Faster DBSCAN and HDBSCAN in Low-Dimensional Euclidean Spaces

Paper: Mark de Berg, Ade Gunawan, Marcel Roeloffzen

Slides: Thomas van Dijk

Noon seminar: 15th Feb 2018

Julius-Maximilians-UNIVERSITÄT WÜRZBURG

Lehrstuhl für INFORMATIK I Effiziente Algorithmen und wissensbasierte Systeme

Institut für Informatik

Clustering

Clustering is classically the problem of finding a partition of a data set such that elements in the same cell (“cluster”) are near each other according to a given distance criterion, while elements in different sets are distant.

Clustering

Clustering is classically the problem of finding a partition of a data set such that elements in the same cell ("cluster") are near each other according to a given distance criterion, while elements in different sets are distant.

Fundamental problem in data mining, but not uniquely defined.

What are you clustering? What are you trying to do with the data?

Clustering

Clustering is classically the problem of finding a partition of a data set such that elements in the same cell ("cluster") are near each other according to a given distance criterion, while elements in different sets are distant.

Fundamental problem in data mining, but not uniquely defined.

What are you clustering? What are you trying to do with the data?

Distance: Euclidean? Metric?

Clustering

Clustering is classically the problem of finding a partition of a data set such that elements in the same cell ("cluster") are near each other according to a given distance criterion, while elements in different sets are distant.

Fundamental problem in data mining, but not uniquely defined.

What are you clustering? What are you trying to do with the data?

Distance: Euclidean? Metric?

How many clusters? What can clusters look like?

Clustering

1956

MATHÉMATIQUE

Sur la division des corps matériels en parties

par

H. STEINHAUS

Présenté le 19 Octobre 1956

Un corps Q est, par définition, une répartition de matière dans l'espace, donnée par une fonction $f(P)$; on appelle cette fonction la *densité* du corps en question; elle est définie pour tous les points P de l'espace; elle est non-négative et mesurable. On suppose que *l'ensemble caracté-*

Clustering

1967

SOME METHODS FOR CLASSIFICATION AND ANALYSIS OF MULTIVARIATE OBSERVATIONS

J. MacQUEEN
University of California, Los Angeles

1. Introduction

The main purpose of this paper is to describe a process for partitioning an N-dimensional population into k sets on the basis of a sample. The process, which is called 'k-means,' appears to give partitions which are reasonably efficient in the sense of within-class variance. That is, if p is the probability mass function for the population, $S = \{S_1, S_2, \cdots, S_k\}$ is a partition of E_N, and u_i,

Clustering

DBSCAN

1996

A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu

Institute for Computer Science, University of Munich
Oettingenstr. 67, D-80538 München, Germany
{ester | kriegel | sander | xwxu}@informatik.uni-muenchen.de

Clustering

DBSCAN

1996

A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise

?

?

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu

Institute for Computer Science, University of Munich
Oettingenstr. 67, D-80538 München, Germany
{ester | kriegel | sander | xwxu}@informatik.uni-muenchen.de

$\geqslant 8 \times 10^3$ citations

KDD "test of time award" 2014

Open source implementations available in many languages

Clustering

DBSCAN

1996

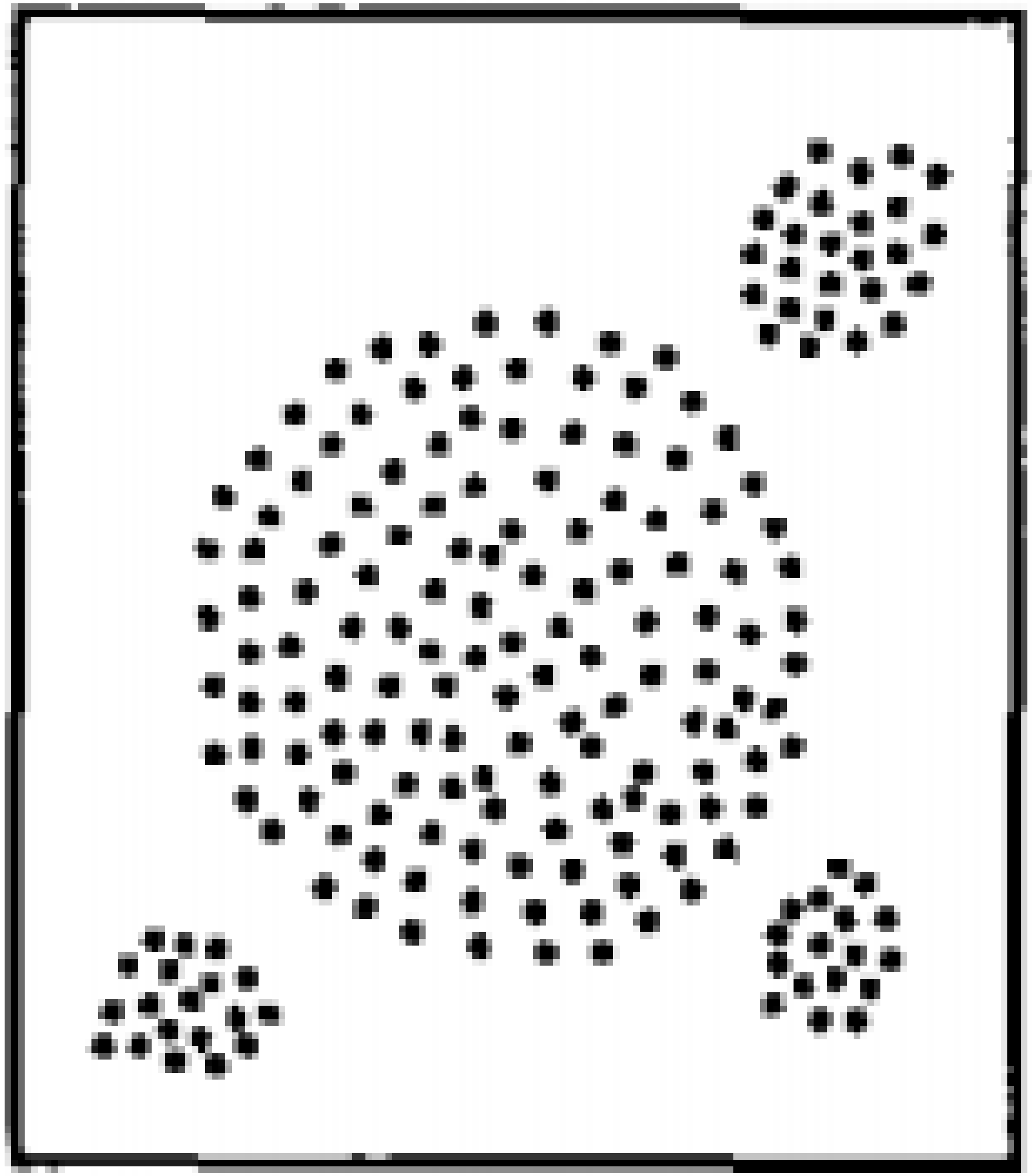
A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu

Institute for Computer Science, University of Munich
Oettingenstr. 67, D-80538 München, Germany
{ester | kriegel | sander | xwxu}@informatik.uni-muenchen.de

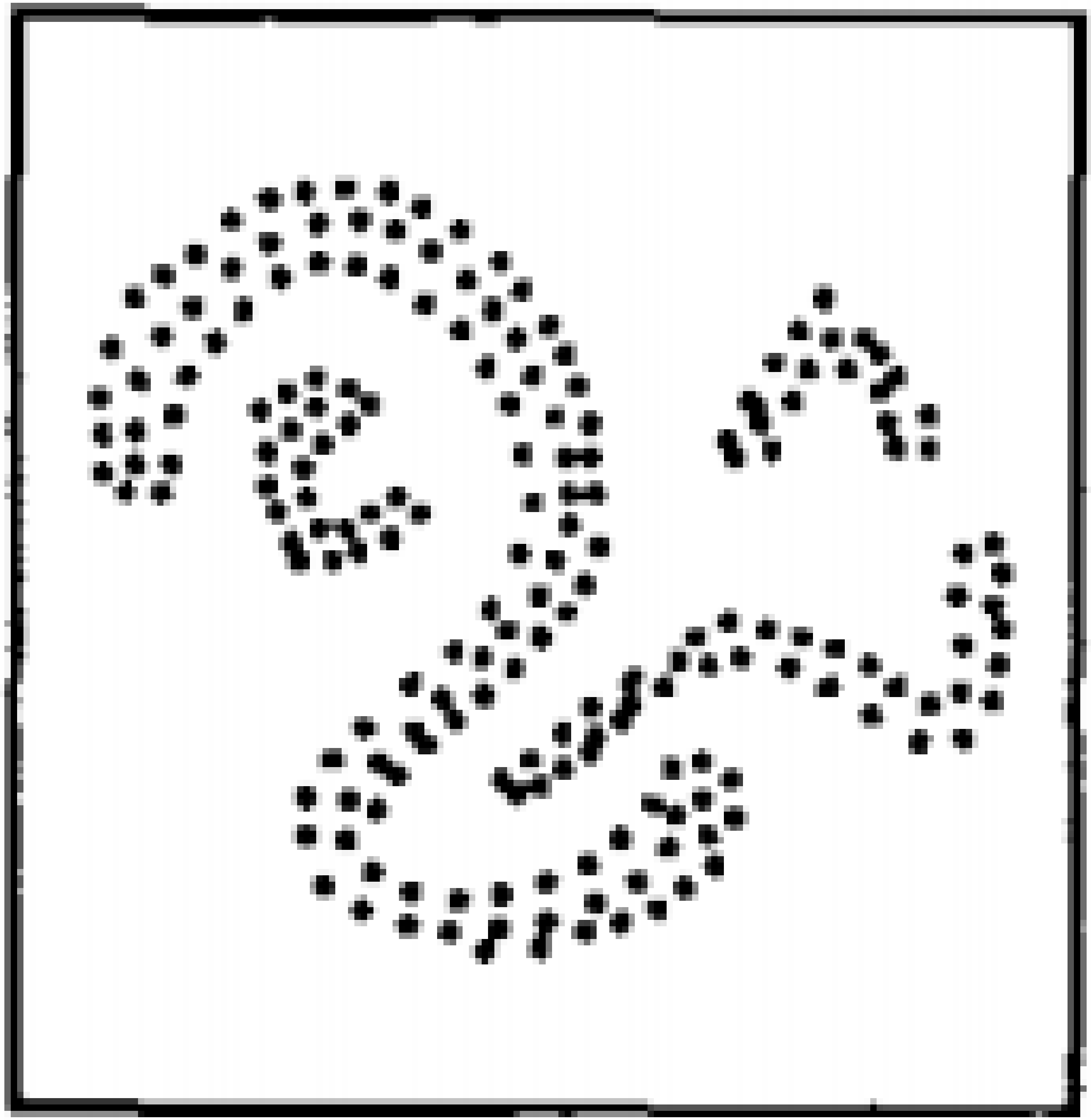
ing an appropriate value for it. It discovers clusters of arbitrary shape. Finally, DBSCAN is efficient even for large spatial databases. The rest of the paper is organized as follows.

Clustering



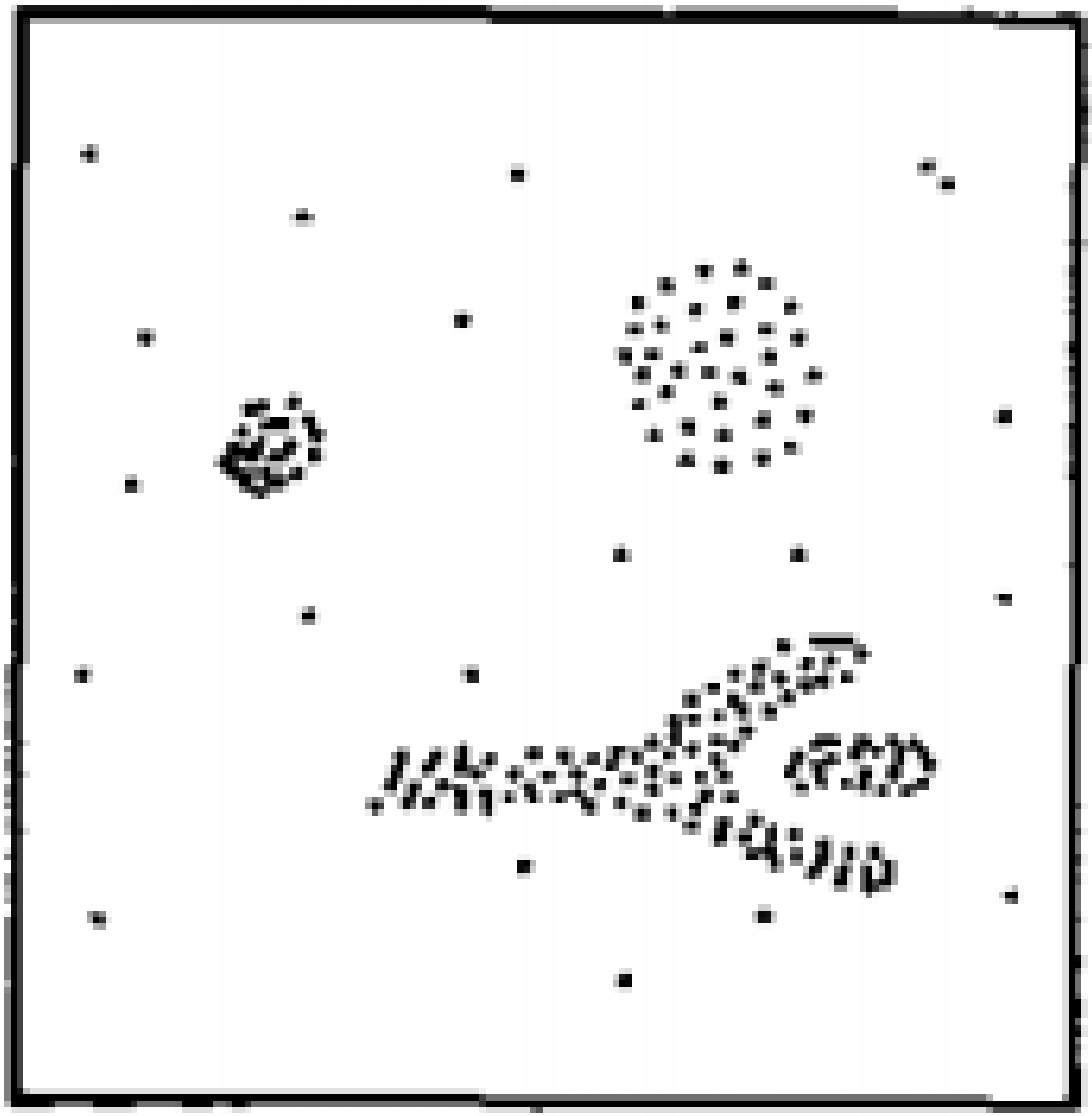
database 1

Clustering



database 2

Clustering



database 3

DBSCAN: Objectives

1. “Minimal requirements of domain knowledge to determine the input parameters, because appropriate values are often not known in advance when dealing with large databases.”

DBSCAN: Objectives

1. “Minimal requirements of domain knowledge to determine the input parameters, because appropriate values are often not known in advance when dealing with large databases.”

2. “Discovery of clusters with arbitrary shape, because the shape of clusters in spatial databases may be spherical, drawn-out, linear, elongated etc.”

DBSCAN: Objectives

1. “Minimal requirements of domain knowledge to determine the input parameters, because appropriate values are often not known in advance when dealing with large databases.”

2. “Discovery of clusters with arbitrary shape, because the shape of clusters in spatial databases may be spherical, drawn-out, linear, elongated etc.”

3. “Good efficiency on large databases, i.e. on databases of significantly more than just a few thousand objects.”

DBSCAN

Given: data points X, distance function $d(\cdot, \cdot)$, thresholds ε and k.

DBSCAN

"Eps" "MinPts"

Given: data points X, distance function $d(\cdot, \cdot)$, thresholds ε and k.

DBSCAN

Given: data points X, distance function $d(\cdot, \cdot)$, thresholds ε and k.

Def. The **ε-neighborhood** of a point $p \in X$ is

$$N_\varepsilon(p) = \{ q \in X \mid d(p, q) \leqslant \varepsilon \}.$$

DBSCAN

Given: data points X, distance function $d(\cdot, \cdot)$, thresholds ε and k.

Def. The **ε-neighborhood** of a point $p \in X$ is

$$N_\varepsilon(p) = \{ q \in X \mid d(p, q) \leqslant \varepsilon \}.$$

Def. A point $p \in X$ is called a **core point** iff $|N_\varepsilon(p)| \geqslant k$.

DBSCAN

Given: data points X, distance function $d(\cdot, \cdot)$, thresholds ε and k.

Def. The **ε-neighborhood** of a point $p \in X$ is

$$N_\varepsilon(p) = \{ q \in X \mid d(p, q) \leqslant \varepsilon \}.$$

Def. A point $p \in X$ is called a **core point** iff $|N_\varepsilon(p)| \geqslant k$.

Def. A point $p \in X$ is **directly density-reachable** from a point q iff:

$$p \in N_\varepsilon(q) \qquad |N_\varepsilon(q)| \geqslant k$$ (q is a core point)

DBSCAN

Given: data points X, distance function $d(\cdot,\cdot)$, thresholds ε and k.

Def. The **ε-neighborhood** of a point $p \in X$ is

$$N_\varepsilon(p) = \{ q \in X \mid d(p,q) \leqslant \varepsilon \}.$$

Def. A point $p \in X$ is called a **core point** iff $|N_\varepsilon(p)| \geqslant k$.

Def. A point $p \in X$ is **directly density-reachable** from a point q iff:

$$p \in N_\varepsilon(q) \qquad |N_\varepsilon(q)| \geqslant k$$ (q is a core point)

Not a symmetric relation!

DBSCAN

Given: data points X, distance function $d(\cdot,\cdot)$, thresholds ε and k.

Def. The **ε-neighborhood** of a point $p \in X$ is

$$N_\varepsilon(p) = \{\, q \in X \mid d(p,q) \leqslant \varepsilon \,\}.$$

Def. A point $p \in X$ is called a **core point** iff $|N_\varepsilon(p)| \geqslant k$.

Def. A point $p \in X$ is **directly density-reachable** from a point q iff:

$$p \in N_\varepsilon(q) \qquad |N_\varepsilon(q)| \geqslant k$$ (q is a core point)

Not a symmetric relation!

Def. A point $p \in X$ is **density reachable** from a point q if there exists a chain of direct density-reachability from q to p.

DBSCAN

Given: data points X, distance function $d(\cdot, \cdot)$, thresholds ε and k.

Def. The **ε-neighborhood** of a point $p \in X$ is

$$N_\varepsilon(p) = \{ q \in X \mid d(p, q) \leqslant \varepsilon \}.$$

Def. A point $p \in X$ is called a **core point** iff $|N_\varepsilon(p)| \geqslant k$.

Def. A point $p \in X$ is **directly density-reachable** from a point q iff:

$$p \in N_\varepsilon(q) \qquad |N_\varepsilon(q)| \geqslant k$$ (q is a core point)

Not a symmetric relation!

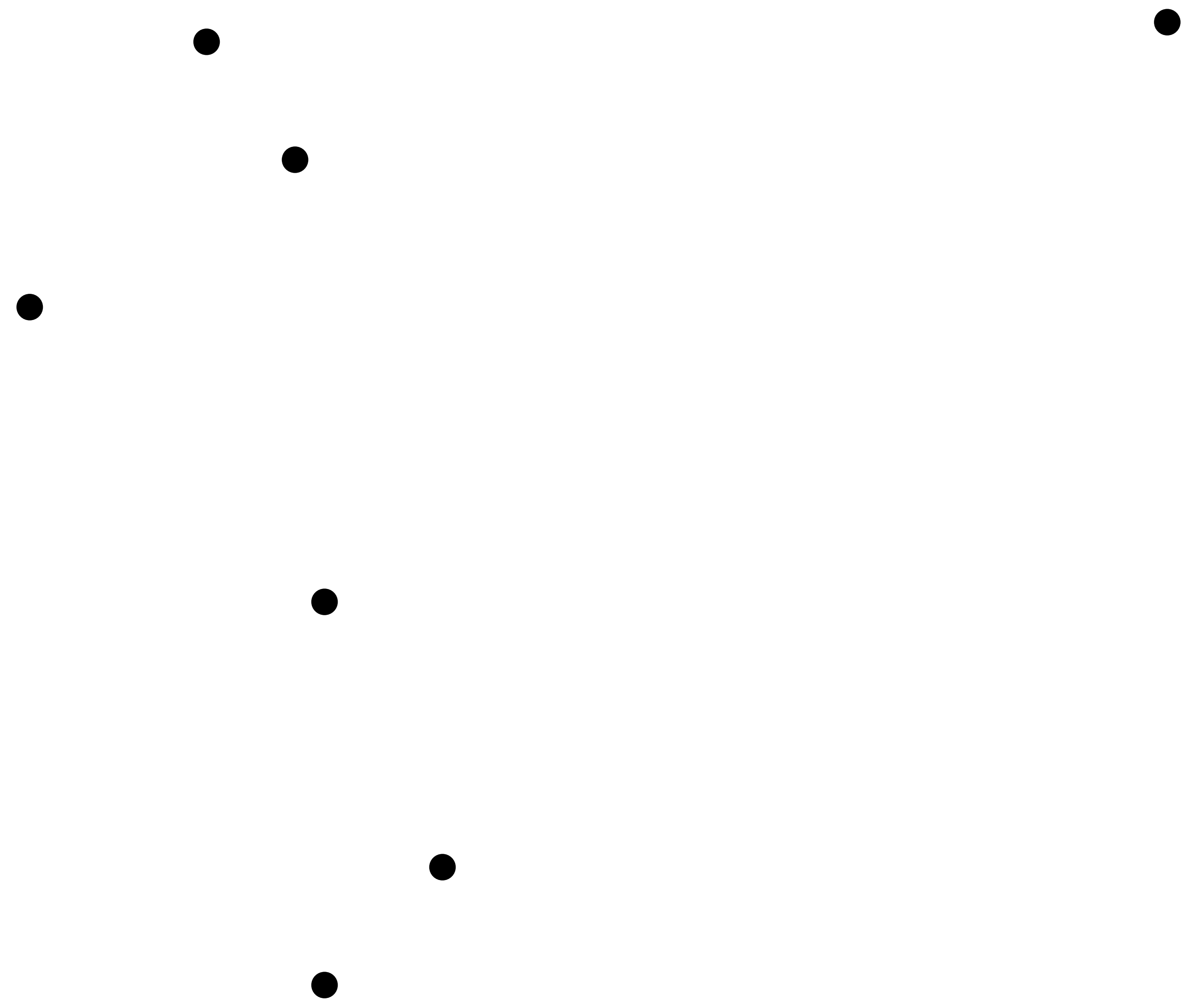
Def. A point $p \in X$ is **density reachable** from a point q if there exists a chain of direct density-reachability from q to p.

Def. A point $p \in X$ is **density connected** to a point q if there exists a (core) point r such that both p and q are density-reachable from r.

DBSCAN example

Legend

$k = 3$

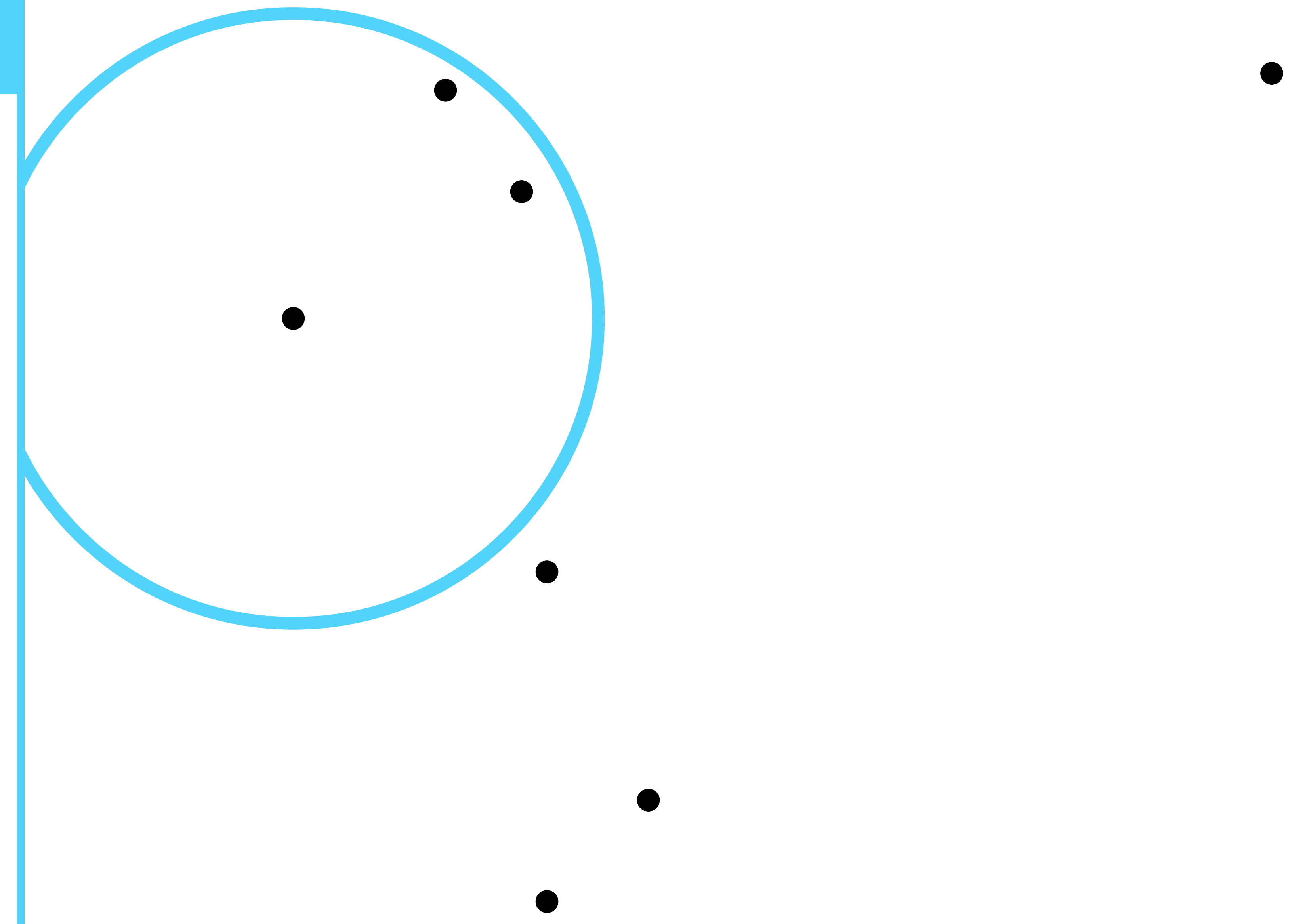


DBSCAN example

Legend

$k = 3$

Distance ε



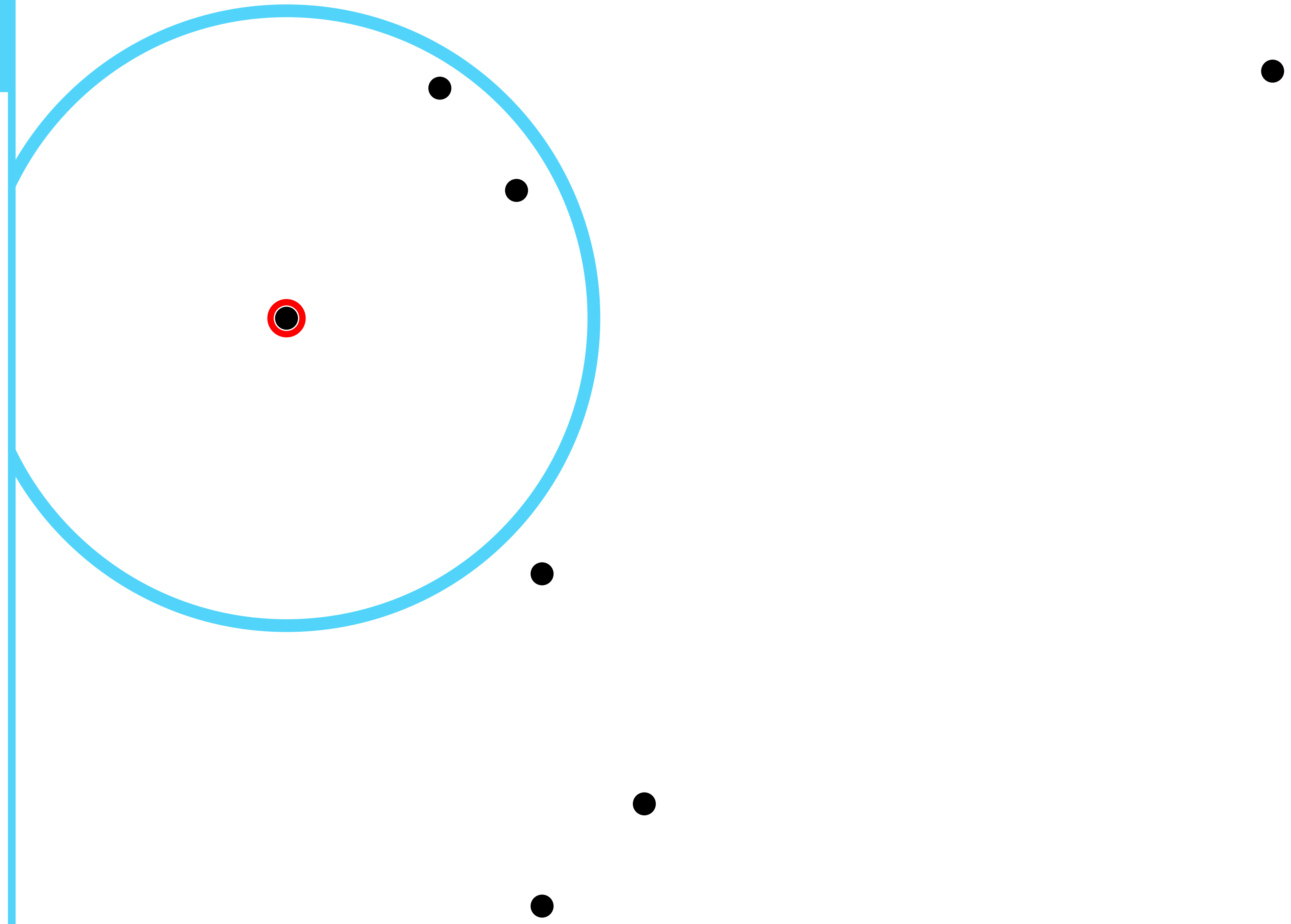
DBSCAN example

Legend

$k = 3$

Distance ε

Core points



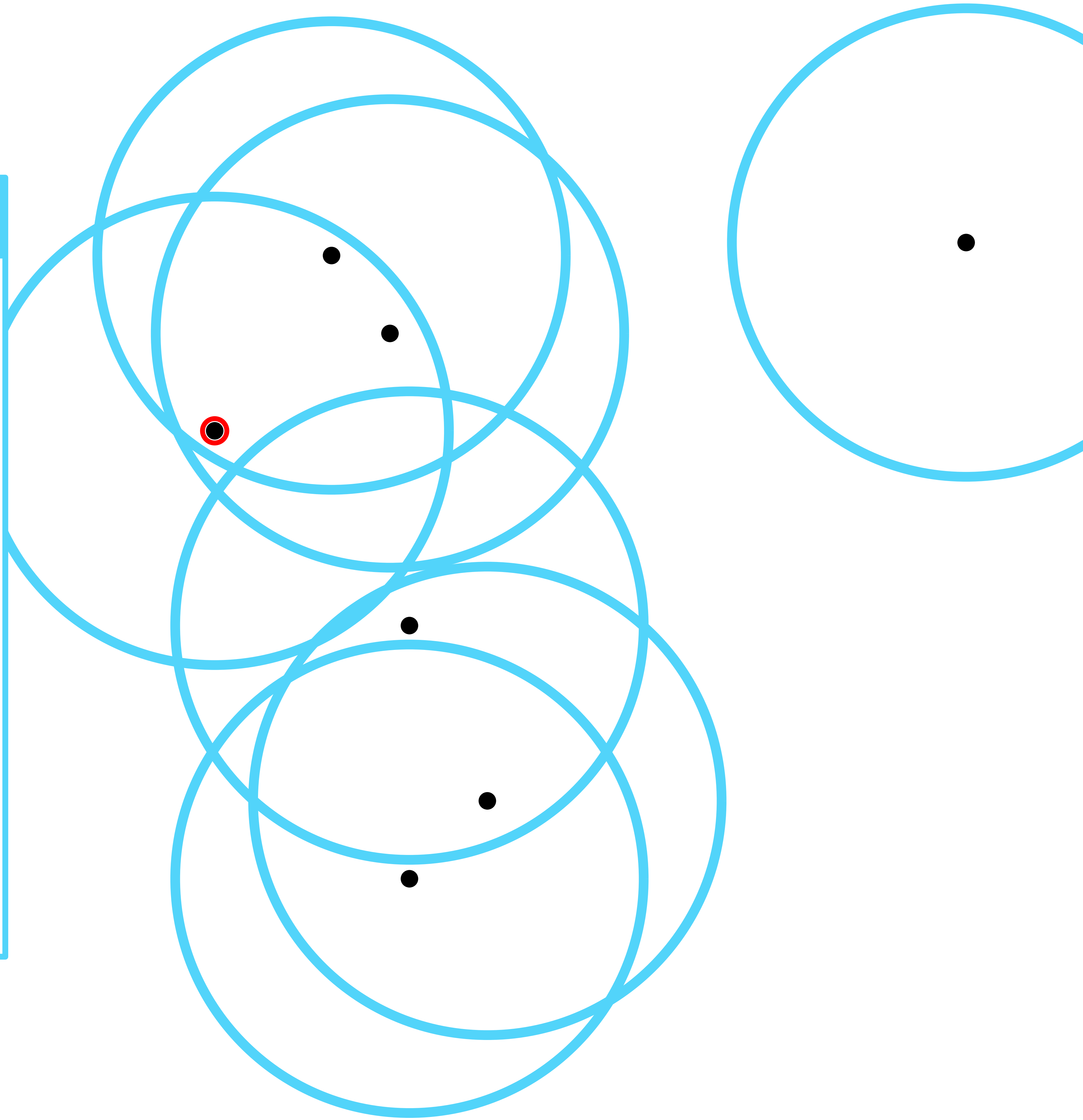
DBSCAN example

Legend

$k = 3$

Distance ε

Core points



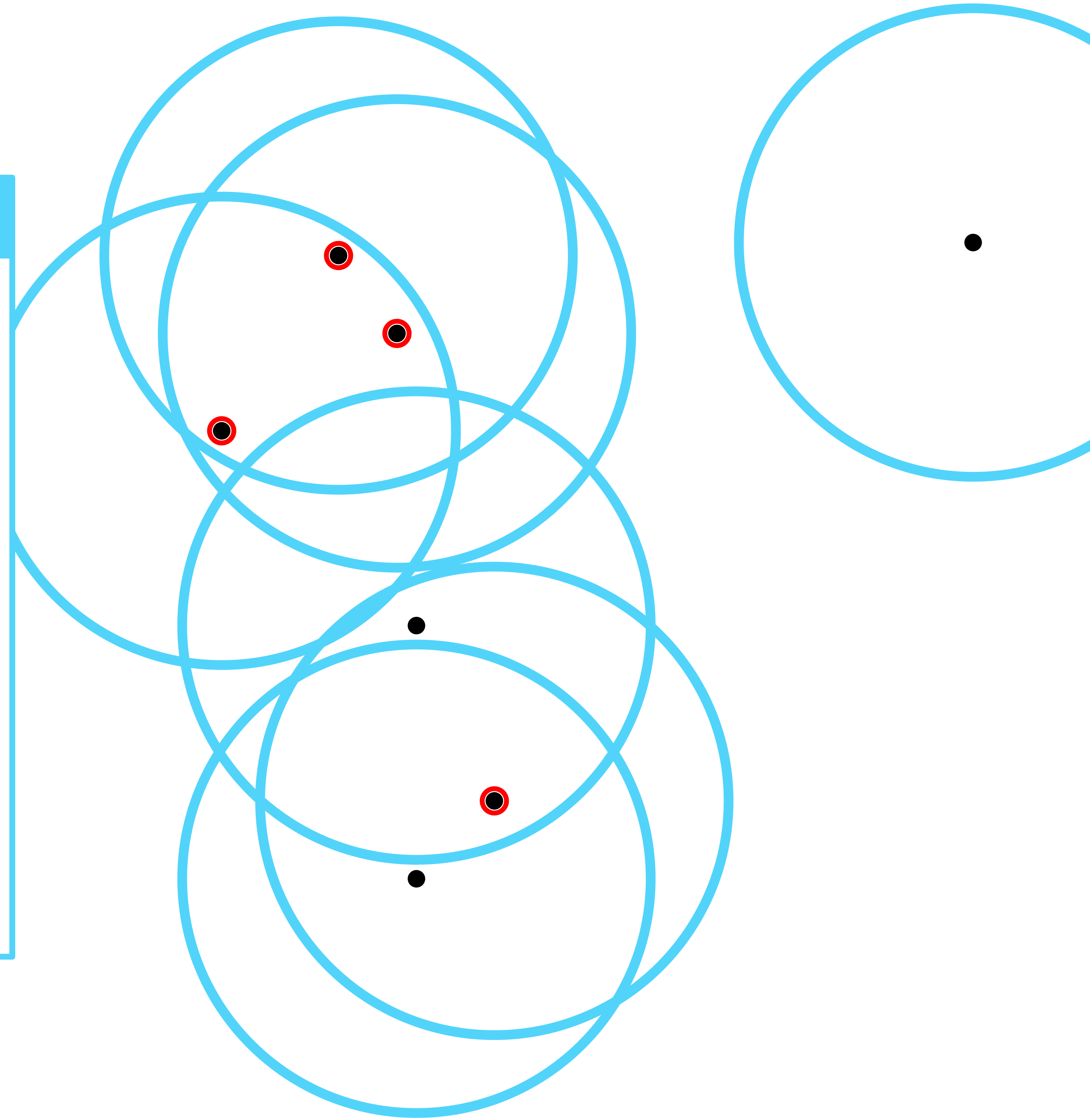
DBSCAN example

Legend

$k = 3$

Distance ε

Core points



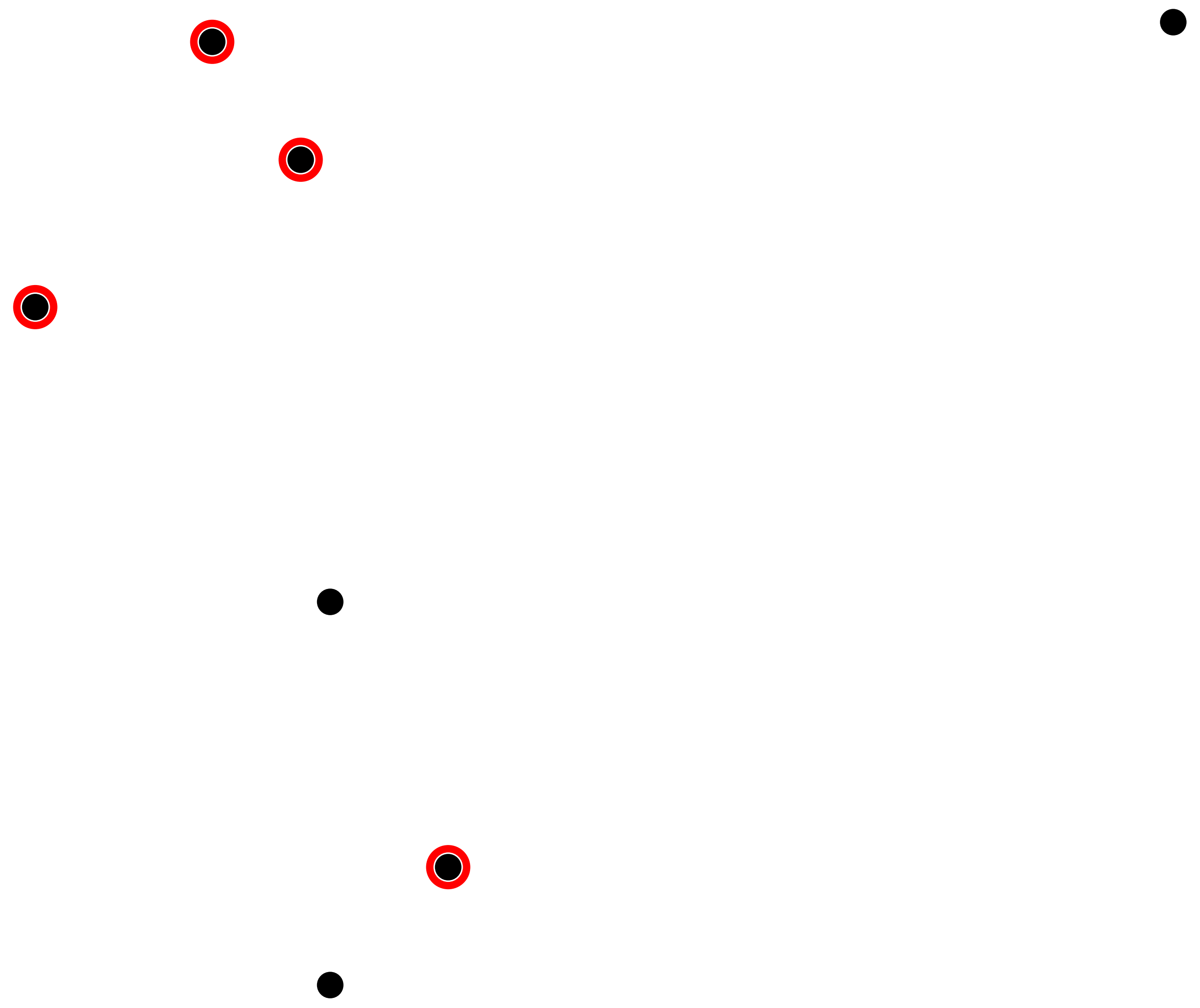
DBSCAN example

Legend

$k = 3$

Distance ε

Core points



DBSCAN example

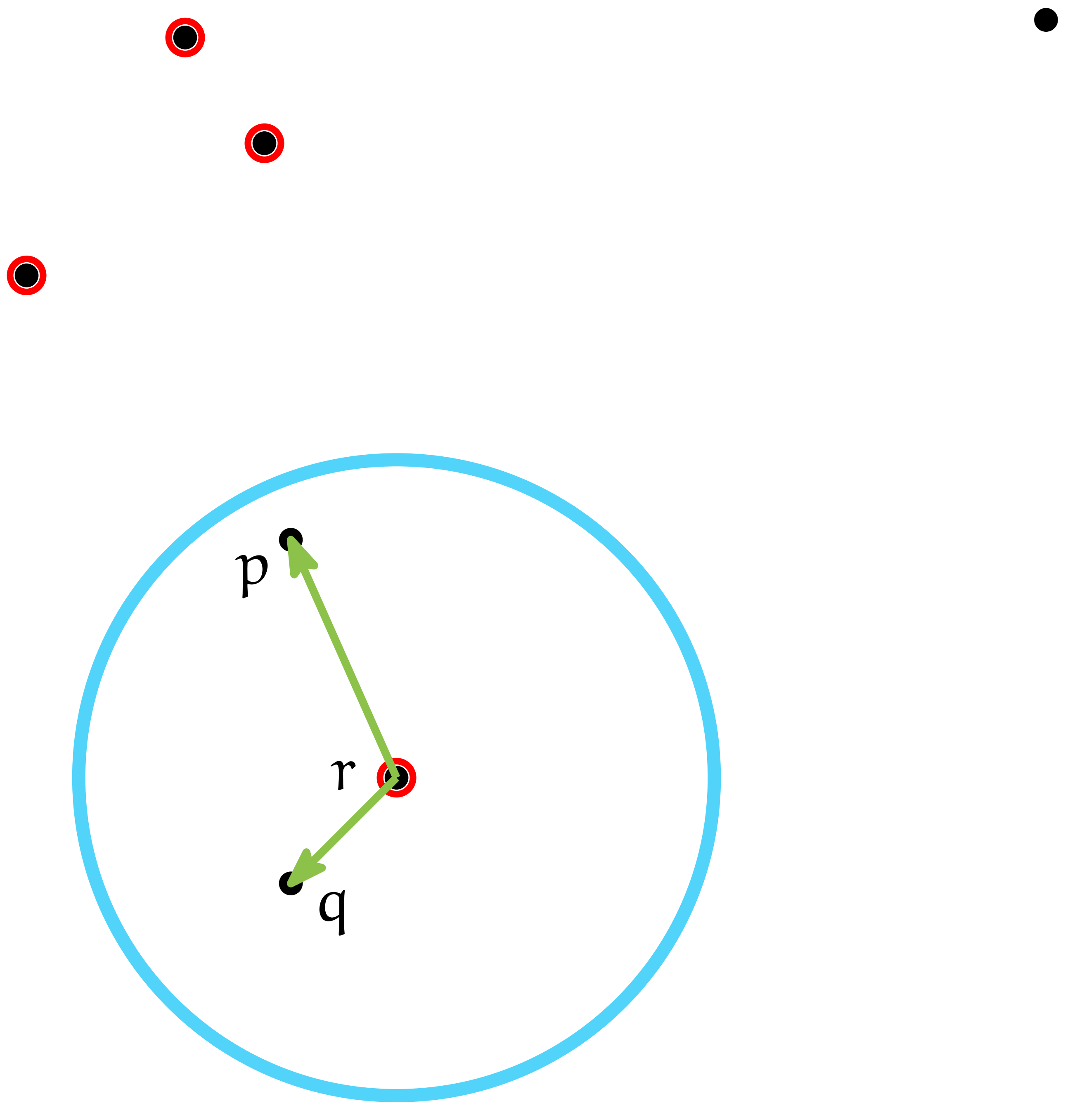
Legend

$k = 3$

Distance ε

Core points

Density connected



DBSCAN example

Legend

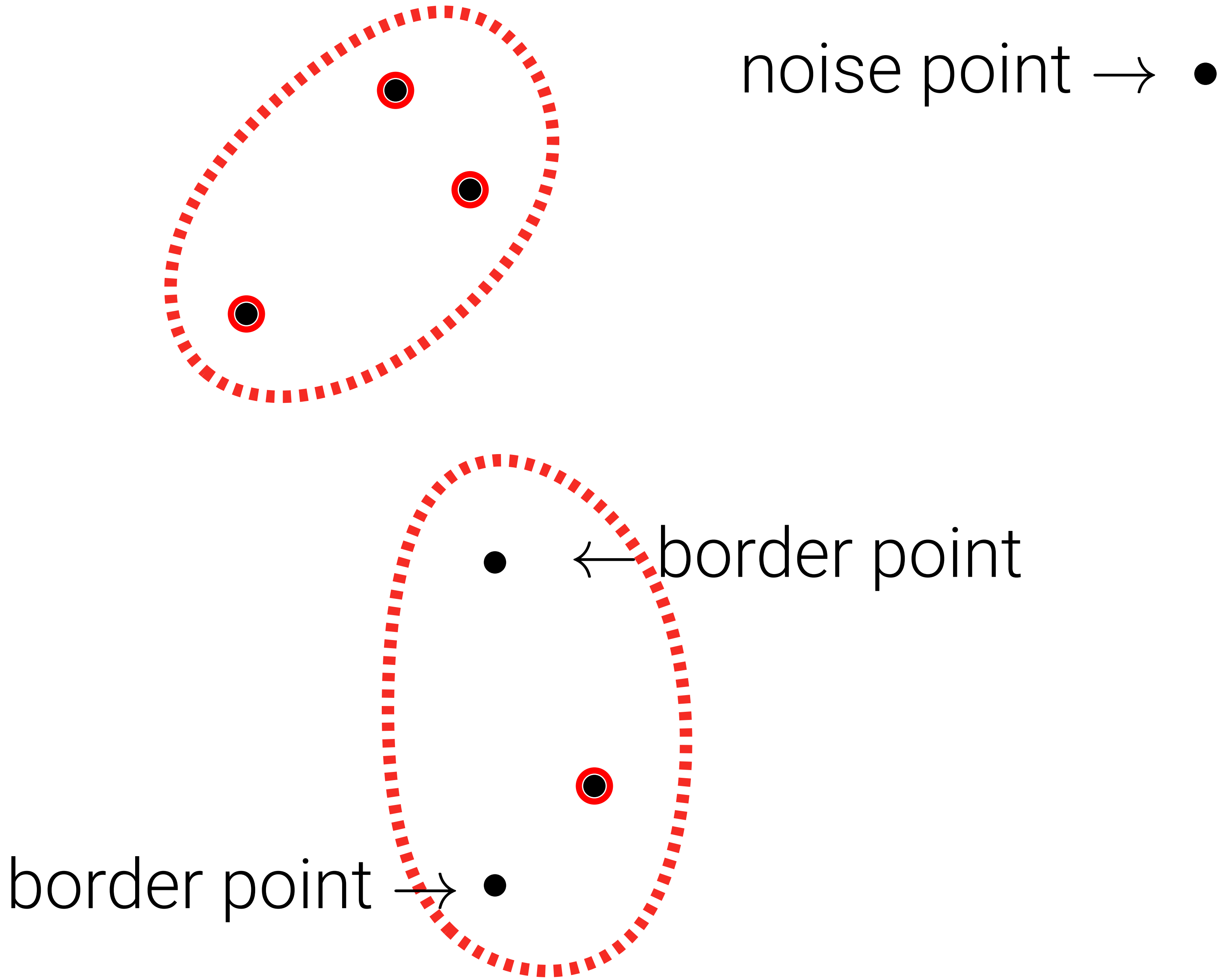
$k = 3$

Distance ε

Core points

Density connected

DBSCAN clustering



DBSCAN example

Legend

$k = 3$

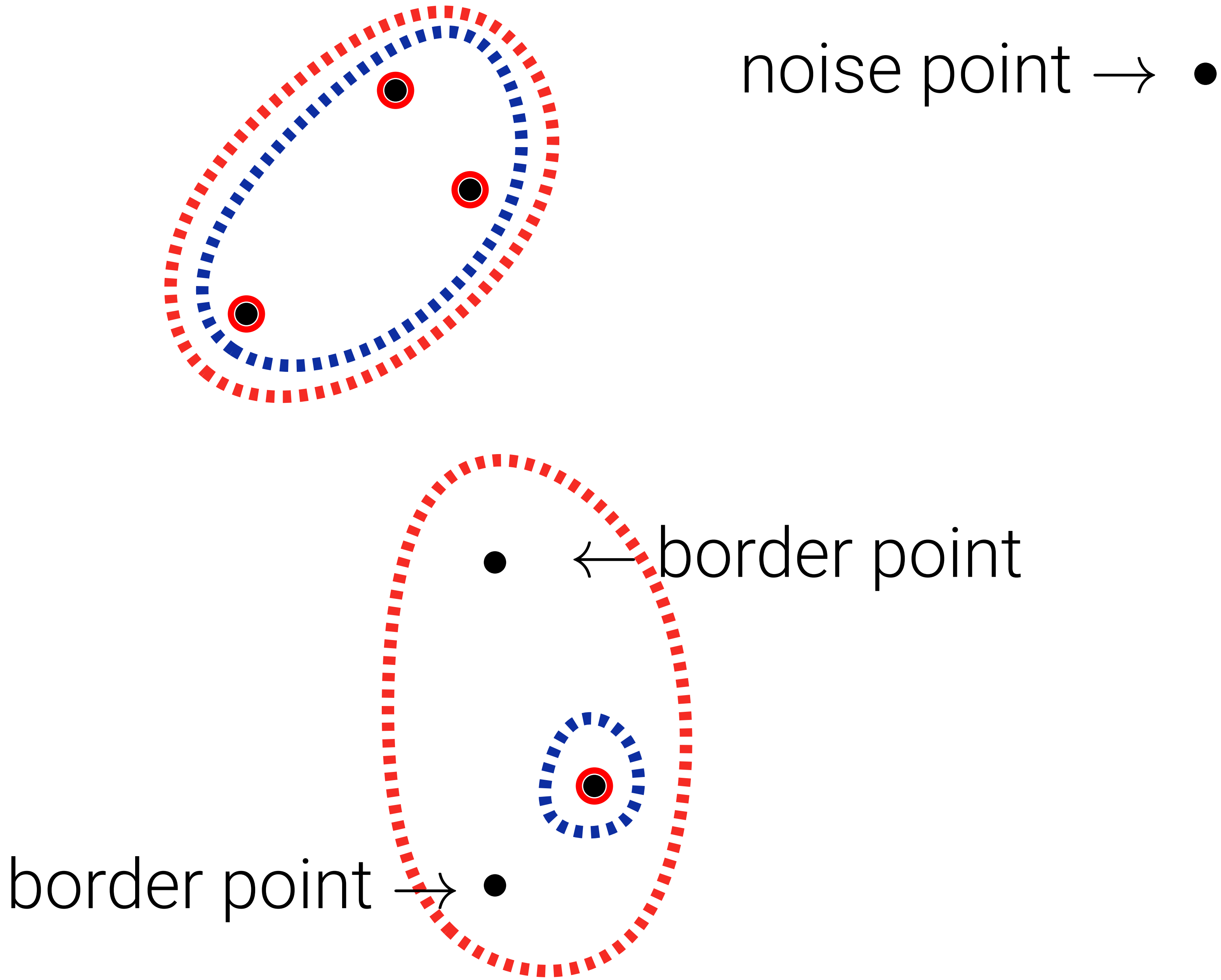
Distance ε

Core points

Density connected

DBSCAN clustering

DBSCAN* clustering



DBSCAN example

Legend

$k = 3$

Distance ε

Core points

Density connected

DBSCAN clustering

DBSCAN* clustering

border

Runtime

Naive algorithm runs in $\mathcal{O}(n^2)$ time.

DBSCAN example

Legend

$k = 3$

Distance ε

Core points

Density connected

DBSCAN clustering

DBSCAN* clustering

Runtime

Naive algorithm runs in $\mathcal{O}(n^2)$ time.

“Since the Eps-Neighborhoods are expected to be small compared to the size of the whole data space, the average run time complexity of a single region query is $\mathcal{O}(\log n)$. (...) Thus, the average run time complexity of DBSCAN is $\mathcal{O}(n * \log n)$.”

Results

Everywhere: ε free, k fixed constant, Euclidean distances

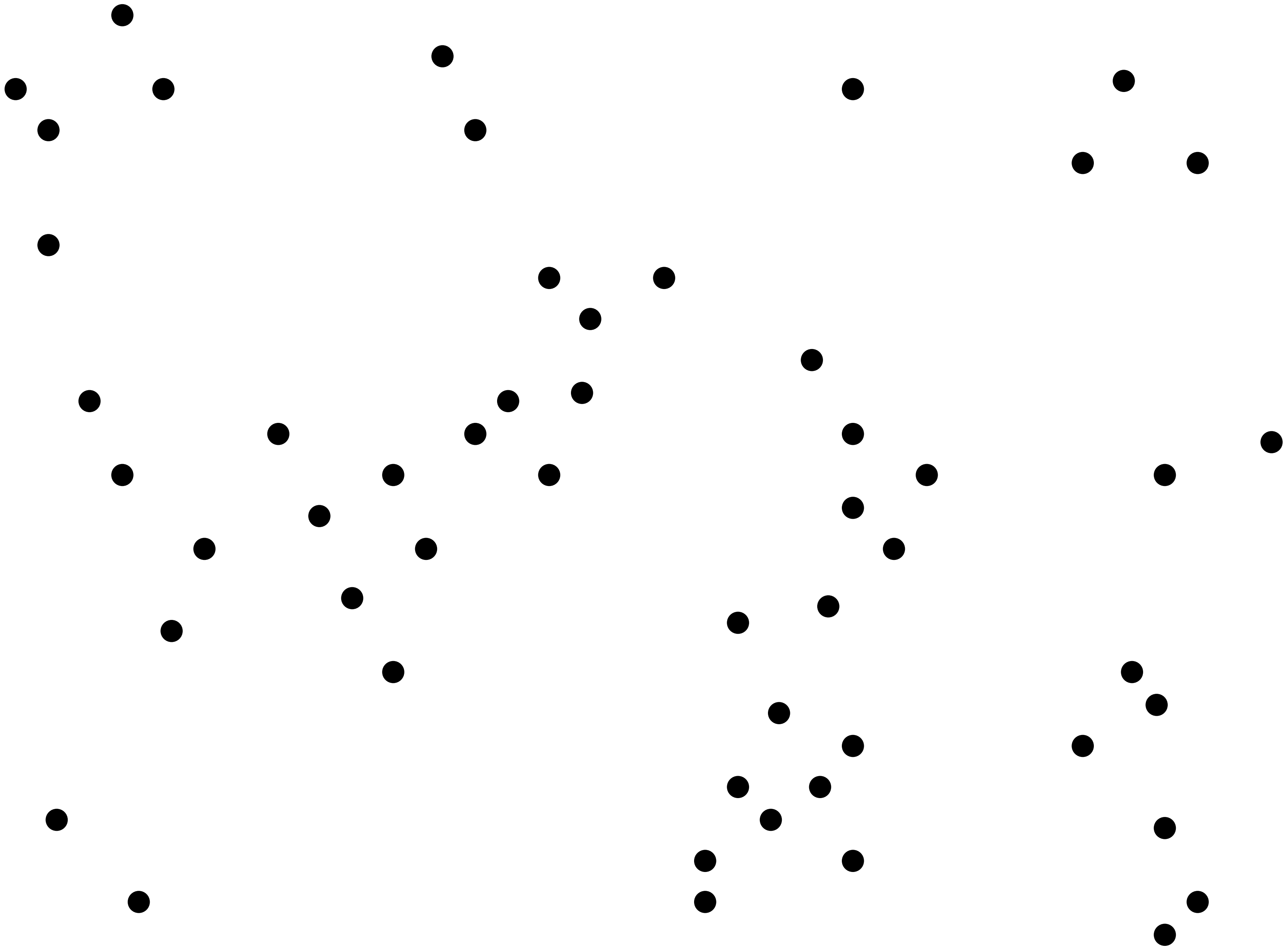
	2D	dD
DBSCAN	$\mathcal{O}(n \log n)$	$\mathcal{O}(n^{2-\frac{2}{\lceil d/2 \rceil + 1}+\gamma})$ $\gamma > 0$
HDBSCAN	$\mathcal{O}(n \log n)$ expected	$\times$

Results

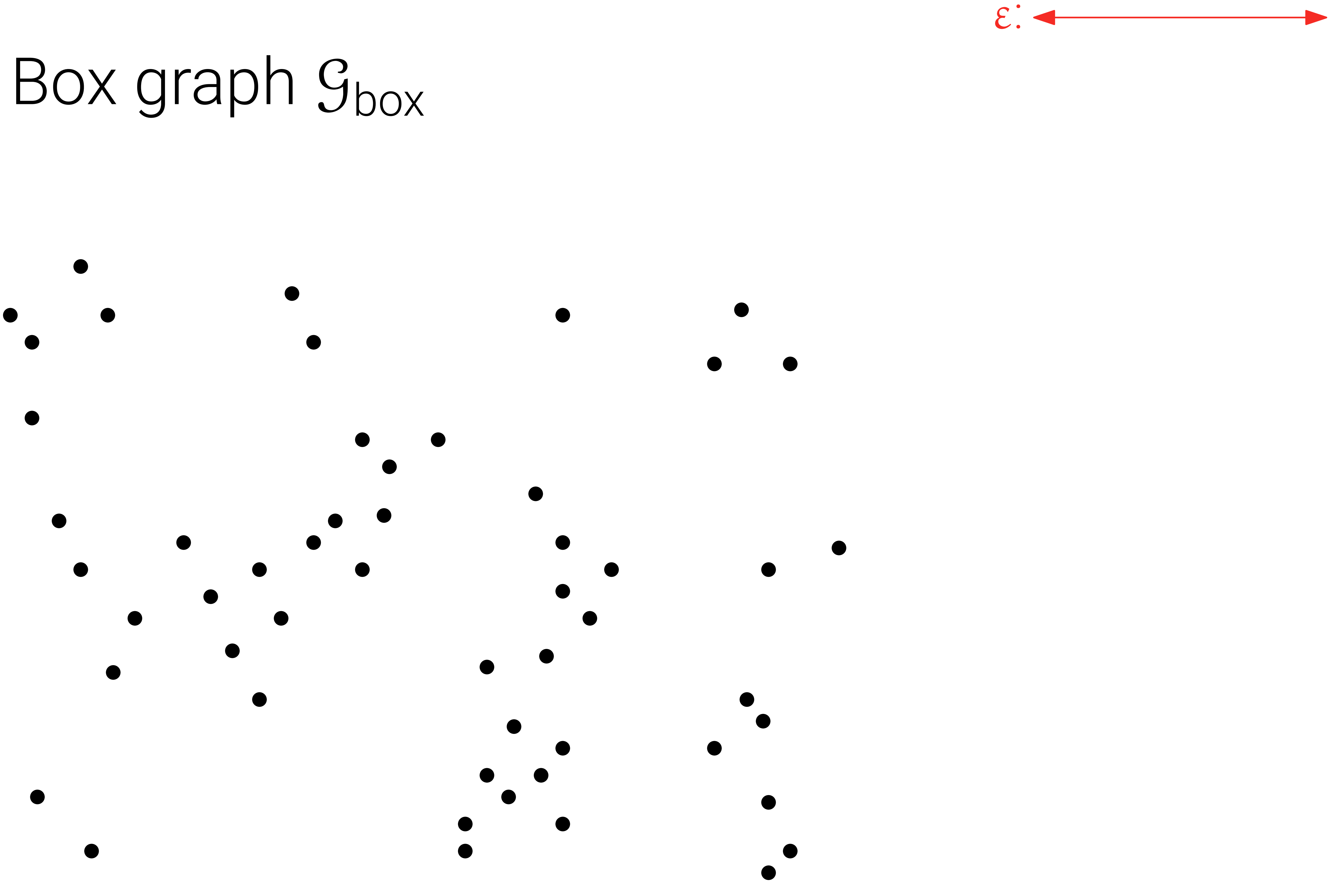
Everywhere: ε free, k fixed constant, Euclidean distances

	2D	dD
DBSCAN	$\mathcal{O}(n \log n)$ ←	$\mathcal{O}(n^{2-\frac{2}{\lceil d/2 \rceil + 1}+\gamma})$ $\gamma > 0$
HDBSCAN	$\mathcal{O}(n \log n)$ expected	$\times$

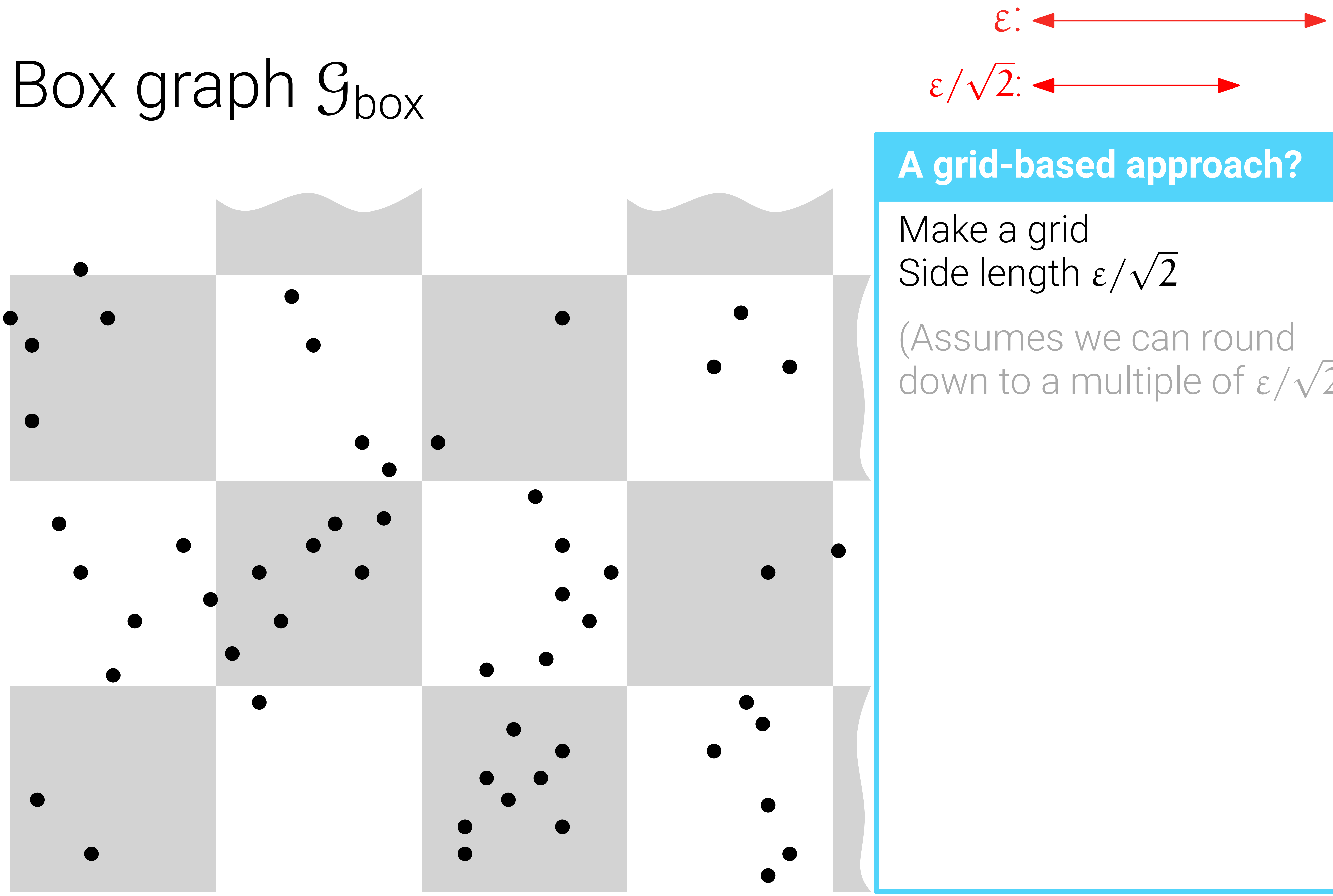
Box graph $\mathcal{G}_{\text{box}}$



Box graph $\mathcal{G}_{\text{box}}$



Box graph $\mathcal{G}_{box}$

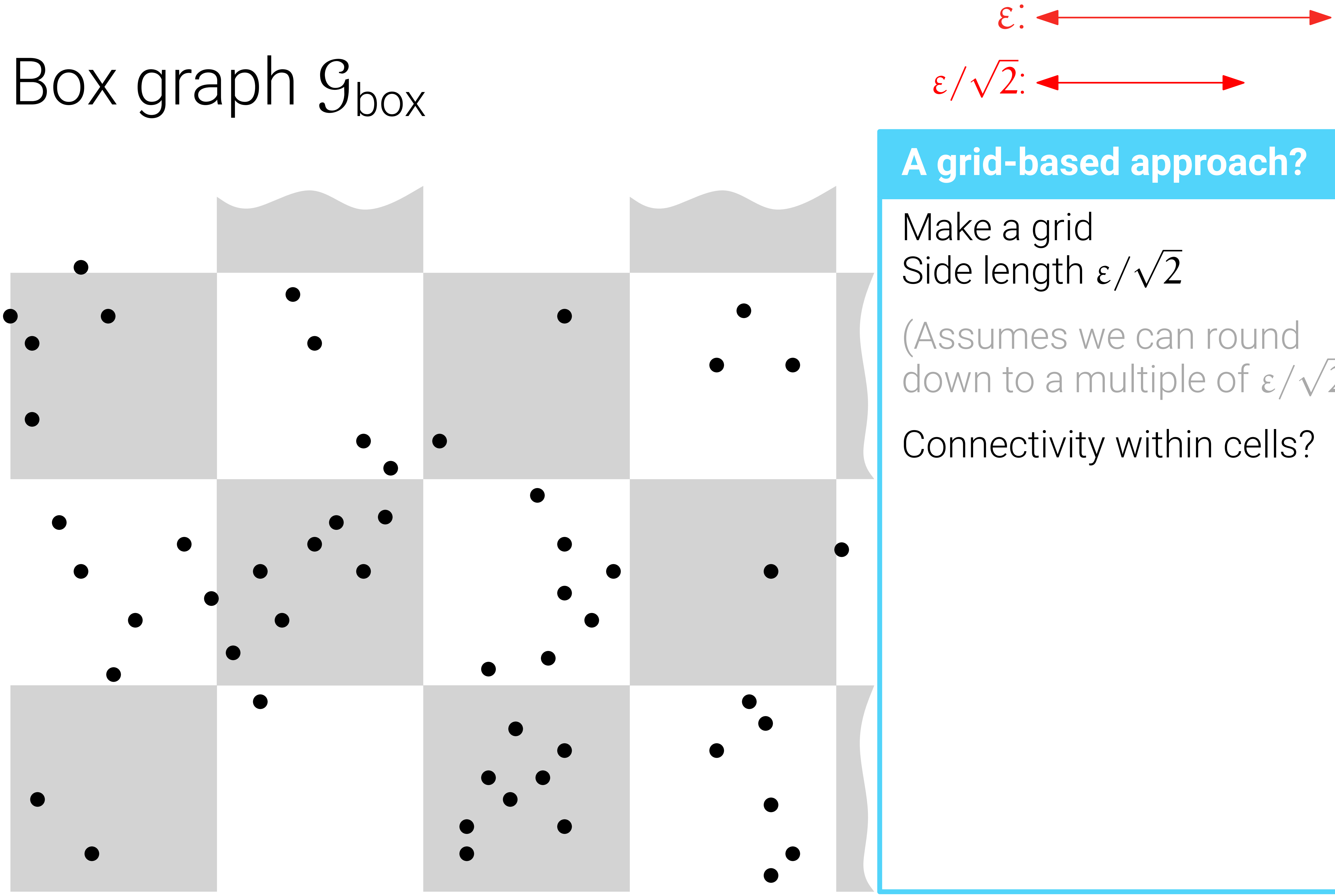


A grid-based approach?

Make a grid
Side length $\varepsilon/\sqrt{2}$

(Assumes we can round down to a multiple of $\varepsilon/\sqrt{2}$)

Box graph $\mathcal{G}_{box}$



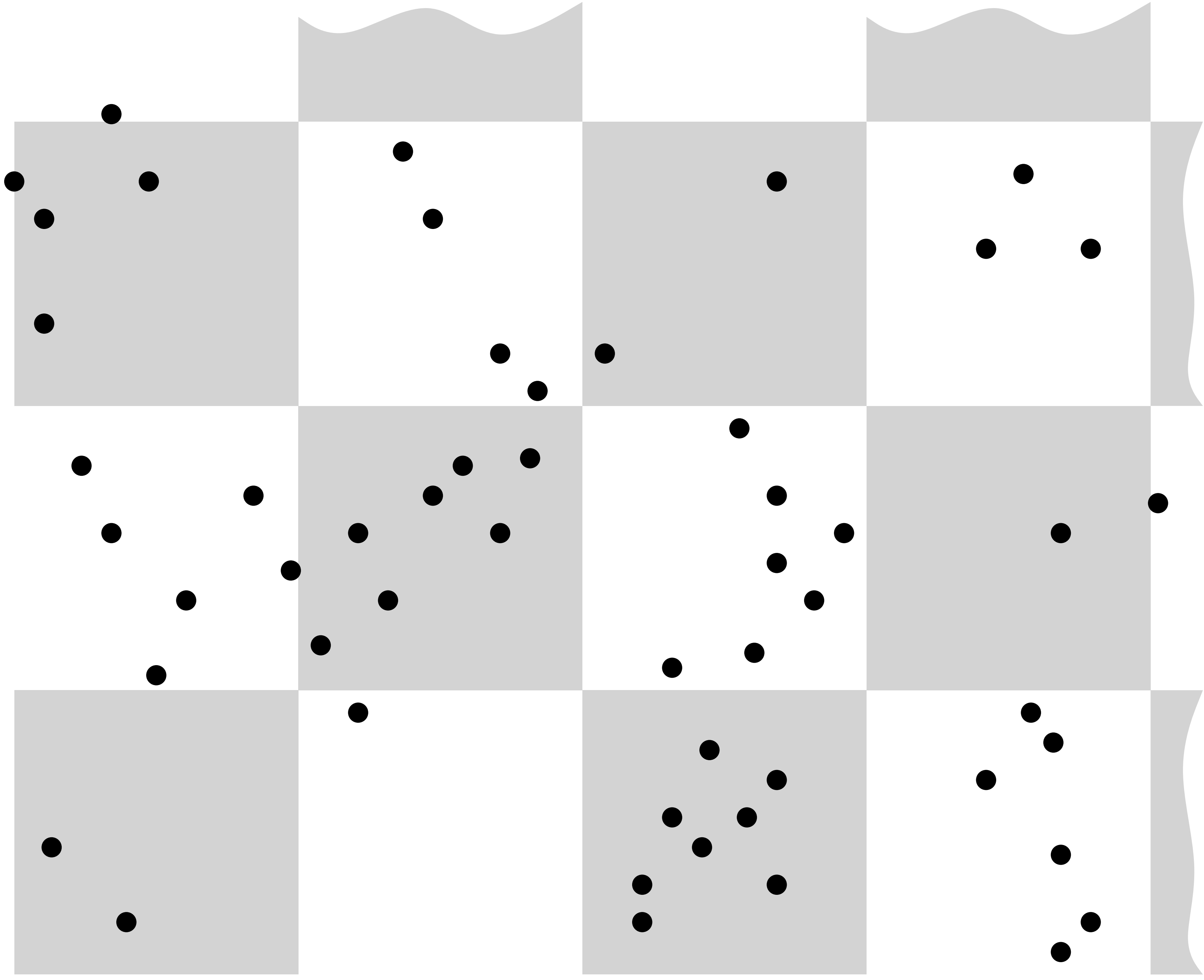
A grid-based approach?

Make a grid
Side length $\varepsilon/\sqrt{2}$

(Assumes we can round down to a multiple of $\varepsilon/\sqrt{2}$)

Connectivity within cells?

Box graph $\mathcal{G}_{\text{box}}$



A grid-based approach?

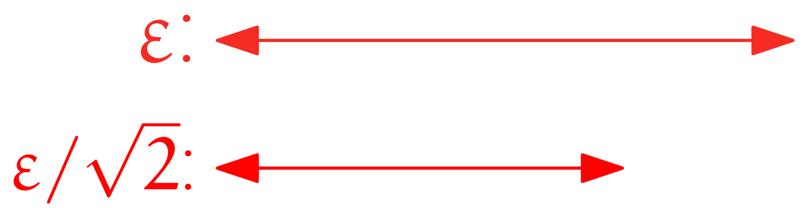
Make a grid
Side length $\varepsilon/\sqrt{2}$

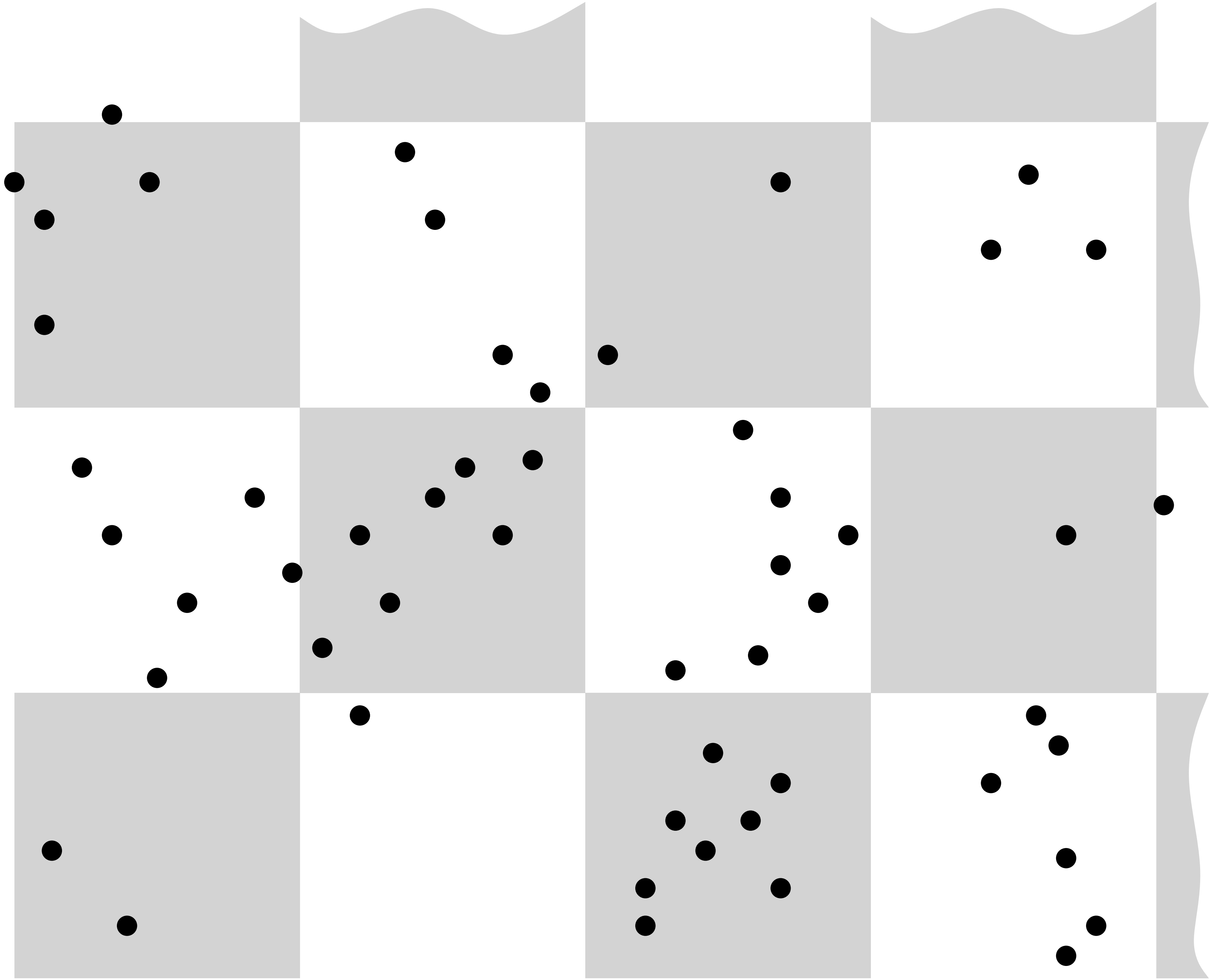
(Assumes we can round down to a multiple of $\varepsilon/\sqrt{2}$)

Connectivity within cells?

Between points in different cells?

Box graph $\mathcal{G}_{\text{box}}$





A grid-based approach?

Make a grid
Side length $\varepsilon/\sqrt{2}$

(Assumes we can round down to a multiple of $\varepsilon/\sqrt{2}$)

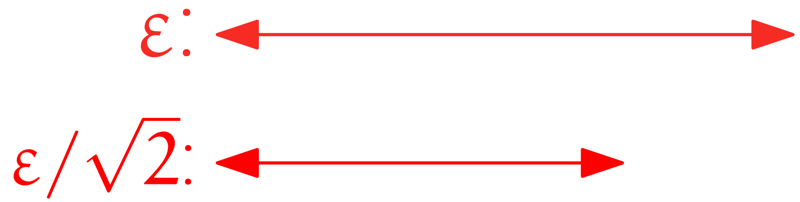
Connectivity within cells?

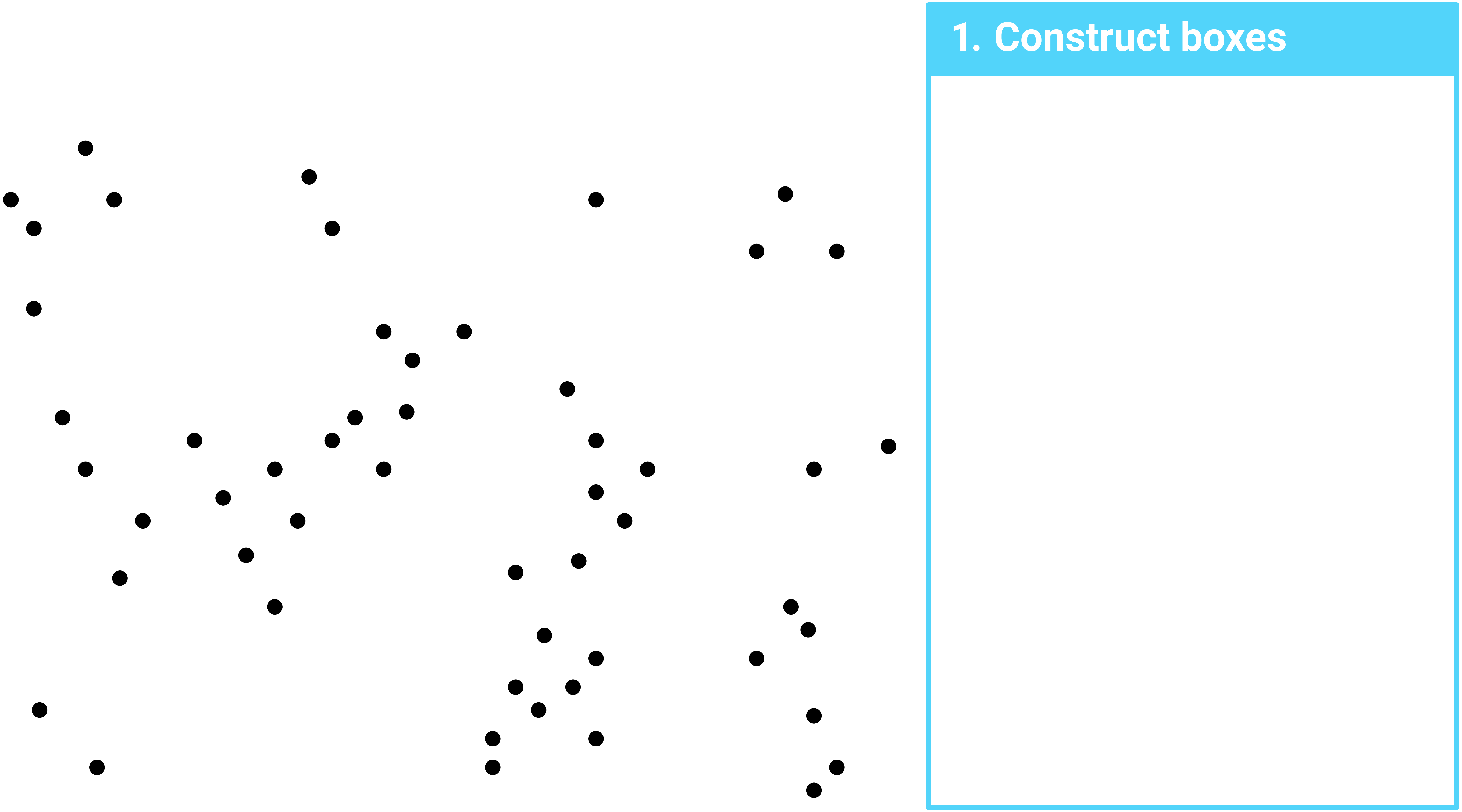
Between points in different cells?

Not clear how to get a runtime bound in n without assumption on the distribution.

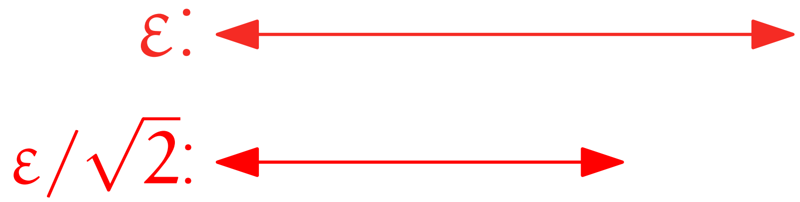
Be more flexible...

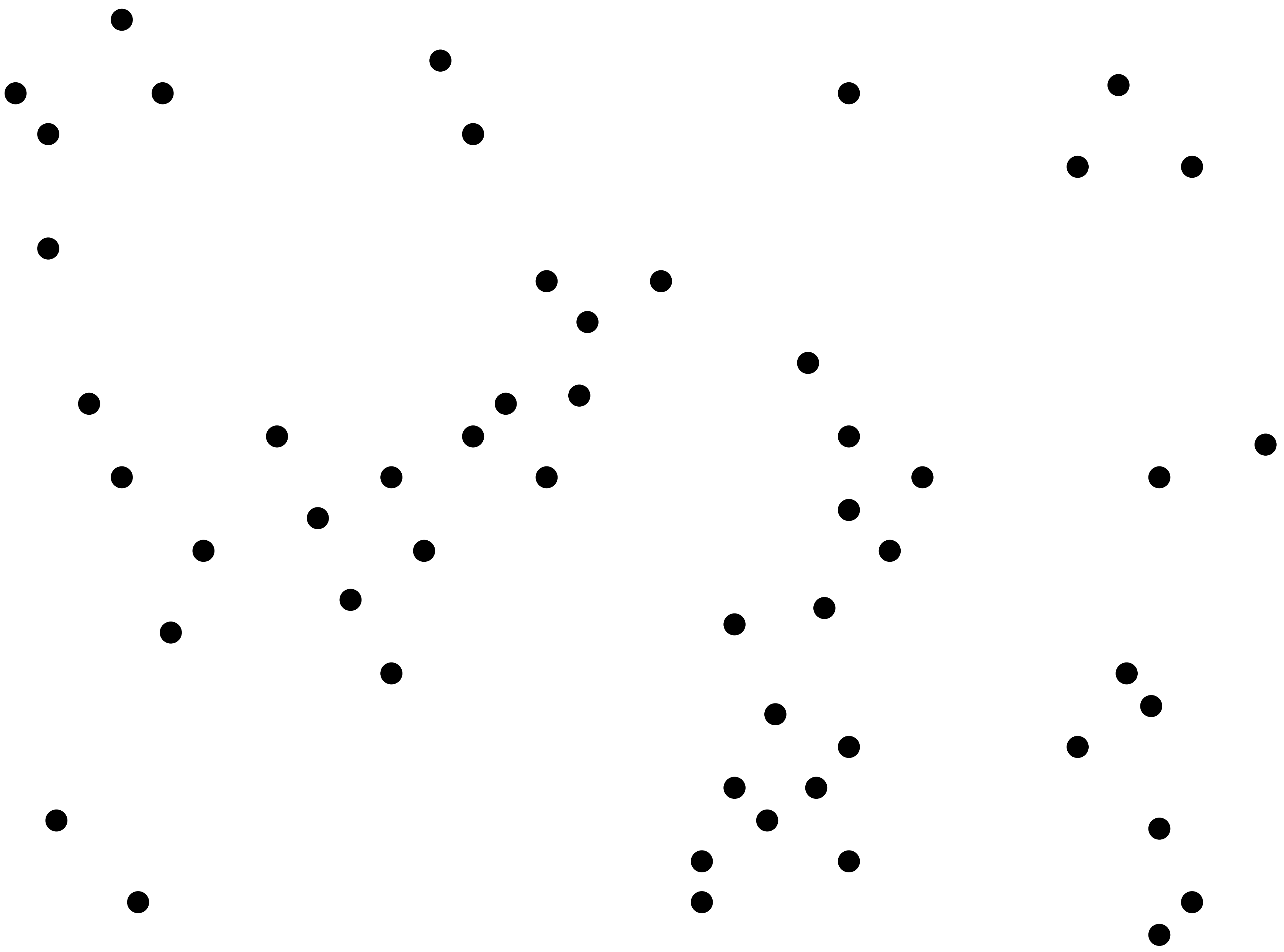
Box graph $\mathcal{G}_{box}$





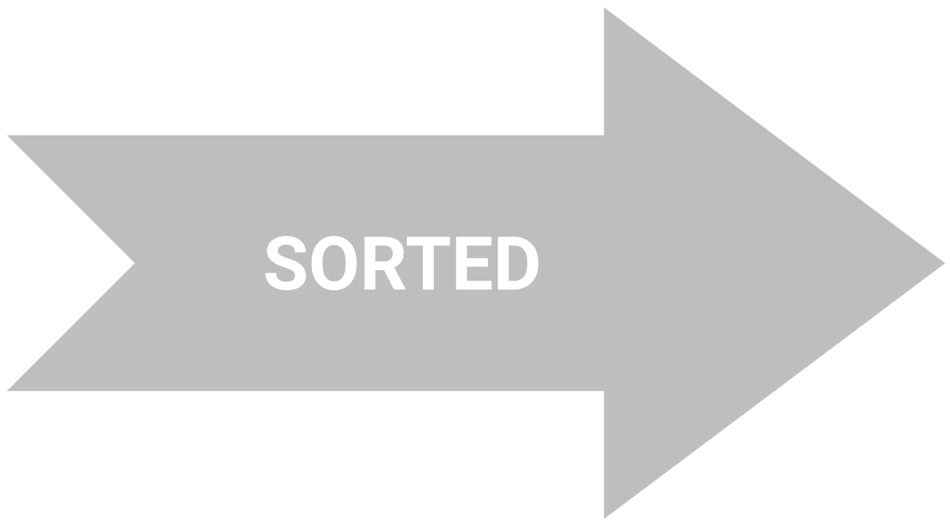
Box graph $\mathcal{G}_{\text{box}}$



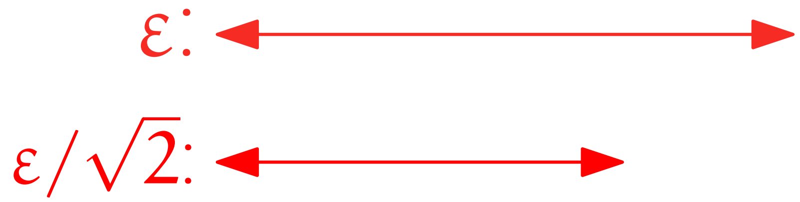


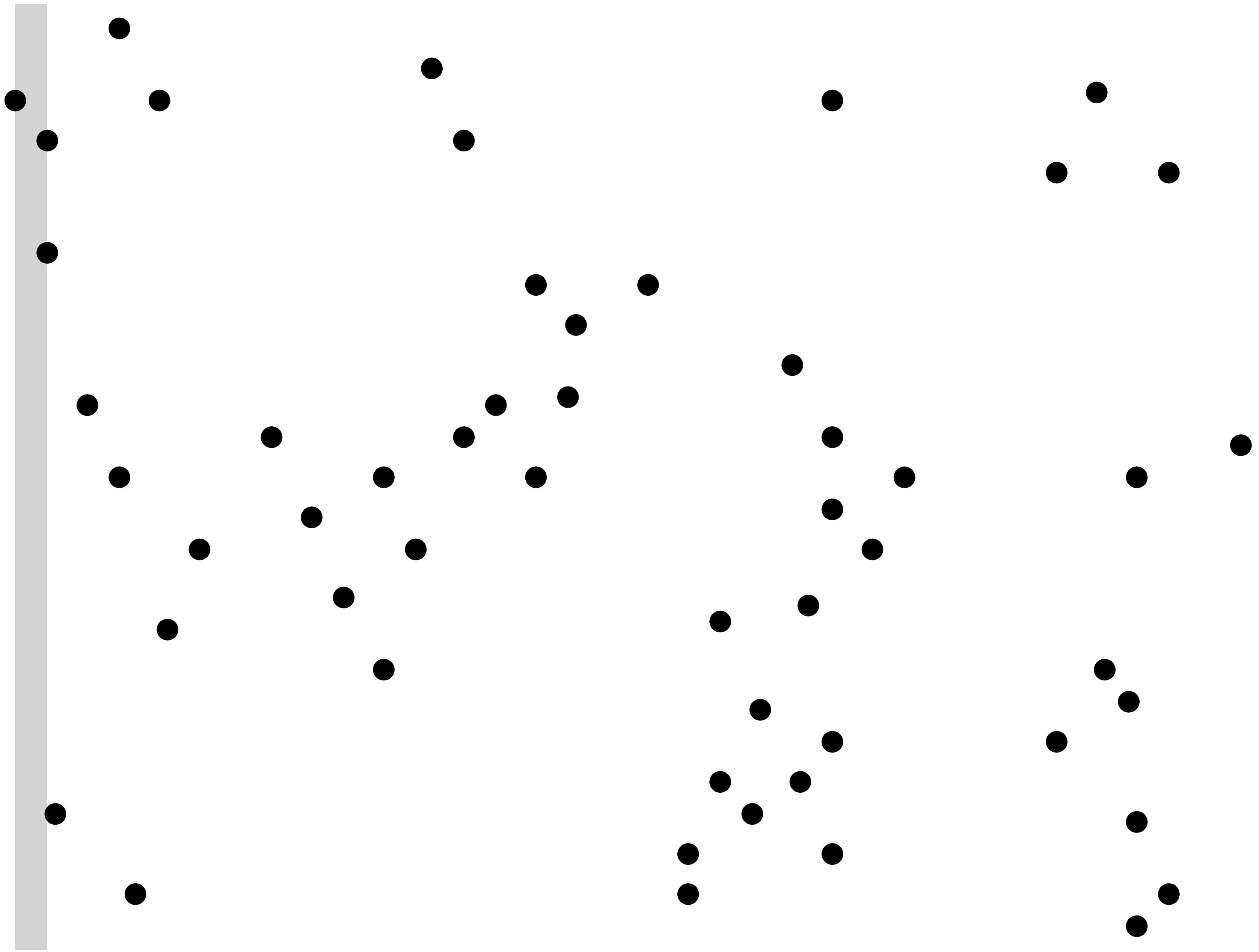
1. Construct boxes

Add points as long as strip width $\leqslant \varepsilon/\sqrt{2}$.



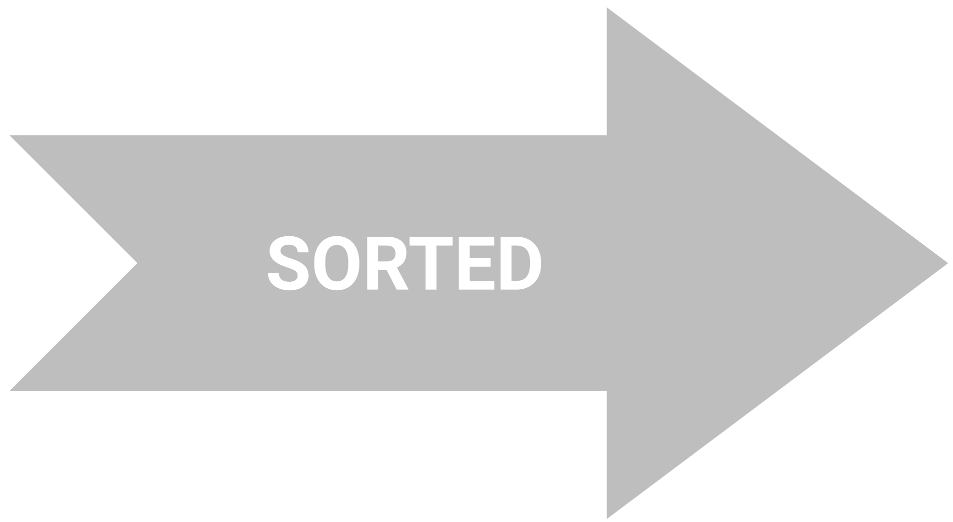
Box graph $\mathcal{G}_{\text{box}}$





1. Construct boxes

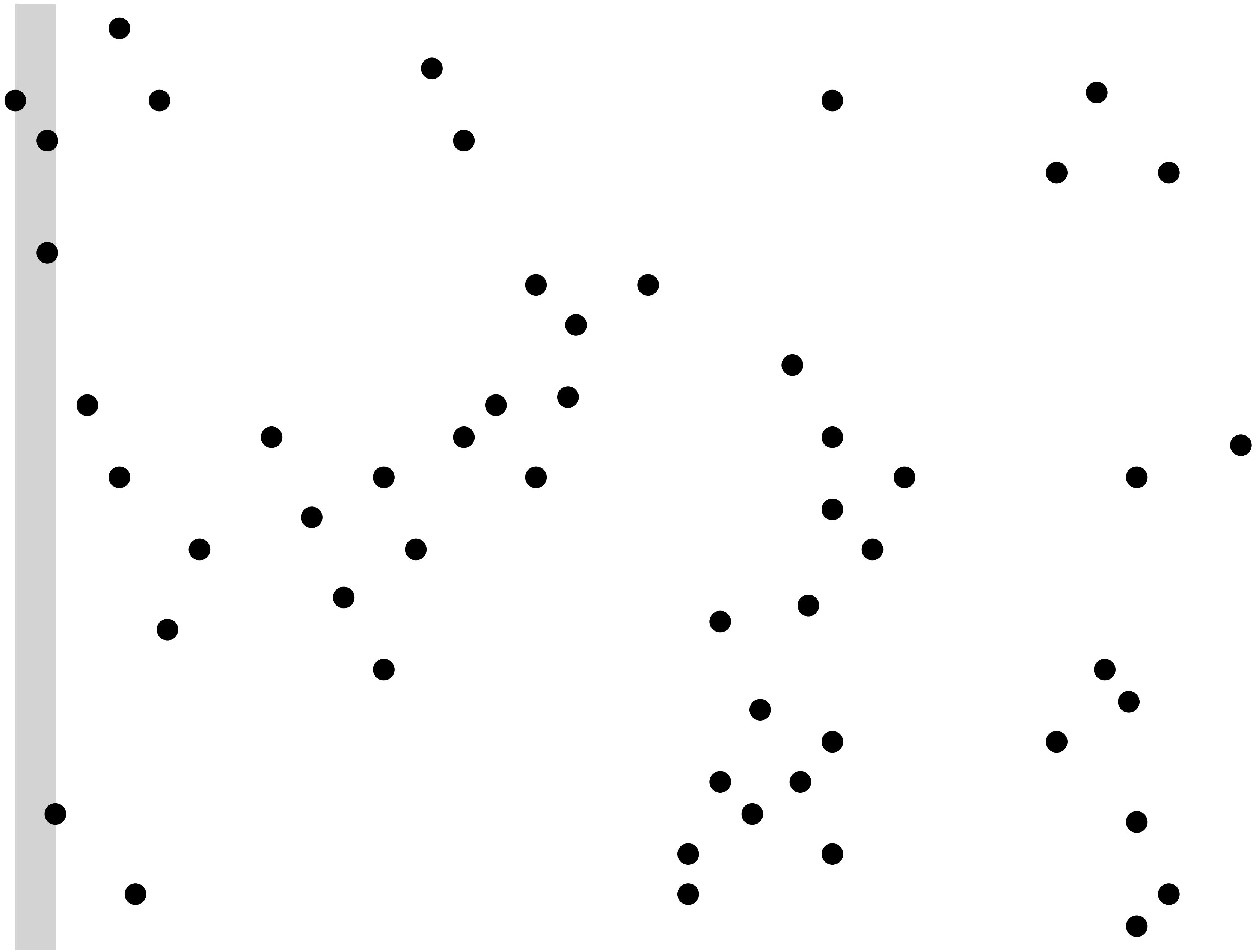
Add points as long as strip width $\leqslant \varepsilon/\sqrt{2}$.



Box graph $\mathcal{G}_{\text{box}}$

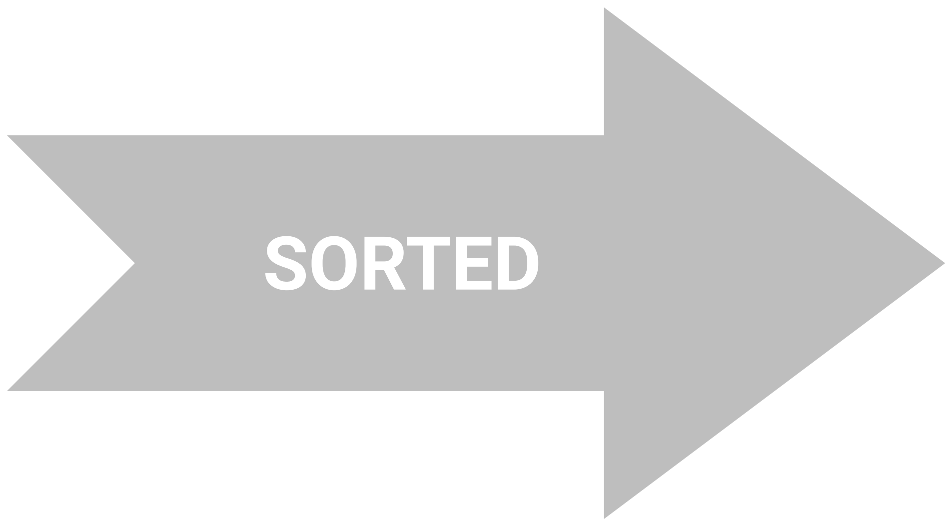
ε:

$\varepsilon/\sqrt{2}$:



1. Construct boxes

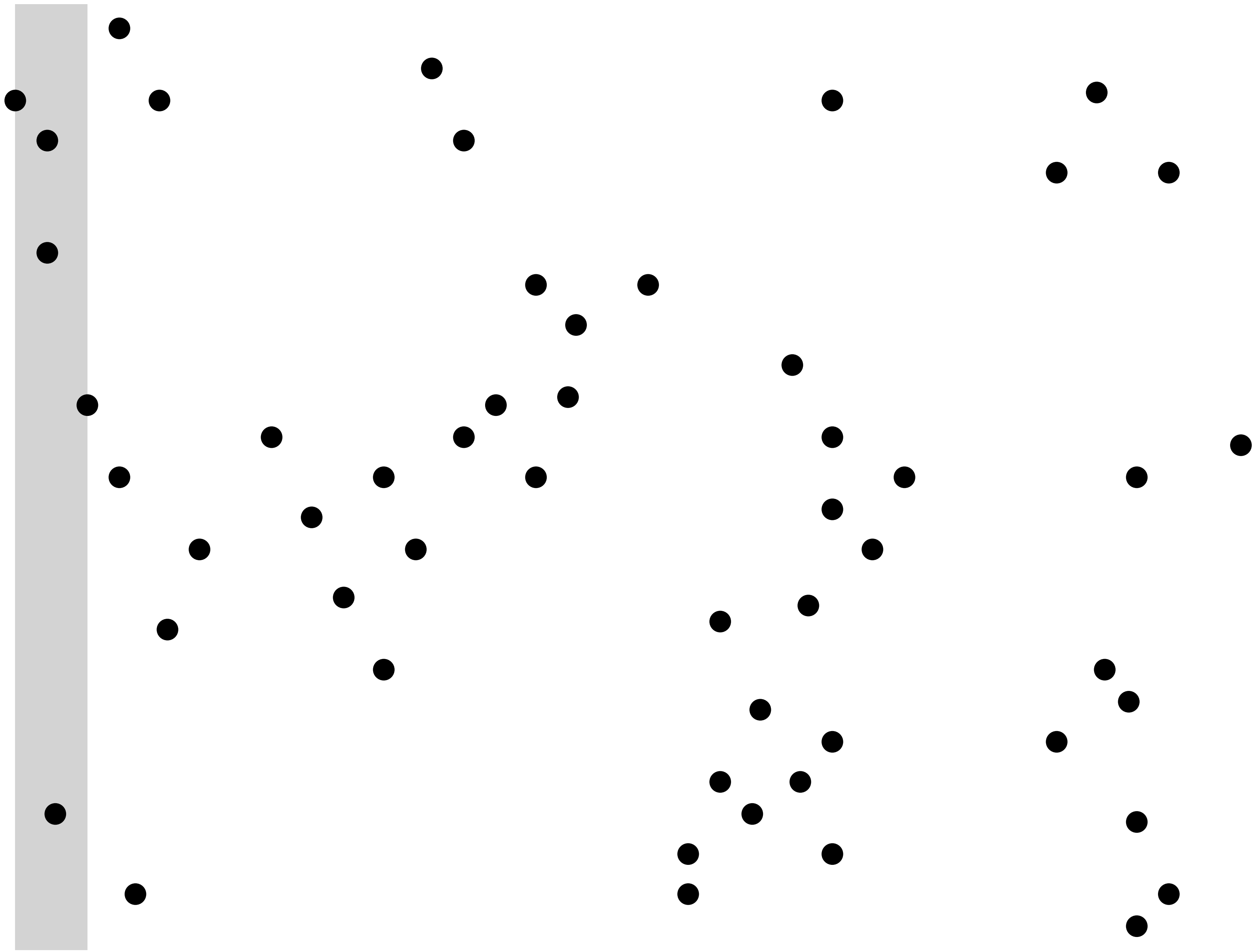
Add points as long as strip width $\leqslant \varepsilon/\sqrt{2}$.



Box graph $\mathcal{G}_{\text{box}}$

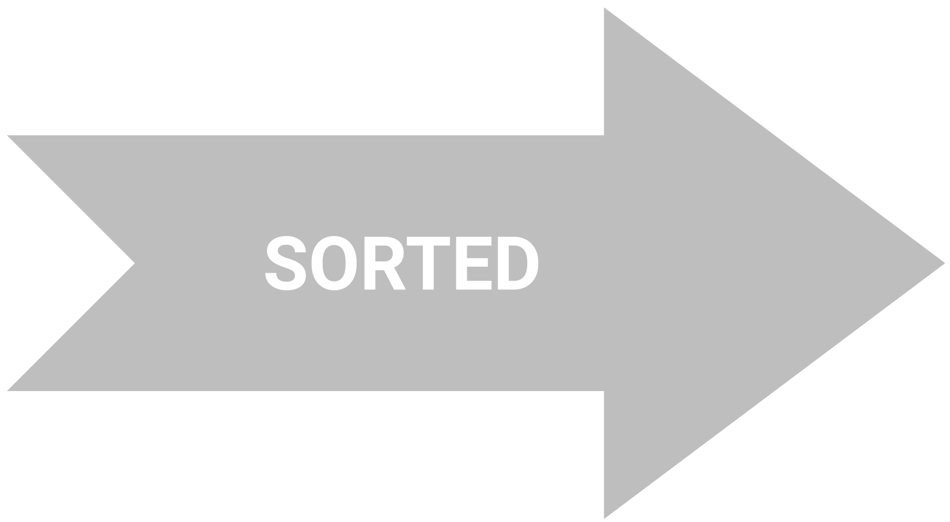
ε:

$\varepsilon/\sqrt{2}$:

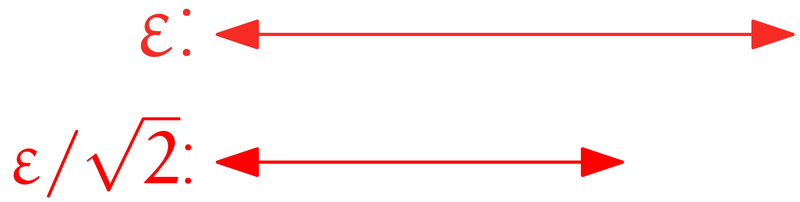


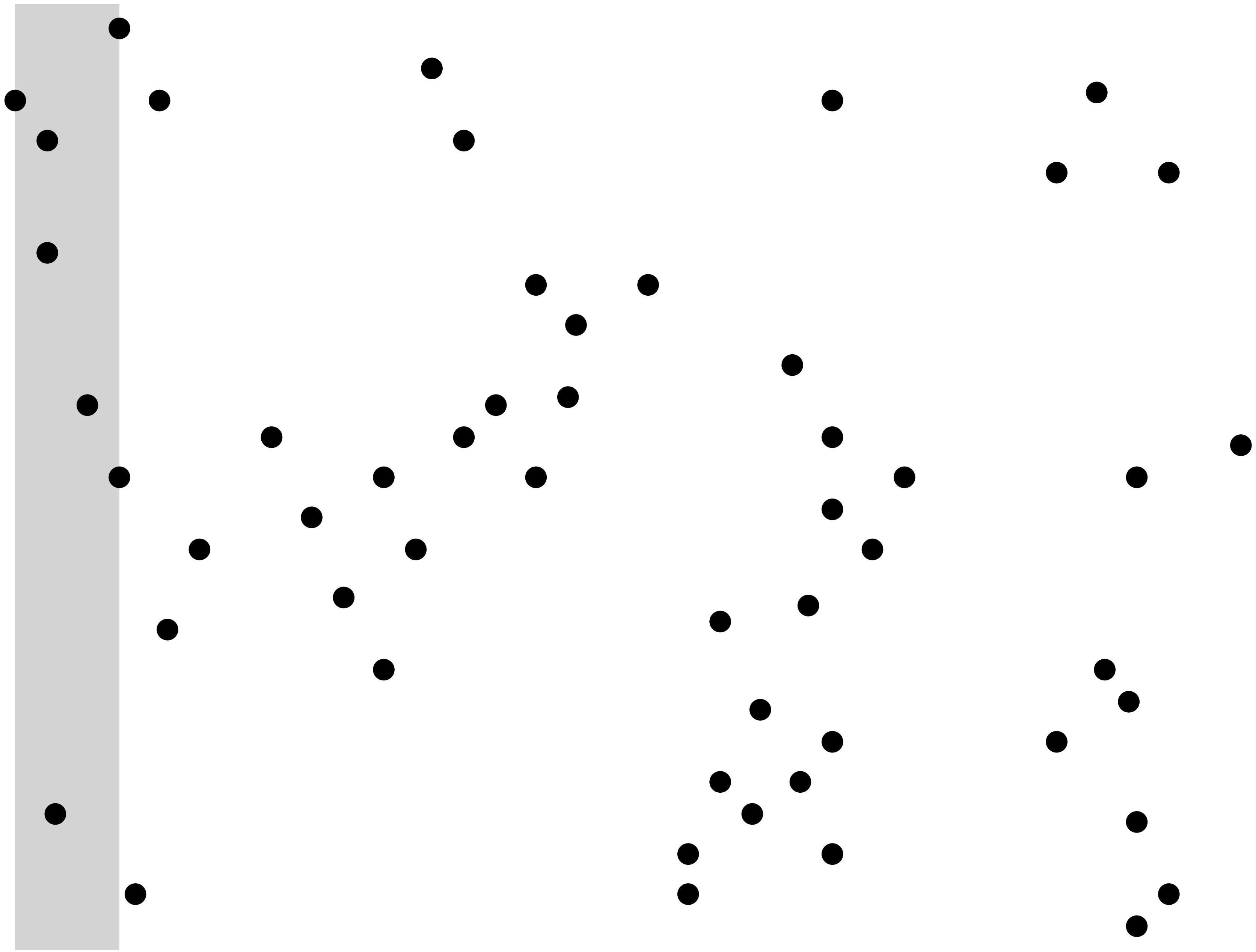
1. Construct boxes

Add points as long as strip width $\leqslant \varepsilon/\sqrt{2}$.



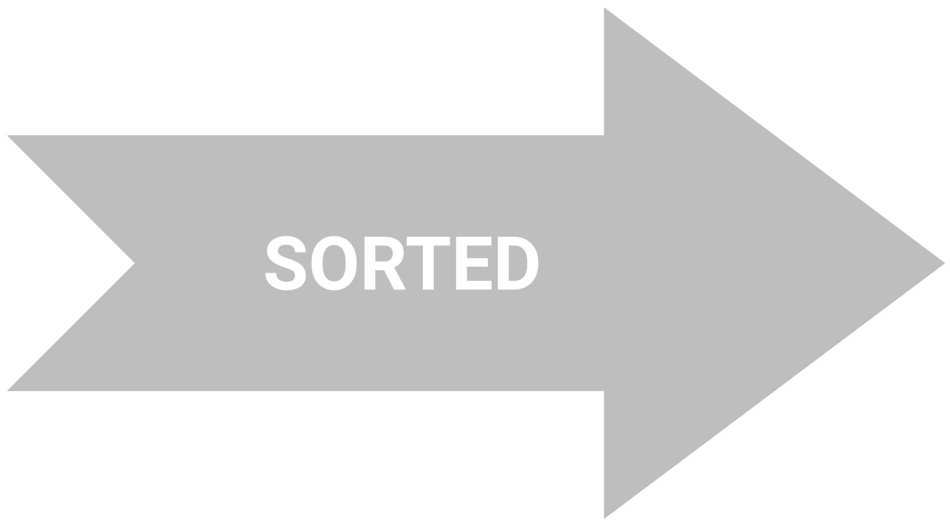
Box graph $\mathcal{G}_{box}$





1. Construct boxes

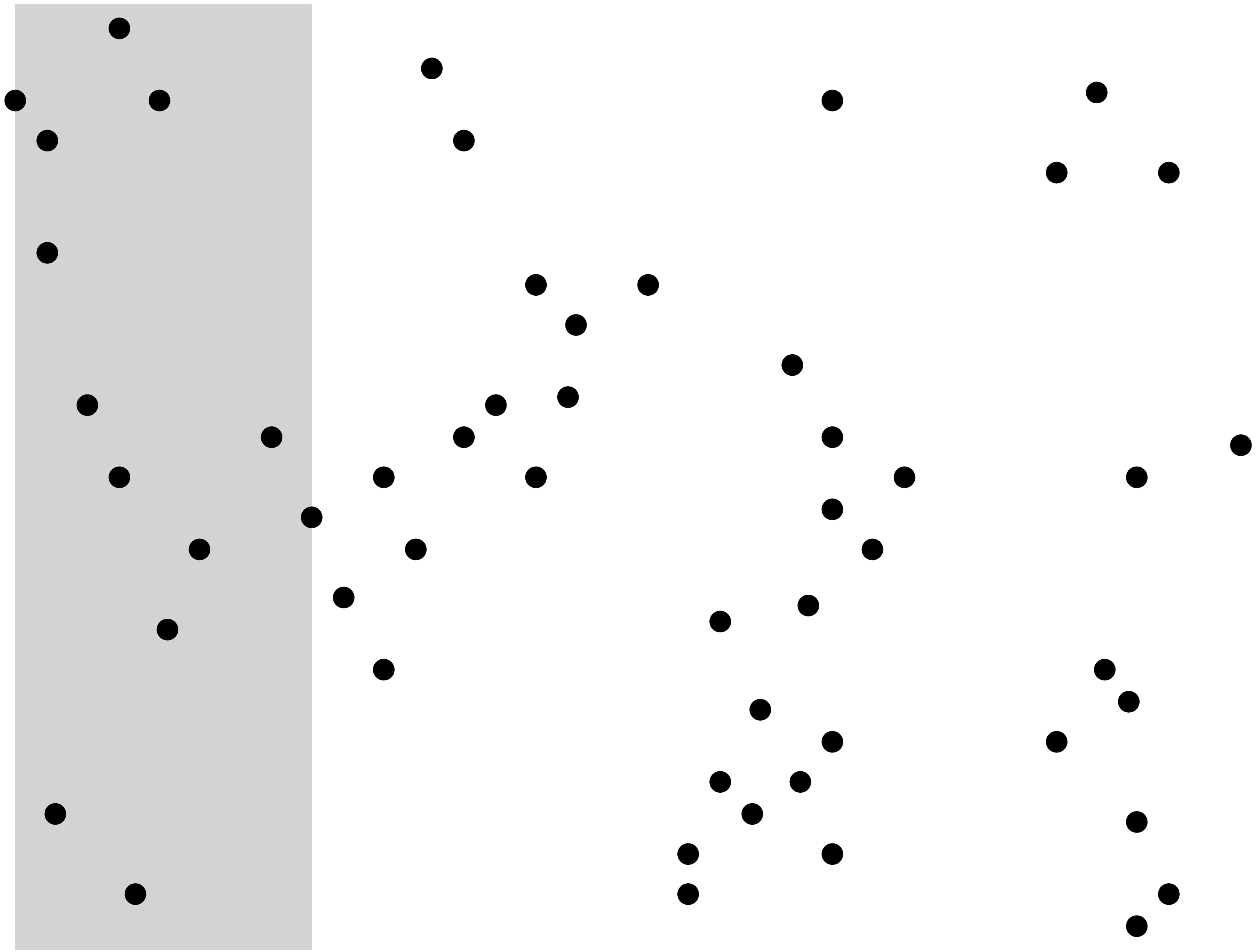
Add points as long as strip width $\leqslant \varepsilon/\sqrt{2}$.



Box graph $\mathcal{G}_{\text{box}}$

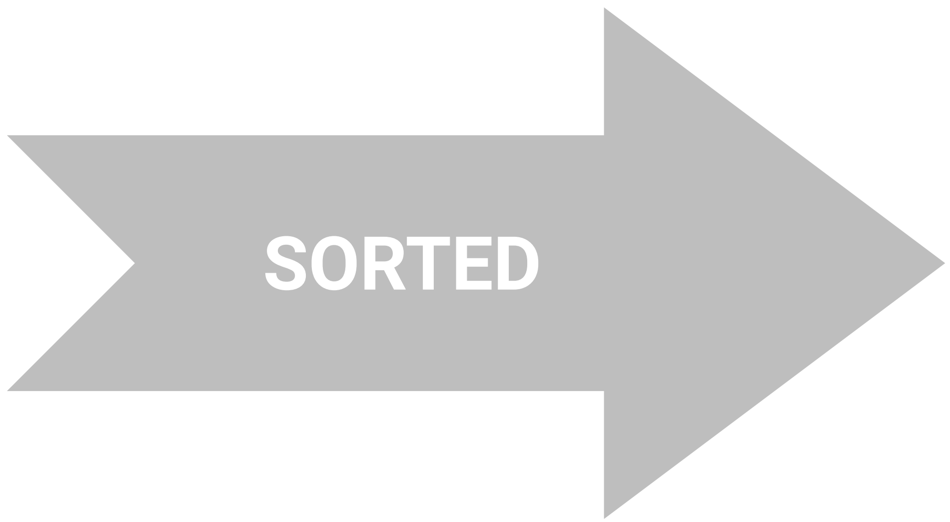
ε:

$\varepsilon/\sqrt{2}$:

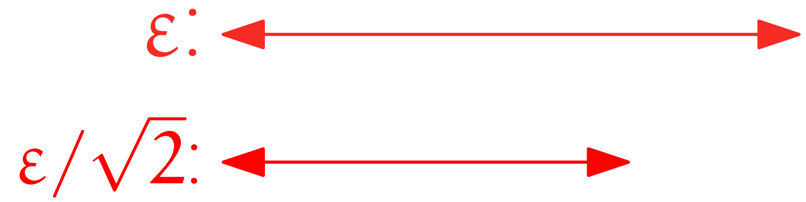


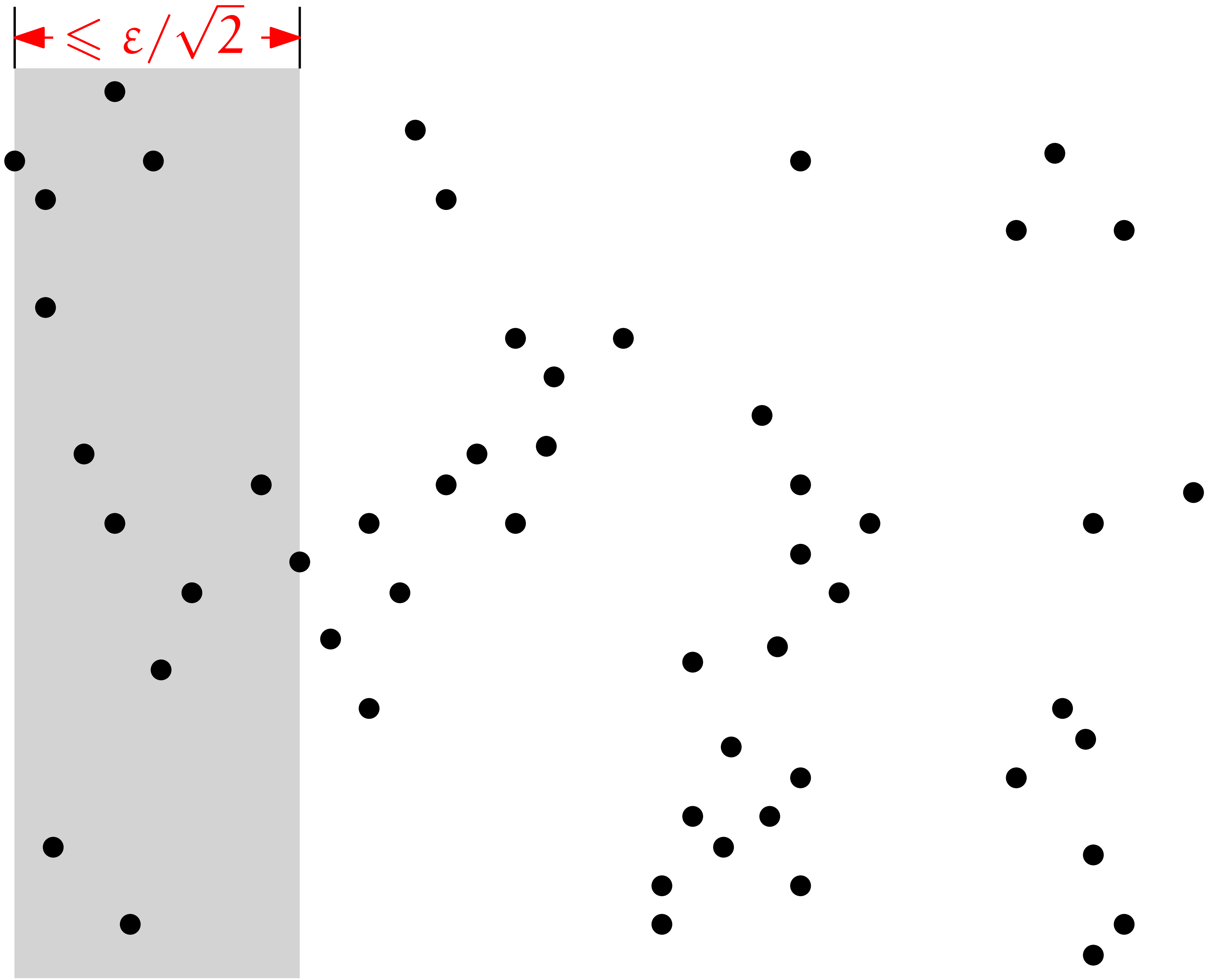
1. Construct boxes

Add points as long as strip width $\leqslant \varepsilon/\sqrt{2}$.



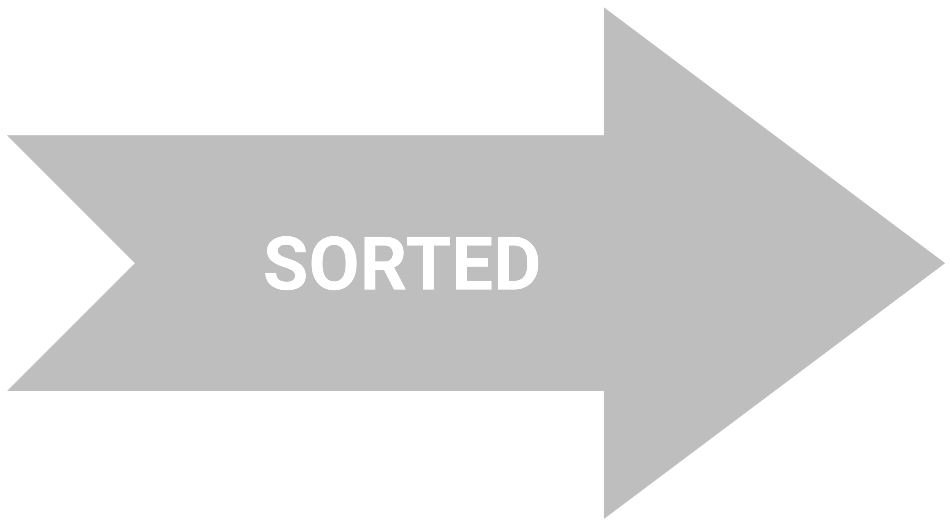
Box graph $\mathcal{G}_{\text{box}}$





1. Construct boxes

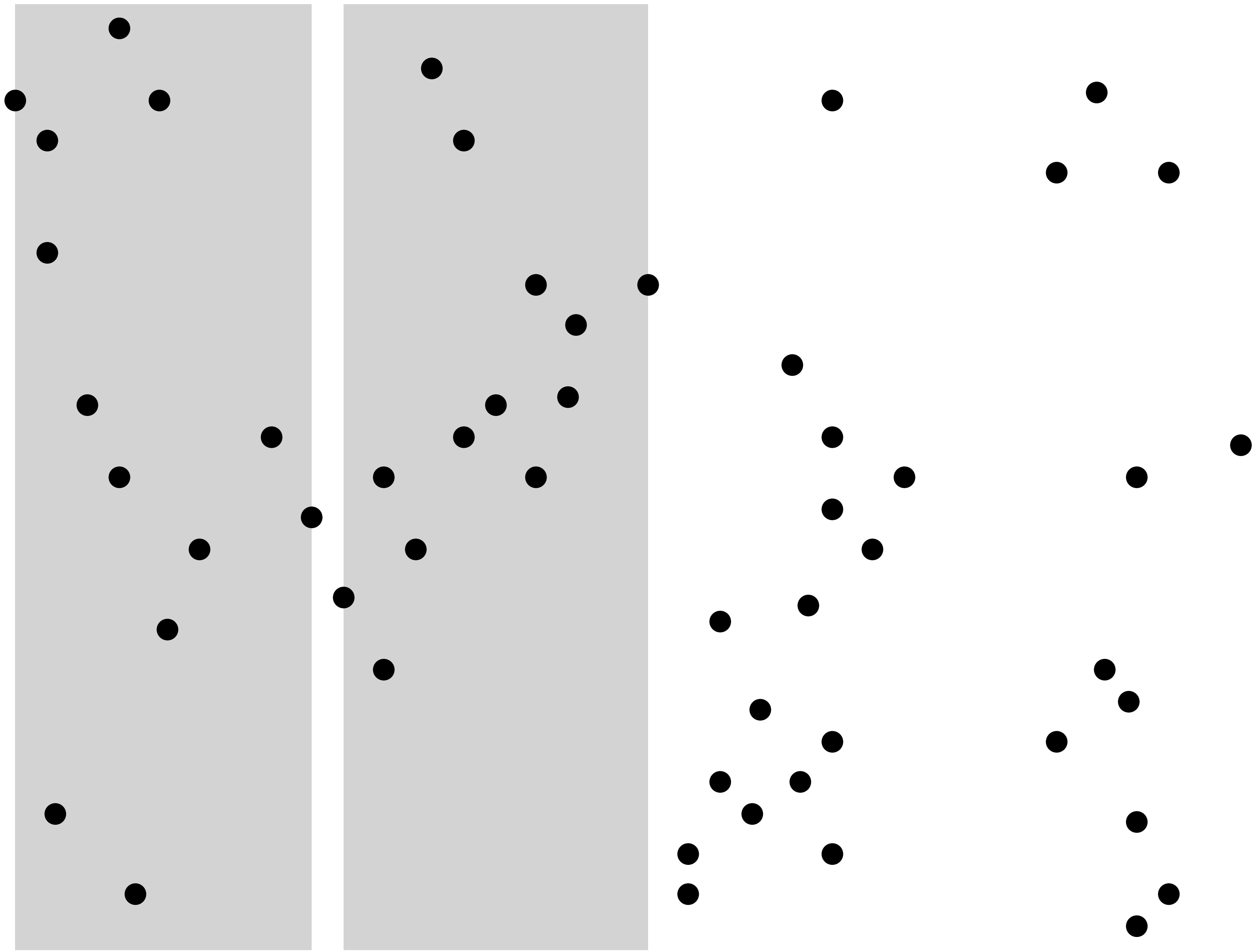
Add points as long as strip width $\leqslant \varepsilon/\sqrt{2}$.



ε:

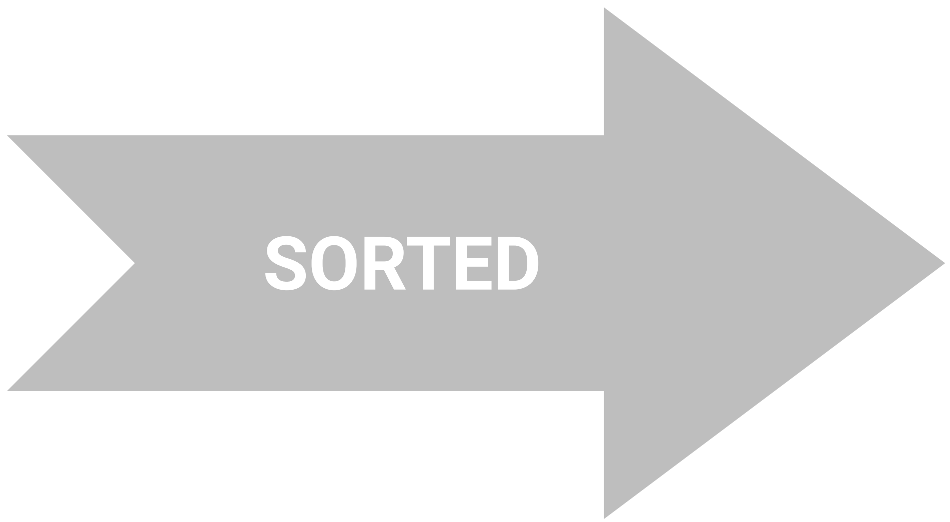
$\varepsilon/\sqrt{2}$:

Box graph $\mathcal{G}_{\text{box}}$

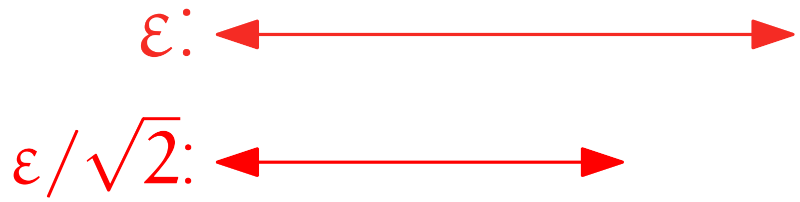


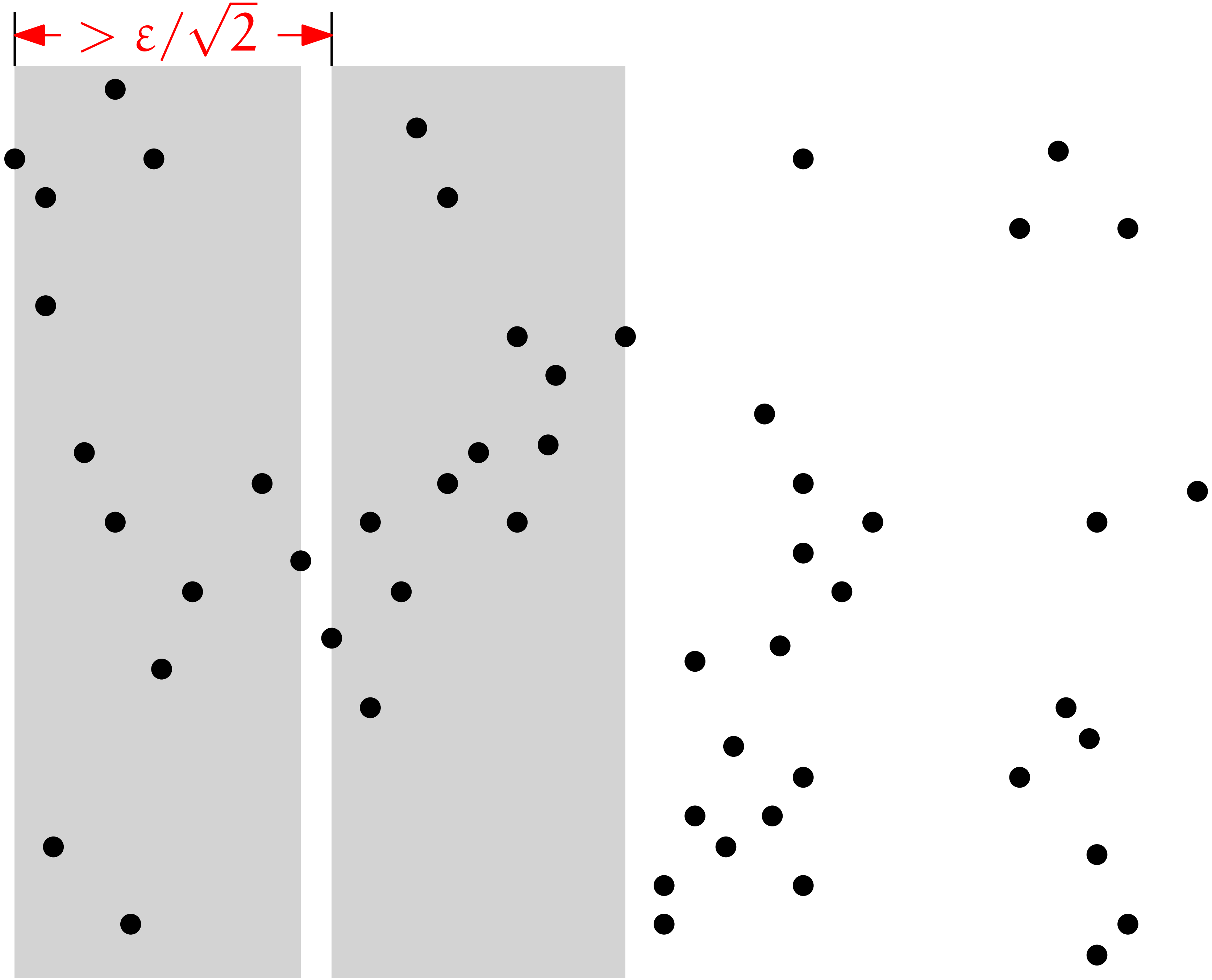
1. Construct boxes

Add points as long as strip width $\leqslant \varepsilon/\sqrt{2}$.



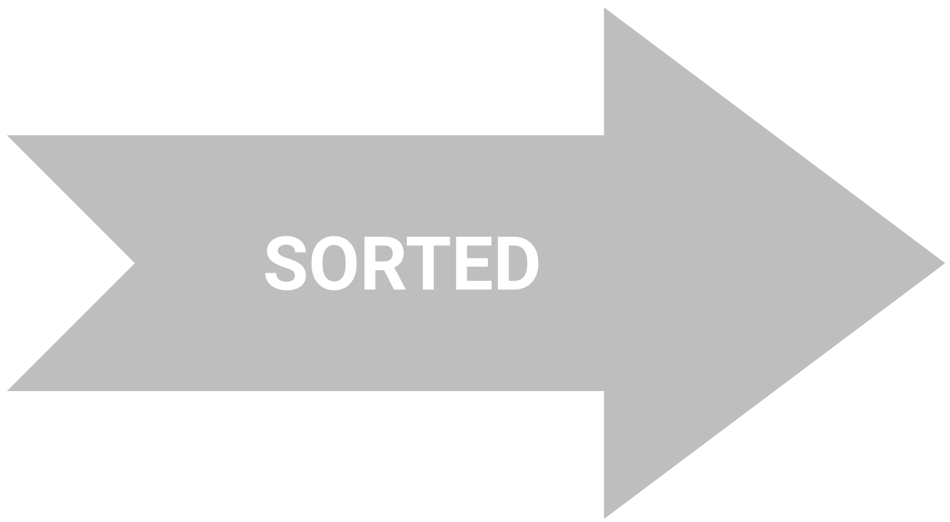
Box graph $\mathcal{G}_{\text{box}}$





1. Construct boxes

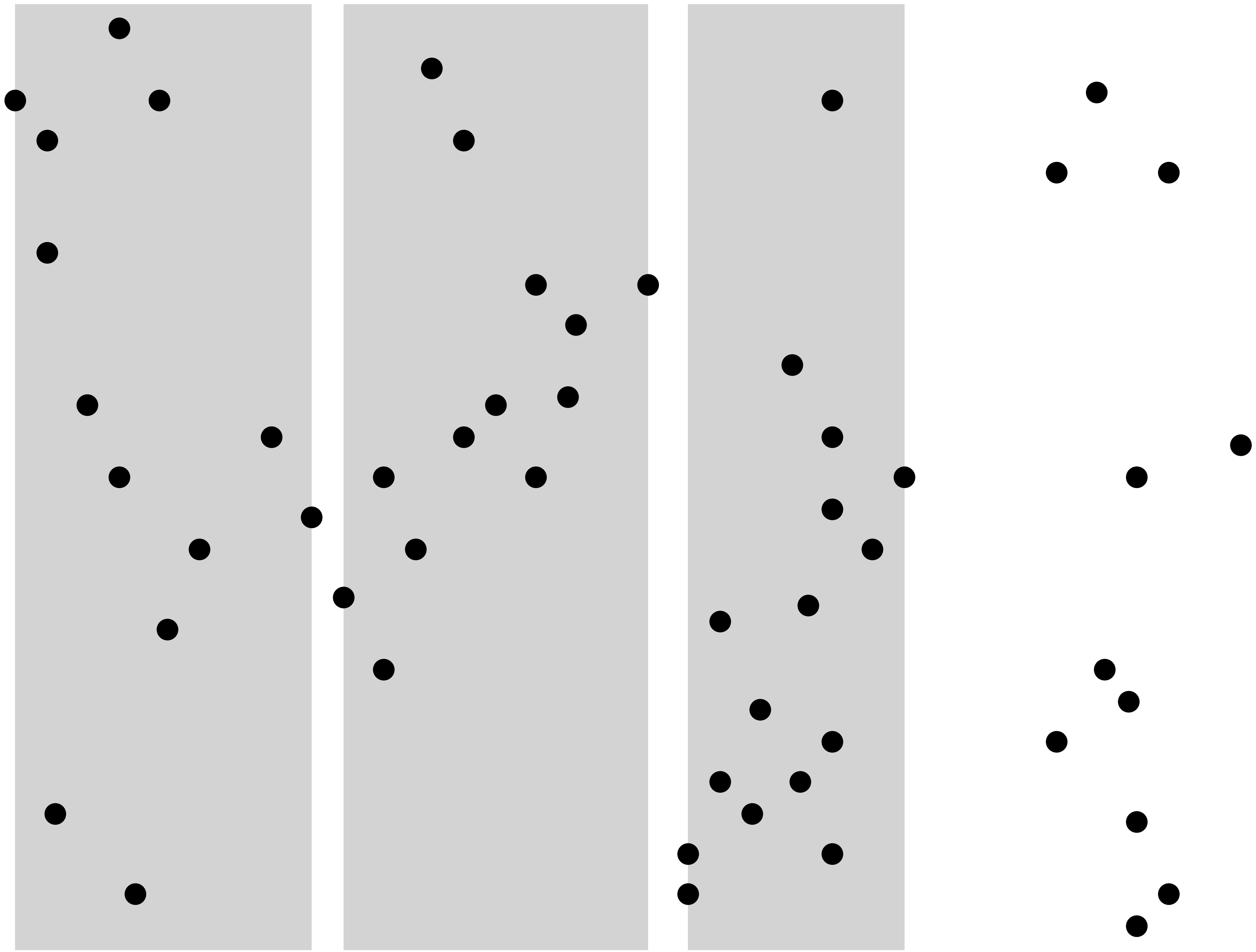
Add points as long as strip width $\leqslant \varepsilon/\sqrt{2}$.



Box graph $\mathcal{G}_{\text{box}}$

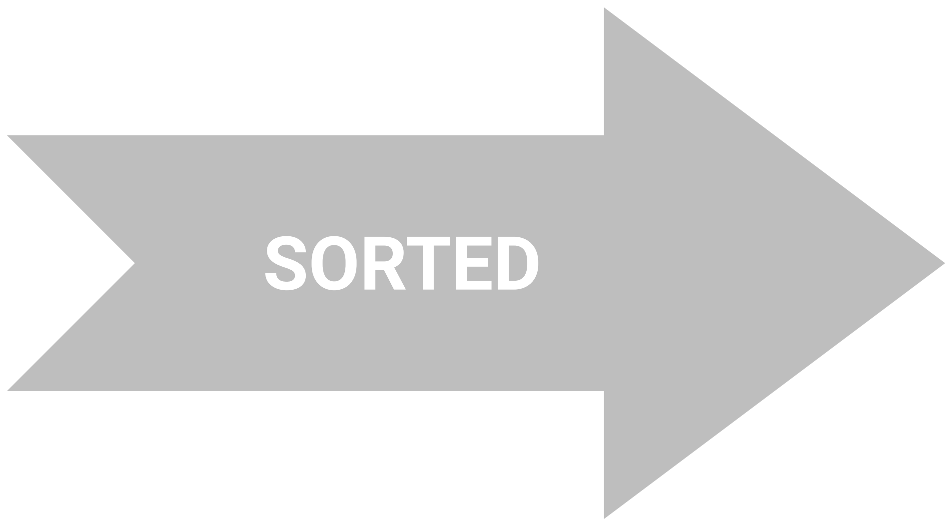
ε:

$\varepsilon/\sqrt{2}$:



1. Construct boxes

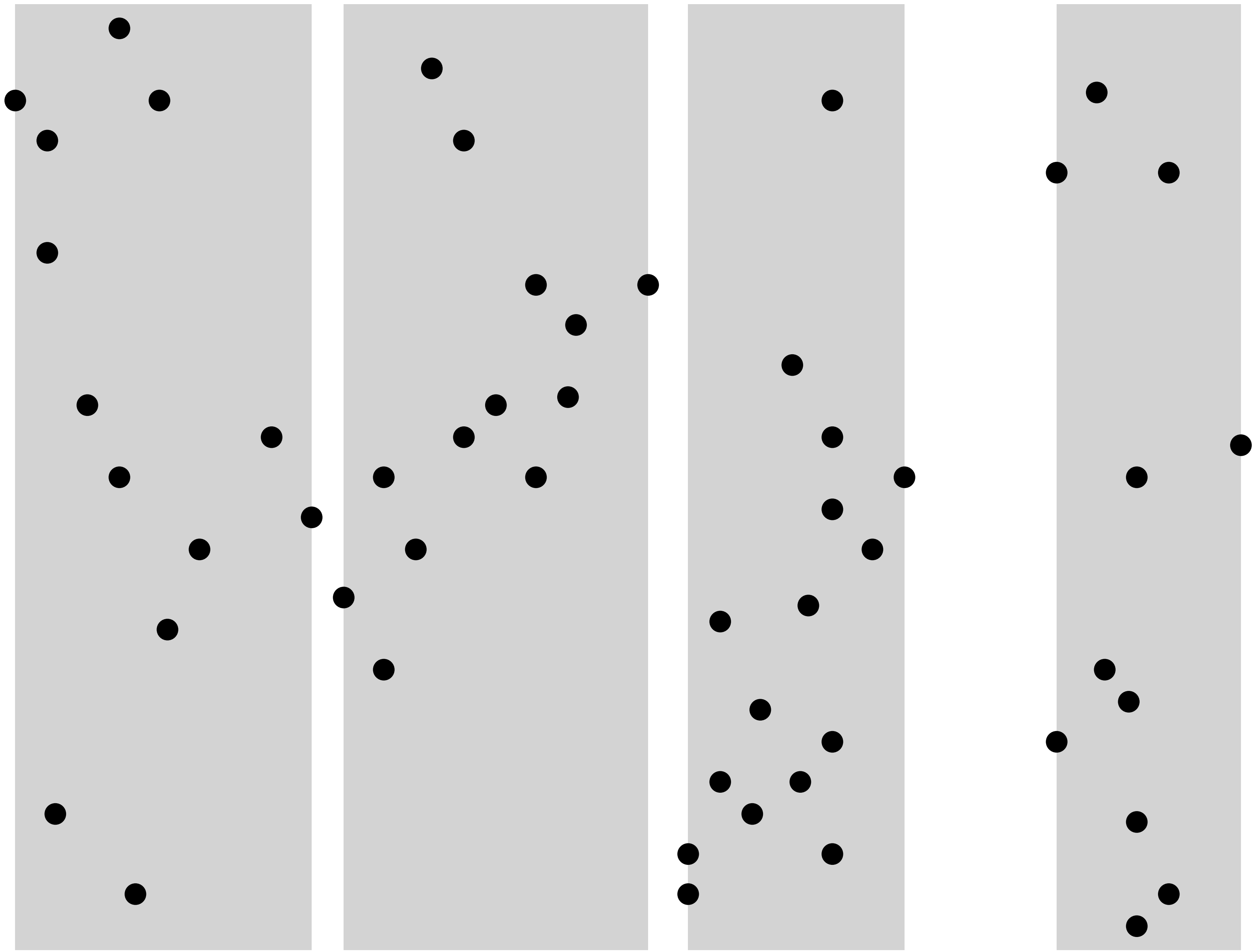
Add points as long as strip width $\leqslant \varepsilon/\sqrt{2}$.



Box graph $\mathcal{G}_{\mathrm{box}}$

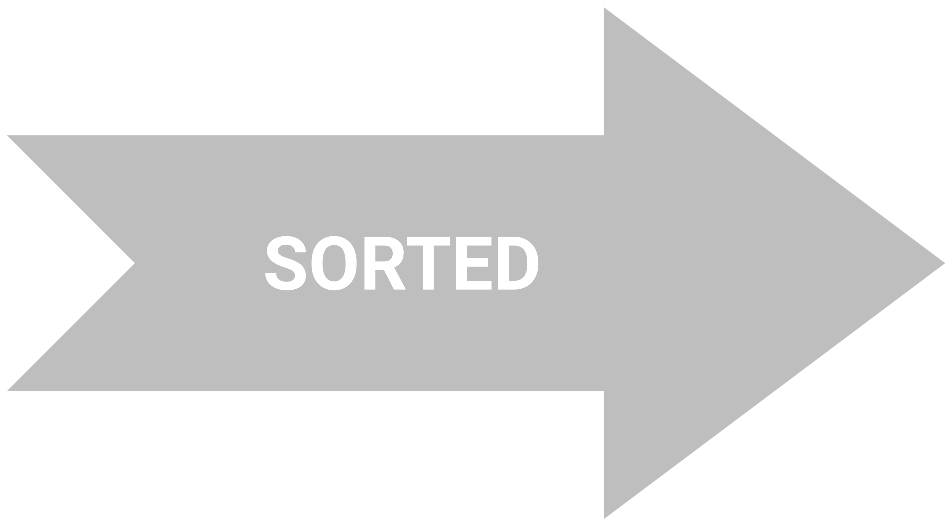
ε:

$\varepsilon/\sqrt{2}$:

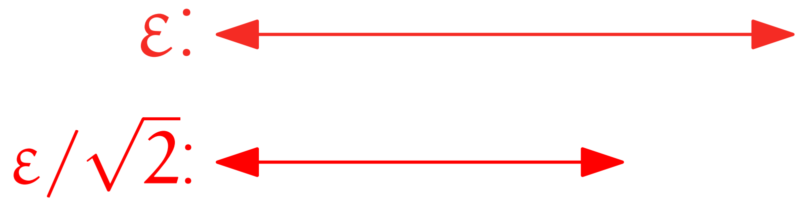


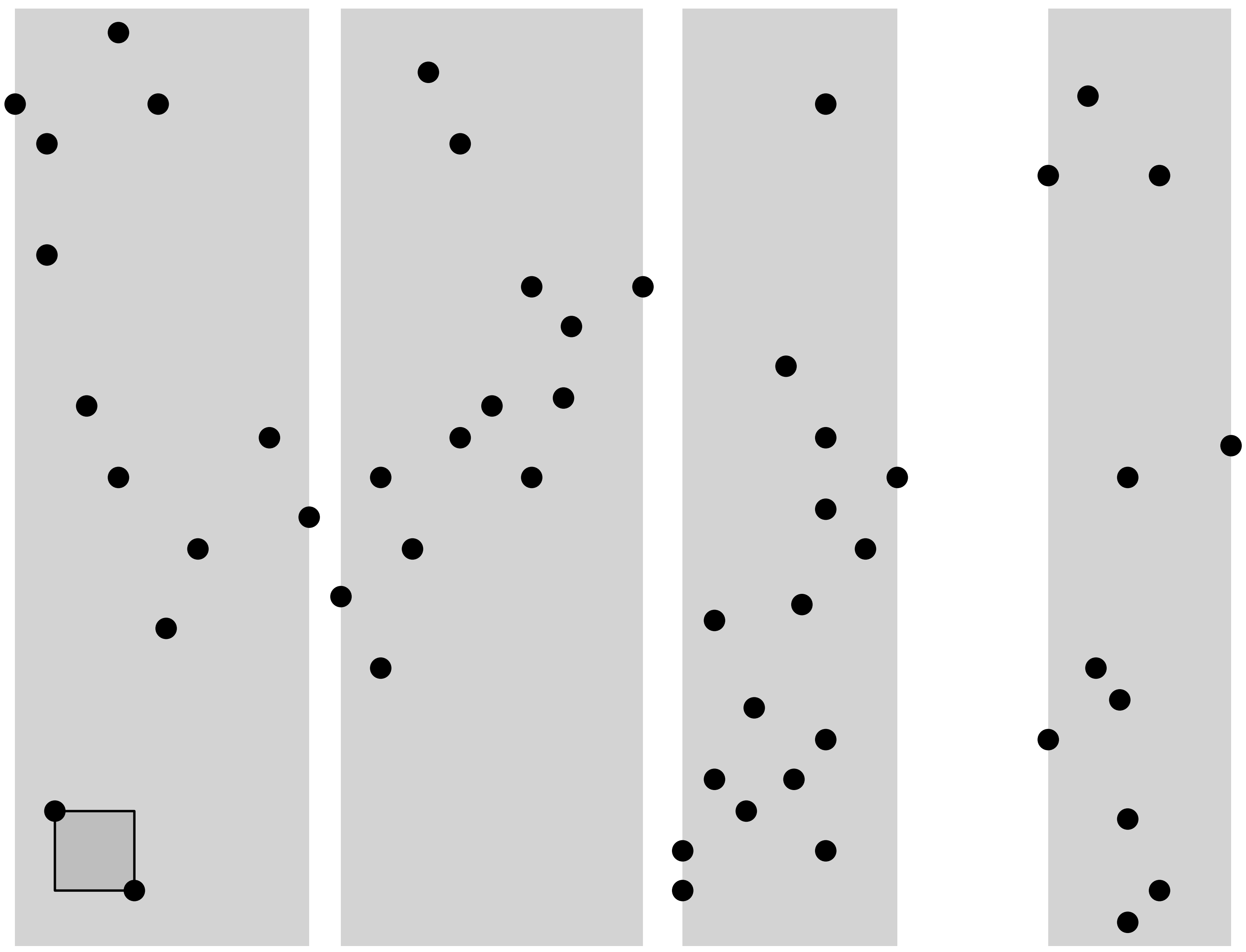
1. Construct boxes

Add points as long as strip width $\leqslant \varepsilon/\sqrt{2}$.



Box graph $\mathcal{G}_{\text{box}}$

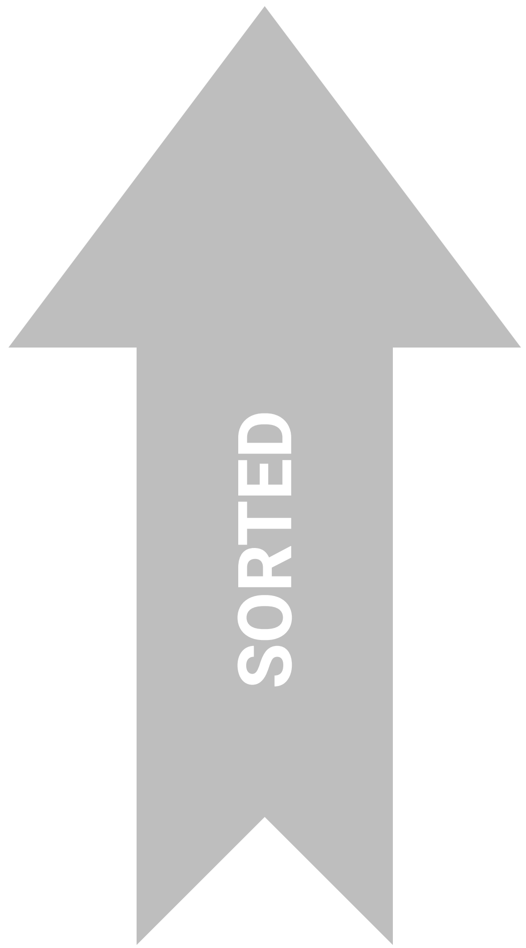




1. Construct boxes

Add points as long as strip width $\leqslant \varepsilon/\sqrt{2}$.

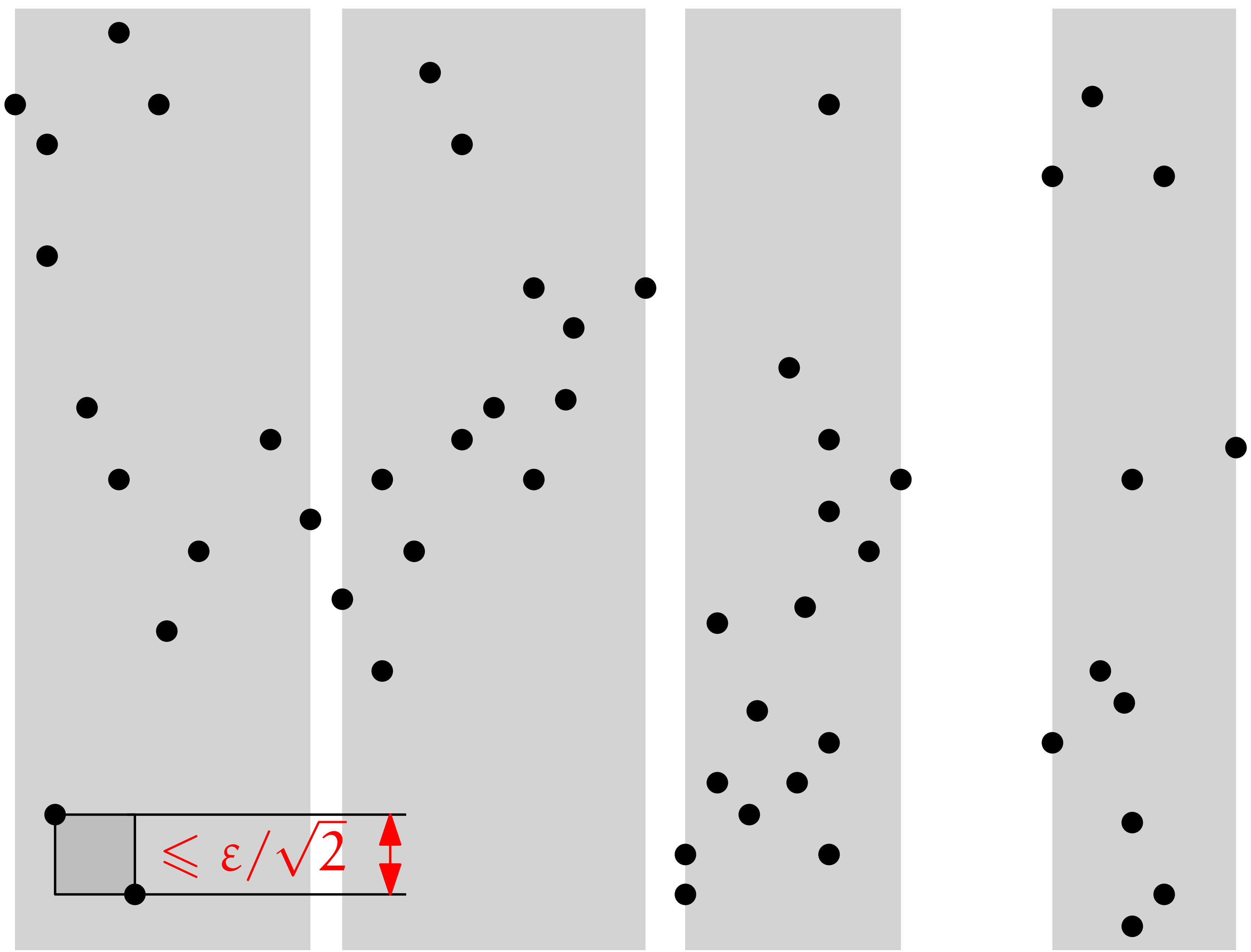
Per strip: add points to box as long as height $\leqslant \varepsilon/\sqrt{2}$.



Box graph $\mathcal{G}_{\text{box}}$

ε: ⟷

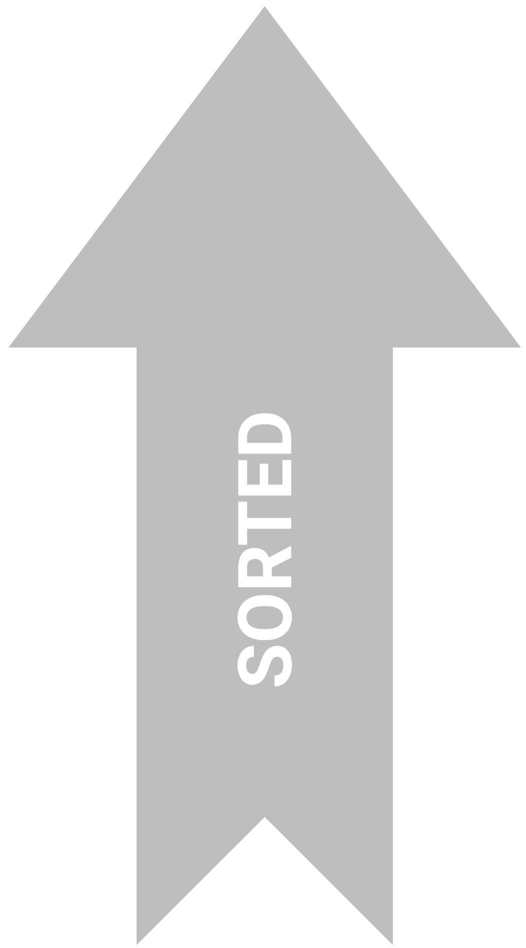
$\varepsilon/\sqrt{2}$: ⟷



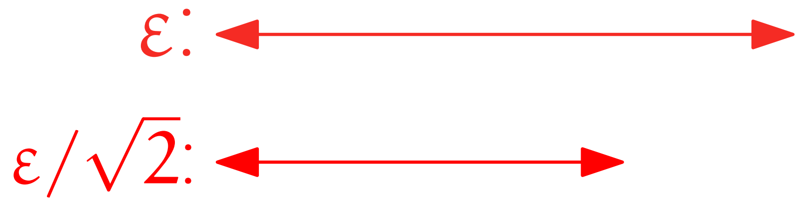
1. Construct boxes

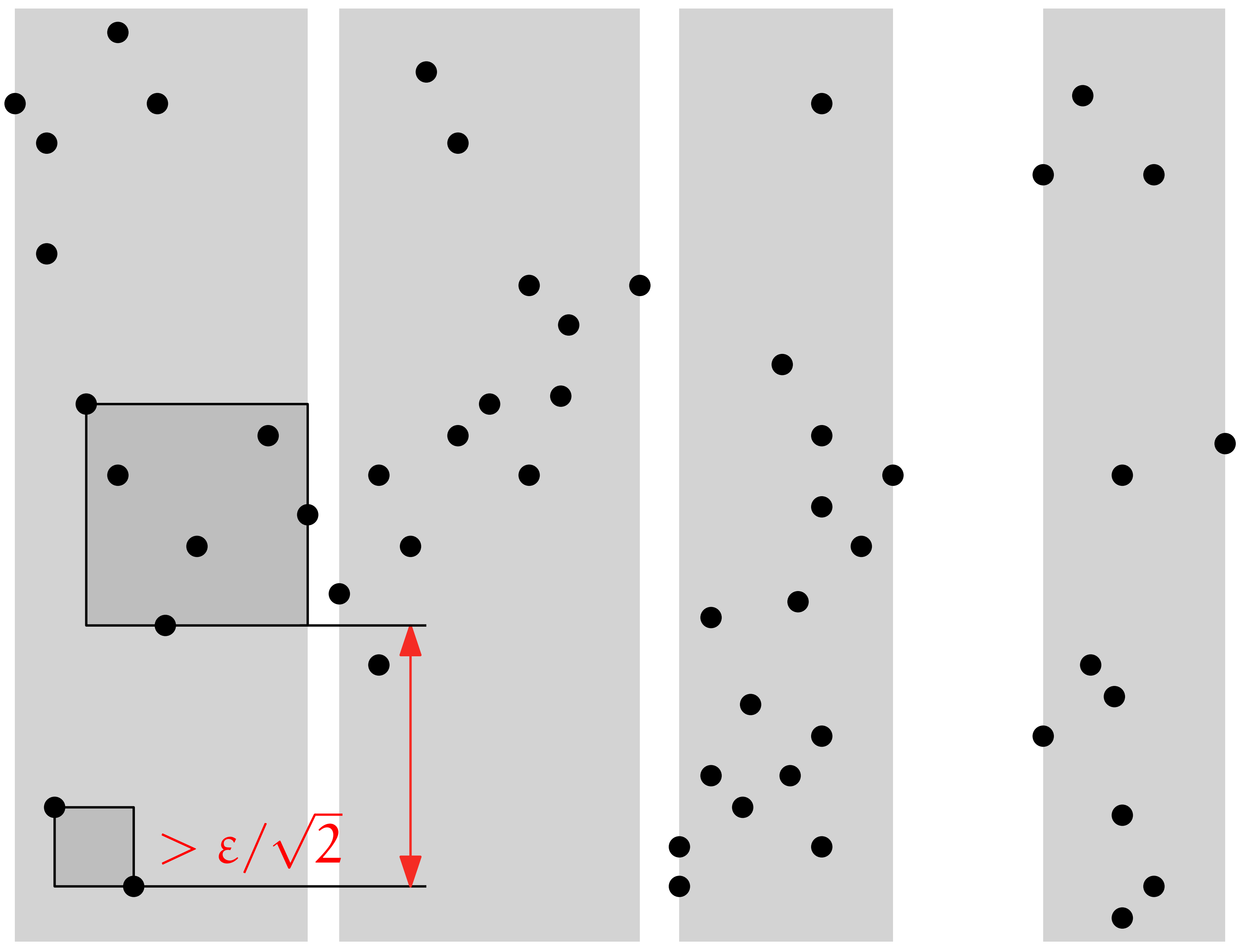
Add points as long as strip width $\leqslant \varepsilon/\sqrt{2}$.

Per strip: add points to box as long as height $\leqslant \varepsilon/\sqrt{2}$.



Box graph $\mathcal{G}_{box}$

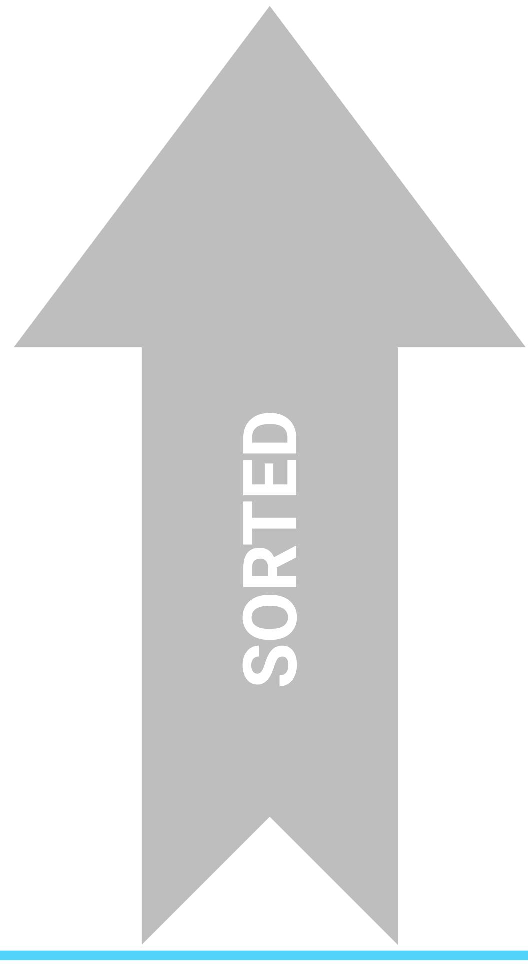




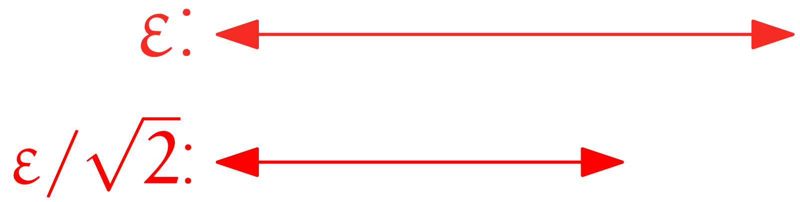
1. Construct boxes

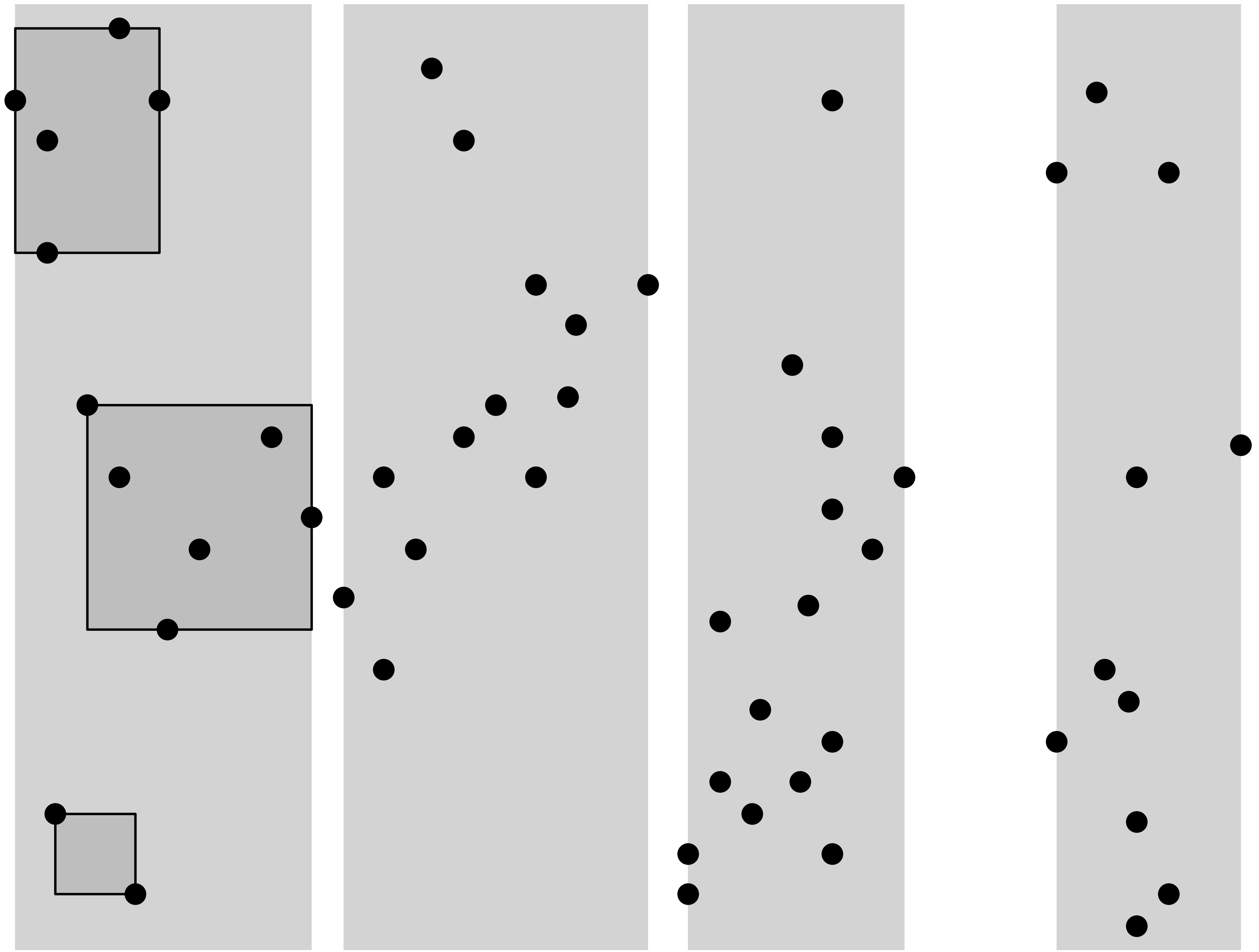
Add points as long as strip width $\leqslant \varepsilon/\sqrt{2}$.

Per strip: add points to box as long as height $\leqslant \varepsilon/\sqrt{2}$.



Box graph $\mathcal{G}_{\text{box}}$

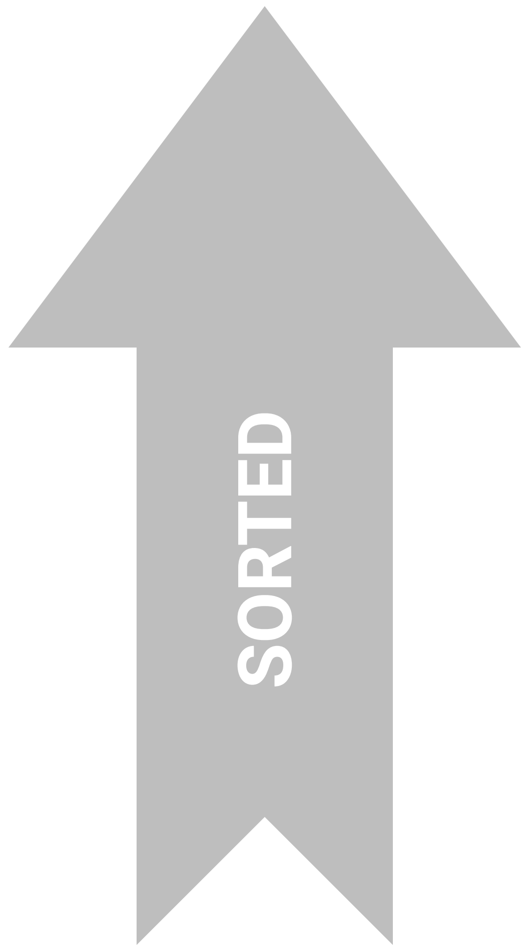


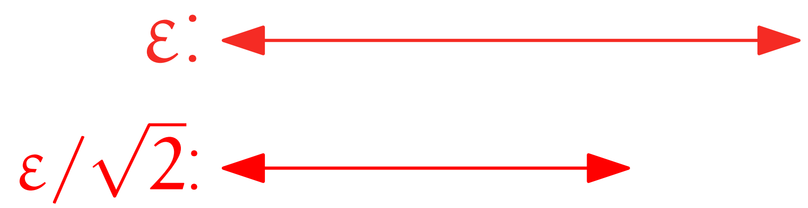


1. Construct boxes

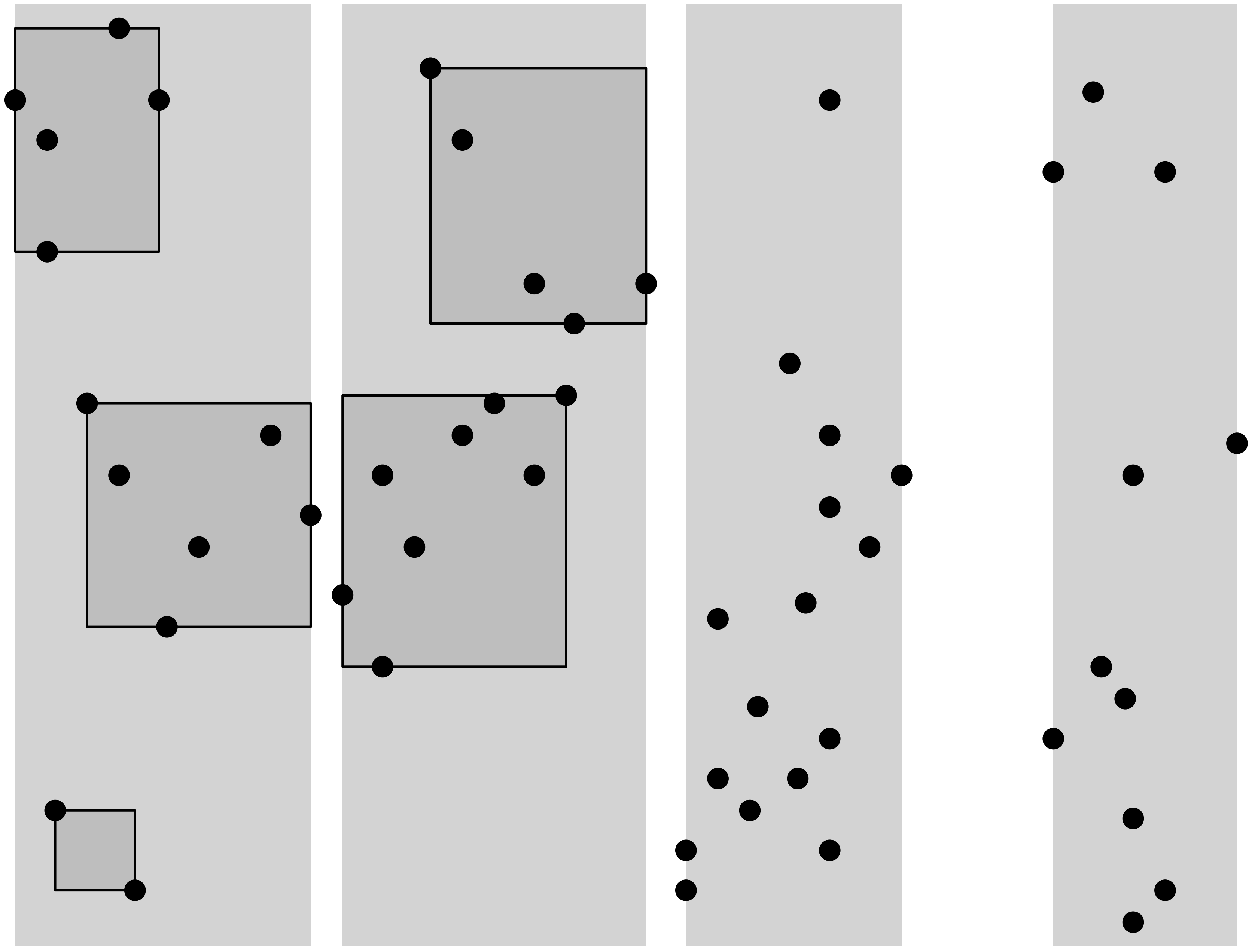
Add points as long as strip width $\leqslant \varepsilon/\sqrt{2}$.

Per strip: add points to box as long as height $\leqslant \varepsilon/\sqrt{2}$.





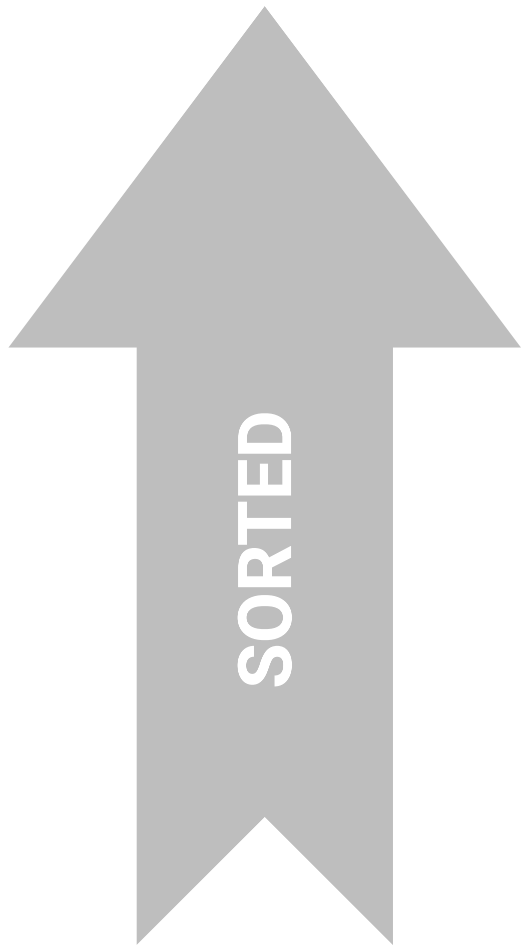
Box graph $\mathcal{G}_{\text{box}}$



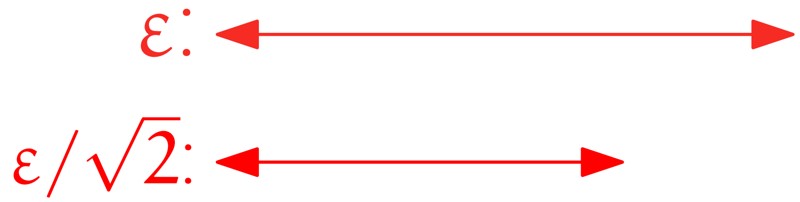
1. Construct boxes

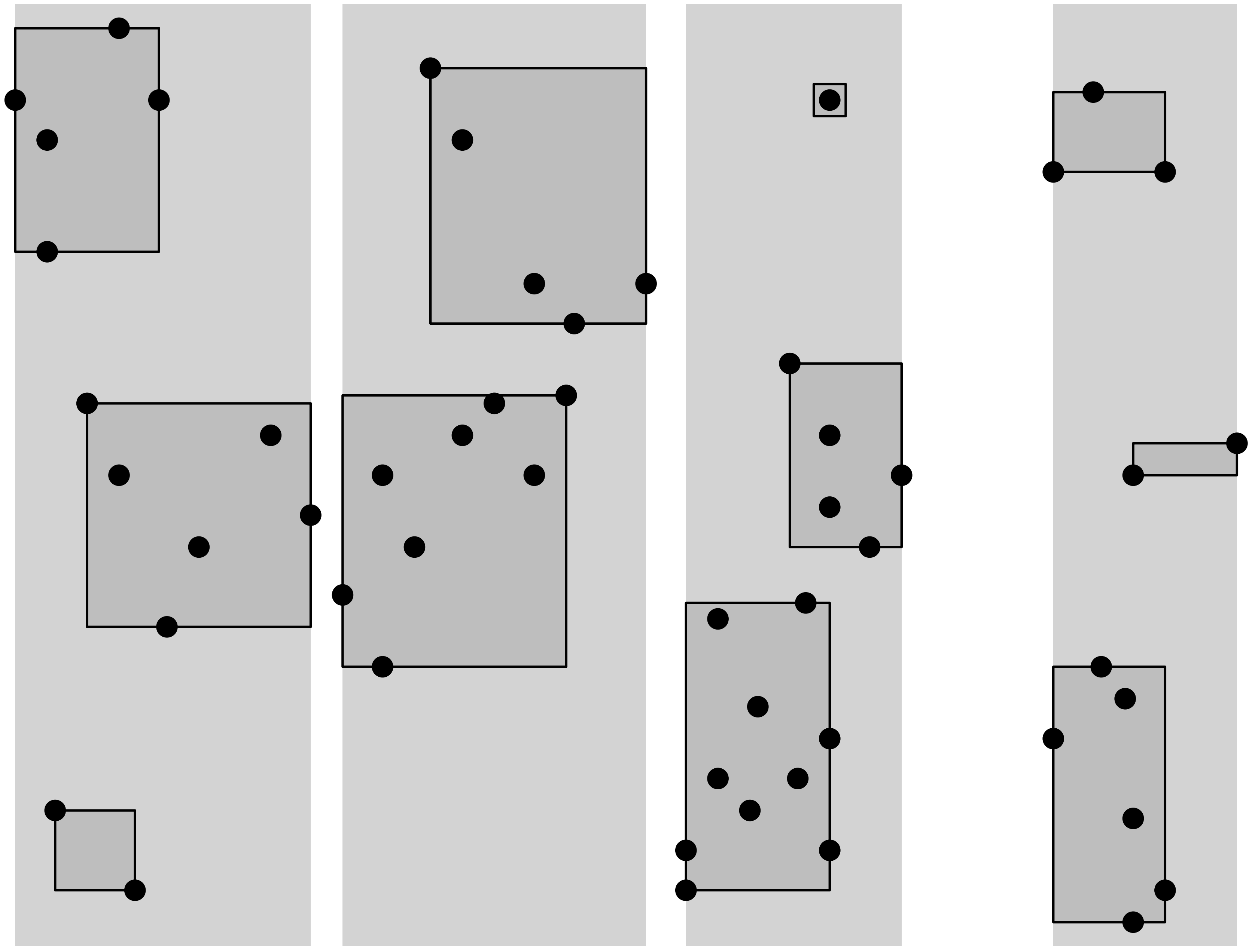
Add points as long as strip width $\leqslant \varepsilon/\sqrt{2}$.

Per strip: add points to box as long as height $\leqslant \varepsilon/\sqrt{2}$.



Box graph $\mathcal{G}_{\text{box}}$

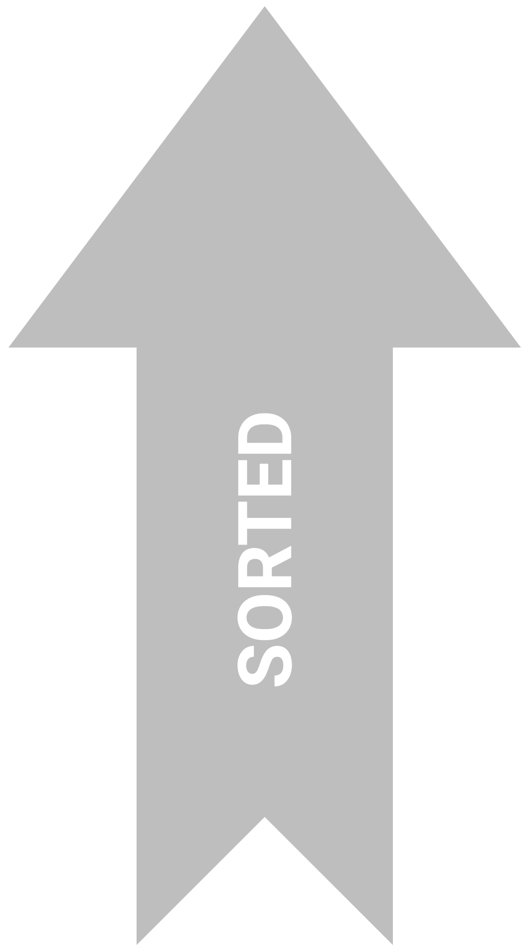




1. Construct boxes

Add points as long as strip width $\leqslant \varepsilon/\sqrt{2}$.

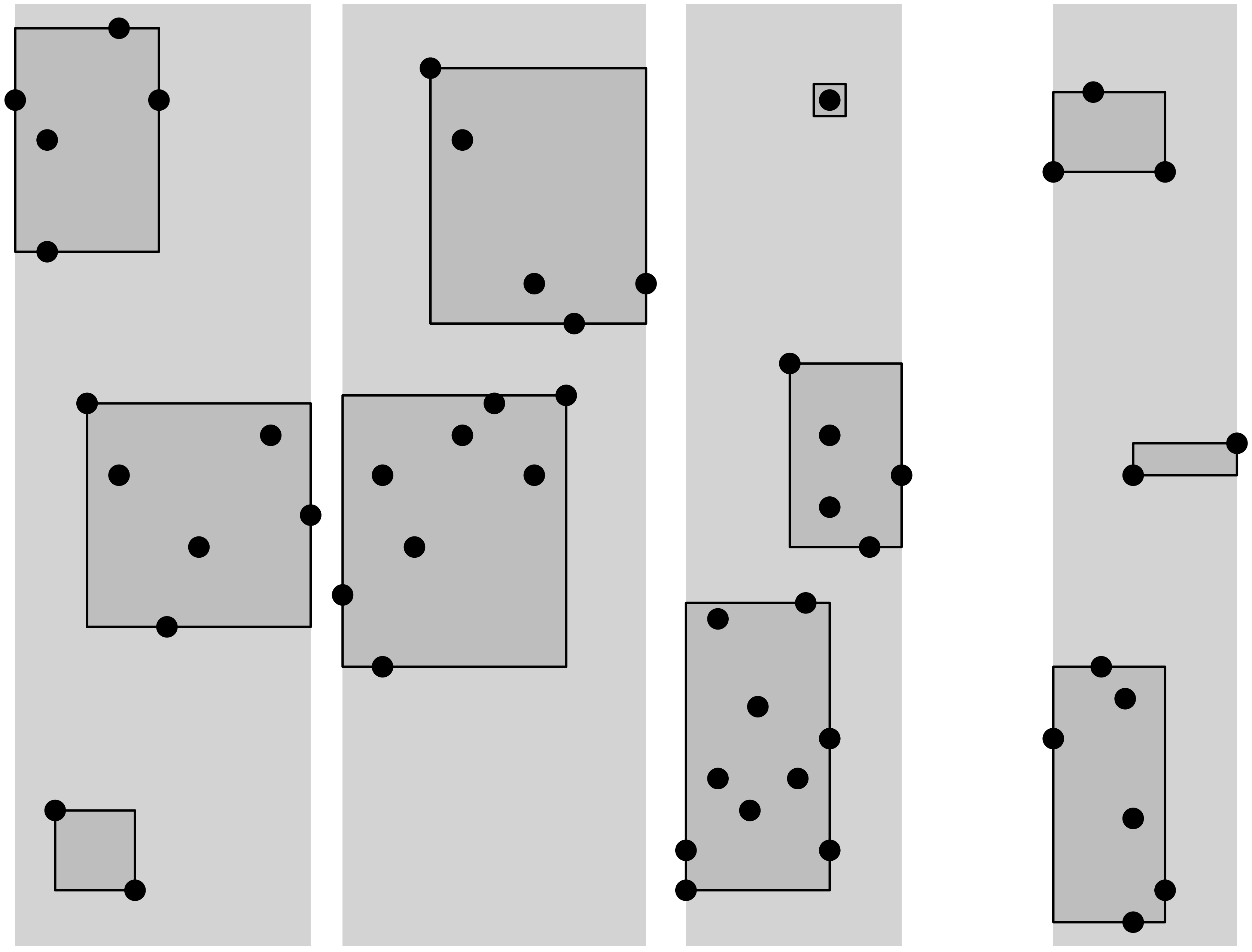
Per strip: add points to box as long as height $\leqslant \varepsilon/\sqrt{2}$.



Box graph $\mathcal{G}_{\text{box}}$

ε:

$\varepsilon/\sqrt{2}$:



1. Construct boxes

Add points as long as strip width $\leqslant \varepsilon/\sqrt{2}$.

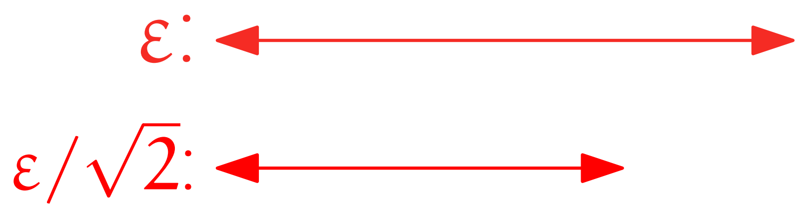
Per strip: add points to box as long as height $\leqslant \varepsilon/\sqrt{2}$.

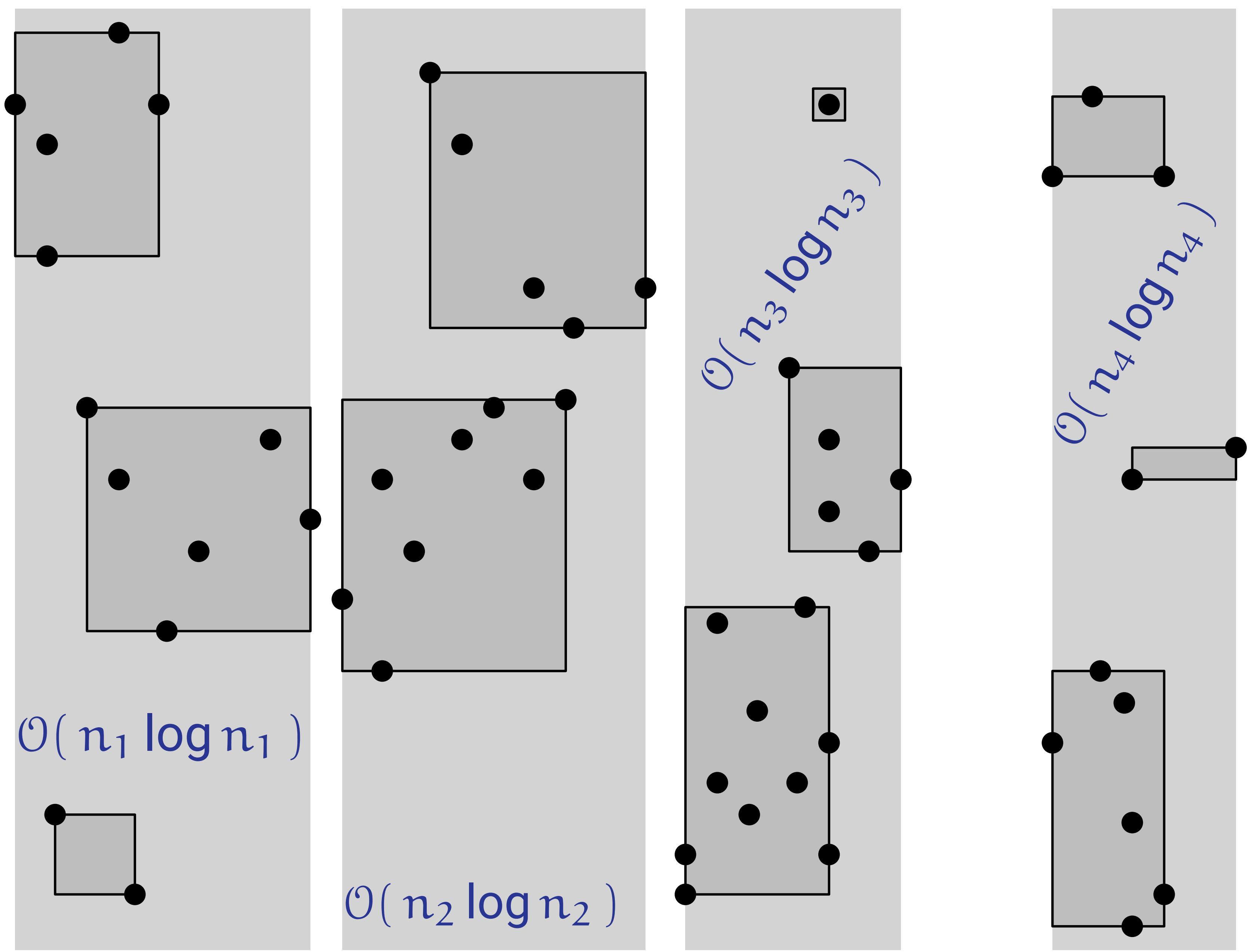
Runtime:

Sort by x

$\mathcal{O}(n \log n)$

Box graph $\mathcal{G}_{\text{box}}$





1. Construct boxes

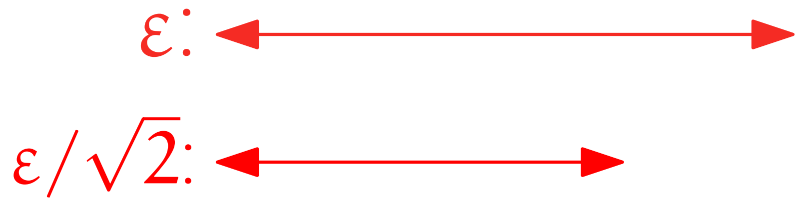
Add points as long as strip width $\leqslant \varepsilon/\sqrt{2}$.

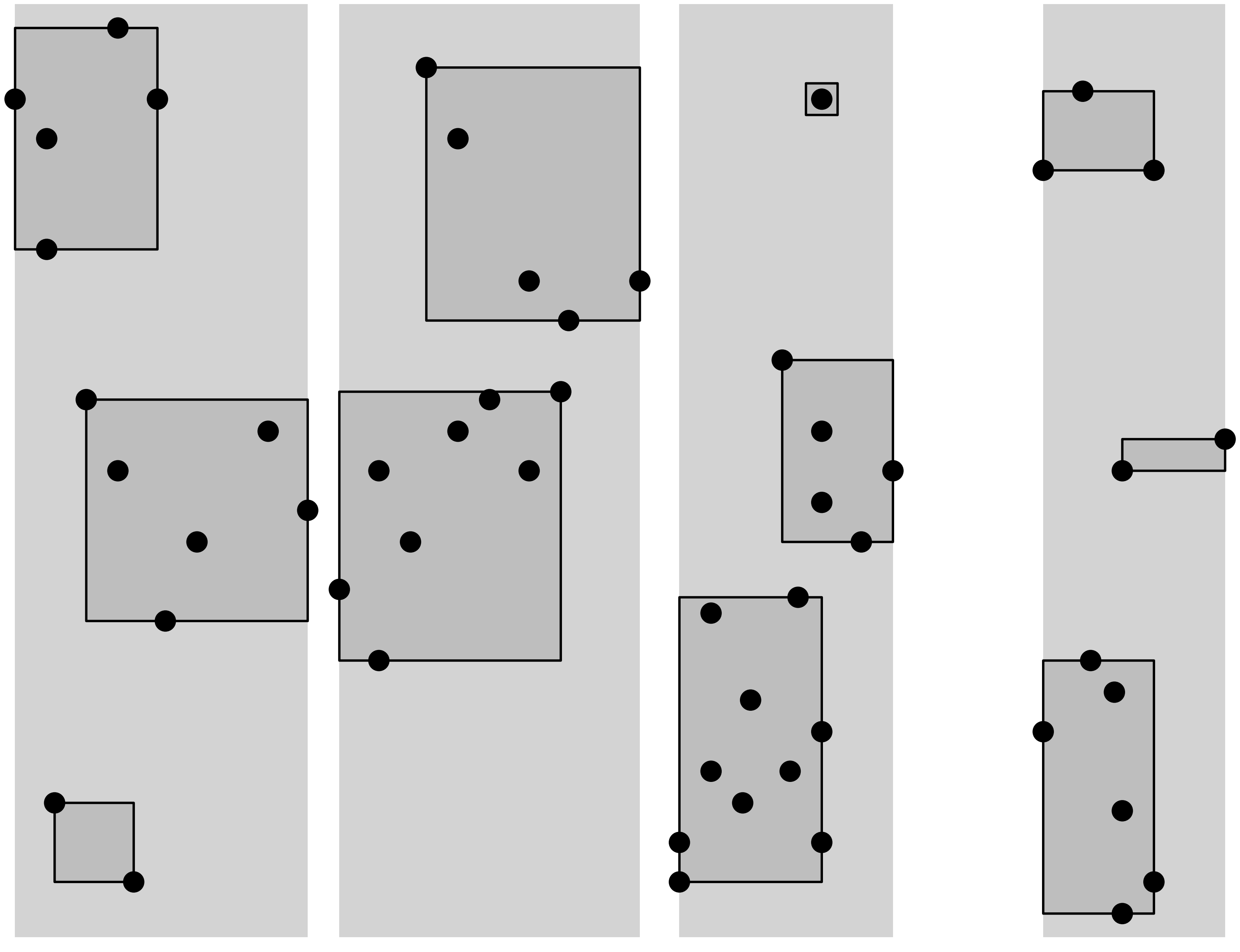
Per strip: add points to box as long as height $\leqslant \varepsilon/\sqrt{2}$.

Runtime:

- Sort by x
  $\mathcal{O}(n \log n)$
- Sort by y per strip
  $\sum_j \mathcal{O}(n_j \log n_j)$
- Total
  $\mathcal{O}(n \log n)$

Box graph $\mathcal{G}_{\text{box}}$

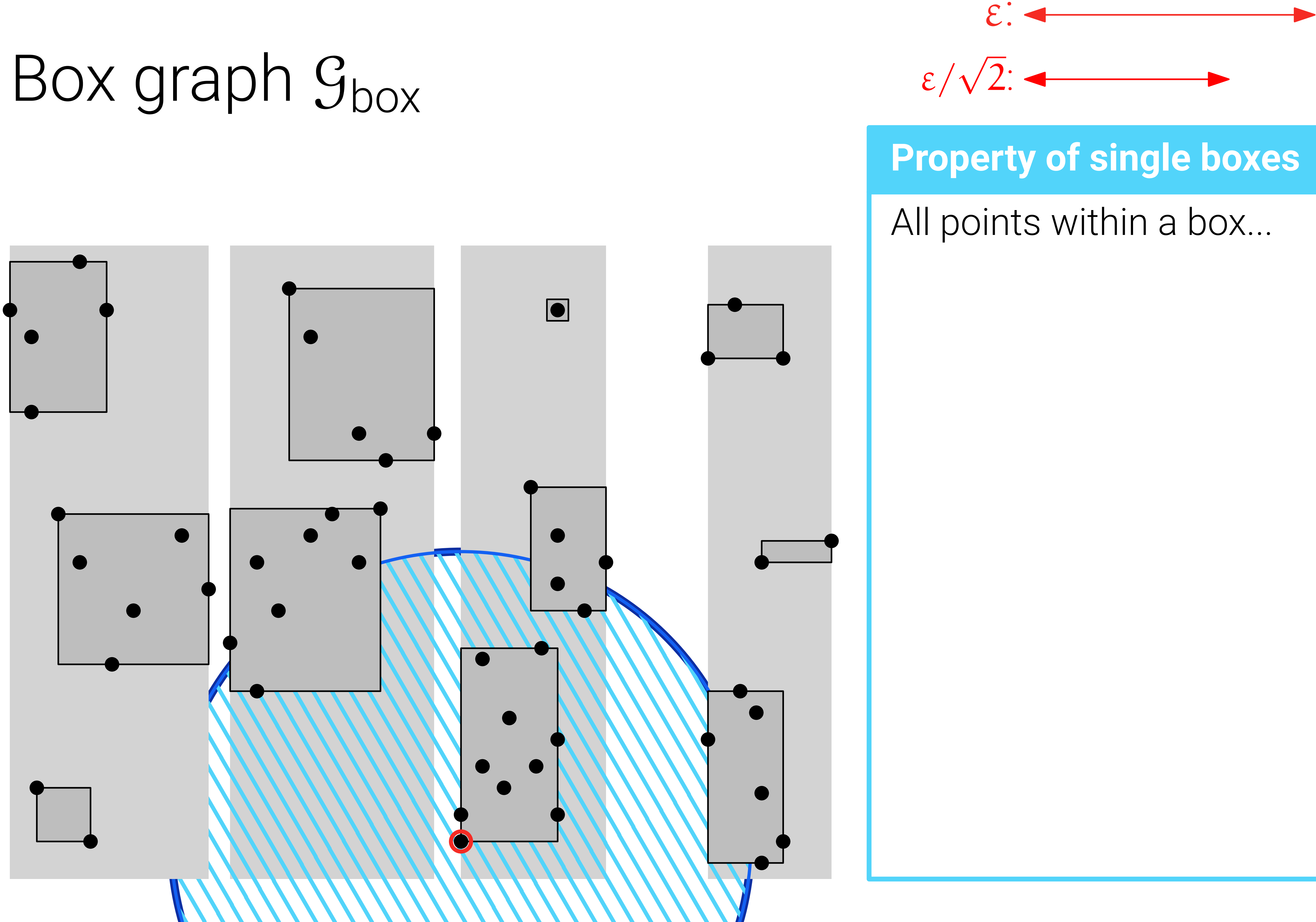




Property of single boxes

All points within a box...

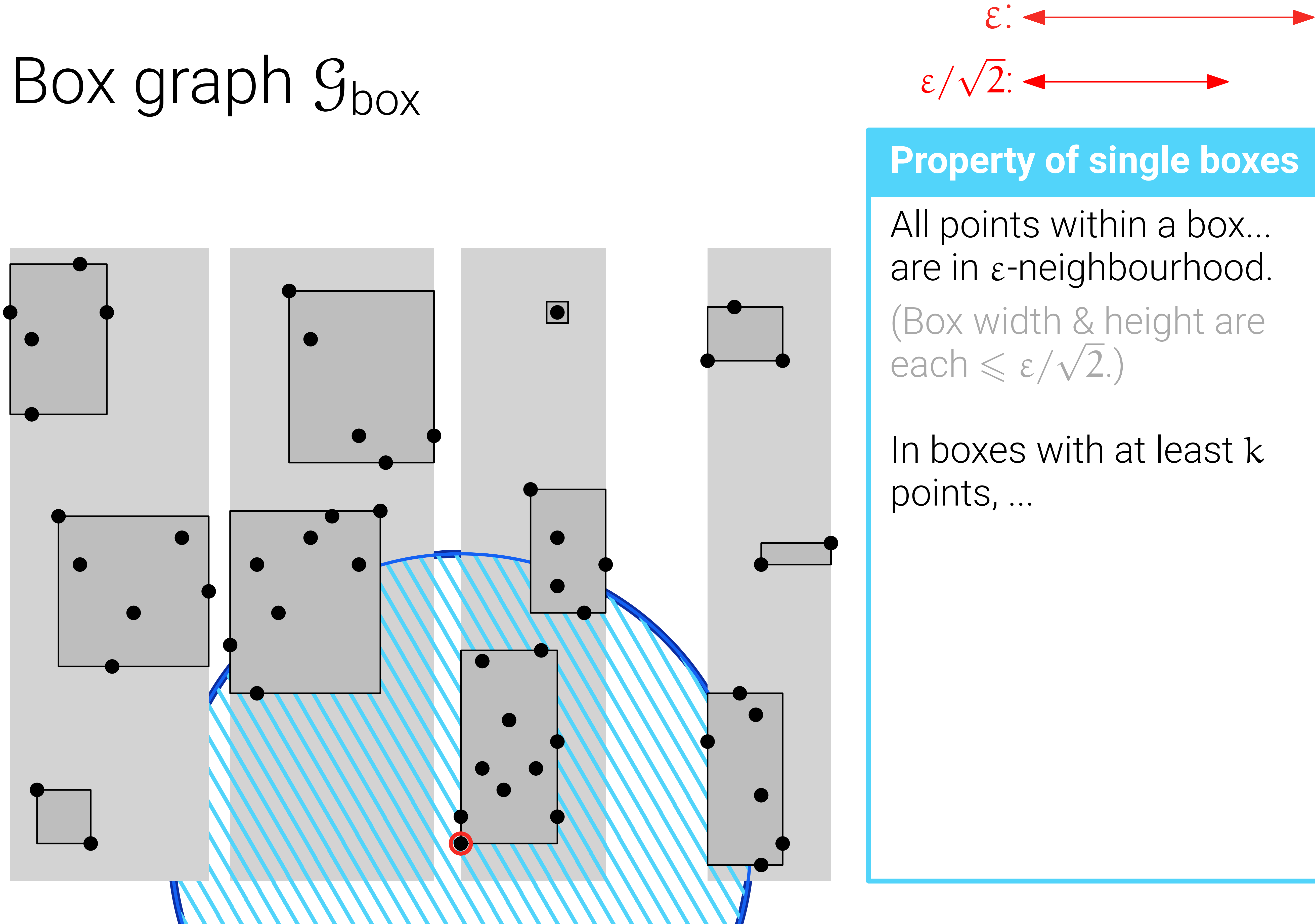
Box graph $\mathcal{G}_{box}$



Property of single boxes

All points within a box...

Box graph $\mathcal{G}_{\text{box}}$



Property of single boxes

All points within a box... are in ε-neighbourhood.

(Box width & height are each $\leqslant \varepsilon/\sqrt{2}$.)

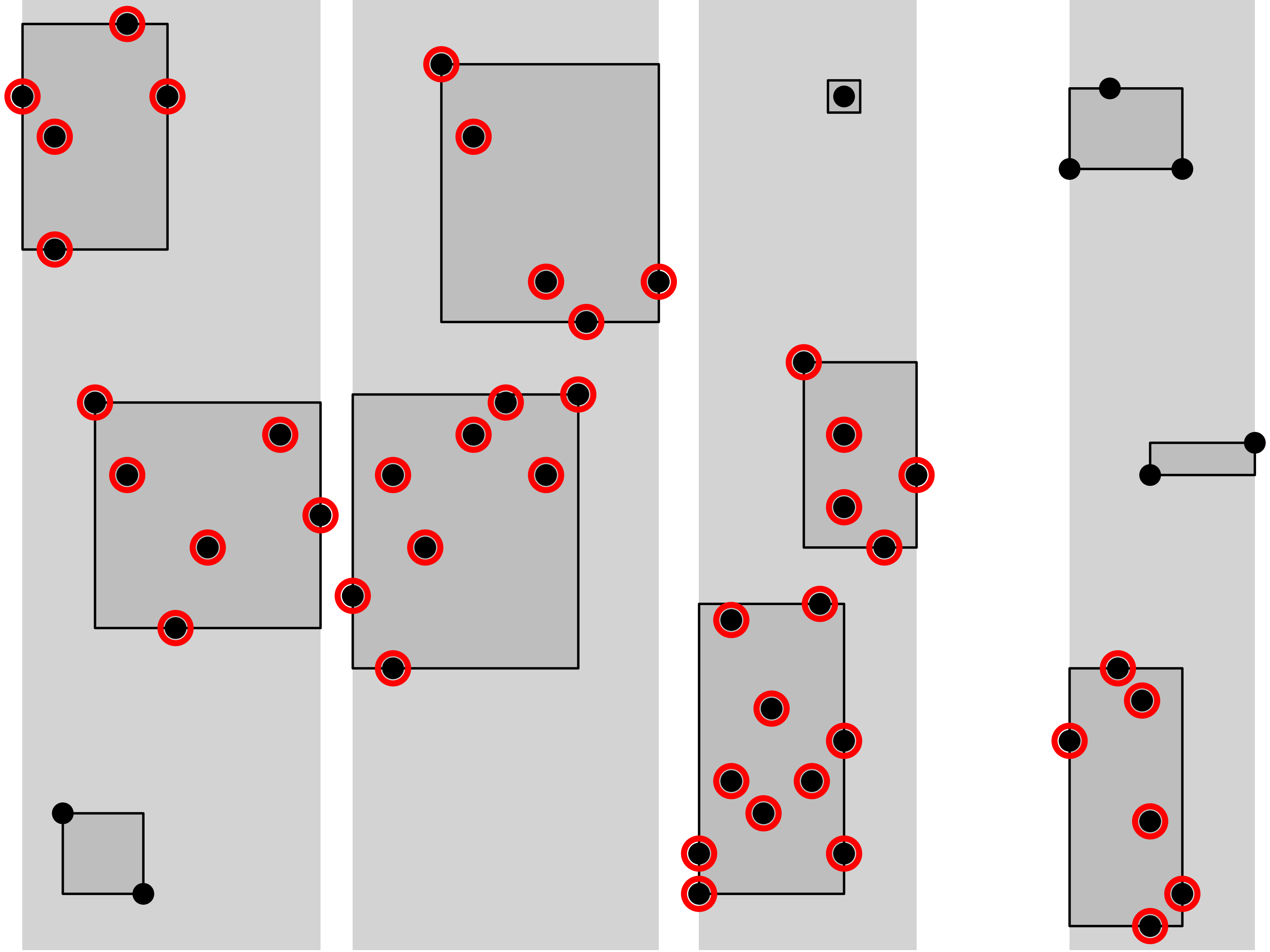
In boxes with at least k points, ...

Box graph $\mathcal{G}_{\text{box}}$

$k = 4$

ε:

$\varepsilon/\sqrt{2}$:



Property of single boxes

All points within a box...
are in ε-neighbourhood.

(Box width & height are each $\leqslant \varepsilon/\sqrt{2}$.)

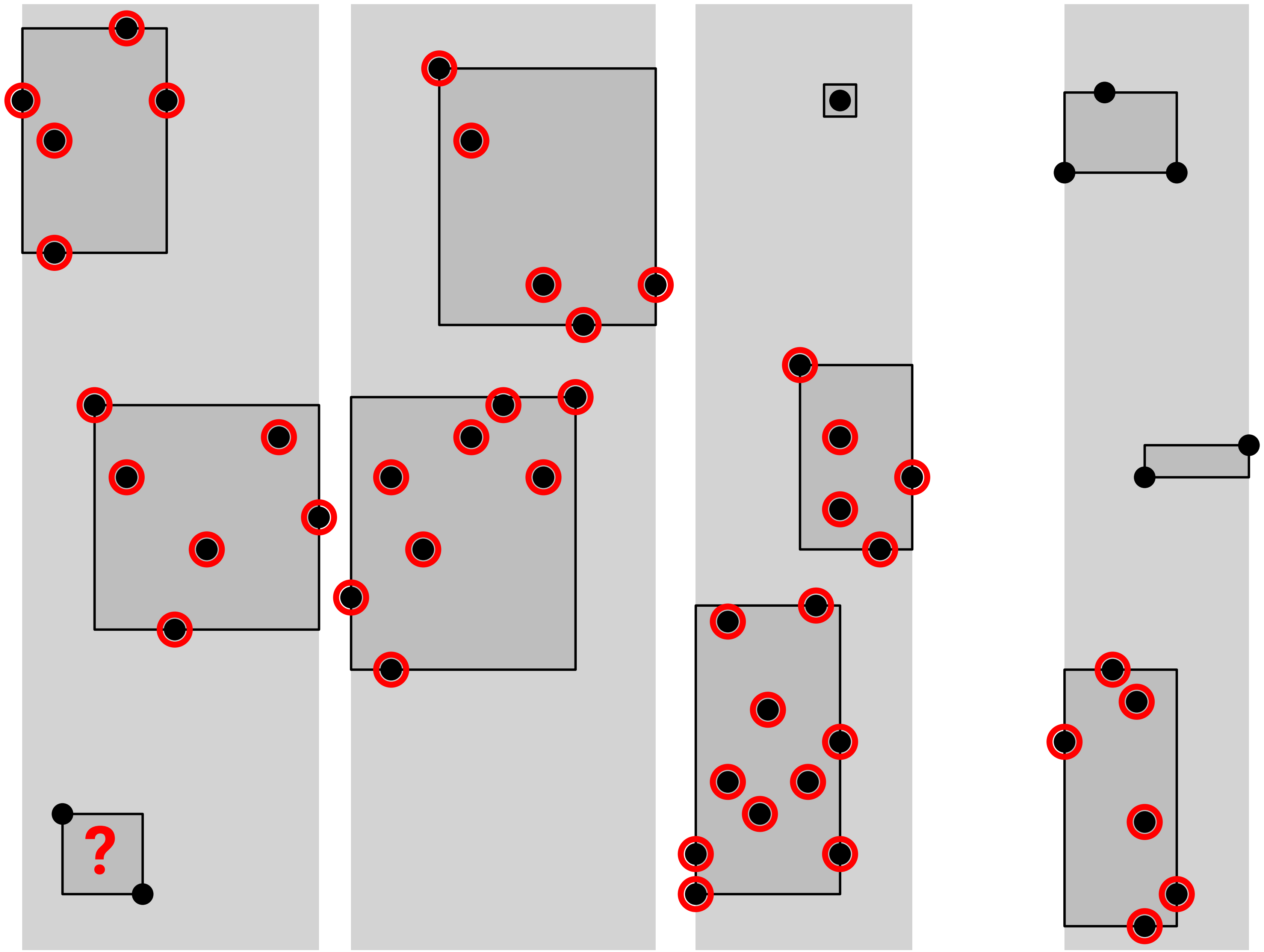
In boxes with at least k points, ...

all points are core points.

$k = 4$ ε: ⟷

$\varepsilon/\sqrt{2}$: ⟷

Box graph $\mathcal{G}_{\text{box}}$



Property of single boxes

All points within a box... are in ε-neighbourhood.

(Box width & height are each $\leqslant \varepsilon/\sqrt{2}$.)

In boxes with at least k points, ...

all points are core points.

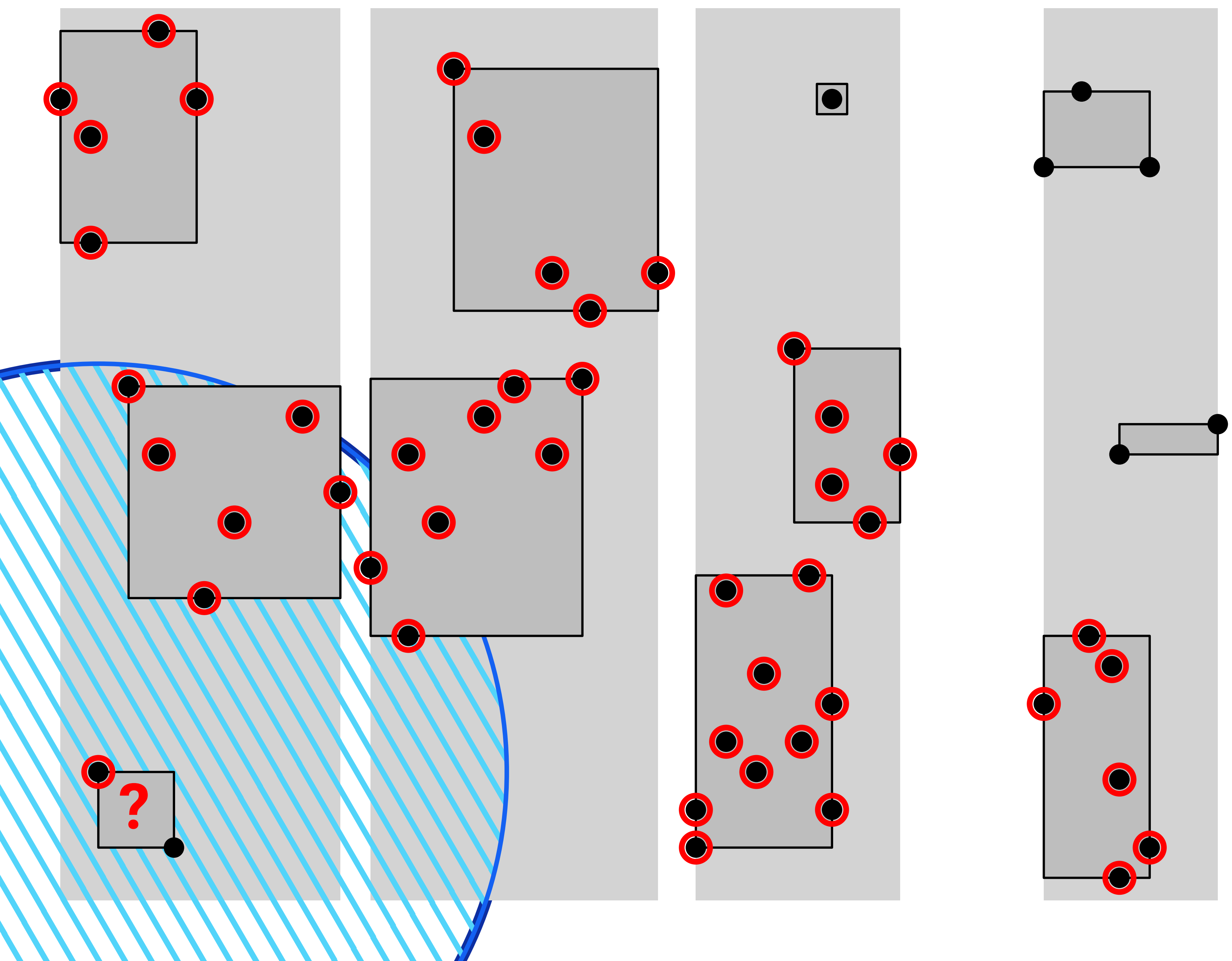
In boxes with fewer than k points, ...

Box graph $\mathcal{G}_{box}$

$k = 4$

ε:

$\varepsilon/\sqrt{2}$:



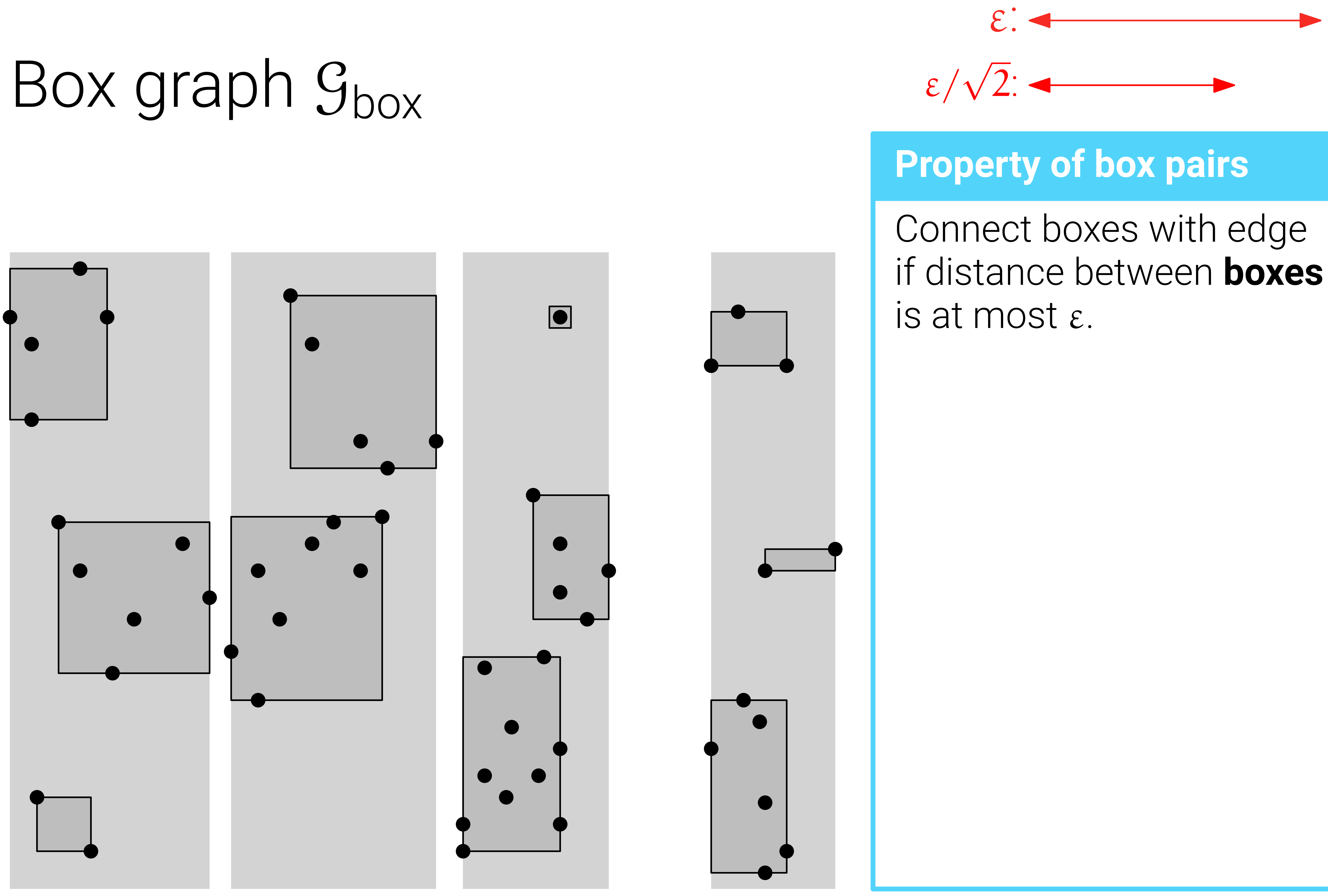
Property of single boxes

All points within a box...
are in ε-neighbourhood.
(Box width & height are each $\leqslant \varepsilon/\sqrt{2}$.)

In boxes with at least k points, ...
all points are core points.

In boxes with fewer than k points, ...
points **can** be core points.

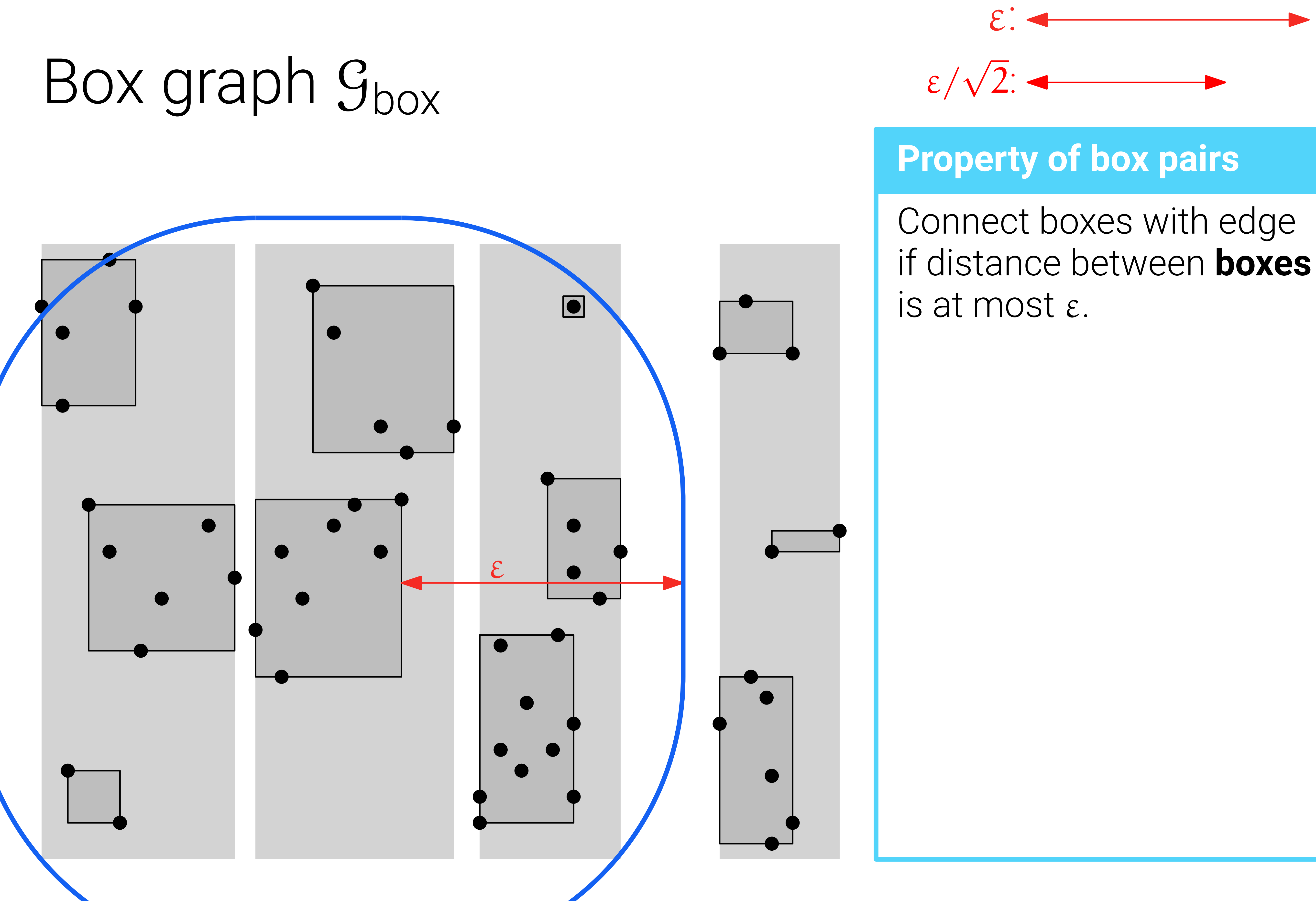
Box graph $\mathcal{G}_{box}$



Property of box pairs

Connect boxes with edge if distance between **boxes** is at most ε.

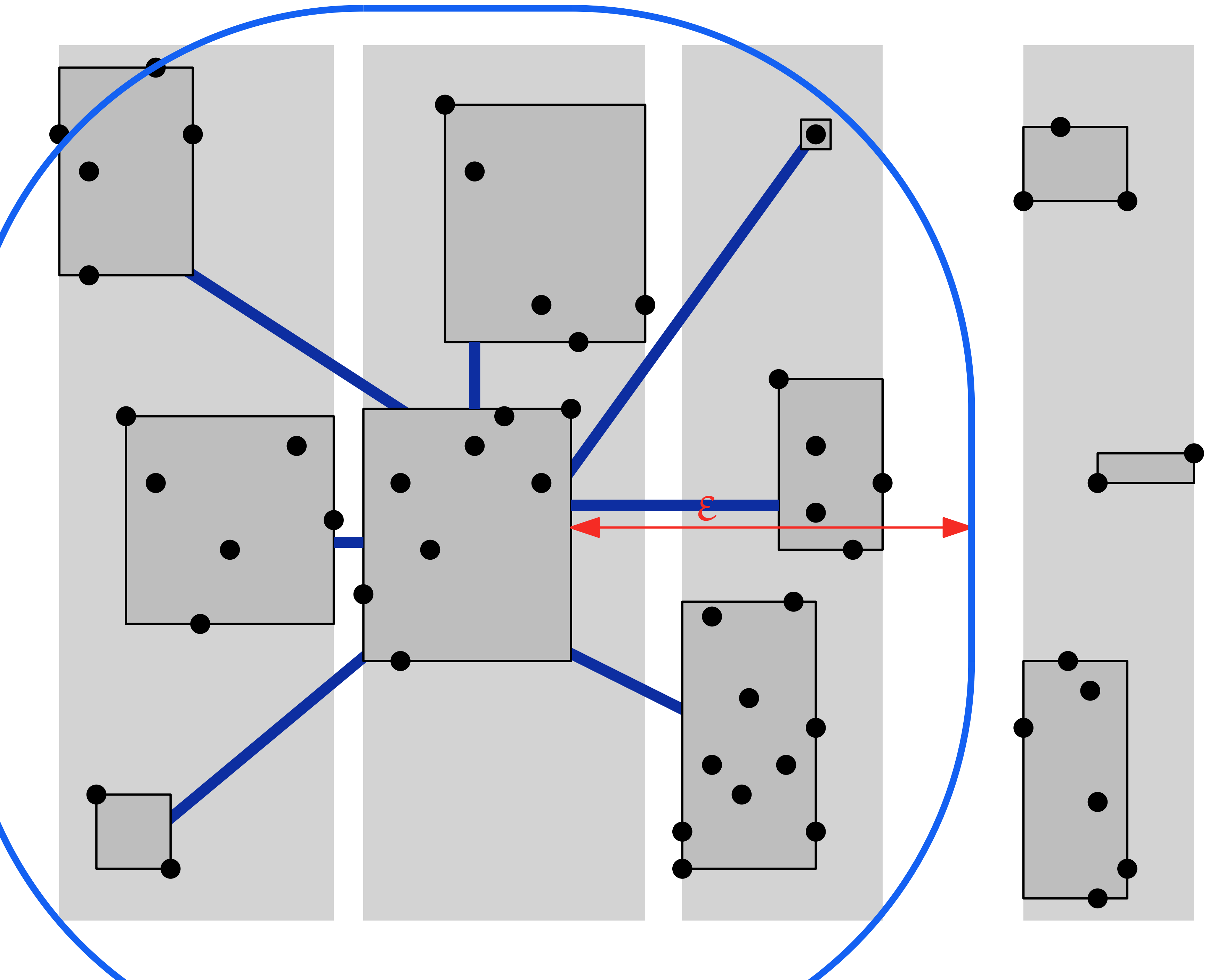
Box graph $\mathcal{G}_{\text{box}}$



Property of box pairs

Connect boxes with edge if distance between **boxes** is at most ε.

Box graph $\mathcal{G}_{box}$



ε:

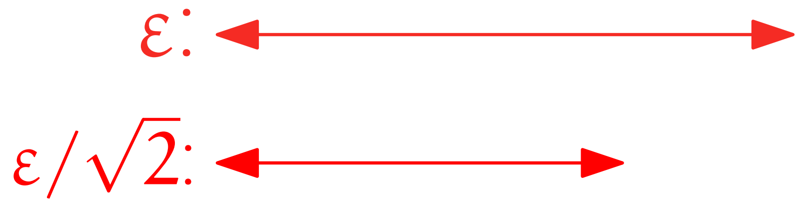
$\varepsilon/\sqrt{2}$:

Property of box pairs

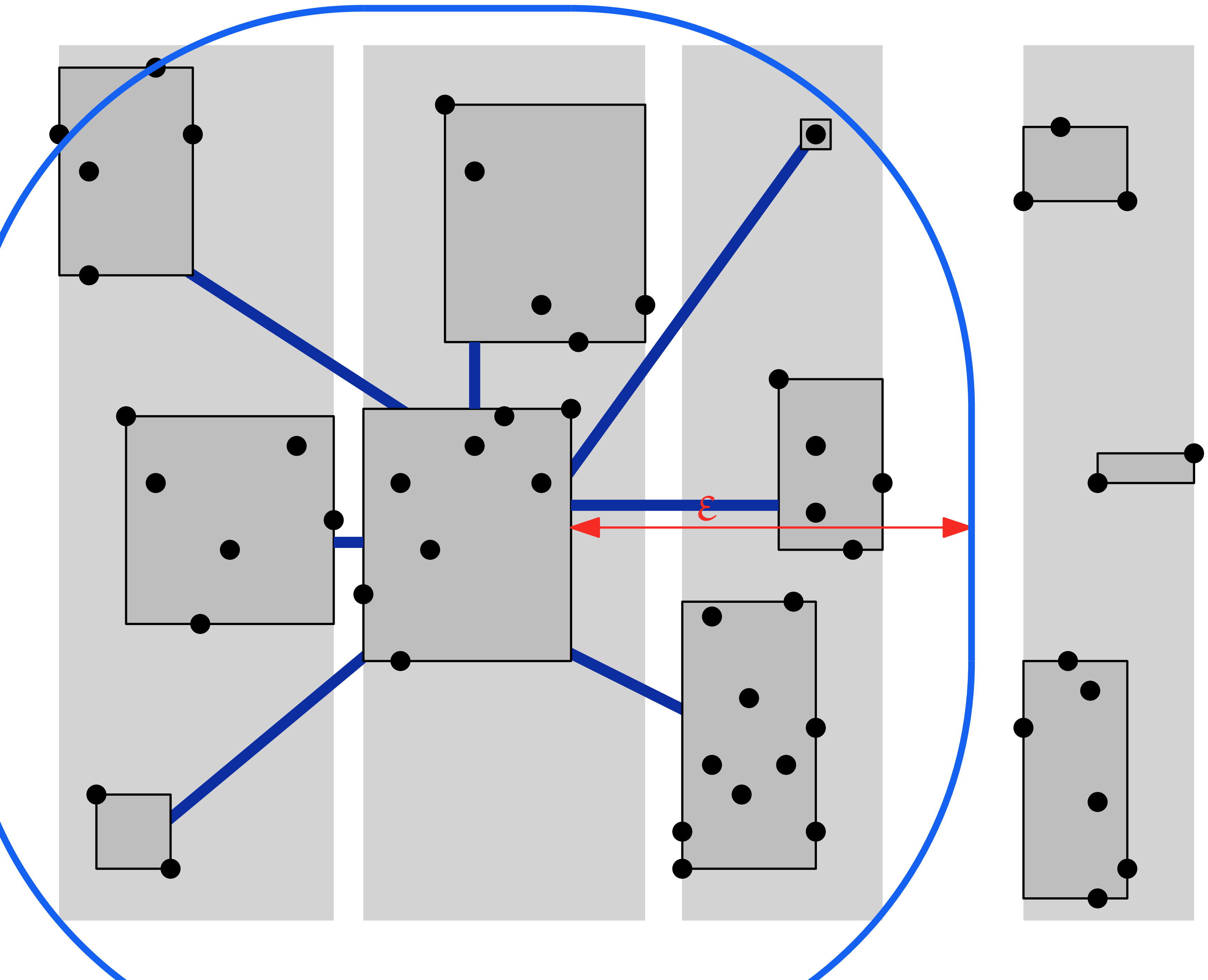
Connect boxes with edge if distance between **boxes** is at most ε.

Nonneighbours in $\mathcal{G}_{box}$:

none of these points are in ε-neighbourhood.



Box graph $\mathcal{G}_{\text{box}}$



Property of box pairs

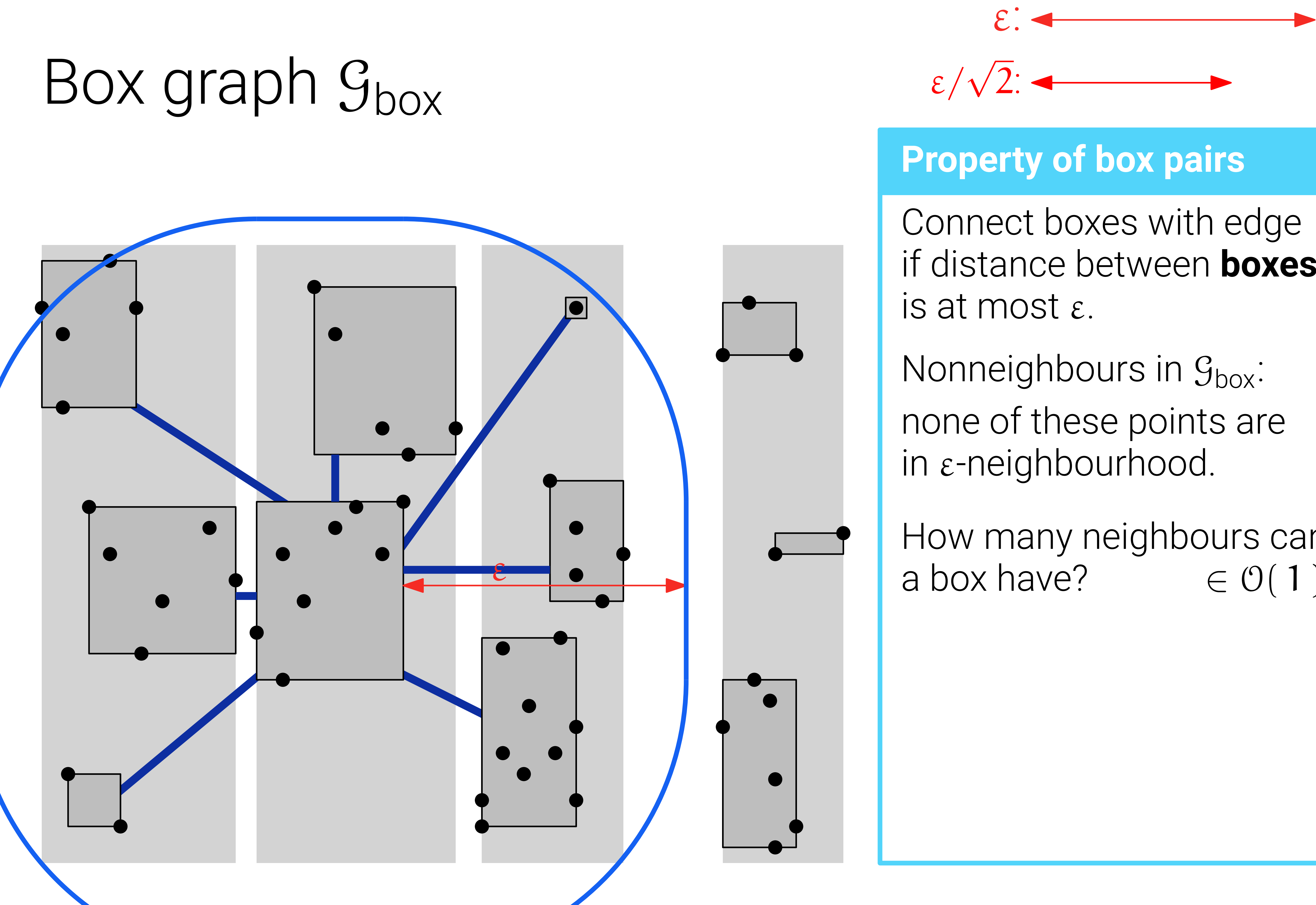
Connect boxes with edge if distance between **boxes** is at most ε.

Nonneighbours in $\mathcal{G}_{\text{box}}$:

none of these points are in ε-neighbourhood.

How many neighbours can a box have?

Box graph $\mathcal{G}_{box}$



Property of box pairs

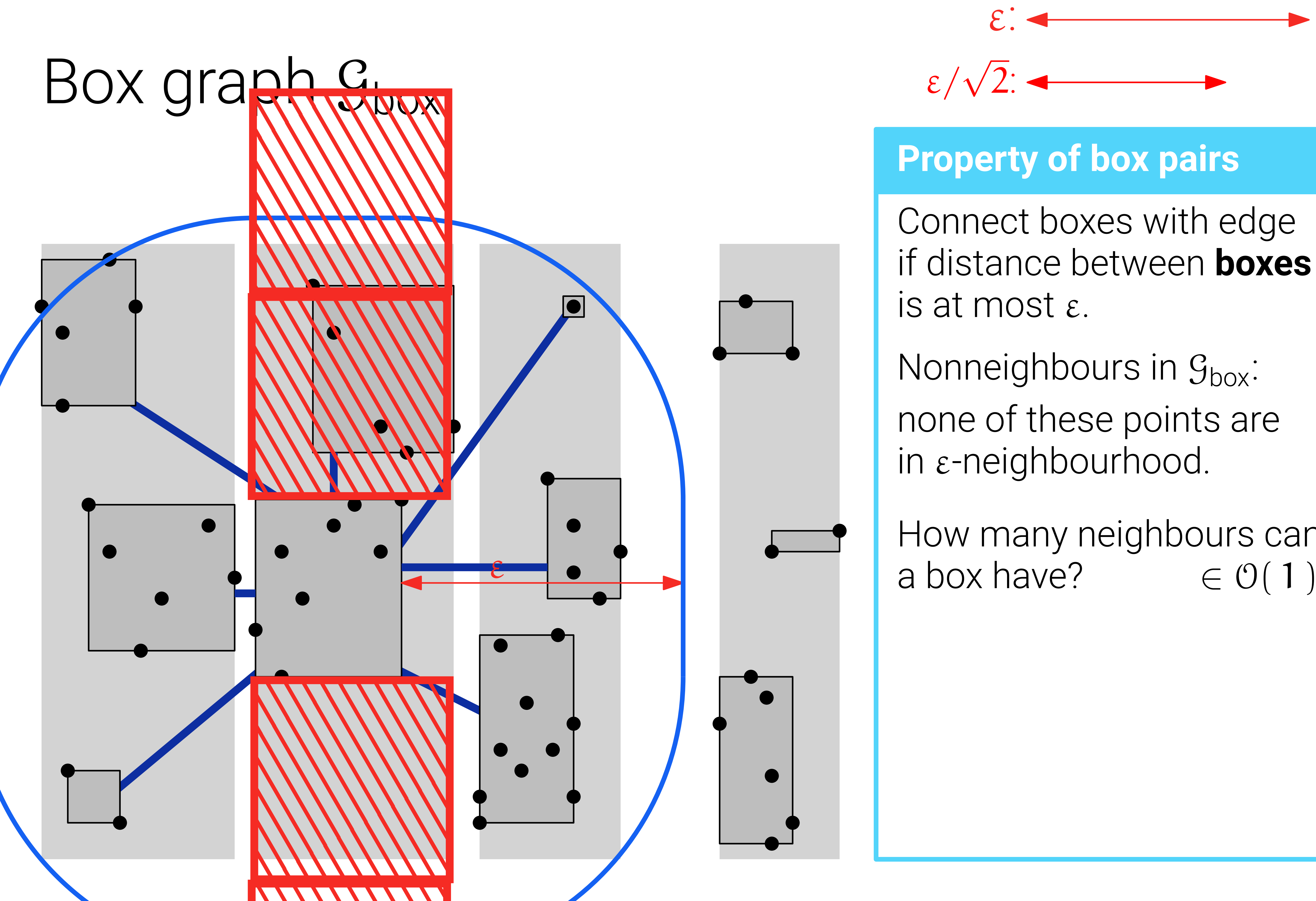
Connect boxes with edge if distance between **boxes** is at most ε.

Nonneighbours in $\mathcal{G}_{box}$:

none of these points are in ε-neighbourhood.

How many neighbours can a box have? $\in \mathcal{O}(1)$

Box graph $\mathcal{G}_{box}$



Property of box pairs

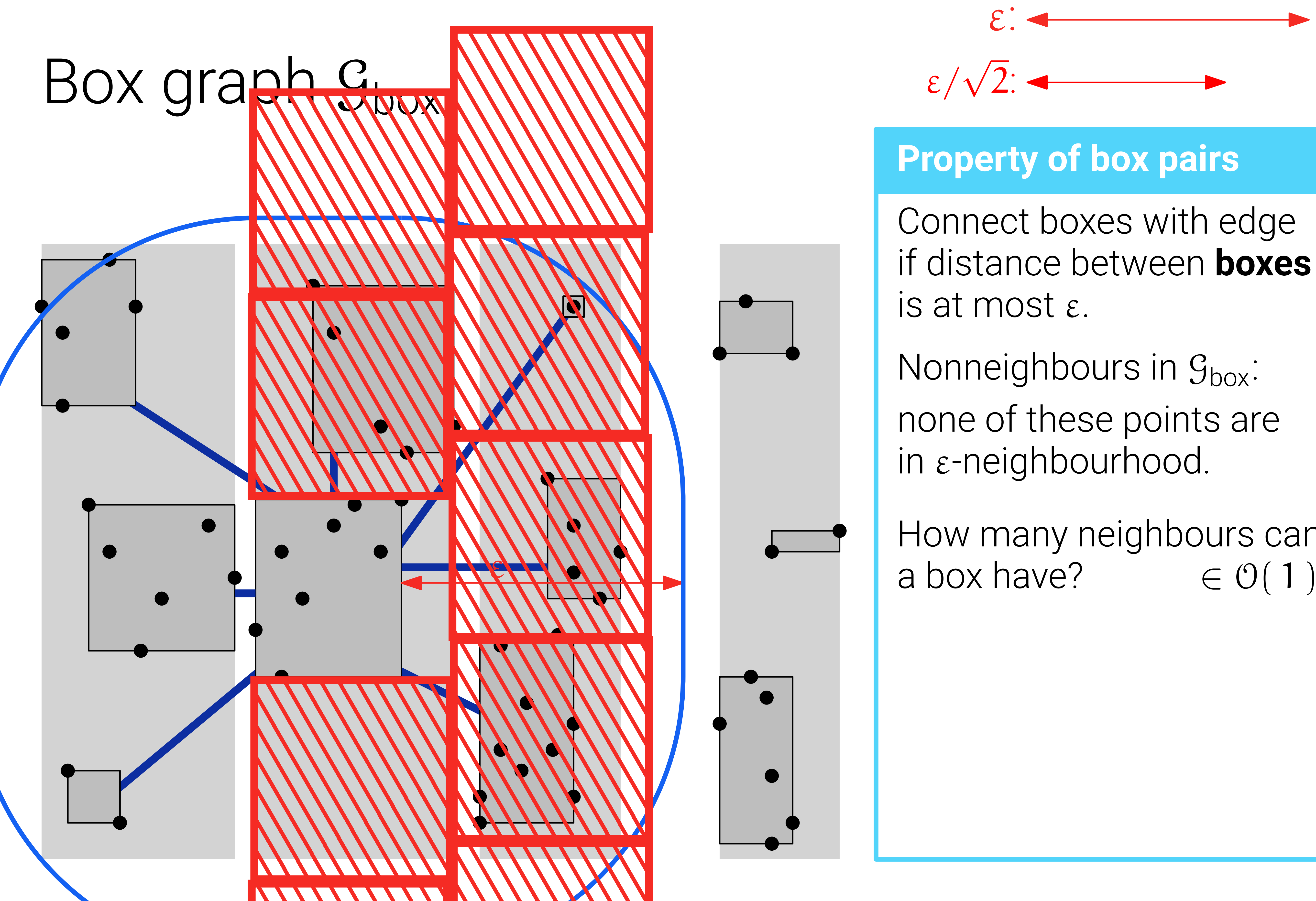
Connect boxes with edge if distance between **boxes** is at most ε.

Nonneighbours in $\mathcal{G}_{box}$:

none of these points are in ε-neighbourhood.

How many neighbours can a box have? $\in \mathcal{O}(1)$

Box graph $\mathcal{G}_{box}$



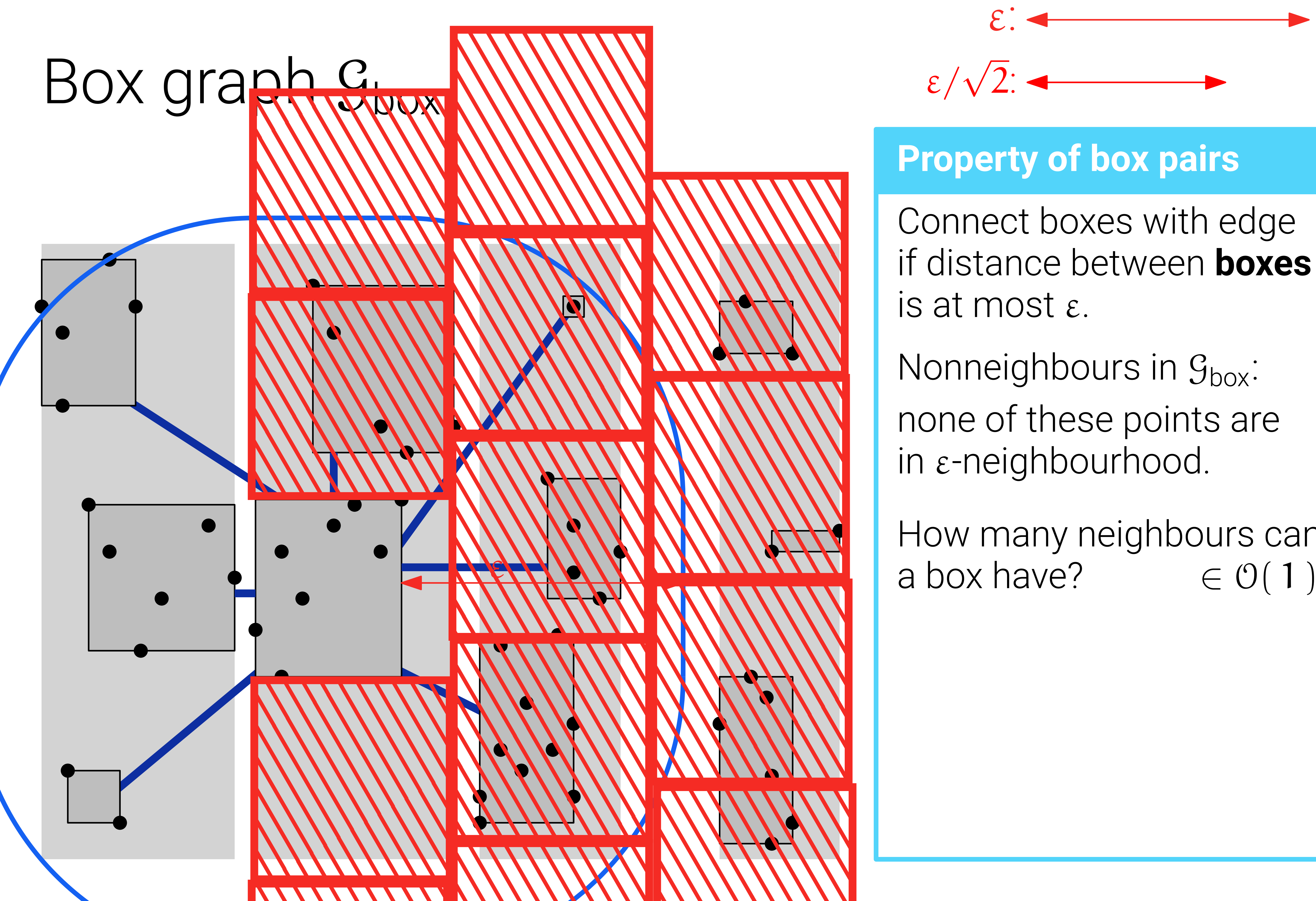
Property of box pairs

Connect boxes with edge if distance between **boxes** is at most ε.

Nonneighbours in $\mathcal{G}_{box}$: none of these points are in ε-neighbourhood.

How many neighbours can a box have? $\in \mathcal{O}(1)$

Box graph $\mathcal{G}_{box}$



Property of box pairs

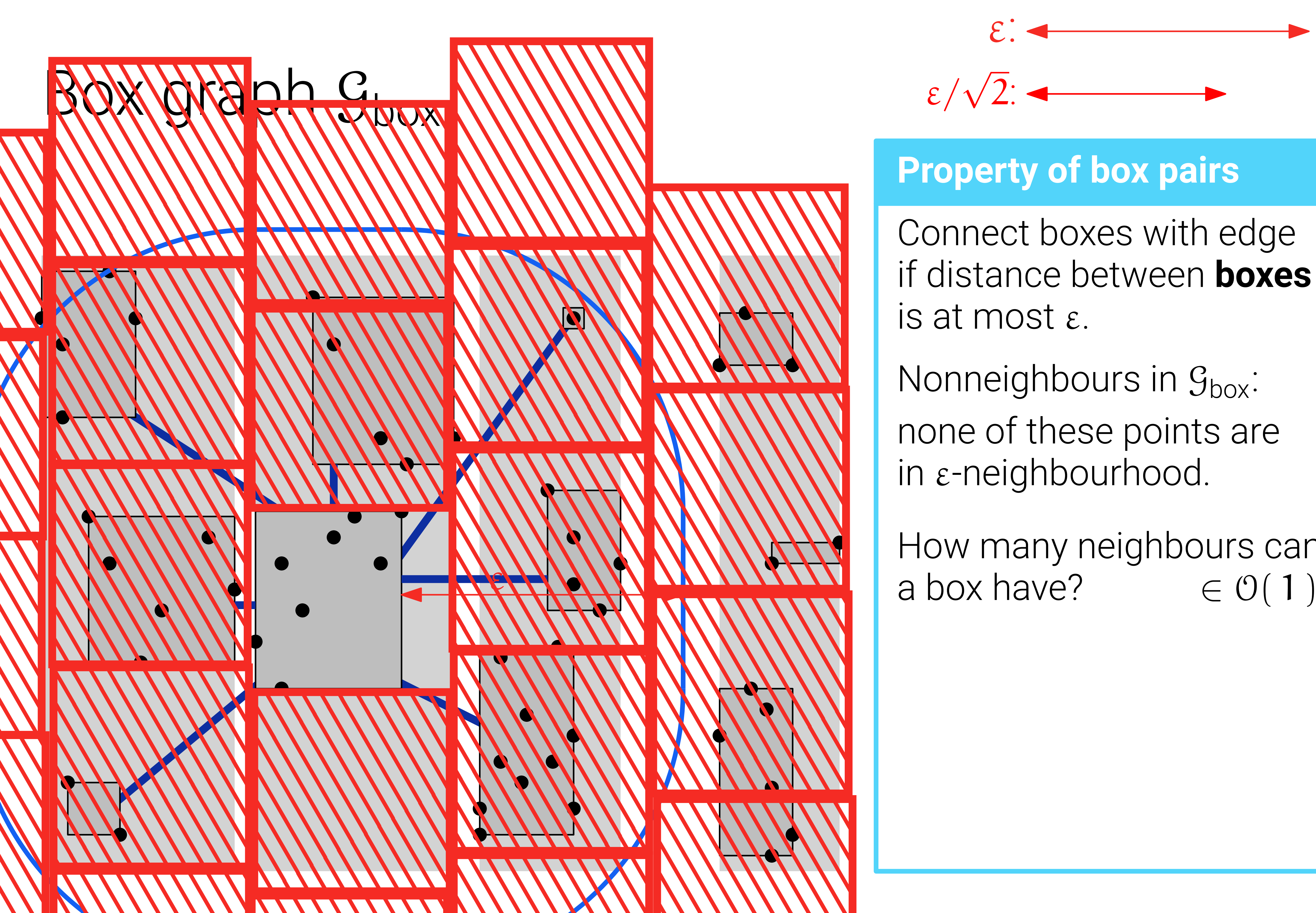
Connect boxes with edge if distance between **boxes** is at most ε.

Nonneighbours in $\mathcal{G}_{box}$:

none of these points are in ε-neighbourhood.

How many neighbours can a box have? $\in \mathcal{O}(1)$

Box graph $\mathcal{G}_{box}$



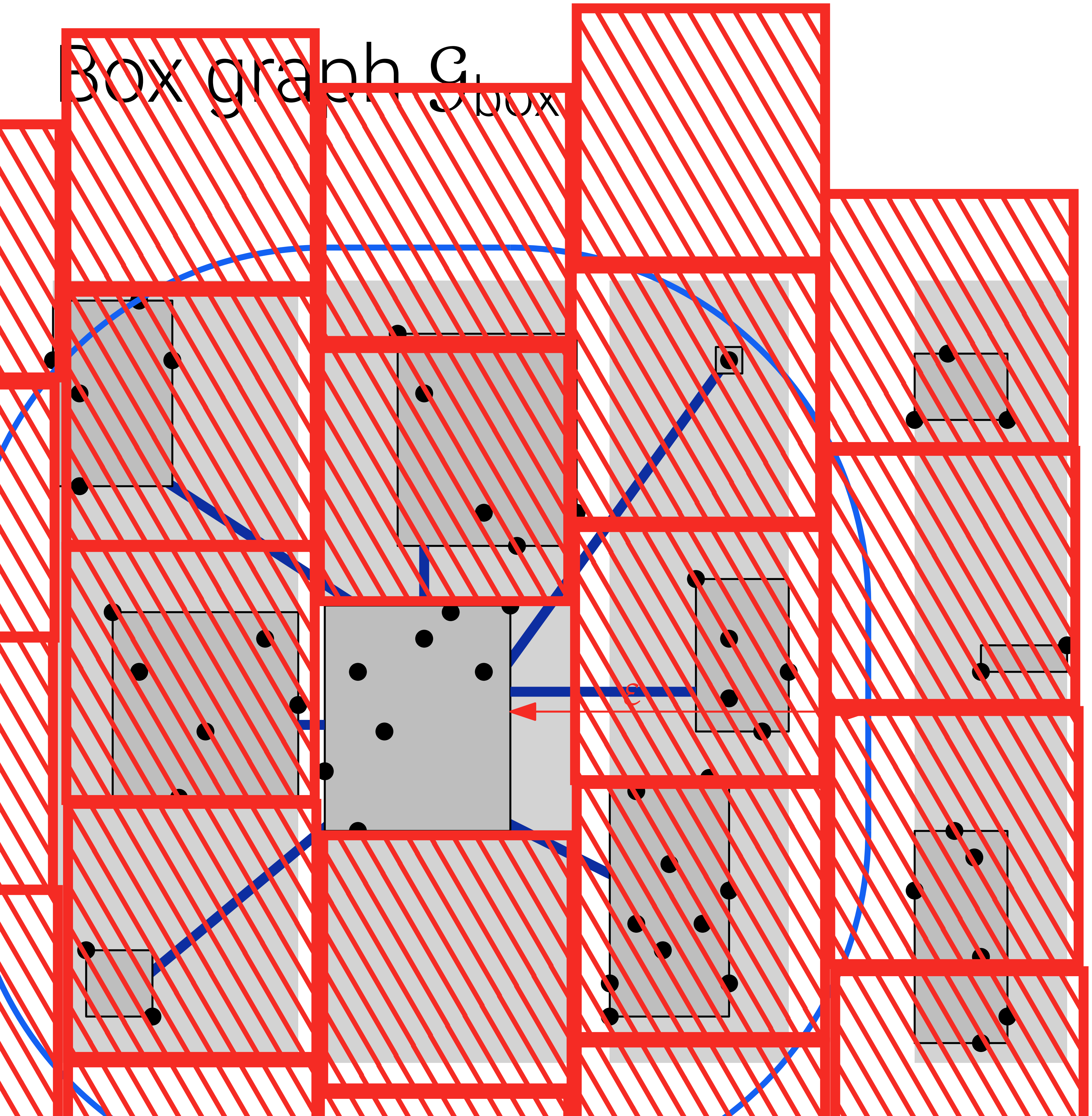
Property of box pairs

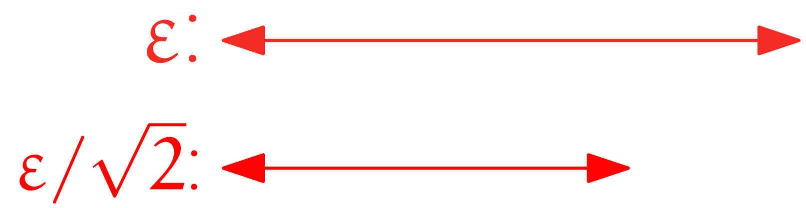
Connect boxes with edge if distance between **boxes** is at most ε.

Nonneighbours in $\mathcal{G}_{box}$: none of these points are in ε-neighbourhood.

How many neighbours can a box have? $\in \mathcal{O}(1)$

Box graph $\mathcal{G}_{\text{box}}$





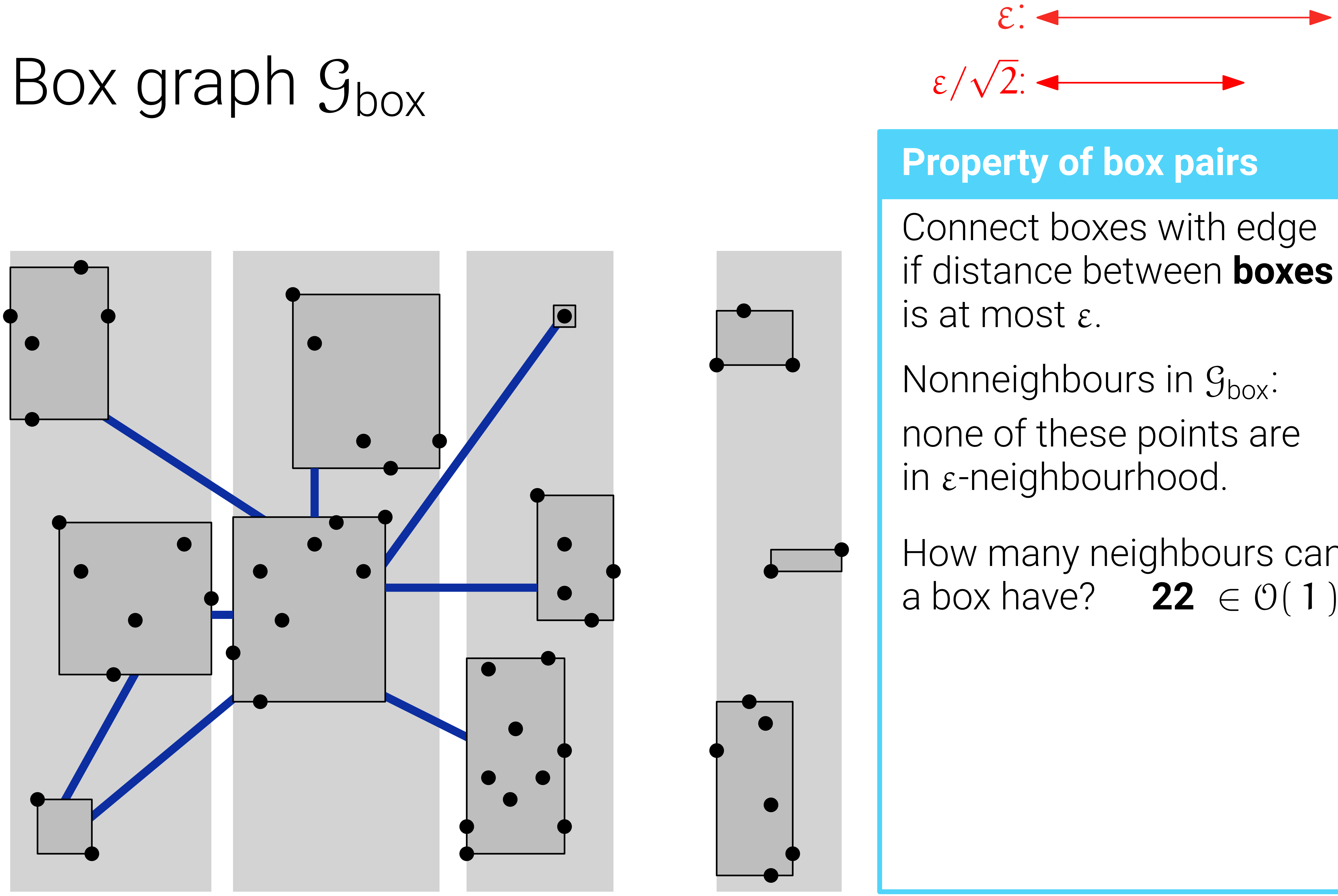
Property of box pairs

Connect boxes with edge if distance between **boxes** is at most ε.

Nonneighbours in $\mathcal{G}_{\text{box}}$: none of these points are in ε-neighbourhood.

How many neighbours can a box have? **22** $\in \mathcal{O}(1)$

Box graph $\mathcal{G}_{box}$



Property of box pairs

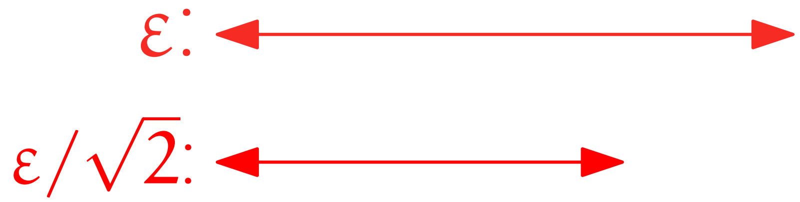
Connect boxes with edge if distance between **boxes** is at most ε.

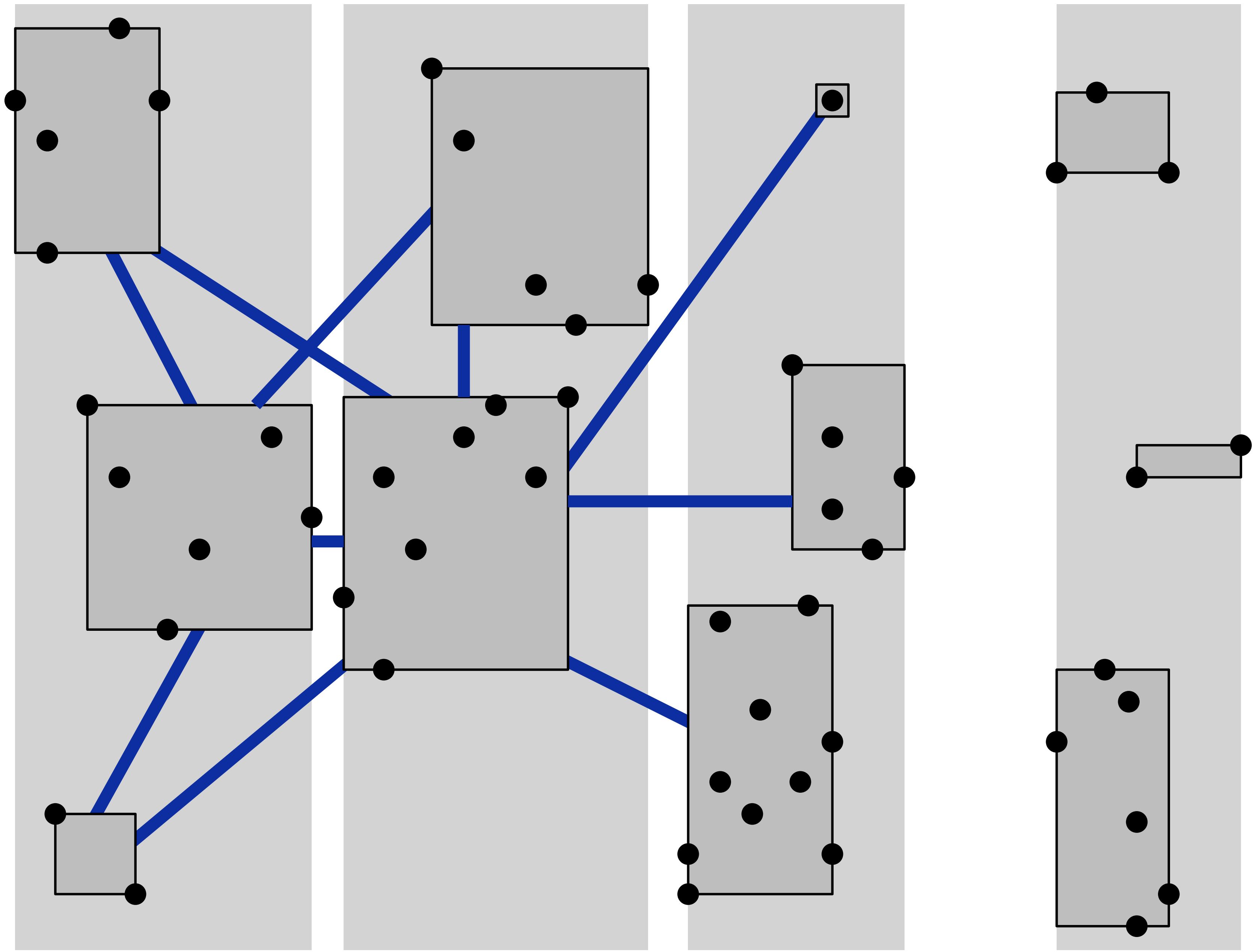
Nonneighbours in $\mathcal{G}_{box}$:

none of these points are in ε-neighbourhood.

How many neighbours can a box have? **22** $\in \mathcal{O}(1)$

Box graph $\mathcal{G}_{box}$





Property of box pairs

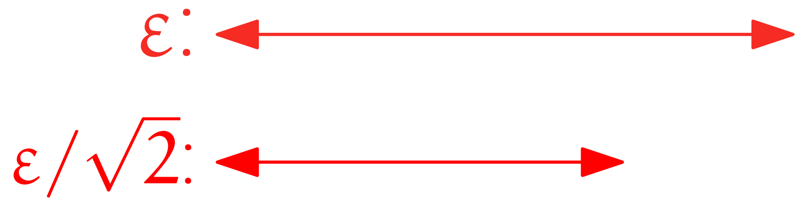
Connect boxes with edge if distance between **boxes** is at most ε.

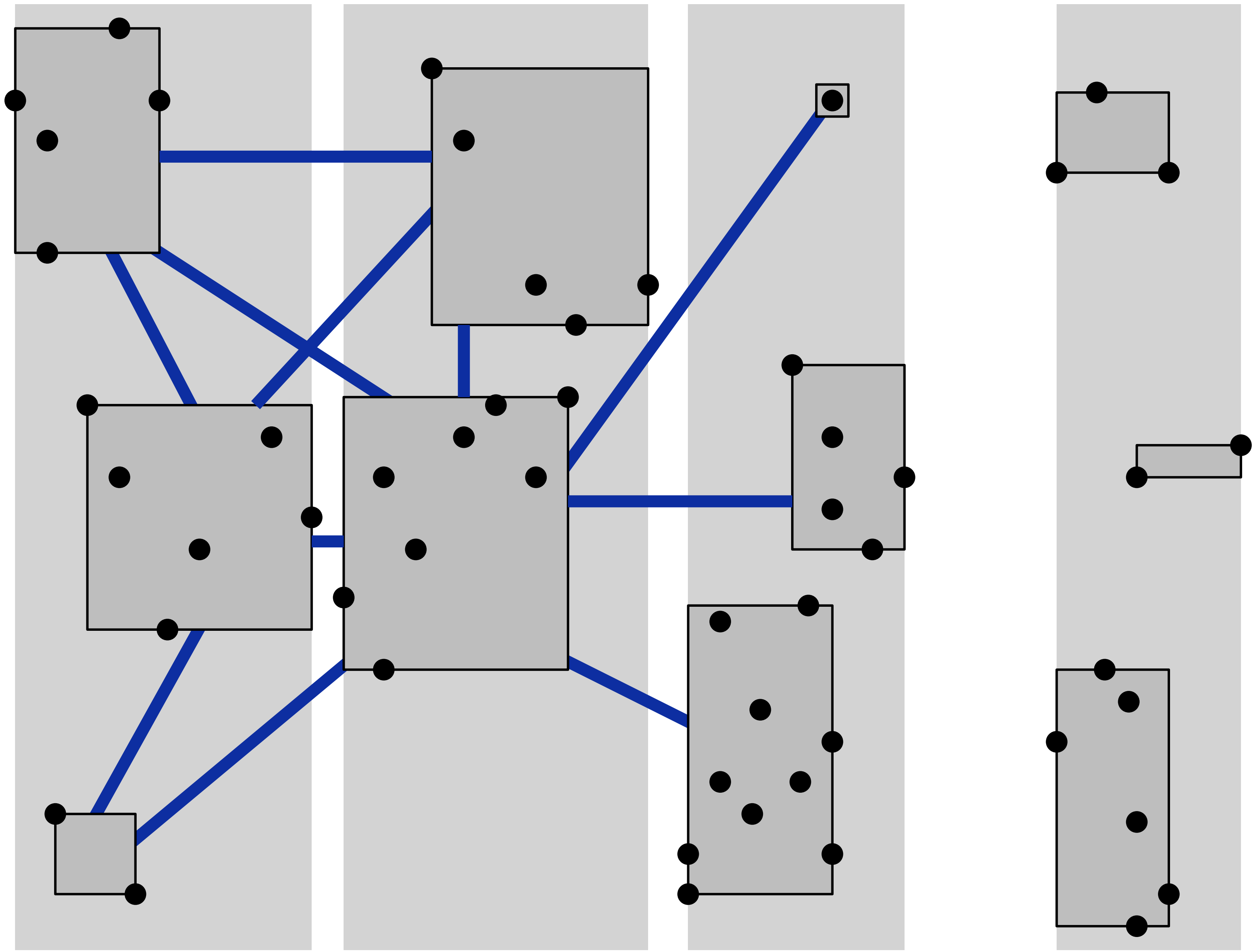
Nonneighbours in $\mathcal{G}_{box}$:

none of these points are in ε-neighbourhood.

How many neighbours can a box have? **22** $\in \mathcal{O}(1)$

Box graph $\mathcal{G}_{box}$





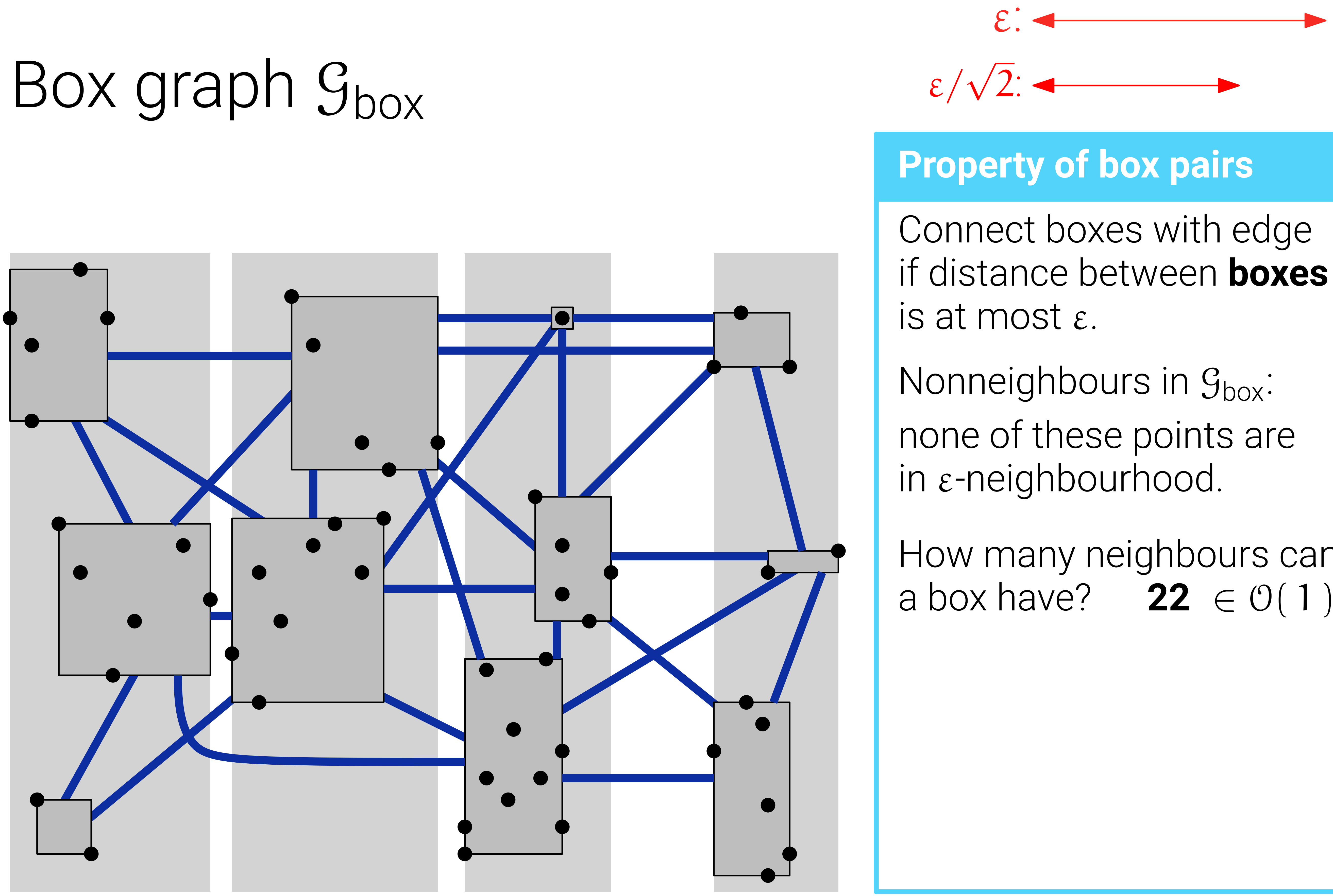
Property of box pairs

Connect boxes with edge if distance between **boxes** is at most ε.

Nonneighbours in $\mathcal{G}_{box}$: none of these points are in ε-neighbourhood.

How many neighbours can a box have? **22** $\in \mathcal{O}(1)$

Box graph $\mathcal{G}_{box}$



Property of box pairs

Connect boxes with edge if distance between **boxes** is at most ε.

Nonneighbours in $\mathcal{G}_{box}$: none of these points are in ε-neighbourhood.

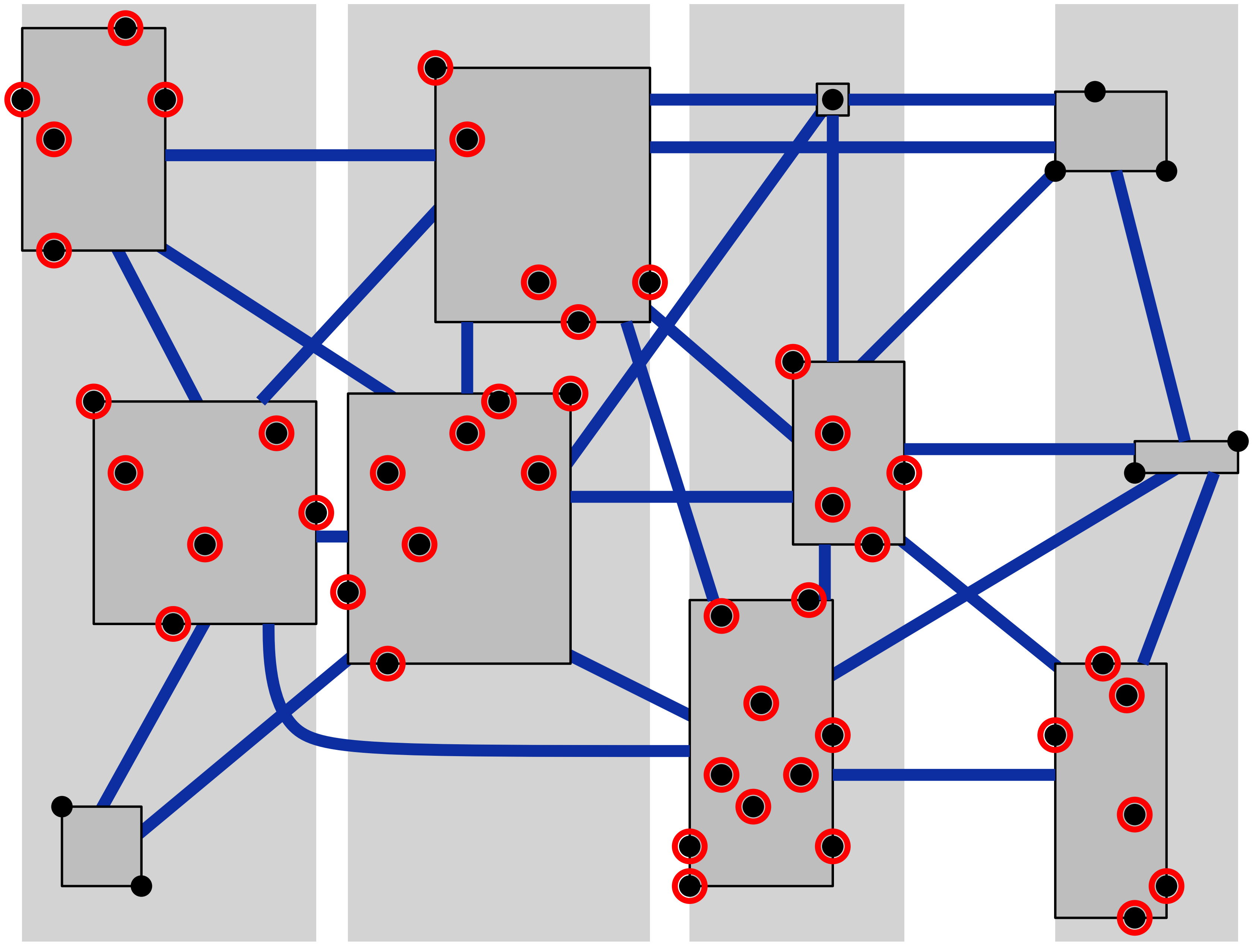
How many neighbours can a box have? **22** $\in \mathcal{O}(1)$

Box graph $\mathcal{G}_{box}$

$k = 4$

ε:

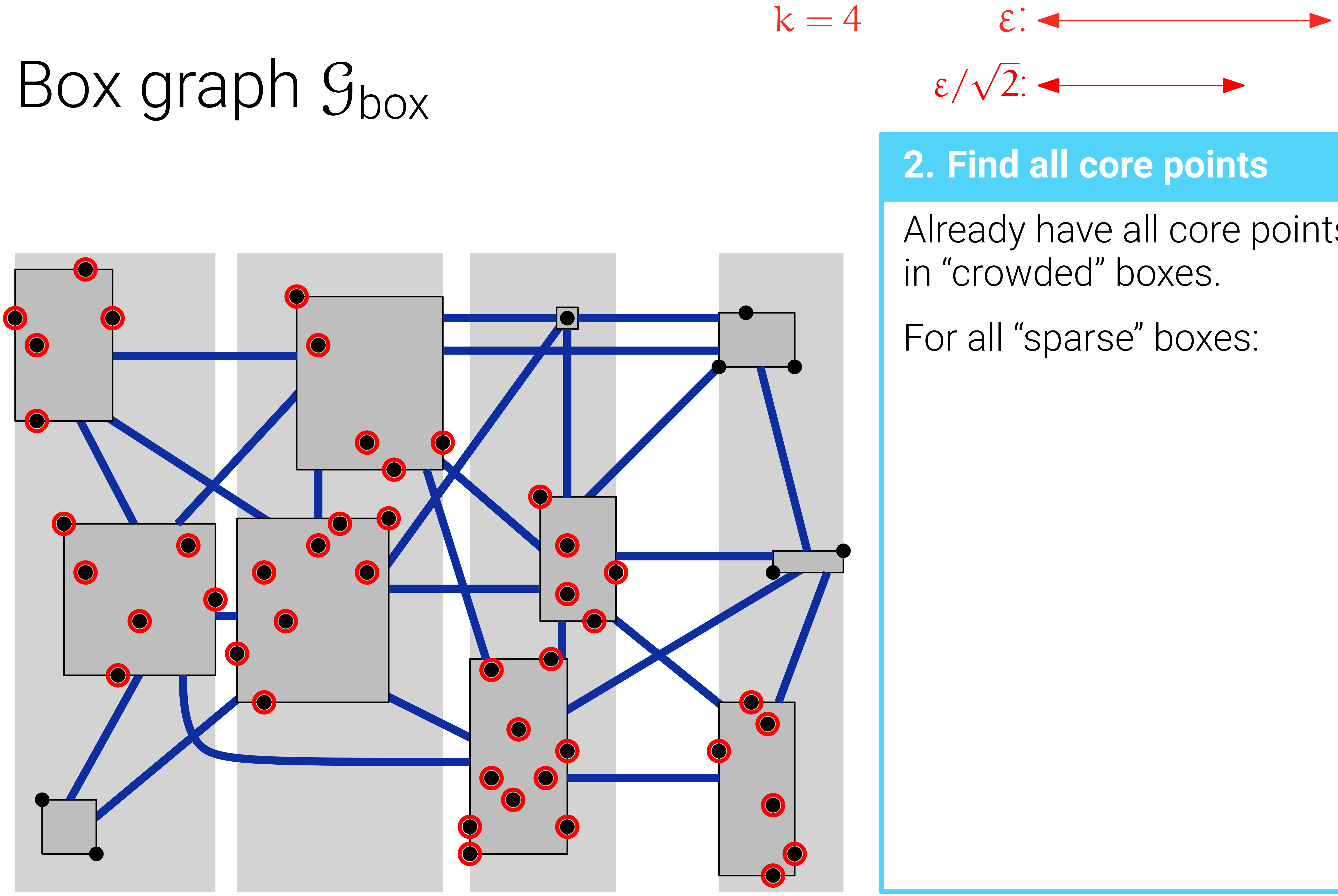
$\varepsilon/\sqrt{2}$:



2. Find all core points

Already have all core points in “crowded” boxes.

Box graph $\mathcal{G}_{box}$

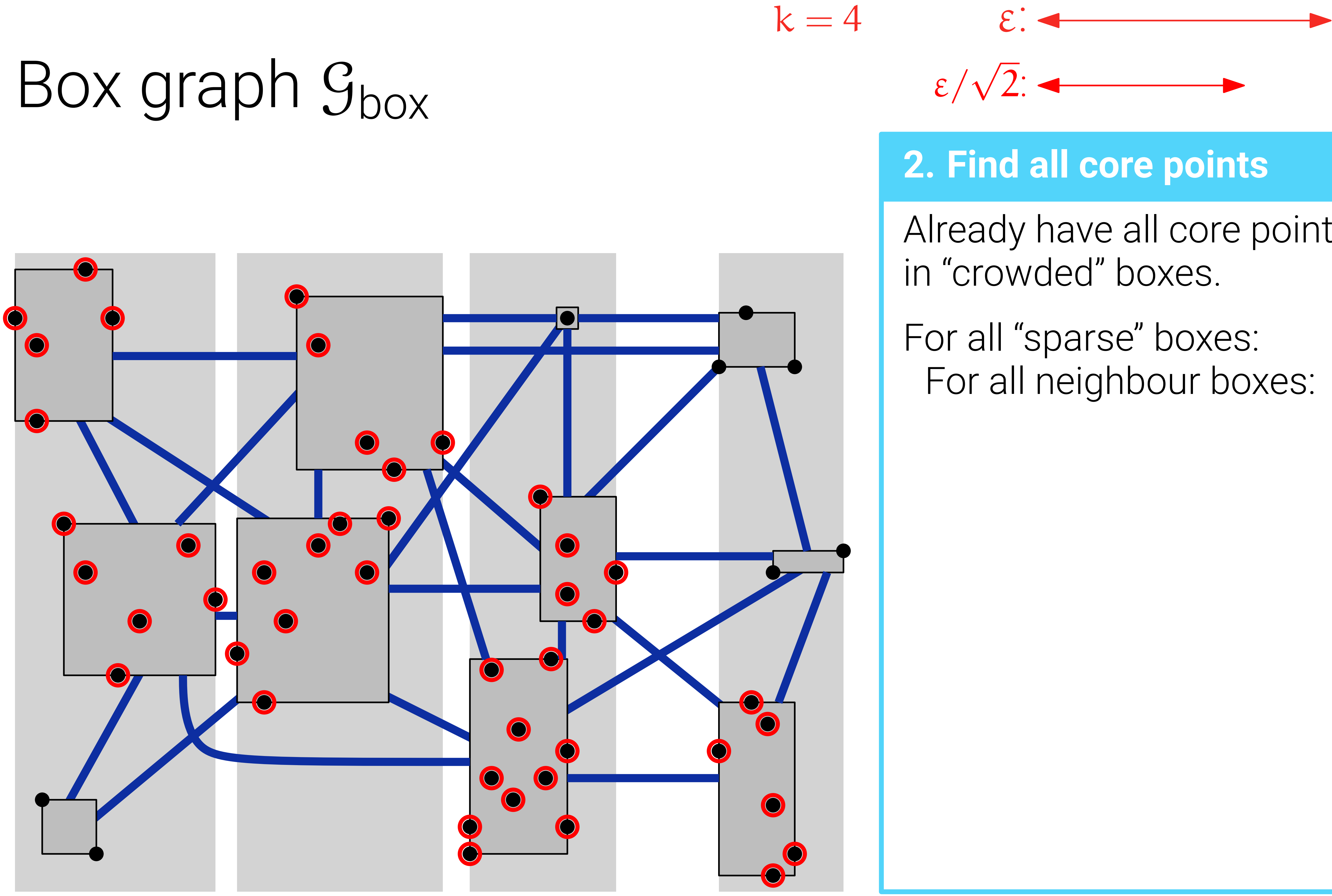


2. Find all core points

Already have all core points in "crowded" boxes.

For all "sparse" boxes:

Box graph $\mathcal{G}_{box}$

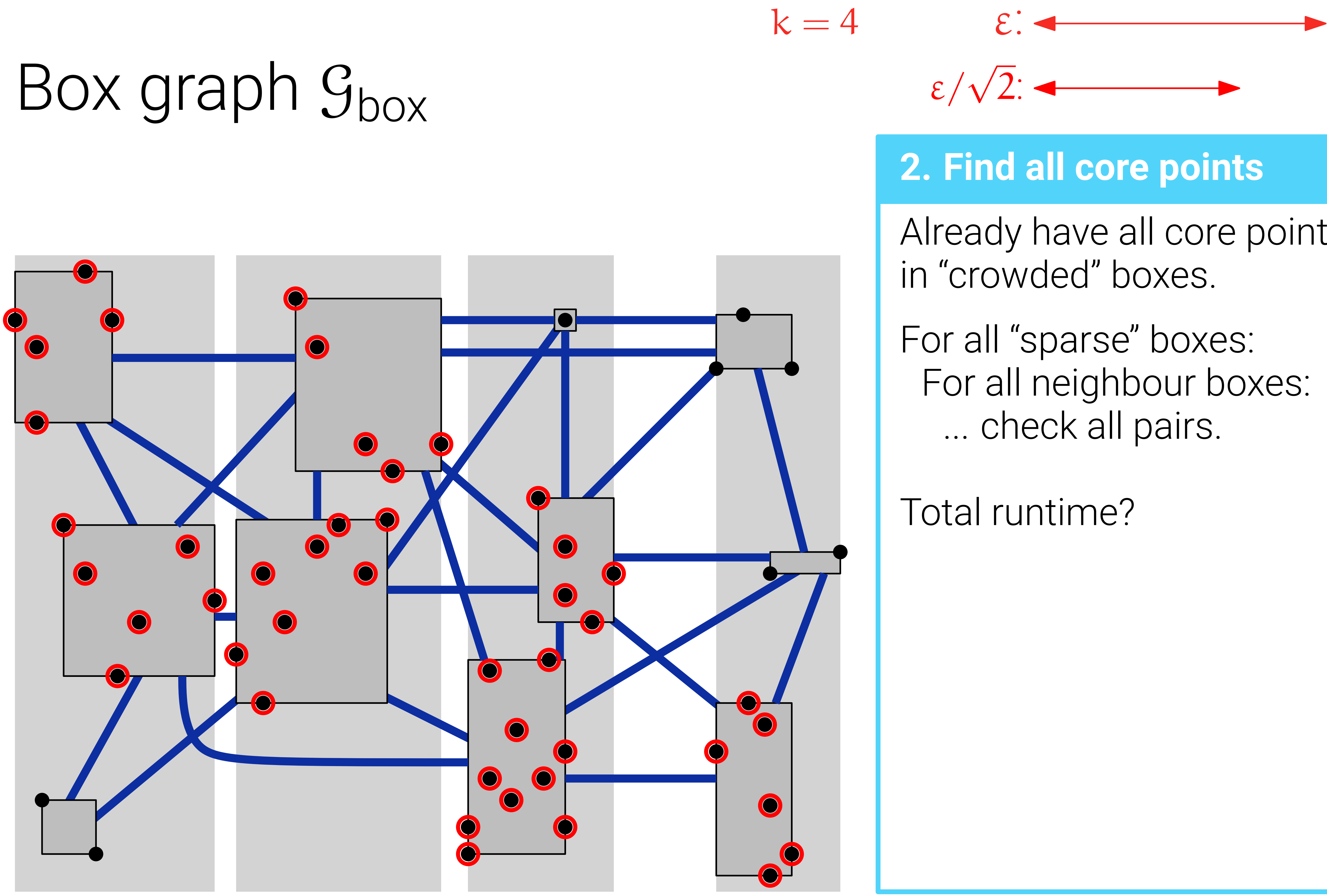


2. Find all core points

Already have all core points in “crowded” boxes.

For all “sparse” boxes:
  For all neighbour boxes:

Box graph $\mathcal{G}_{box}$



2. Find all core points

Already have all core points in "crowded" boxes.

For all "sparse" boxes:
  For all neighbour boxes:
    ... check all pairs.

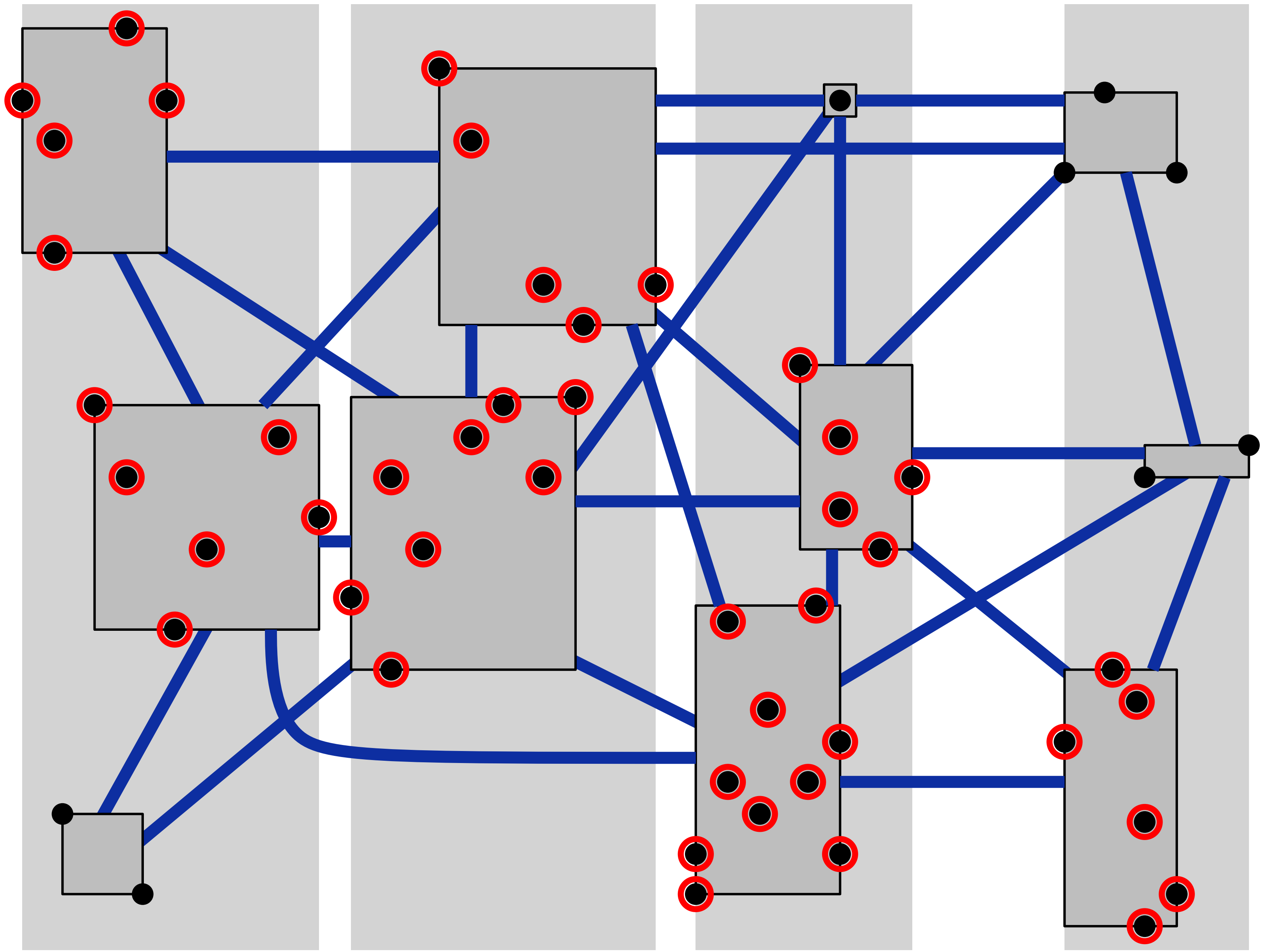
Total runtime?

Box graph $\mathcal{G}_{box}$

$k = 4$

ε:

$\varepsilon/\sqrt{2}$:



2. Find all core points

Already have all core points in "crowded" boxes.

For all "sparse" boxes:
  For all neighbour boxes:
    ... check all pairs.

Total runtime?

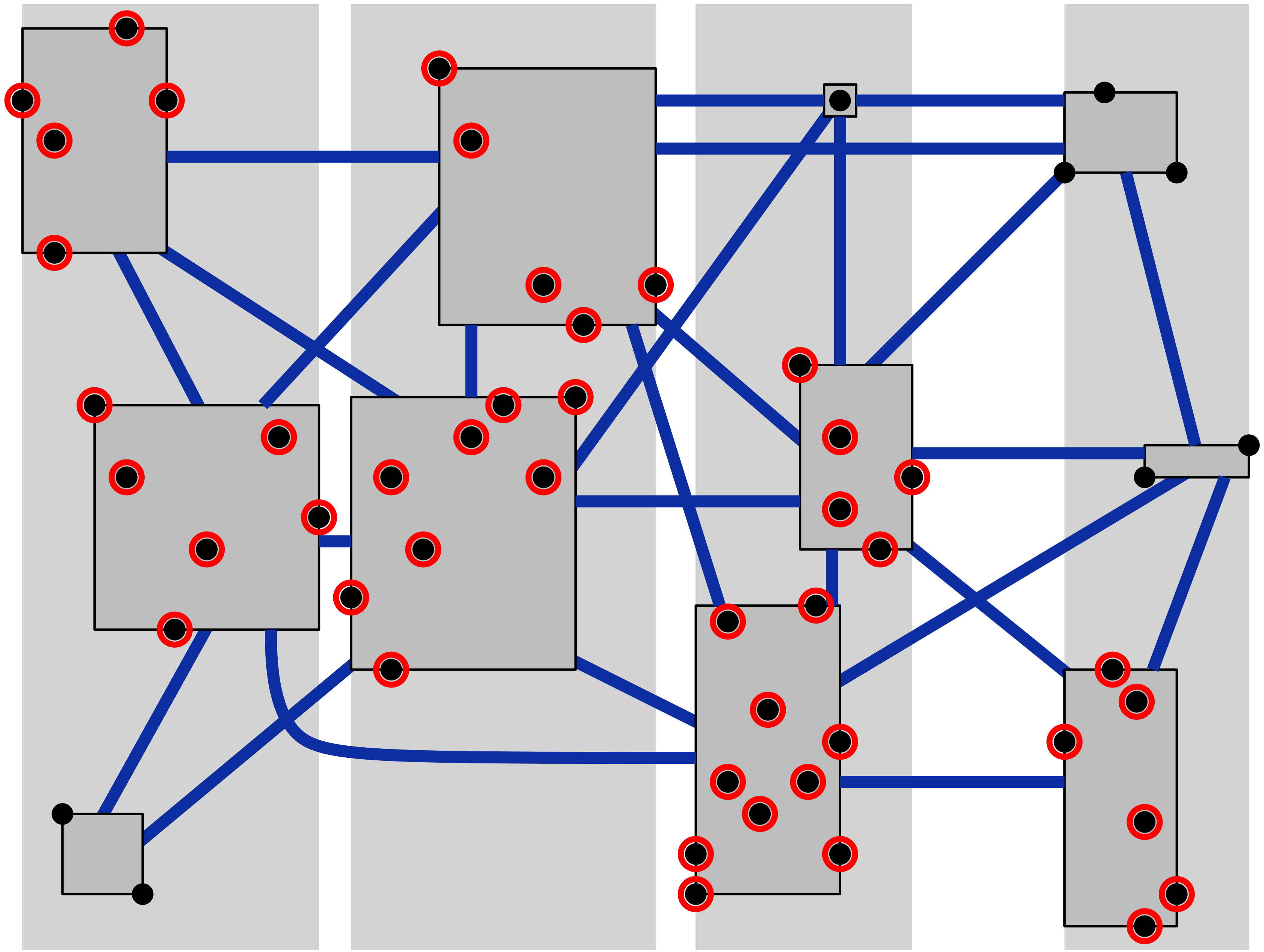
  Other box is sparse:

Box graph $\mathcal{G}_{box}$

$k = 4$

ε:

$\varepsilon/\sqrt{2}$:



2. Find all core points

Already have all core points in "crowded" boxes.

For all "sparse" boxes:
  For all neighbour boxes:
    ... check all pairs.

Total runtime?

Other box is sparse:

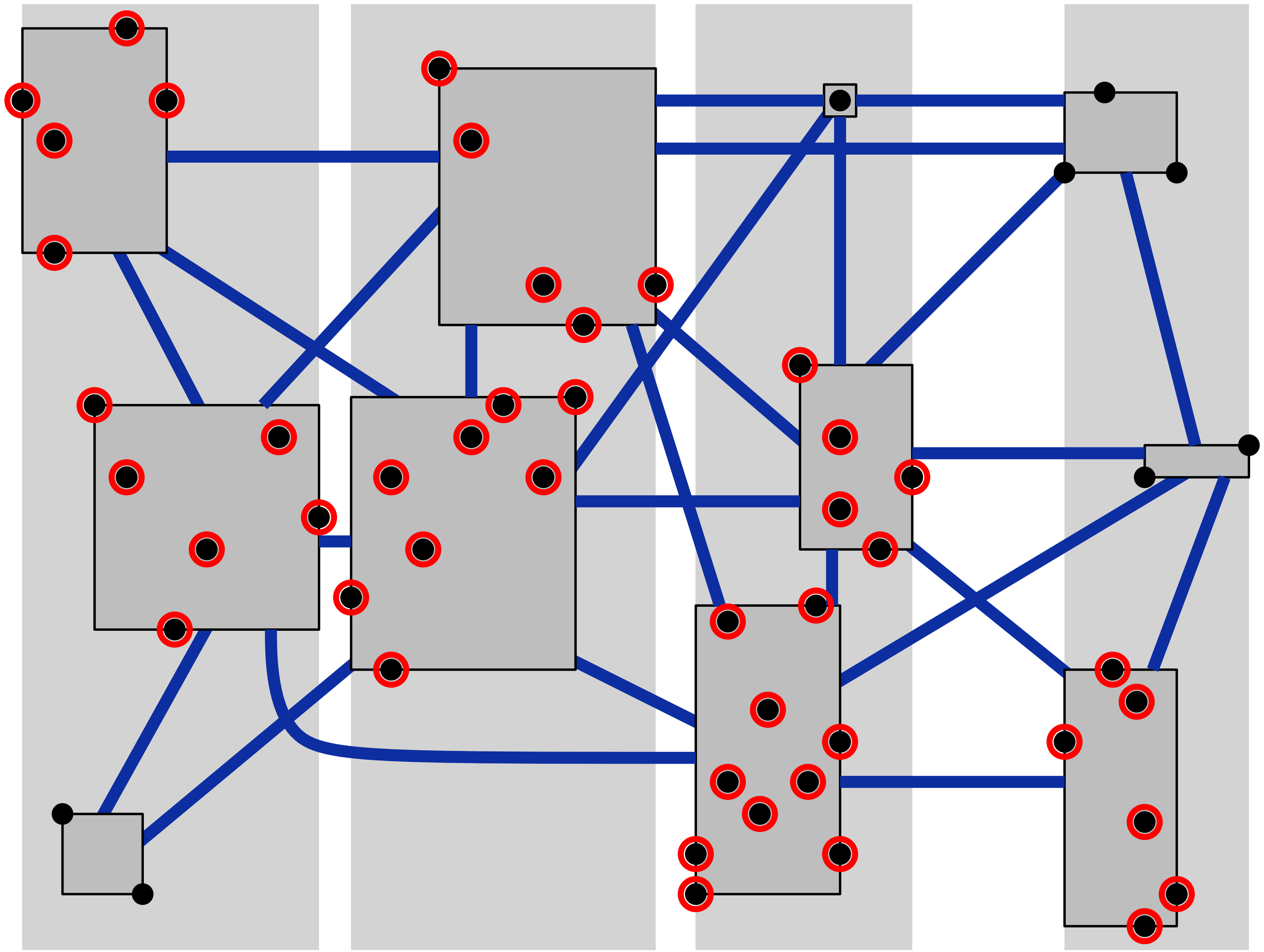
$$\mathcal{O}(k^2) = \mathcal{O}(1)$$

Box graph $\mathcal{G}_{box}$

$k = 4$

ε:

$\varepsilon/\sqrt{2}$:



2. Find all core points

Already have all core points in "crowded" boxes.

For all "sparse" boxes:
  For all neighbour boxes:
    ... check all pairs.

Total runtime?

  Other box is sparse:

  $\mathcal{O}(k^2) = \mathcal{O}(1)$

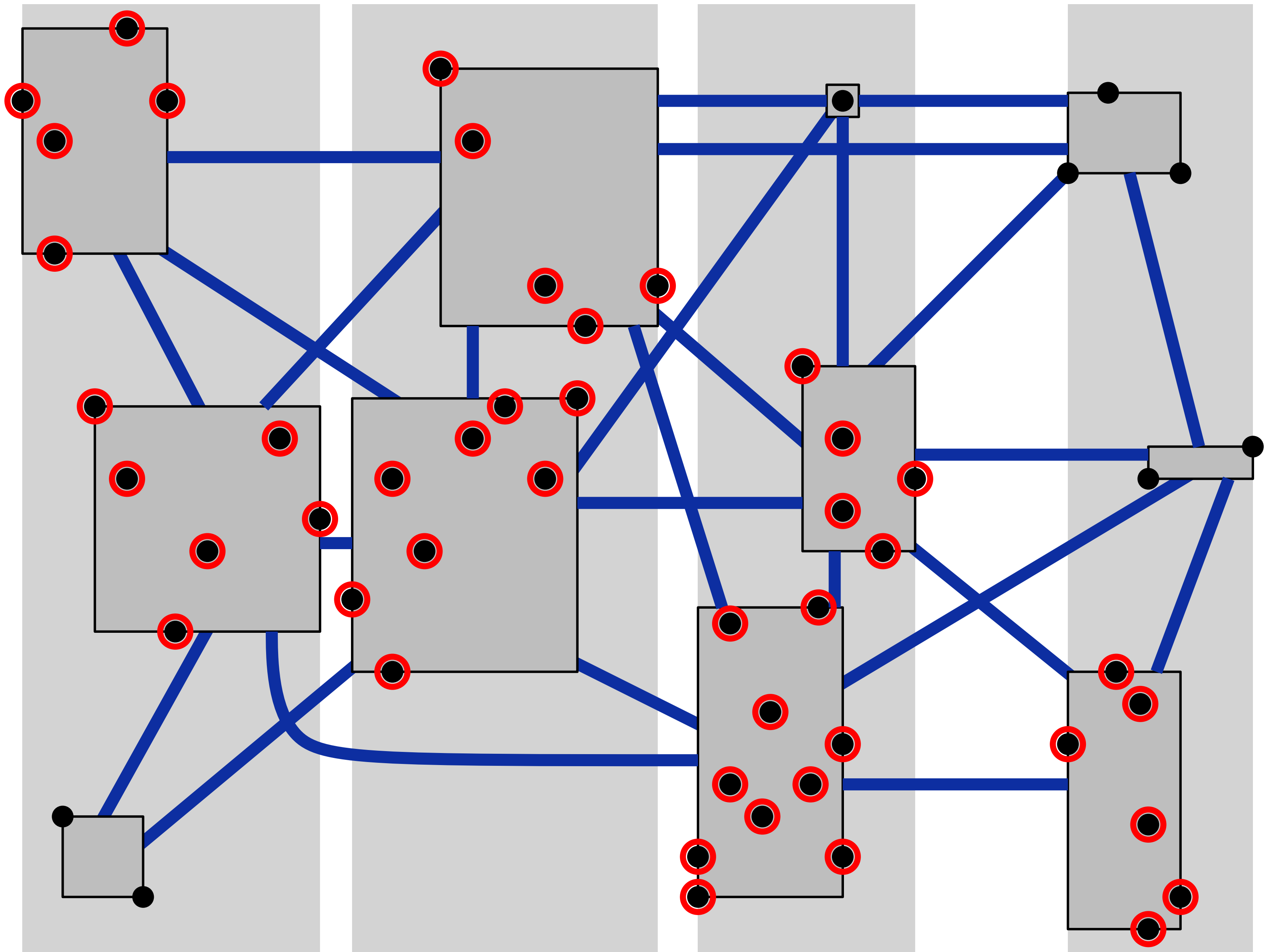
  Other box is crowded:

Box graph $\mathcal{G}_{box}$

$k = 4$

ε:

$\varepsilon/\sqrt{2}$:



2. Find all core points

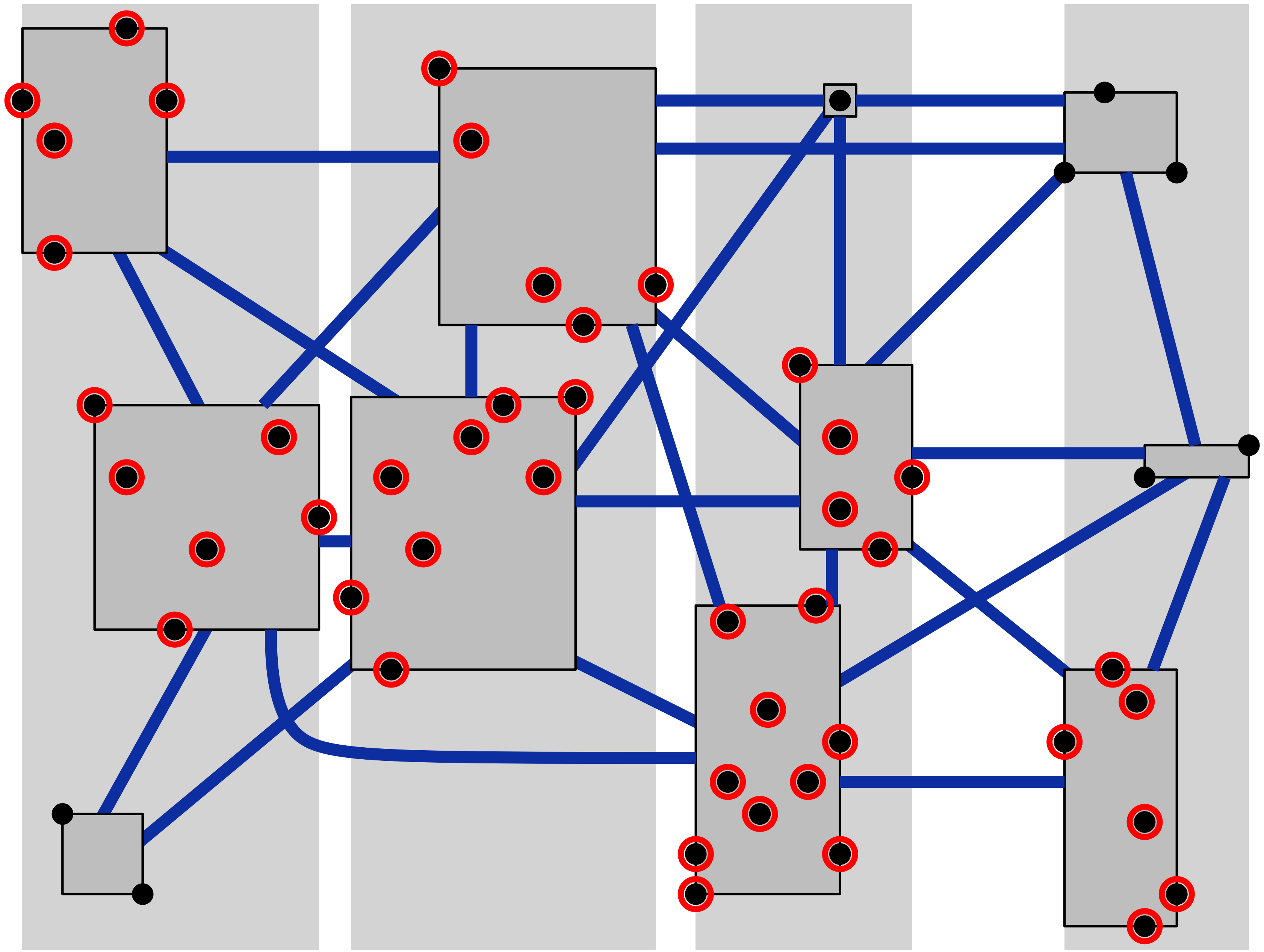
Already have all core points in "crowded" boxes.

For all "sparse" boxes:
  For all neighbour boxes:
    ... check all pairs.

Total runtime?
  Other box is sparse:
    $\mathcal{O}(k^2) = \mathcal{O}(1)$
  Other box is crowded:
    Charge to crowded box

Box graph $\mathcal{G}_{box}$

$k = 4$ ε: $\varepsilon/\sqrt{2}$:



2. Find all core points

Already have all core points in “crowded” boxes.

For all “sparse” boxes:
  For all neighbour boxes:
    ... check all pairs.

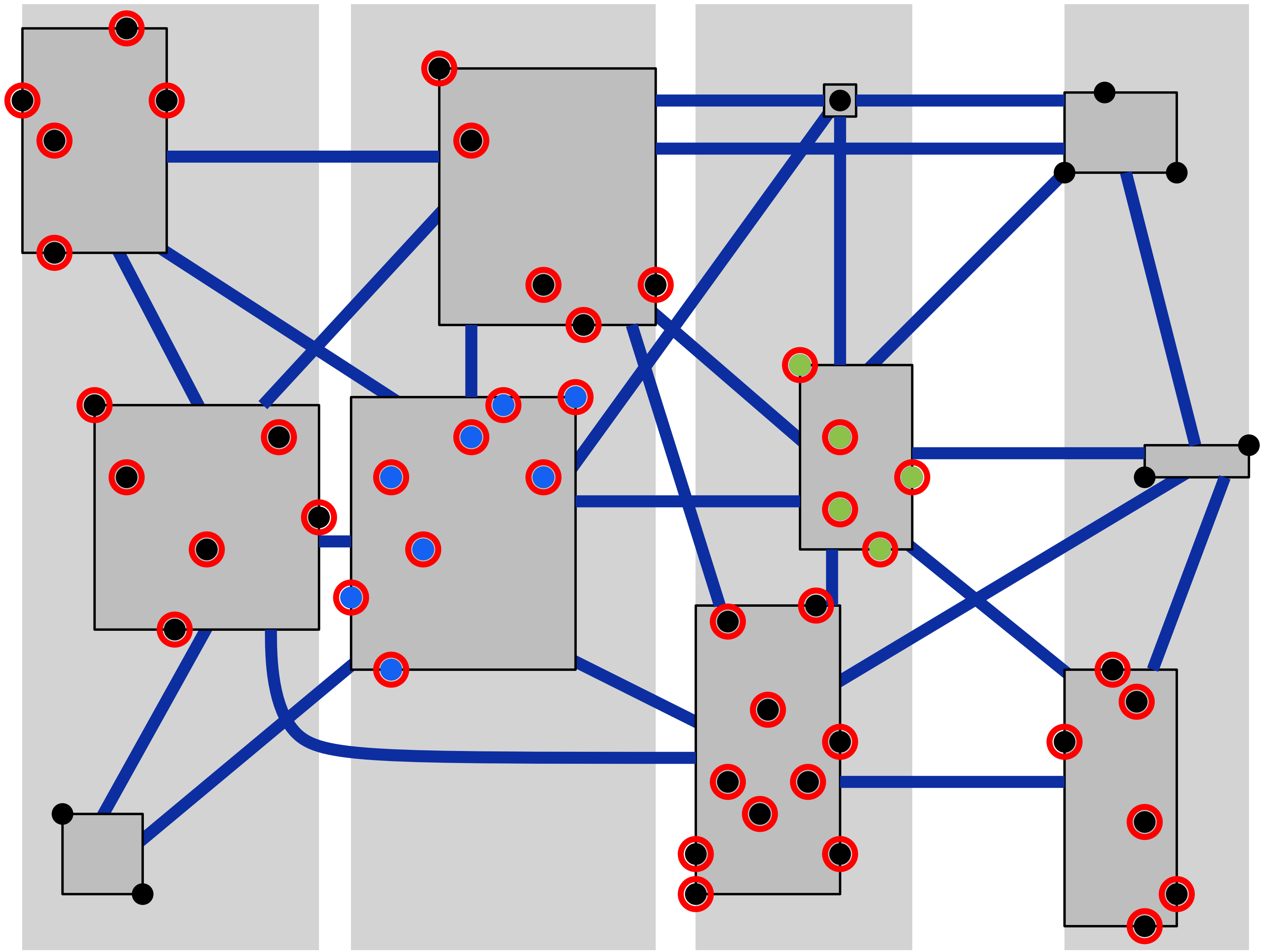
Total runtime?
  Other box is sparse:
    $\mathcal{O}(k^2) = \mathcal{O}(1)$
  Other box is crowded:
    Charge to crowded box
    Point in crowded box checked $\leqslant 22k$ times.

Box graph $\mathcal{G}_{\text{box}}$

$k = 4$

ε:

$\varepsilon/\sqrt{2}$:

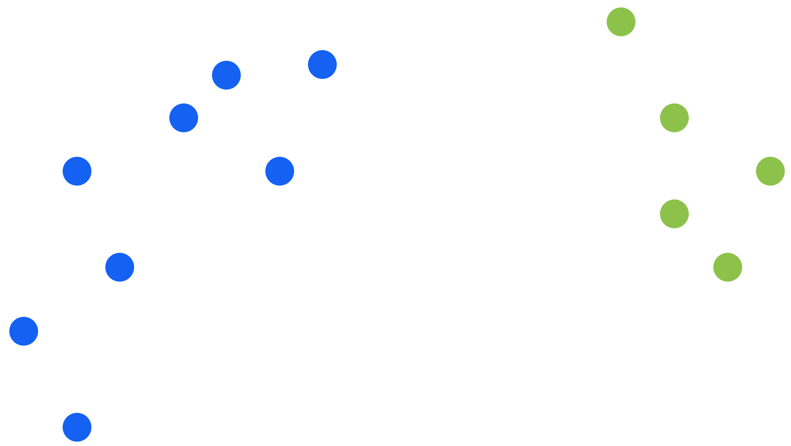


Pairs of crowded boxes

These are all core points.

Are the **the same cluster**?

Box graph $\mathcal{G}_{\text{box}}$

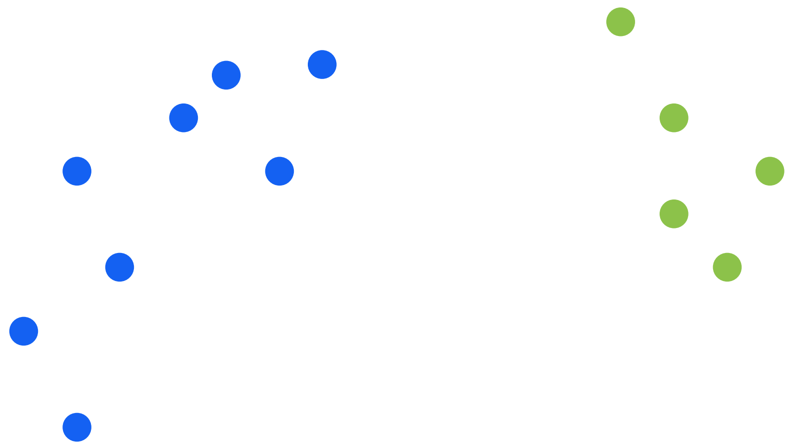


Pairs of crowded boxes

These are all core points.

Are the **the same cluster**?

Box graph $\mathcal{G}_{\text{box}}$



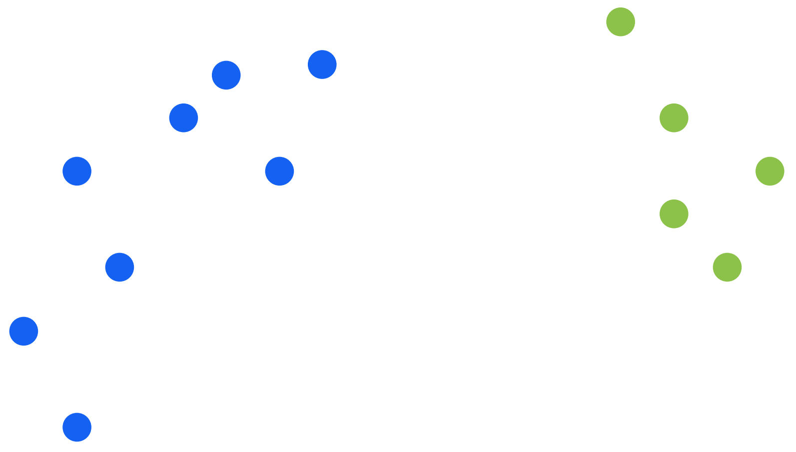
Pairs of crowded boxes

These are all core points.

Are the **the same cluster**?

BICHROMATIC CLOSEST PAIR

Box graph $\mathcal{G}_{\text{box}}$



Pairs of crowded boxes

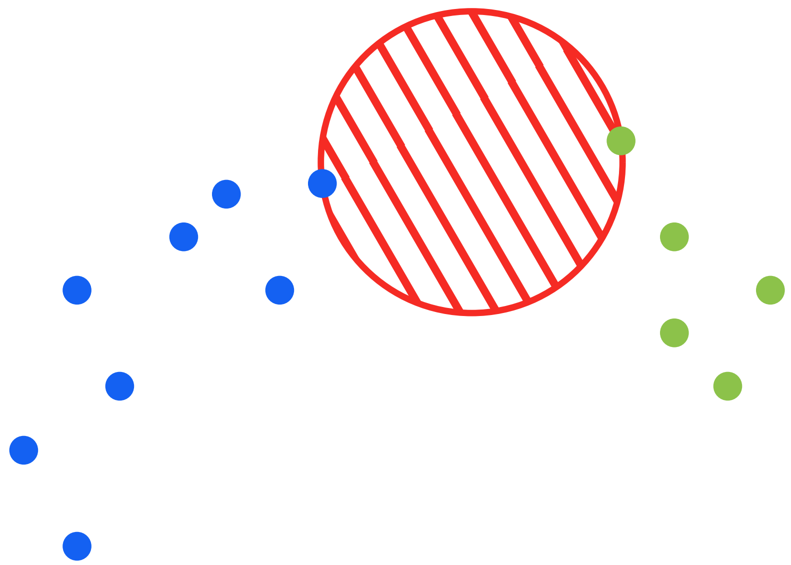
These are all core points.

Are the **the same cluster**?

BICHROMATIC CLOSEST PAIR

In Euclidean 2D?

Box graph $\mathcal{G}_{\text{box}}$



Pairs of crowded boxes

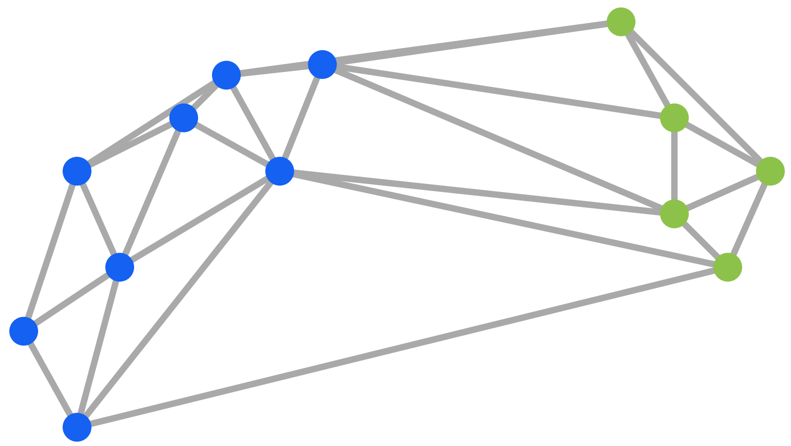
These are all core points.

Are the **the same cluster**?

BICHROMATIC CLOSEST PAIR

In Euclidean 2D?

Box graph $\mathcal{G}_{box}$



Pairs of crowded boxes

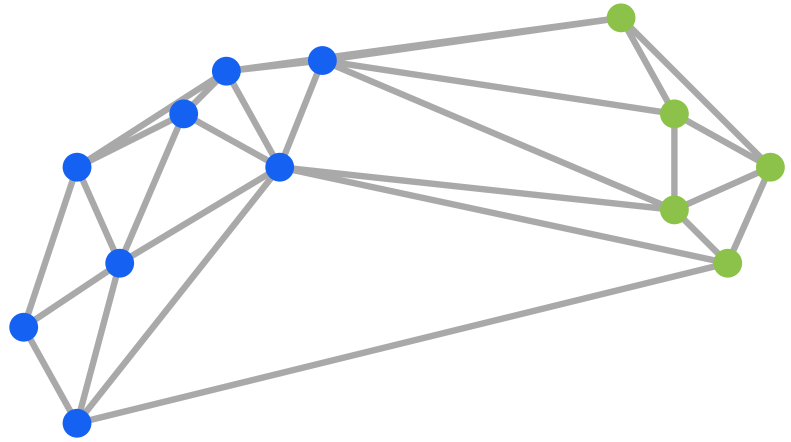
These are all core points.

Are the **the same cluster**?

BICHROMATIC CLOSEST PAIR

In Euclidean 2D?

Box graph $\mathcal{G}_{\text{box}}$



Pairs of crowded boxes

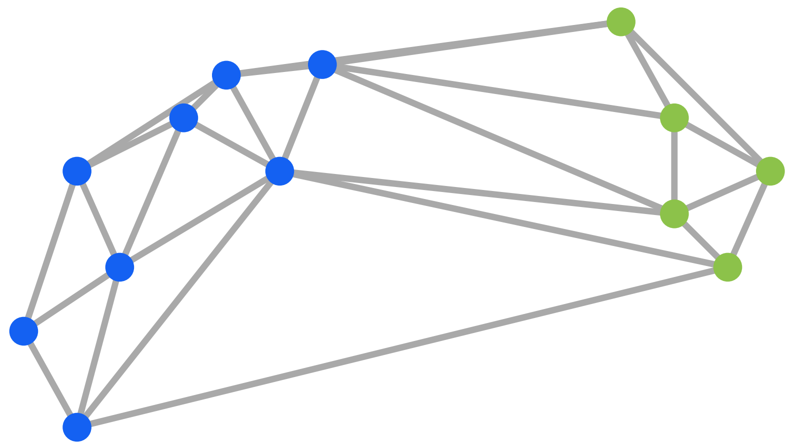
These are all core points.

Are the **the same cluster**?

BICHROMATIC CLOSEST PAIR

In Euclidean 2D?
Delaunay triangulation has this edge.

Box graph $\mathcal{G}_{\text{box}}$



Pairs of crowded boxes

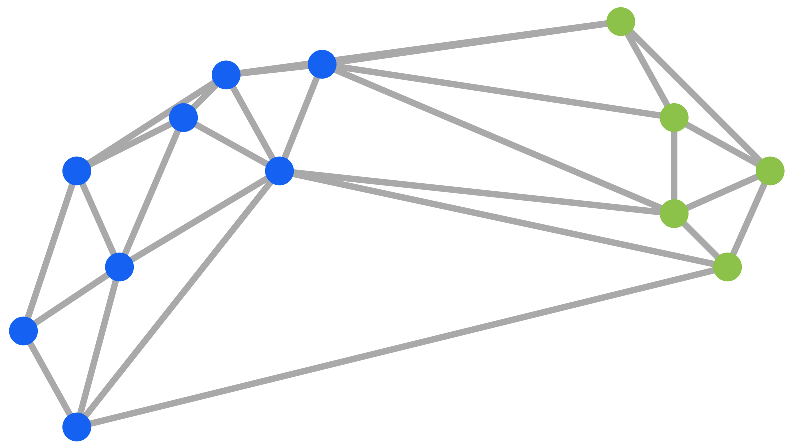
These are all core points.

Are the **the same cluster**?

BICHROMATIC CLOSEST PAIR

In Euclidean 2D?
Delaunay triangulation has this edge.
$\mathcal{O}(n \log n)$

Box graph $\mathcal{G}_{\text{box}}$



Pairs of crowded boxes

These are all core points.

Are the **the same cluster**?

BICHROMATIC CLOSEST PAIR

In Euclidean 2D?
Delaunay triangulation has this edge.
$\mathcal{O}(n \log n)$

Charge to edges in $\mathcal{G}_{\text{box}}$

Results

Everywhere: ε free, k fixed constant, Euclidean distances

	2D	dD
DBSCAN	$\mathcal{O}(n \log n)$	$\mathcal{O}(n^{2-\frac{2}{\lceil d/2 \rceil+1}+\gamma})$ $\gamma > 0$
HDBSCAN	$\mathcal{O}(n \log n)$ expected	$\times$

Results

Everywhere: ε free, k fixed constant, Euclidean distances

	2D	dD
DBSCAN	$\mathcal{O}(n \log n)$	$\mathcal{O}(n^{2-\frac{2}{\lceil d/2 \rceil+1}+\gamma})$ $\gamma > 0$
HDBSCAN	$\mathcal{O}(n \log n)$ expected	$\times$

1. Construct $\mathcal{G}_{\text{box}}$
2. Find core points
3. Merge clusters

(4. Assign border points.)

Results

Everywhere: ε free, k fixed constant, Euclidean distances

	2D	dD
DBSCAN	$\mathcal{O}(n \log n)$	$\mathcal{O}(n^{2-\frac{2}{\lceil d/2 \rceil+1}+\gamma})$ $\gamma > 0$
HDBSCAN	$\mathcal{O}(n \log n)$ expected	$\times$

1. Construct $\mathcal{G}_{\text{box}}$
2. Find core points
3. Merge clusters

(4. Assign border points.)

BICHROMATIC CLOSEST POINT instead of Delauney triangulation.

Results

Everywhere: ε free, k fixed constant, Euclidean distances

	2D	dD	
DBSCAN	$\mathcal{O}(n \log n)$	$\mathcal{O}(n^{2-\frac{2}{\lceil d/2 \rceil+1}+\gamma})$	$\gamma > 0$
HDBSCAN	$\mathcal{O}(n \log n)$ expected	$\times$	

1. Construct $\mathcal{G}_{\text{box}}$
2. Find core points
3. Merge clusters

(4. Assign border points.)

BICHROMATIC CLOSEST POINT instead of Delauney triangulation.

(Agarwal, Edelsbrunner, Schwarzkopf, 1991)

Results

Everywhere: ε free, k fixed constant, Euclidean distances

	2D	dD
DBSCAN	$\mathcal{O}(n \log n)$	$\mathcal{O}(n^{2-\frac{2}{\lceil d/2 \rceil+1}+\gamma})$ $\gamma > 0$
HDBSCAN	$\mathcal{O}(n \log n)$ expected	$\times$

1. Construct $\mathcal{G}_{\text{box}}$
2. Find core points
3. Merge clusters

(4. Assign border points.)

BICHROMATIC CLOSEST POINT instead of Delauney triangulation.

(Agarwal, Edelsbrunner, Schwarzkopf, 1991)

HDBSCAN

Use DBSCAN* and sweep ε from 0 to ∞.

HDBSCAN

Use DBSCAN* and sweep ε from 0 to ∞.

Initially all points are noise; eventually everything is one cluster.

Three types of "events."

HDBSCAN

Use DBSCAN* and sweep ε from 0 to ∞.

Initially all points are noise; eventually everything is one cluster.

Three types of “events.”

- Noise point becomes core point. Call this value $d_{core}(p)$.

HDBSCAN

Use DBSCAN* and sweep ε from 0 to ∞.

Initially all points are noise; eventually everything is one cluster.

Three types of “events.”

- Noise point becomes core point. Call this value $d_{core}(p)$.
- New cluster forms.

HDBSCAN

Use DBSCAN* and sweep ε from 0 to ∞.

Initially all points are noise; eventually everything is one cluster.

Three types of "events."

- Noise point becomes core point. Call this value $d_{core}(p)$.
- New cluster forms.
- Two clusters merge

HDBSCAN

Use DBSCAN* and sweep ε from 0 to ∞.

Initially all points are noise; eventually everything is one cluster.

Three types of "events."

- Noise point becomes core point. Call this value $\mathbf{d}_{\mathbf{core}}(\mathbf{p})$.
- New cluster forms.
- Two clusters merge

Events only happen when $\varepsilon = \mathbf{d}(\mathbf{p}, \mathbf{q})$ for some $\mathbf{p}, \mathbf{q}$.

HDBSCAN

Use DBSCAN* and sweep ε from 0 to ∞.

Initially all points are noise; eventually everything is one cluster.

Three types of "events."

- Noise point becomes core point. Call this value $\mathbf{d}_{\mathbf{core}}(\mathbf{p})$.
- New cluster forms.
- Two clusters merge

Events only happen when $\varepsilon = \mathbf{d}(\mathbf{p}, \mathbf{q})$ for some $\mathbf{p}, \mathbf{q}$.

Store this tree structure of cluster creation and merges: HDBSCAN.

Mutual reachability

Starting at which value of ε will these points be in the same cluster?

Mutual reachability

Starting at which value of ε will these points be in the same cluster?

Both need to be core point, so at least $\mathbf{d}_{\text{core}}(\mathbf{p})$ and $\mathbf{d}_{\text{core}}(\mathbf{q})$.

Mutual reachability

Starting at which value of ε will these points be in the same cluster?

Both need to be core point, so at least $\mathbf{d}_{\text{core}}(\mathbf{p})$ and $\mathbf{d}_{\text{core}}(\mathbf{q})$.

Either $\varepsilon \geqslant \mathbf{d}(\mathbf{p}, \mathbf{q})$, or they must be connected through other points.

Mutual reachability

Starting at which value of ε will these points be in the same cluster?

Both need to be core point, so at least $d_{\text{core}}(p)$ and $d_{\text{core}}(q)$.
Either $\varepsilon \geqslant d(p, q)$, or they must be connected through other points.

Def. Let $d_{\text{mr}}(p, q) = \max\{\, d_{\text{core}}(p),\ d_{\text{core}}(q),\ d(p, q)\,\}$.

Mutual reachability

Starting at which value of ε will these points be in the same cluster?

Both need to be core point, so at least $d_{\text{core}}(p)$ and $d_{\text{core}}(q)$.
Either $\varepsilon \geqslant d(p, q)$, or they must be connected through other points.

Def. Let $d_{\text{mr}}(p, q) = \textbf{max}\{\ d_{\text{core}}(p),\ d_{\text{core}}(q),\ d(p, q)\ \}$.

Def. Mutual reachability graph $\mathcal{G}_{\text{mr}}$: complete, edge weights d_{mr}.

Mutual reachability

Starting at which value of ε will these points be in the same cluster?

Both need to be core point, so at least $d_{\text{core}}(p)$ and $d_{\text{core}}(q)$.
Either $\varepsilon \geqslant d(p, q)$, or they must be connected through other points.

Def. Let $d_{\text{mr}}(p, q) = \max\{\, d_{\text{core}}(p),\ d_{\text{core}}(q),\ d(p, q) \,\}$.

Def. Mutual reachability graph $\mathcal{G}_{\text{mr}}$: complete, edge weights d_{mr}.

Algorithm: 1. Compute d_{core} for all points.

Mutual reachability

Starting at which value of ε will these points be in the same cluster?

Both need to be core point, so at least $d_{\text{core}}(p)$ and $d_{\text{core}}(q)$.
Either $\varepsilon \geqslant d(p, q)$, or they must be connected through other points.

Def. Let $d_{\text{mr}}(p, q) = \textbf{max}\{\, d_{\text{core}}(p),\; d_{\text{core}}(q),\; d(p, q)\,\}$.

Def. Mutual reachability graph $\mathcal{G}_{\text{mr}}$: complete, edge weights d_{mr}.

Algorithm:

1. Compute d_{core} for all points.
2. Construct $\mathcal{G}_{\text{mr}}$ and compute a minimum spanning tree $\mathcal{T}$.

Mutual reachability

Starting at which value of ε will these points be in the same cluster?

Both need to be core point, so at least $d_{\text{core}}(p)$ and $d_{\text{core}}(q)$.
Either $\varepsilon \geqslant d(p, q)$, or they must be connected through other points.

Def. Let $d_{\text{mr}}(p, q) = \textbf{max}\{\, d_{\text{core}}(p),\; d_{\text{core}}(q),\; d(p, q)\,\}$.

Def. Mutual reachability graph $\mathcal{G}_{\text{mr}}$: complete, edge weights d_{mr}.

Algorithm:

1. Compute d_{core} for all points. $\mathcal{O}(\, n \log n\,)$ **time (Vaidya, 1989)**
2. Construct $\mathcal{G}_{\text{mr}}$ and compute a minimum spanning tree $\mathcal{T}$.
3. Convert $\mathcal{T}$ into HDBSCAN tree.

2. Construct $\mathcal{G}_{\text{mr}}$ and compute an MST.

Cannot look at all edges: too slow.

2. Construct $\mathcal{G}_{\mathrm{mr}}$ and compute an MST.

Cannot look at all edges: too slow.

Def. $\{p, q\}$ is a Delaunay edge “iff” there exists a circle with:

- p and q on the boundary
- no points in its interior

2. Construct $\mathcal{G}_{mr}$ and compute an MST.

Cannot look at all edges: too slow.

k^{th}-order

Def. $\{p, q\}$ is a Delaunay edge "iff" there exists a circle with:

- p and q on the boundary
- ~~no~~ $\leqslant k$ points in its interior

a "k-OD edge"

2. Construct $\mathcal{G}_{\text{mr}}$ and compute an MST.

Cannot look at all edges: too slow.

k^{th}-order

Def. $\{p, q\}$ is a Delaunay edge "iff" there exists a circle with:

- p and q on the boundary
- ~~no~~ $\leqslant k$ points in its interior

a "k-OD edge"

Theorem (Gudmundsson, Hammer, v. Kreveld, 2002) The k^{th}-order Delaunay graph has $\mathcal{O}(\,n(k+1)\,)$ edges and can be computed in $\mathcal{O}(\,n(k+1)\,\textbf{log}\, n\,)$ expected time by randomized incremental construction.

2. Construct $\mathcal{G}_{\text{mr}}$ and compute an MST.

Cannot look at all edges: too slow.

k^{th}-order

Def. $\{p, q\}$ is a ~~Delaunay~~ k^{th}-order Delaunay edge "iff" there exists a circle with:

- p and q on the boundary
- ~~no~~ $\leqslant k$ points in its interior

a "k-OD edge"

Theorem (Gudmundsson, Hammer, v. Kreveld, 2002) The k^{th}-order Delaunay graph has $\mathcal{O}(\, n(k+1)\,)$ edges and can be computed in $\mathcal{O}(\, n(k+1) \log n\,)$ expected time by randomized incremental construction.

Claim: The MST of $\mathcal{G}_{\text{mr}}$ uses only k-OD edges.

The MST of $\mathcal{G}_{\mathrm{mr}}$ uses only k-OD edges.

Consider applying Kruskal's algorithm to $\mathcal{G}_{\mathrm{mr}}$:

- Looks at edges in order of increasing cost.
- With weights $\mathbf{d}_{\mathrm{mr}}$ this corresponds to the HDBSCAN events.

The MST of $\mathcal{G}_{\text{mr}}$ uses only k-OD edges.

Consider applying Kruskal's algorithm to $\mathcal{G}_{\text{mr}}$:

- Looks at edges in order of increasing cost.
- With weights d_{mr} this corresponds to the HDBSCAN events.

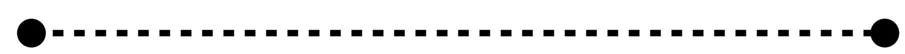
Claim: Whenever Kruskal looks at a non-k-OD edge $\{p, q\}$,
p and q are already in the same cluster, and thus ignores the edge.

The MST of $\mathcal{G}_{\mathrm{mr}}$ uses only k-OD edges.

Consider applying Kruskal's algorithm to $\mathcal{G}_{\mathrm{mr}}$:

- Looks at edges in order of increasing cost.
- With weights d_{mr} this corresponds to the HDBSCAN events.

Claim: Whenever Kruskal looks at a non-k-OD edge $\{p, q\}$,
p and q are already in the same cluster, and thus ignores the edge.

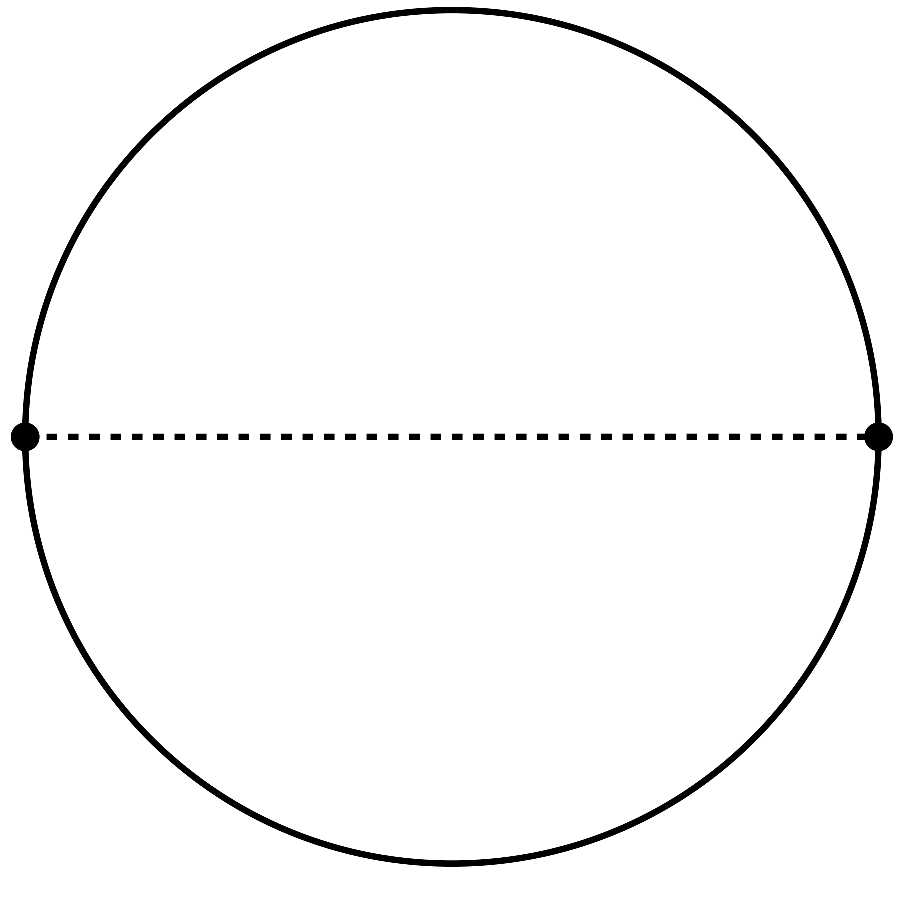


The MST of $\mathcal{G}_{\text{mr}}$ uses only k-OD edges.

Consider applying Kruskal's algorithm to $\mathcal{G}_{\text{mr}}$:

- Looks at edges in order of increasing cost.
- With weights d_{mr} this corresponds to the HDBSCAN events.

Claim: Whenever Kruskal looks at a non-k-OD edge $\{p, q\}$, p and q are already in the same cluster, and thus ignores the edge.



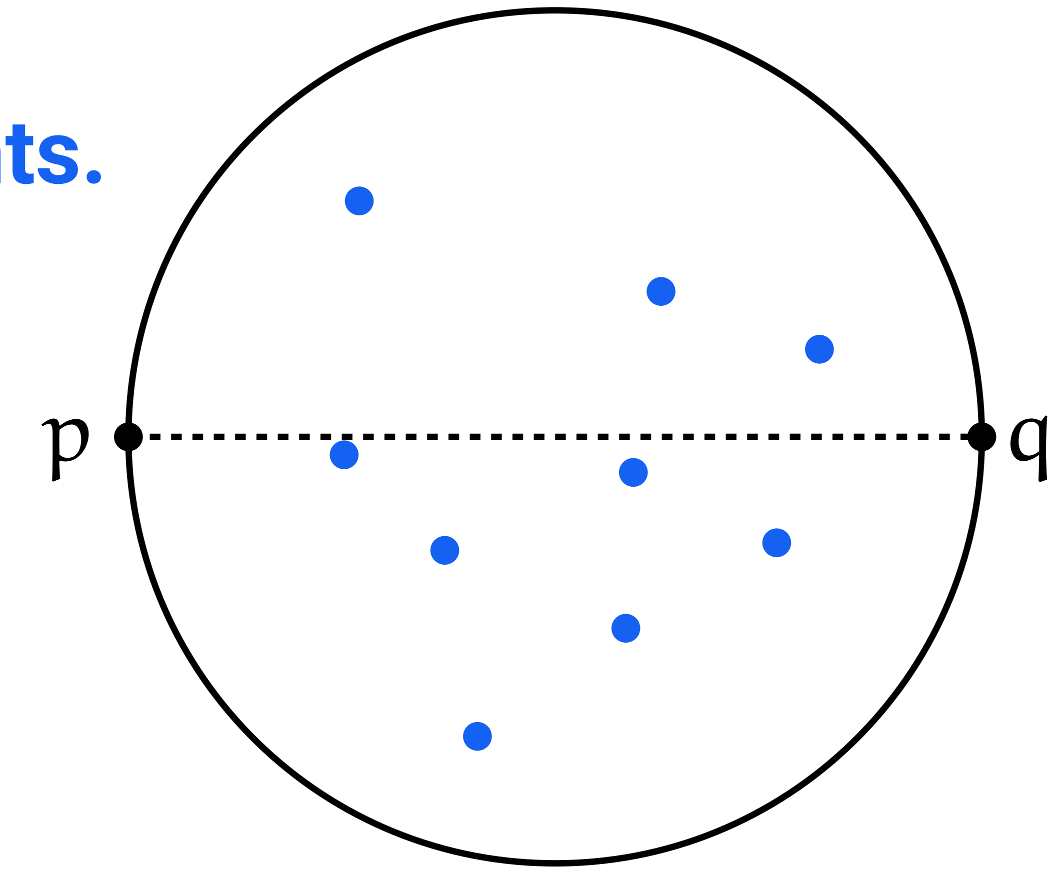
The MST of $\mathcal{G}_{\text{mr}}$ uses only k-OD edges.

Consider applying Kruskal's algorithm to $\mathcal{G}_{\text{mr}}$:

- Looks at edges in order of increasing cost.
- With weights d_{mr} this corresponds to the HDBSCAN events.

Claim: Whenever Kruskal looks at a non-k-OD edge $\{p, q\}$, p and q are already in the same cluster, and thus ignores the edge.

Not a k-OD edge, so more than k points.



The MST of $\mathcal{G}_{mr}$ uses only k-OD edges.

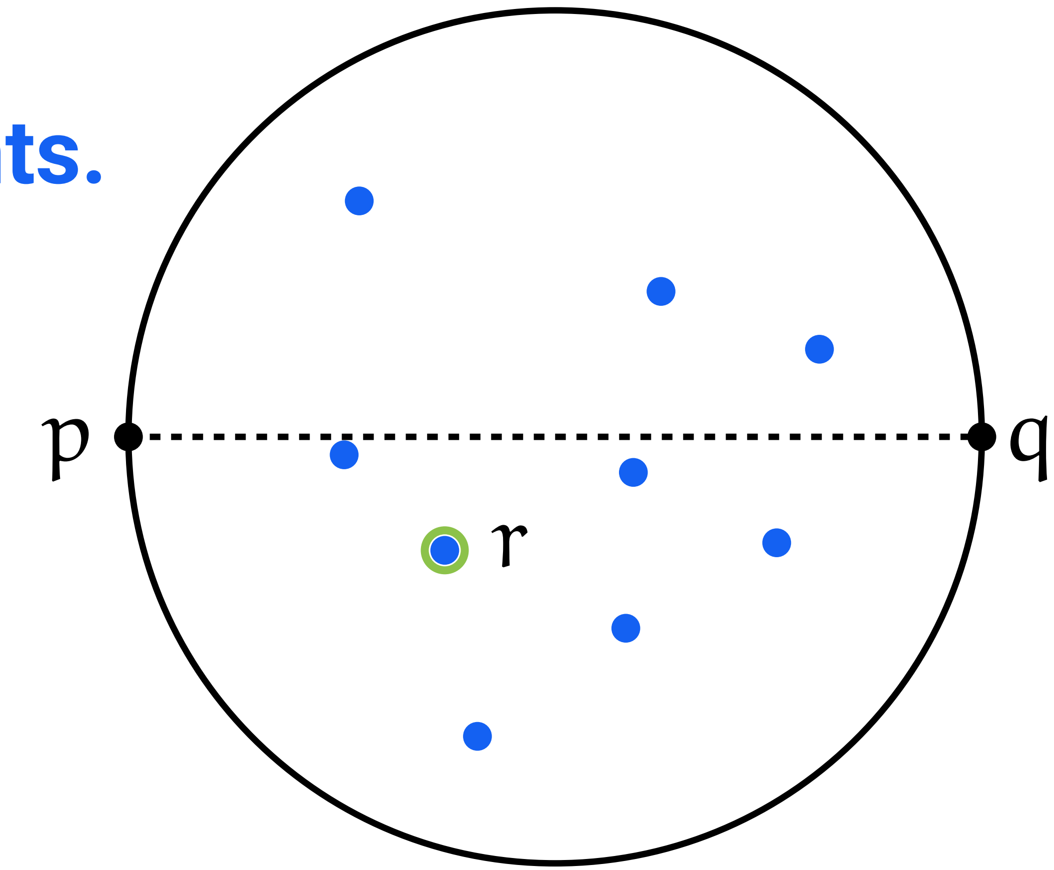
Consider applying Kruskal's algorithm to $\mathcal{G}_{mr}$:

- Looks at edges in order of increasing cost.
- With weights d_{mr} this corresponds to the HDBSCAN events.

Claim: Whenever Kruskal looks at a non-k-OD edge $\{p, q\}$,
p and q are already in the same cluster, and thus ignores the edge.

Not a k-OD edge, so more than k points.

Pick any point.



The MST of $\mathcal{G}_{\text{mr}}$ uses only k-OD edges.

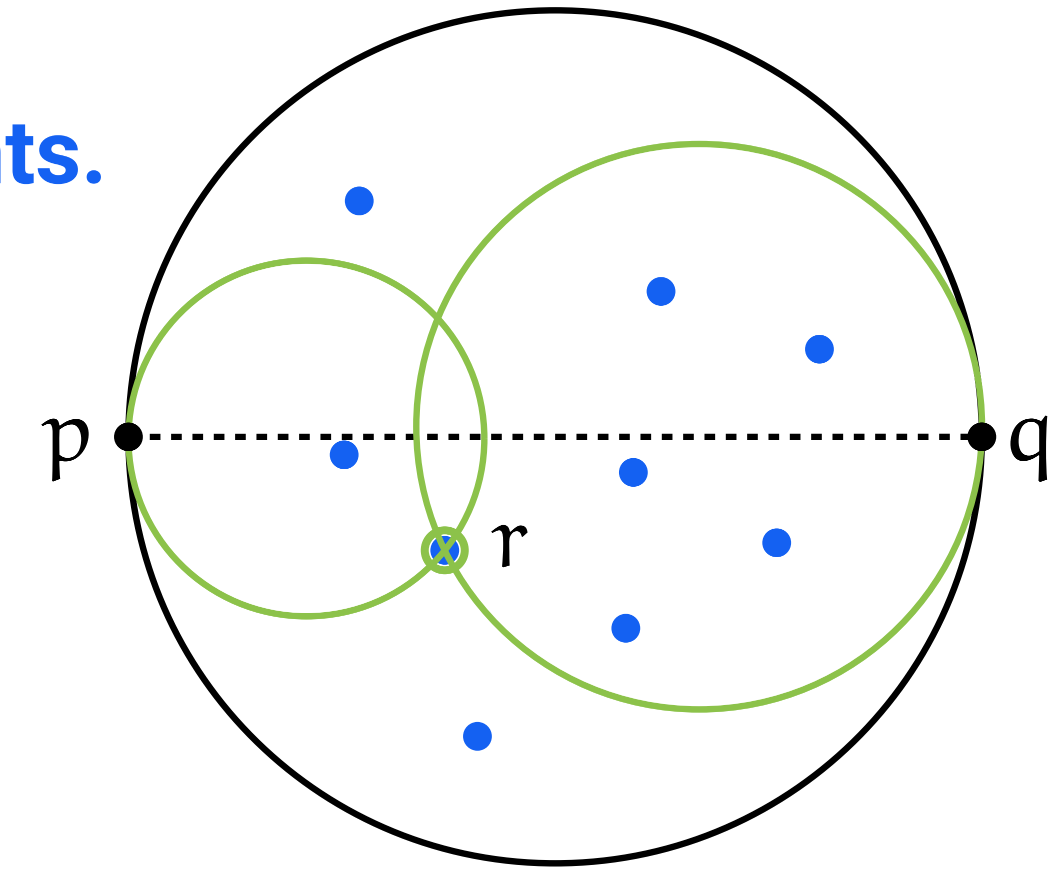
Consider applying Kruskal's algorithm to $\mathcal{G}_{\text{mr}}$:

- Looks at edges in order of increasing cost.
- With weights d_{mr} this corresponds to the HDBSCAN events.

Claim: Whenever Kruskal looks at a non-k-OD edge $\{p, q\}$, p and q are already in the same cluster, and thus ignores the edge.

Not a k-OD edge, so more than k points.

Pick any point.



The MST of $\mathcal{G}_{\mathrm{mr}}$ uses only k-OD edges.

Consider applying Kruskal's algorithm to $\mathcal{G}_{\mathrm{mr}}$:

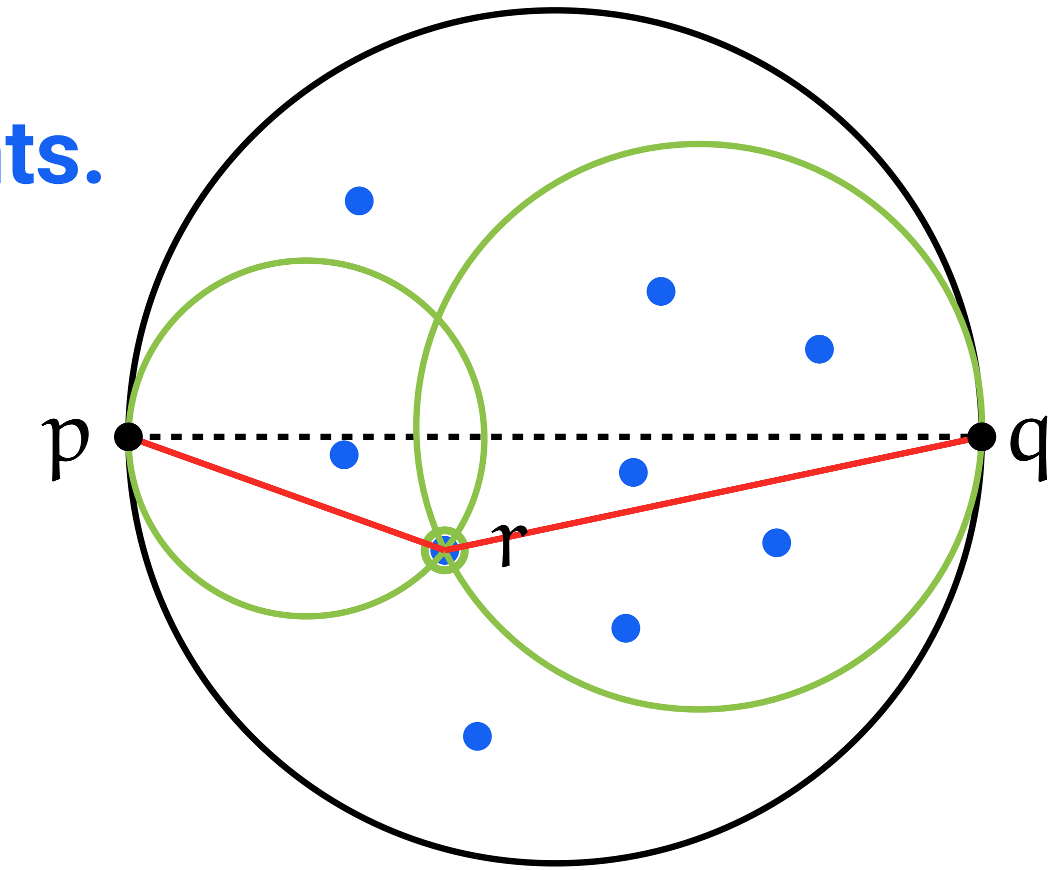
- Looks at edges in order of increasing cost.
- With weights d_{mr} this corresponds to the HDBSCAN events.

Claim: Whenever Kruskal looks at a non-k-OD edge $\{p, q\}$, p and q are already in the same cluster, and thus ignores the edge.

Not a k-OD edge, so more than k points.

Pick any point.

Recurse until only k-OD edges.



The MST of $\mathcal{G}_{\text{mr}}$ uses only k-OD edges.

Consider applying Kruskal's algorithm to $\mathcal{G}_{\text{mr}}$:

- Looks at edges in order of increasing cost.
- With weights d_{mr} this corresponds to the HDBSCAN events.

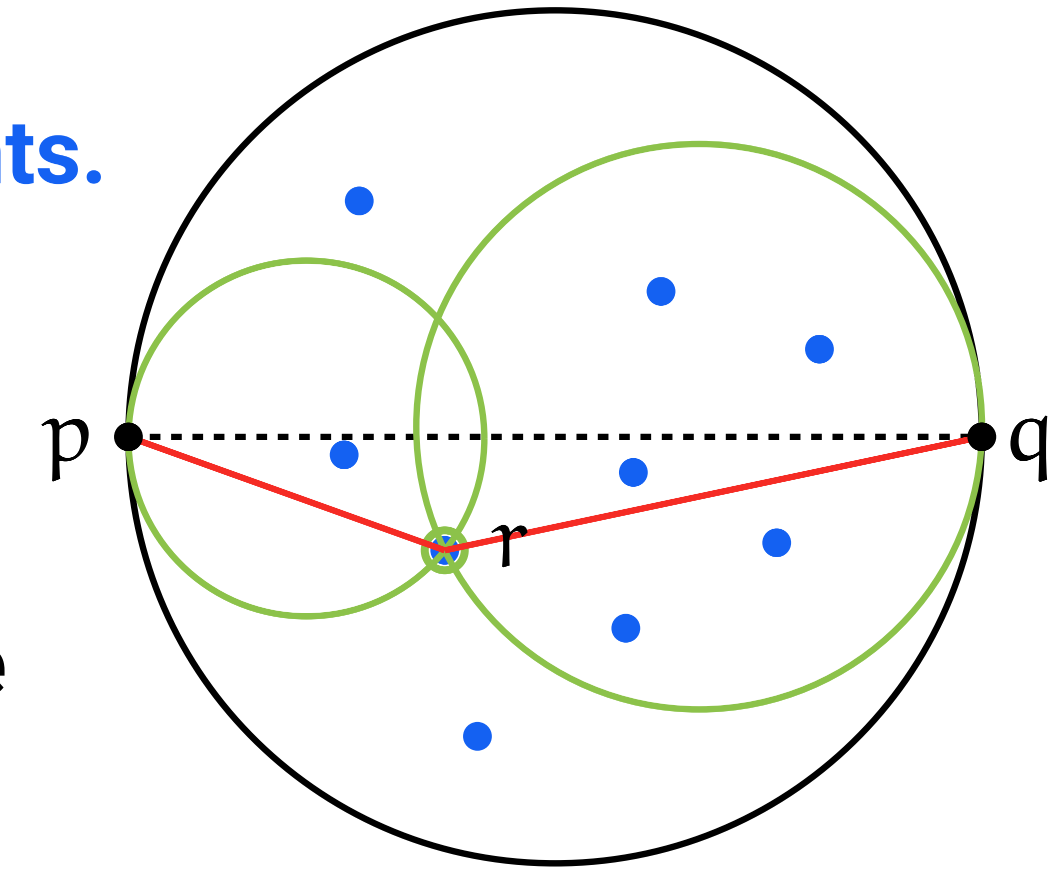
Claim: Whenever Kruskal looks at a non-k-OD edge $\{p, q\}$, p and q are already in the same cluster, and thus ignores the edge.

Not a k-OD edge, so more than k points.

Pick any point.

Recurse until only k-OD edges.

Kruskal has already considered those edges, so p and q already connected.



Results

Everywhere: ε free, k fixed constant, Euclidean distances

	2D	dD
DBSCAN	$\mathcal{O}(n \log n)$	$\mathcal{O}(n^{2-\frac{2}{\lceil d/2 \rceil+1}+\gamma})$ $\gamma > 0$
HDBSCAN	$\mathcal{O}(n \log n)$ expected	$\times$