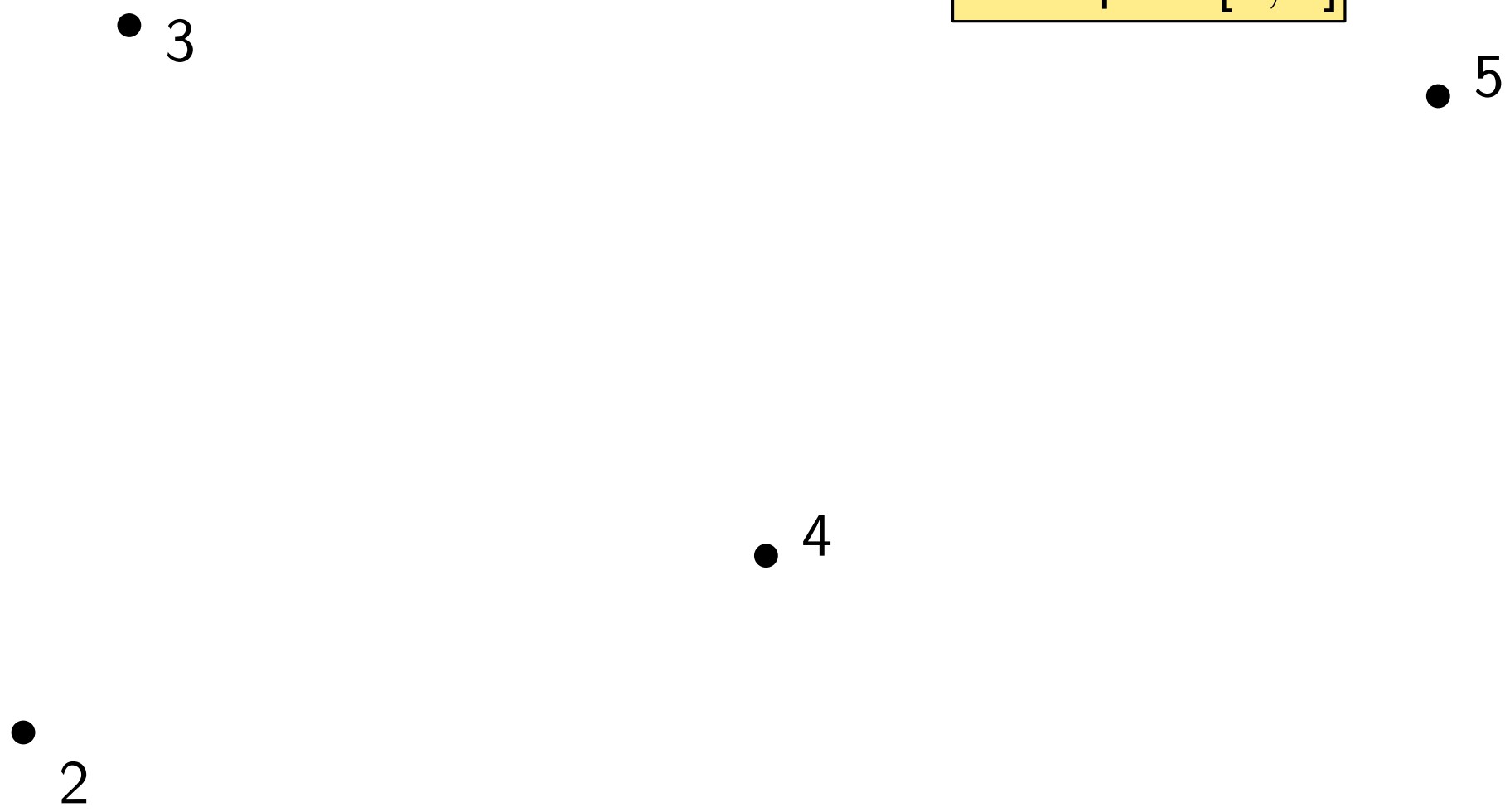
Preprocess a set of n time-labeled points, to efficiently determine if the points within the query time window have diameter > 1.



Example: Time-Windowed Diameter Decision

Preprocess a set of n time-labeled points, to efficiently determine if the points within the query time window have diameter > 1.

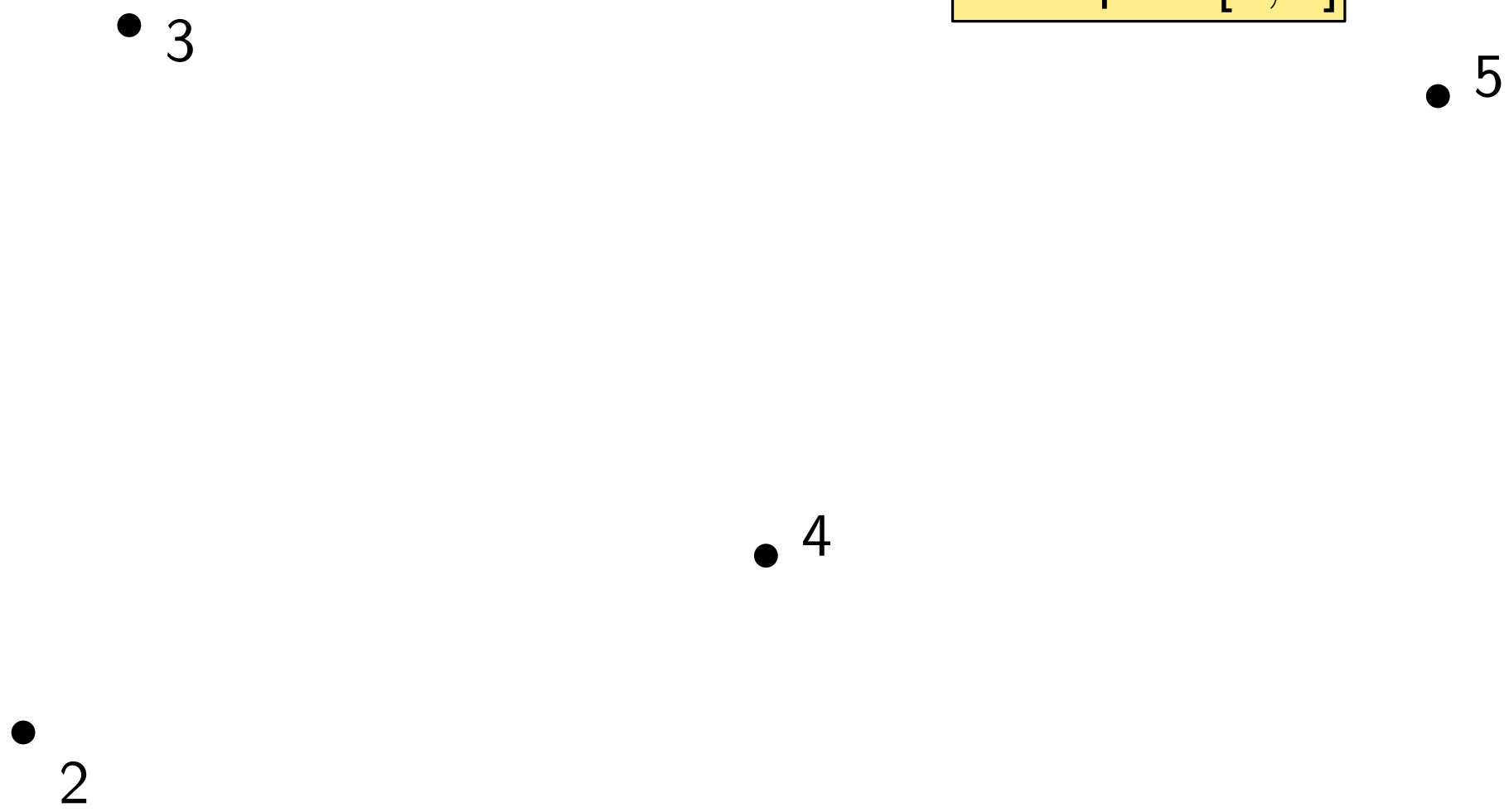
Example: $[2, 5]$



Example: Time-Windowed Diameter Decision

Preprocess a set of n time-labeled points, to efficiently determine if the points within the query time window have diameter > 1.

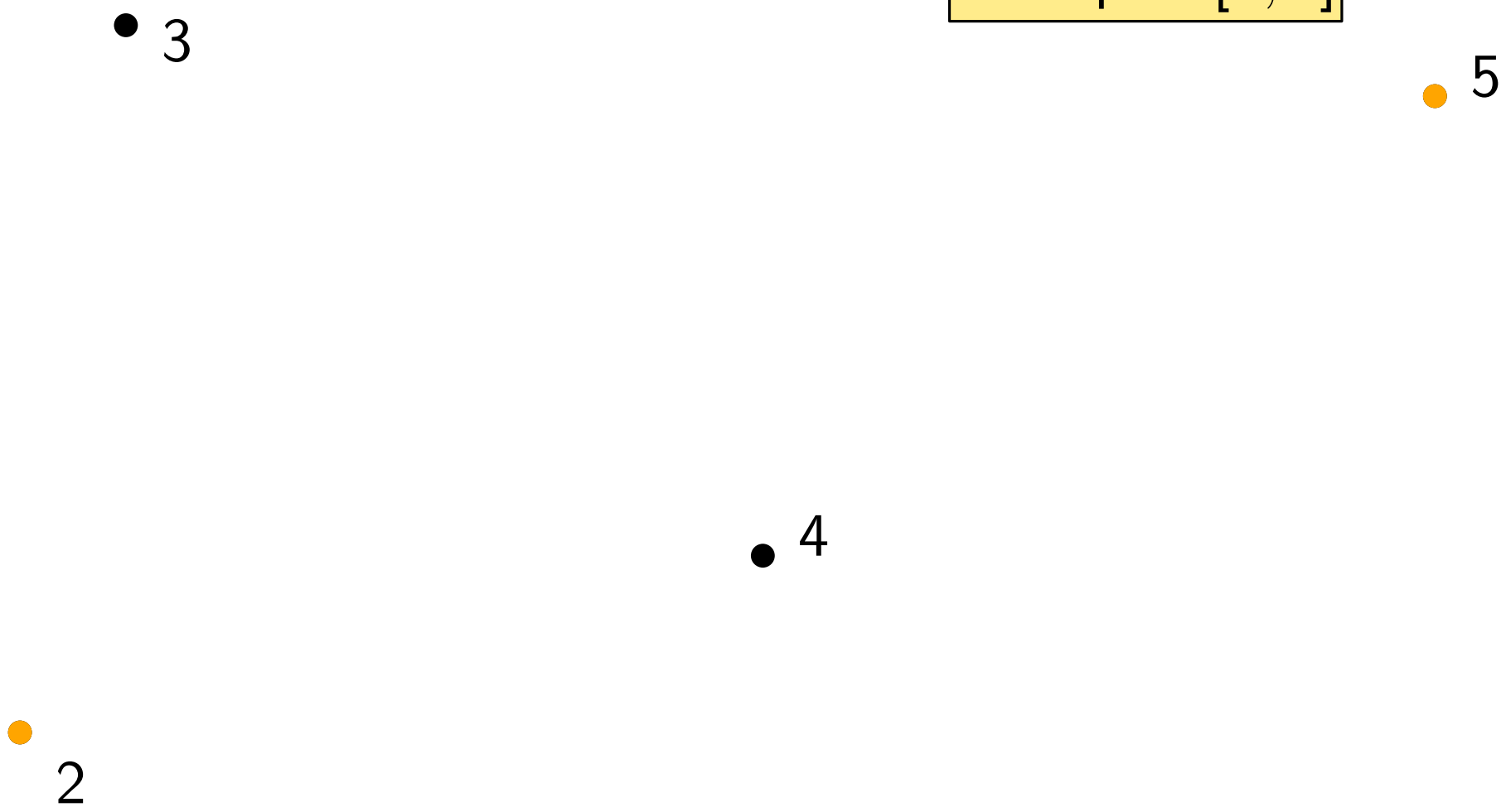
Example: $[2, 5]$



Example: Time-Windowed Diameter Decision

Preprocess a set of n time-labeled points, to efficiently determine if the points within the query time window have diameter > 1.

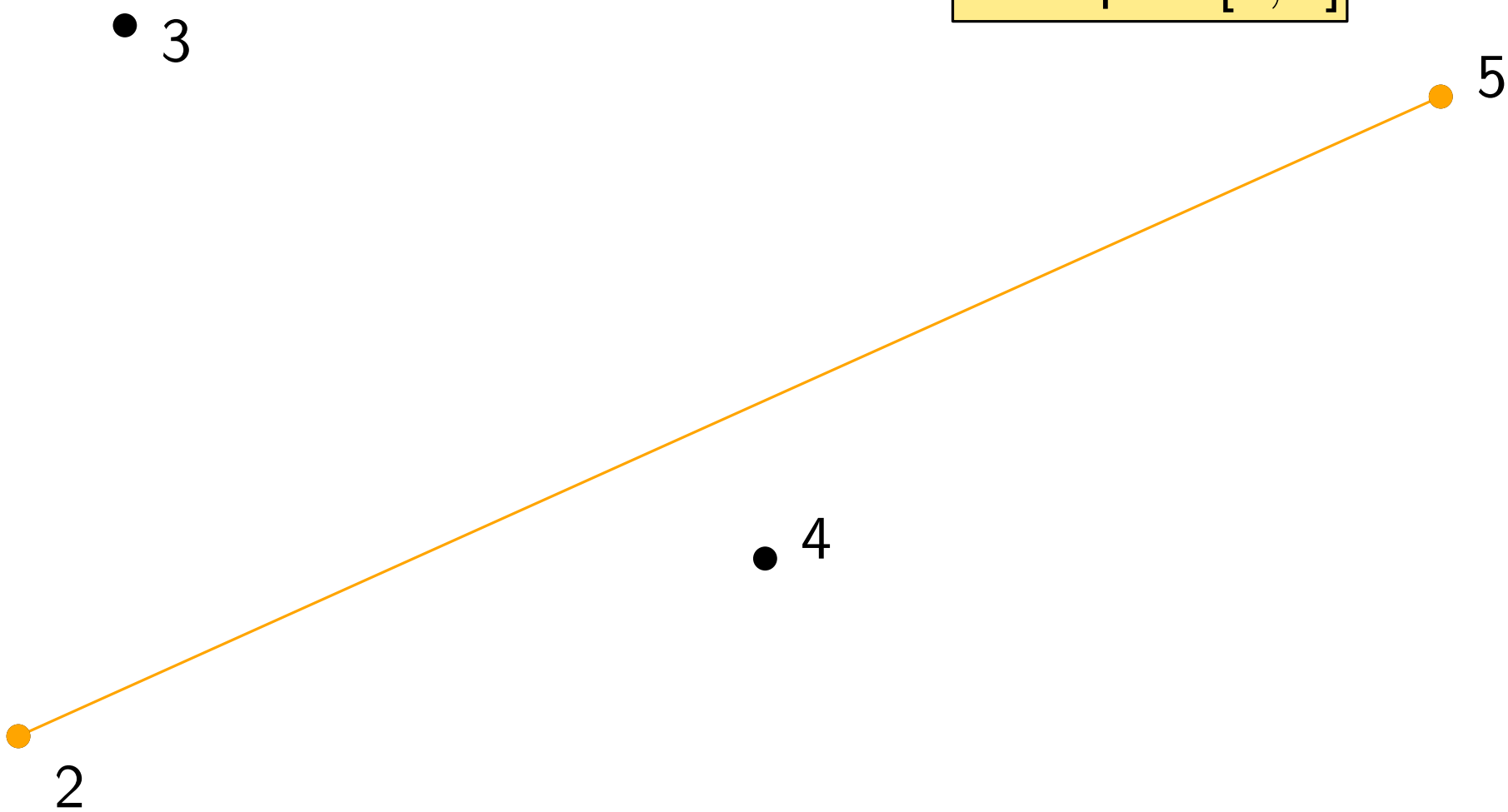
Example: $[2, 5]$



Example: Time-Windowed Diameter Decision

Preprocess a set of n time-labeled points, to efficiently determine if the points within the query time window have diameter > 1.

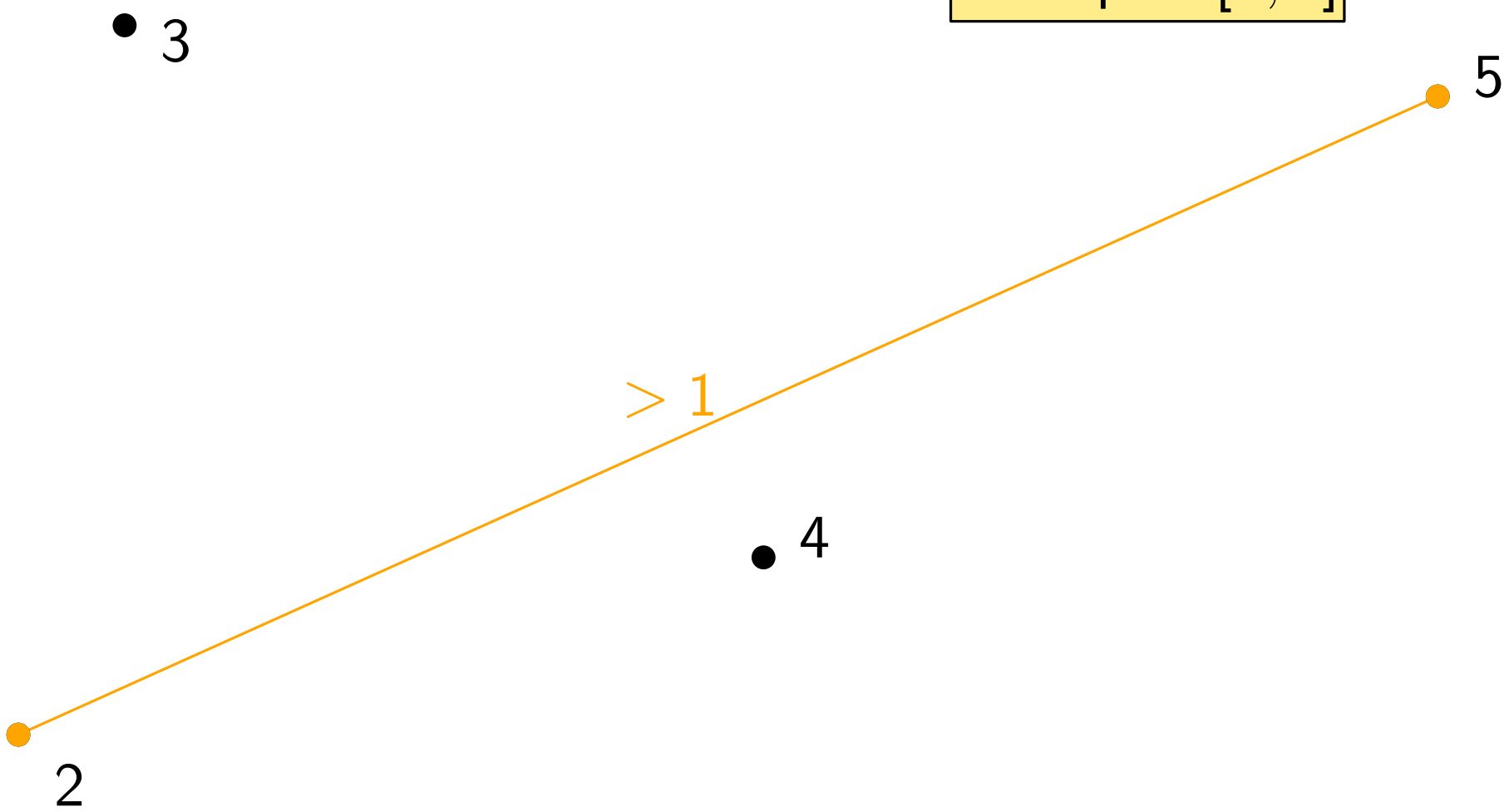
Example: $[2, 5]$



Example: Time-Windowed Diameter Decision

Preprocess a set of n time-labeled points, to efficiently determine if the points within the query time window have diameter > 1.

Example: $[2, 5]$



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 $\mathcal{P}$.

For each i:

i
1
$\vdots$
i
$\vdots$
n

Hereditary Properties

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

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

Query: $[i, k]$

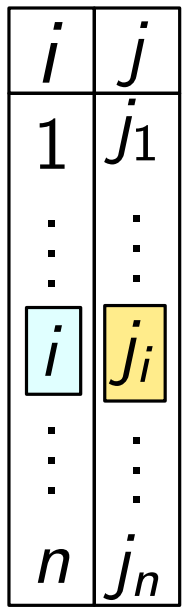
i	j
1	j_1
⋮	⋮
i	j_i
⋮	⋮
n	j_n

Hereditary Properties

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

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

Query: $[i, k]$



Hereditary Properties

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

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

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

$O(n)$ bits
$O(1)$ query time

Hereditary Properties

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

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

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

$O(n)$ bits
$O(1)$ query time

Henceforth, only concerned with preprocessing.

Previous Results

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

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

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

Previous Results

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

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

Hereditary

Previous Results

Chan and Pratt; *CCCG 2015*:

- Time-windowed closest pair decision problem
  - Preprocessing: $O(n)$
  - Query: $O(1)$
  - Space: $O(n)$ *bits* of space
- Time-windowed closest pair
  - Preprocessing: $O(n \log n \log \log n)$
  - Query: $O(\log n \log \log n)$
  - Space: $O(n \log n)$

Previous Results

Chan and Pratt; *CCCG 2015*:

- Time-windowed closest pair decision problem
  - Preprocessing: $O(n)$
  - Query: $O(1)$
  - Space: $O(n)$ *bits* of space
- Time-windowed closest pair
  - Preprocessing: $O(n \log n \log \log n)$
  - Query: $O(\log n \log \log n)$
  - Space: $O(n \log n)$

Hereditary

Previous Results

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: $O(n \log n \log \log n)$
  - 2D monotone paths
    - Preprocessing: $O(n)$

Previous Results

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: $O(n \log n \log \log n)$
  - 2D monotone paths
    - Preprocessing: $O(n)$

New Results

Approach #1: Generalized Range Successor

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

New Results

Approach #1: Generalized Range Successor

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

Approach #2: FIFO Update Sequence

Problem	Preprocessing Time
2D convex hull area decision	$O(n \log n \log \log n)$
$\rightarrow$ improved to	$O(n\alpha(n) \log n)$
2D width decision	$O(n \log^8 n)$

New Results

Approach #1: Generalized Range Successor

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

Approach #2: FIFO Update Sequence

Problem	Preprocessing Time
2D convex hull area decision	$O(n \log n \log \log n)$
$\rightarrow$ improved to	$O(n\alpha(n) \log n)$
2D width decision	$O(n \log^8 n)$

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.

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

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

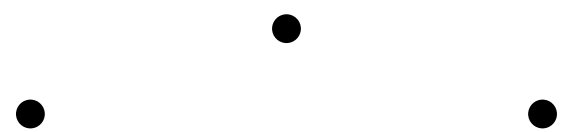
Generalized Range Successor

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. $d(p, q) > 1$?



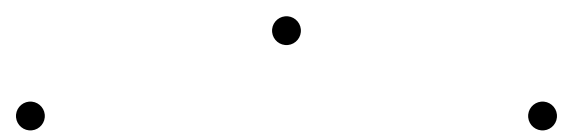
Generalized Range Successor

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

Generalized Range Successor

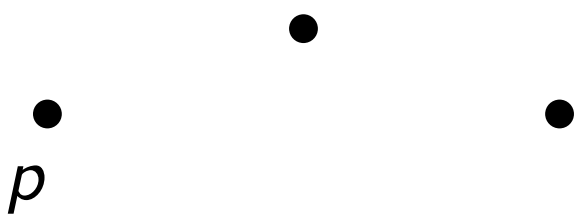
The *generalized range successor problem* is to preprocess the input to find the *successor* of each object among that object's range.

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



Generalized Range Successor

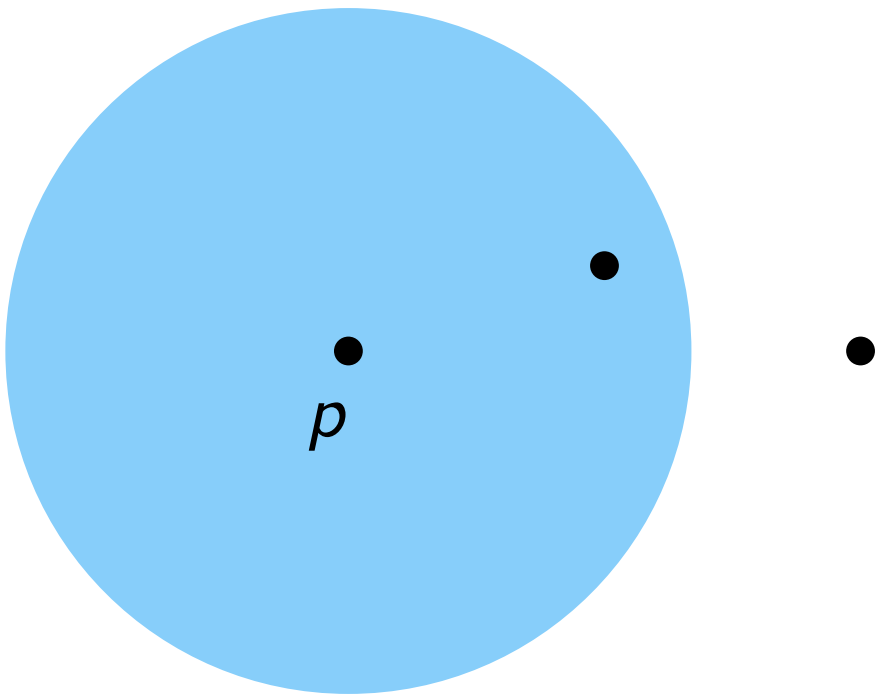
The *generalized range successor problem* is to preprocess the input to find the *successor* of each object among that object's range.

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



Generalized Range Successor

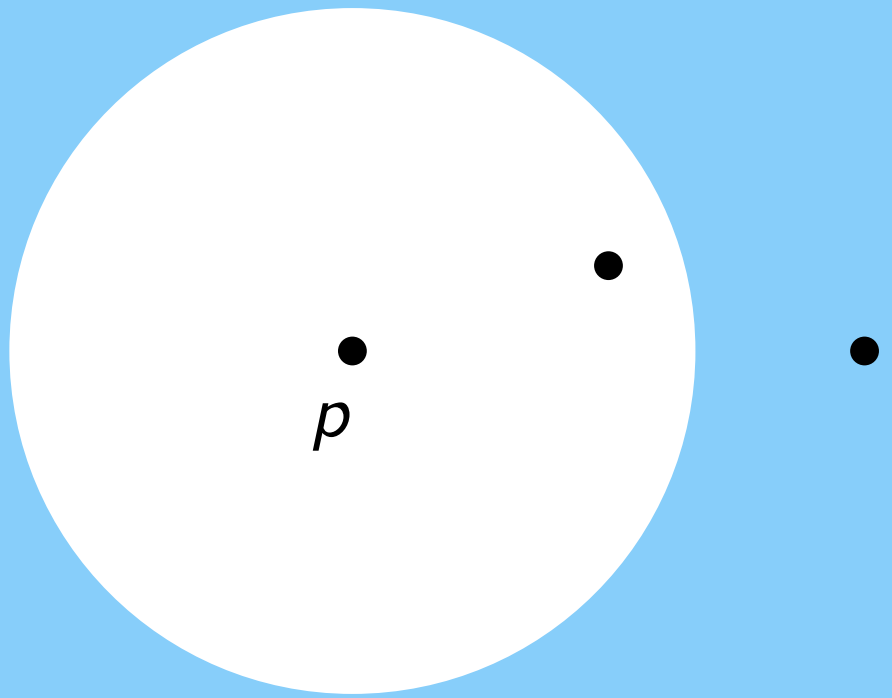
The *generalized range successor problem* is to preprocess the input to find the *successor* of each object among that object's range.

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



Generalized Range Successor

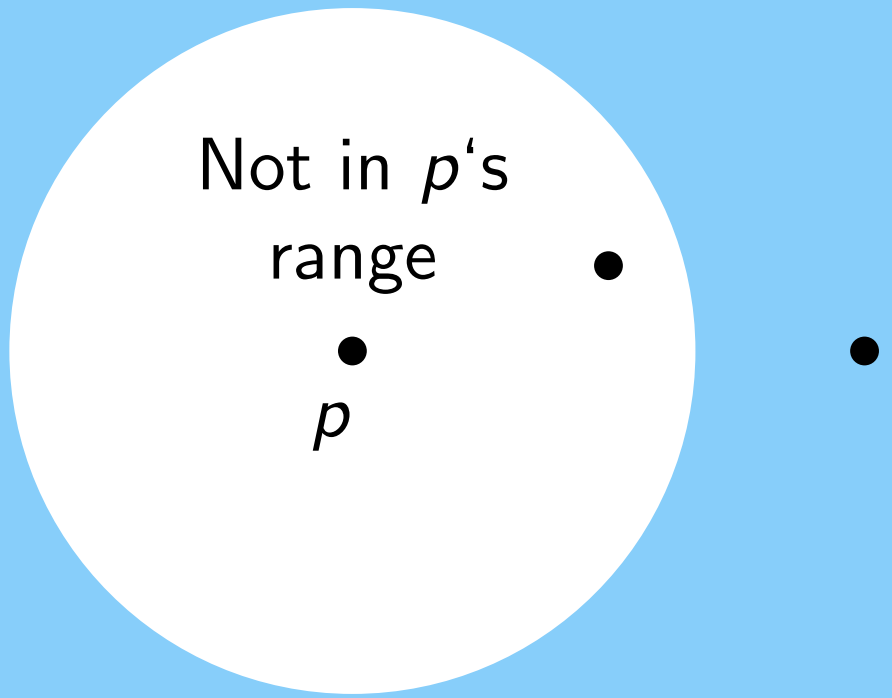
The *generalized range successor problem* is to preprocess the input to find the *successor* of each object among that object's range.

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



Generalized Range Successor

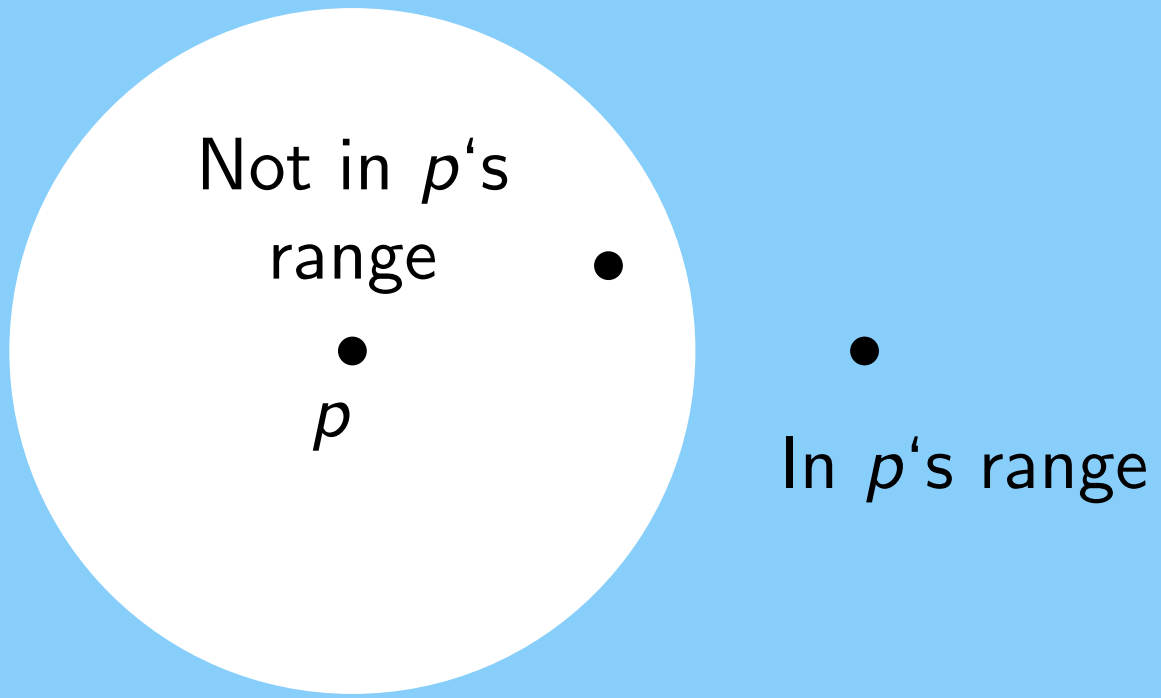
The *generalized range successor problem* is to preprocess the input to find the *successor* of each object among that object's range.

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



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?

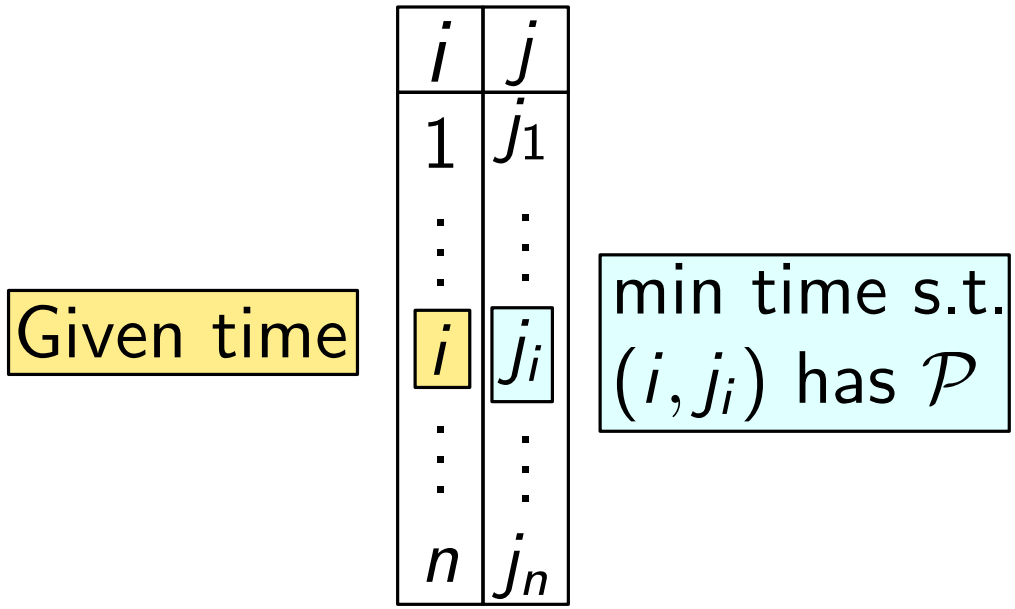
Recall this table

Given time

i	j
1	j_1
⋮	⋮
i	j_i
⋮	⋮
n	j_n

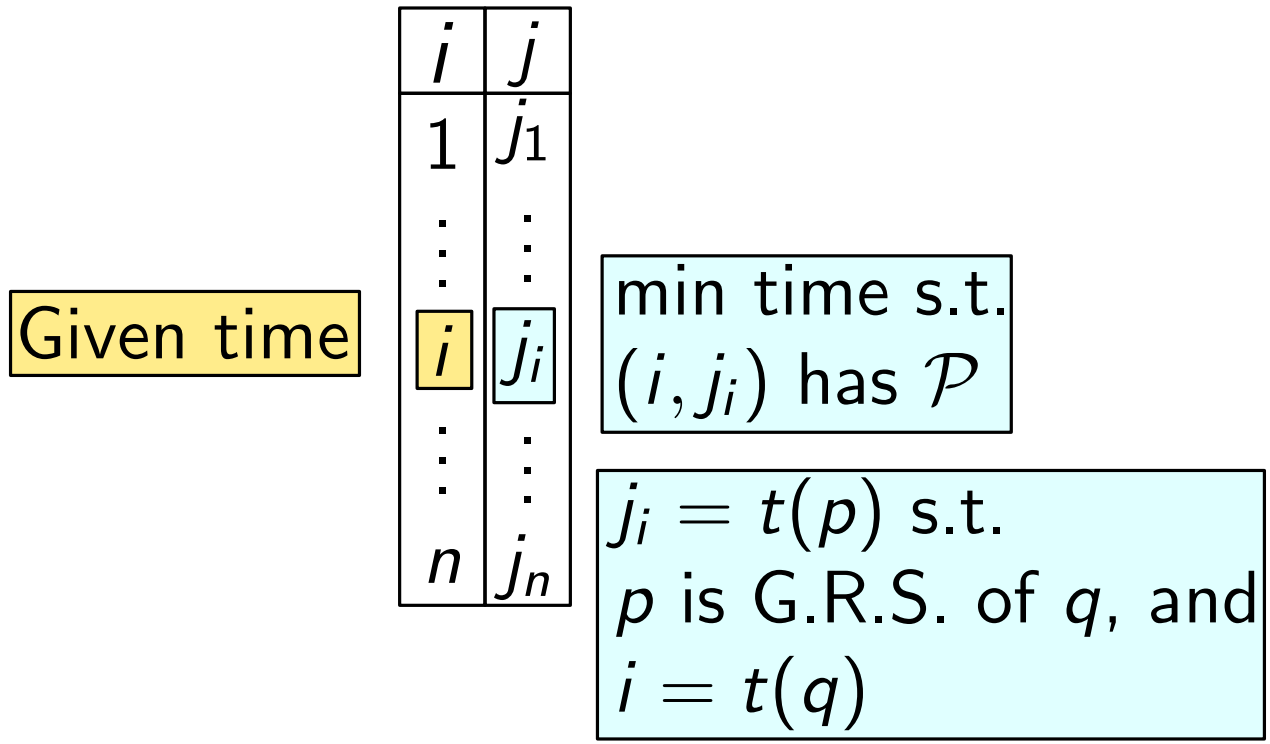
Why Generalized Range Successor?

Recall this table



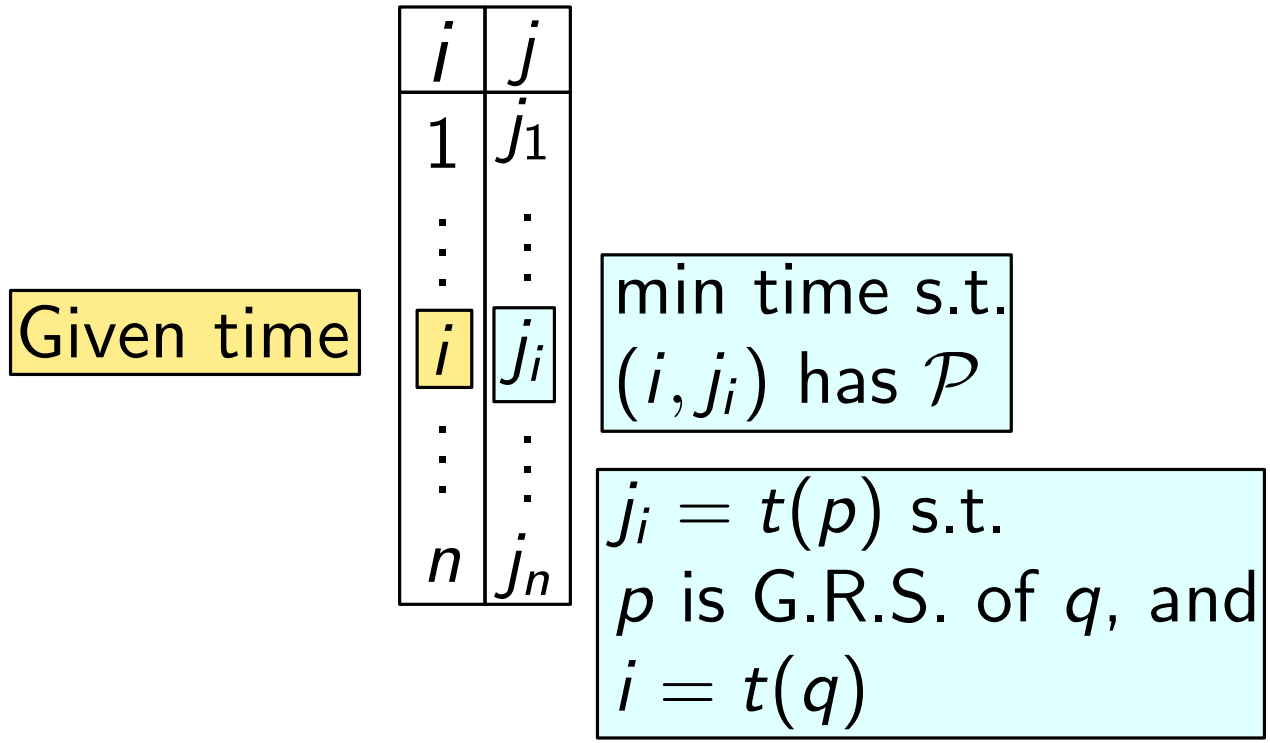
Why Generalized Range Successor?

Recall this table



Why Generalized Range Successor?

Recall this table

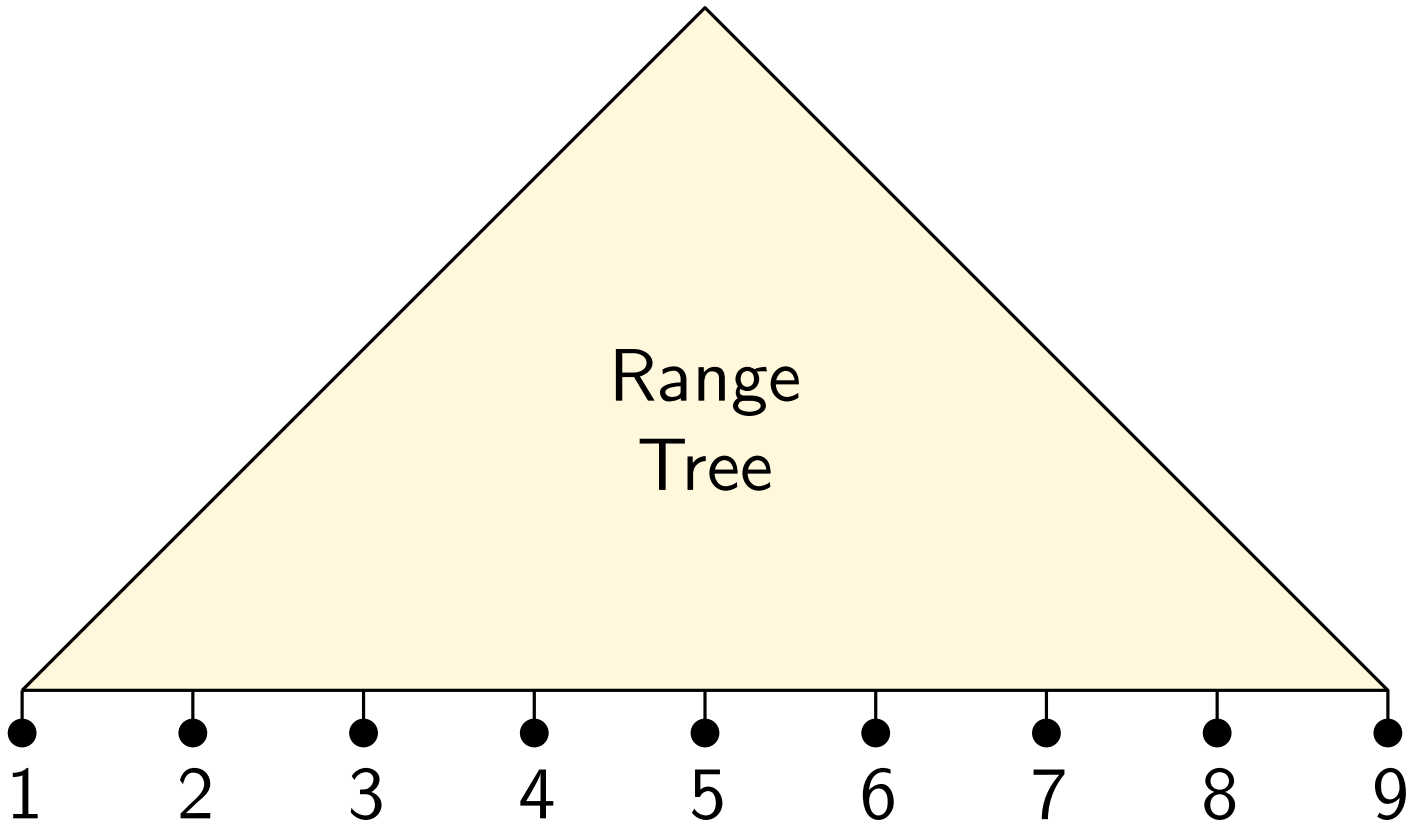


$\therefore$ can preprocess for some time-windowed decision problems by solving n instances of G.R.S.

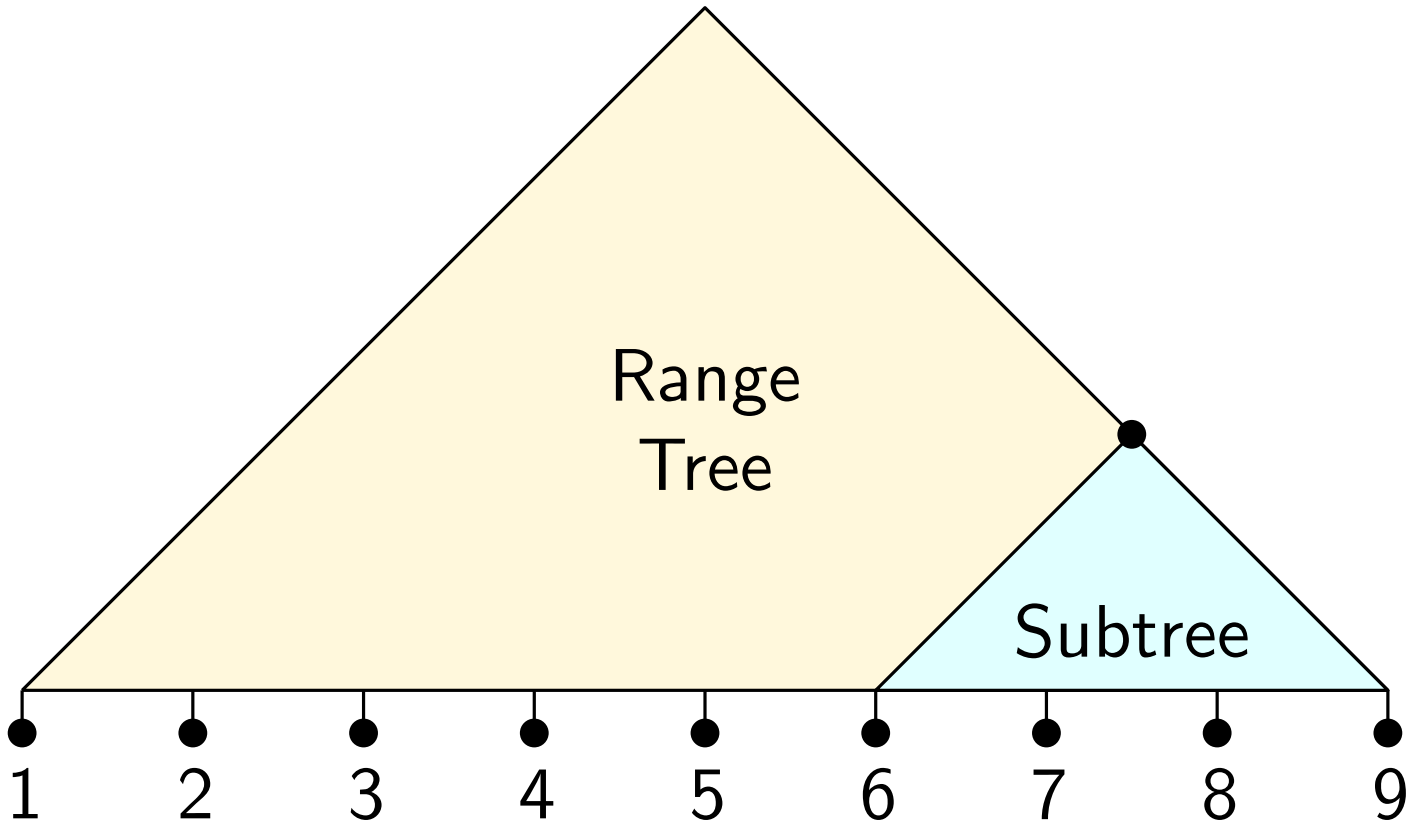
Generalized Range Successor

Consider the objects in time-order

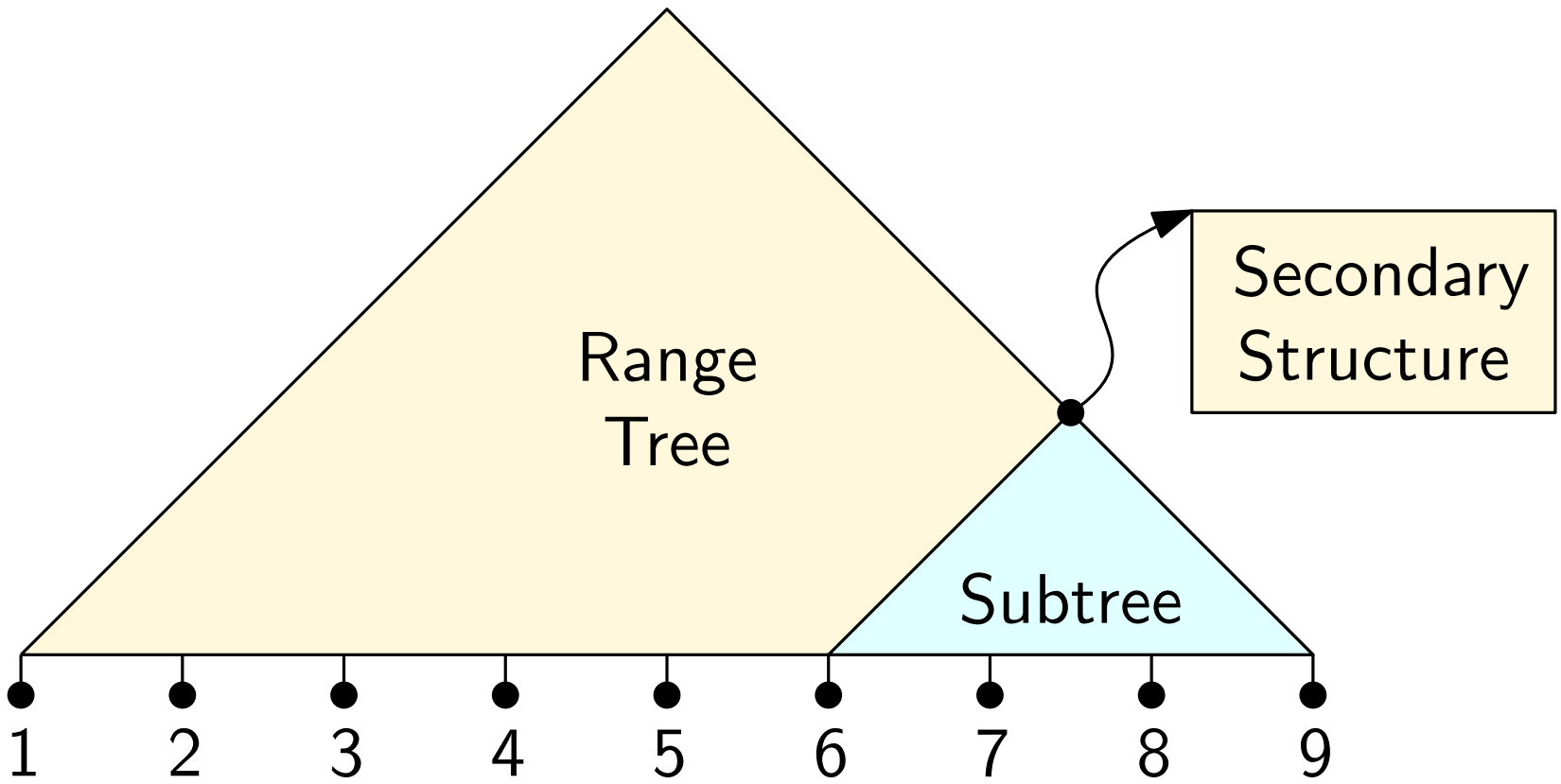
Generalized Range Successor



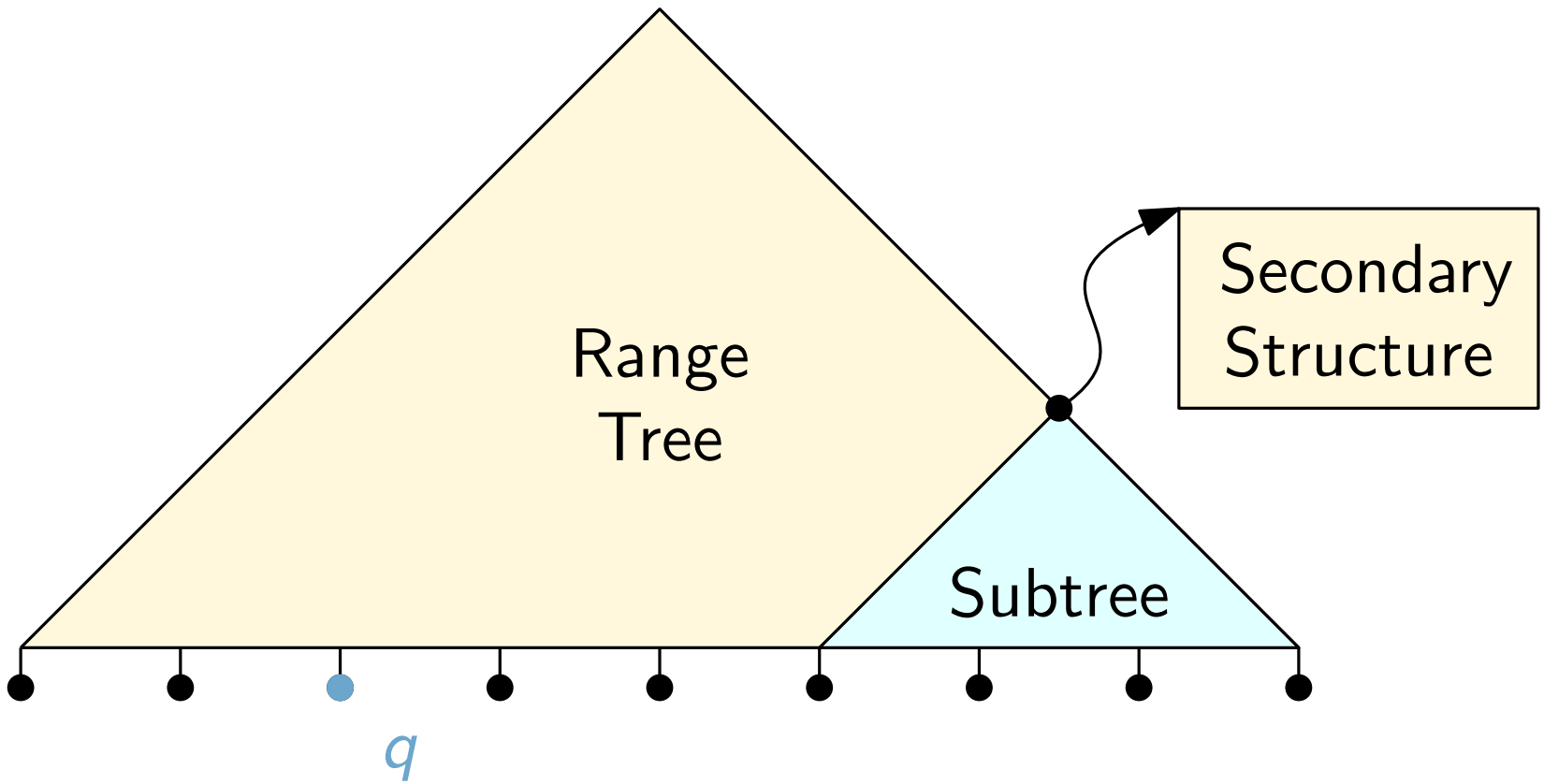
Generalized Range Successor



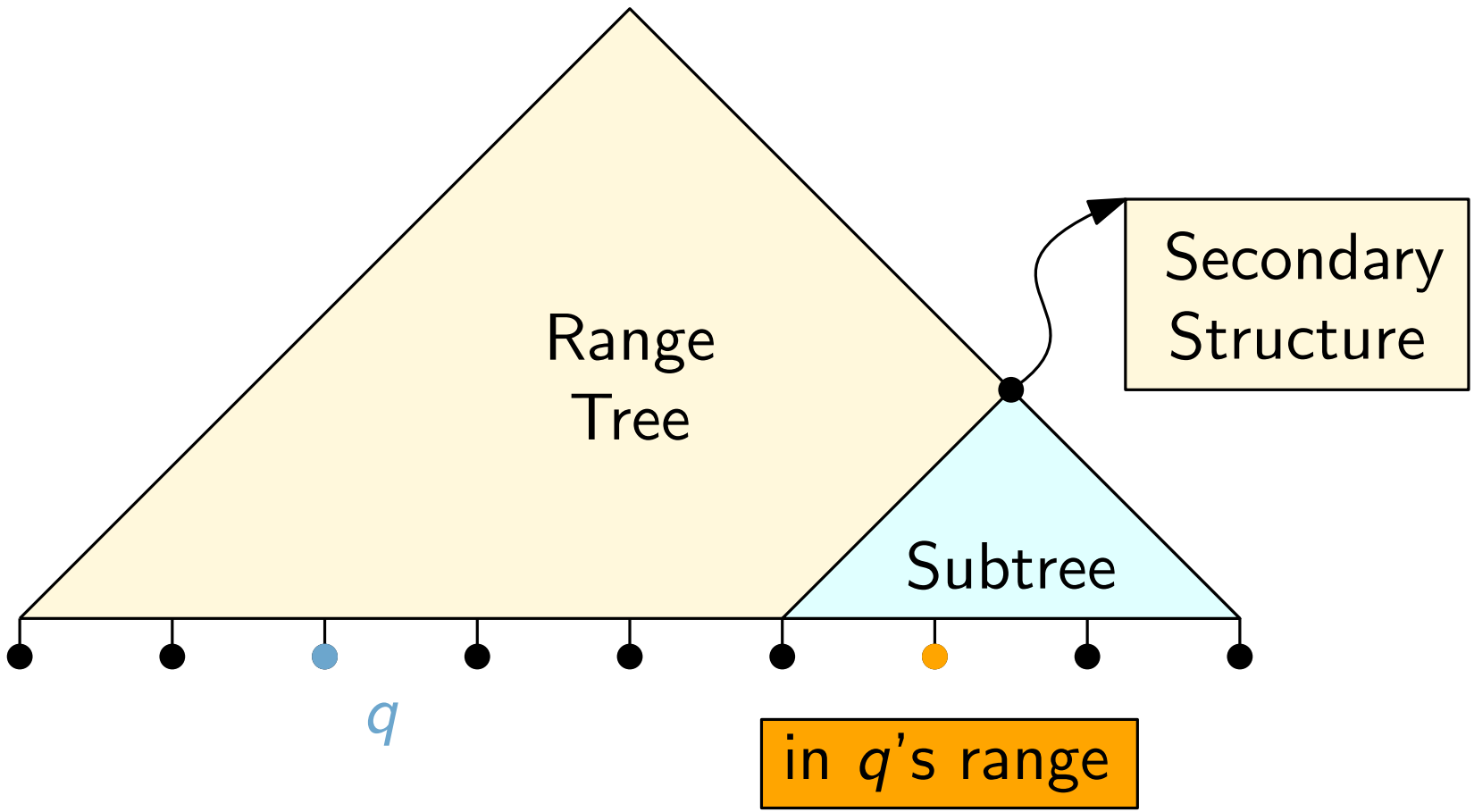
Generalized Range Successor



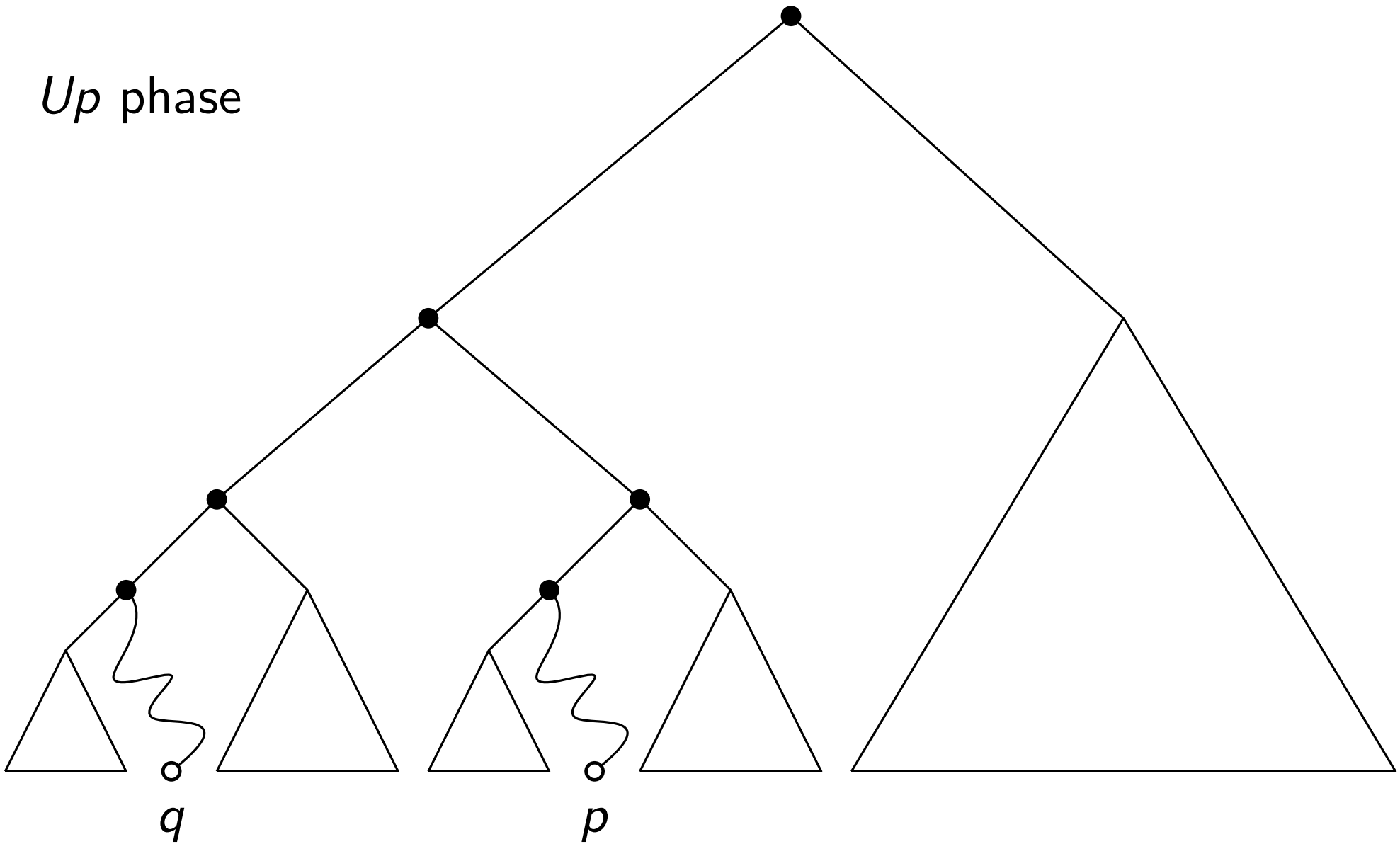
Generalized Range Successor



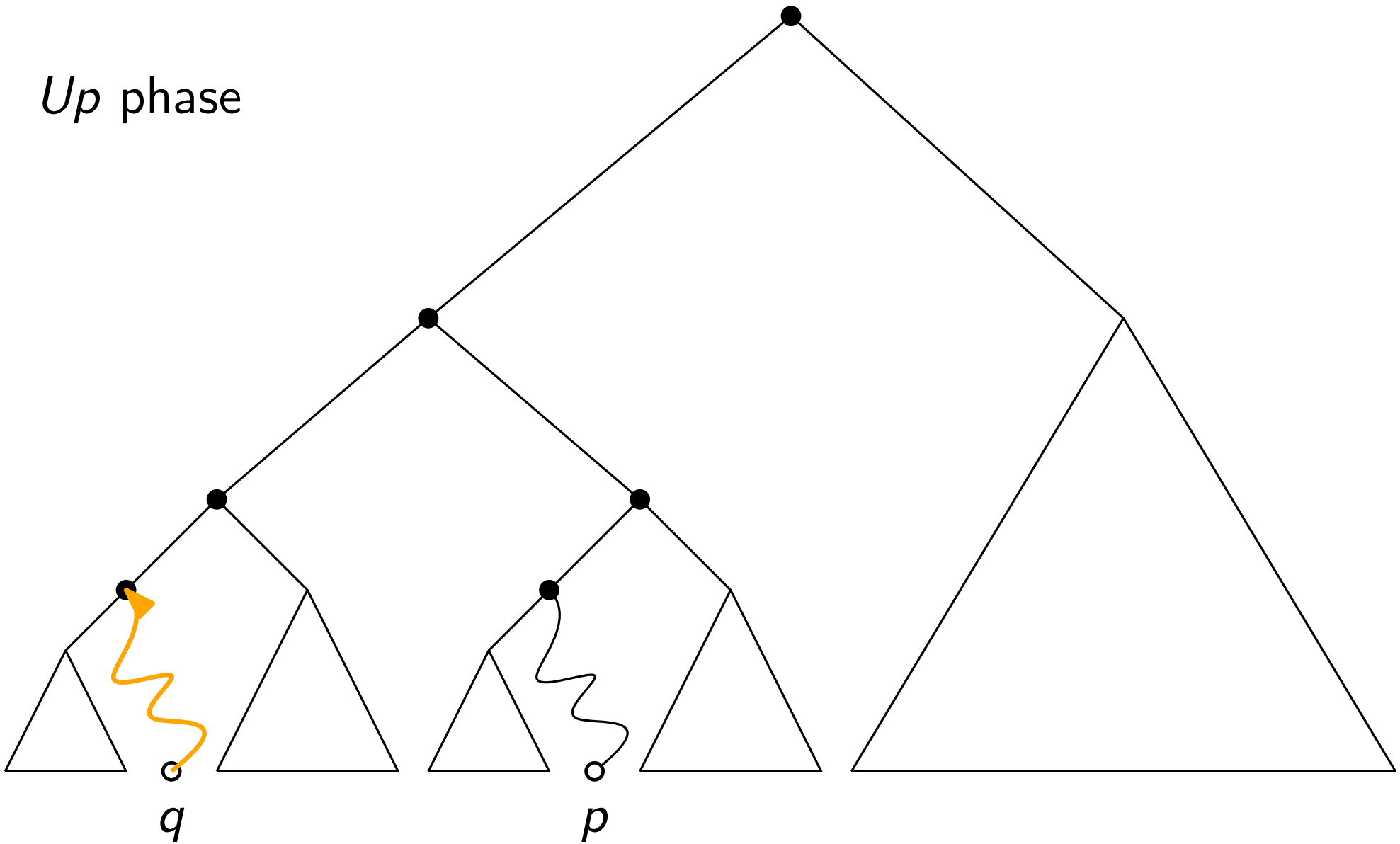
Generalized Range Successor



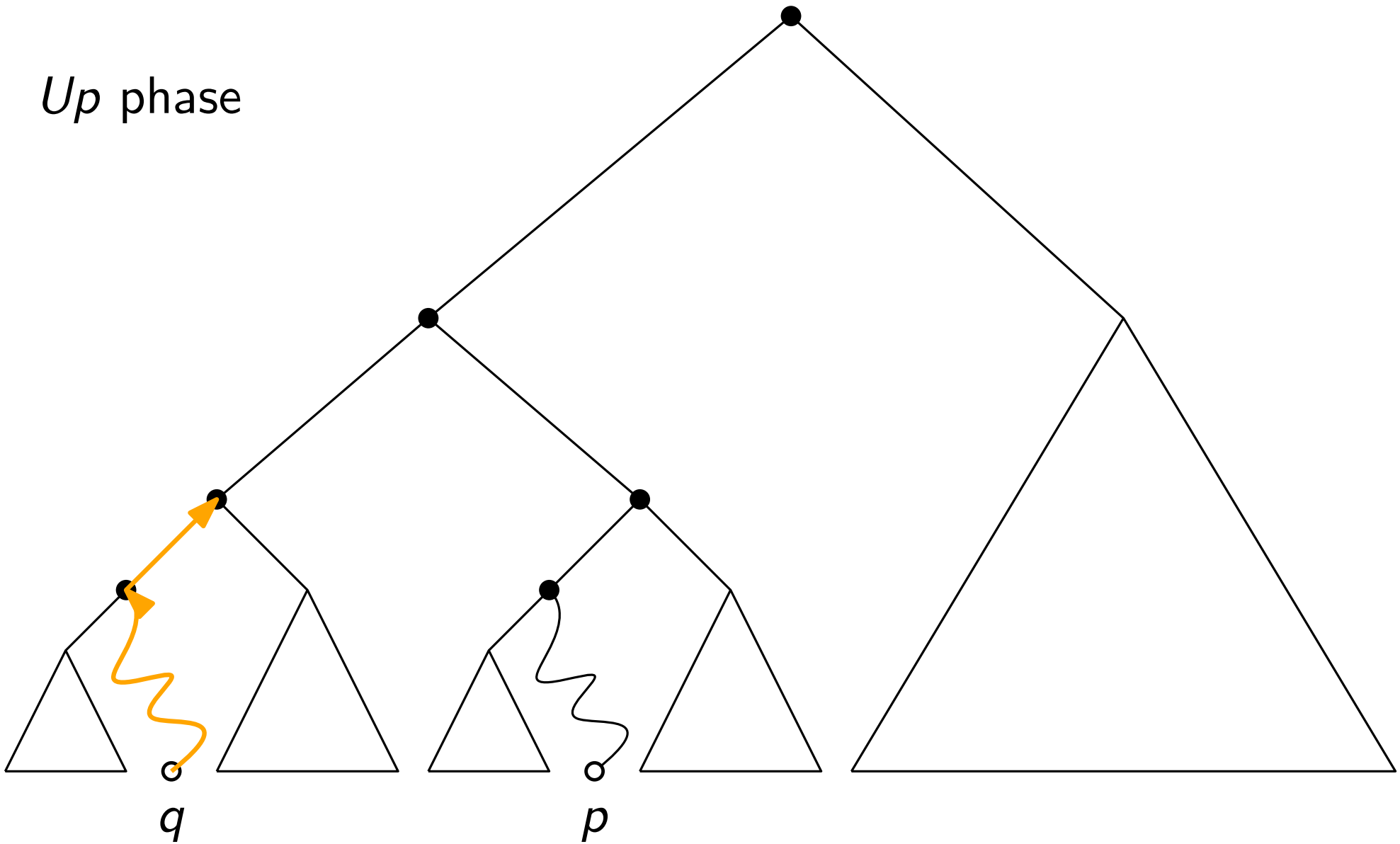
Avoiding Binary Search



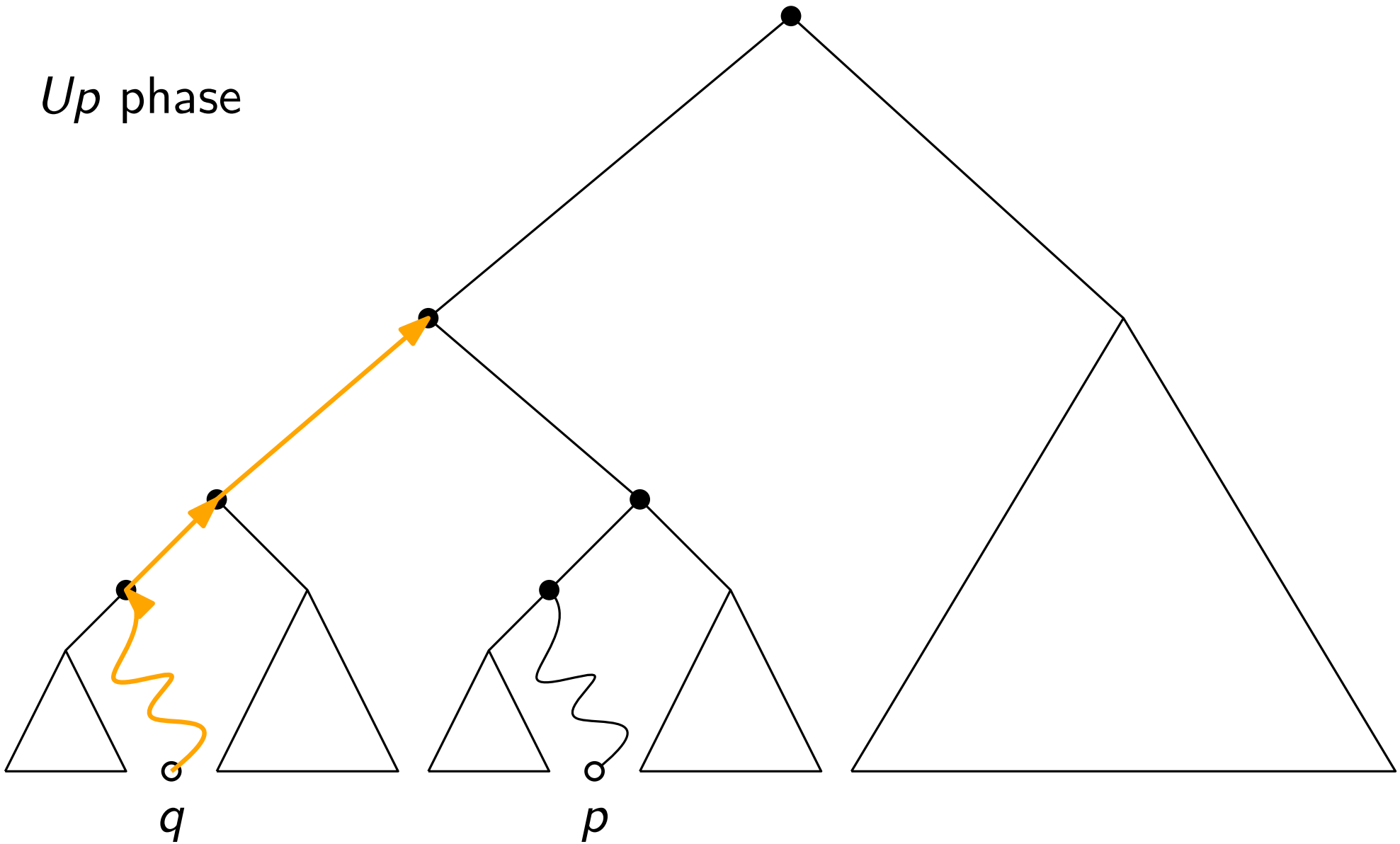
Avoiding Binary Search



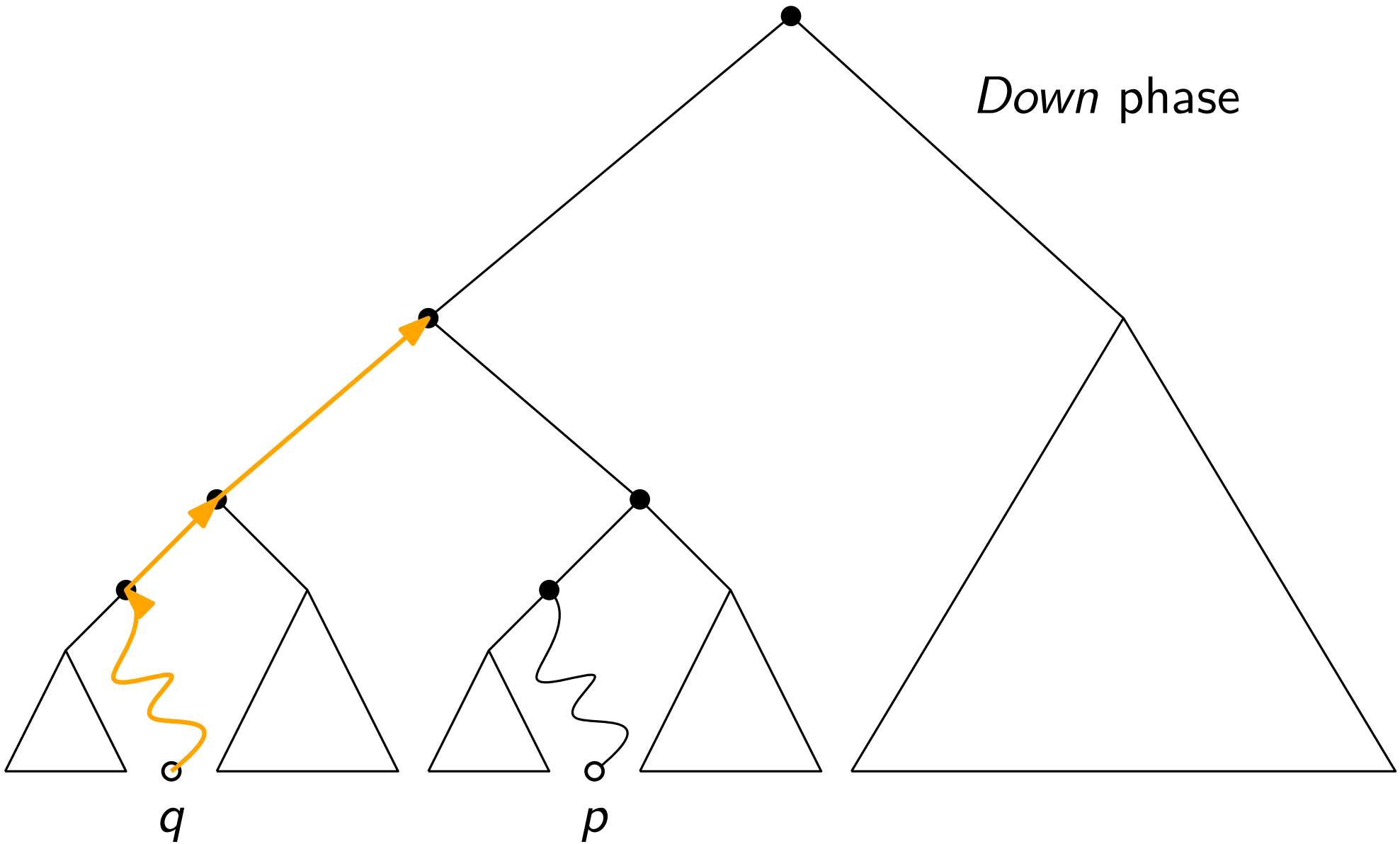
Avoiding Binary Search



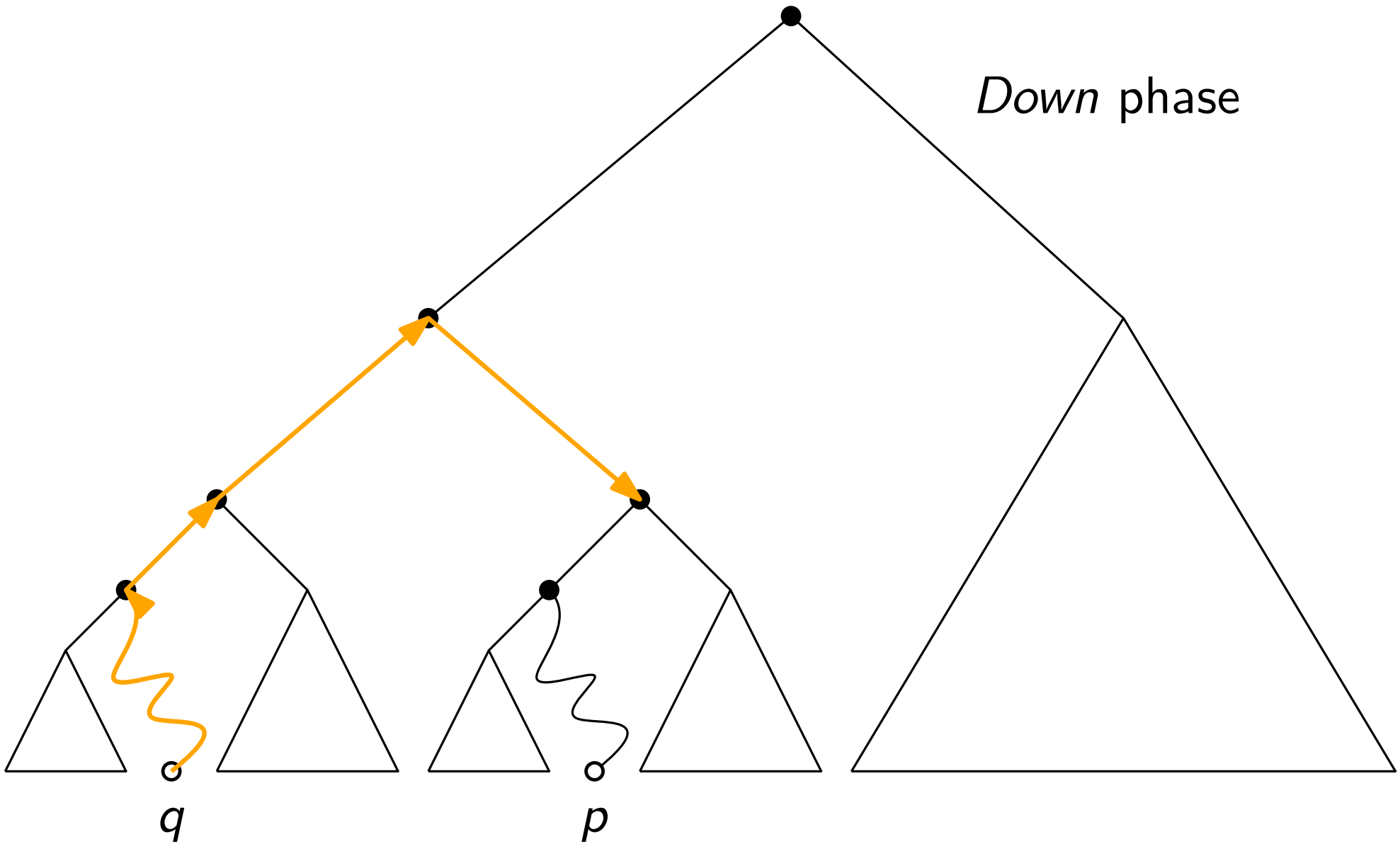
Avoiding Binary Search



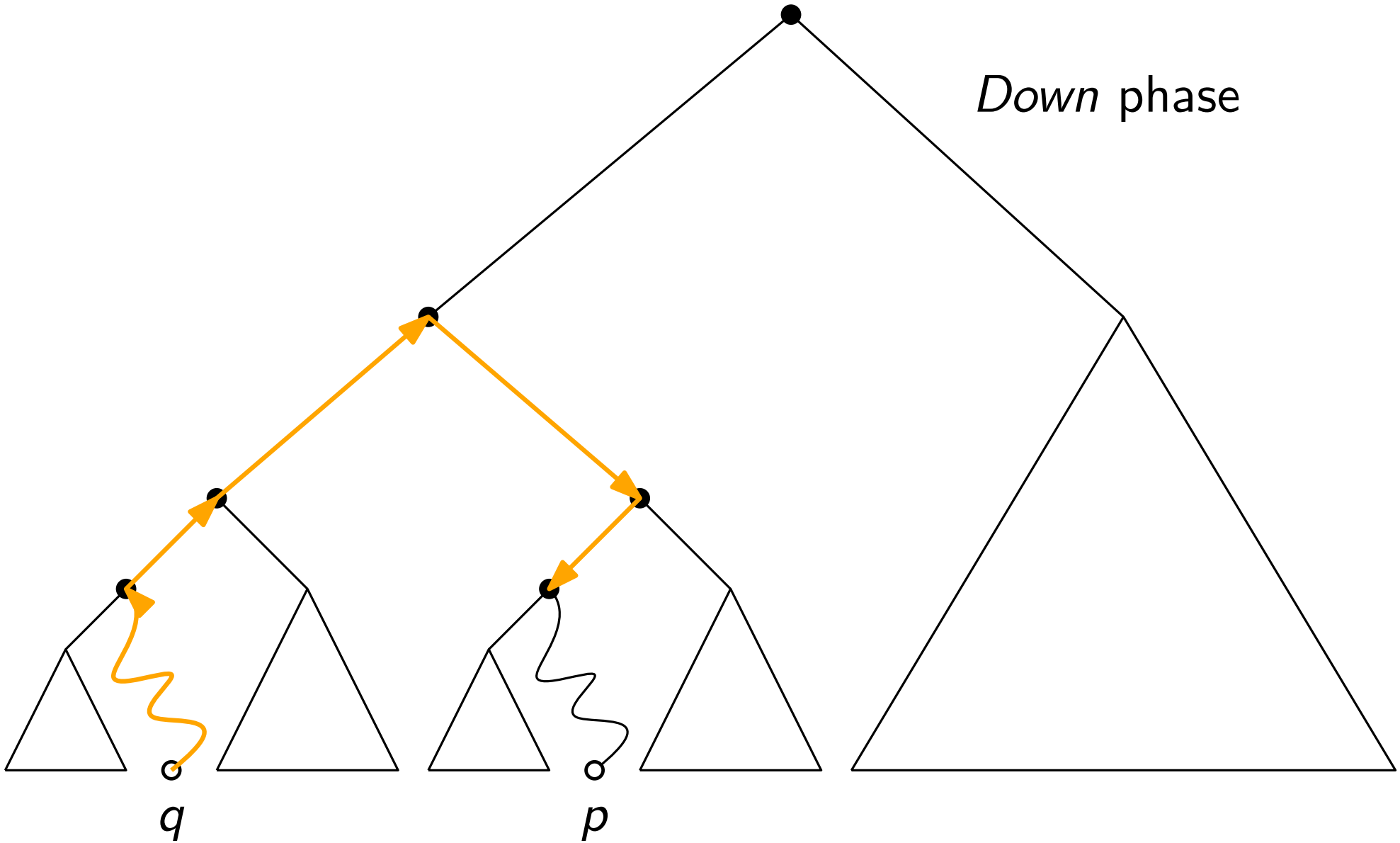
Avoiding Binary Search



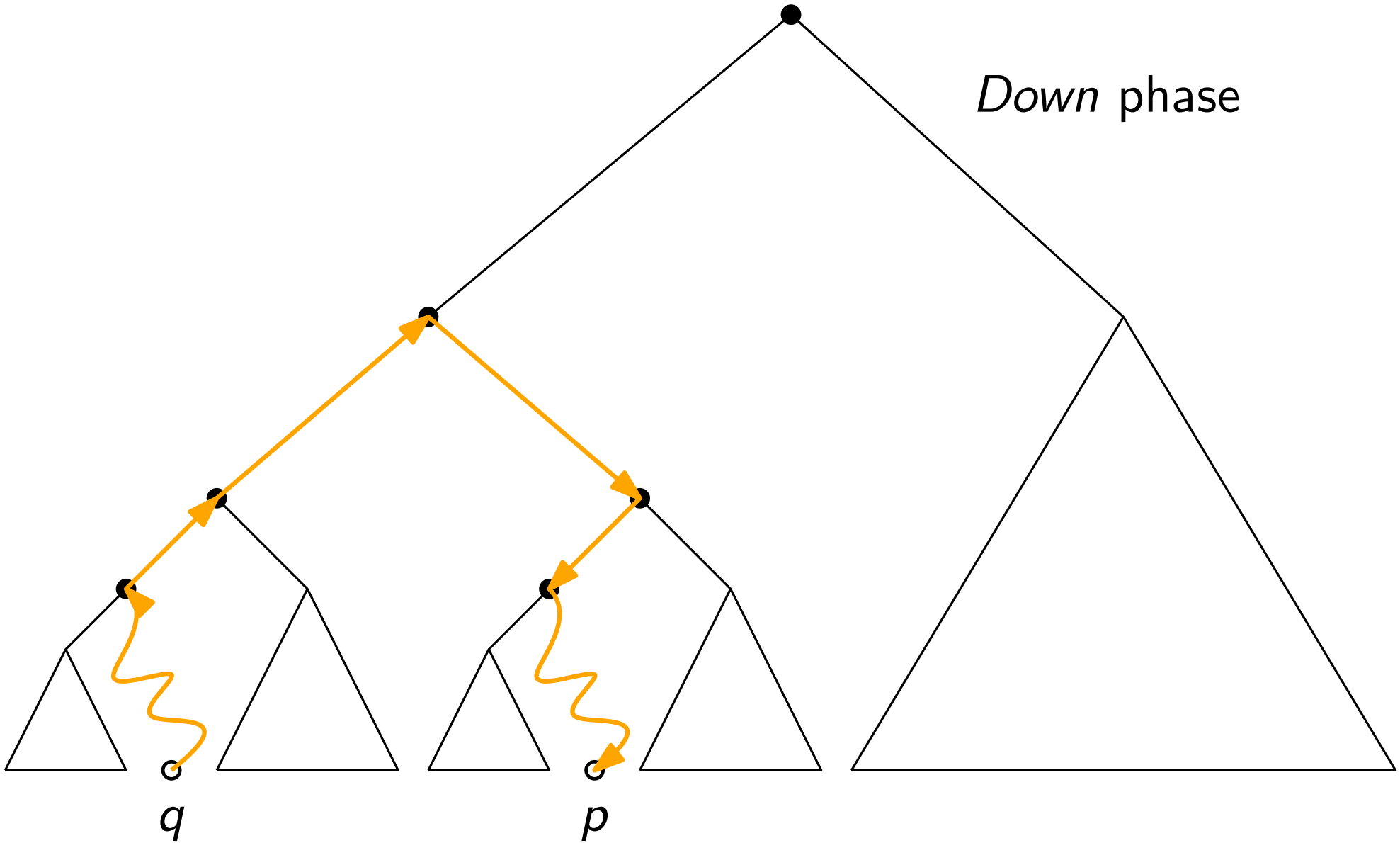
Avoiding Binary Search



Avoiding Binary Search



Avoiding Binary Search



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: Generalized Range Successor

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

Approach #2: FIFO Update Sequence

Problem	Preprocessing Time
2D convex hull area decision	$O(n \log n \log \log n)$
$\rightarrow$ improved to	$O(n\alpha(n) \log n)$
2D width decision	$O(n \log^8 n)$

New Results

Approach #1: Generalized Range Successor

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

Approach #2: FIFO Update Sequence

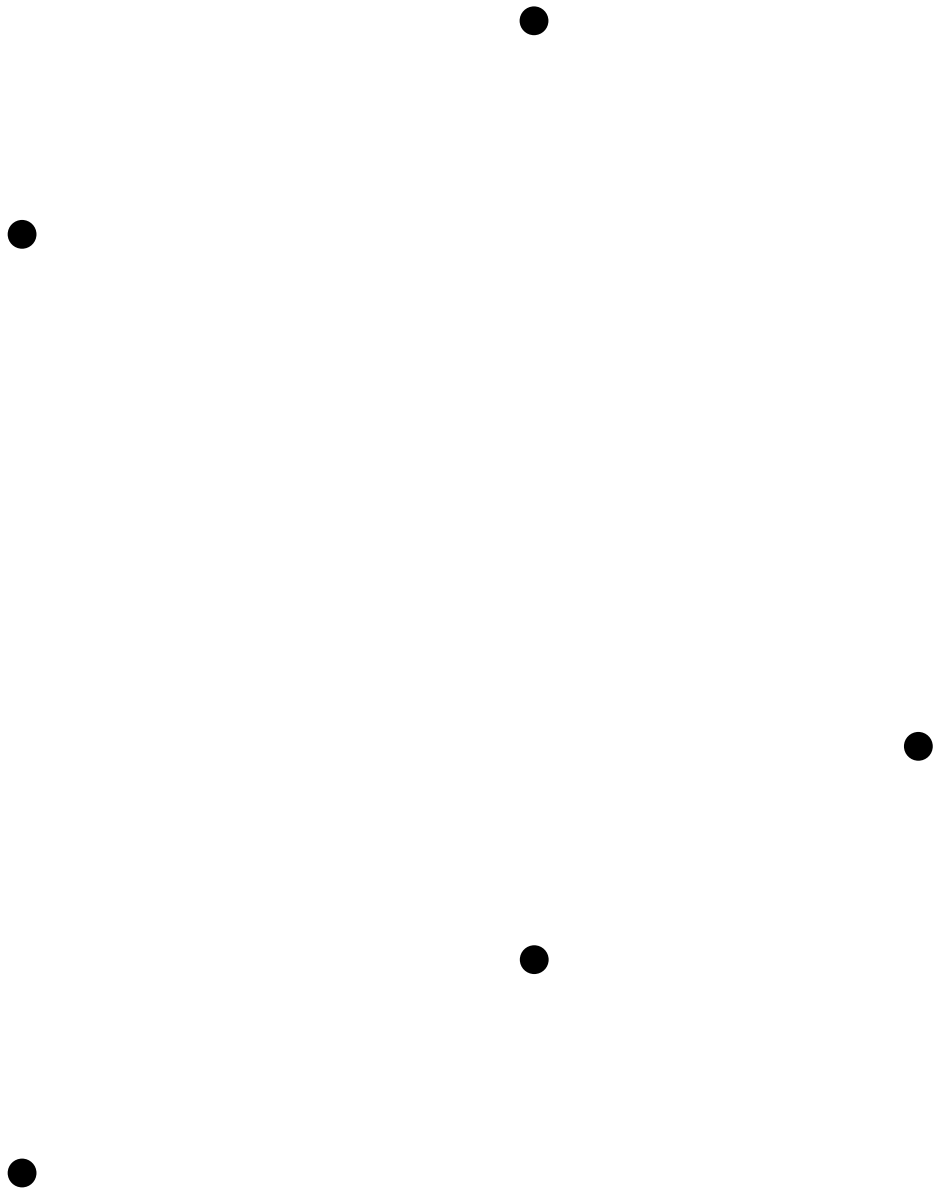
Problem	Preprocessing Time
2D convex hull area decision	$O(n \log n \log \log n)$
$\rightarrow$ improved to	$O(n\alpha(n) \log n)$
2D width decision	$O(n \log^8 n)$

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?

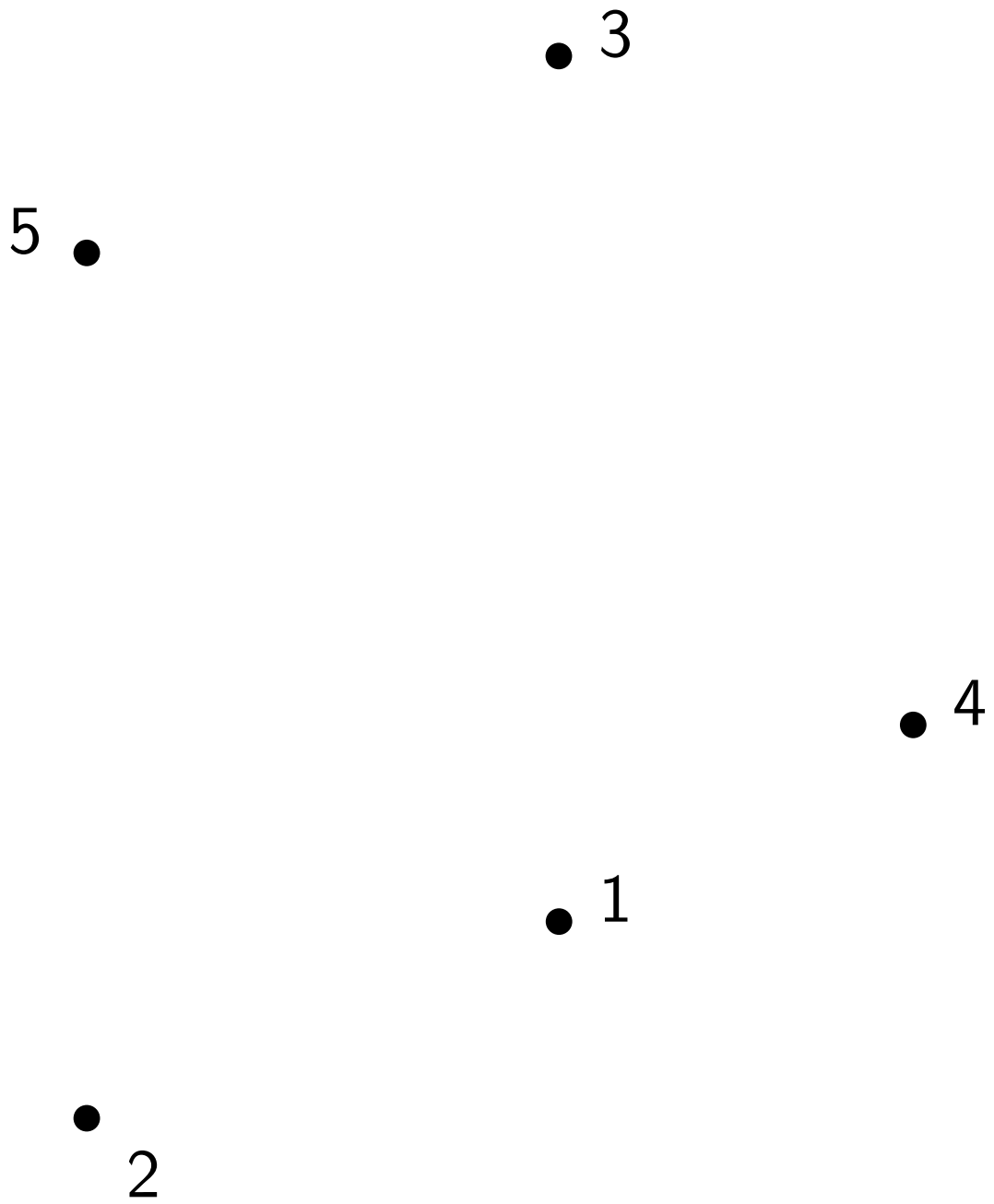
Time-Windowed $\rightarrow$ Dynamic

CH area > 1?



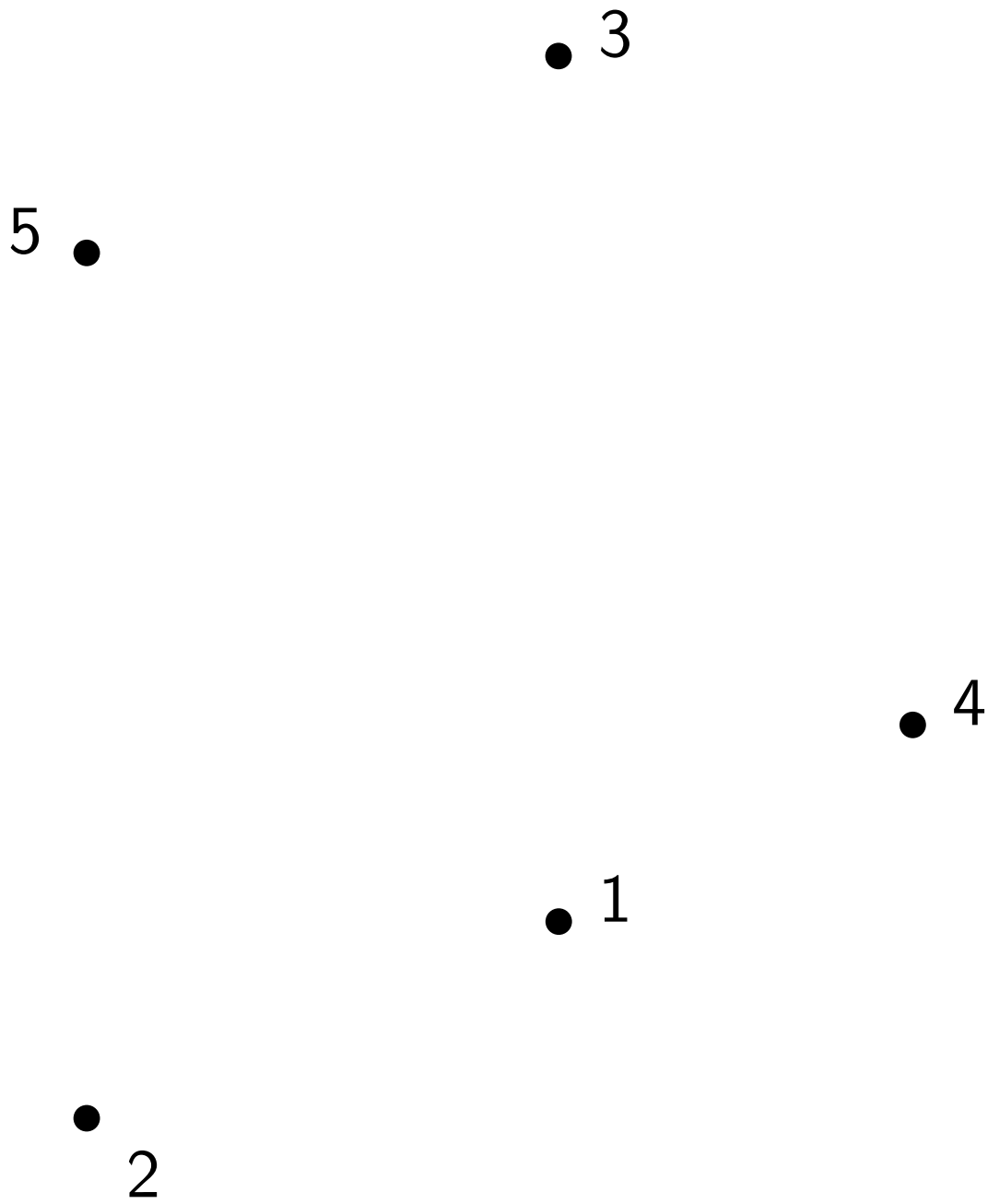
Time-Windowed → Dynamic

CH area > 1?



Time-Windowed $\rightarrow$ Dynamic

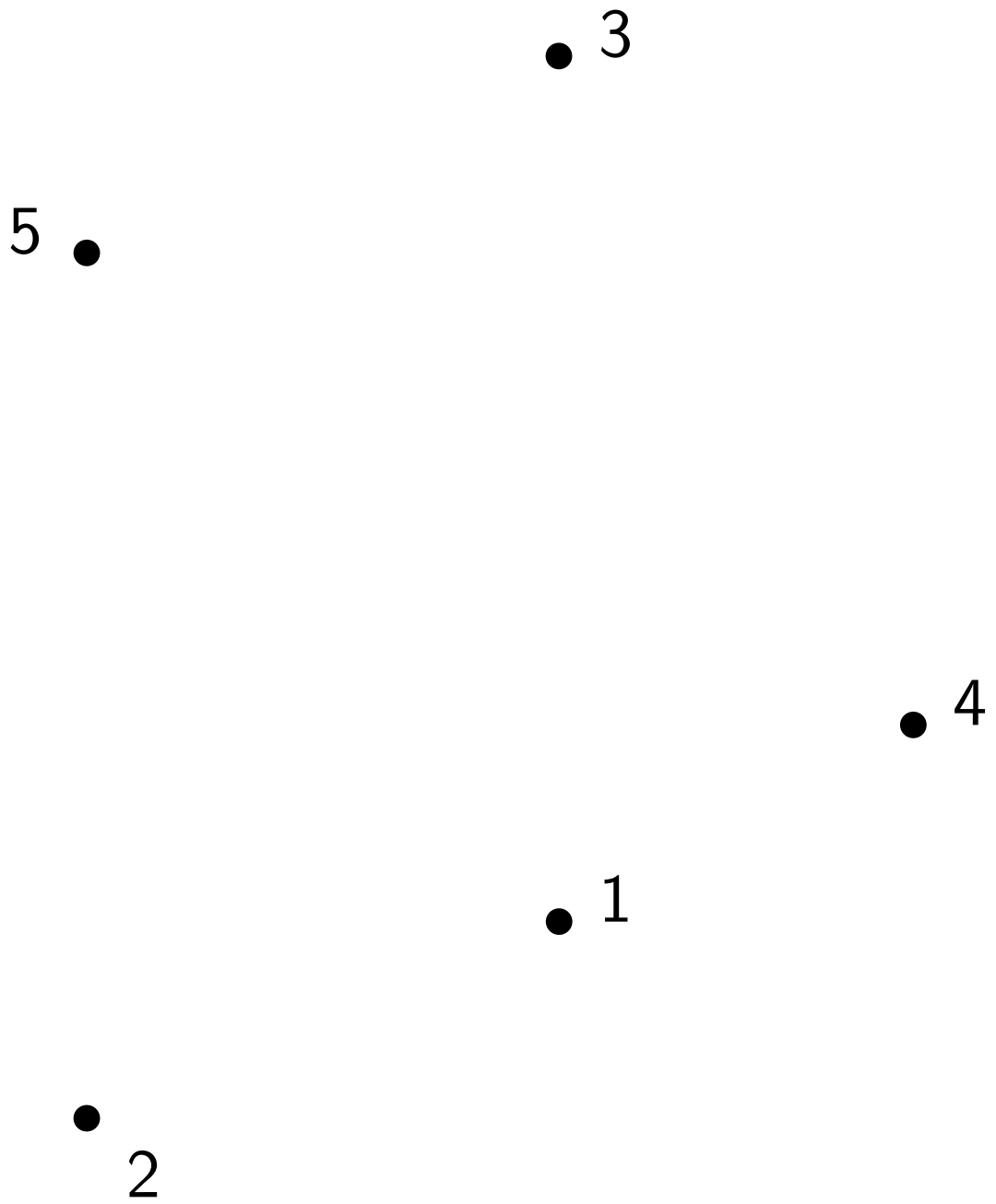
CH area > 1?



i
1
2
3
4
5

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?

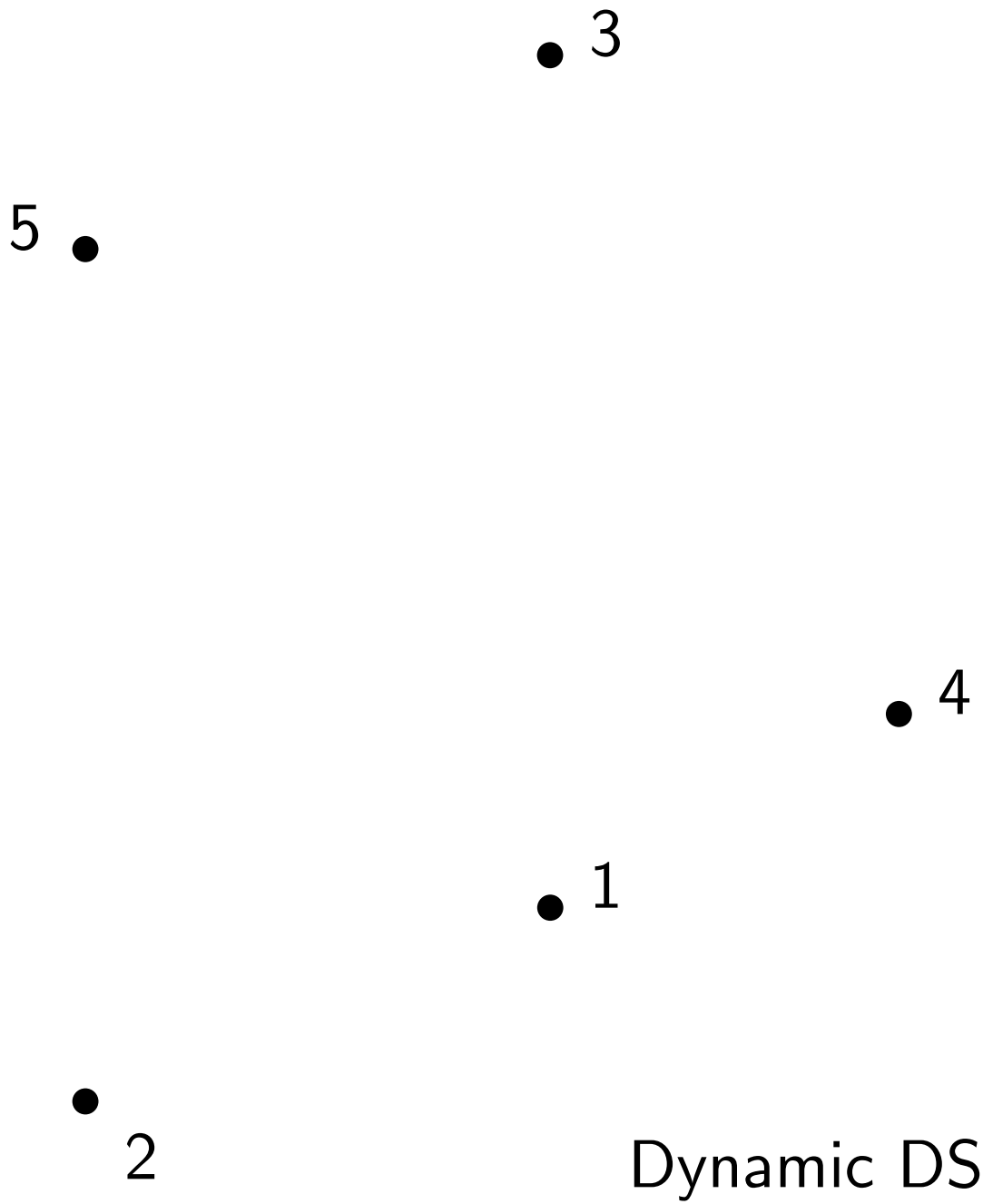


i	j
1	
2	
3	
4	
5	

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?

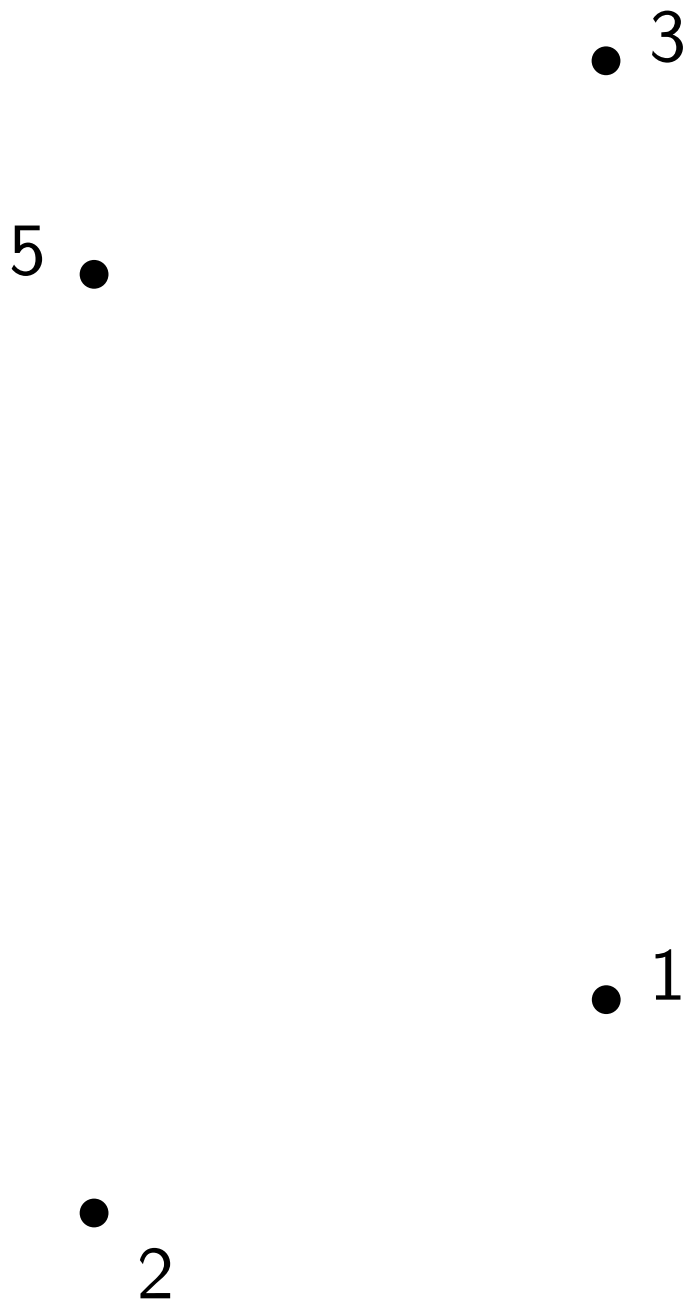
i	j
1	
2	
3	
4	
5	



Dynamic DS

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?



i	j
1	
2	
3	
4	
5	

Dynamic DS : Has property? No

Time-Windowed → Dynamic

CH area > 1?

i	j
1	
2	
3	
4	
5	

Dynamic DS : Has property? No

Time-Windowed → Dynamic

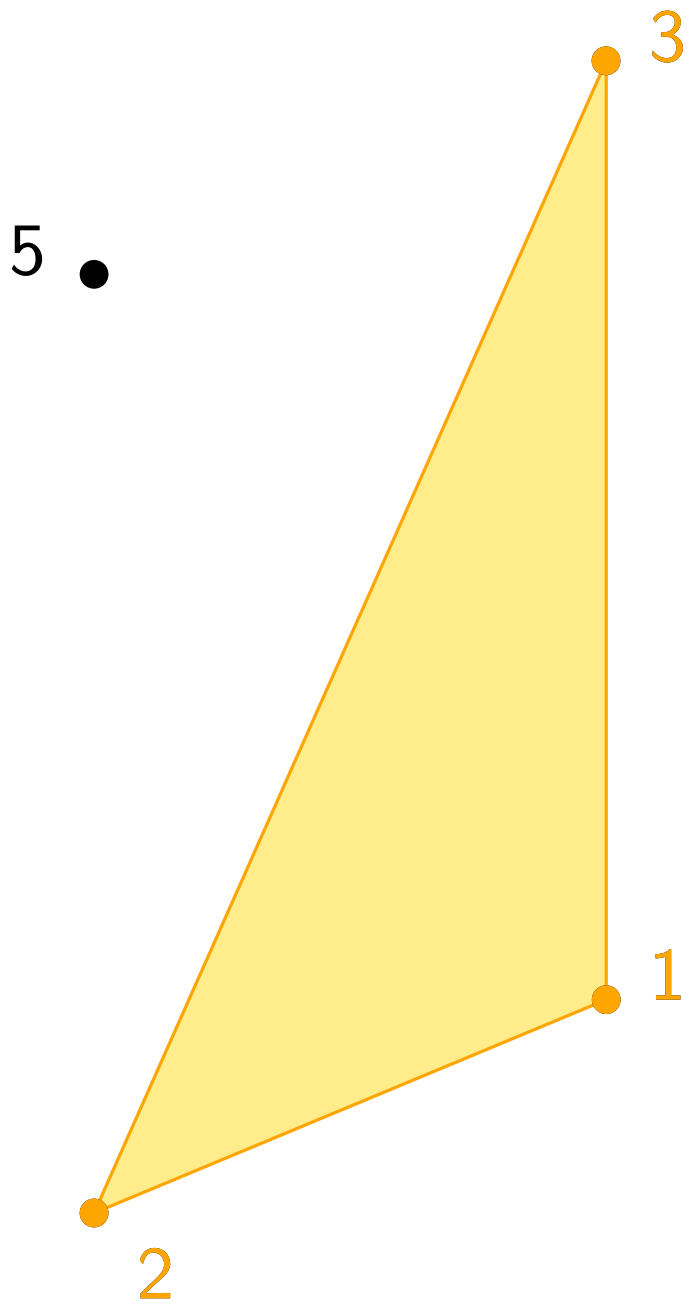
CH area > 1?

i	j
1	
2	
3	
4	
5	

Dynamic DS : Has property? No

Time-Windowed → Dynamic

CH area > 1?

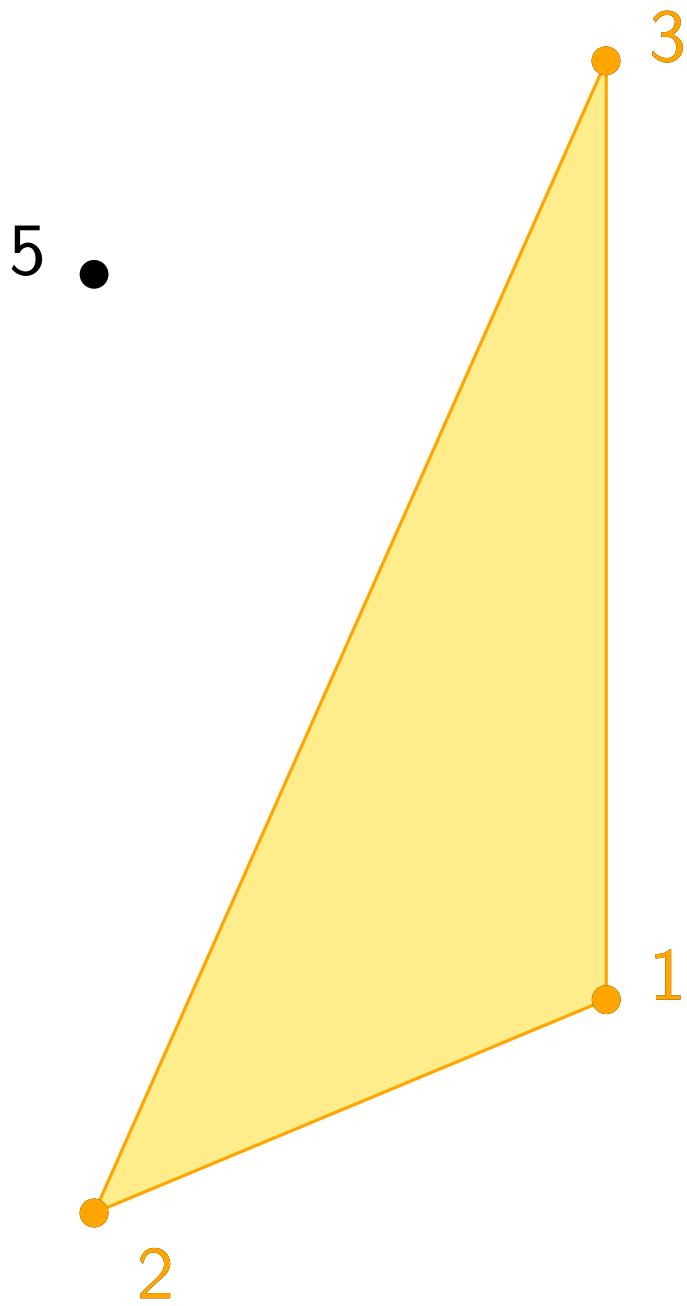


i	j
1	
2	
3	
4	
5	

Dynamic DS : Has property? Yes

Time-Windowed → Dynamic

CH area > 1?

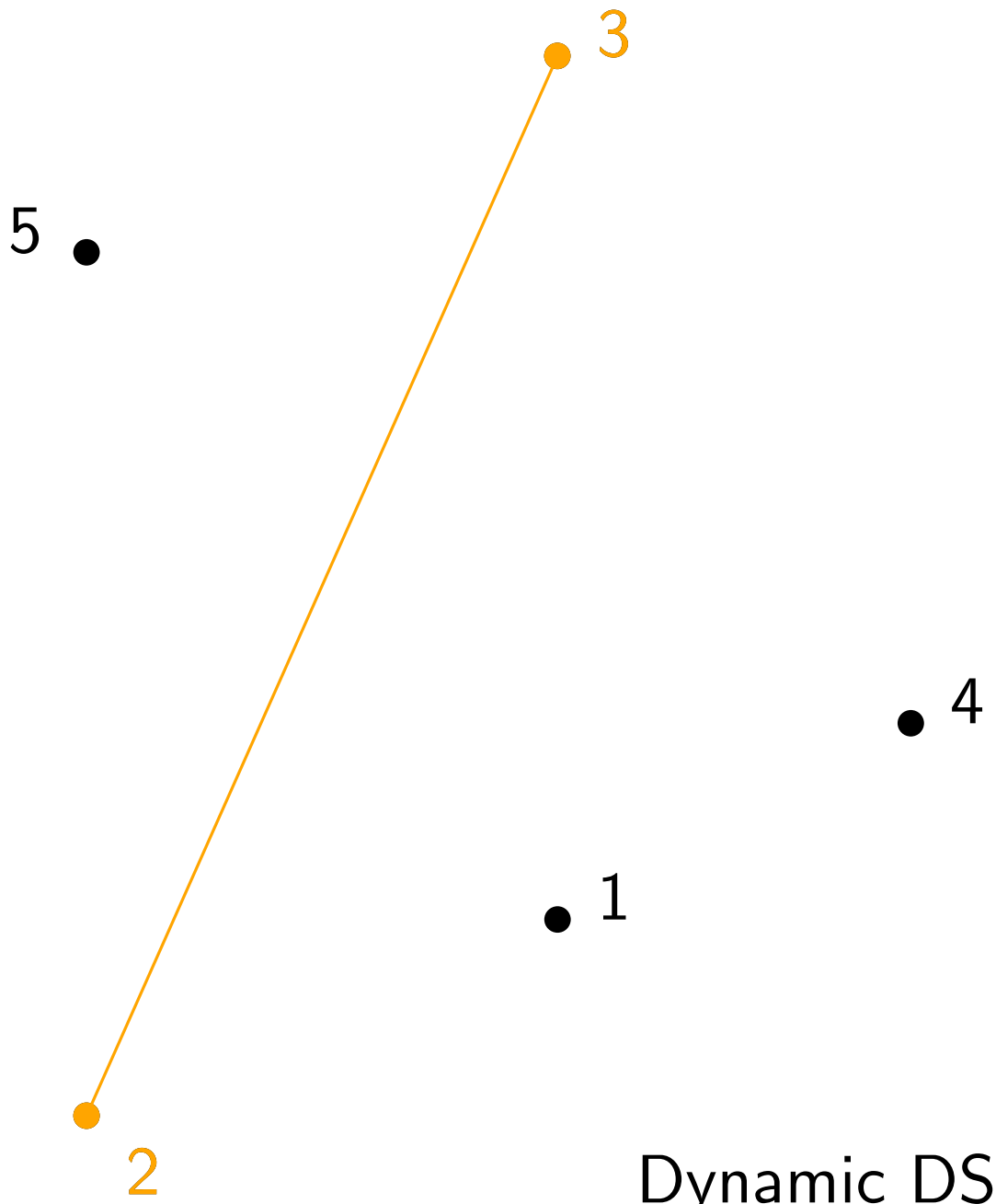


i	j
1	3
2	
3	
4	
5	

Dynamic DS : Has property? Yes

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?

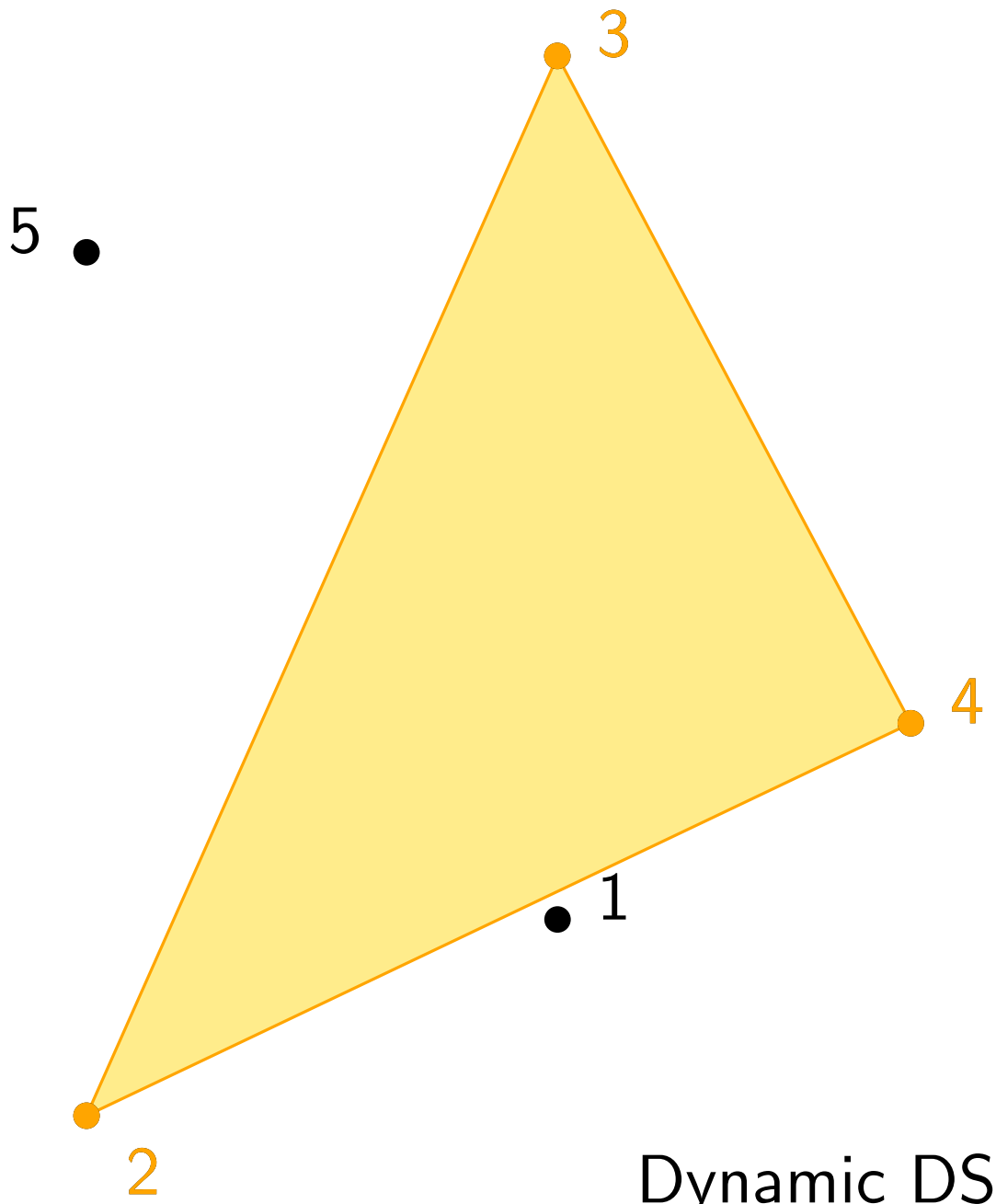


i	j
1	3
2	
3	
4	
5	

Dynamic DS : Has property? No

Time-Windowed → Dynamic

CH area > 1?

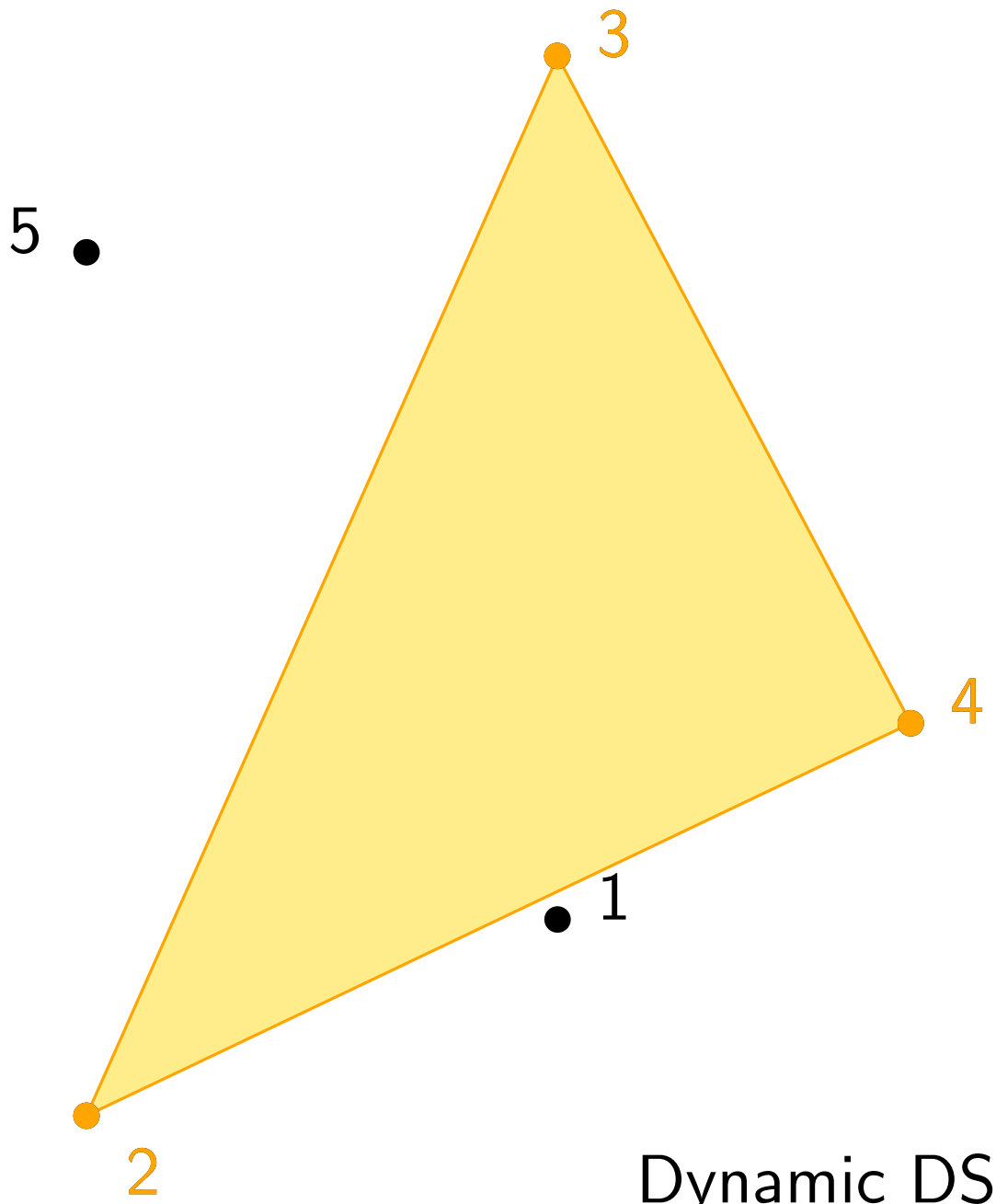


i	j
1	3
2	
3	
4	
5	

Dynamic DS : Has property? Yes

Time-Windowed → Dynamic

CH area > 1?

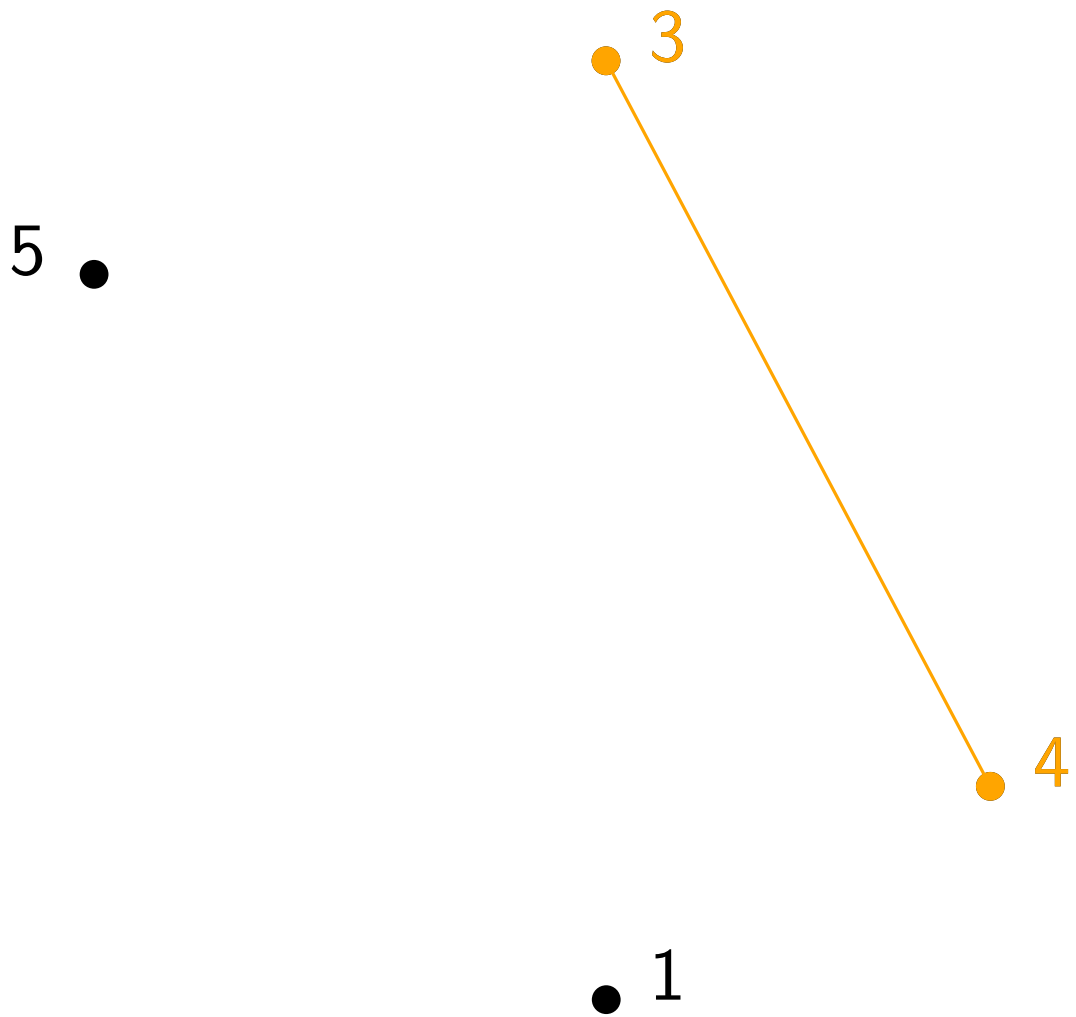


i	j
1	3
2	4
3	
4	
5	

Dynamic DS : Has property? Yes

Time-Windowed → Dynamic

CH area > 1?

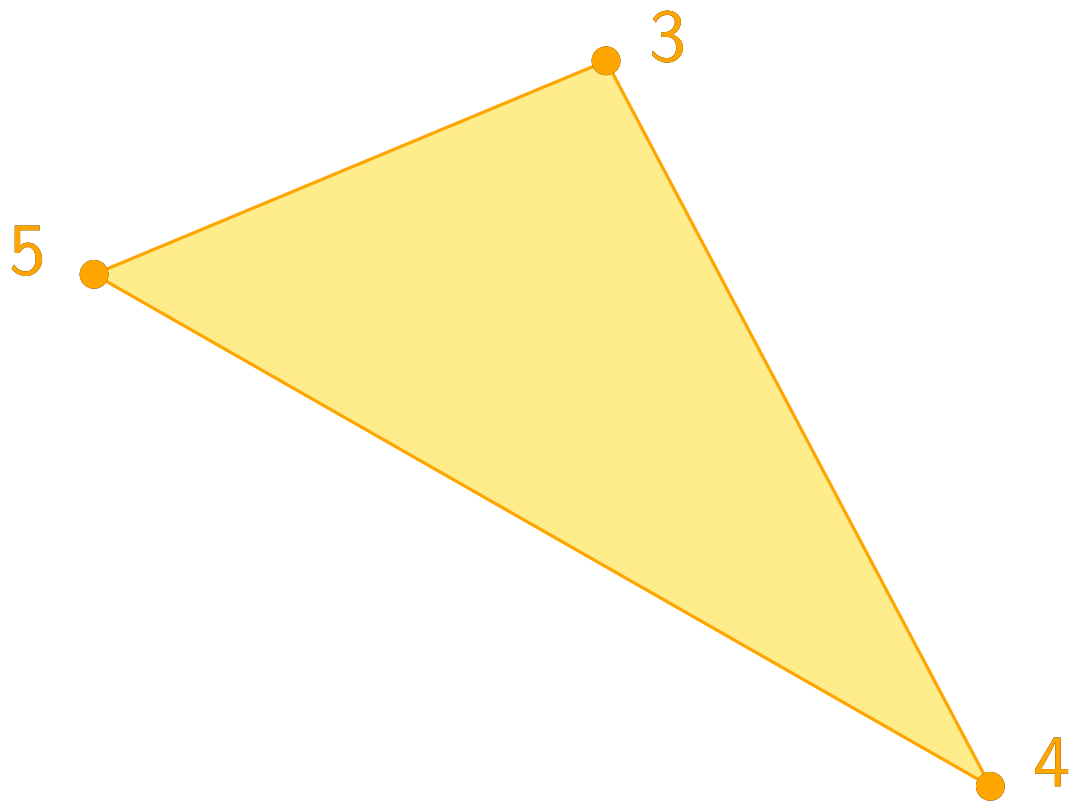


i	j
1	3
2	4
3	
4	
5	

Dynamic DS : Has property? No

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?



i	j
1	3
2	4
3	
4	
5	

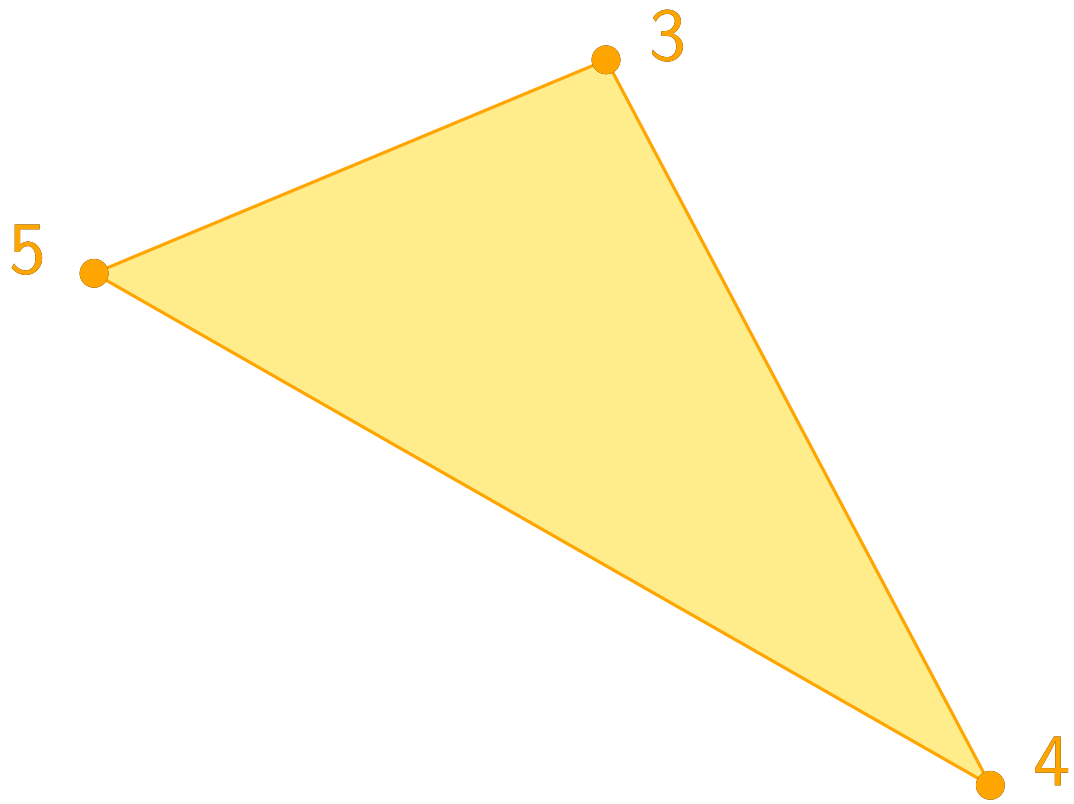
1

2

Dynamic DS : Has property? Yes

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?

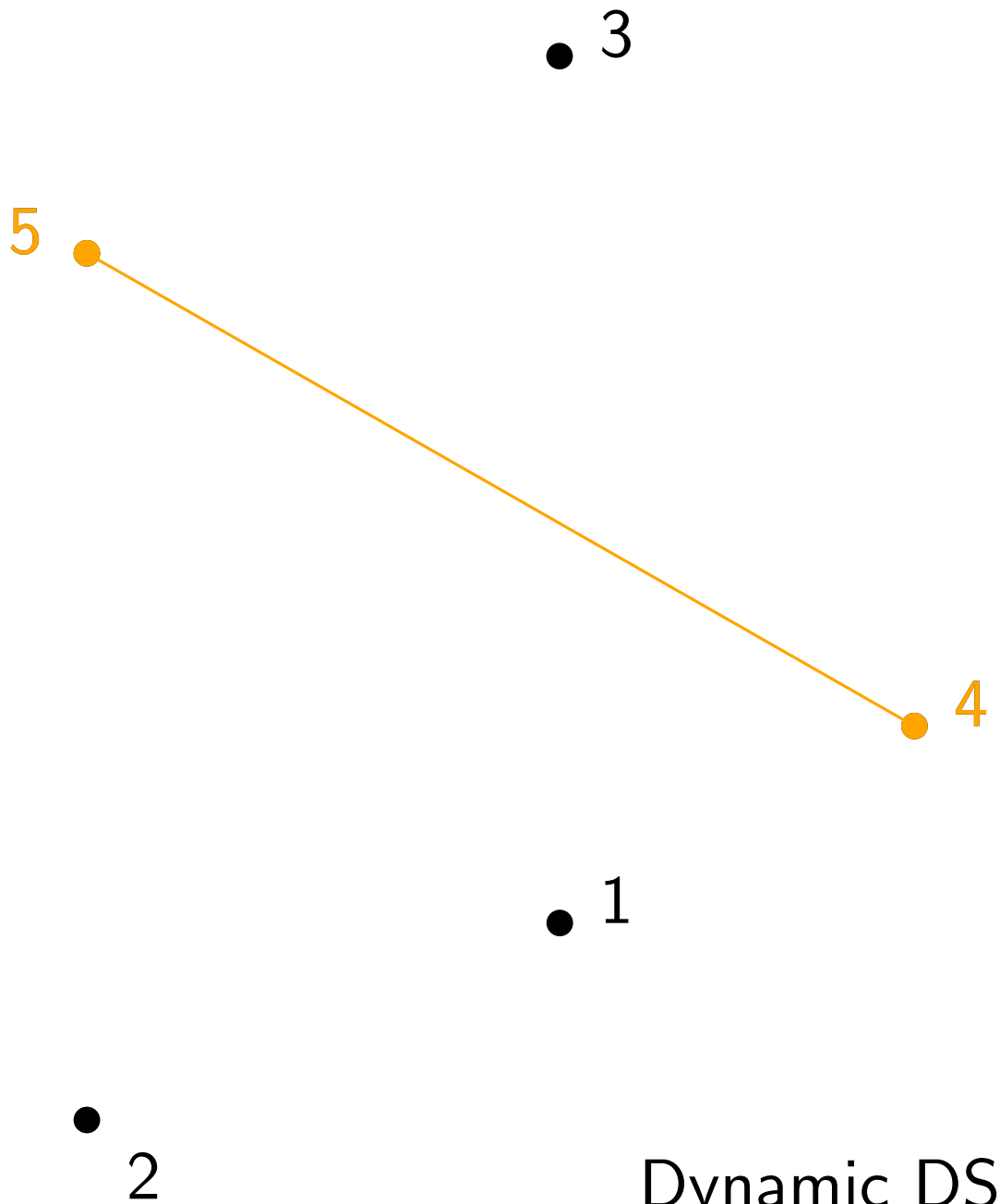


i	j
1	3
2	4
3	5
4	
5	

Dynamic DS : Has property? Yes

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?

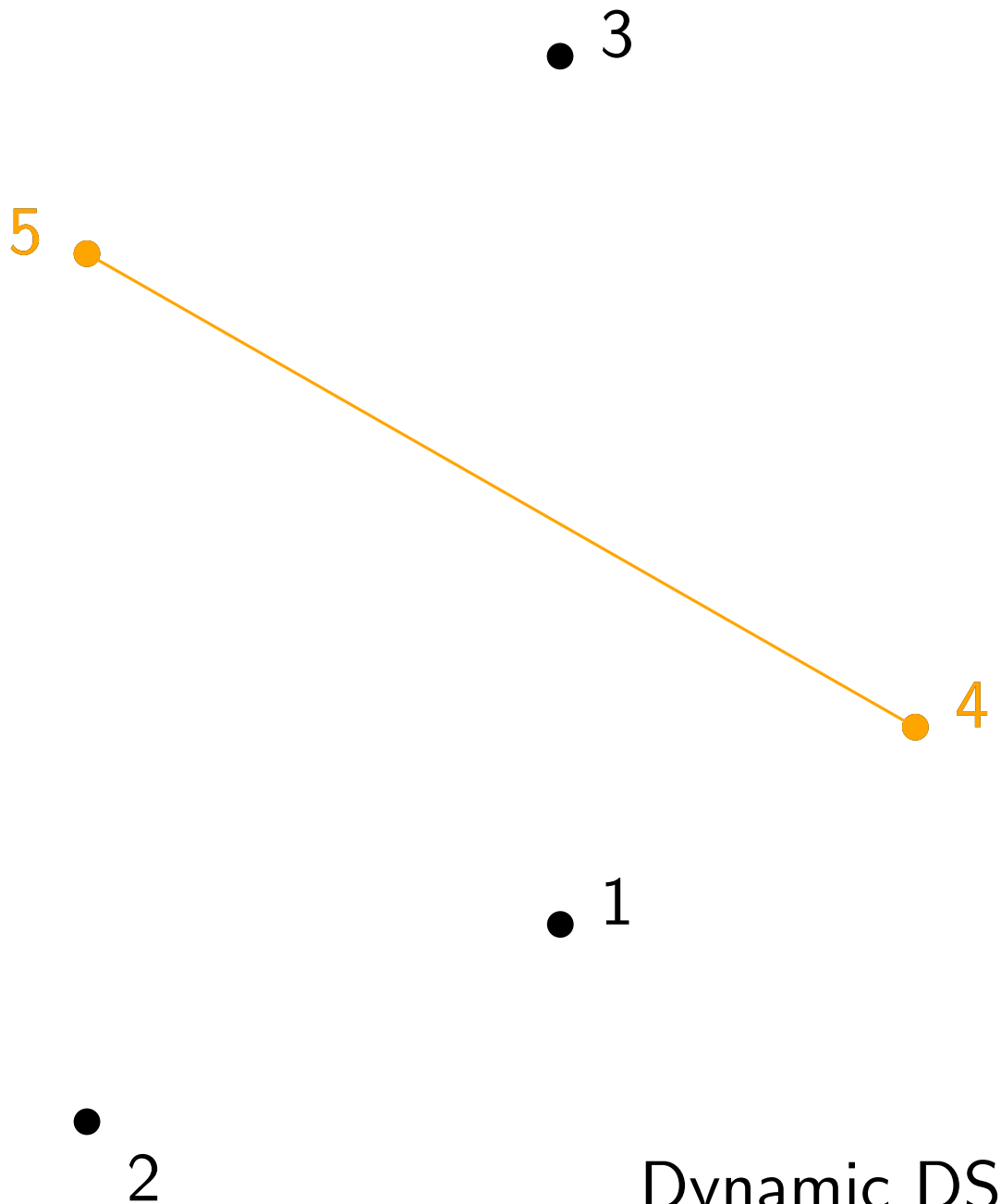


i	j
1	3
2	4
3	5
4	
5	

Dynamic DS : Has property? No

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?

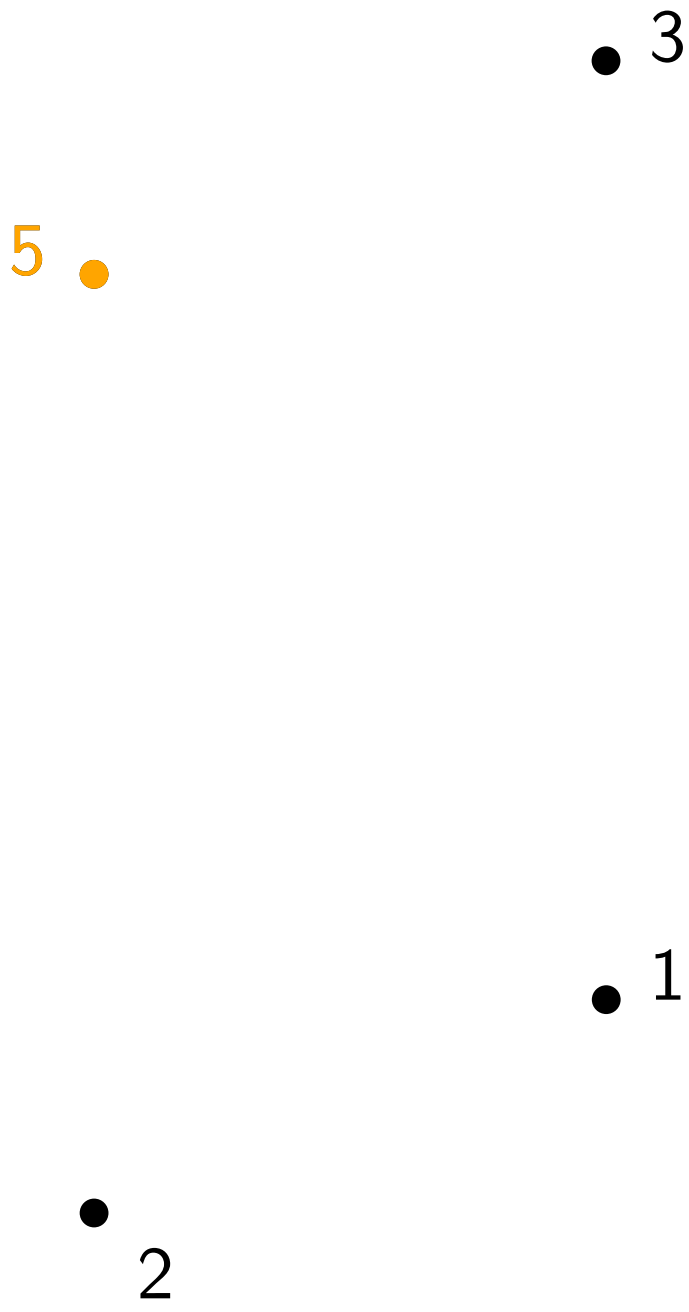


i	j
1	3
2	4
3	5
4	6
5	

Dynamic DS : Has property? No

Time-Windowed → Dynamic

CH area > 1?

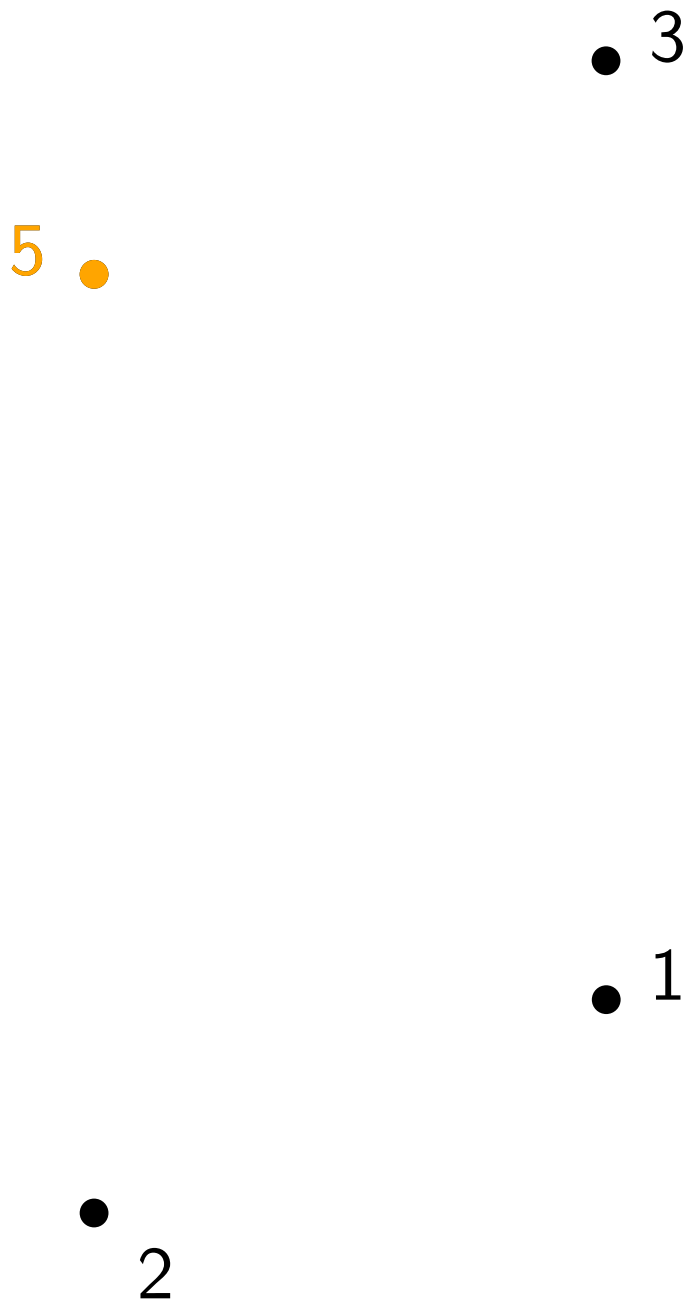


i	j
1	3
2	4
3	5
4	6
5	

Dynamic DS : Has property? No

Time-Windowed → Dynamic

CH area > 1?

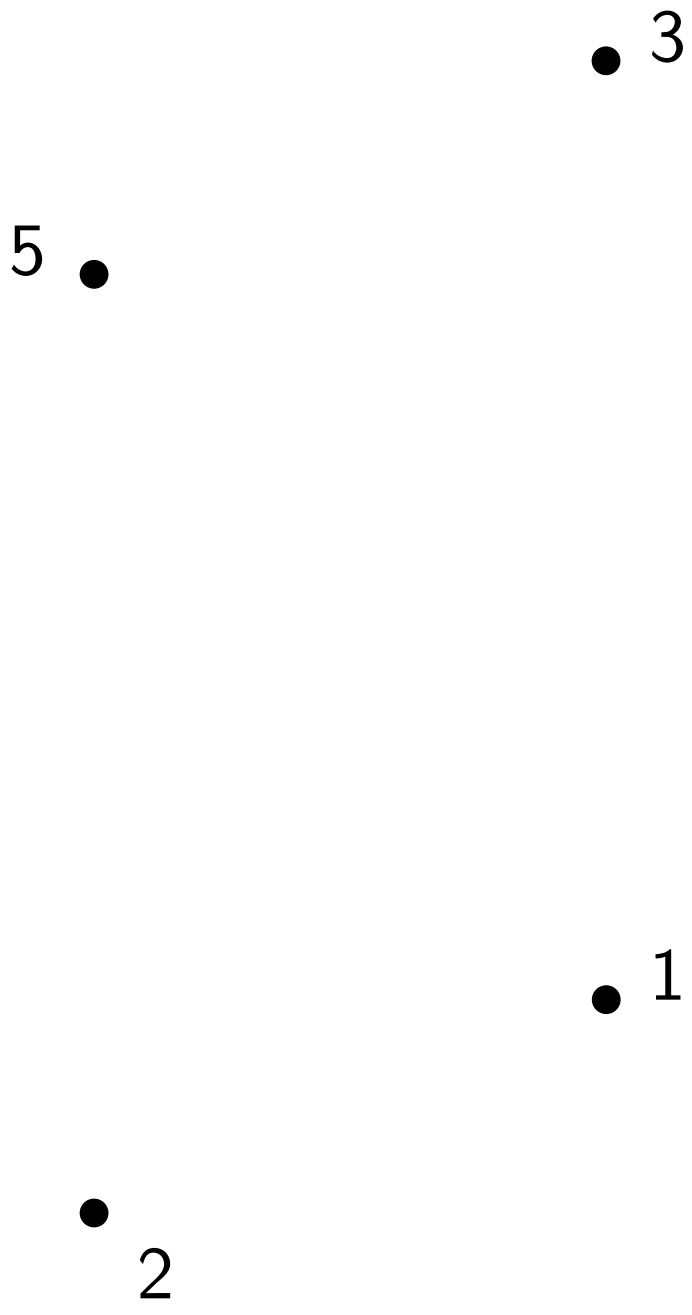


i	j
1	3
2	4
3	5
4	6
5	6

Dynamic DS : Has property? No

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?

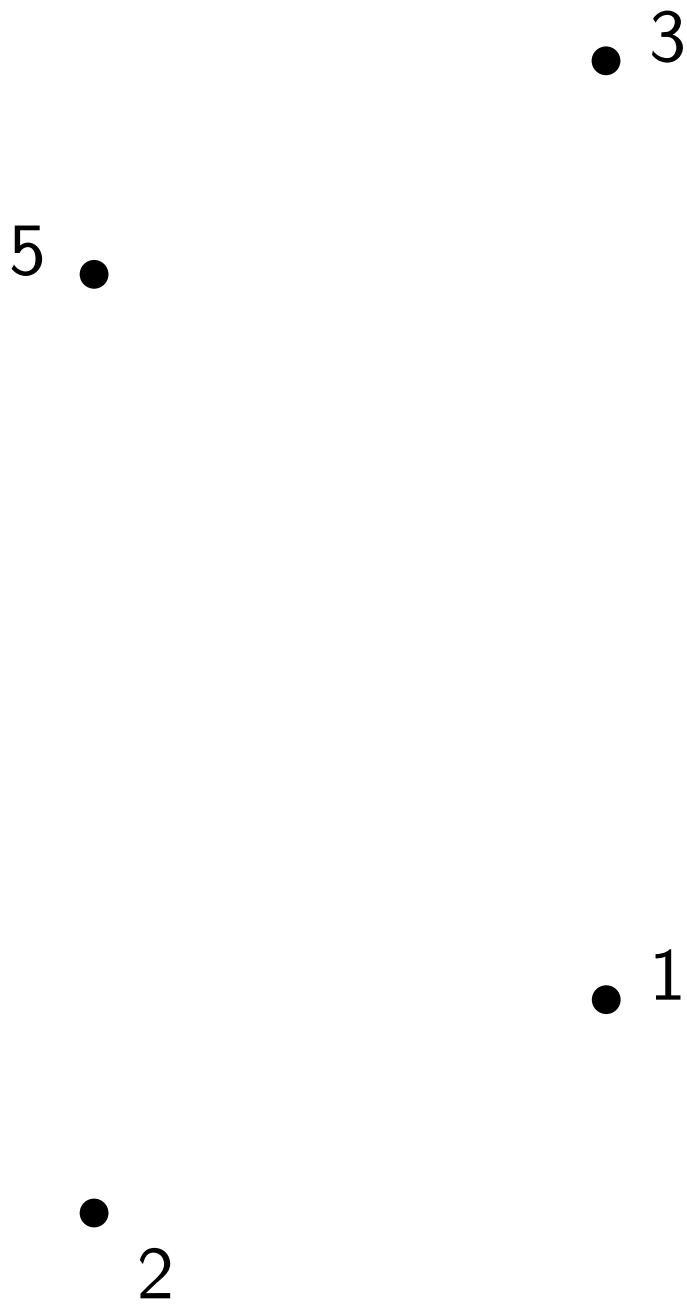


i	j
1	3
2	4
3	5
4	6
5	6

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	6
5	6

Dynamic DS : Has property? No

Time-Windowed → Dynamic

CH area > 1?

Notice the order of insertion/deletion

i	j
1	3
2	4
3	5
4	6
5	6

Dynamic DS : Has property? No

Time-Windowed → Dynamic

CH area > 1?

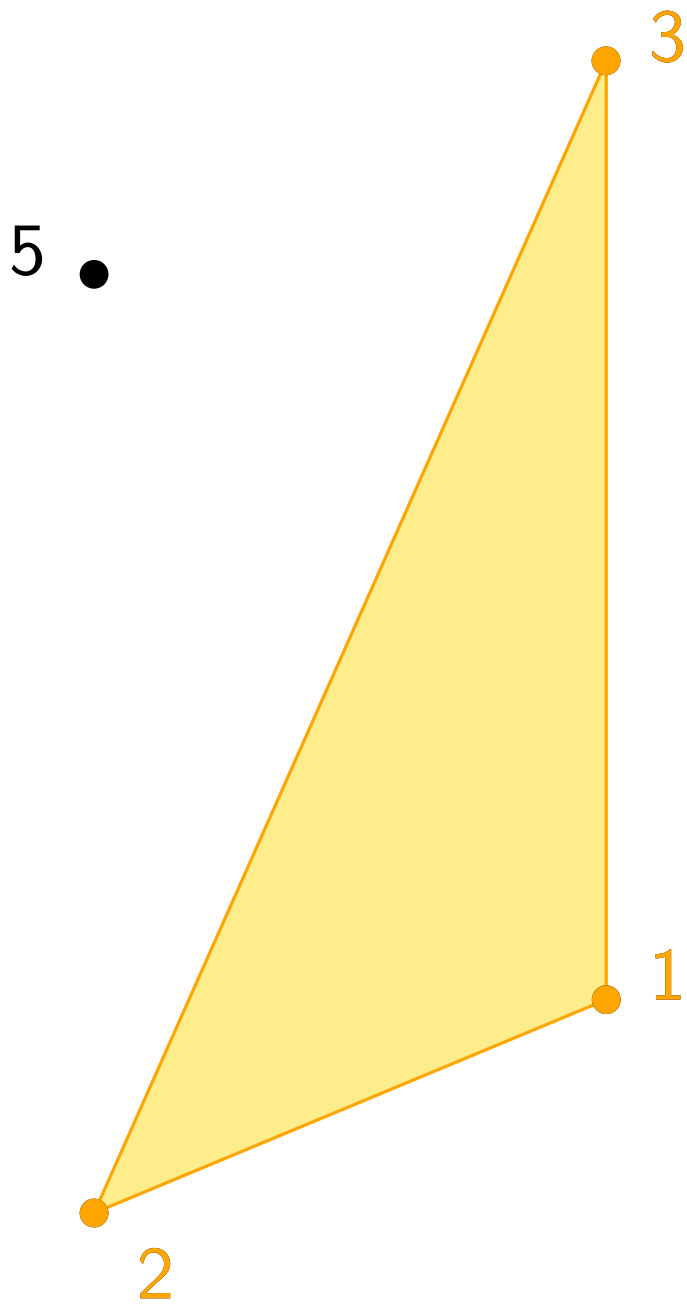
Notice the order of insertion/deletion

i	j
1	3
2	4
3	5
4	6
5	6

Dynamic DS : Has property? No

Time-Windowed → Dynamic

CH area > 1?



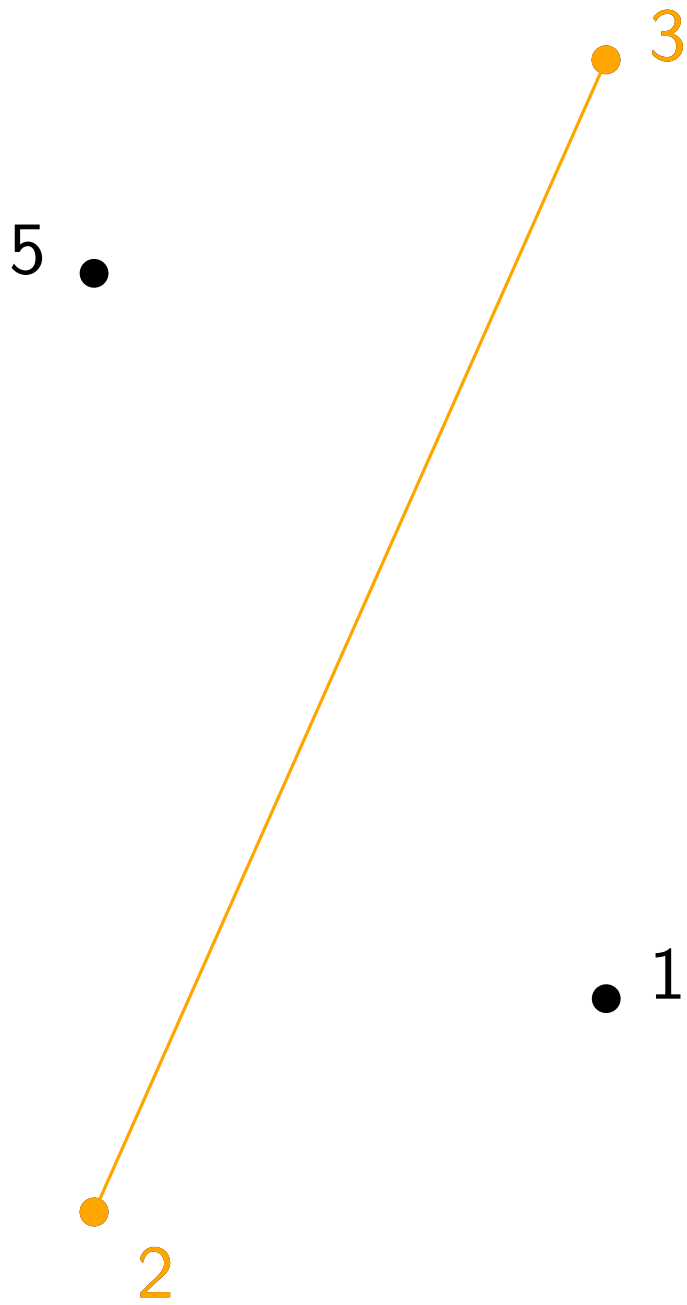
Notice the order of insertion/deletion

i	j
1	3
2	4
3	5
4	6
5	6

Dynamic DS : Has property? No

Time-Windowed → Dynamic

CH area > 1?



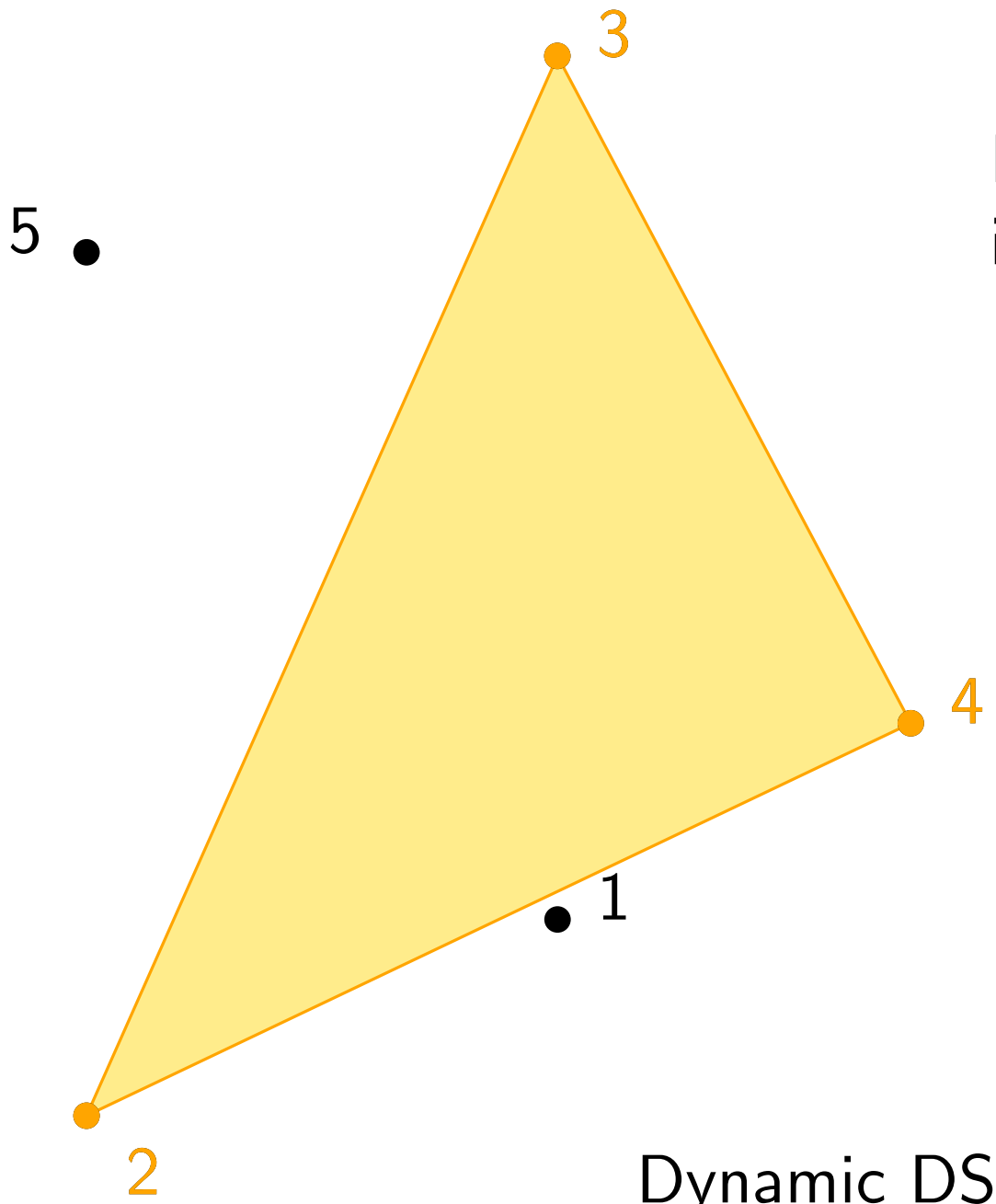
Notice the order of insertion/deletion

i	j
1	3
2	4
3	5
4	6
5	6

Dynamic DS : Has property? No

Time-Windowed → Dynamic

CH area > 1?



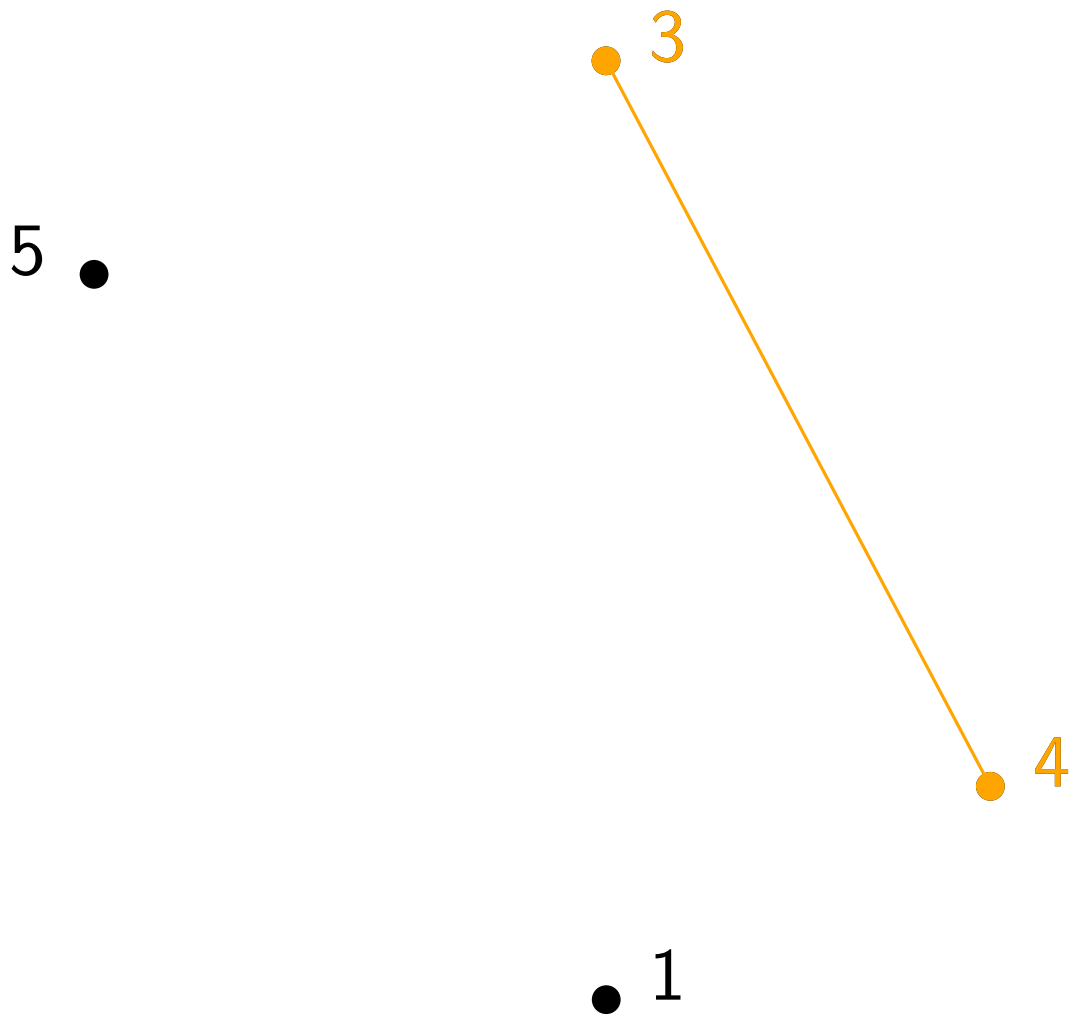
Notice the order of insertion/deletion

i	j
1	3
2	4
3	5
4	6
5	6

Dynamic DS : Has property? No

Time-Windowed → Dynamic

CH area > 1?



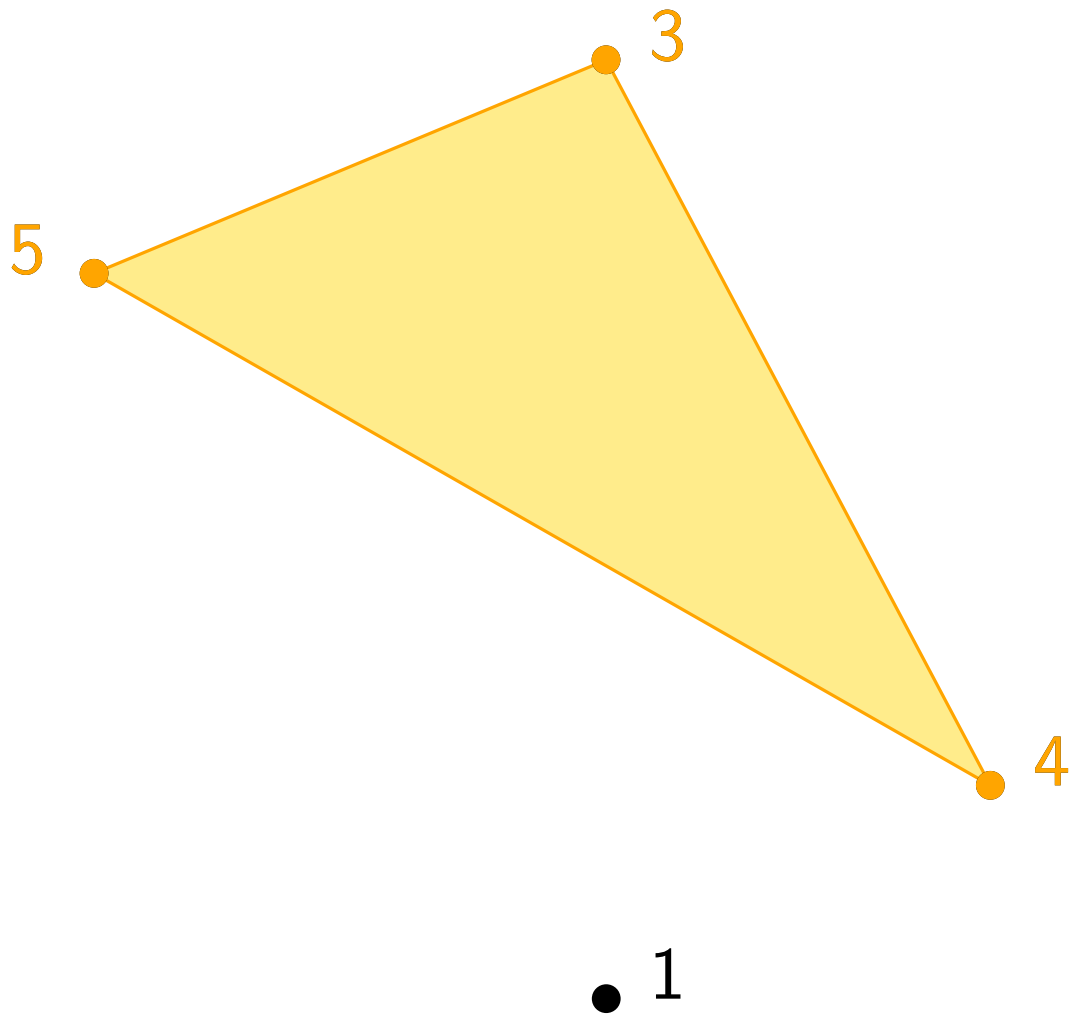
Notice the order of insertion/deletion

i	j
1	3
2	4
3	5
4	6
5	6

Dynamic DS : Has property? No

Time-Windowed → Dynamic

CH area > 1?



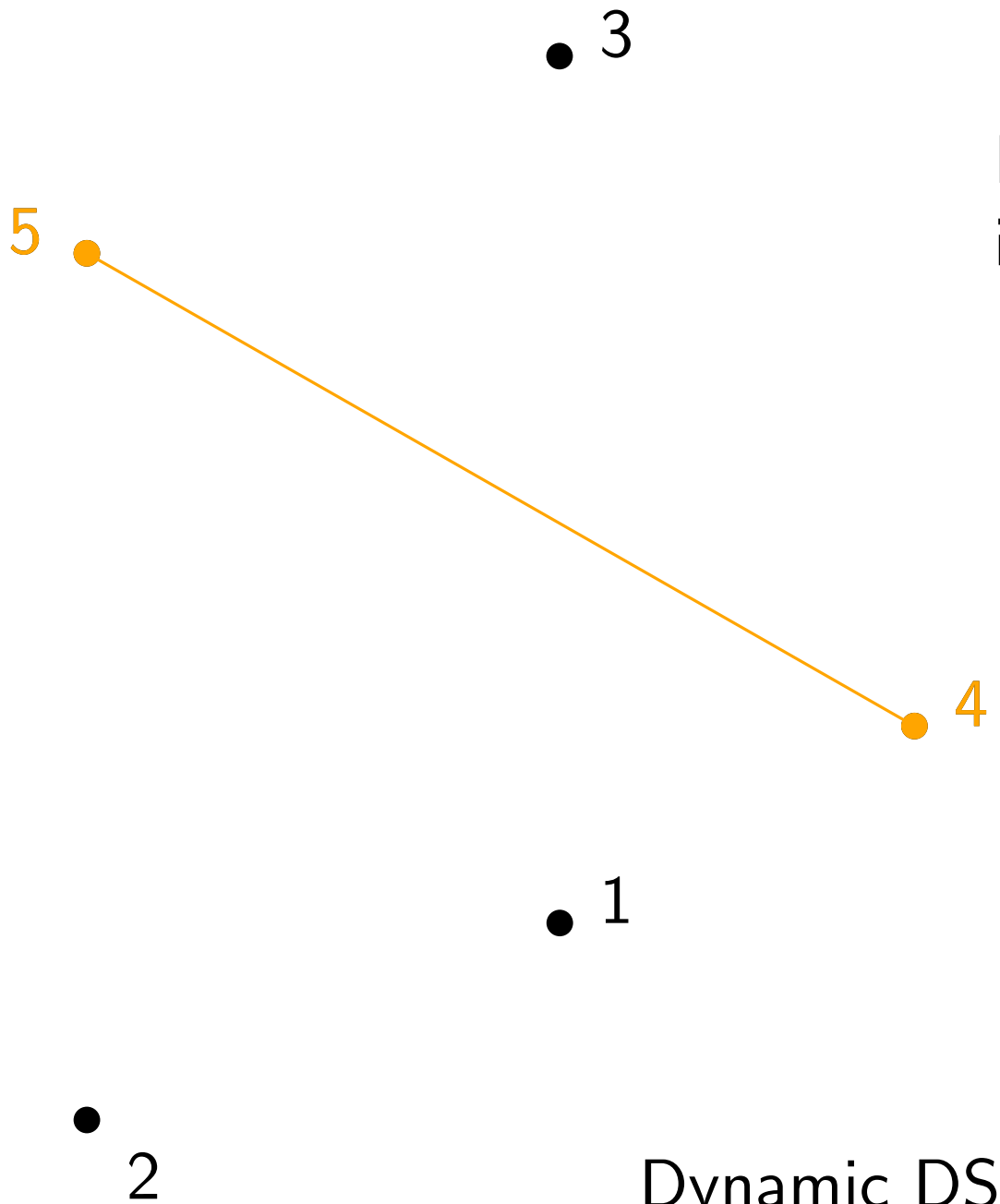
Notice the order of insertion/deletion

i	j
1	3
2	4
3	5
4	6
5	6

Dynamic DS : Has property? No

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?



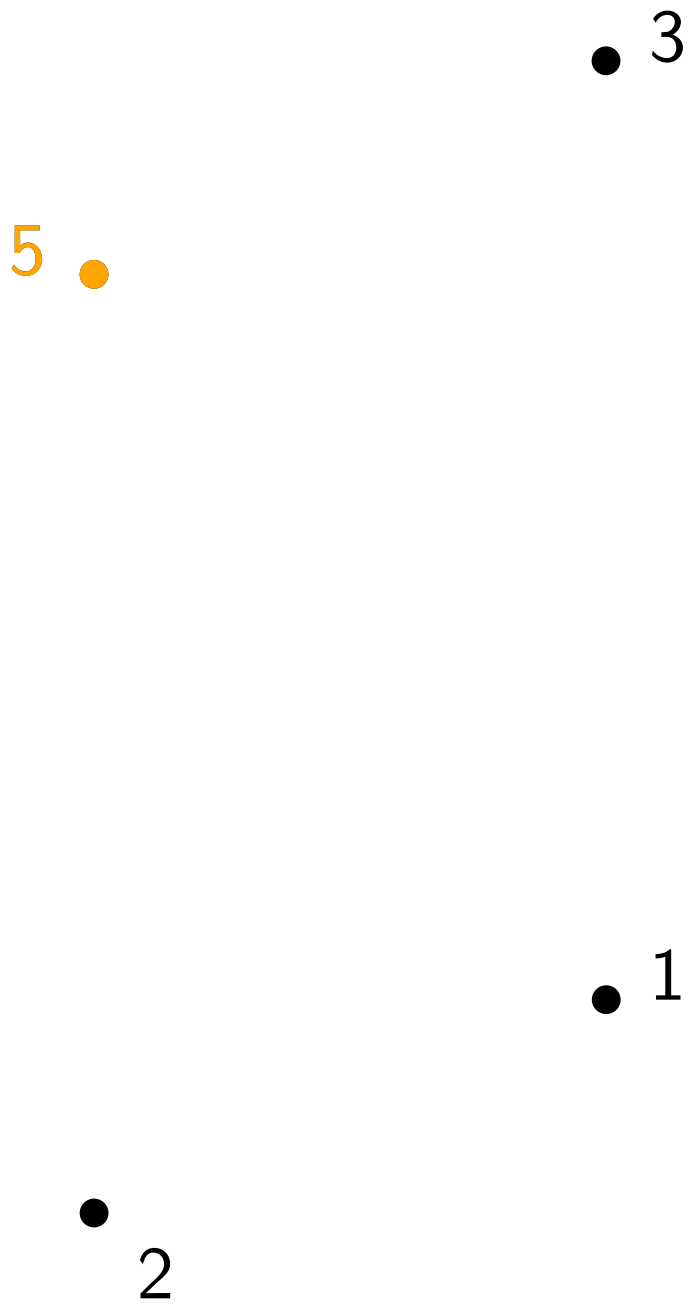
Notice the order of insertion/deletion

i	j
1	3
2	4
3	5
4	6
5	6

Dynamic DS : Has property? No

Time-Windowed → Dynamic

CH area > 1?



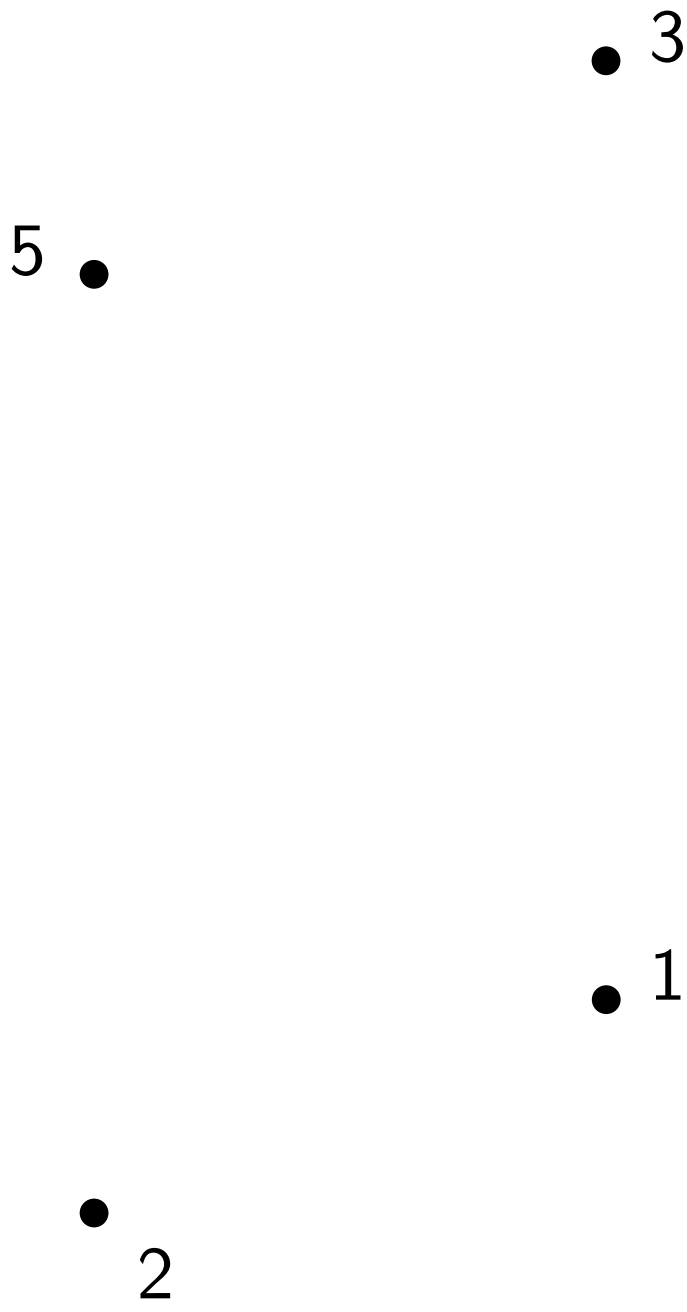
Notice the order of insertion/deletion

i	j
1	3
2	4
3	5
4	6
5	6

Dynamic DS : Has property? No

Time-Windowed $\rightarrow$ Dynamic

CH area > 1?



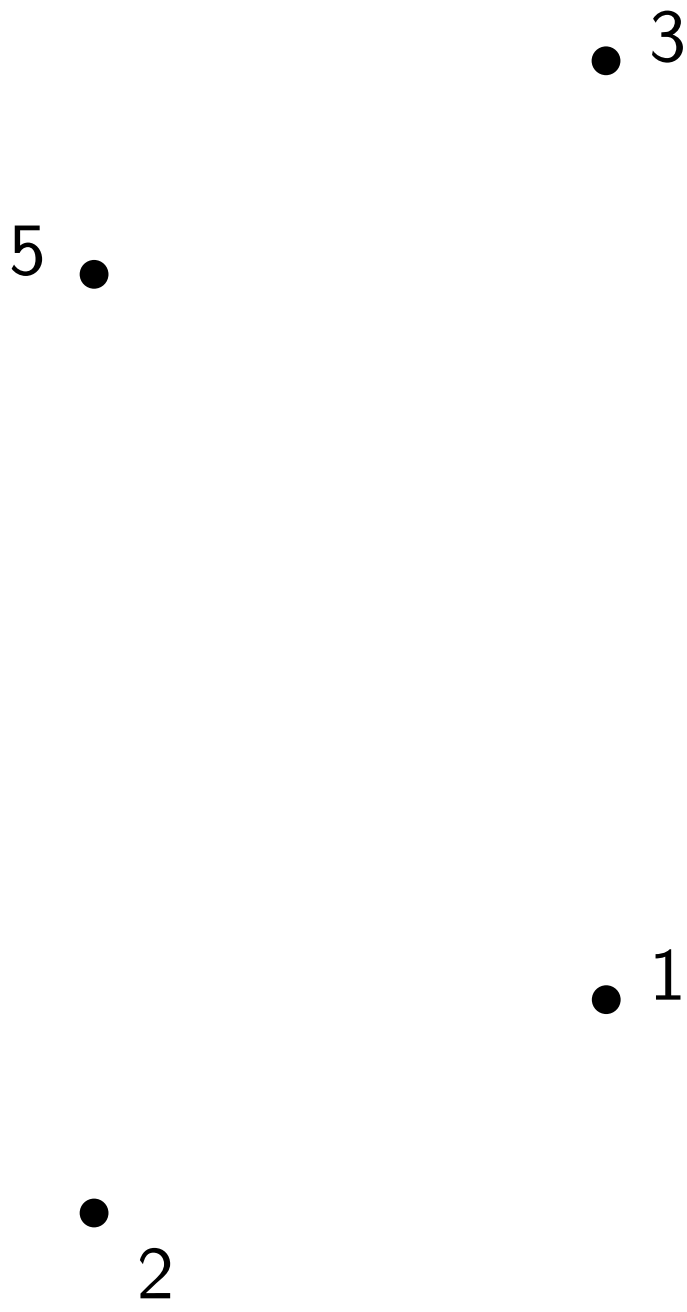
Notice the order of insertion/deletion

i	j
1	3
2	4
3	5
4	6
5	6

Dynamic DS : Has property? No

Time-Windowed → Dynamic

CH area > 1?



Notice the order of insertion/deletion

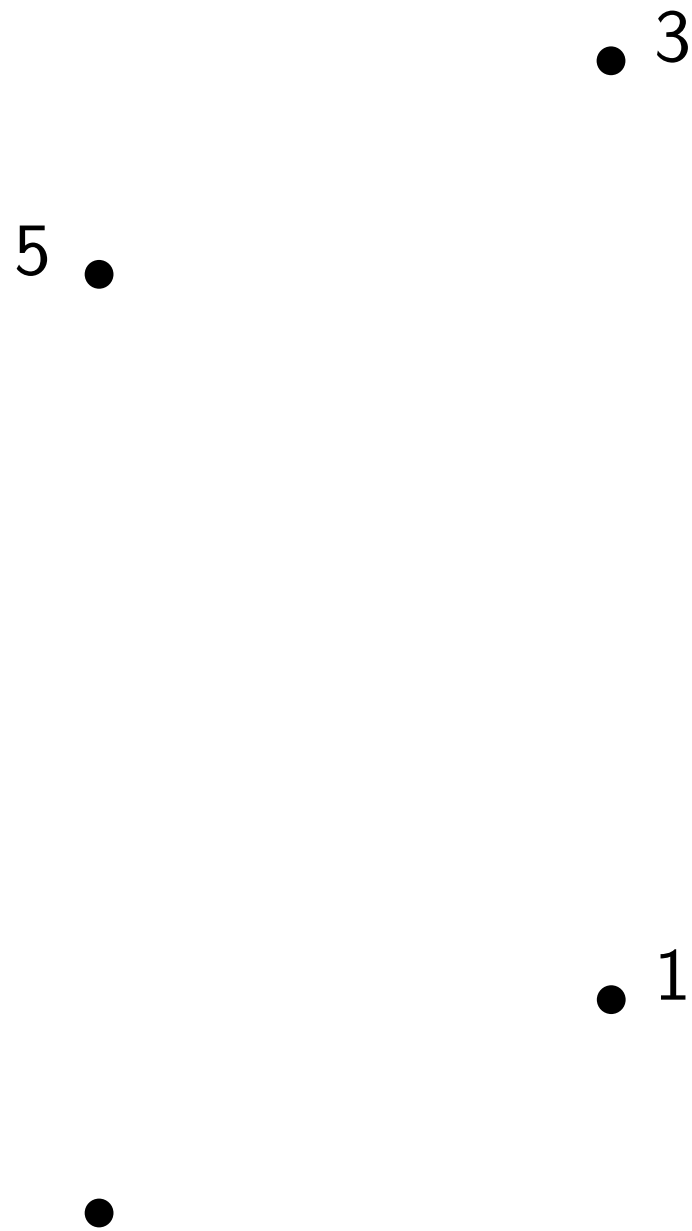
First-in, first-out: like a queue

i	j
1	3
2	4
3	5
4	6
5	6

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	6
5	6

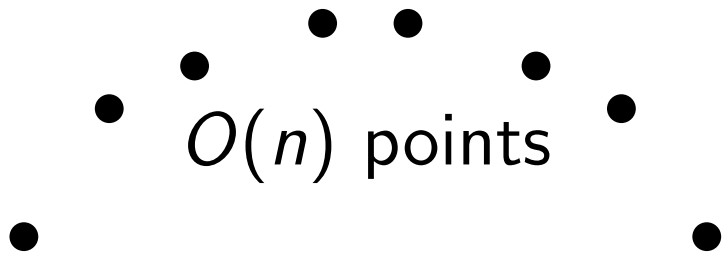
Dynamic DS : Has property? No

Dynamic Convex Hull Complexity

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

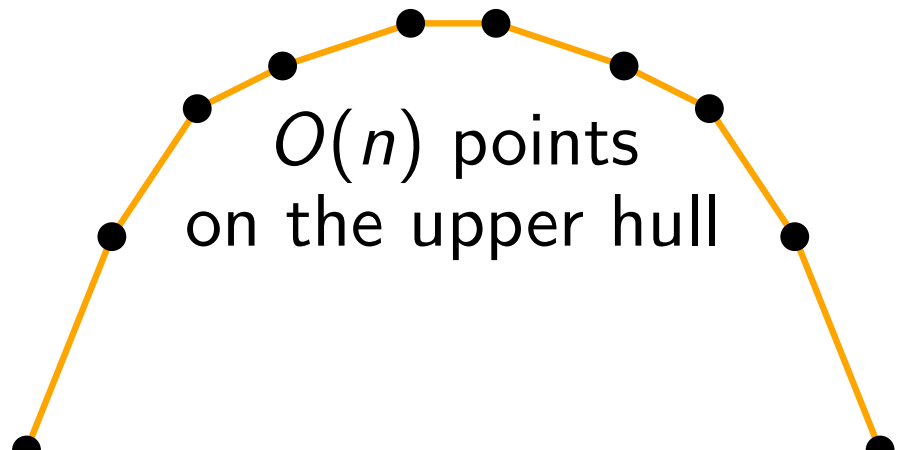
Dynamic Convex Hull Complexity

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



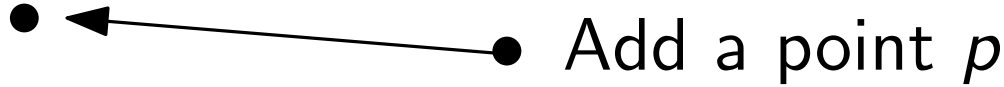
Dynamic Convex Hull Complexity

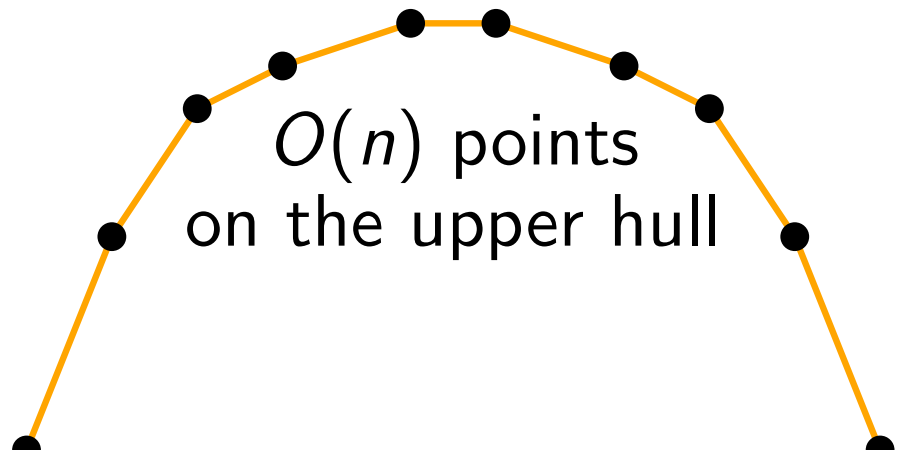
Generally, $O(n^2)$ changes over $O(n)$ updates



Dynamic Convex Hull Complexity

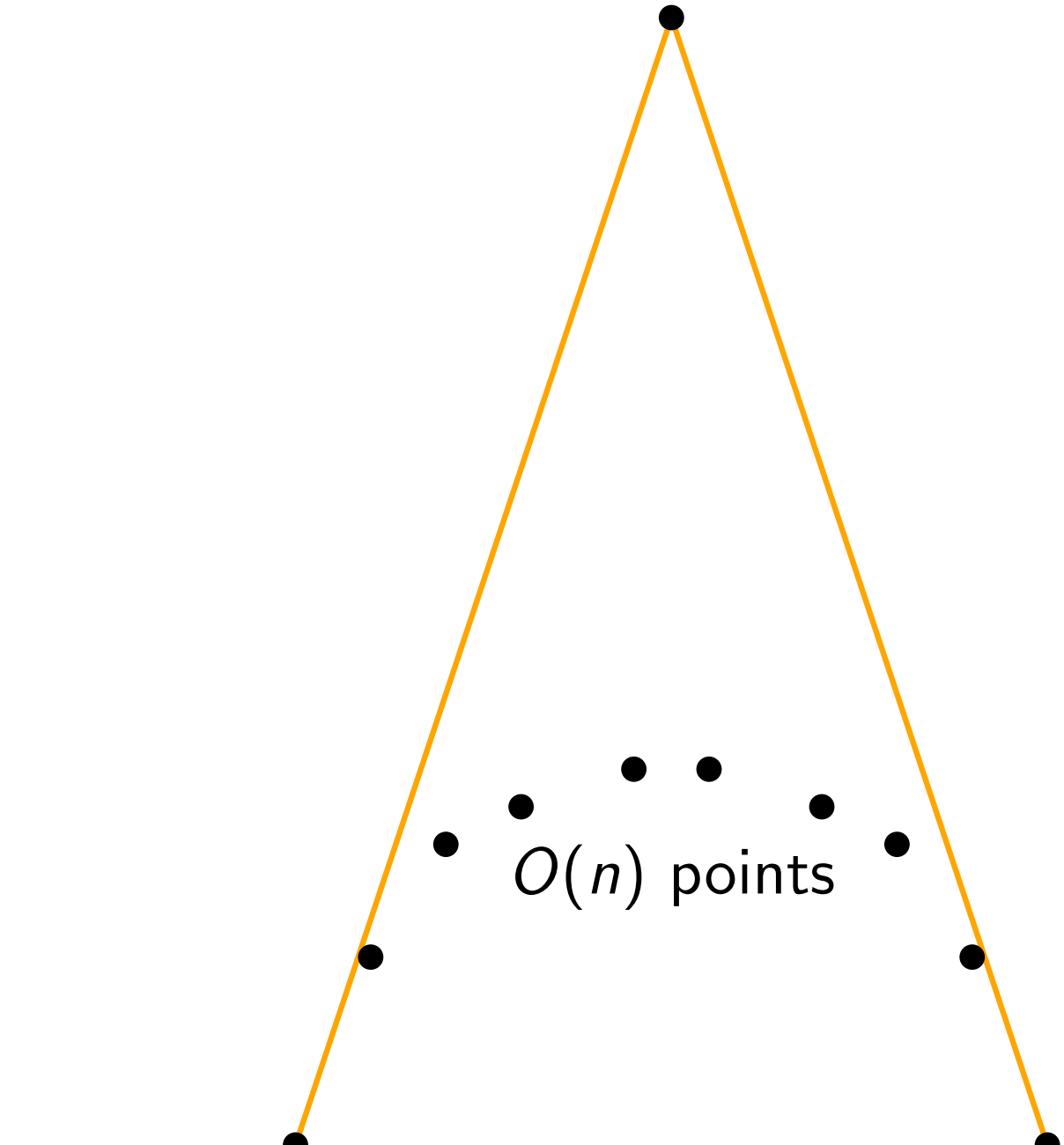
Generally, $O(n^2)$ changes over $O(n)$ updates





Dynamic Convex Hull Complexity

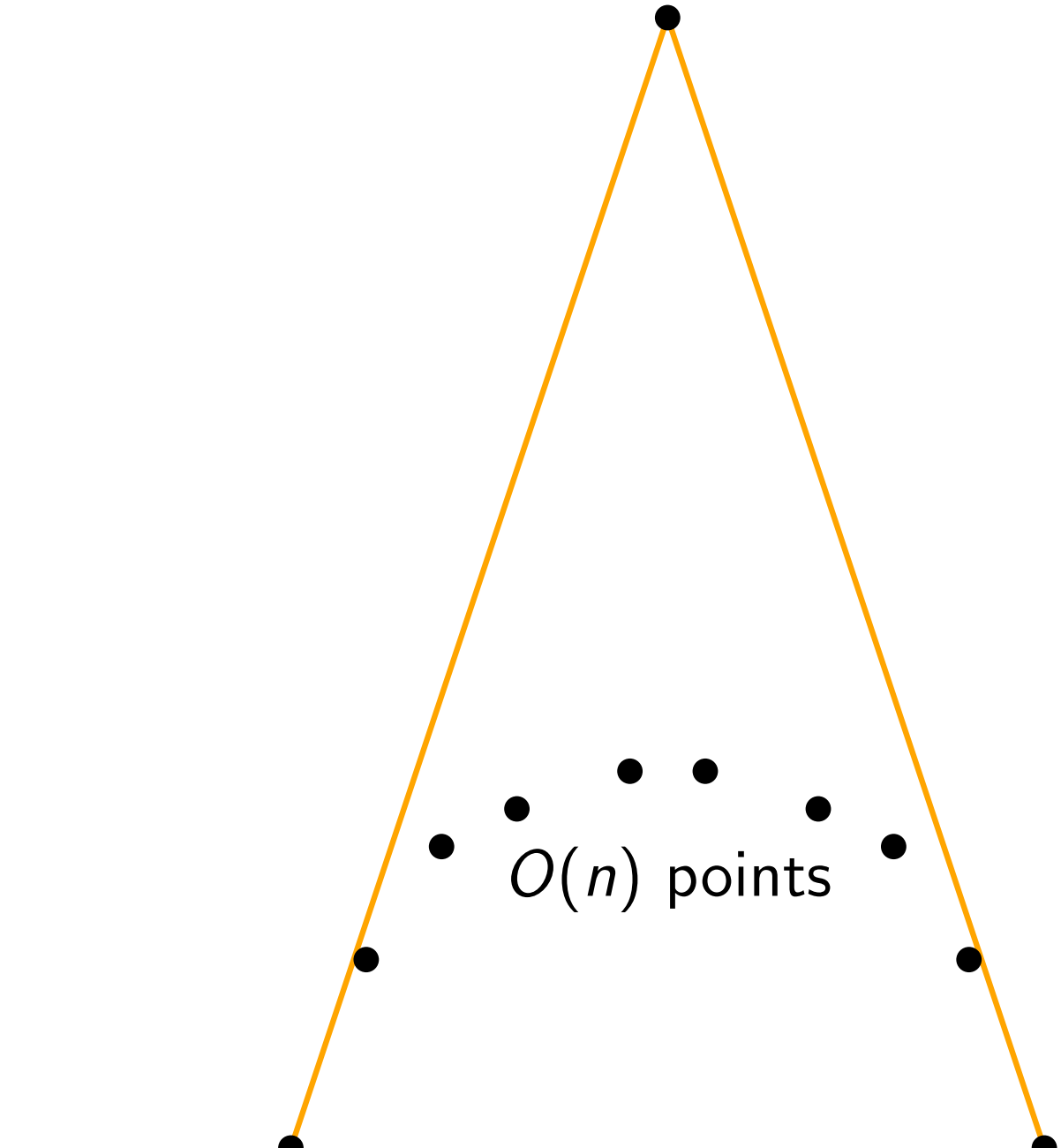
Generally, $O(n^2)$ changes over $O(n)$ updates



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

Dynamic Convex Hull Complexity

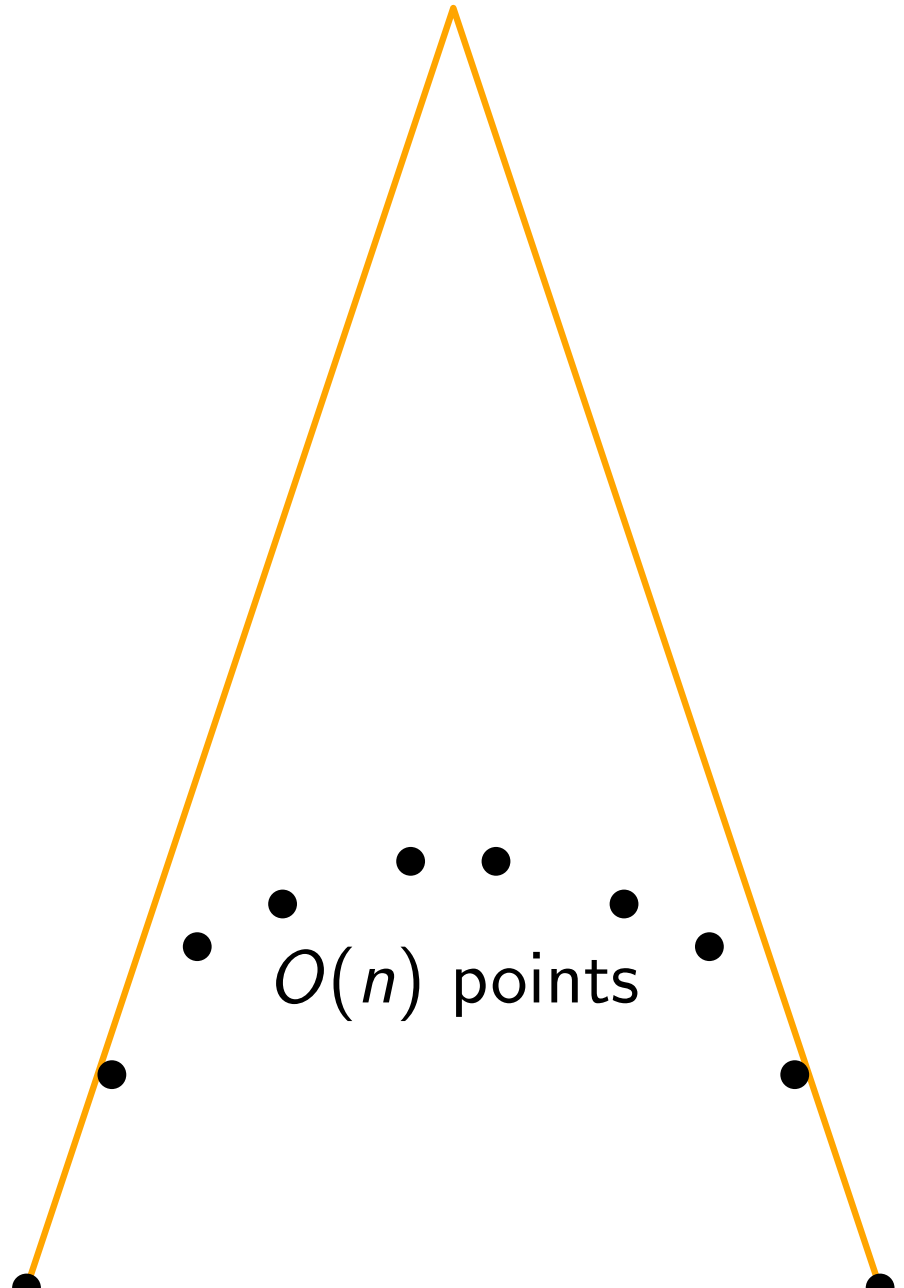
Generally, $O(n^2)$ changes over $O(n)$ updates



- 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

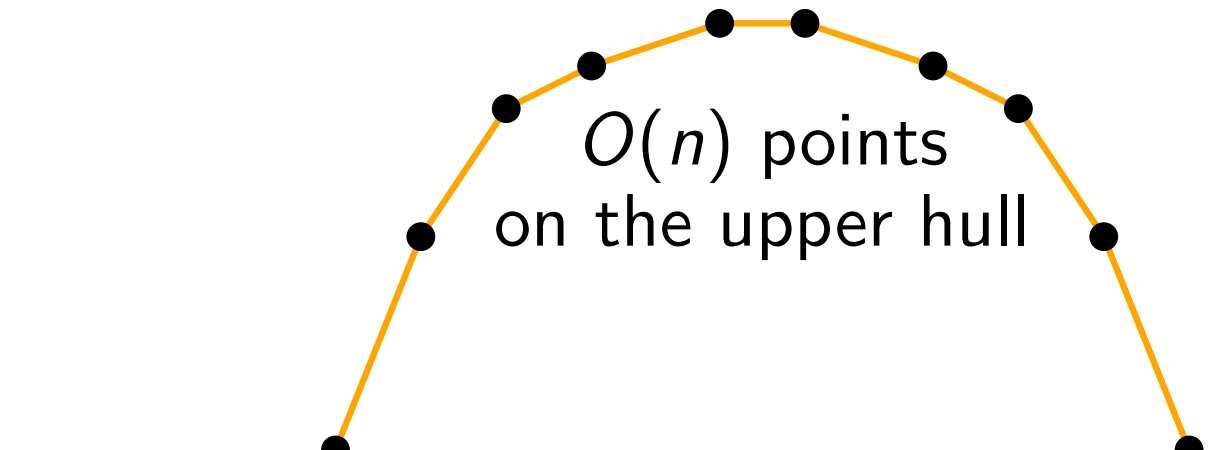


- Add a point p
- $O(n)$ changes
- Add/remove p O(n) times

Dynamic Convex Hull Complexity

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

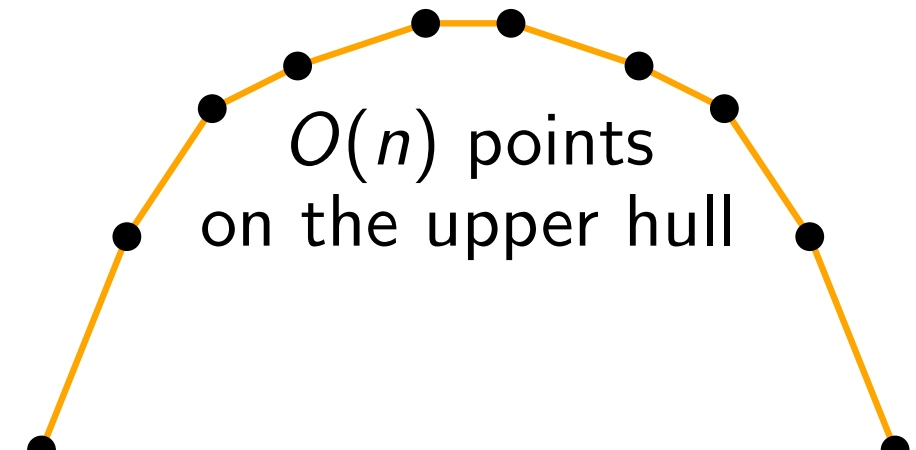
- Add a point p
- $O(n)$ changes
- Add/remove p O(n) times



Dynamic Convex Hull Complexity

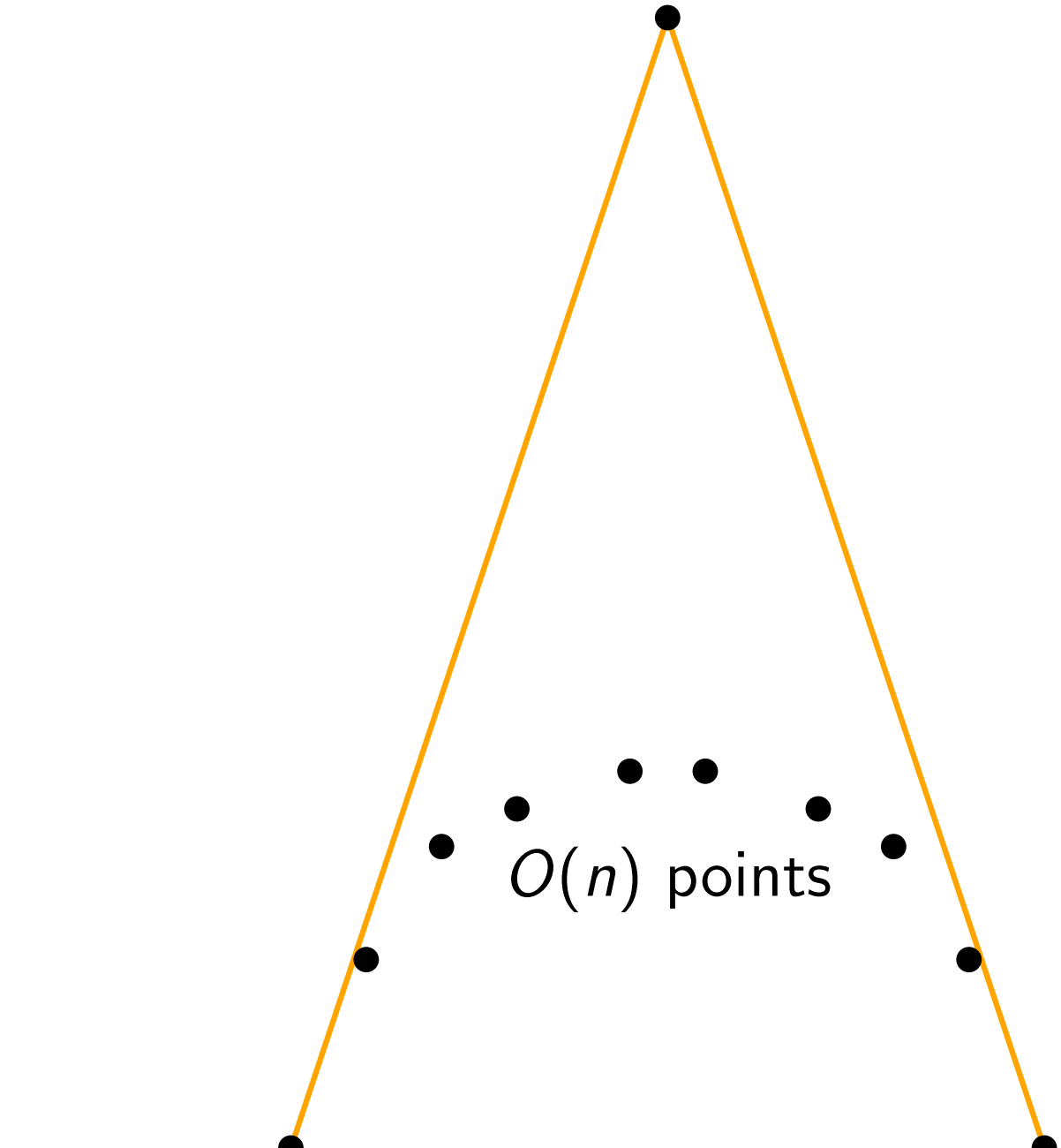
Generally, $O(n^2)$ changes over $O(n)$ updates

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



Dynamic Convex Hull Complexity

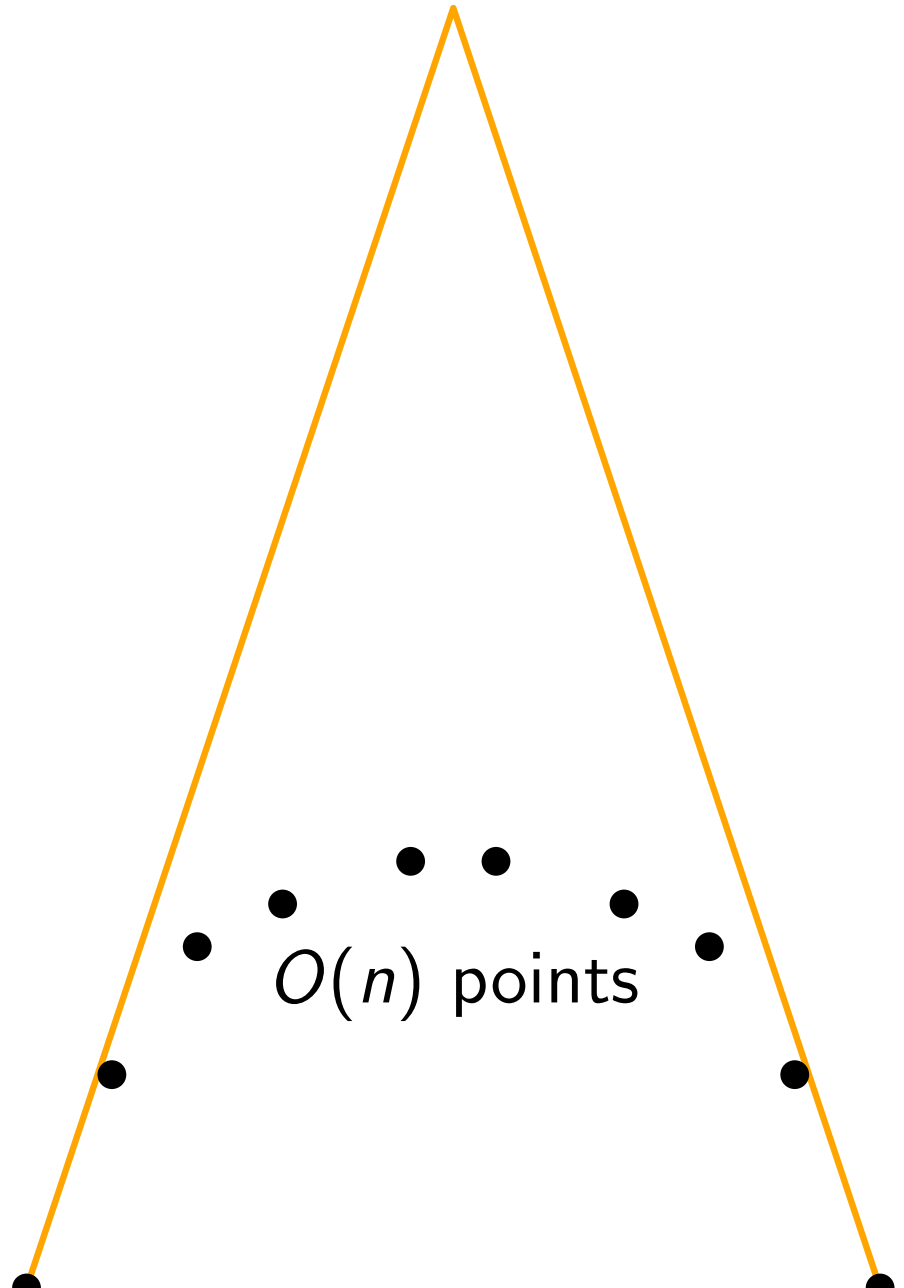
Generally, $O(n^2)$ changes over $O(n)$ updates



- 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

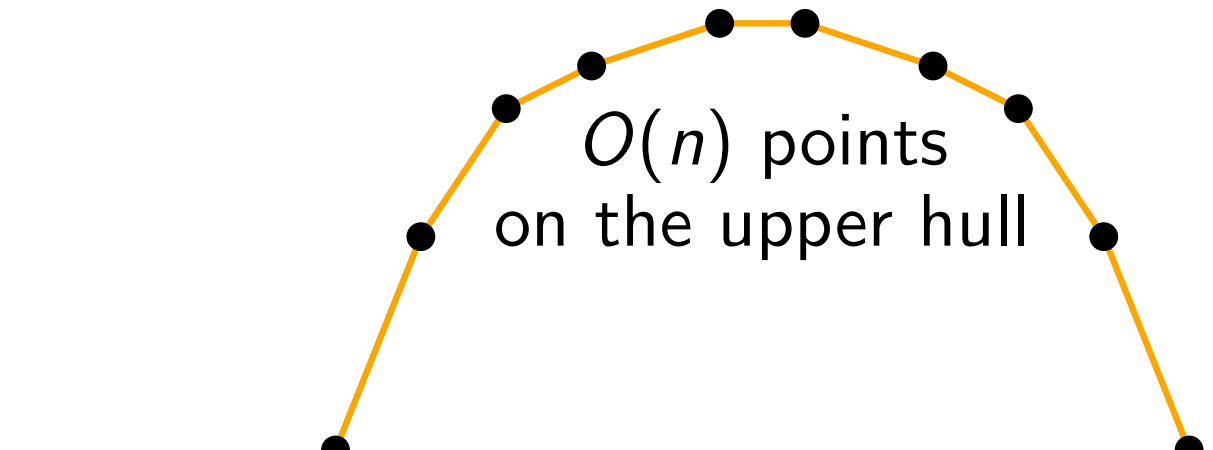


- Add a point p
- $O(n)$ changes
- Add/remove p O(n) times

Dynamic Convex Hull Complexity

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

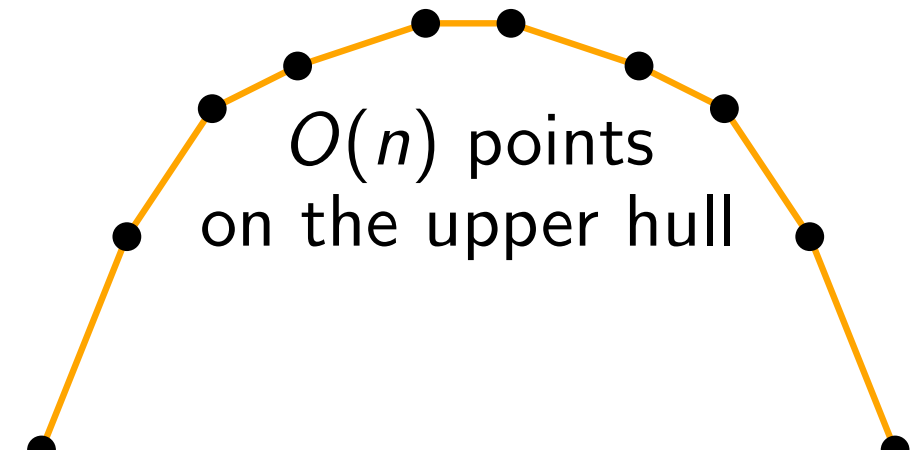
- Add a point p
- $O(n)$ changes
- Add/remove p O(n) times



Dynamic Convex Hull Complexity

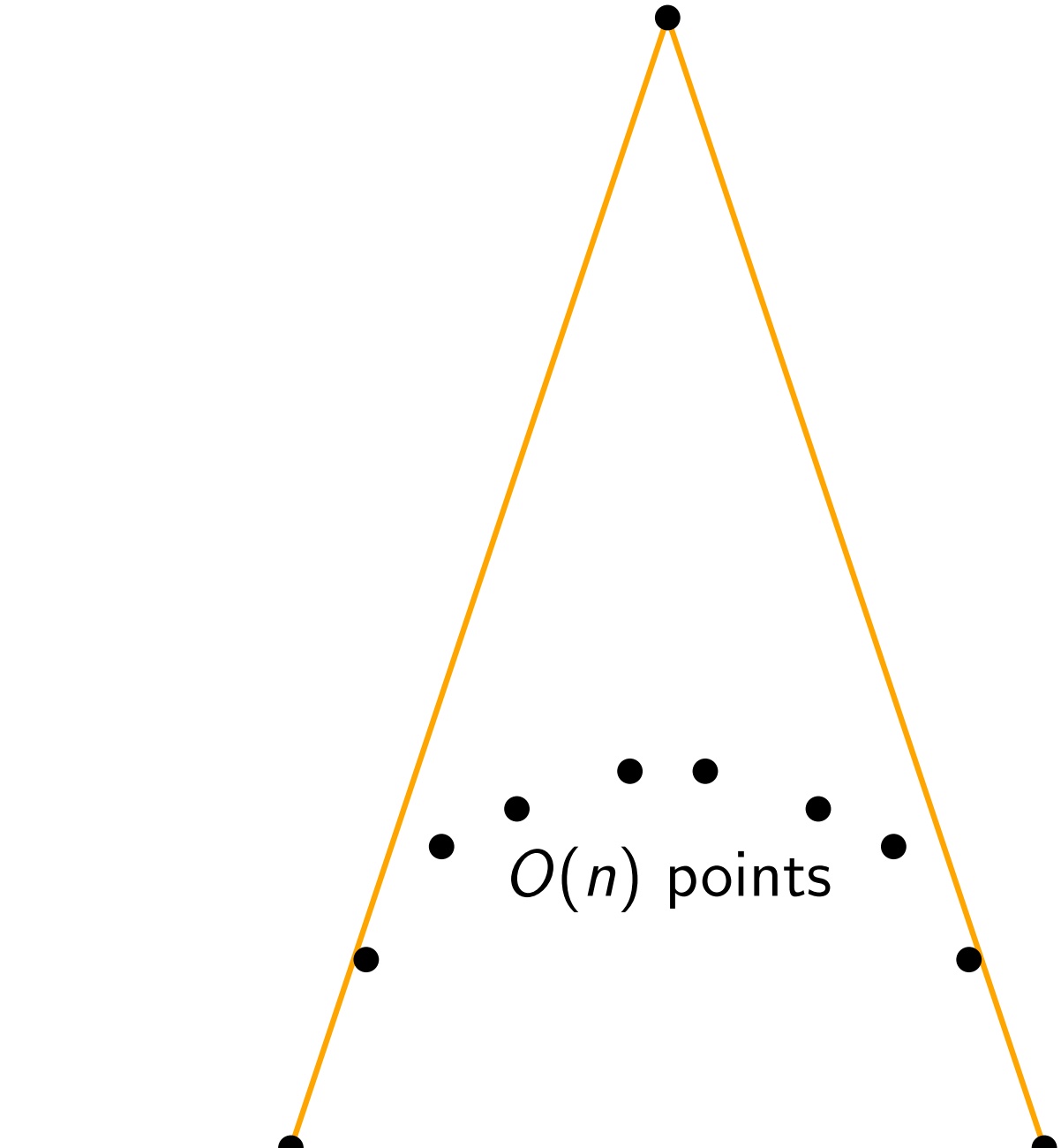
Generally, $O(n^2)$ changes over $O(n)$ updates

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



Dynamic Convex Hull Complexity

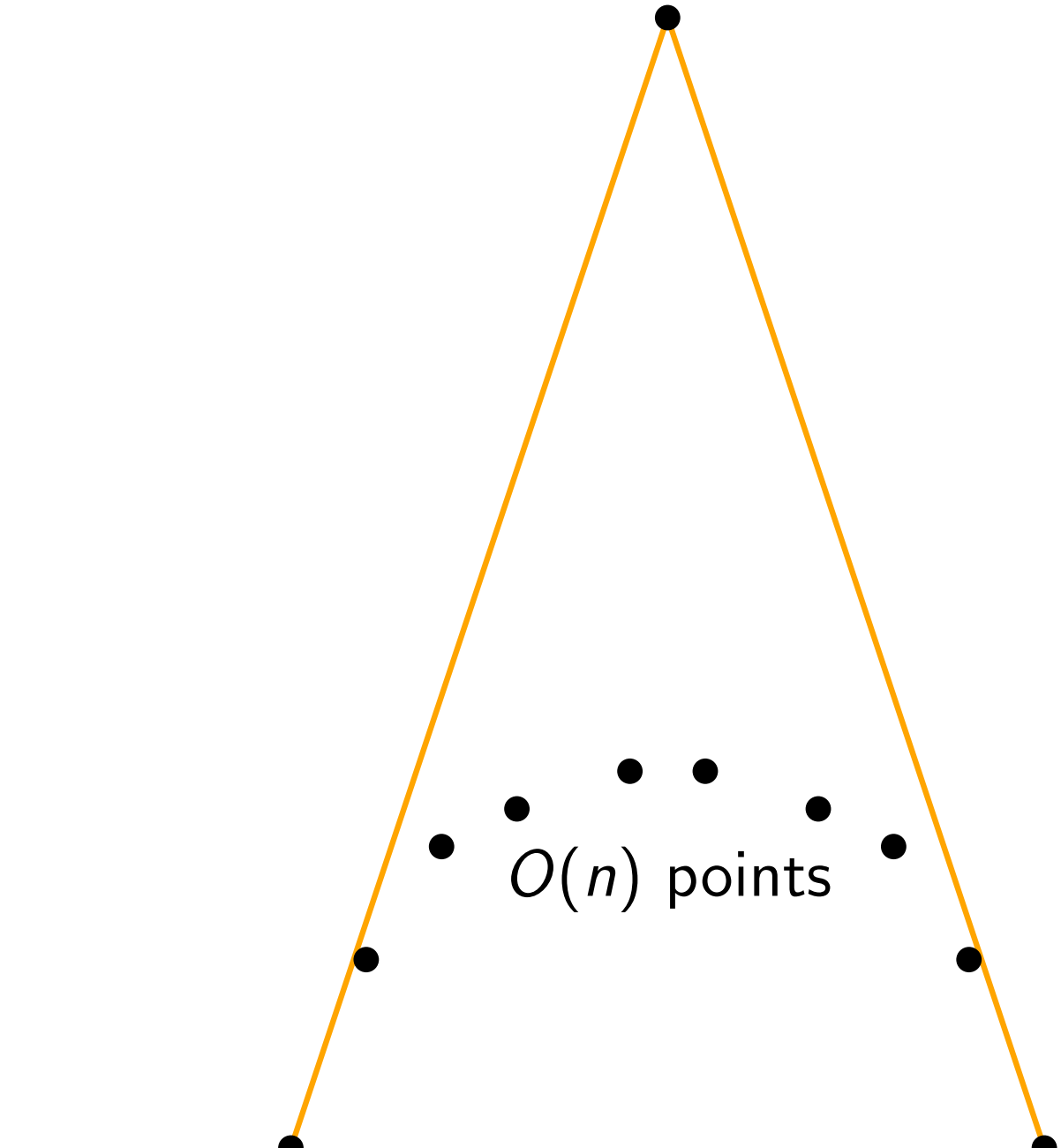
Generally, $O(n^2)$ changes over $O(n)$ updates



- 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

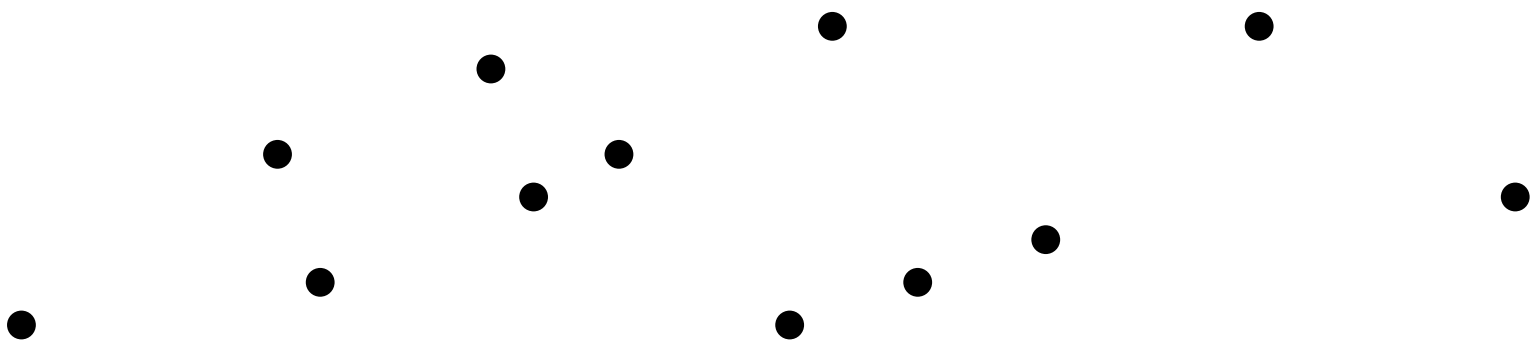
Lemma: # of structural changes to upper hull after n FIFO updates is $O(n \log n)$.

distinct upper hull edges over n updates

FIFO Convex Hull Complexity

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

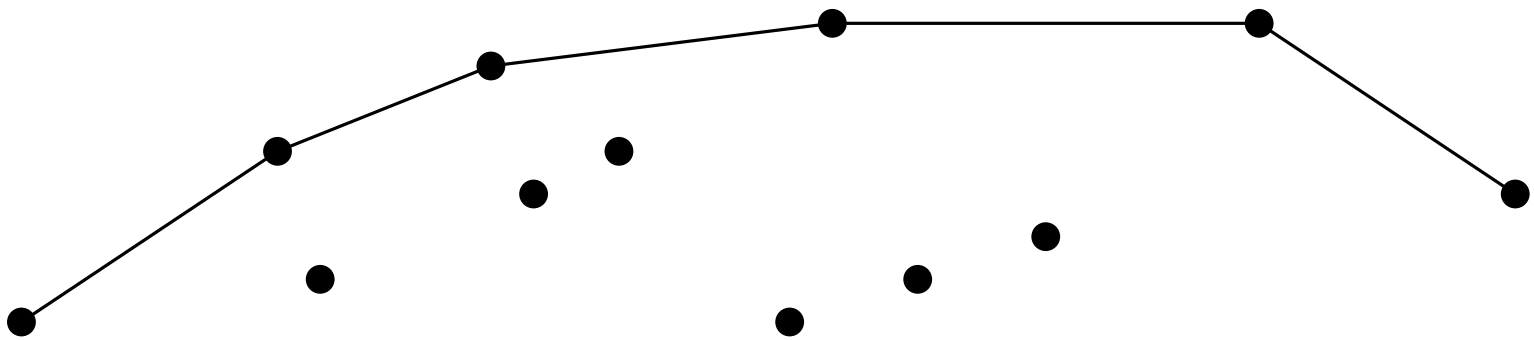
distinct upper hull edges over n updates



FIFO Convex Hull Complexity

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

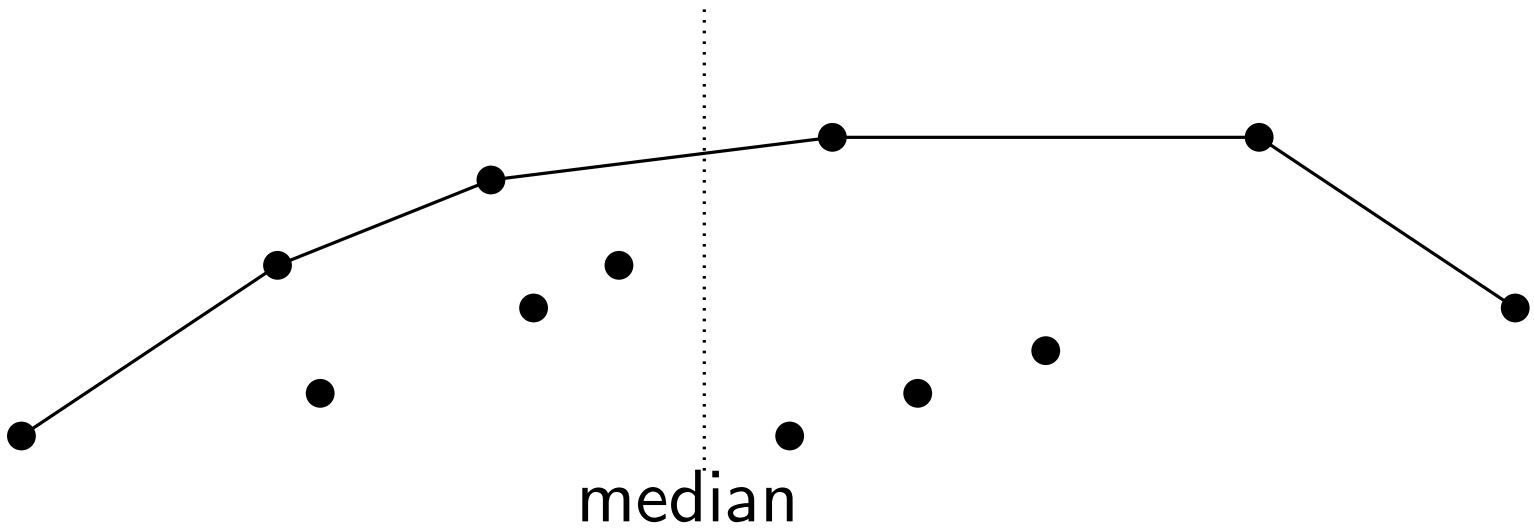
distinct upper hull edges over n updates



FIFO Convex Hull Complexity

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

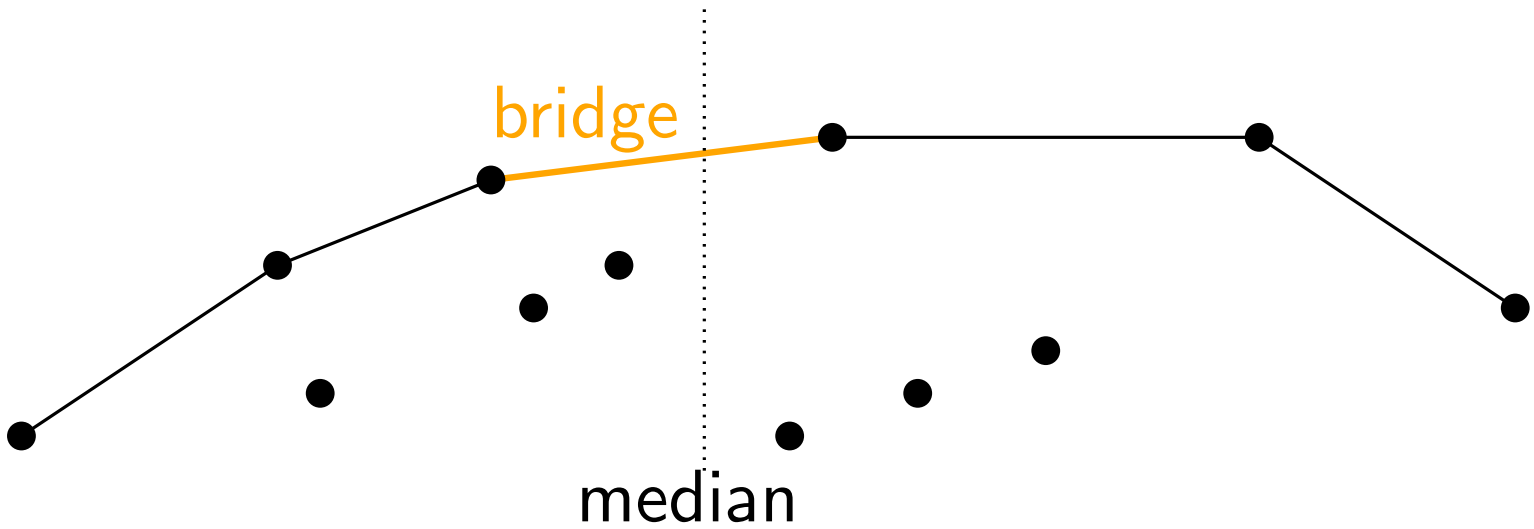
distinct upper hull edges over n updates



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

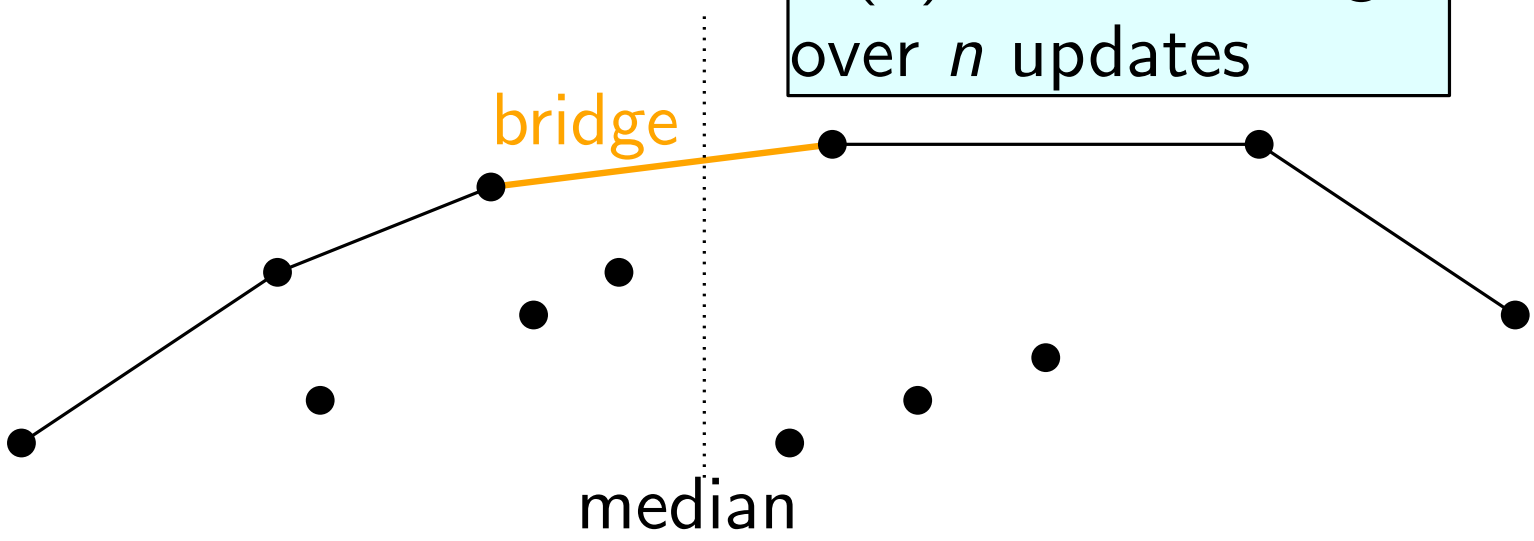


FIFO Convex Hull Complexity

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



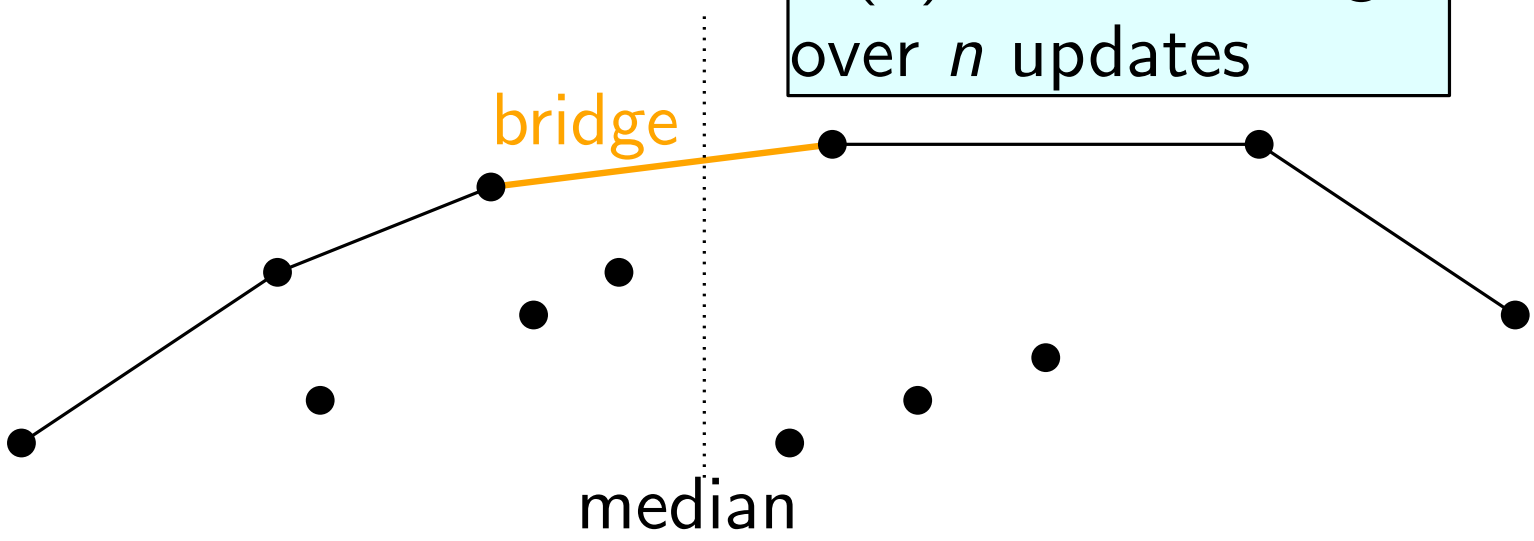
FIFO Convex Hull Complexity

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



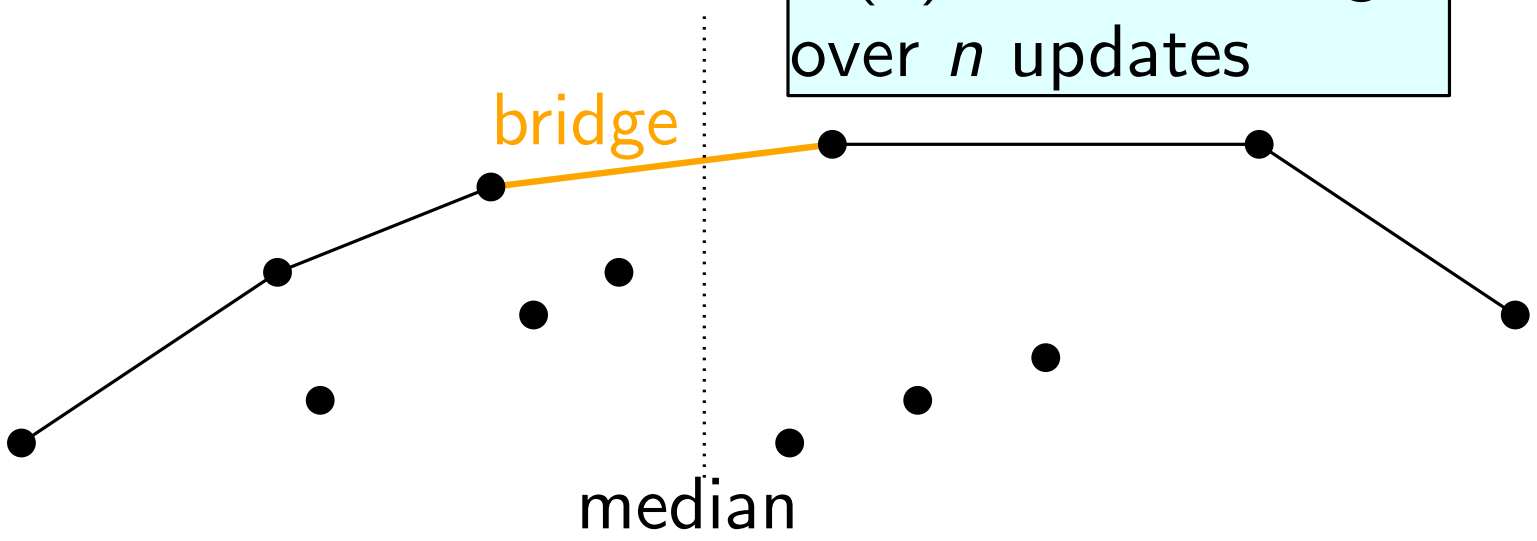
FIFO Convex Hull Complexity

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

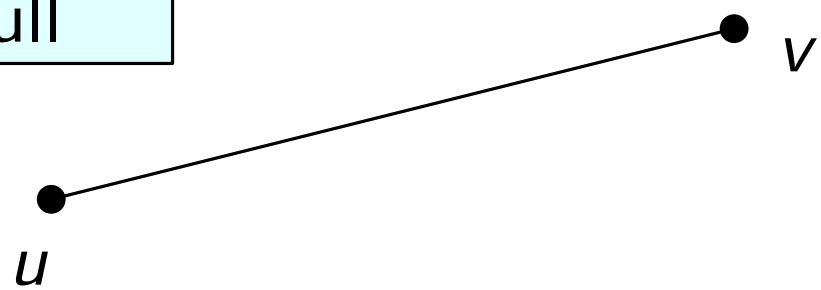
FIFO Convex Hull Complexity

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



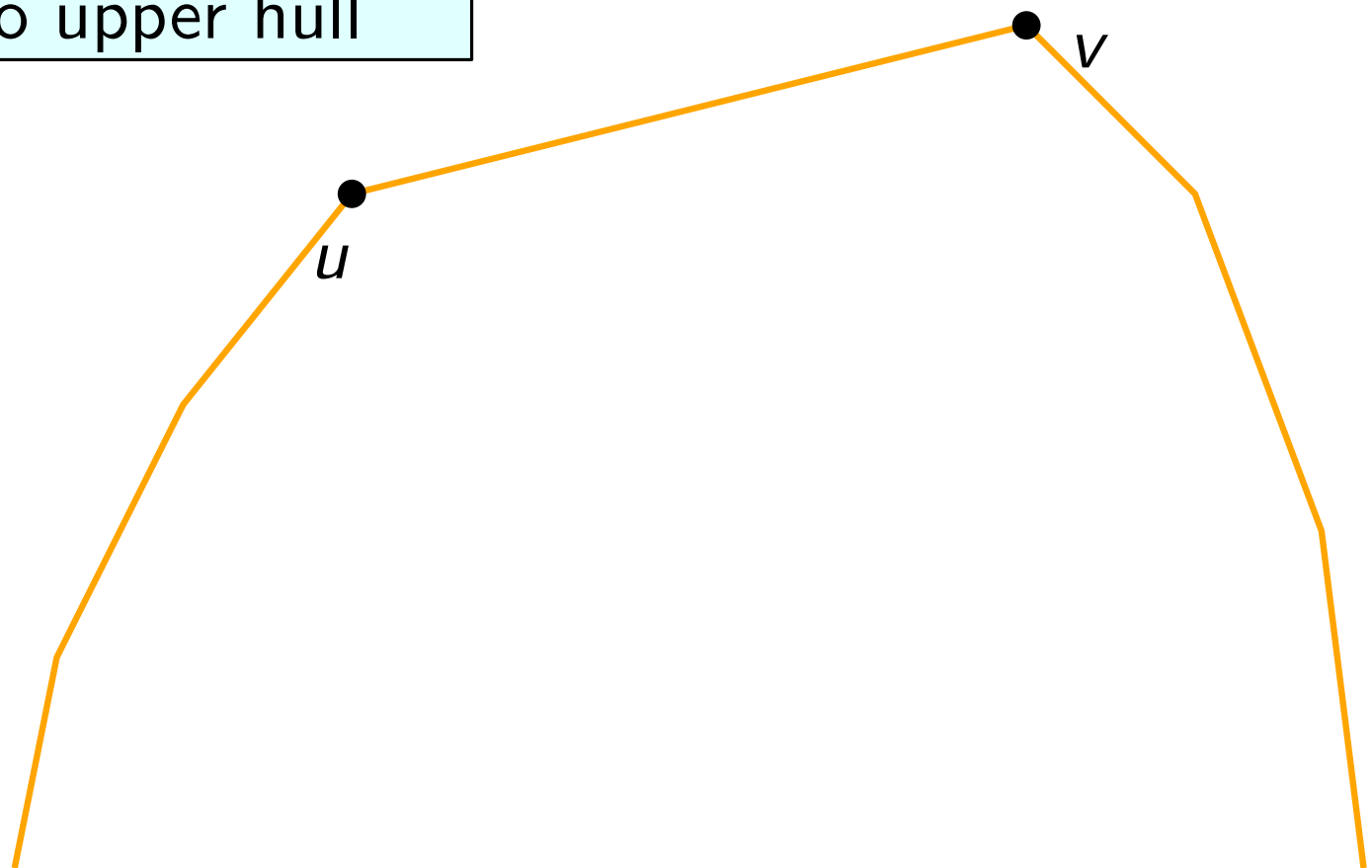
FIFO Convex Hull Complexity

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



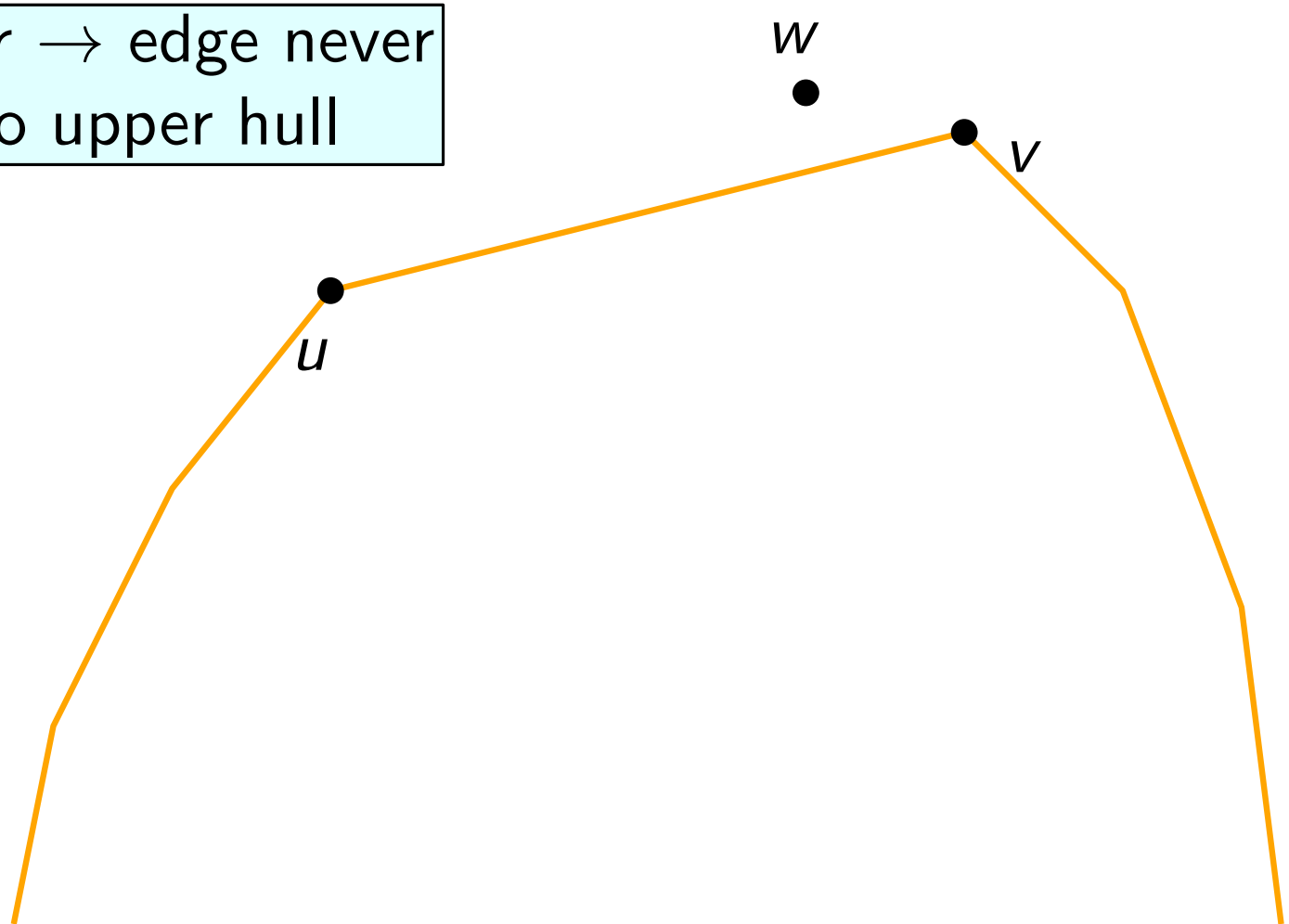
FIFO Convex Hull Complexity

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



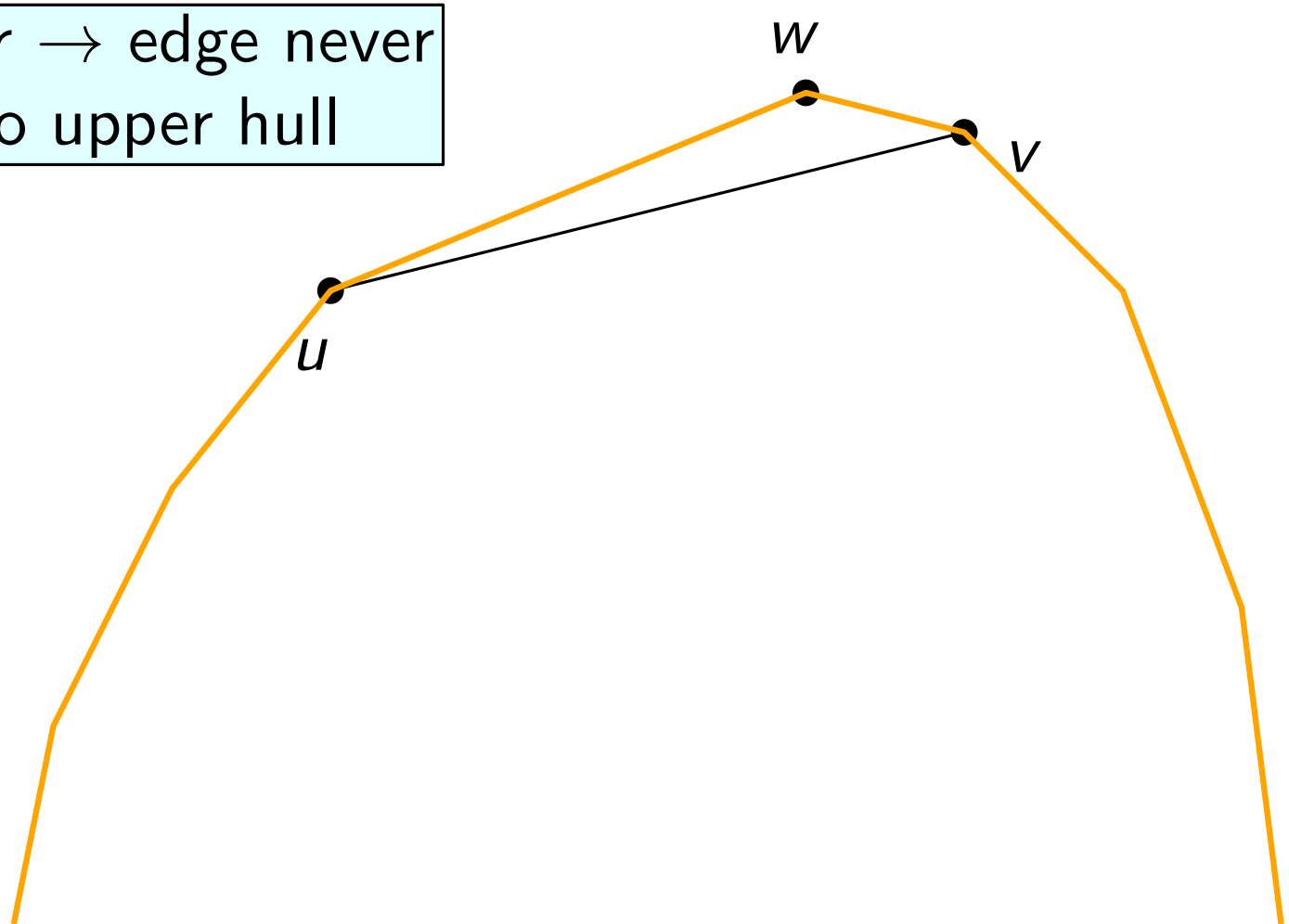
FIFO Convex Hull Complexity

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



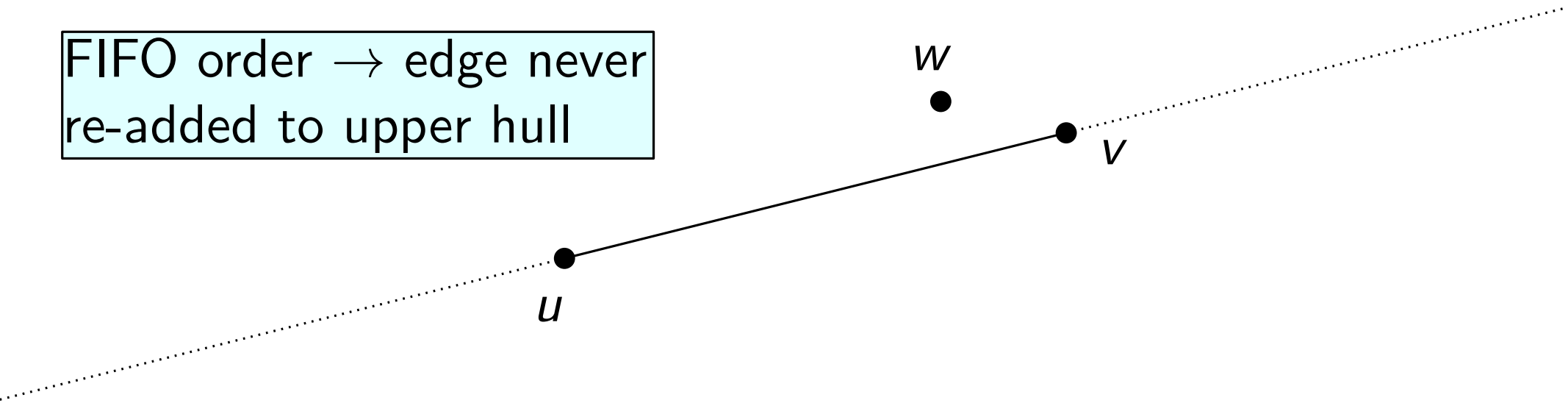
FIFO Convex Hull Complexity

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



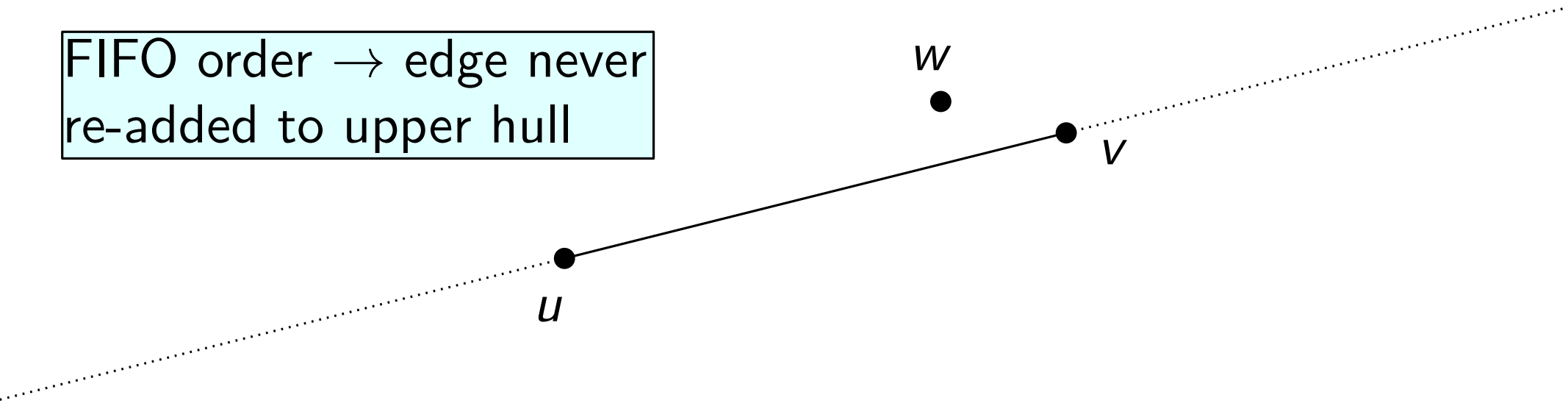
FIFO Convex Hull Complexity

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

2D width decision

FIFO: Conclusion

2D width decision

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

FIFO: Conclusion

2D width 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

2D width 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

2D width 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
  $O(n \log^8 n)$

Open Problem

Result:
The number of structural changes to the upper hull of a set of points in $\mathbb{R}^2$ over n FIFO updates is at most $O(n \log n)$.

Open Problem

Result:
The number of structural changes to the upper hull of a set of points in $\mathbb{R}^2$ over n FIFO updates is at most $O(n \log n)$.

Open Problem:
Can this combinatorial bound be improved? Perhaps to $O(n)$?