The power of grids

Geometric Approximation Algorithms

Overview

techniques

grids

randomization and backward analysis

geometric problems

closest pair

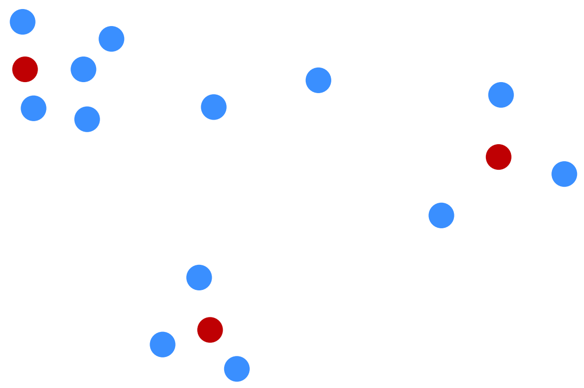
smallest disk enclosing k points

cluster radius (exercise)

Cluster Radius

Exercise 1.2.A from book:

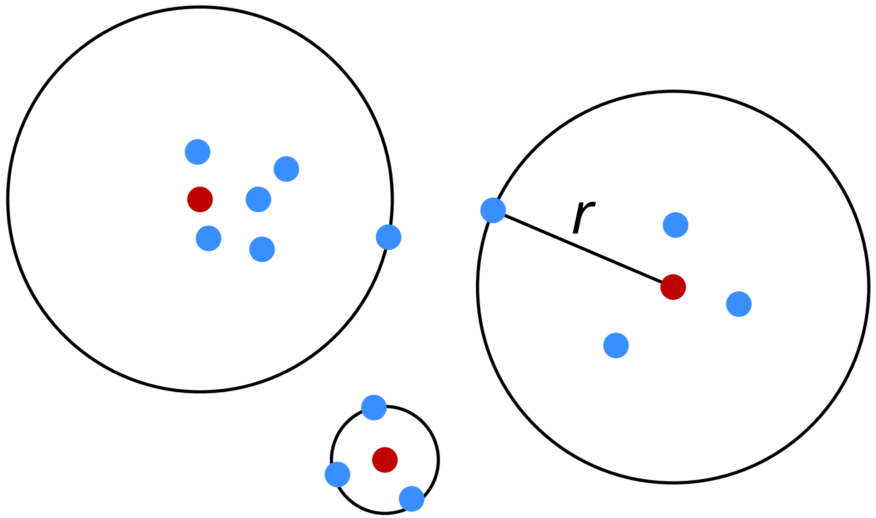
Let C and P be two given sets of points in the plane, such that $k = |C|$ and $n = |P|$. Let $r = \max_{p \in P} \min_{c \in C} \|c - p\|$ be the covering radius of P by C.



Cluster Radius

Exercise 1.2.A from book:

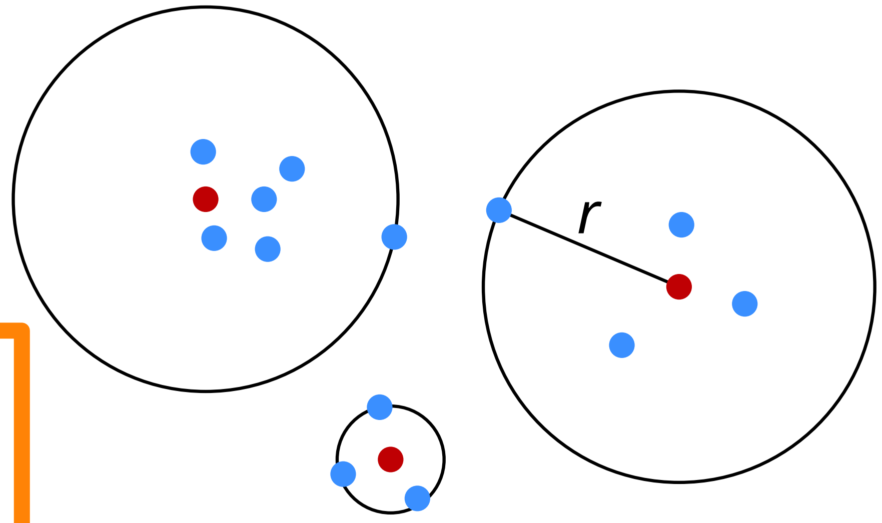
Let C and P be two given sets of points in the plane, such that $k = |C|$ and $n = |P|$. Let $r = \max_{p \in P} \min_{c \in C} \|c - p\|$ be the covering radius of P by C.



Cluster Radius

Exercise 1.2.A from book:

Let C and P be two given sets of points in the plane, such that $k = |C|$ and $n = |P|$. Let $r = \max_{p \in P} \min_{c \in C} \|c - p\|$ be the covering radius of P by C.



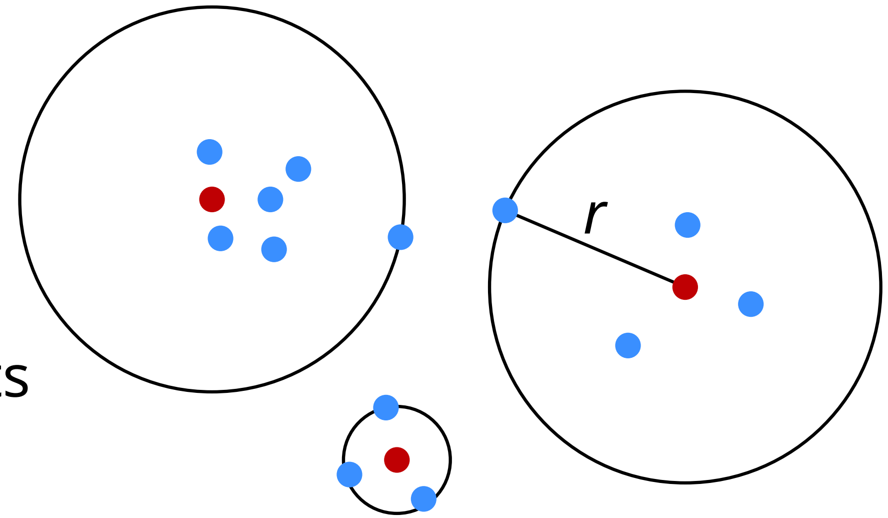
Question:

How fast can we compute the clustering radius?

Cluster Radius

Exercise 1.2.A from book:

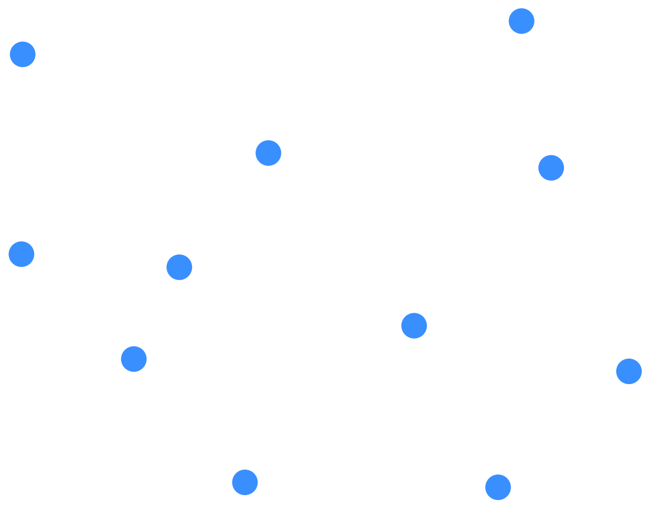
Let C and P be two given sets of points in the plane, such that $k = |C|$ and $n = |P|$. Let $r = \max_{p \in P} \min_{c \in C} \|c - p\|$ be the covering radius of P by C.



Give an $O(n + k \log n)$ expected time algorithm that outputs a number α, such that $r \leq \alpha \leq 10r$.

Closest Pair Problem

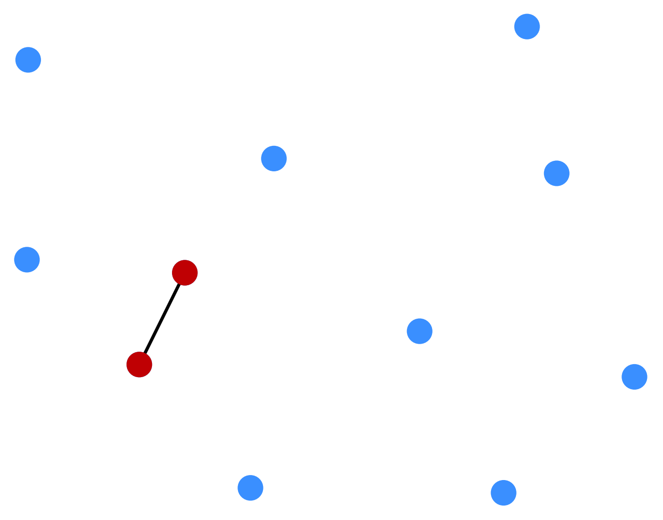
Input: point set *P* in the plane



Closest Pair Problem

Input: point set P in the plane

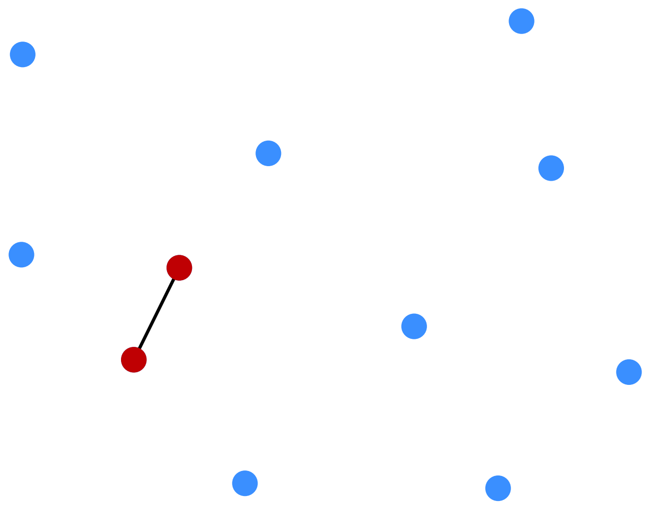
Output: $p, q \in P$, $(p \neq q)$, minimizing $\|p - q\|$



Closest Pair Problem

Input: point set P in the plane

Output: $p, q \in P$, $(p \neq q)$, minimizing $\|p - q\|$



Motivation

- Fundamental problem in Computational Geometry
- Applications in Geographic Information Systems,
  e.g., find closest airplanes for air traffic control
- Subroutine in other algorithms, e.g., for clustering or matching
- Computing closest pair with grids instrumental for field of
  randomized algorithms

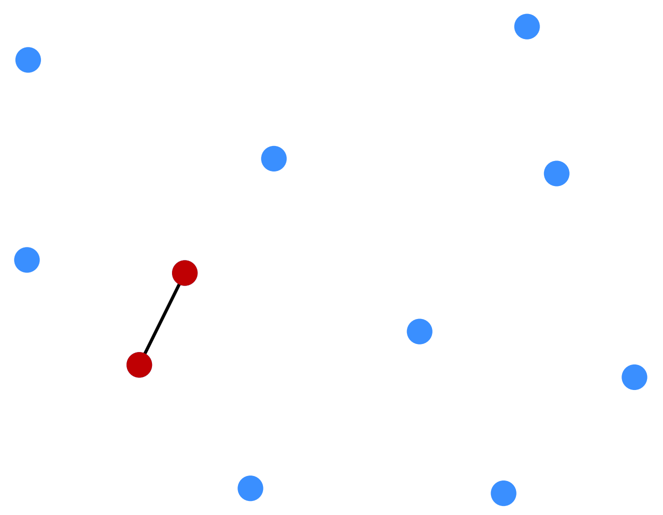
Closest Pair Problem

Input: point set P in the plane

Output: $p, q \in P$, $(p \neq q)$, minimizing $\|p - q\|$

Question:

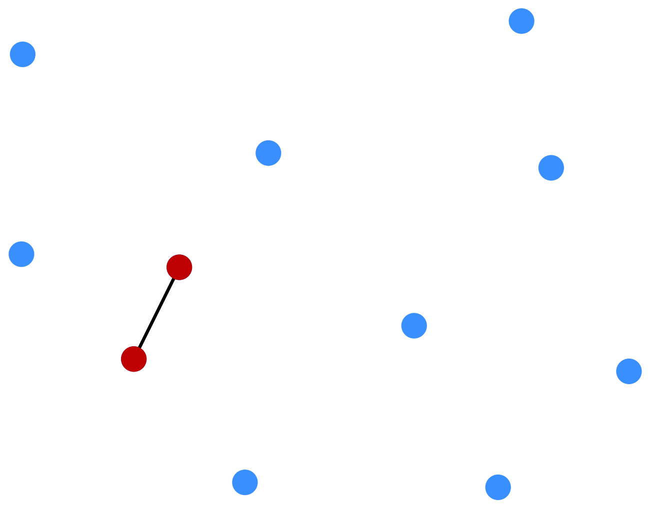
How fast can we compute the closest pair?



Closest Pair Problem

Input: point set P in the plane

Output: $p, q \in P$, $(p \neq q)$, minimizing $\|p - q\|$



Question:

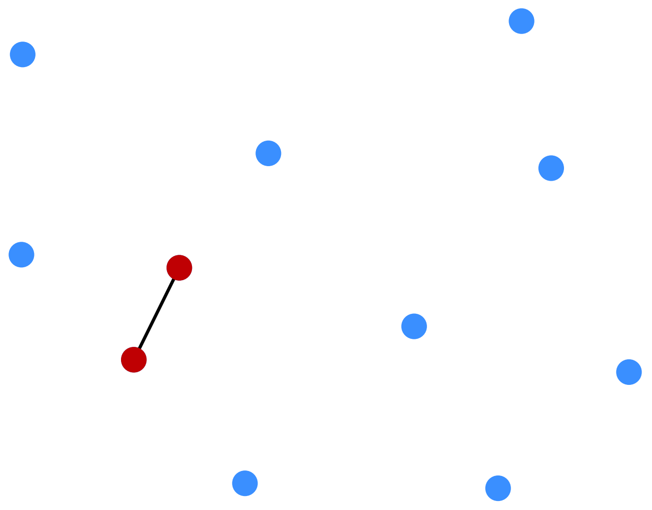
How fast can we compute the closest pair?

simple: $\Theta(n^2)$,   with geometric techniques: $\Theta(n \log n)$

Closest Pair Problem

Input: point set P in the plane

Output: $p, q \in P$, $(p \neq q)$, minimizing $\|p - q\|$



Question:

How fast can we compute the closest pair?

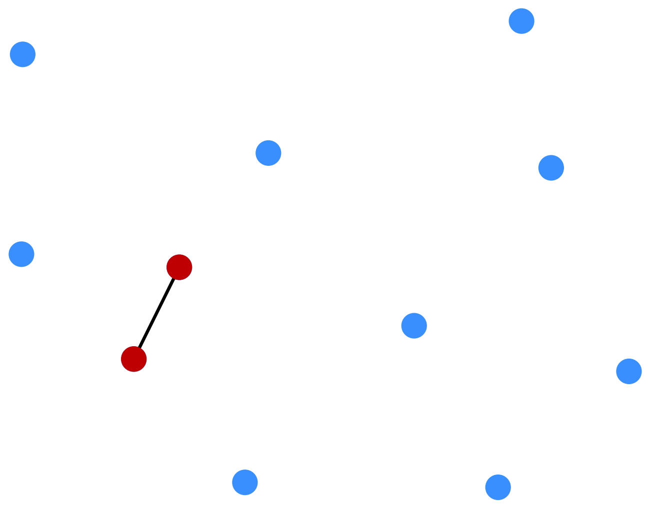
simple: $\Theta(n^2)$, with geometric techniques: $\Theta(n \log n)$

Lower bound: $\Omega(n \log n)$ (algebraic computation tree model of computation)

Closest Pair Problem

Input: point set P in the plane

Output: $p, q \in P$, $(p \neq q)$, minimizing $\|p - q\|$



Question:

How fast can we compute the closest pair?

simple: $\Theta(n^2)$, with geometric techniques: $\Theta(n \log n)$

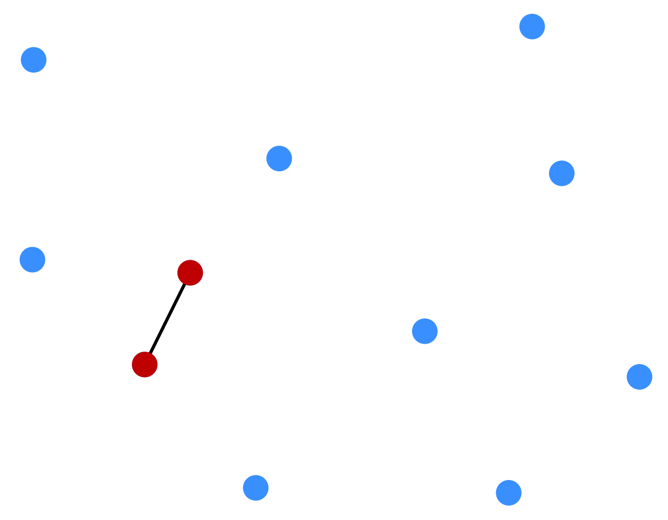
Lower bound: $\Omega(n \log n)$ (algebraic computation tree model of computation)

Our model: real RAM + $O(1)$-time floor (+ *log*-function) + hashing + randomization
(Warning: floor function dangerous from perspective of complexity theory)

Closest Pair Problem

Input: point set P in the plane

Output: $p, q \in P$, $(p \neq q)$, minimizing $\|p - q\|$



Question:

How fast can we compute the closest pair?

simple: $\Theta(n^2)$,   with geometric techniques: $\Theta(n \log n)$

Lower bound: $\Omega(n \log n)$ (algebraic computation tree model of computation)

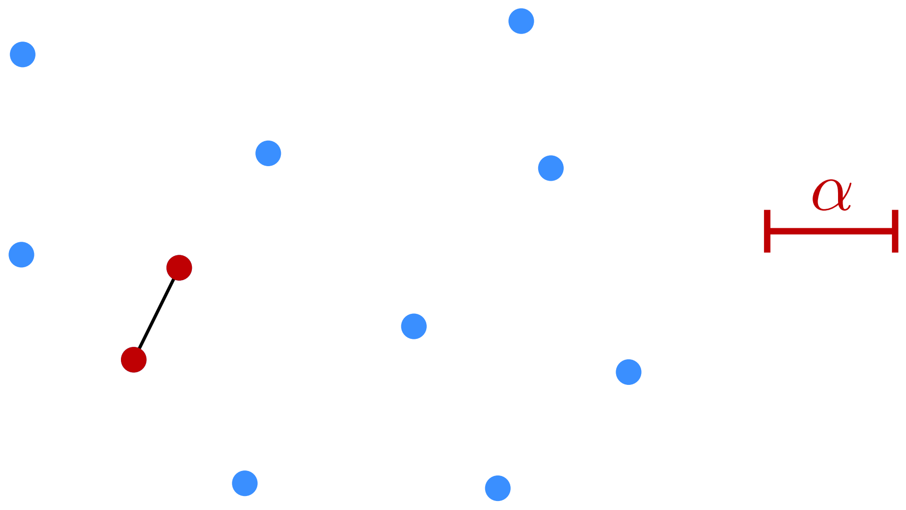
Our model: real RAM + $O(1)$-time floor (+ *log*-function) + hashing + randomization
(Warning: floor function dangerous from perspective of complexity theory)

Next: $O(n)$-time algorithm using grids + randomization

Closest Pair Problem

Input: point set P in the plane

Output: $p, q \in P$, $(p \neq q)$, minimizing $\|p - q\|$



Question:

How fast can we compute the closest pair?

simple: $\Theta(n^2)$, with geometric techniques: $\Theta(n \log n)$

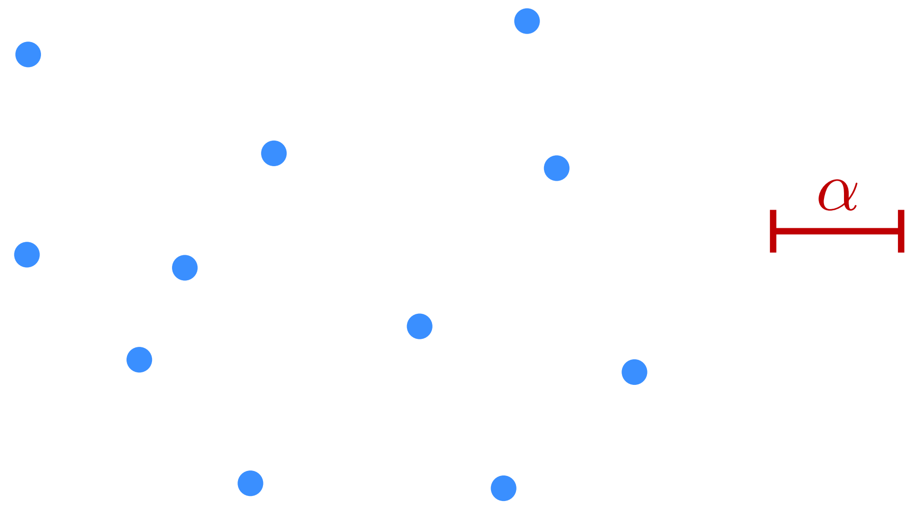
Lower bound: $\Omega(n \log n)$ (algebraic computation tree model of computation)

Our model: real RAM + $O(1)$-time floor (+ *log*-function) + hashing + randomization
(Warning: floor function dangerous from perspective of complexity theory)

Next: $O(n)$-time algorithm using grids + randomization

First step $O(n)$-time decision algorithm: $\exists p, q \in P \colon \|p - q\| < \alpha\,?$ $(p \neq q)$.

Closest Pair: Decision Problem (Algorithm)



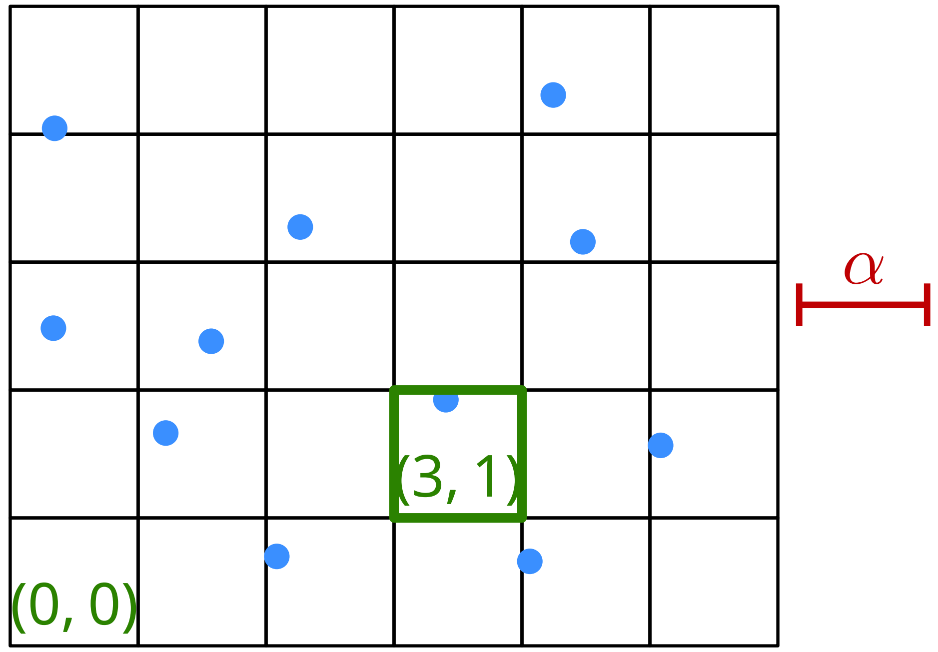
$\exists p, q \in P\colon \|p - q\| < \alpha?$

Closest Pair: Decision Problem (Algorithm)

grid notation

G_α grid with side length α

cell with id (i, j) all points (x, y) with
$\alpha i \leq x < \alpha(i+1)$ and $\alpha j \leq y < \alpha(j+1)$



$\exists p, q \in P \colon \|p - q\| < \alpha?$

Closest Pair: Decision Problem (Algorithm)

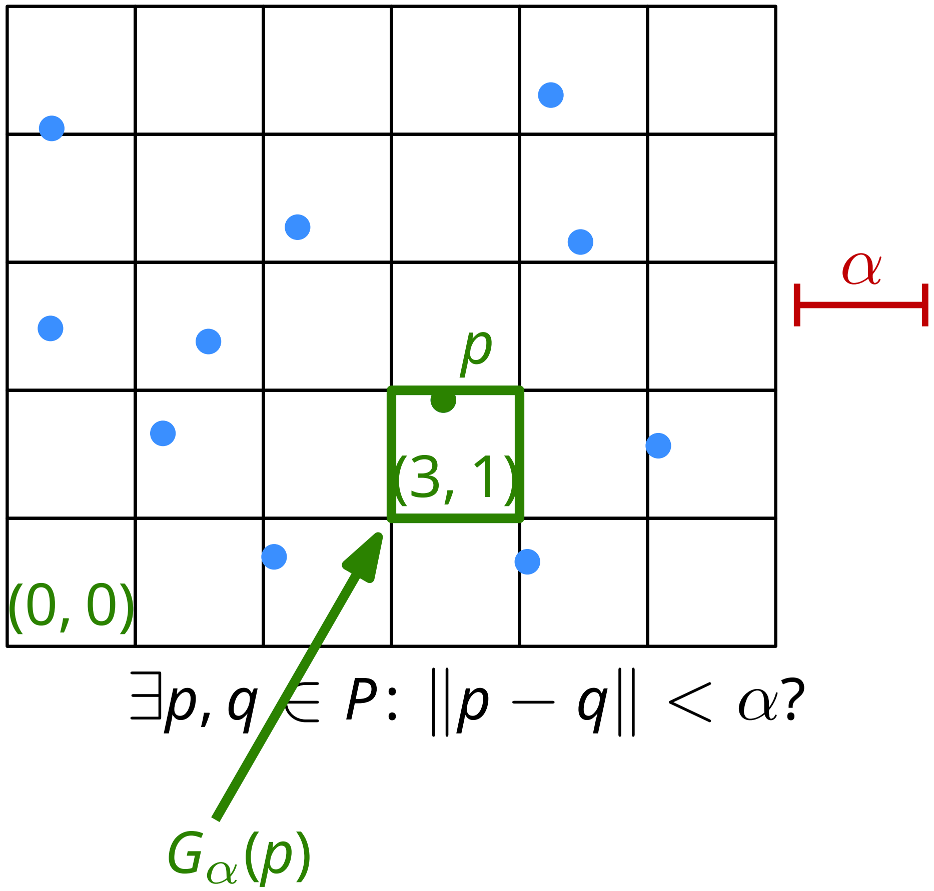
grid notation

G_α grid with side length α

cell with id (i, j) all points (x, y) with
$\alpha i \leq x < \alpha(i+1)$ and $\alpha j \leq y < \alpha(j+1)$

for $p = (x, y)$: $G_\alpha(p) = (\lfloor \frac{x}{\alpha} \rfloor \alpha, \lfloor \frac{y}{\alpha} \rfloor \alpha)$ lower left grid point of cell containing p

for $p = (x, y)$: $id(p) = (\lfloor \frac{x}{\alpha} \rfloor, \lfloor \frac{y}{\alpha} \rfloor)$



Closest Pair: Decision Problem (Algorithm)

grid notation

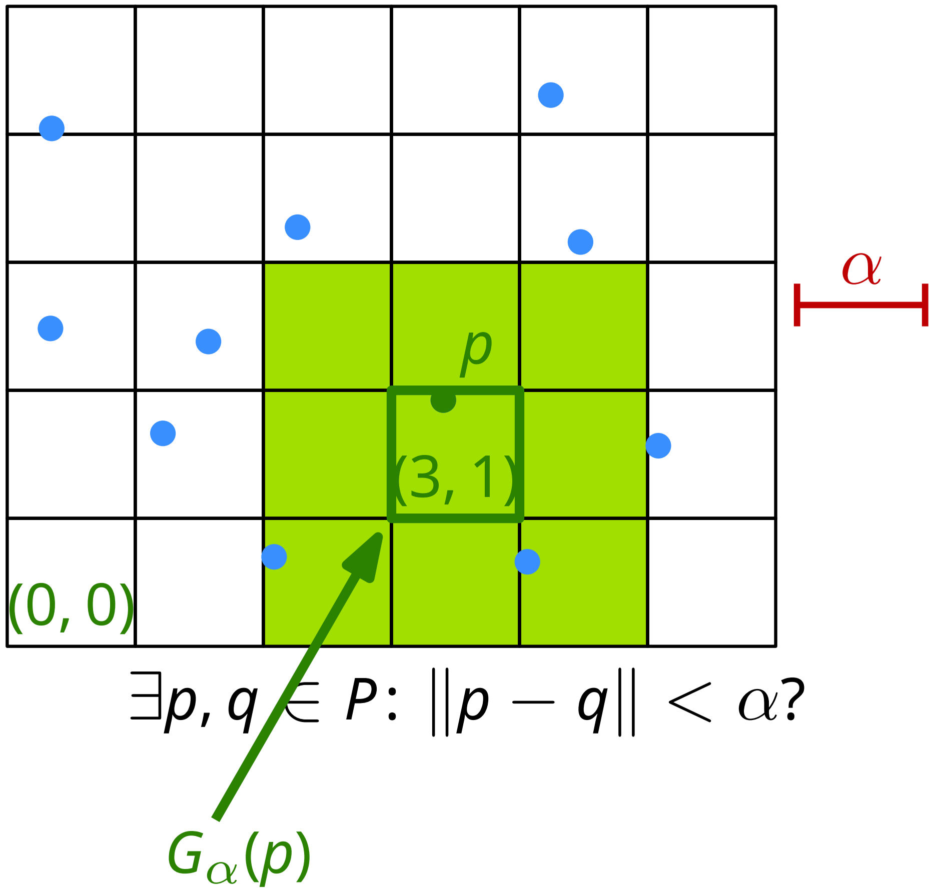
G_α grid with side length α

cell with id (i, j) all points (x, y) with
$\alpha i \leq x < \alpha(i+1)$ and $\alpha j \leq y < \alpha(j+1)$

for $p = (x, y)$: $G_\alpha(p) = (\lfloor \frac{x}{\alpha} \rfloor \alpha, \lfloor \frac{y}{\alpha} \rfloor \alpha)$ lower left grid point of cell containing p

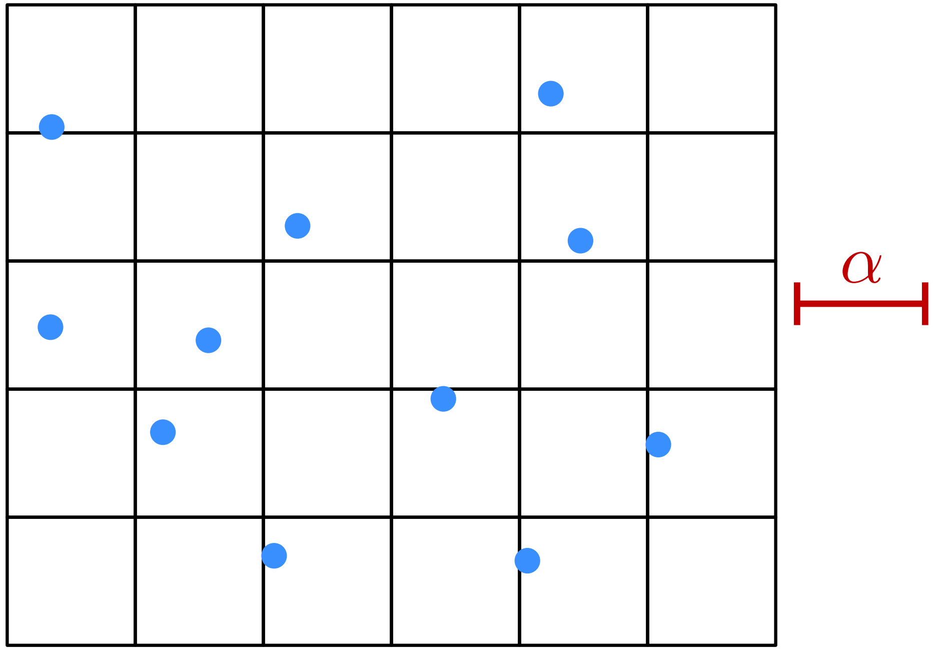
for $p = (x, y)$: $id(p) = (\lfloor \frac{x}{\alpha} \rfloor, \lfloor \frac{y}{\alpha} \rfloor)$

grid cluster: block of 3×3 cells



Closest Pair: Decision Problem (Algorithm)

1. Hash every point $p = (p_x, p_y)$ to grid cell $\left(\left\lfloor \frac{p_x}{\alpha} \right\rfloor, \left\lfloor \frac{p_y}{\alpha} \right\rfloor\right)$

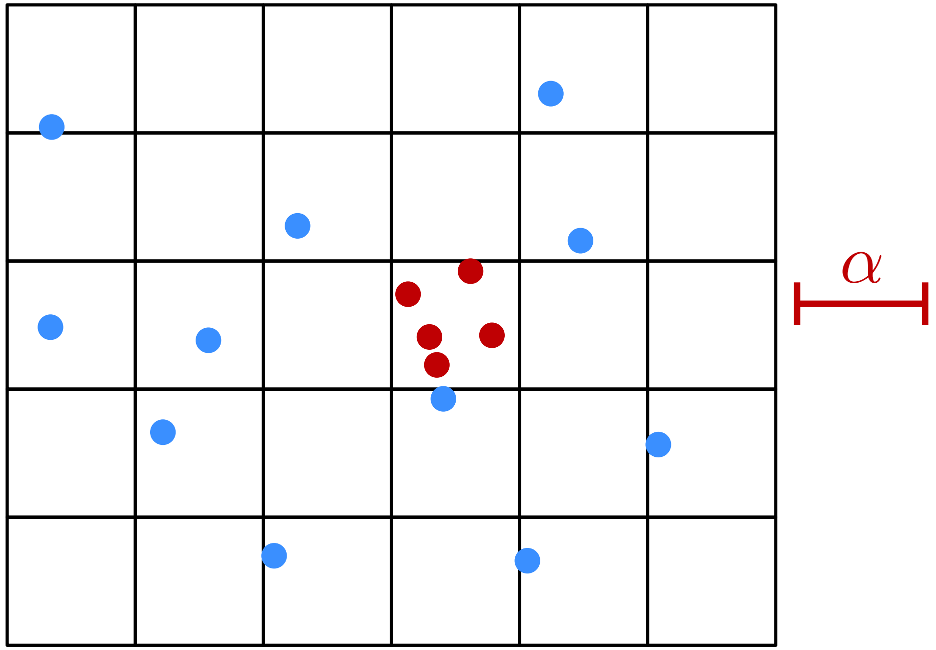


$\exists p, q \in P\colon \|p - q\| < \alpha?$

Closest Pair: Decision Problem (Algorithm)

1. Hash every point $p = (p_x, p_y)$ to grid cell $(\lfloor \frac{p_x}{\alpha} \rfloor, \lfloor \frac{p_y}{\alpha} \rfloor)$

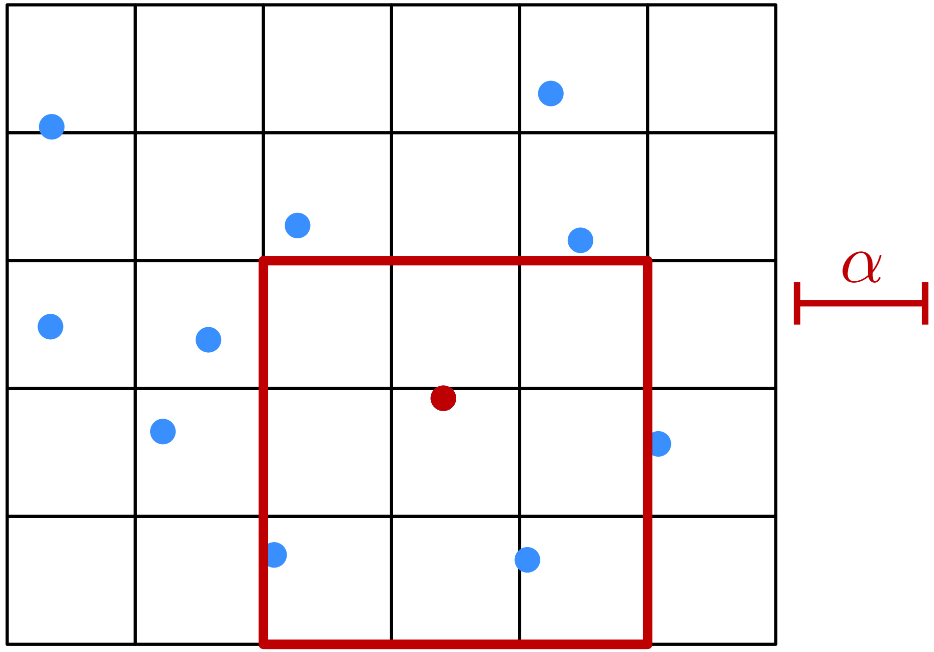
2. If $\exists$ cell with $>$ 4 points: return true



$\exists p, q \in P\colon \|p - q\| < \alpha$?

Closest Pair: Decision Problem (Algorithm)

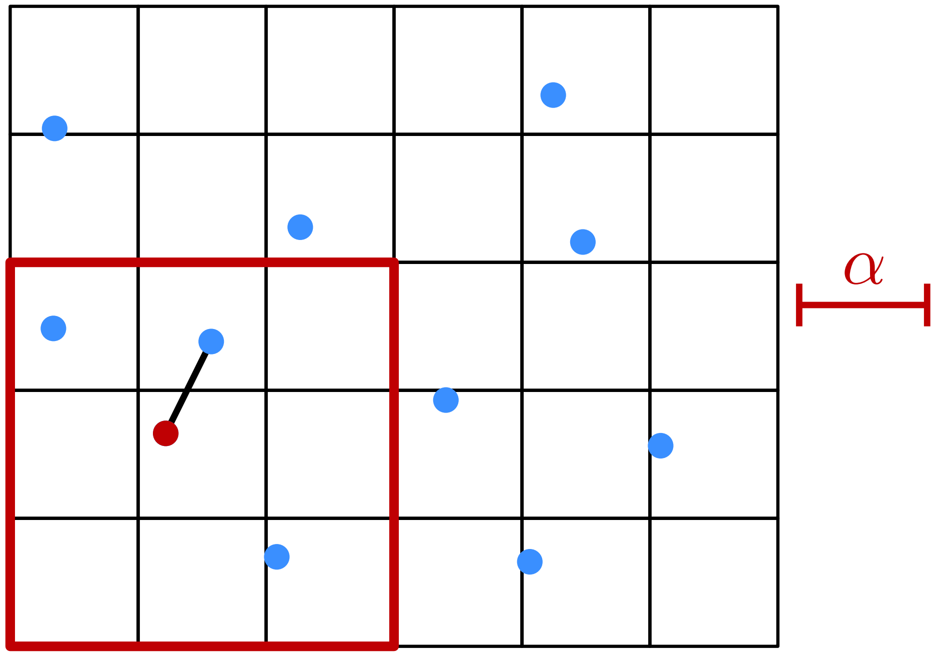
1. Hash every point $p = (p_x, p_y)$ to grid cell $(\lfloor \frac{p_x}{\alpha} \rfloor, \lfloor \frac{p_y}{\alpha} \rfloor)$
2. If $\exists$ cell with $>$ 4 points: return true
3. For every $p \in P$: check distance to other points in grid cluster (cell +8 neighboring cells)
   return true if distance $< \alpha$ found



$\exists p, q \in P\colon \|p - q\| < \alpha$?

Closest Pair: Decision Problem (Algorithm)

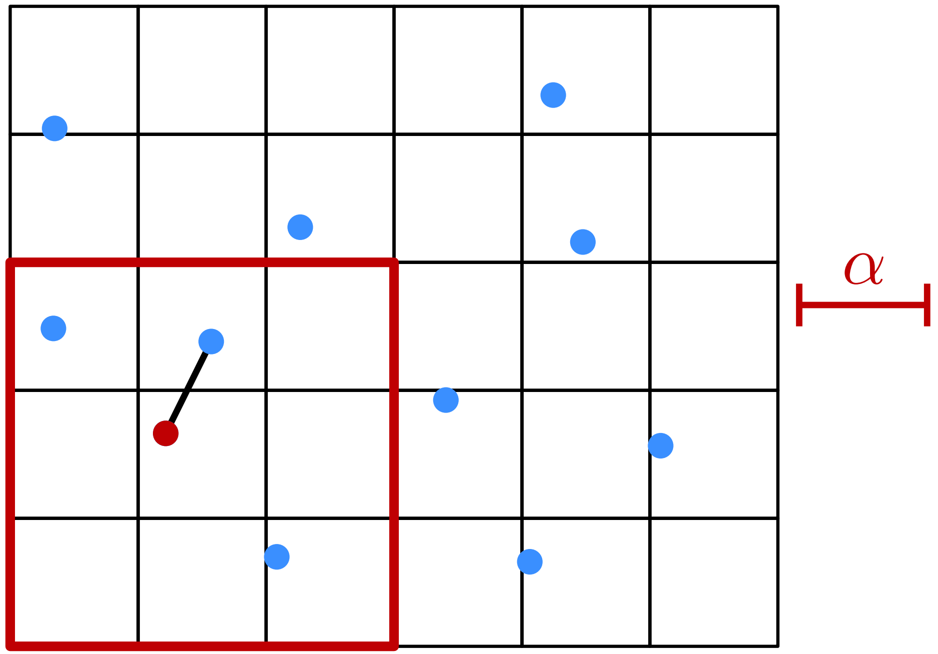
1. Hash every point $p = (p_x, p_y)$ to grid cell $(\lfloor \frac{p_x}{\alpha} \rfloor, \lfloor \frac{p_y}{\alpha} \rfloor)$
2. If $\exists$ cell with $>$ 4 points: return true
3. For every $p \in P$: check distance to other points in grid cluster (cell +8 neighboring cells)
   return true if distance $< \alpha$ found



$\exists p, q \in P\colon \|p - q\| < \alpha$?

Closest Pair: Decision Problem (Algorithm)

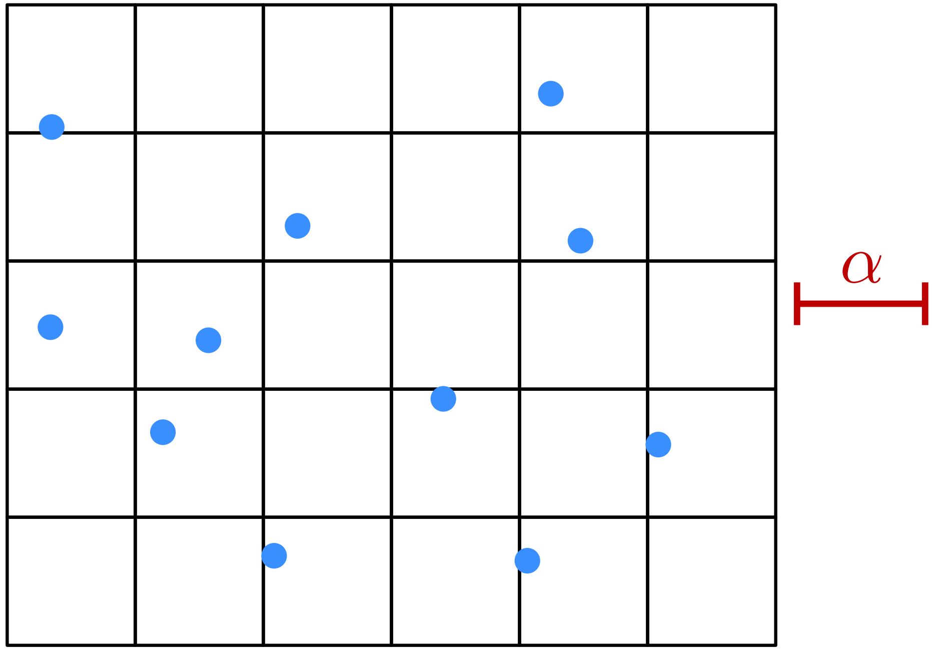
1. Hash every point $p = (p_x, p_y)$ to grid cell $(\lfloor \frac{p_x}{\alpha} \rfloor, \lfloor \frac{p_y}{\alpha} \rfloor)$
2. If $\exists$ cell with > 4 points: return true
3. For every $p \in P$: check distance to other points in grid cluster (cell +8 neighboring cells)
   return true if distance $< \alpha$ found
4. If checks in 2.+3. fail, return false



$\exists p, q \in P \colon \|p - q\| < \alpha$?

Closest Pair: Decision Problem (Algorithm)

1. Hash every point $p = (p_x, p_y)$ to grid cell $(\lfloor \frac{p_x}{\alpha} \rfloor, \lfloor \frac{p_y}{\alpha} \rfloor)$
2. If $\exists$ cell with $>$ 4 points: return true
3. For every $p \in P$: check distance to other points in grid cluster (cell +8 neighboring cells)
   return true if distance $< \alpha$ found
4. If checks in 2.+3. fail, return false



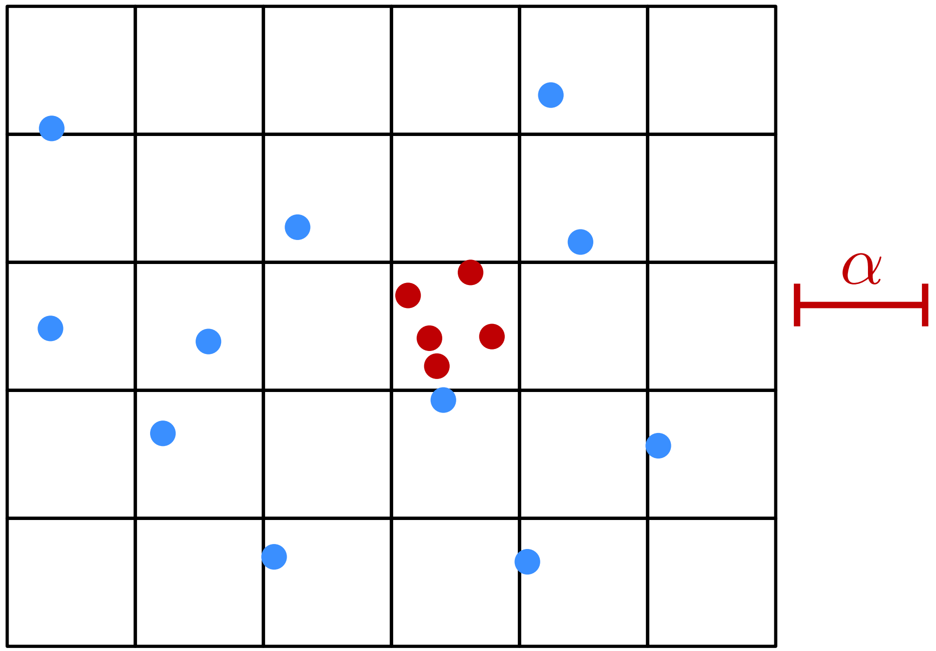
$\exists p, q \in P\colon \|p - q\| < \alpha$?

Correctness

???

Closest Pair: Decision Problem (Algorithm)

1. Hash every point $p = (p_x, p_y)$ to grid cell $(\lfloor \frac{p_x}{\alpha} \rfloor, \lfloor \frac{p_y}{\alpha} \rfloor)$
2. If $\exists$ cell with $>$ 4 points: return true
3. For every $p \in P$: check distance to other points in grid cluster (cell +8 neighboring cells)
   return true if distance $< \alpha$ found
4. If checks in 2.+3. fail, return false



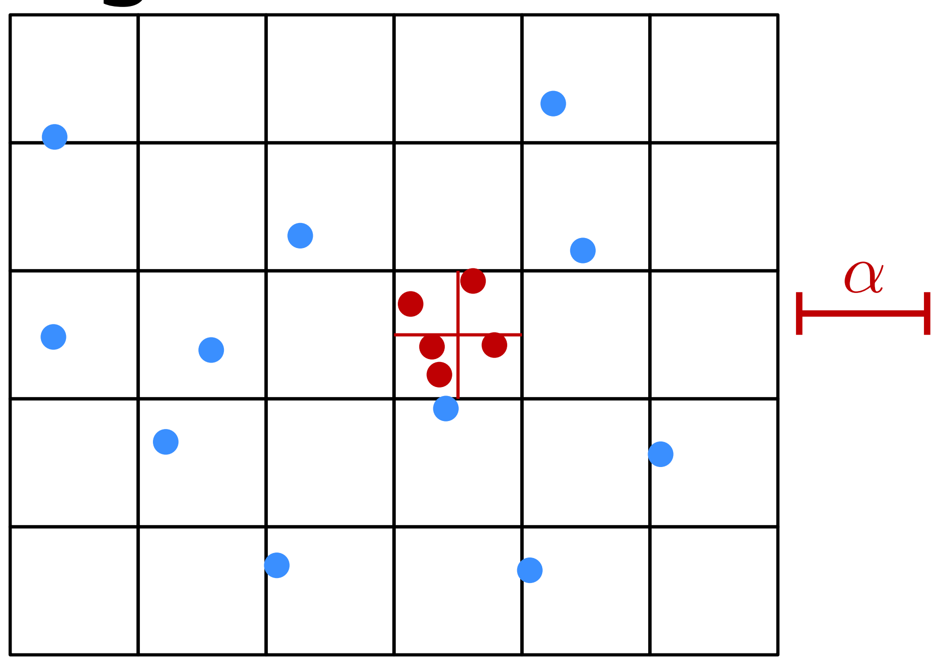
$\exists p, q \in P\colon \|p - q\| < \alpha$?

Correctness

Line 2: If $>$ 4 points in cell

Closest Pair: Decision Problem (Algorithm)

1. Hash every point $p = (p_x, p_y)$ to grid cell $(\lfloor \frac{p_x}{\alpha} \rfloor, \lfloor \frac{p_y}{\alpha} \rfloor)$
2. If $\exists$ cell with > 4 points: return true
3. For every $p \in P$: check distance to other points in grid cluster (cell +8 neighboring cells)
   return true if distance $< \alpha$ found
4. If checks in 2.+3. fail, return false



$\exists p, q \in P \colon \|p - q\| < \alpha?$

Correctness

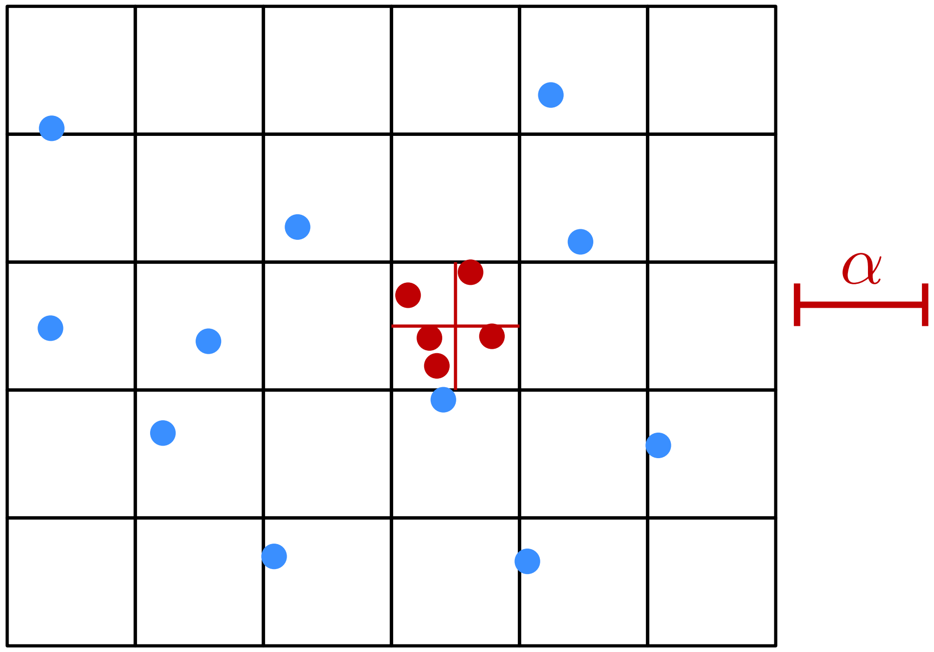
Line 2: If > 4 points in cell

$\Rightarrow \exists p, q$ in a subcell of sidelength $\frac{\alpha}{2}$

$\Rightarrow \|p - q\| < \sqrt{2}\frac{\alpha}{2} < \alpha$

Closest Pair: Decision Problem (Algorithm)

1. Hash every point $p = (p_x, p_y)$ to grid cell $(\lfloor \frac{p_x}{\alpha} \rfloor, \lfloor \frac{p_y}{\alpha} \rfloor)$
2. If $\exists$ cell with > 4 points: return true
3. For every $p \in P$: check distance to other points in grid cluster (cell +8 neighboring cells)
   return true if distance $< \alpha$ found
4. If checks in 2.+3. fail, return false



$\exists p, q \in P \colon \|p - q\| < \alpha?$

Correctness

Line 2: If > 4 points in cell

$\Rightarrow \exists p, q$ in a subcell of sidelength $\frac{\alpha}{2}$

$\Rightarrow \|p - q\| < \sqrt{2}\frac{\alpha}{2} < \alpha$

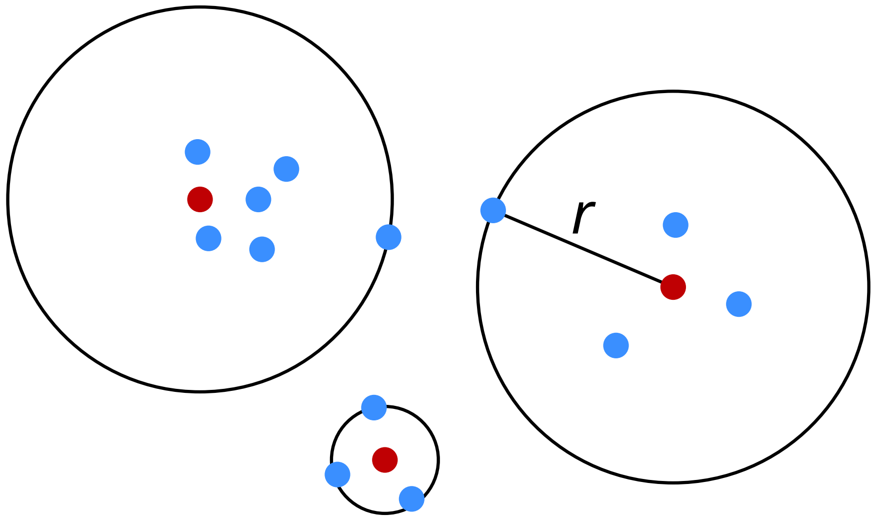
Running time

$O(n) + O(n) + n \cdot O(9 \cdot 4) + O(1) = O(n)$

Exercise 1.2 (A): Decision Problem

Compute clustering radius (Exercise 1.2 from book)

Let C and P be two given sets of points in the plane, such that $k = |C|$ and $n = |P|$. Let $r = \max_{p \in P} \min_{c \in C} \|c - p\|$ be the covering radius of P by C.

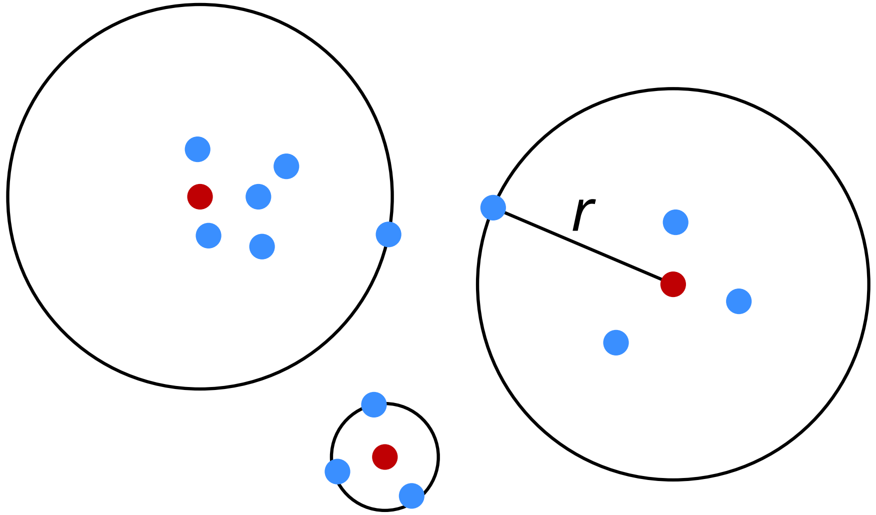


(A) Give an $O(n + k \log n)$ expected time algorithm that outputs a number α, such that $r \leq \alpha \leq 10r$.

Exercise 1.2 (A): Decision Problem

Compute clustering radius (Exercise 1.2 from book)

Let C and P be two given sets of points in the plane, such that $k = |C|$ and $n = |P|$. Let $r = \max_{p \in P} \min_{c \in C} \|c - p\|$ be the covering radius of P by C.



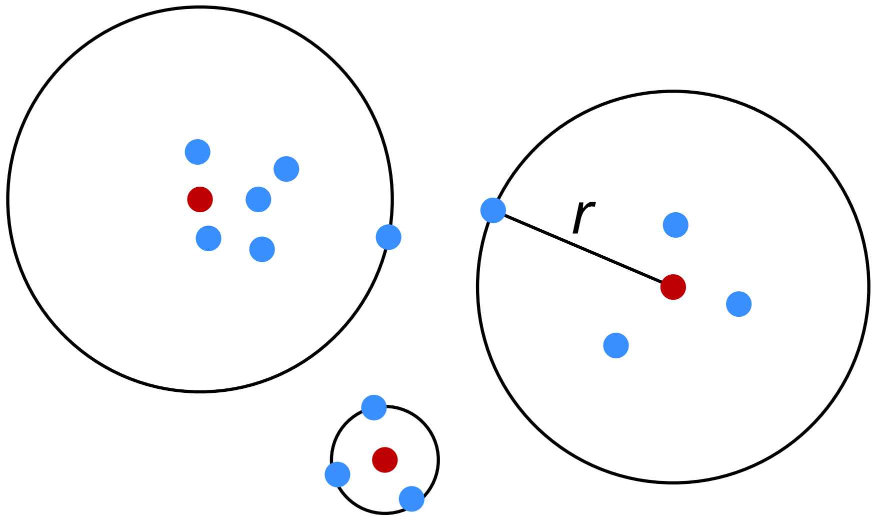
(A) Give an $O(n + k \log n)$ expected time algorithm that outputs a number α, such that $r \leq \alpha \leq 10r$.

First step: Decision problem

Exercise 1.2 (A): Decision Problem

Compute clustering radius (Exercise 1.2 from book)

Let C and P be two given sets of points in the plane, such that $k = |C|$ and $n = |P|$. Let $r = \max_{p \in P} \min_{c \in C} \|c - p\|$ be the covering radius of P by C.



A) Give an $O(n + k \log n)$ expected time algorithm that outputs a number α, such that $r \leq \alpha \leq 10r$.

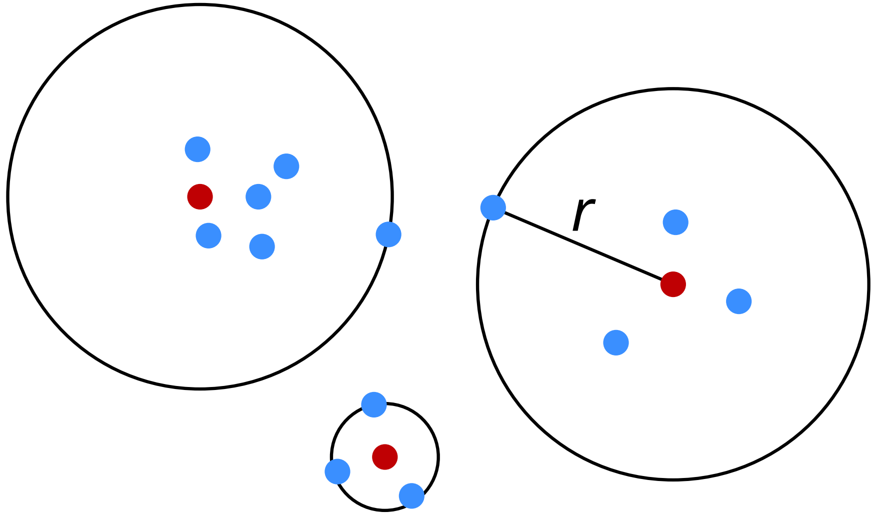
First step: Decision problem

A') Give an $O(n + k)$ algorithm (for given $\alpha' \geq 0$) that outputs
$$\begin{cases} \text{true} & \text{if } r \leq \alpha' \\ \text{false} & \text{if } r > 2\sqrt{2}\alpha' \\ \text{true or false} & \text{if } \alpha' < r \leq 2\sqrt{2}\alpha' \end{cases}$$

Exercise 1.2 (A): Decision Problem

Compute clustering radius (Exercise 1.2 from book)

Let C and P be two given sets of points in the plane, such that $k = |C|$ and $n = |P|$. Let $r = \max_{p \in P} \min_{c \in C} \|c - p\|$ be the covering radius of P by C.



A) Give an $O(n + k \log n)$ expected time algorithm that outputs a number α, such that $r \leq \alpha \leq 10r$.

First step: Decision problem

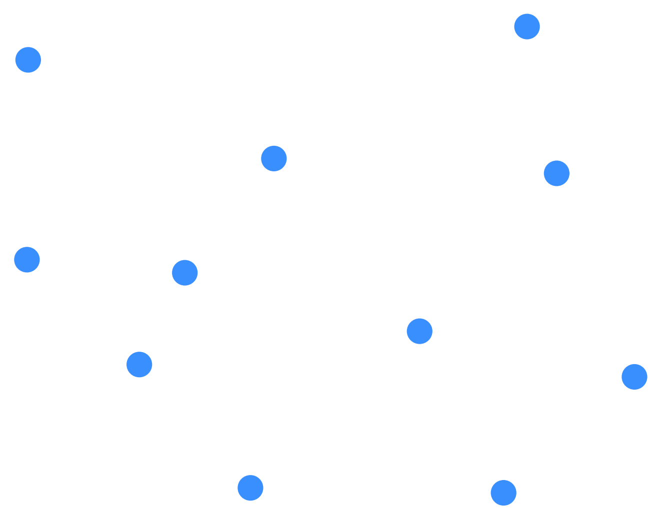
A') Give an $O(n + k)$ algorithm (for given $\alpha' \geq 0$) that outputs

$$\begin{cases} \text{true} & \text{if } r \leq \alpha' \\ \text{false} & \text{if } r > 2\sqrt{2}\alpha' \\ \text{true or false} & \text{if } \alpha' < r \leq 2\sqrt{2}\alpha' \end{cases}$$

Needs to correctly decide $r \leq \alpha'$, except if r only slightly larger

Closest Pair: Randomized Algorithm

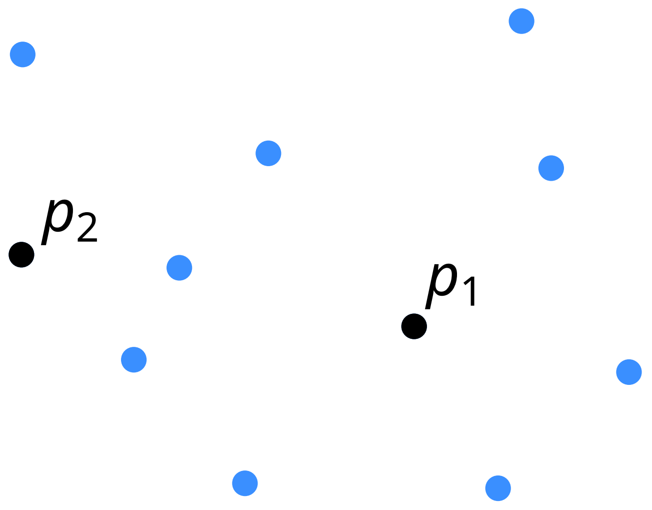
Compute $\|p - q\|$ of closest pair



Closest Pair: Randomized Algorithm

Compute $\|p - q\|$ of closest pair

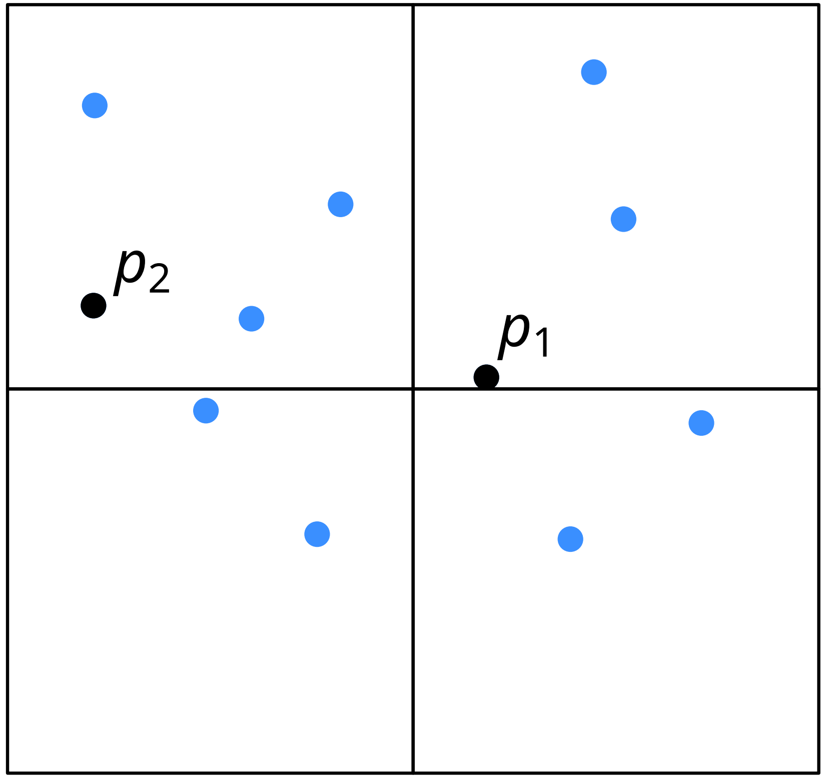
1. $p_1, \dots, p_n$ points of P in random order



Closest Pair: Randomized Algorithm

Compute $\|p - q\|$ of closest pair

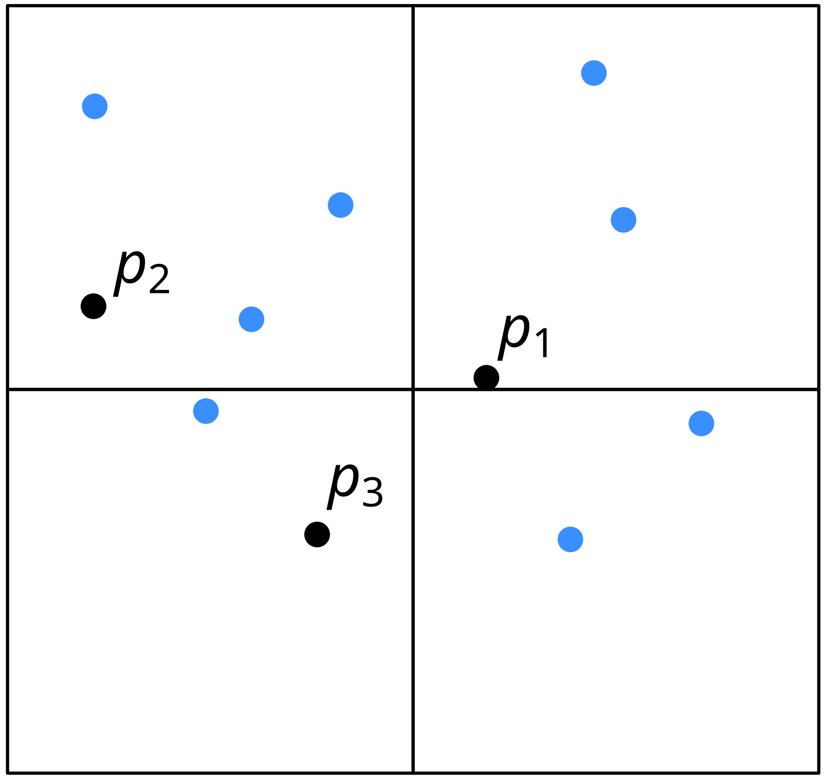
1. $p_1, \dots, p_n$ points of P in random order
2. $\alpha = \|p_1 - p_2\|$



Closest Pair: Randomized Algorithm

Compute $\|p - q\|$ of closest pair

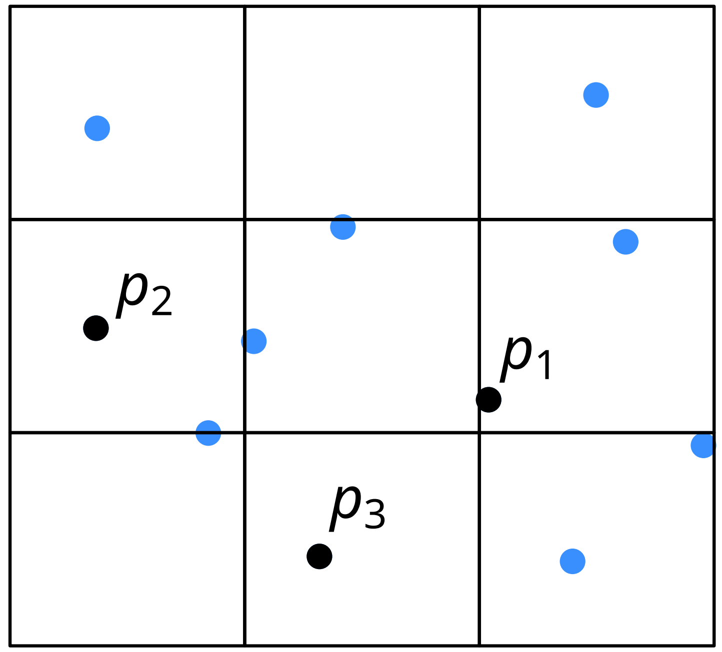
1. $p_1, \dots, p_n$ points of P in random order
2. $\alpha = \|p_1 - p_2\|$
3. Run decision algorithm incrementally:
   add p_i one-by-one
   if $\exists j < i$: $\alpha' := \|p_i - p_j\| < \alpha$, restart with $\alpha = \alpha'$



Closest Pair: Randomized Algorithm

Compute $\|p - q\|$ of closest pair

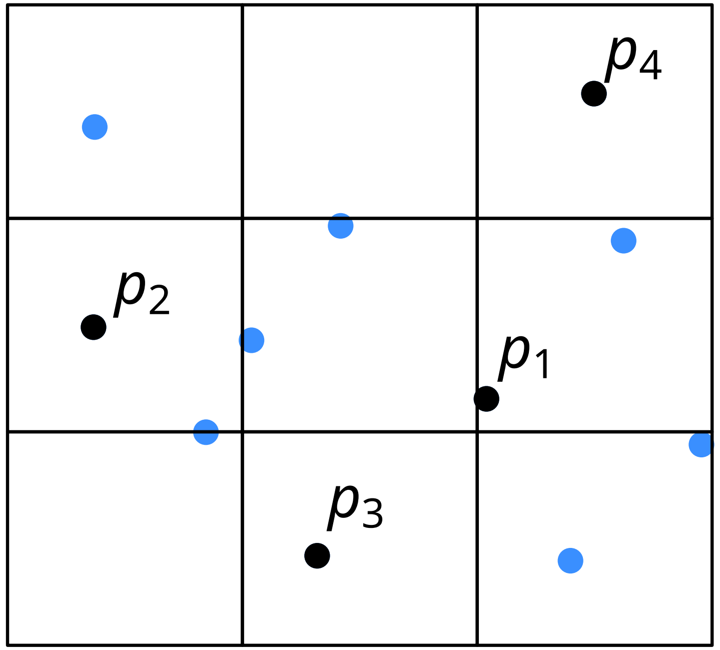
1. $p_1, \ldots, p_n$ points of P in random order
2. $\alpha = \|p_1 - p_2\|$
3. Run decision algorithm incrementally:
   add p_i one-by-one
   if $\exists j < i$: $\alpha' := \|p_i - p_j\| < \alpha$, restart with $\alpha = \alpha'$



Closest Pair: Randomized Algorithm

Compute $\|p - q\|$ of closest pair

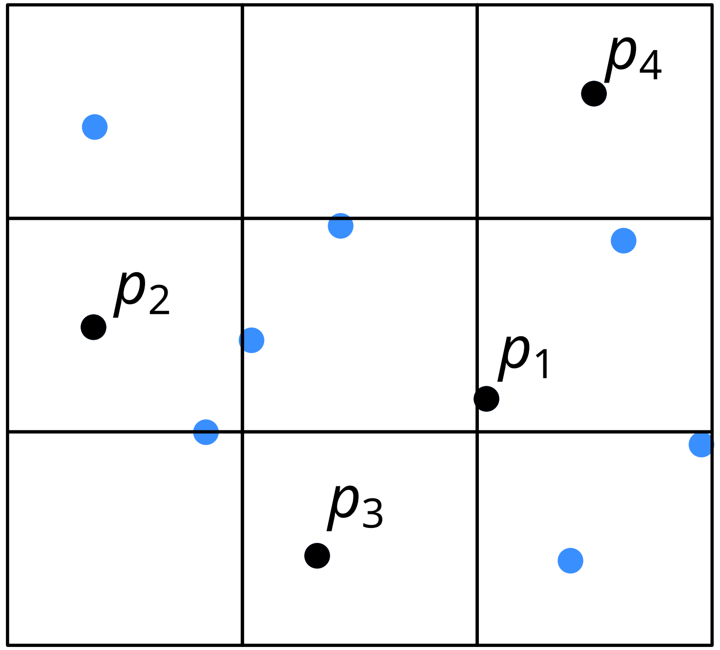
1. $p_1, \dots, p_n$ points of P in random order
2. $\alpha = \|p_1 - p_2\|$
3. Run decision algorithm incrementally:
   add p_i one-by-one
   if $\exists j < i$: $\alpha' := \|p_i - p_j\| < \alpha$, restart with $\alpha = \alpha'$



Closest Pair: Randomized Algorithm

Compute $\|p - q\|$ of closest pair

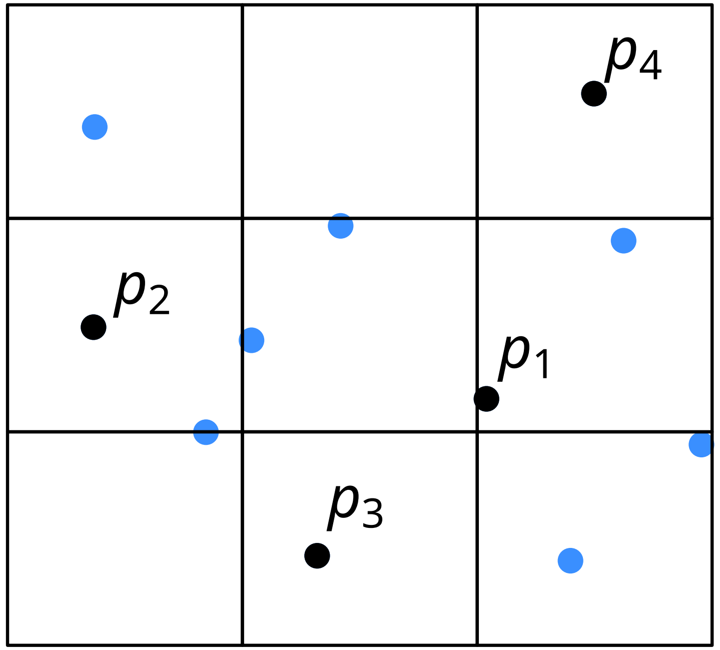
1. $p_1, \dots, p_n$ points of P in random order
2. $\alpha = \|p_1 - p_2\|$
3. Run decision algorithm incrementally:
   add p_i one-by-one
   if $\exists j < i$: $\alpha' := \|p_i - p_j\| < \alpha$, restart with $\alpha = \alpha'$
4. return α



Closest Pair: Randomized Algorithm

Compute $\|p - q\|$ of closest pair

1. $p_1, \dots, p_n$ points of P in random order
2. $\alpha = \|p_1 - p_2\|$
3. Run decision algorithm incrementally:
   add p_i one-by-one
   if $\exists j < i$: $\alpha' := \|p_i - p_j\| < \alpha$, restart with $\alpha = \alpha'$
4. return α



Correctness

Follows from correctness of decision algorithm

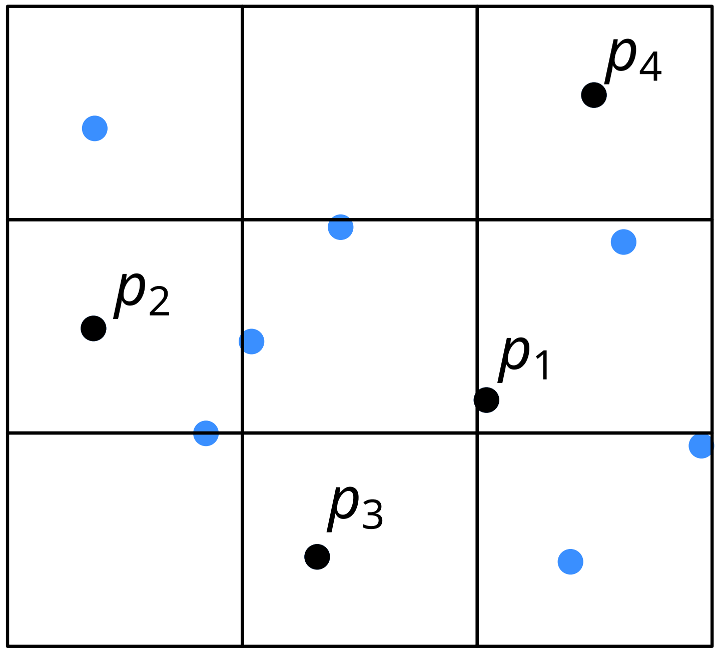
Running time

???

Closest Pair: Randomized Algorithm

Compute $\|p - q\|$ of closest pair

1. $p_1, \ldots, p_n$ points of P in random order
2. $\alpha = \|p_1 - p_2\|$
3. Run decision algorithm incrementally:
   add p_i one-by-one
   if $\exists j < i$: $\alpha' := \|p_i - p_j\| < \alpha$, restart with $\alpha = \alpha'$
4. return α



Correctness

Follows from correctness of decision algorithm

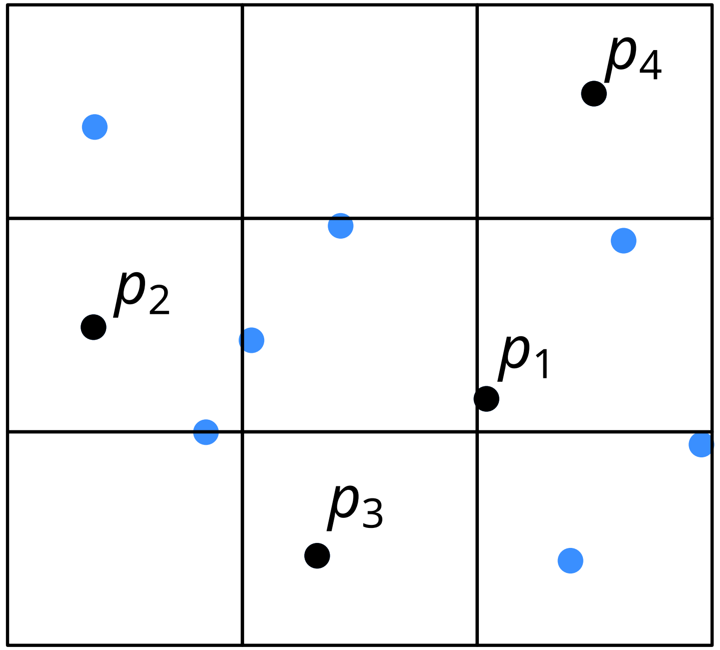
Running time

worst-case: restart for every i: $\sum \Theta(i) = \Theta(n^2)$

Closest Pair: Randomized Algorithm

Compute $\|p - q\|$ of closest pair

1. $p_1, \dots, p_n$ points of P in random order
2. $\alpha = \|p_1 - p_2\|$
3. Run decision algorithm incrementally:
   add p_i one-by-one
   if $\exists j < i$: $\alpha' := \|p_i - p_j\| < \alpha$, restart with $\alpha = \alpha'$
4. return α



Correctness

Follows from correctness of decision algorithm

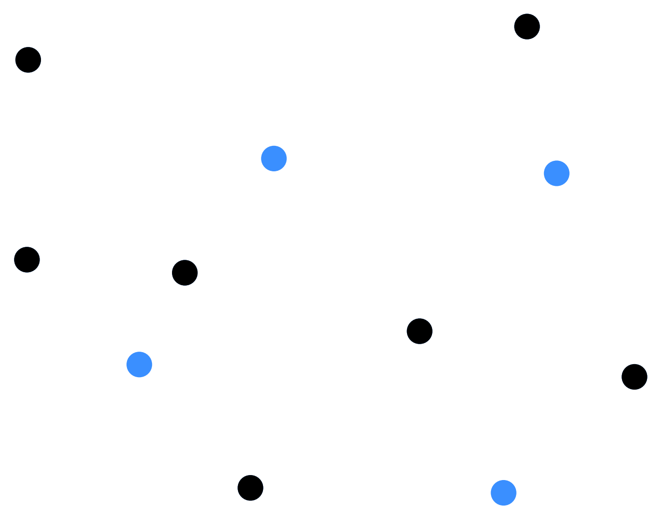
Running time

worst-case: restart for every i: $\sum \Theta(i) = \Theta(n^2)$

expected case?

Closest Pair: Expected case

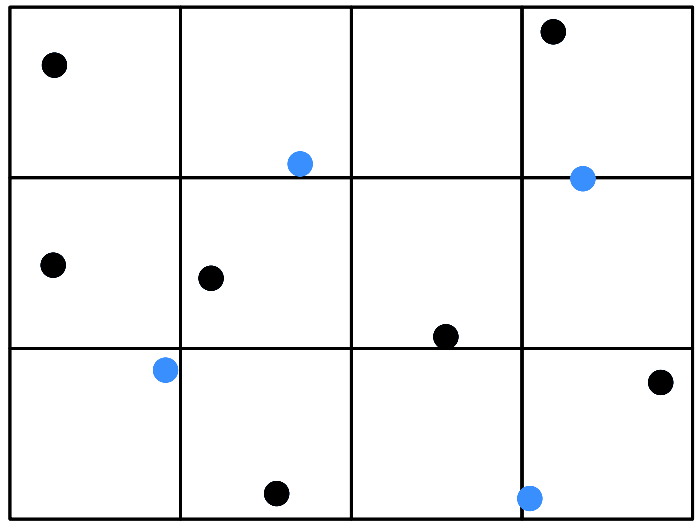
Backwards analysis



Closest Pair: Expected case

Backwards analysis

To analyze insertion of p_i: consider grid after inserting p_i.

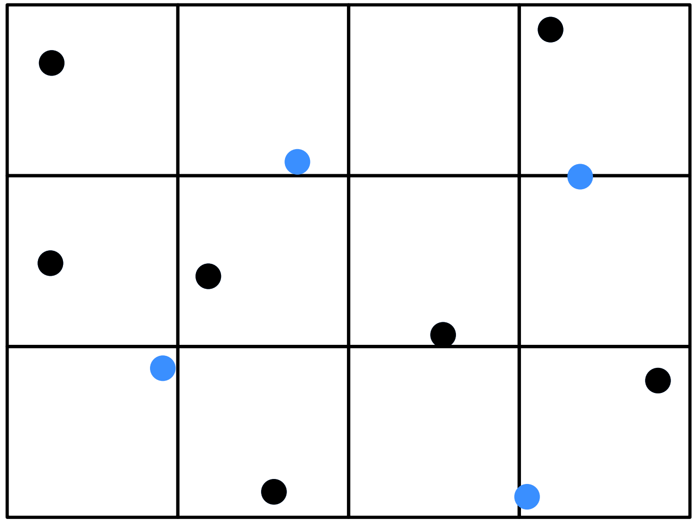


Closest Pair: Expected case

Backwards analysis

To analyze insertion of p_i: consider grid after inserting p_i.

Which points would have caused the grid to change?
$\rightarrow$ probability

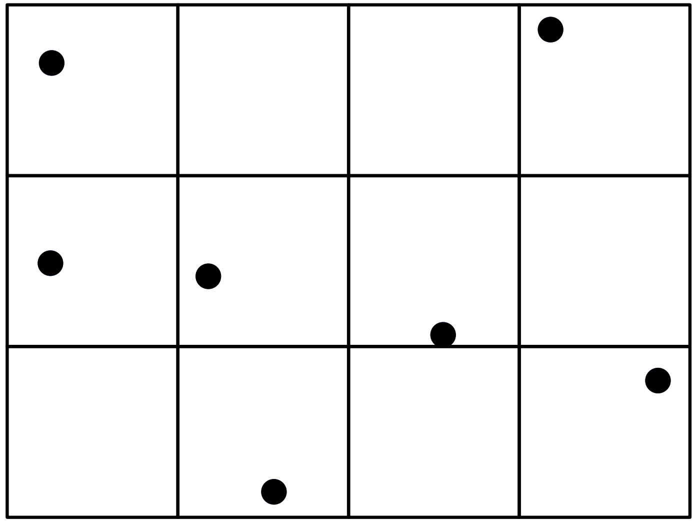


Closest Pair: Expected case

Backwards analysis

To analyze insertion of p_i: consider grid after inserting p_i.

Which points would have caused the grid to change?
$\rightarrow$ probability

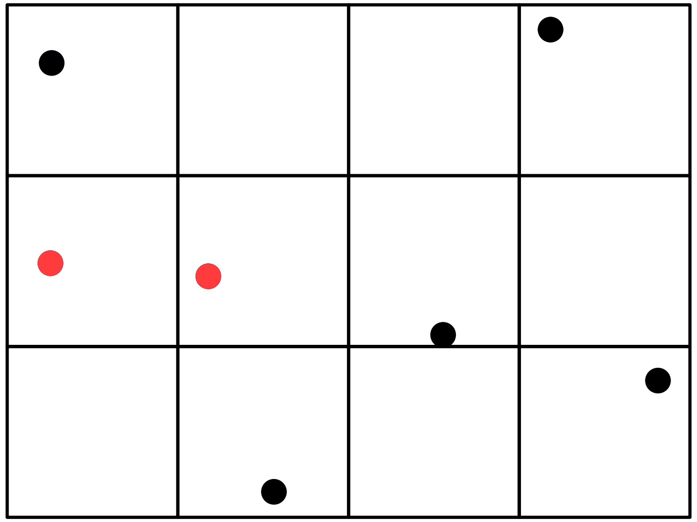


Closest Pair: Expected case

Backwards analysis

To analyze insertion of p_i: consider grid after inserting p_i.

Which points would have caused the grid to change?
$\rightarrow$ probability

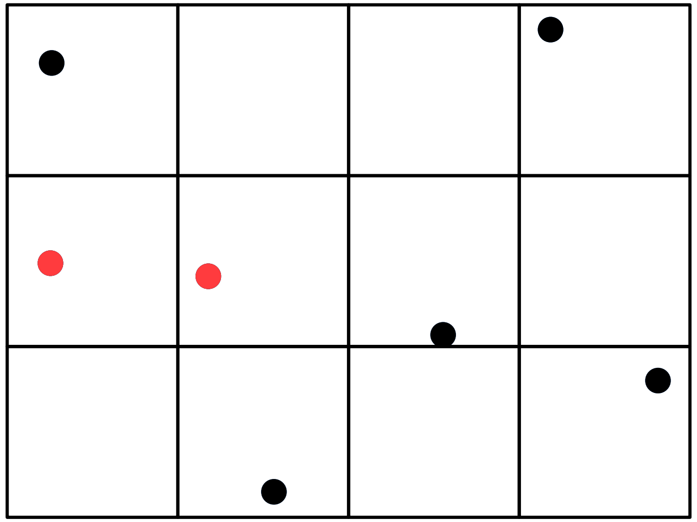


Closest Pair: Expected case

Backwards analysis

To analyze insertion of p_i: consider grid after inserting p_i.

Which points would have caused the grid to change?
$\rightarrow$ probability $\leq 2/i$

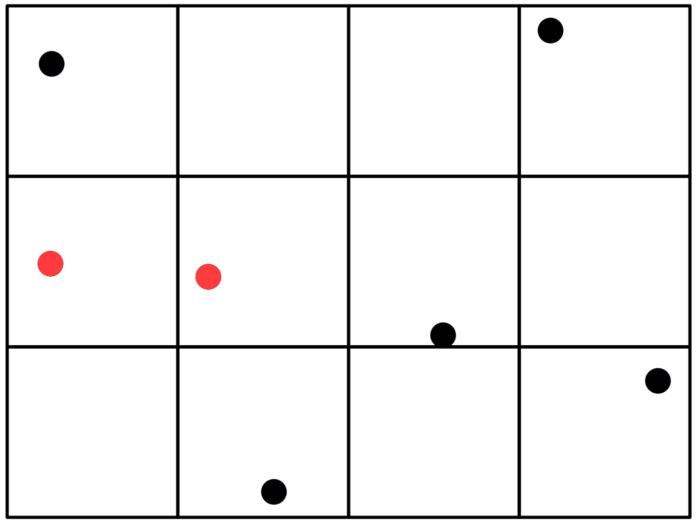


Closest Pair: Expected case

Backwards analysis

To analyze insertion of p_i: consider grid after inserting p_i.

Which points would have caused the grid to change?
$\rightarrow$ probability $\leq 2/i$



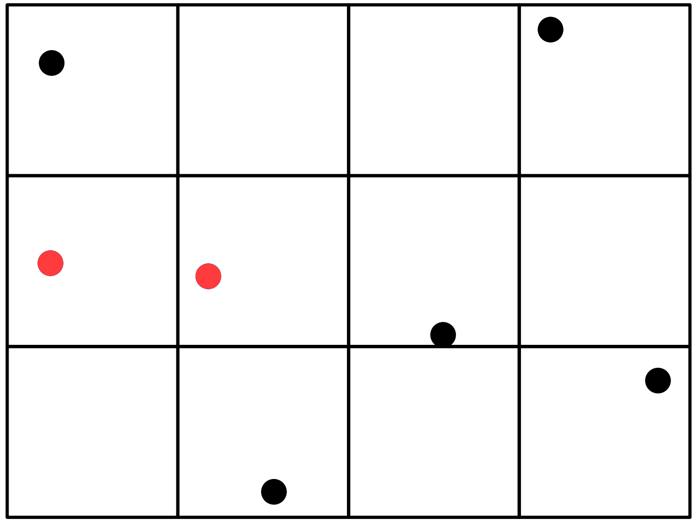
E[Time to insert p_i] =

Closest Pair: Expected case

Backwards analysis

To analyze insertion of p_i: consider grid after inserting p_i.

Which points would have caused the grid to change?
$\rightarrow$ probability $\leq 2/i$



E[Time to insert p_i] = $\underset{\substack{\uparrow\\ \text{time}}}{O(i)} \cdot \underset{\substack{\uparrow \leq\\ \text{probability}}}{2/i}$

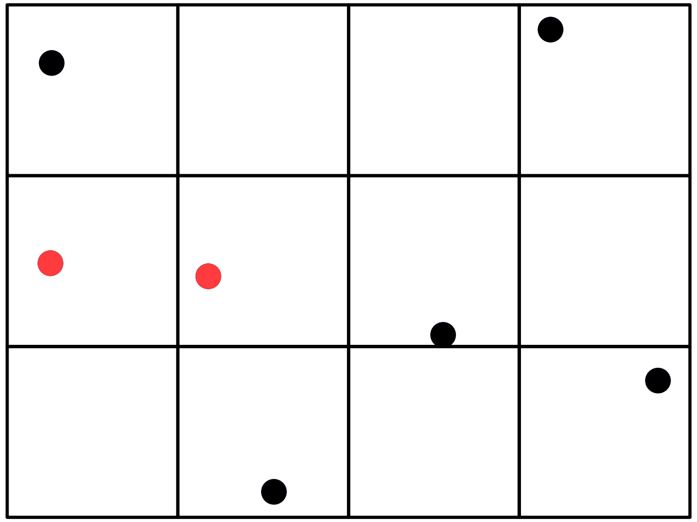
time · probability
rebuilding the grid

Closest Pair: Expected case

Backwards analysis

To analyze insertion of p_i: consider grid after inserting p_i.

Which points would have caused the grid to change?
$\rightarrow$ probability $\leq 2/i$



E[Time to insert p_i] = $O(i) \cdot 2/i$ + $O(1) \cdot 1$

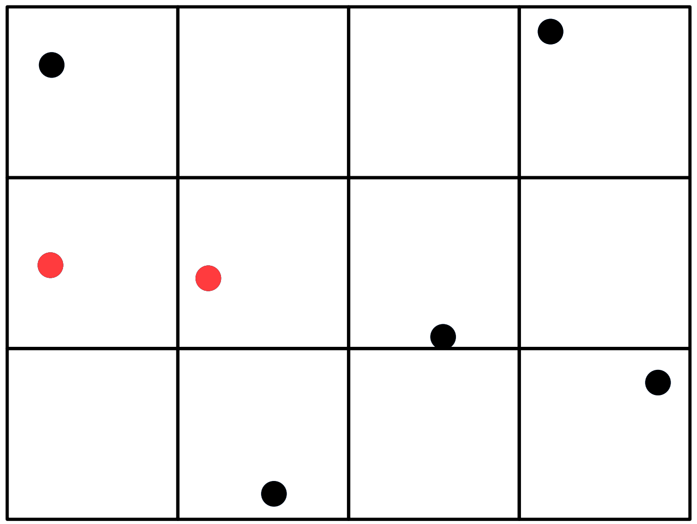
time · probability $\leq$ rebuilding the grid

otherwise

Closest Pair: Expected case

Backwards analysis

To analyze insertion of p_i: consider grid after inserting p_i.



Which points would have caused the grid to change?
$\rightarrow$ probability $\leq 2/i$

$$E[\text{ Time to insert } p_i] = O(i) \cdot 2/i \quad + \quad O(1) \cdot 1 \quad = O(2i/i + 1) = O(1)$$

$\leq$

time · probability otherwise

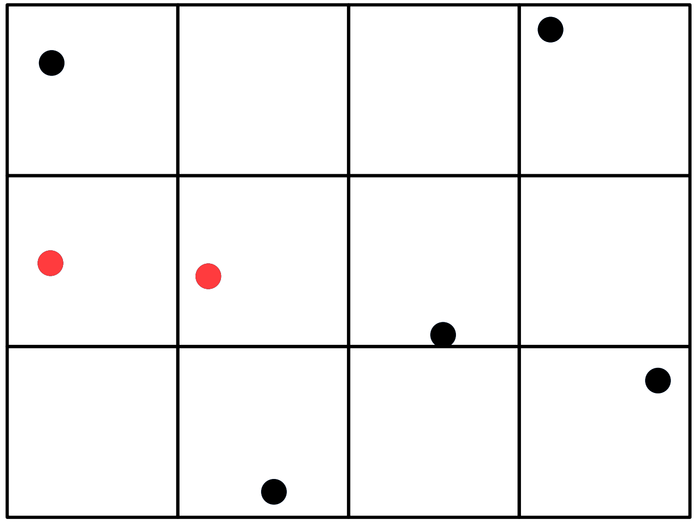
rebuilding the grid

Closest Pair: Expected case

Backwards analysis

To analyze insertion of p_i: consider grid after inserting p_i.

Which points would have caused the grid to change?
$\rightarrow$ probability $\leq 2/i$



E[Time to insert p_i] = $O(i) \cdot 2/i$ + $O(1) \cdot 1$ $= O(2i/i + 1) = O(1)$

time · probability rebuilding the grid

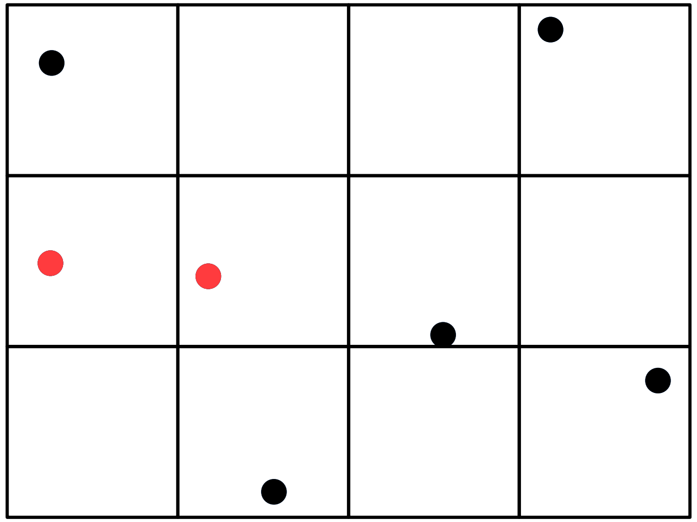
otherwise

E[Overall running time] = $n \cdot O(1) = O(n)$

Closest Pair: Expected case

Backwards analysis

To analyze insertion of p_i: consider grid after inserting p_i.



Which points would have caused the grid to change?
$\rightarrow$ probability $\leq 2/i$

more formally

α_i: closest pair distance of first i points

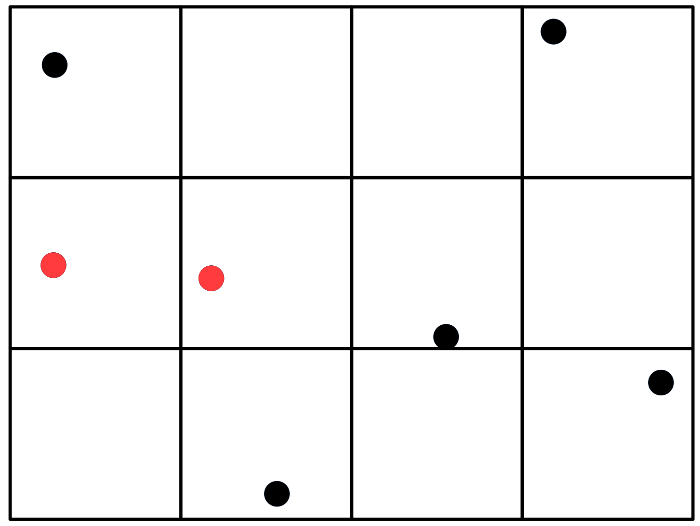
$X_i = \mathbb{1}_{\{\alpha_{i-1} < \alpha_i\}}$ indicator variable

Closest Pair: Expected case

Backwards analysis

To analyze insertion of p_i: consider grid after inserting p_i.

Which points would have caused the grid to change?
$\to$ probability $\leq 2/i$



more formally

α_i: closest pair distance of first i points

$X_i = \mathbb{1}_{\{\alpha_{i-1} < \alpha_i\}}$ indicator variable

running time proportional to $R = 1 + \sum_{i=3}^{n}(1 + iX_i)$

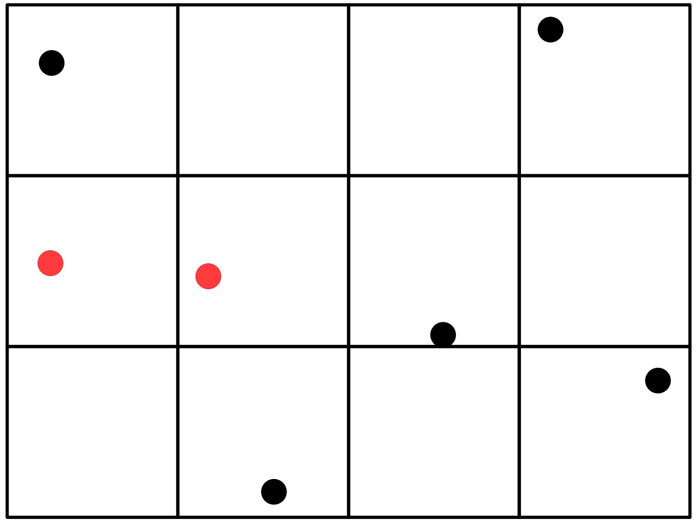
$$E[R] = E[1 + \sum_{i=3}^{n}(1 + i \cdot X_i)] \leq n + \sum_{i=3}^{n} i \cdot E[X_i] = n + \sum_{i=3}^{n} i \cdot Pr[X_i = 1]$$

$$\leq n + \sum_{i=3}^{n} i \cdot 2/i \leq 3n$$

Closest Pair: Expected case

Backwards analysis

To analyze insertion of p_i: consider grid after inserting p_i.



Which points would have caused the grid to change?
$\rightarrow$ probability $\leq 2/i$

more formally

α_i: closest pair distance of first i points

$X_i = \mathbb{1}_{\{\alpha_{i-1} < \alpha_i\}}$ indicator variable

running time proportional to $R = 1 + \sum_{i=3}^{n}(1 + iX_i)$

$$E[R] = E[1 + \sum_{i=3}^{n}(1 + i \cdot X_i)] \leq n + \sum_{i=3}^{n} i \cdot E[X_i] = n + \sum_{i=3}^{n} i \cdot Pr[X_i = 1]$$
$$\leq n + \sum_{i=3}^{n} i \cdot 2/i \leq 3n$$

Using grids we can solve the closest pair problem in expected linear time.

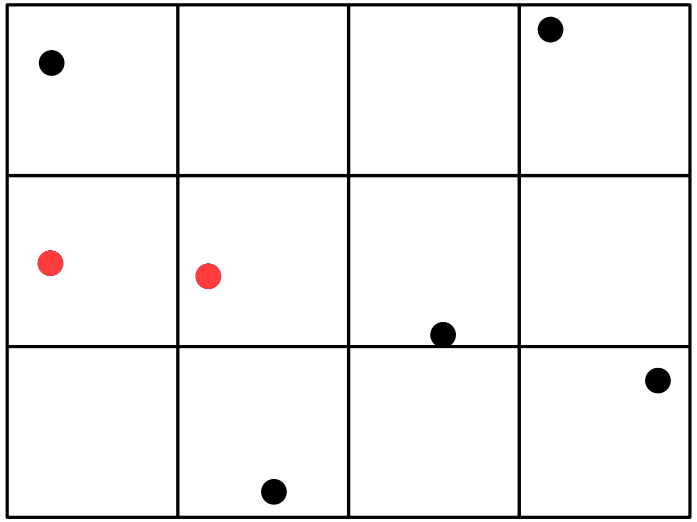
Quiz

How often do we need to rebuild the grid in expectation?

A $O(1)$

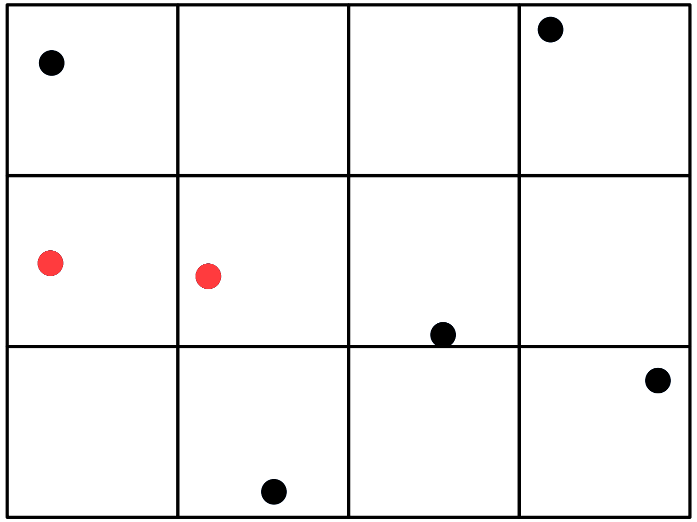
B $O(\log n)$

C $O(n)$



Quiz

How often do we need to rebuild the grid in expectation?



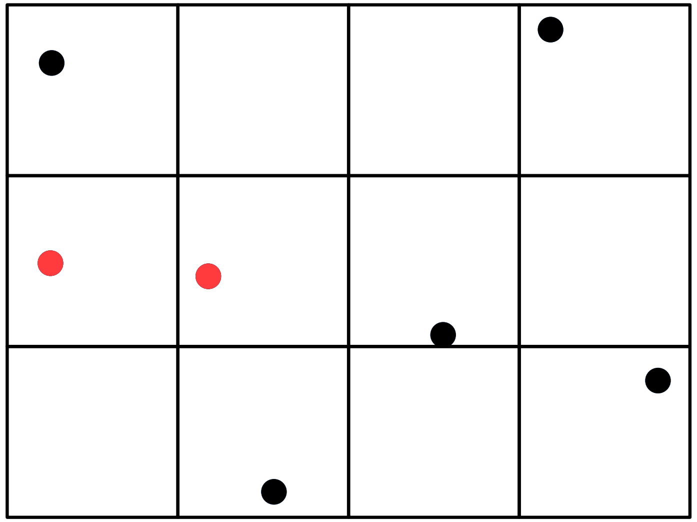
A $O(1)$

B $O(\log n)$

C $O(n)$

Quiz

How often do we need to rebuild the grid in expectation?



A $O(1)$

$\boxed{\text{B} \quad O(\log n)}$

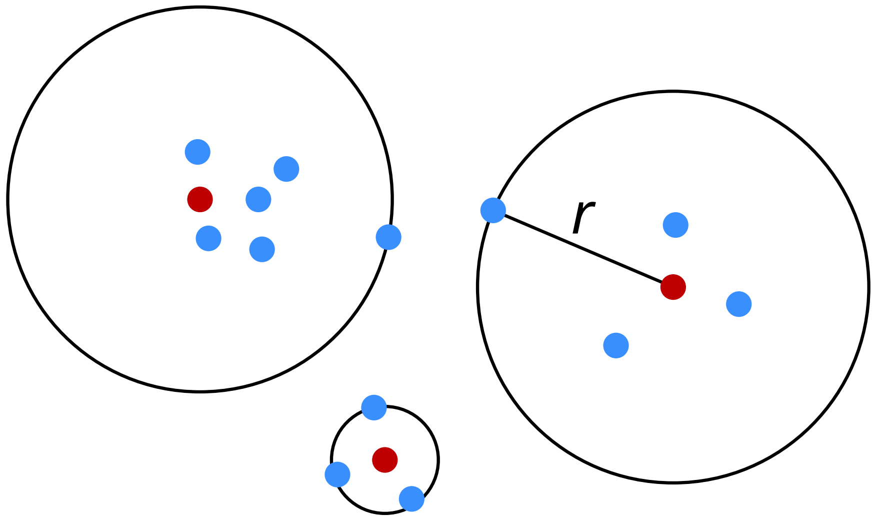
C $O(n)$

$$E\left[\sum_{i=3}^{n} X_i\right] = \sum_{i=3}^{n} E[X_i] \leq \sum_{i=3}^{n} 2/i \leq \int_{3}^{n+1} 1/x \, dx = O(\log n)$$

Exercise 1.2 (A): Decision Problem

Compute clustering radius (Exercise 1.2 from book)

Let C and P be two given sets of points in the plane, such that $k = |C|$ and $n = |P|$. Let $r = \max_{p \in P} \min_{c \in C} \|c - p\|$ be the covering radius of P by C.

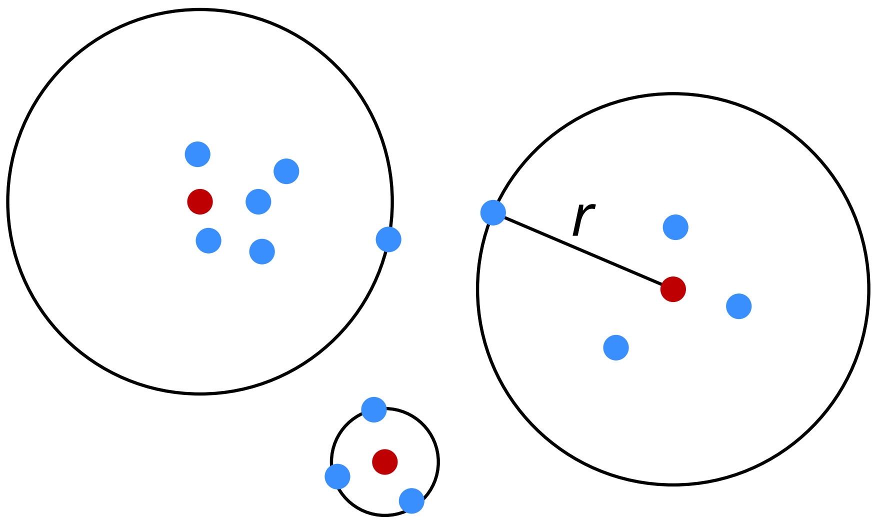


A) Give an $O(n + k \log n)$ expected time algorithm that outputs a number α, such that $r \leq \alpha \leq 10r$.

Exercise 1.2 (A): Decision Problem

Compute clustering radius (Exercise 1.2 from book)

Let C and P be two given sets of points in the plane, such that $k = |C|$ and $n = |P|$. Let $r = \max_{p \in P} \min_{c \in C} \|c - p\|$ be the covering radius of P by C.



A) Give an $O(n + k \log n)$ expected time algorithm that outputs a number α, such that $r \leq \alpha \leq 10r$.

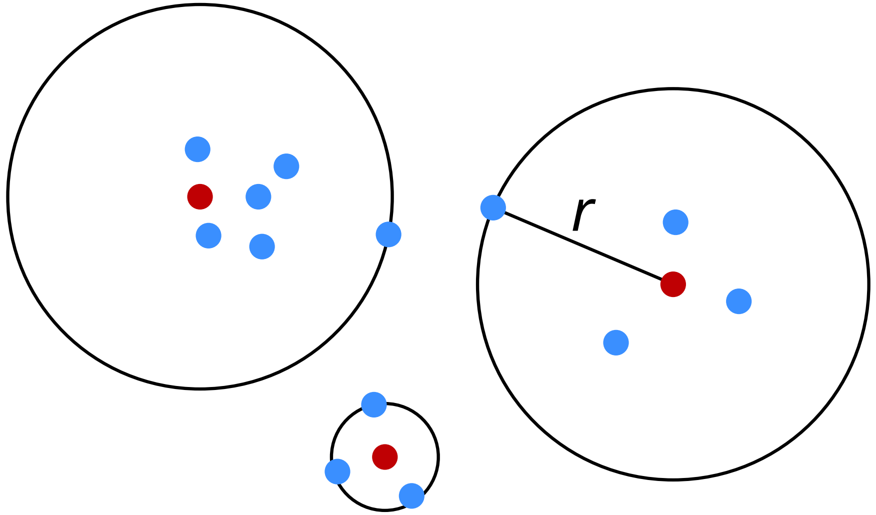
We already know how to:

A') Give an $O(n + k)$ algorithm (for given $\alpha' \geq 0$) that outputs
$$\begin{cases} \text{true} & \text{if } r \leq \alpha' \\ \text{false} & \text{if } r > 2\sqrt{2}\alpha' \\ \text{true or false} & \text{if } \alpha' < r \leq 2\sqrt{2}\alpha' \end{cases}$$

Exercise 1.2 (A): Decision Problem

Compute clustering radius (Exercise 1.2 from book)

Let C and P be two given sets of points in the plane, such that $k = |C|$ and $n = |P|$. Let $r = \max_{p \in P} \min_{c \in C} \|c - p\|$ be the covering radius of P by C.



A) Give an $O(n + k \log n)$ expected time algorithm that outputs a number α, such that $r \leq \alpha \leq 10r$.

We already know how to:

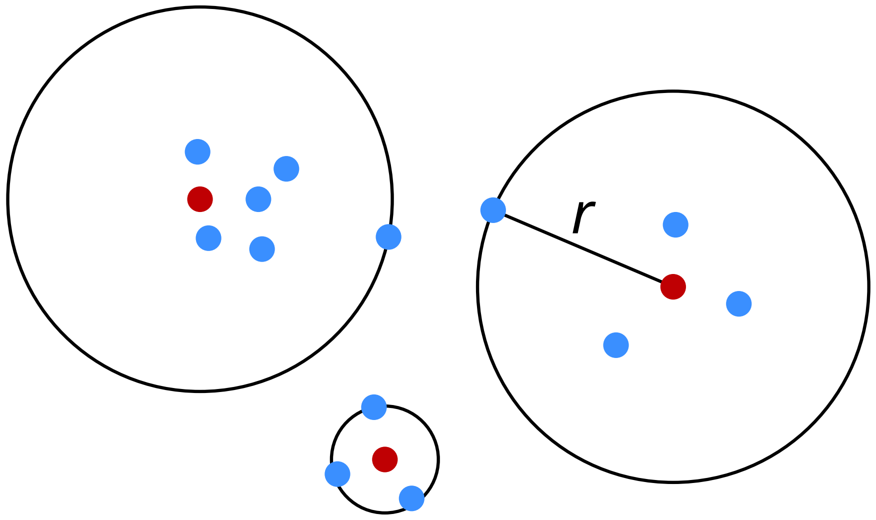
A') For given α', build a grid on C and return for a given $p \in P$ with distance r to C:

$$\begin{cases} \text{true} & \text{if } r \leq \alpha' \\ \text{false} & \text{if } r > 2\sqrt{2}\alpha' \\ \text{true or false} & \text{if } \alpha' < r \leq 2\sqrt{2}\alpha' \end{cases}$$

Exercise 1.2 (A): Decision Problem

Compute clustering radius (Exercise 1.2 from book)

Let C and P be two given sets of points in the plane, such that $k = |C|$ and $n = |P|$. Let $r = \max_{p \in P} \min_{c \in C} \|c - p\|$ be the covering radius of P by C.



A) Give an $O(n + k \log n)$ expected time algorithm that outputs a number α, such that $r \leq \alpha \leq 10r$.

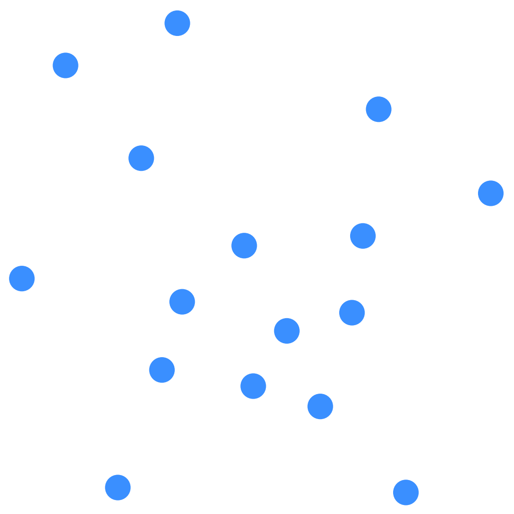
We already know how to:

A') For given α', build a grid on C and return for a given $p \in P$ with distance r to C:

$$\begin{cases} \text{true} & \text{if } r \leq \alpha' \\ \text{false} & \text{if } r > 2\sqrt{2}\alpha' \\ \text{true or false} & \text{if } \alpha' < r \leq 2\sqrt{2}\alpha' \end{cases}$$

Note: Careful in backward analysis (e.g. use canonical grids, i.e., change size by powers of 2)

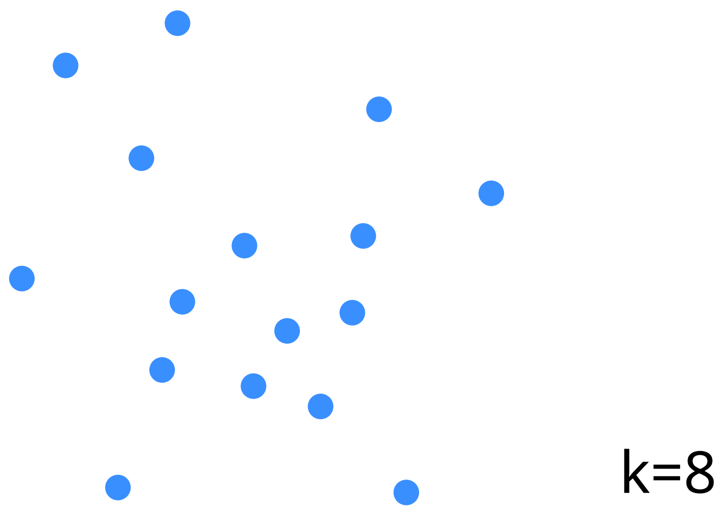
k-enclosing Disk Problem



k=8

k-enclosing Disk Problem

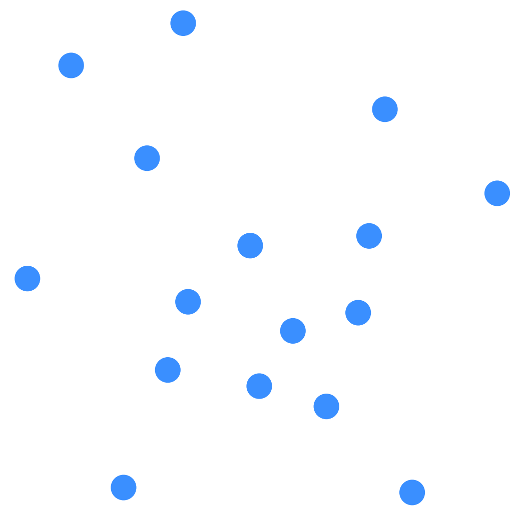
Input: point set *P* in the plane and $k \in \mathbb{N}$



k-enclosing Disk Problem

Input: point set P in the plane and $k \in \mathbb{N}$

Output: smallest k-enclosing disk, i.e., $|B_\varepsilon \cap P| \geq k$



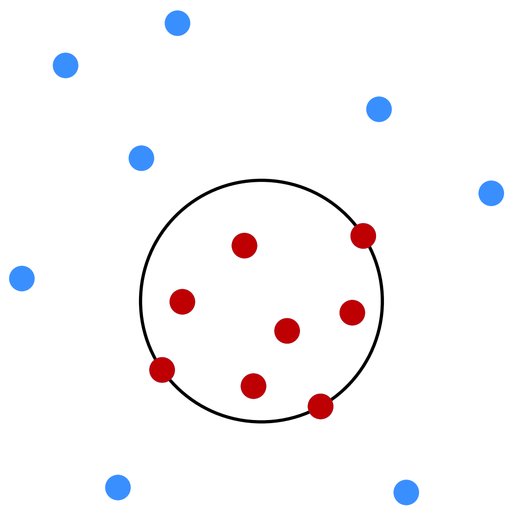
k=8

k-enclosing Disk Problem

Input: point set P in the plane and $k \in \mathbb{N}$

Output: smallest k-enclosing disk, i.e., $|B_\varepsilon \cap P| \geq k$

Observe: For a set P of n points, $q \in R^2$ and $k \in \mathbb{N}$, the k closest points to q can be found in $O(n)$ time.



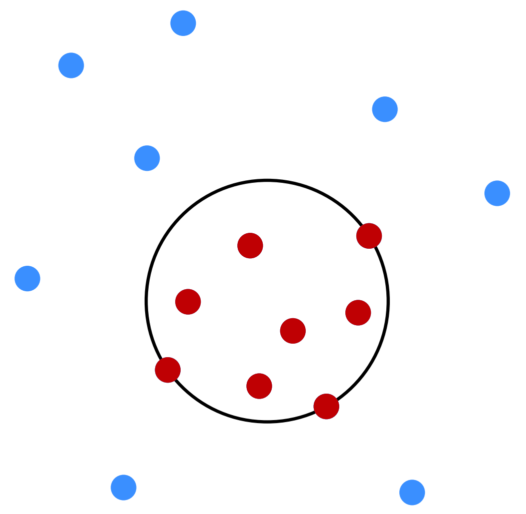
k=8

k-enclosing Disk Problem

Input: point set P in the plane and $k \in \mathbb{N}$

Output: smallest k-enclosing disk, i.e., $|B_\varepsilon \cap P| \geq k$

Observe: For a set P of n points, $q \in R^2$ and $k \in \mathbb{N}$,
the k closest points to q can be found in $O(n)$ time.



k=8

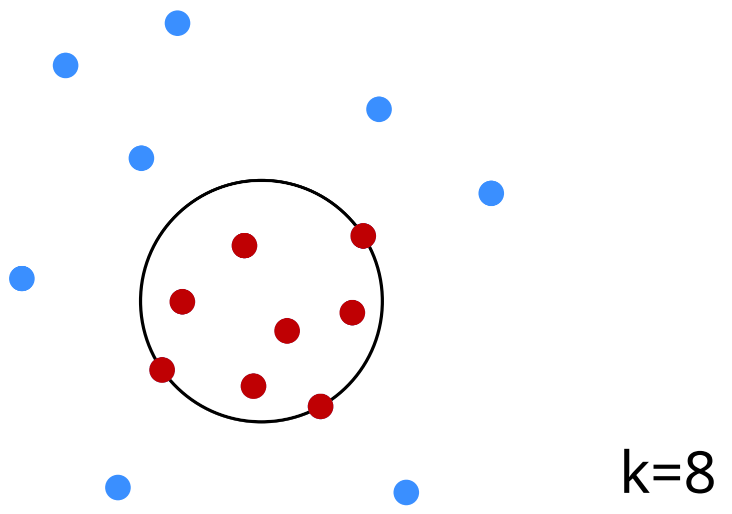
Approximation algorithm

Question: If we generalize the closest pair algorithm, which issues arise?

k-enclosing Disk Problem

Input: point set P in the plane and $k \in \mathbb{N}$

Output: smallest k-enclosing disk, i.e., $|B_\varepsilon \cap P| \geq k$



Observe: For a set P of n points, $q \in R^2$ and $k \in \mathbb{N}$, the k closest points to q can be found in $O(n)$ time.

Approximation algorithm

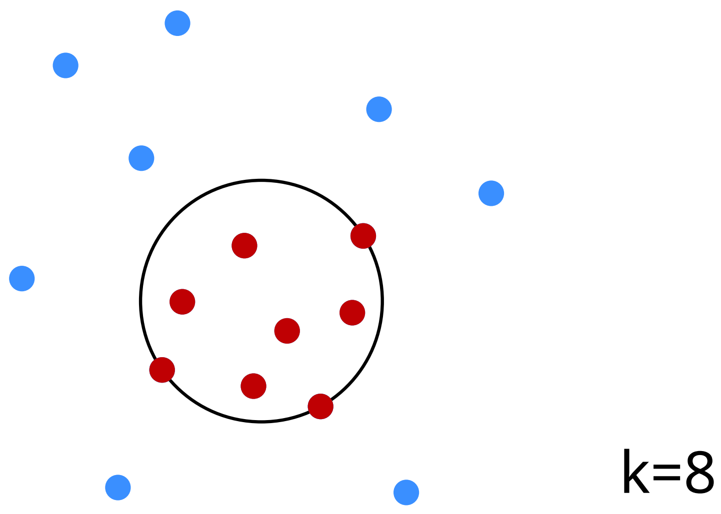
Question: If we generalize the closest pair algorithm, which issues arise?

Question: Where do we place (the centers of) disks?

k-enclosing Disk Problem

Input: point set P in the plane and $k \in \mathbb{N}$

Output: smallest k-enclosing disk, i.e., $|B_\varepsilon \cap P| \geq k$



Observe: For a set P of n points, $q \in R^2$ and $k \in \mathbb{N}$, the k closest points to q can be found in $O(n)$ time.

Approximation algorithm

Question: If we generalize the closest pair algorithm, which issues arise?

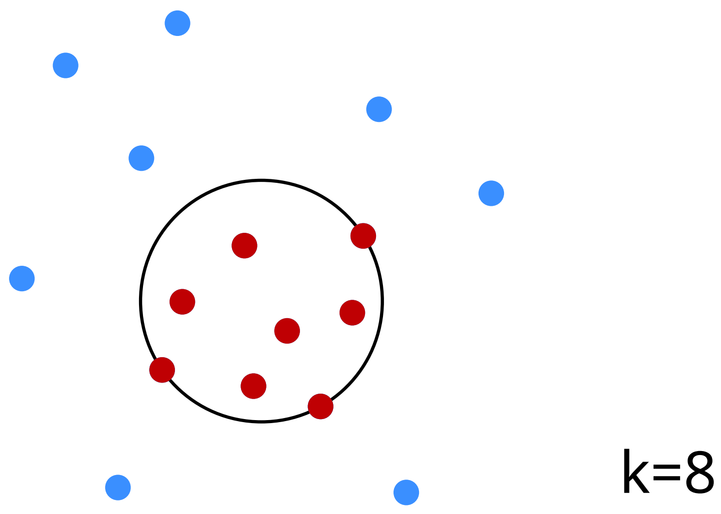
Question: Where do we place (the centers of) disks?

Question: Simple $O(n^2)$-time 2-approximation?

k-enclosing Disk Problem

Input: point set P in the plane and $k \in \mathbb{N}$

Output: smallest k-enclosing disk, i.e., $|B_\varepsilon \cap P| \geq k$



Observe: For a set P of n points, $q \in R^2$ and $k \in \mathbb{N}$, the k closest points to q can be found in $O(n)$ time.

Approximation algorithm

Question: If we generalize the closest pair algorithm, which issues arise?

Question: Where do we place (the centers of) disks?

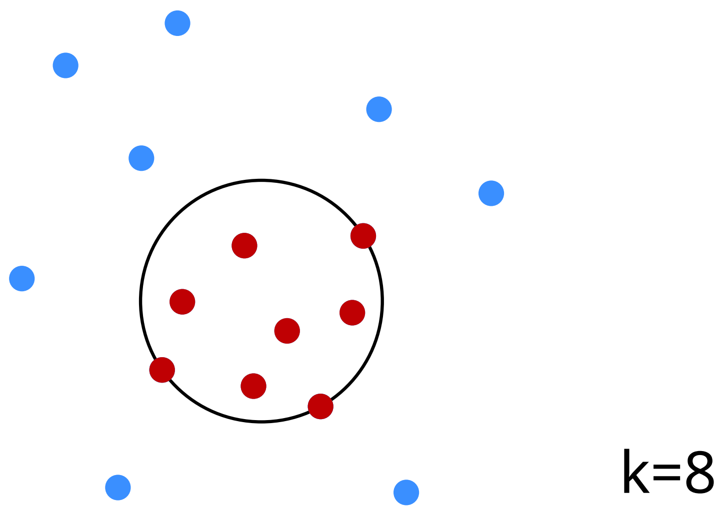
Question: Simple $O(n^2)$-time 2-approximation?

Question: If we generalize the closest pair algorithm with centers at point, what is the expected running time?

k-enclosing Disk Problem

Input: point set P in the plane and $k \in \mathbb{N}$

Output: smallest k-enclosing disk, i.e., $|B_\varepsilon \cap P| \geq k$



Observe: For a set P of n points, $q \in R^2$ and $k \in \mathbb{N}$, the k closest points to q can be found in $O(n)$ time.

Approximation algorithm

Question: If we generalize the closest pair algorithm, which issues arise?

Question: Where do we place (the centers of) disks?

Question: Simple $O(n^2)$-time 2-approximation?

Question: If we generalize the closest pair algorithm with centers at point, what is the expected running time?

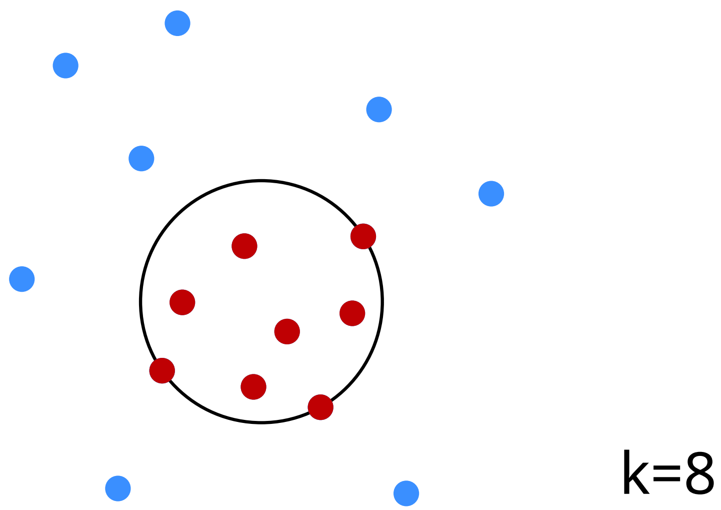
$\rightarrow$ $O(nk^2)$, too many centers, too many updates

k-enclosing Disk Problem

Input: point set P in the plane and $k \in \mathbb{N}$

Output: smallest k-enclosing disk, i.e., $|B_\varepsilon \cap P| \geq k$

Observe: For a set P of n points, $q \in R^2$ and $k \in \mathbb{N}$,
the k closest points to q can be found in $O(n)$ time.



Approximation algorithm

Question: If we generalize the closest pair algorithm, which issues arise?

Question: Where do we place (the centers of) disks?

Question: Simple $O(n^2)$-time 2-approximation?

Question: If we generalize the closest pair algorithm with centers at point, what is the expected running time?

$\rightarrow O(nk^2)$, too many centers, too many updates

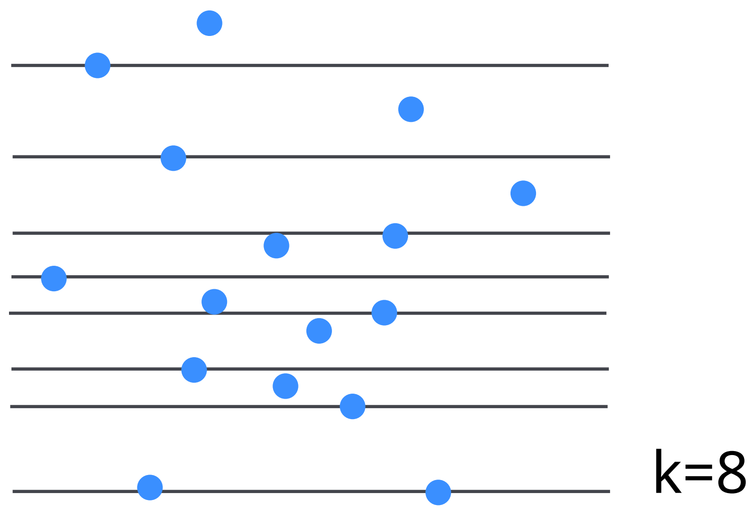
next: $O(n(n/k)^2)$-time 2-approximation; can be used to get $O(n)$-time 2-approximation

k-enclosing Disk Problem

Input: point set P in the plane and $k \in \mathbb{N}$

Output: smallest k-enclosing disk, i.e., $|B_\varepsilon \cap P| \geq k$

Observe: For a set P of n points, $q \in R^2$ and $k \in \mathbb{N}$, the k closest points to q can be found in $O(n)$ time.



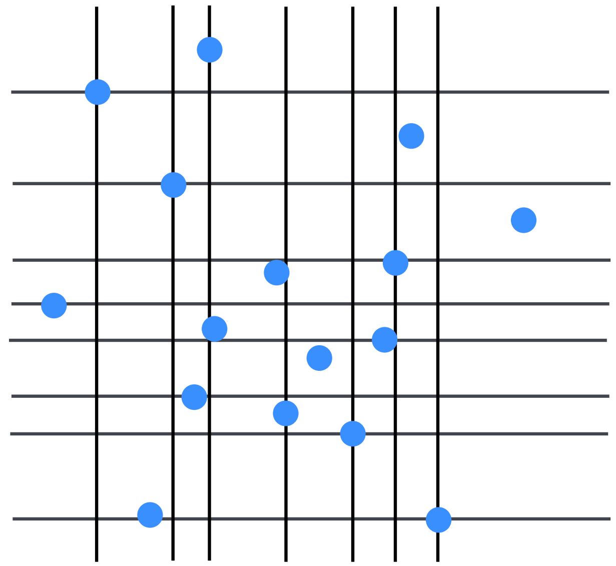
Approximation algorithm

- compute $m = O(n/k)$ horizontal lines $h_1, \dots, h_m$, s.t. there are at most $k/4$ points of P in between two consecutive lines

k-enclosing Disk Problem

Input: point set P in the plane and $k \in \mathbb{N}$

Output: smallest k-enclosing disk, i.e., $|B_\varepsilon \cap P| \geq k$



k=8

Observe: For a set P of n points, $q \in R^2$ and $k \in \mathbb{N}$, the k closest points to q can be found in $O(n)$ time.

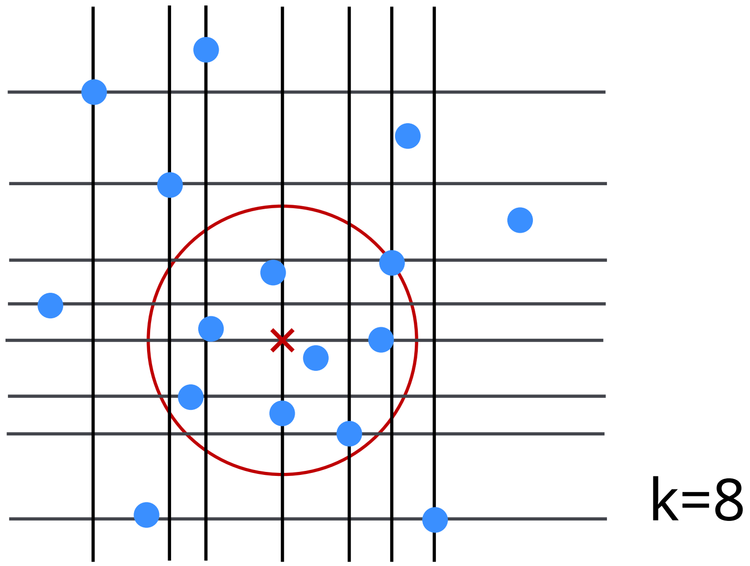
Approximation algorithm

- compute $m = O(n/k)$ horizontal lines $h_1, \dots, h_m$, s.t. there are at most $k/4$ points of P in between two consecutive lines
- compute $m = O(n/k)$ vertical lines $v_1, \dots, v_m$, s.t. there are at most $k/4$ points of P in between two consecutive lines

k-enclosing Disk Problem

Input: point set P in the plane and $k \in \mathbb{N}$

Output: smallest k-enclosing disk, i.e., $|B_\varepsilon \cap P| \geq k$



Observe: For a set P of n points, $q \in R^2$ and $k \in \mathbb{N}$, the k closest points to q can be found in $O(n)$ time.

Approximation algorithm

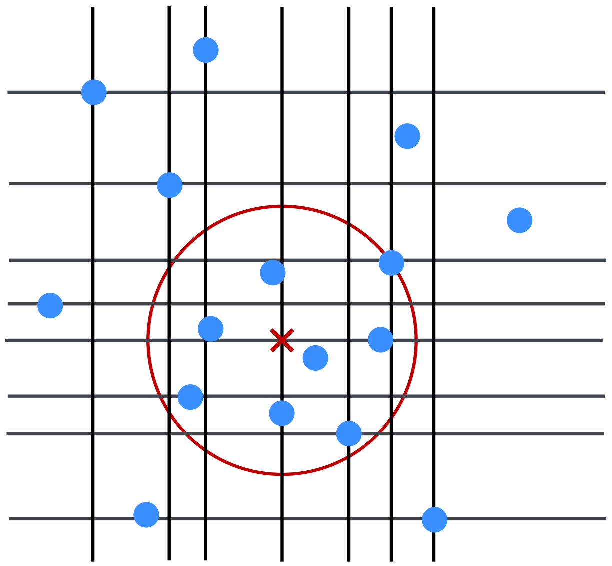
- compute $m = O(n/k)$ horizontal lines $h_1, \ldots, h_m$, s.t. there are at most $k/4$ points of P in between two consecutive lines
- compute $m = O(n/k)$ vertical lines $v_1, \ldots, v_m$, s.t. there are at most $k/4$ points of P in between two consecutive lines
- for each intersection point x of these lines

  compute the smallest disk containing k points of P

k-enclosing Disk Problem

Input: point set P in the plane and $k \in \mathbb{N}$

Output: smallest k-enclosing disk, i.e., $|B_\varepsilon \cap P| \geq k$



k=8

Observe: For a set P of n points, $q \in R^2$ and $k \in \mathbb{N}$, the k closest points to q can be found in $O(n)$ time.

Approximation algorithm

- compute $m = O(n/k)$ horizontal lines $h_1, \dots, h_m$,
  s.t. there are at most $k/4$ points of P in between two consecutive lines
- compute $m = O(n/k)$ vertical lines $v_1, \dots, v_m$,
  s.t. there are at most $k/4$ points of P in between two consecutive lines
- for each intersection point x of these lines
  compute the smallest disk containing k points of P
- return the smallest disk found

k-enclosing Disk Problem: Analysis

Runtime:

- Steps 1,2 can be achieved by recursively adding median lines, until at most $k/4$ points left

k-enclosing Disk Problem: Analysis

Runtime:

- Steps 1,2 can be achieved by recursively adding median lines, until at most $k/4$ points left

$$\text{Recurrence: } T(n) = 2T(n/2) + O(n) \text{ stopping at } n \leq k/4 \quad \rightarrow \quad O(n \log(n/k))$$

k-enclosing Disk Problem: Analysis

Runtime:

- Steps 1,2 can be achieved by recursively adding median lines, until at most $k/4$ points left

  Recurrence: $T(n) = 2T(n/2) + O(n)$ stopping at $n \leq k/4 \quad \rightarrow \quad O(n \log(n/k))$

- Step 3 takes $O(n)$ time for each of the m^2 grid points, where $m = O(n/k)$
  $\rightarrow O(n(n/k))^2$ time

k-enclosing Disk Problem: Analysis

Runtime:

- Steps 1,2 can be achieved by recursively adding median lines, until at most $k/4$ points left

  Recurrence: $T(n) = 2T(n/2) + O(n)$ stopping at $n \leq k/4 \quad \rightarrow \quad O(n \log(n/k))$
- Step 3 takes $O(n)$ time for each of the m^2 grid points, where $m = O(n/k)$
  $\rightarrow O(n(n/k))^2$ time

Approximation ratio: Observe that the optimal disk contains at least one grid point
$\rightarrow$ 2-Approximation

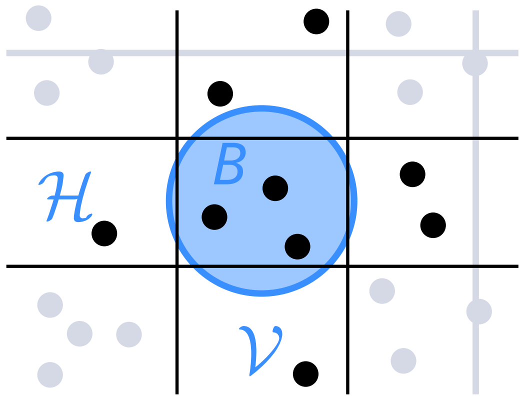
k-enclosing Disk Problem: Analysis

Runtime:

- Steps 1,2 can be achieved by recursively adding median lines, until at most $k/4$ points left

  Recurrence: $T(n) = 2T(n/2) + O(n)$ stopping at $n \leq k/4 \quad \rightarrow \quad O(n \log(n/k))$
- Step 3 takes $O(n)$ time for each of the m^2 grid points, where $m = O(n/k)$
  $\rightarrow O(n(n/k))^2$ time

Approximation ratio: Observe that the optimal disk contains at least one grid point
$\rightarrow$ 2-Approximation



If disk B contains no grid point, then $B \subset \mathcal{H} \cup \mathcal{V}$ for a horizonal slab $\mathcal{H}$ and vertical slab $\mathcal{V}$

k-enclosing Disk Problem: Analysis

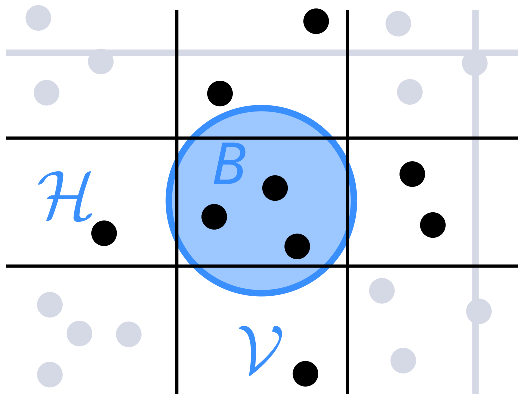
Runtime:

- Steps 1,2 can be achieved by recursively adding median lines, until at most $k/4$ points left

  Recurrence: $T(n) = 2T(n/2) + O(n)$ stopping at $n \leq k/4 \quad \rightarrow \quad O(n \log(n/k))$

- Step 3 takes $O(n)$ time for each of the m^2 grid points, where $m = O(n/k)$
  $\rightarrow O(n(n/k))^2$ time

Approximation ratio: Observe that the optimal disk contains at least one grid point
$\rightarrow$ 2-Approximation



If disk B contains no grid point, then $B \subset \mathcal{H} \cup \mathcal{V}$ for a horizonal slab $\mathcal{H}$ and vertical slab $\mathcal{V}$
$\rightarrow \; |P \cap B| \leq k/4 + k/4 = k/2 < k$

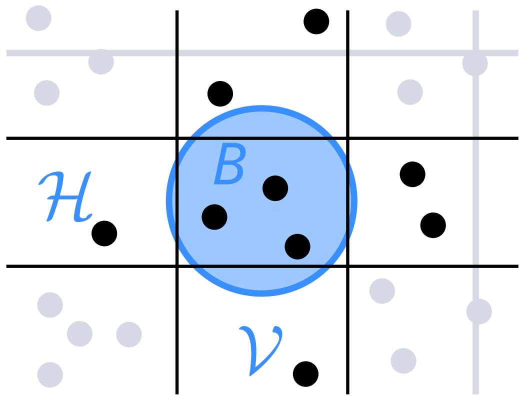
k-enclosing Disk Problem: Analysis

Runtime:

- Steps 1,2 can be achieved by recursively adding median lines, until at most $k/4$ points left

  Recurrence: $T(n) = 2T(n/2) + O(n)$ stopping at $n \leq k/4 \quad \rightarrow \quad O(n \log(n/k))$
- Step 3 takes $O(n)$ time for each of the m^2 grid points, where $m = O(n/k)$
  $\rightarrow O(n(n/k))^2$ time

Approximation ratio: Observe that the optimal disk contains at least one grid point
$\rightarrow$ 2-Approximation



If disk B contains no grid point, then $B \subset \mathcal{H} \cup \mathcal{V}$ for a horizonal slab $\mathcal{H}$ and vertical slab $\mathcal{V}$
$\rightarrow \ |P \cap B| \leq k/4 + k/4 = k/2 < k$

Using grids we can compute a 2-approximation of the k-enclosing disk in $O(n(n/k)^2)$ time.

Quiz

Given P let r_{OPT} be the radius of the smallest k-enclosing disk. If $\alpha < 2r_{OPT}$, what can we say about the maximum number gd_α of points in any cell of G_α?

A $gd_\alpha \leq 1$

B $gd_\alpha \leq 5k$

C gd_α can be arbitrary large ($\leq n$)

Quiz

Given P let r_{OPT} be the radius of the smallest k-enclosing disk. If $\alpha < 2r_{OPT}$, what can we say about the maximum number gd_α of points in any cell of G_α?

A $gd_\alpha \leq 1$

B $gd_\alpha \leq 5k$

C gd_α can be arbitrary large ($\leq n$)

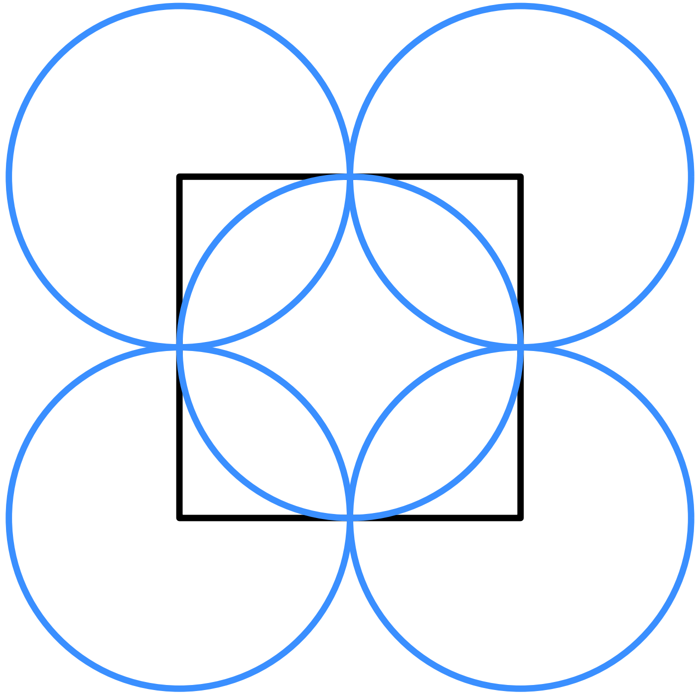
Quiz

Given P let r_{OPT} be the radius of the smallest k-enclosing disk. If $\alpha < 2r_{OPT}$, what can we say about the maximum number gd_α of points in any cell of G_α?

A $gd_\alpha \leq 1$

B $\boxed{gd_\alpha \leq 5k}$

C gd_α can be arbitrary large ($\leq n$)



each disk of radius r_OPT contains at most k points, cell covered by 5 disks

Linear-time 2-approximation for k-enclosing disk

k-Gradation: $P_1 \subset P_2 \subset P_m = P$ such that

(a) any point $p \in P_{i+1}$ is indepently with probability 1/2 included in P_i,

(b) $|P_1| \leq k$ and $|P_2| > k$

we name previous algorithm **algSlow:** outputs an α

Linear-time 2-approximation for k-enclosing disk

k-Gradation: $P_1 \subset P_2 \subset P_m = P$ such that

(a) any point $p \in P_{i+1}$ is indepently with probability 1/2 included in P_i,

(b) $|P_1| \leq k$ and $|P_2| > k$

we name previous algorithm **algSlow**: outputs an α

Algorithm

Compute gradation $P_1, \ldots, P_m$

$\alpha_1 \leftarrow$ **algSlow**(P_1)

for $i = 2, \ldots, m$ do

$\alpha_i \leftarrow$ **algGrow**(P_i, α_{i-1})

return α_m

Linear-time 2-approximation for k-enclosing disk

k-Gradation: $P_1 \subset P_2 \subset P_m = P$ such that

(a) any point $p \in P_{i+1}$ is indepently with probability 1/2 included in P_i,

(b) $|P_1| \leq k$ and $|P_2| > k$

we name previous algorithm **algSlow**: outputs an α

Algorithm

Compute gradation $P_1, \dots, P_m$

$\alpha_1 \leftarrow$ **algSlow**(P_1)

for $i = 2, \dots, m$ do

$\alpha_i \leftarrow$ **algGrow**(P_i, α_{i-1})

return α_m

AlgGrow(P_i, α_{i-1})

for every grid cluster c with $\geq k$ points: compute $\alpha_c \rightarrow$ **algSlow**($c \cap P_i$)

return the minimum of these α_c

Linear-time 2-approximation for k-enclosing disk

k-Gradation: $P_1 \subset P_2 \subset P_m = P$ such that

(a) any point $p \in P_{i+1}$ is indepently with probability 1/2 included in P_i,

(b) $|P_1| \leq k$ and $|P_2| > k$

we name previous algorithm **algSlow**: outputs an α

Algorithm

Compute gradation $P_1, \dots, P_m$

$\alpha_1 \leftarrow$ **algSlow**(P_1)

for $i = 2, \dots, m$ do

  $\alpha_i \leftarrow$ **algGrow**(P_i, α_{i-1})

return α_m

AlgGrow(P_i, α_{i-1})

for every grid cluster c with $\geq k$ points: compute $\alpha_c \rightarrow$ **algSlow**$(c \cap P_i)$

return the minimum of these α_c

Analysis:discussion + see book

Outline

1. Closest Pair Problem – part 1
   Exercise 1.2 (A) – part 1

2. Closest Pair Problem – part 2
   Exercise 1.2 (A) – part 2

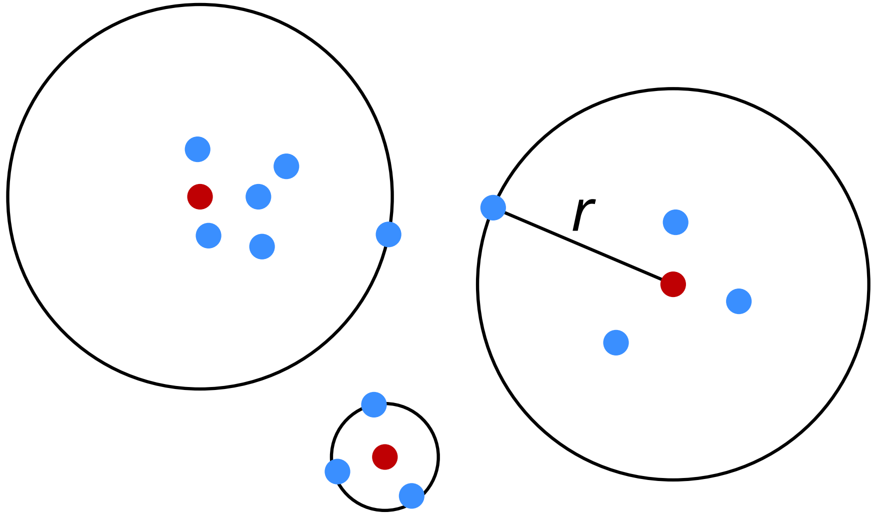
3. *k*-enclosing Disk Problem

4. (1 + ε)-approximation: Exercise 1.2 (B)

Exercise 1.2 (B)

Compute clustering radius (Exercise 1.2 from book)

Let C and P be two given sets of points in the plane, such that $k = |C|$ and $n = |P|$. Let $r = \max_{p \in P} \min_{c \in C} \|c - p\|$ be the covering radius of P by C.



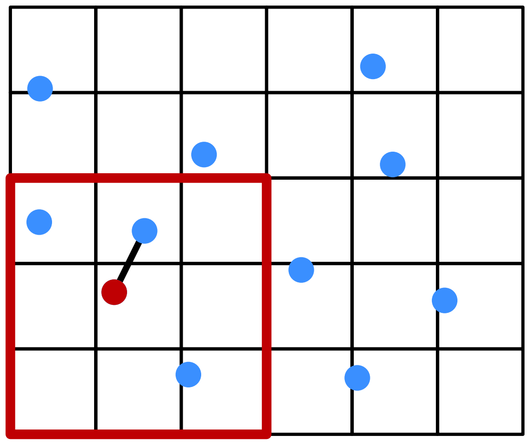
A) Give an $O(n + k \log n)$ expected time algorithm that outputs a number α, such that $r \leq \alpha \leq 10r$.

B) For $\varepsilon > 0$, give an $O(n + k\varepsilon^{-2} \log n)$ expected time algorithm that outputs a number α, such that $\alpha \leq r \leq (1 + \varepsilon)\alpha$.

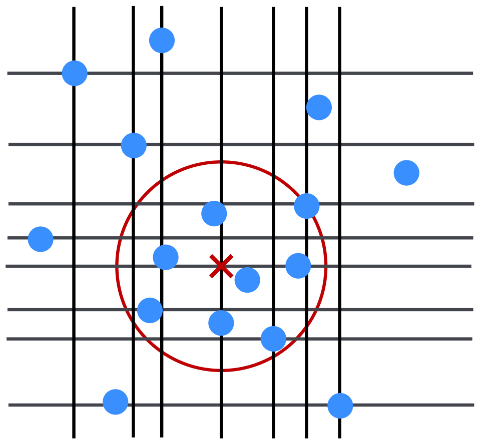
Exercise 1.2 (B)

For $\varepsilon > 0$, give an $O(n + k\varepsilon^{-2} \log n)$ expected time algorithm that outputs a number α, such that $\alpha \leq r \leq (1 + \varepsilon)\alpha$.

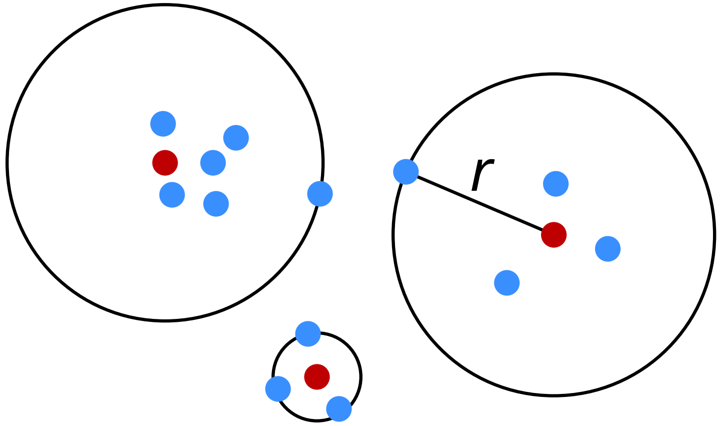
Summary
Using grids for approximation



closest pair (exact)

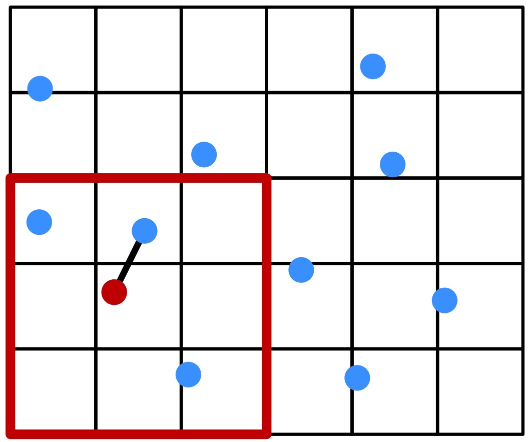


k-enclosing disk

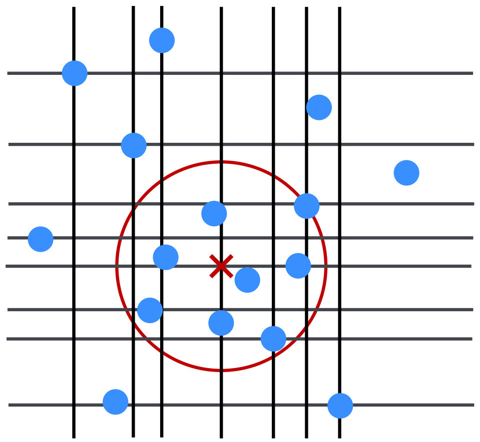


cluster radius

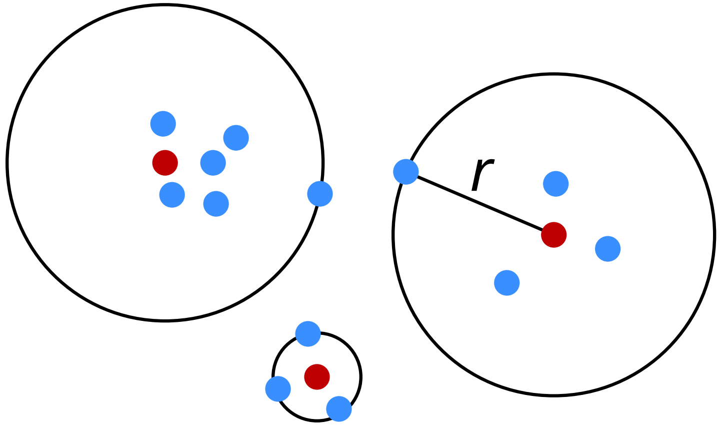
Summary
Using grids for approximation



closest pair (exact)



k-enclosing disk



cluster radius

Techniques

- approximate decision problem allows to fix the grid
- reduce optimization problem to decision problem using few calls: randomization (in other settings also binary search)
- grid vertices as candidate solution
- $(1 + \varepsilon)$-approximation: grid cells of sidelength $O(\alpha\varepsilon)$ (do Exercise 1.2.B!)