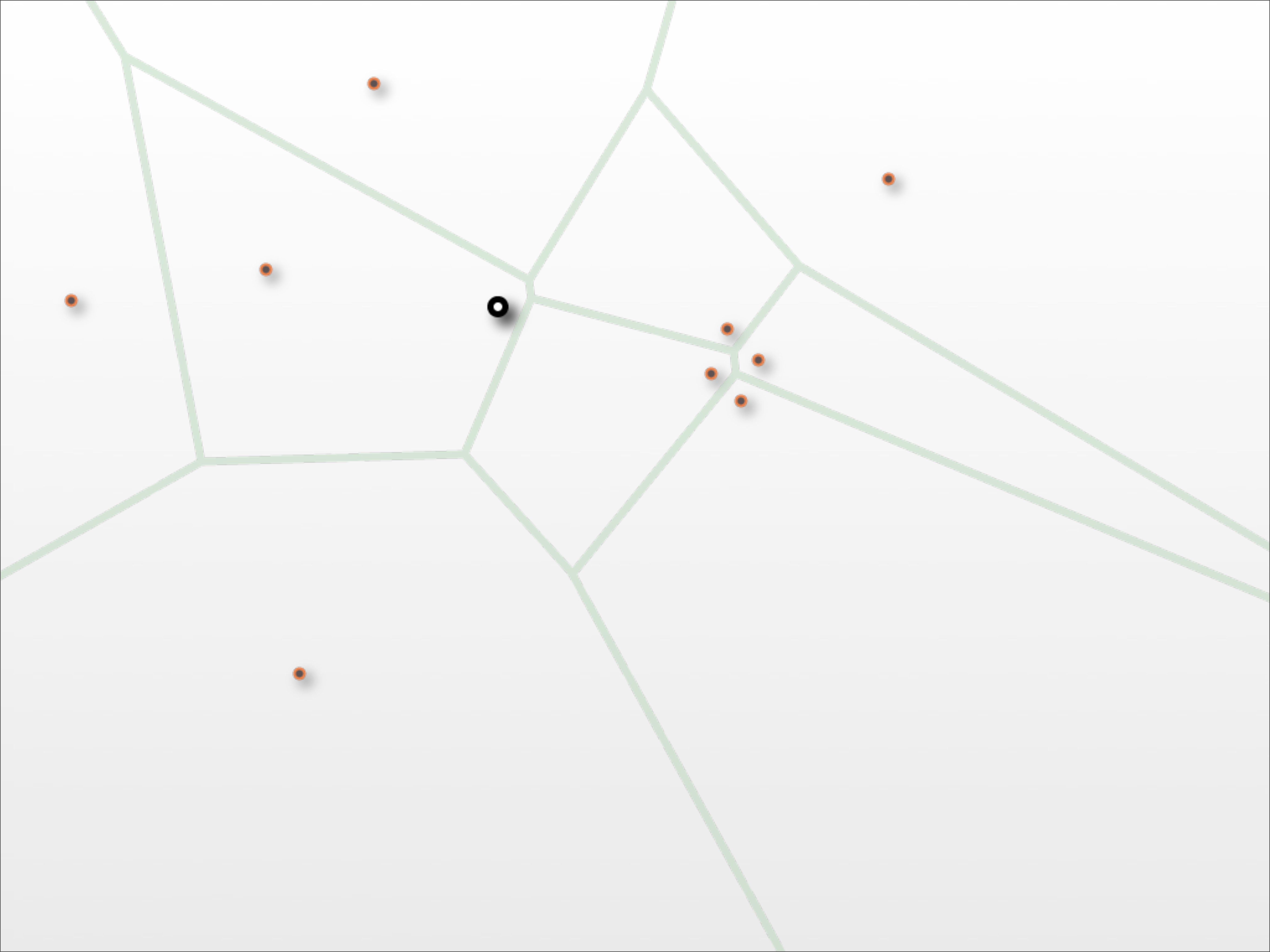
Achieving Spatial Adaptivity in Approximate Nearest Neighbor Search

Jonathan Derryberry, Don Sheehy,
Daniel Sleator, and Maverick Woo



p is a c-approximate nearest neighbor of q if

$d(q, p) < c\, d(q, NN(q))$.

p is a c-approximate nearest neighbor of q if

$d(q, p) < c\, d(q, NN(q))$.

Goals

- $O(\log n)$ queries

p is a c-approximate nearest neighbor of q if

$d(q, p) < c\, d(q, NN(q))$.

Goals

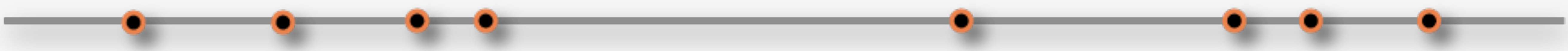
- $O(\log n)$ queries
- $O(d^{3/2})$-approximation

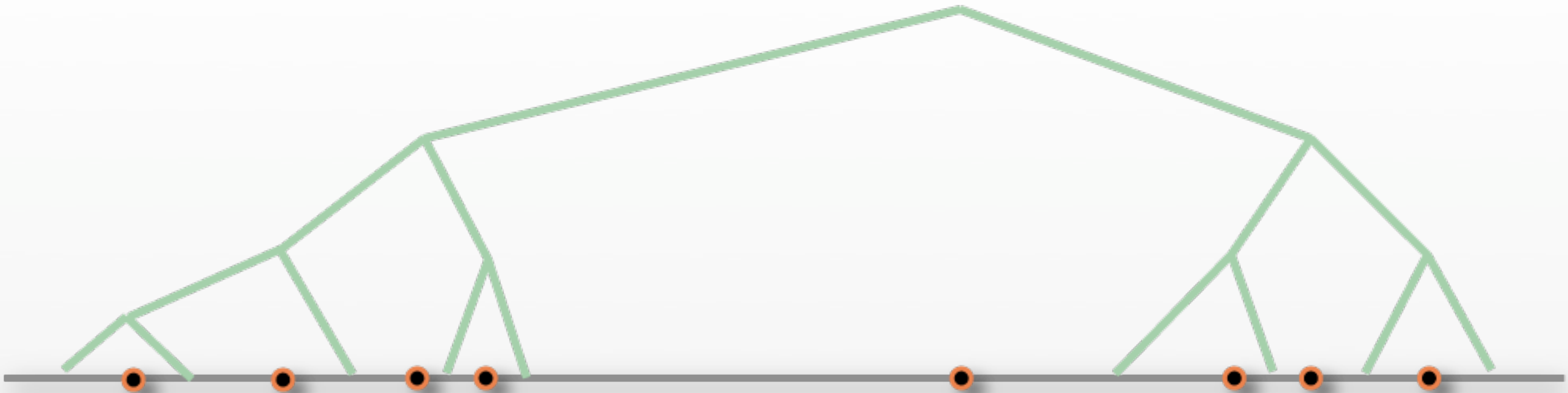
p is a c-approximate nearest neighbor of q if

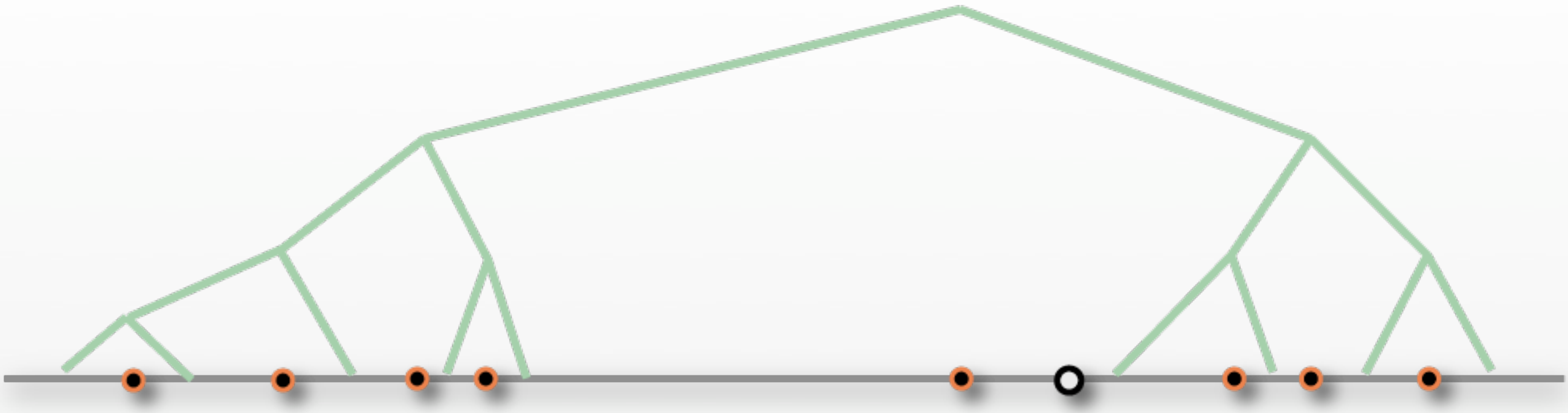
$d(q, p) < c\, d(q, NN(q))$.

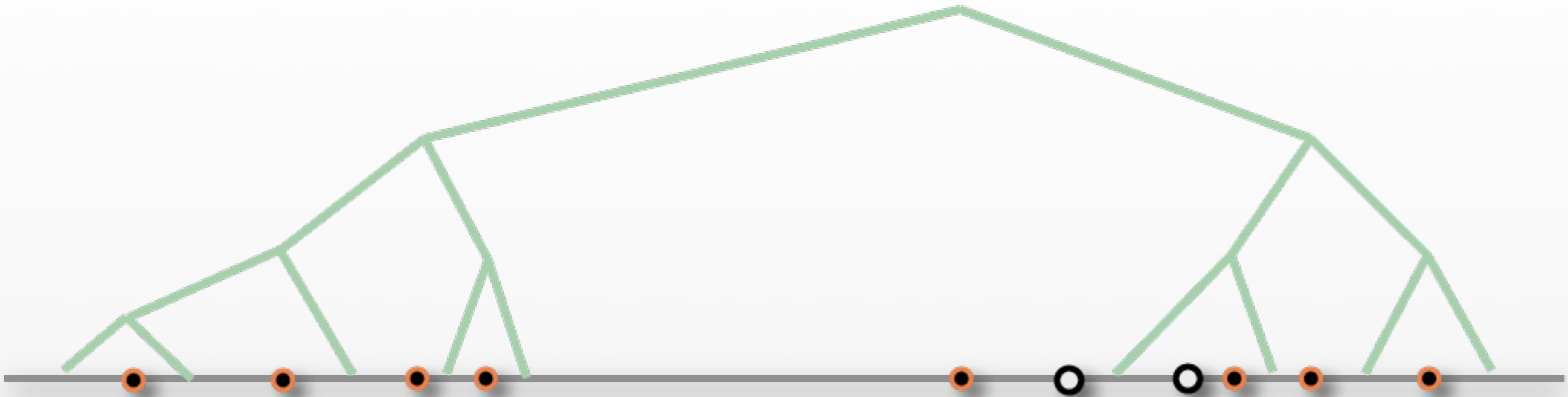
Goals

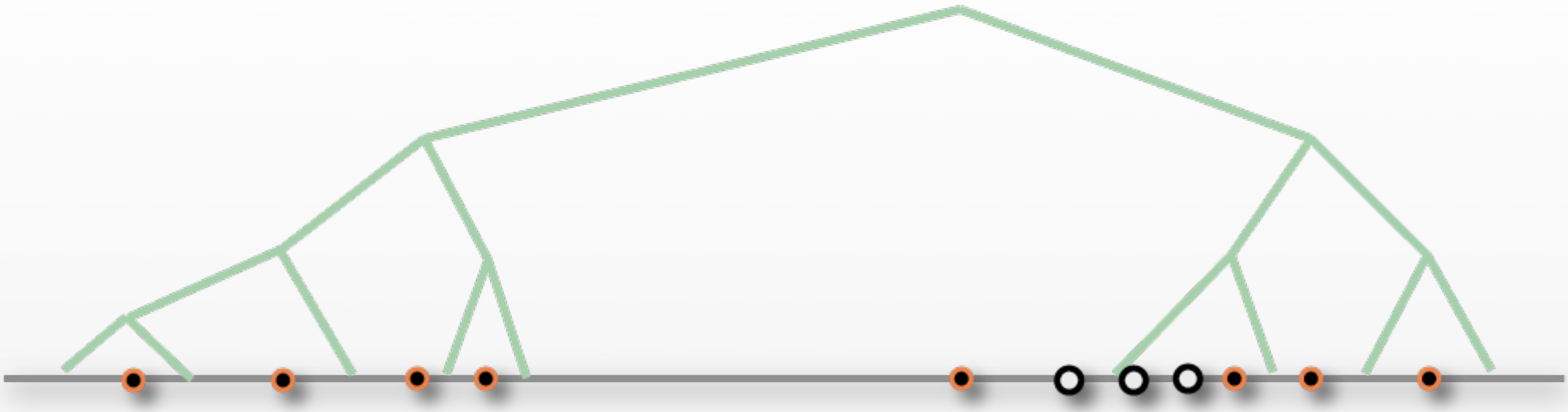
- $O(\log n)$ queries
- $O(d^{3/2})$-approximation
- Spatially Adaptive

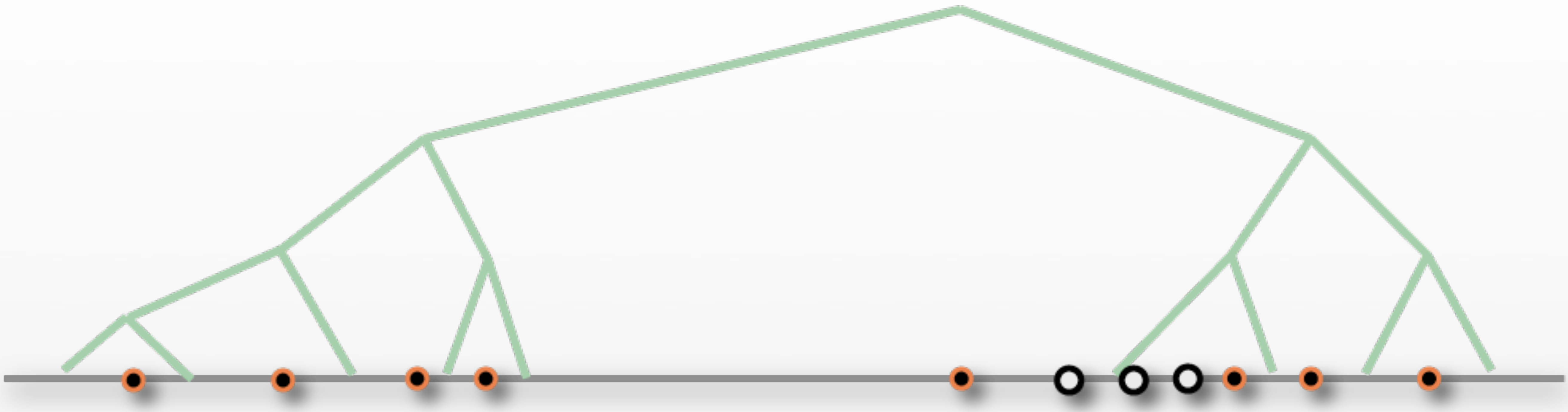








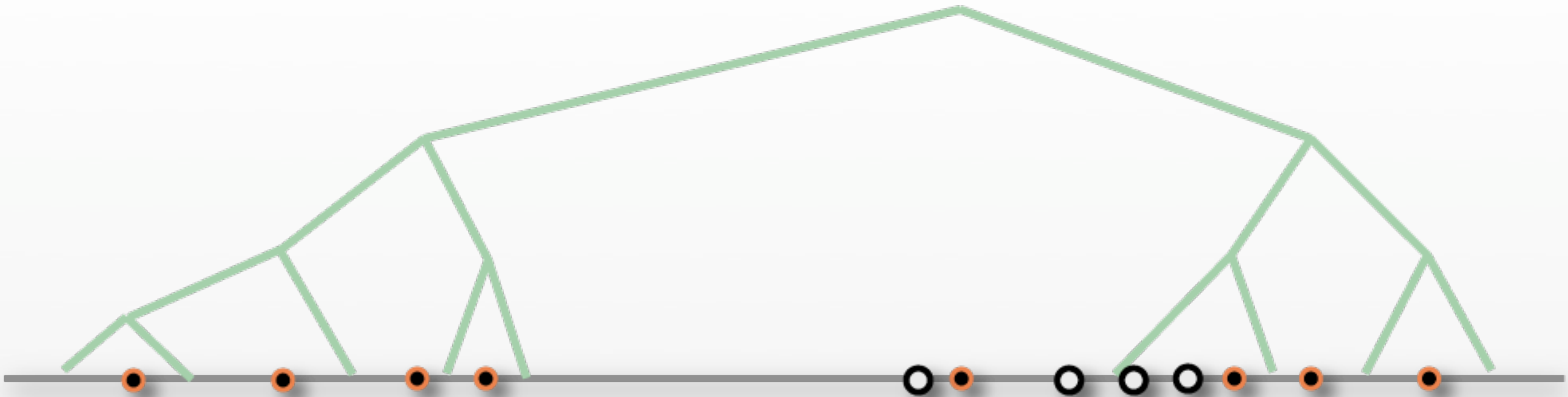


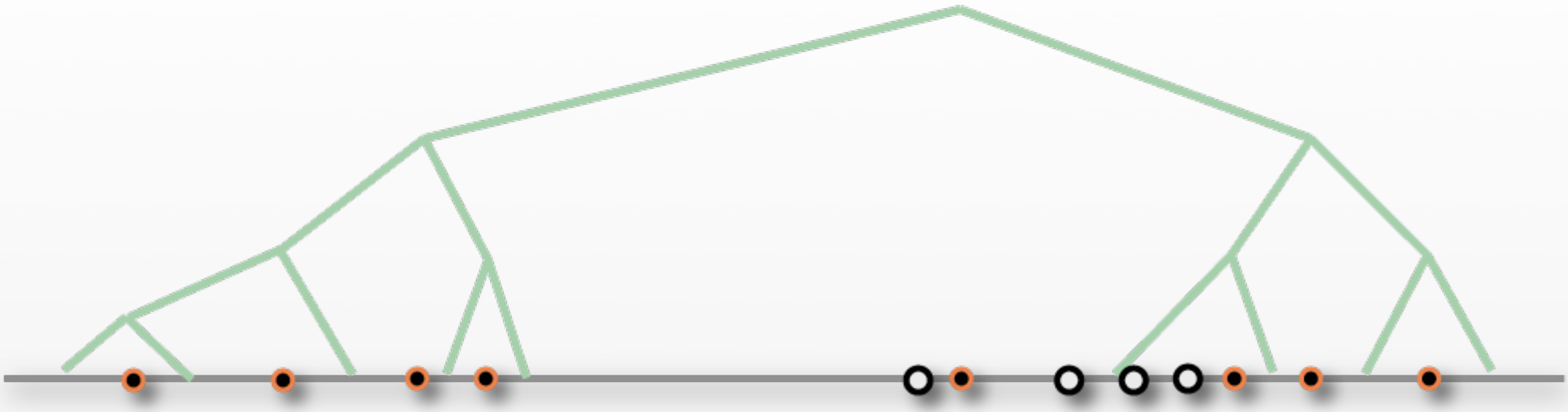


Dynamic Finger Property:

Query time is $O(\log(\delta(p, q)))$

- where p is the previous query result.
- q is the current query.
- $\delta(p,q)$ is the number of input points between p and q.





Finger search pretty well solved in 1D

[pointer machine: Brodal et al],[RAM, Andersson & Thorup]

Skip lists work.

Finger search pretty well solved in 1D
[pointer machine: Brodal et al],[RAM, Andersson & Thorup]

Previous Work in 2D

- Proximate Point Search [Demaine, Iacono, and Langerman, ’02/’04]
- Proximate Point Location [Iacono and Langerman ’03]

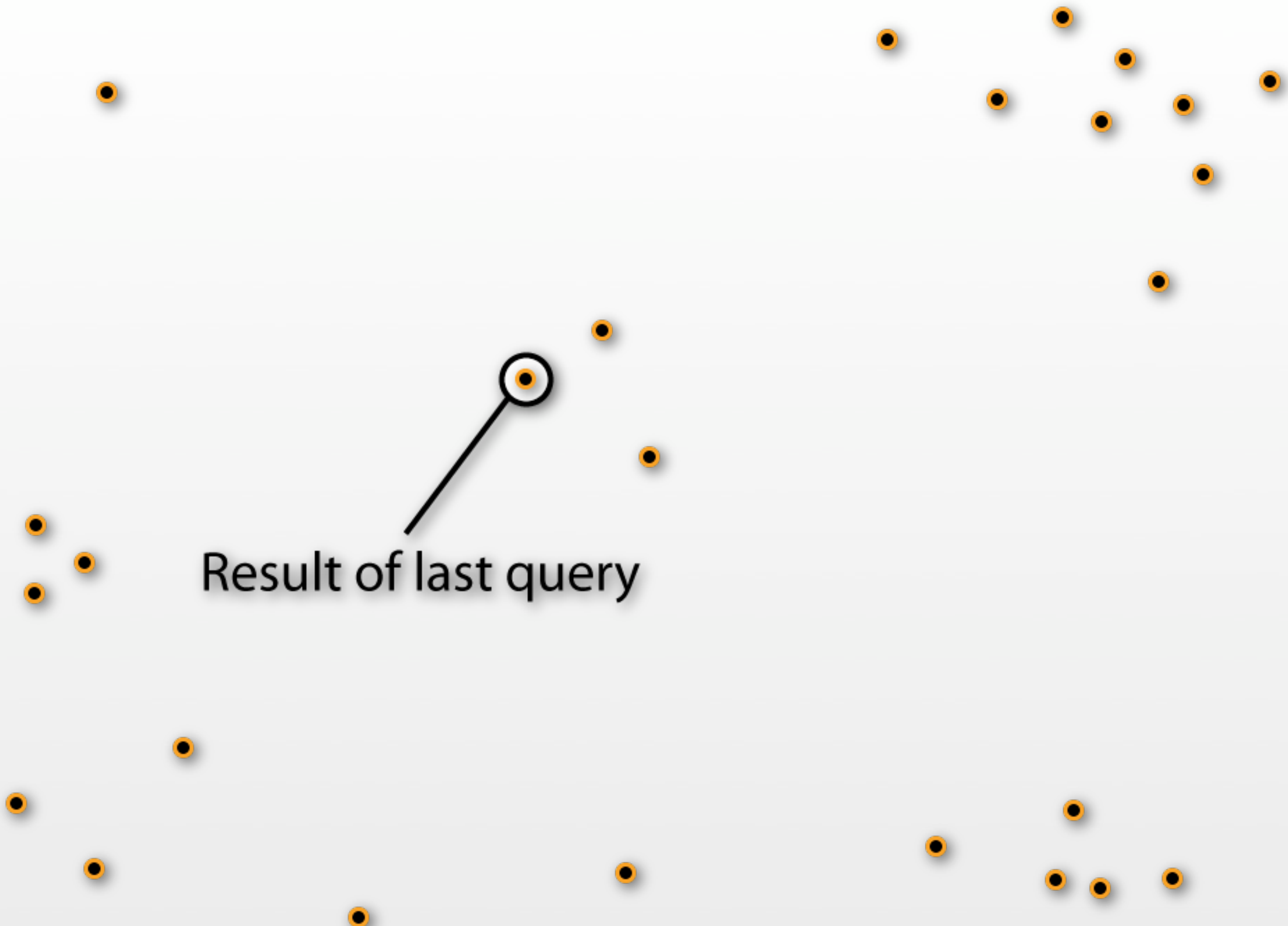
This work.

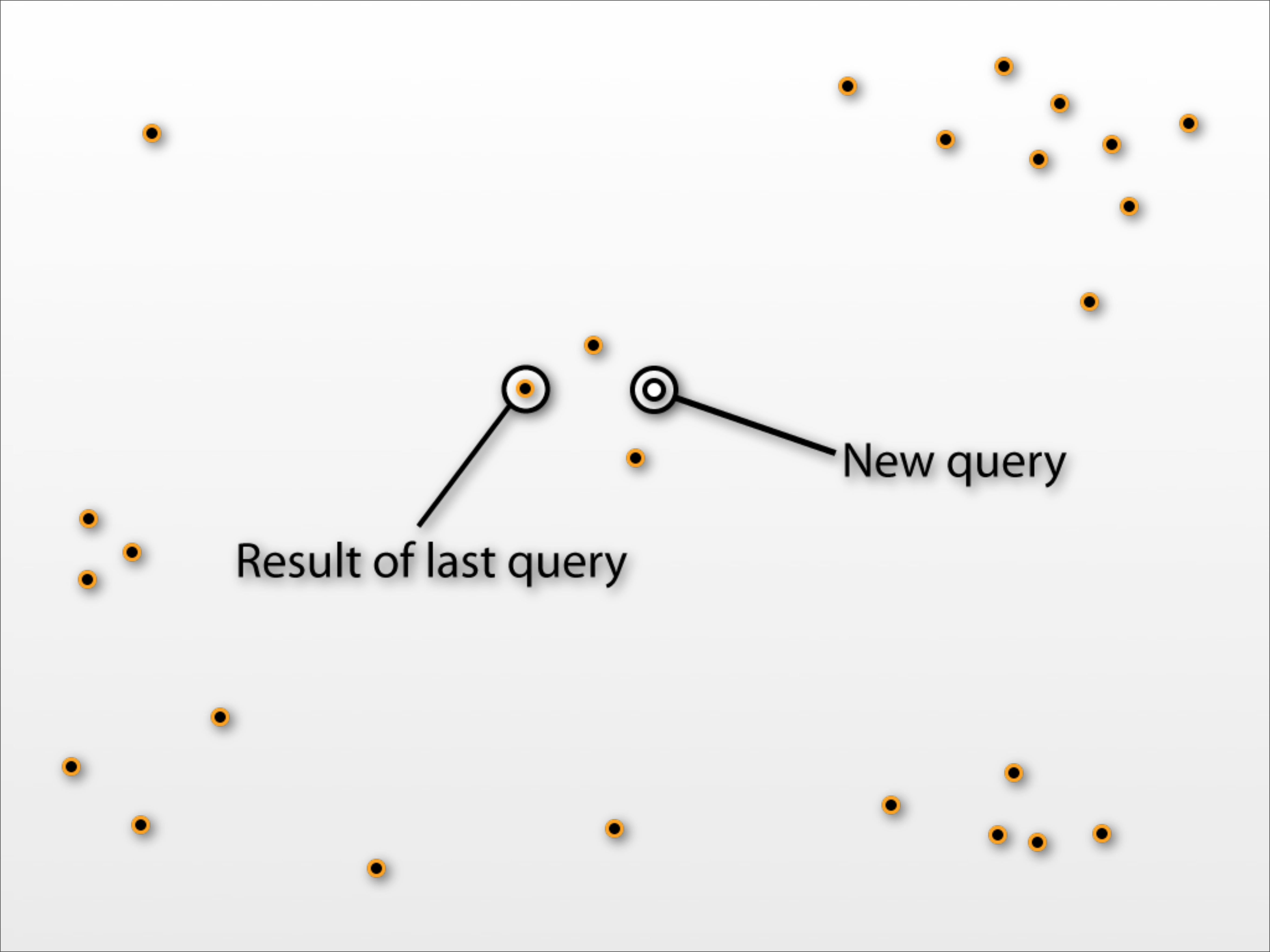
- $O(d^{3/2})$ ANN in R^d
- $O(d^2 \lg(\delta(p,q)))$ queries. (Finger Search!)
- $O(dn)$ space.
- $O(d^2 n \lg n)$ preprocessing time

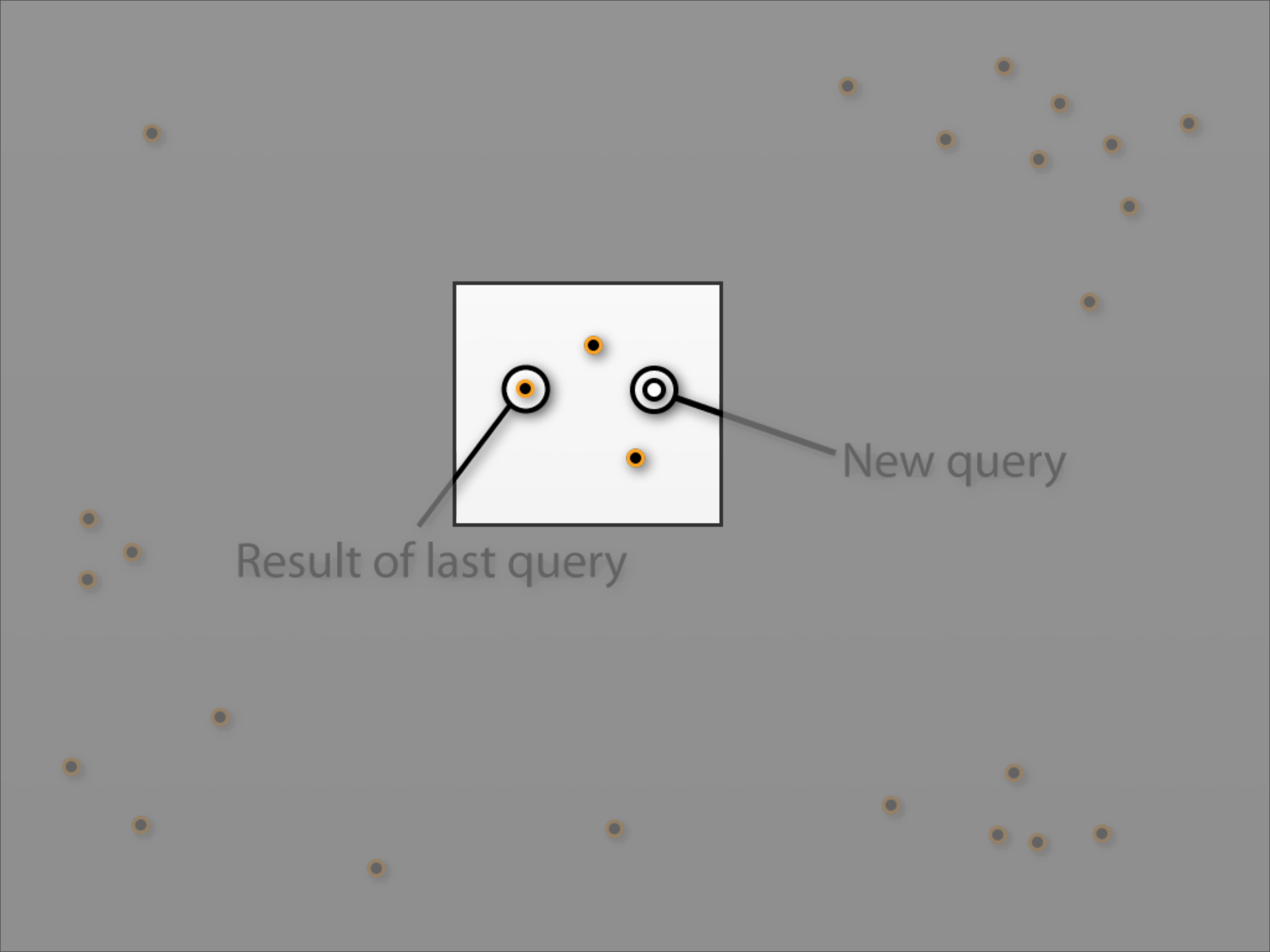
This work.

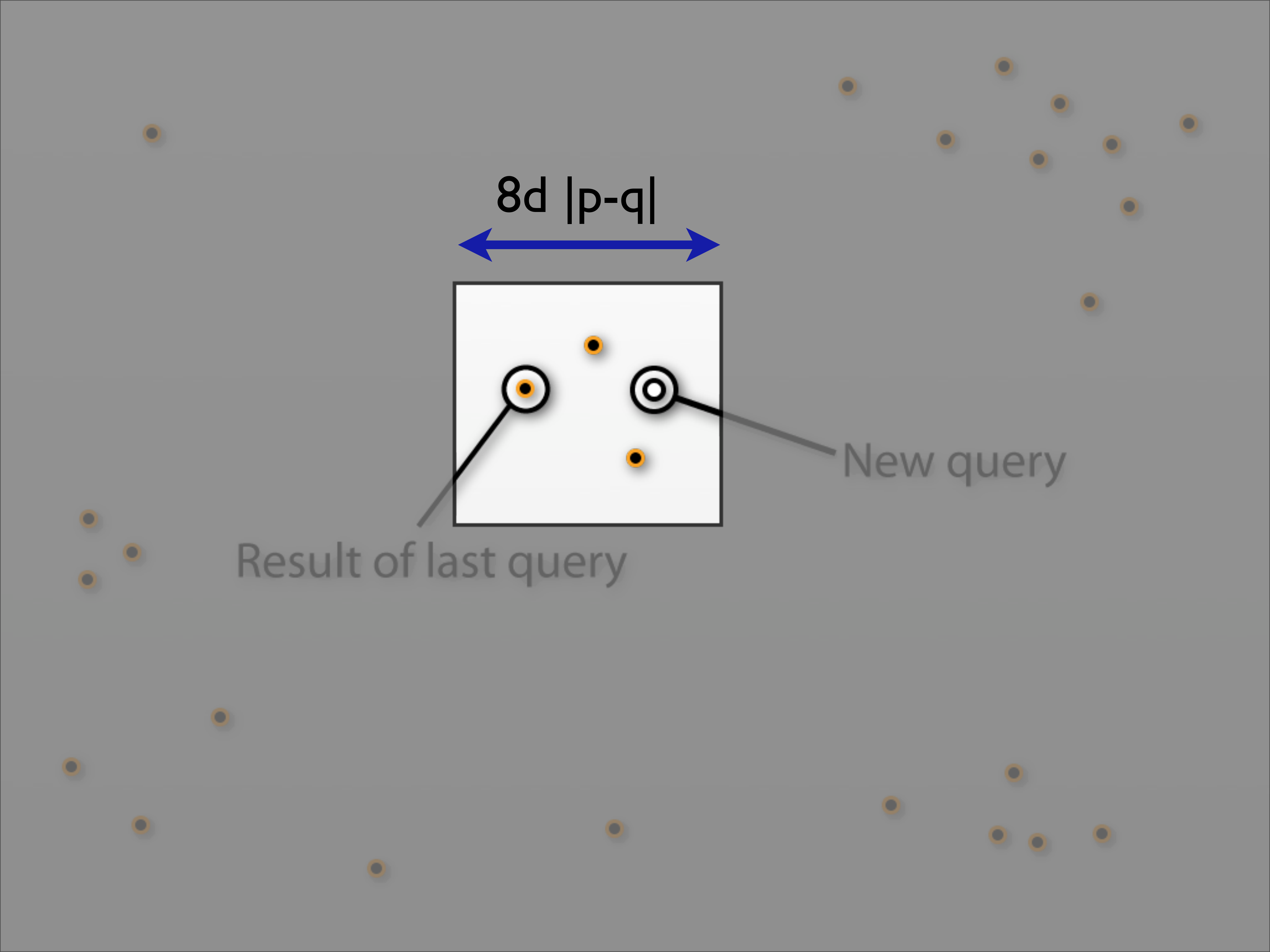
- $O(d^{3/2})$ ANN in R^d
- $O(d^2 \lg(\delta(p,q)))$ queries. (Finger Search!)
- $O(dn)$ space.
- $O(d^2 n \lg n)$ preprocessing time

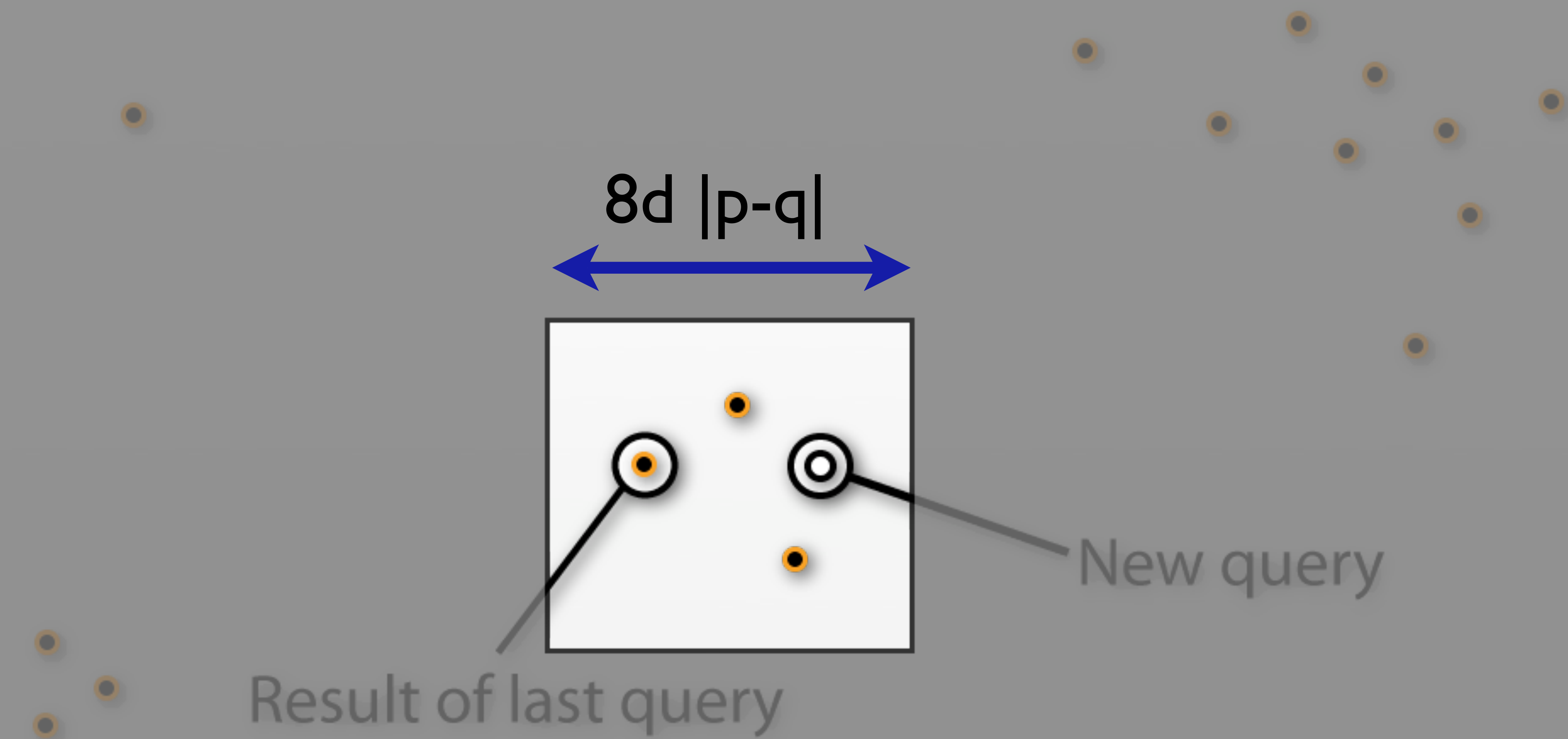
?


Result of last query


New query
Result of last query


New query
Result of last query


8d |p-q|
New query
Result of last query



$\delta(p, q)$ = # of input points in this box.

(11111, 00000)

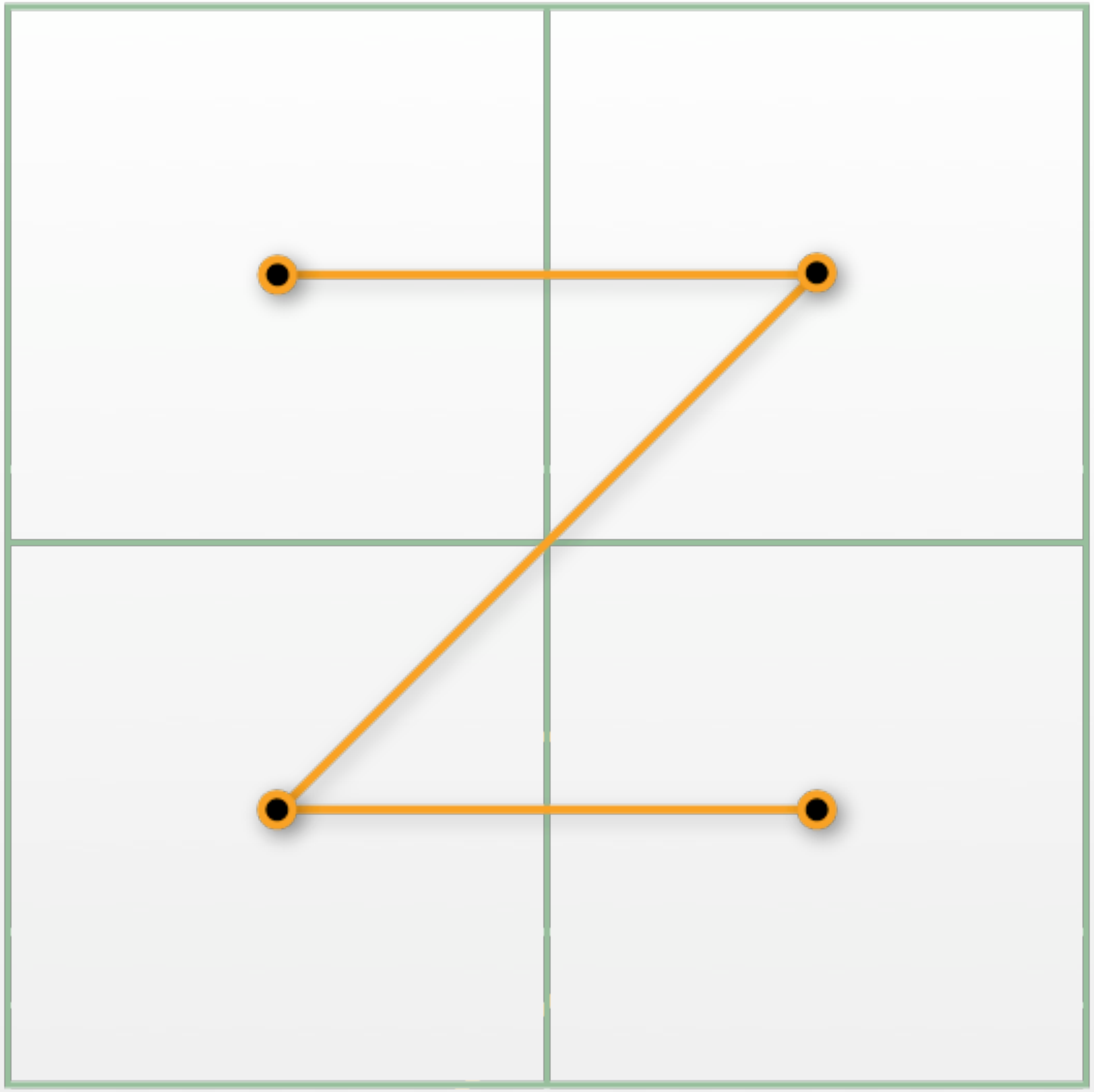
$\downarrow$

1010101010

(11111, 00000)

↓

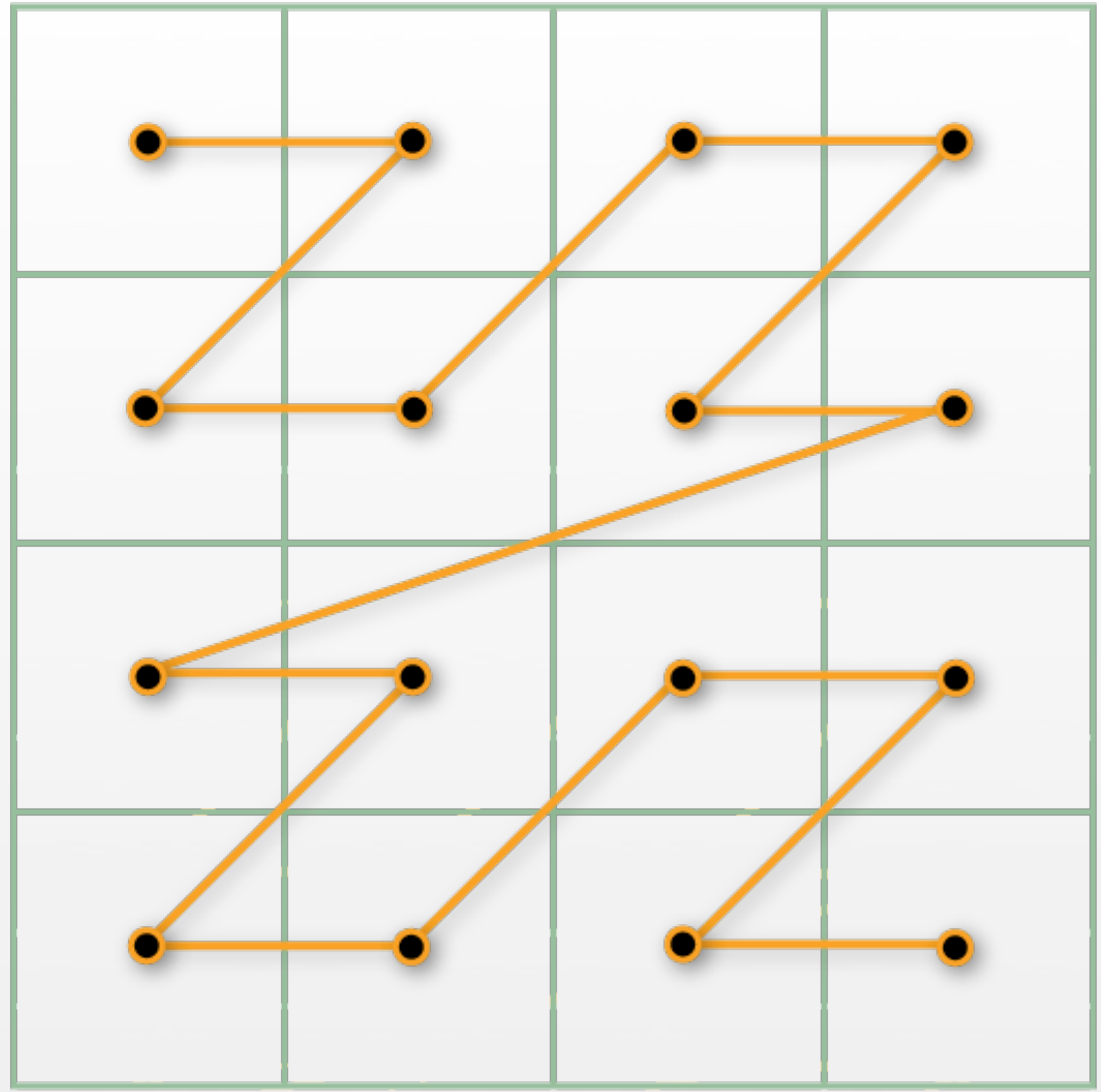
1010101010



(11111, 00000)

↓

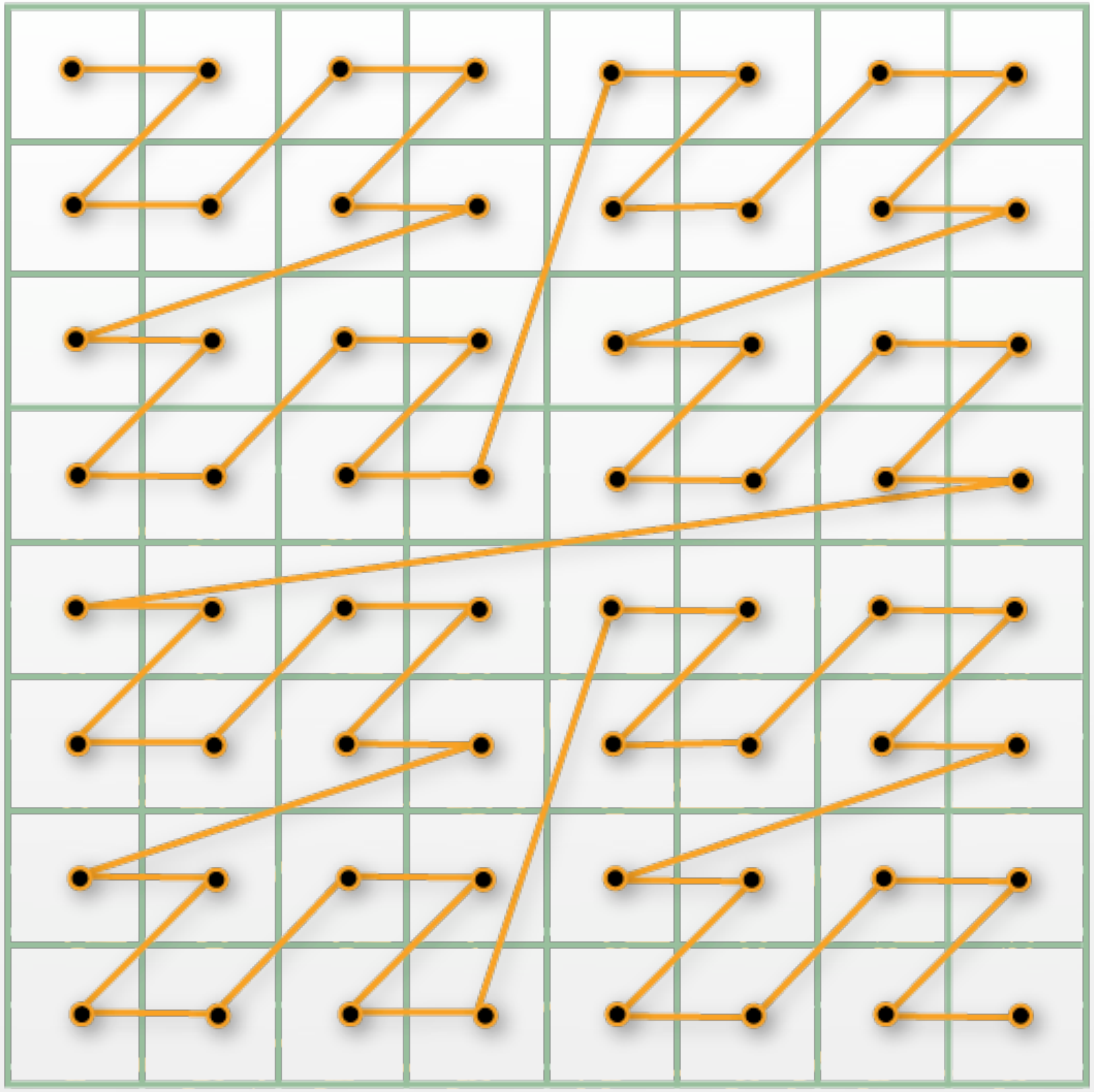
1010101010



(11111, 00000)

↓

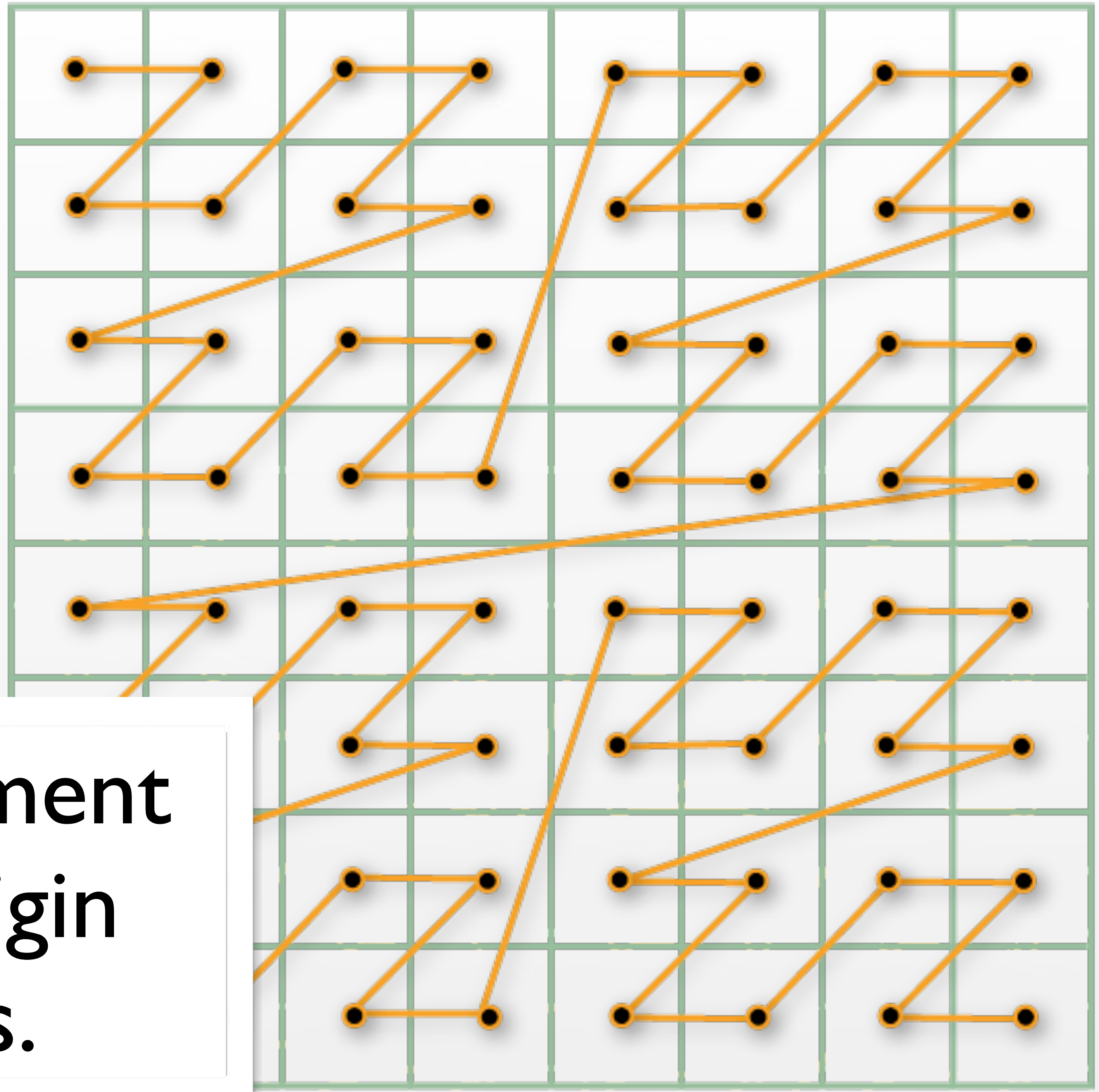
1010101010



(11111, 00000)

↓

1010101010



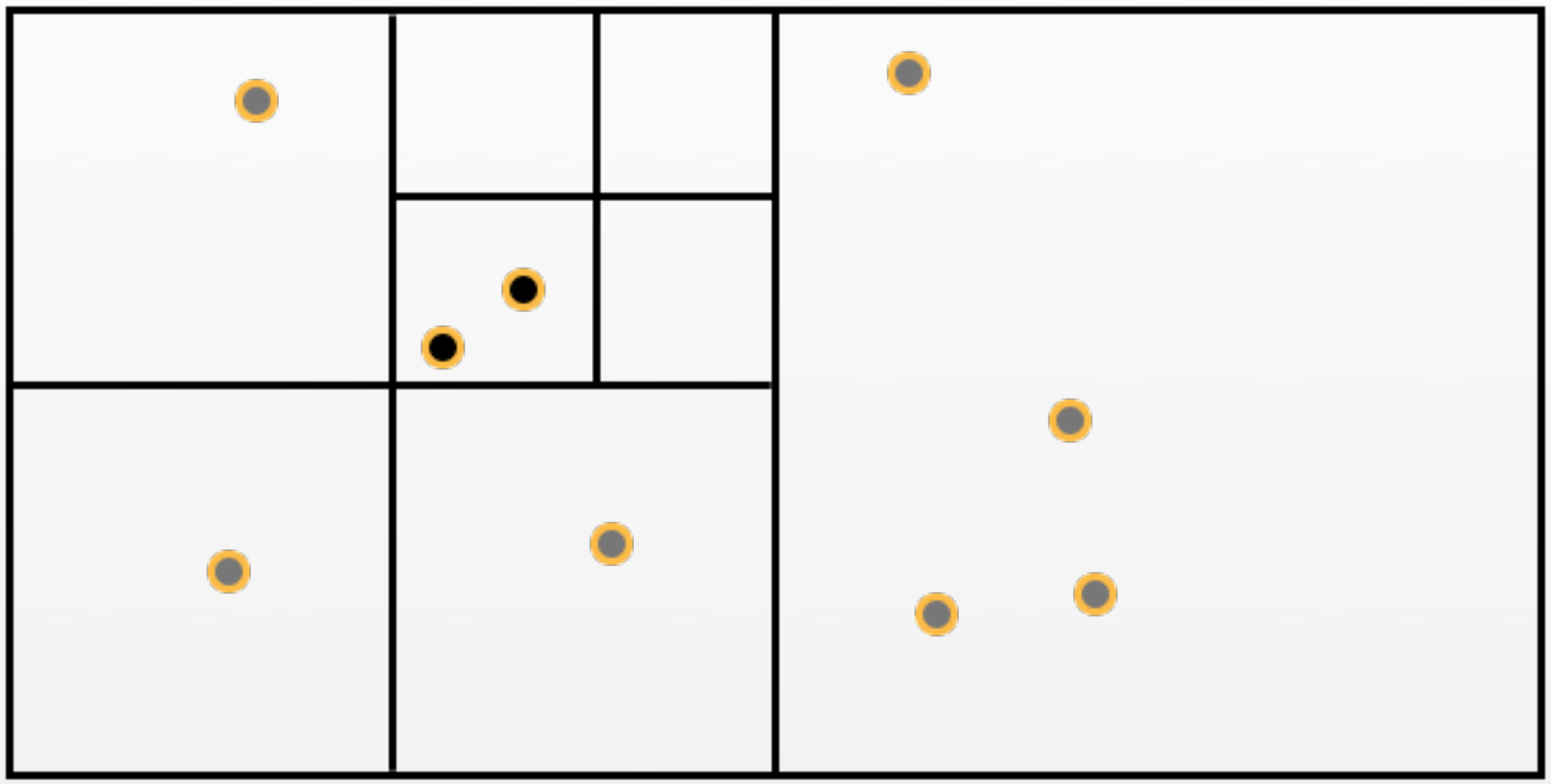
The placement of the origin matters.

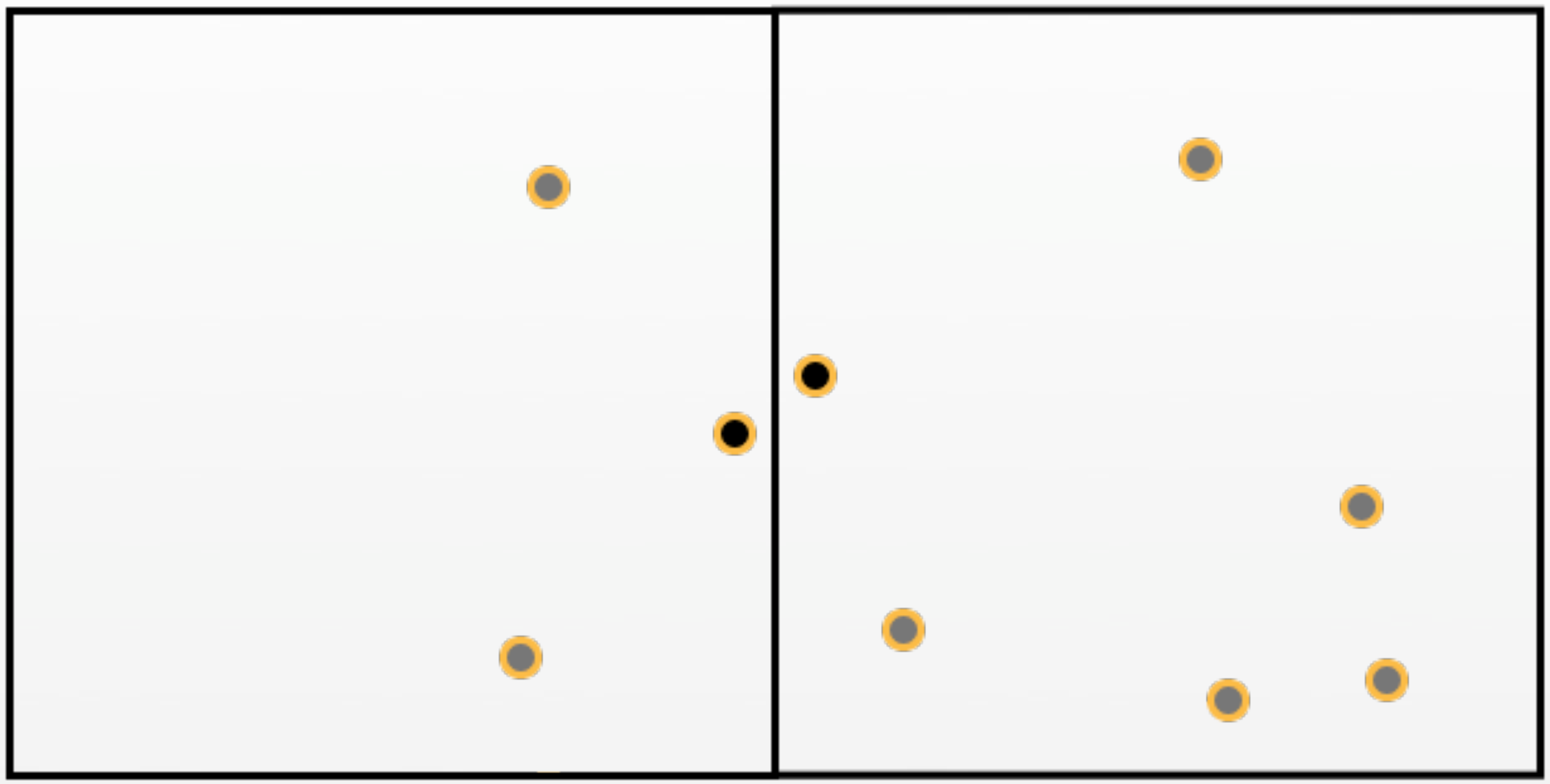
A classic $O(d^{3/2})$-ANN Algorithm:
[Bern93, Chan98, Liao01 et al]

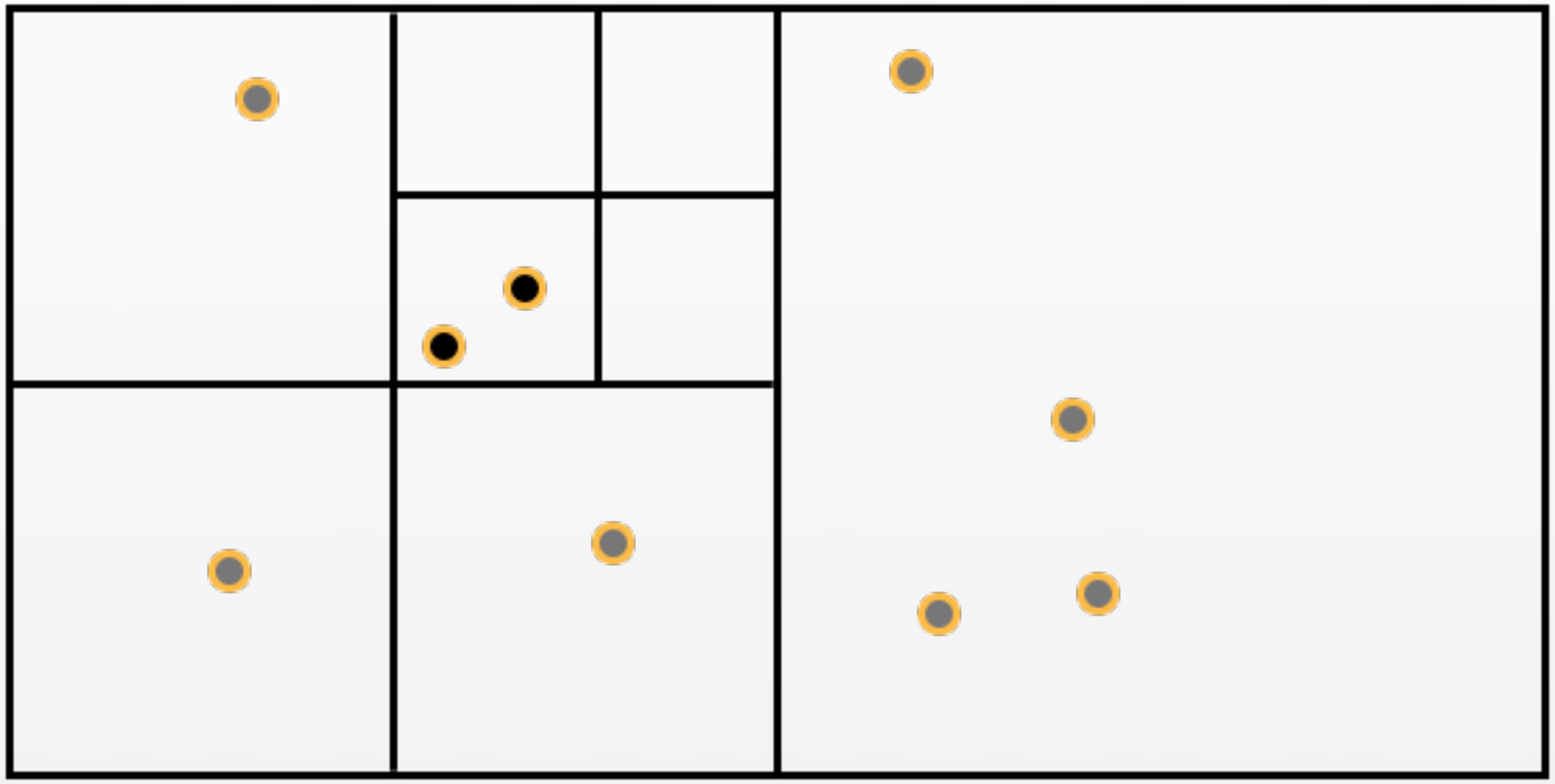
Put the input points in your favorite 1-dimensional data structure, ordered by the Z-order.

Construct d+1 of these data structures, such that all points inserted into the i^{th} are shifted by (i/d+1) x diameter in every coordinate.

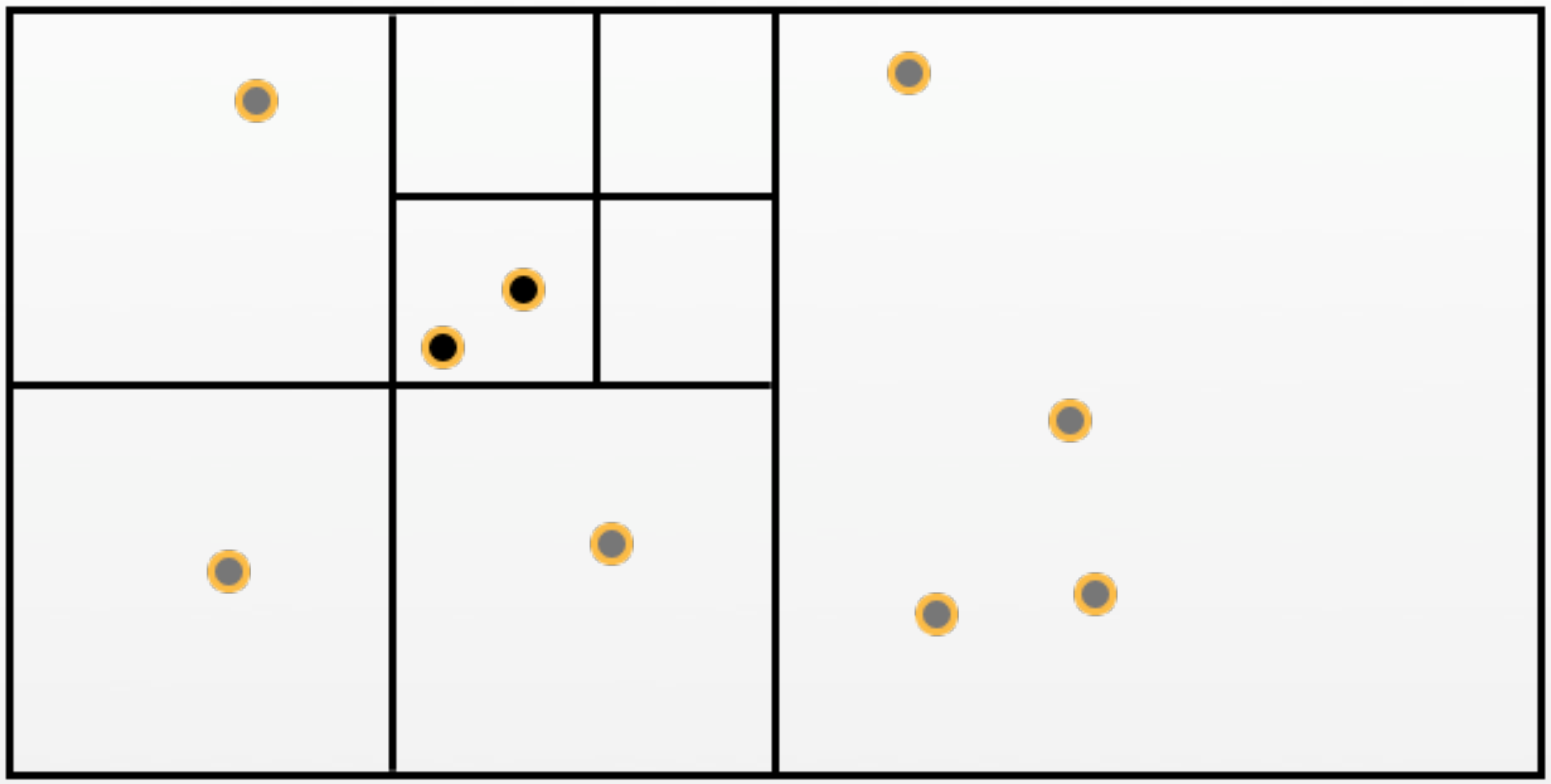
For each query, search all d+1 data structures for a nearest neighbor in the shifted Z-order. Return the one that is closest in R^d.







Lemma [Chan]: With at least d+1 shifts, at most 1 is “bad” for each dimension.



Lemma [Chan]: With at least d+1 shifts, at most 1 is “bad” for each dimension.

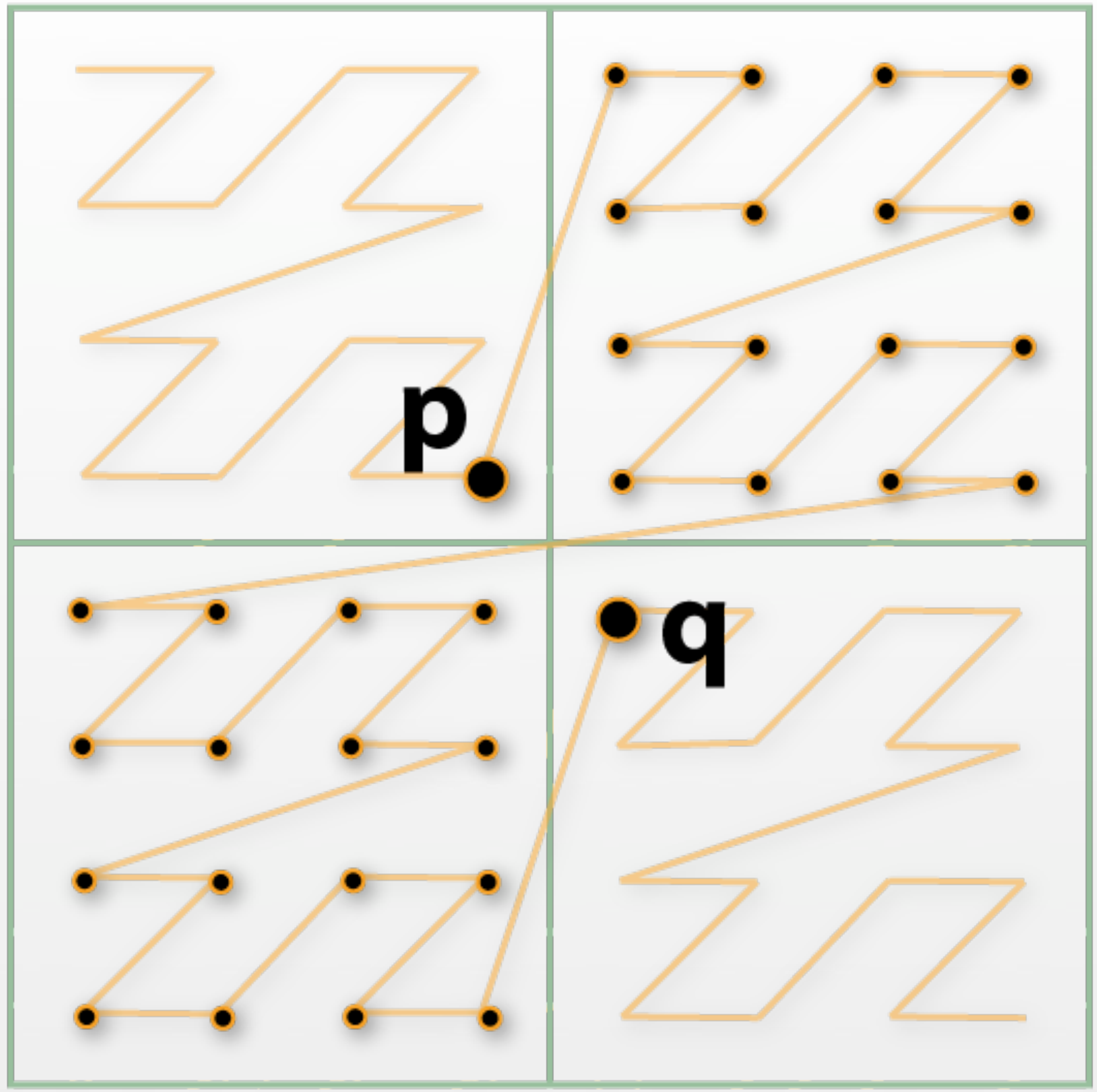
So x and NN_x share a small QT cell.

Are we done?

Z-order reduces d-dimensions to 1-dimension.

It works for ANN.

Finger Search in 1-dimension is solved.


p
q

Making it work.

1. Use 2d+1 1D data structures that support finger search.

2. Run all 2d+1 searches in parallel.

3. Stop when d+1 are finished.

4. Return the best answer among the d+1 that finished.

5. Manually update the finger pointers in all 2d+1 structures.

If q and NN_q are in a small QT box relative to their distance then the approximation is good.

If q and NN_q are in a small QT box relative to their distance then the approximation is good.

If p and q are in a small QT box relative to their distance then the runtime is good.

If q and NN_q are in a small QT box relative to their distance then the approximation is good.

If p and q are in a small QT box relative to their distance then the runtime is good.

We apply the Chan Lemma twice.

- Since at most d are bad for p and q, we can stop after all but d finish and the runtime is good.
- Since d+1 are left over, at least one will be a good approximation.

Summary

A new data structure.

- $O(d^{3/2})$ ANN in R^d
- $O(d^2 \lg(\delta(p,q)))$ queries. (Finger Search!)
- $O(dn)$ space.
- $O(d^2 n \lg n)$ preprocessing time

Thank you.

Thank you.

Questions?