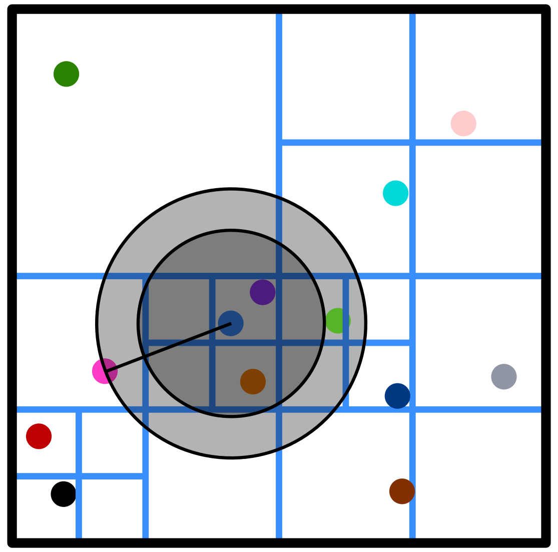
Semi-separated pair decomposition &

low-quality approximate nearest neighbors

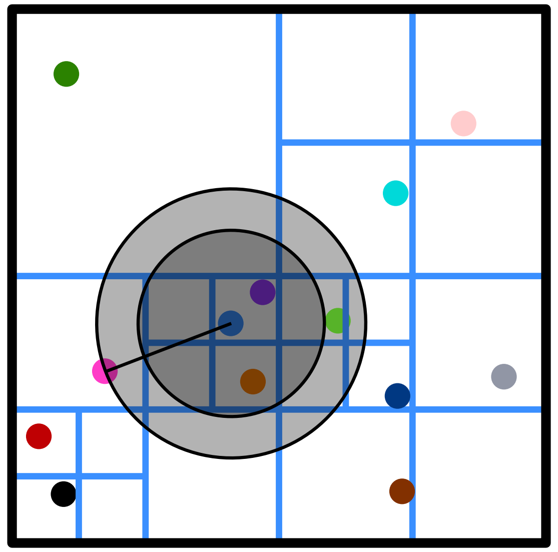
Motivation: low-quality approximate nearest neighbors

1. (Compressed) quadtree returns $(1+\varepsilon)$-approximate nearest neighbor of q in time $O(1/\varepsilon^d + \log(1/r))$ time, where $r = d(q, nn(q))$



Motivation: low-quality approximate nearest neighbors

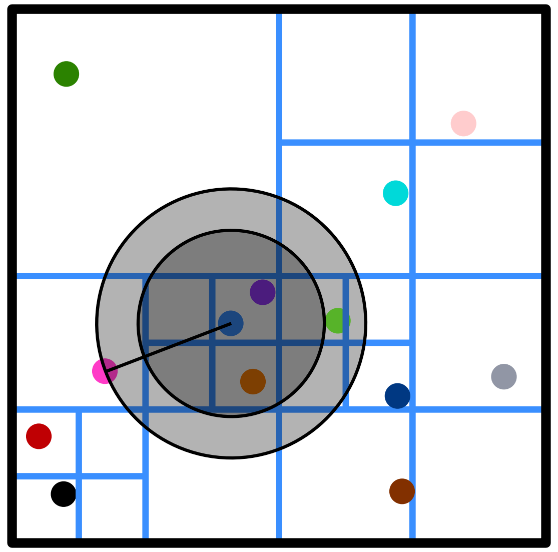
1. (Compressed) quadtree returns $(1+\varepsilon)$-approximate nearest neighbor of q in time $O(1/\varepsilon^d + \log(1/r))$ time, where $r = d(q, nn(q))$



Problem: r can be arbitrary small.

Motivation: low-quality approximate nearest neighbors

1. (Compressed) quadtree returns $(1+\varepsilon)$-approximate nearest neighbor of q in time $O(1/\varepsilon^d + \log(1/r))$ time, where $r = d(q, nn(q))$

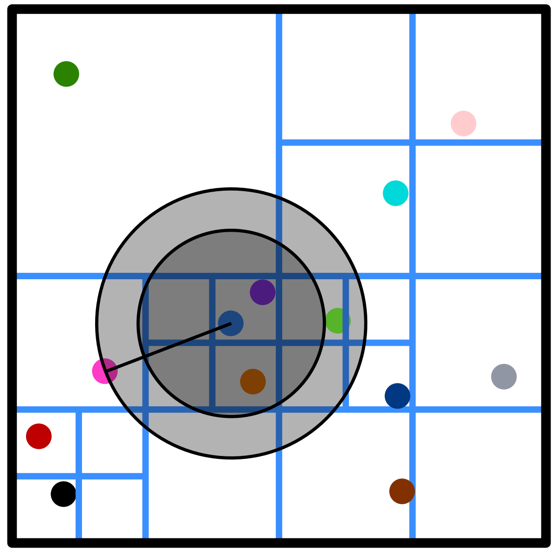


Problem: r can be arbitrary small.

2. If we can compute an $O(n)$-ANN p, then we can

Motivation: low-quality approximate nearest neighbors

1. (Compressed) quadtree returns $(1+\varepsilon)$-approximate nearest neighbor of q in time $O(1/\varepsilon^d + \log(1/r))$ time, where $r = d(q, nn(q))$

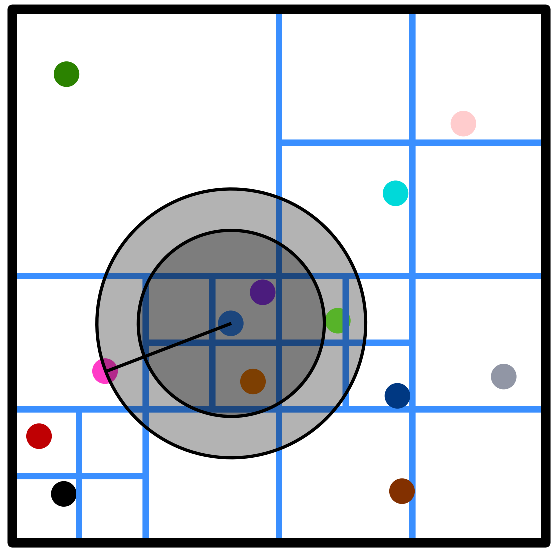


Problem: r can be arbitrary small.

2. If we can compute an $O(n)$-ANN p, then we can
   (a) Find the $O(1)$ cells of G_α that could contain $nn(q)$,
   where $\alpha = \|p - q\|$ rounded down to the next power 2^{-i}

Motivation: low-quality approximate nearest neighbors

1. (Compressed) quadtree returns $(1+\varepsilon)$-approximate nearest neighbor of q in time $O(1/\varepsilon^d + \log(1/r))$ time, where $r = d(q, nn(q))$

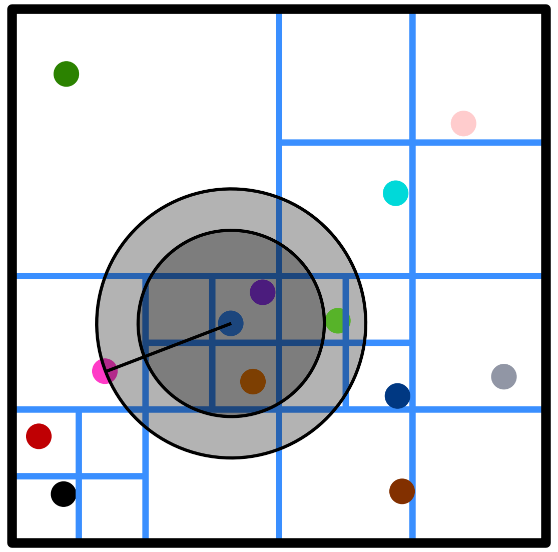


Problem: r can be arbitrary small.

2. If we can compute an $O(n)$-ANN p, then we can
   (a) Find the $O(1)$ cells of G_α that could contain $nn(q)$, where $\alpha = \|p - q\|$ rounded down to the next power 2^{-i}
   (b) Use (1.), starting from these cells

Motivation: low-quality approximate nearest neighbors

1. (Compressed) quadtree returns $(1+\varepsilon)$-approximate nearest neighbor of q in time $O(1/\varepsilon^d + \log(1/r))$ time, where $r = d(q, nn(q))$



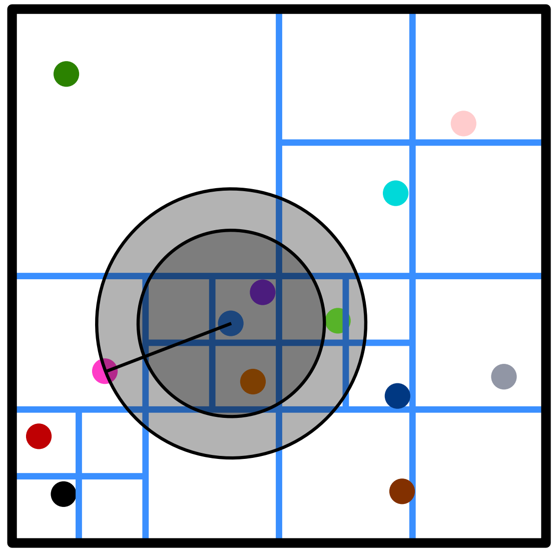
Problem: r can be arbitrary small.

2. If we can compute an $O(n)$-ANN p, then we can
   (a) Find the $O(1)$ cells of G_α that could contain $nn(q)$,
   where $\alpha = \|p - q\|$ rounded down to the next power 2^{-i}
   (b) Use (1.), starting from these cells

Now, $1/r = O(n)$ relative to $\alpha = O(nr)$

Motivation: low-quality approximate nearest neighbors

1. (Compressed) quadtree returns $(1+\varepsilon)$-approximate nearest neighbor of q in time $O(1/\varepsilon^d + \log(1/r))$ time, where $r = d(q, nn(q))$



Problem: r can be arbitrary small.

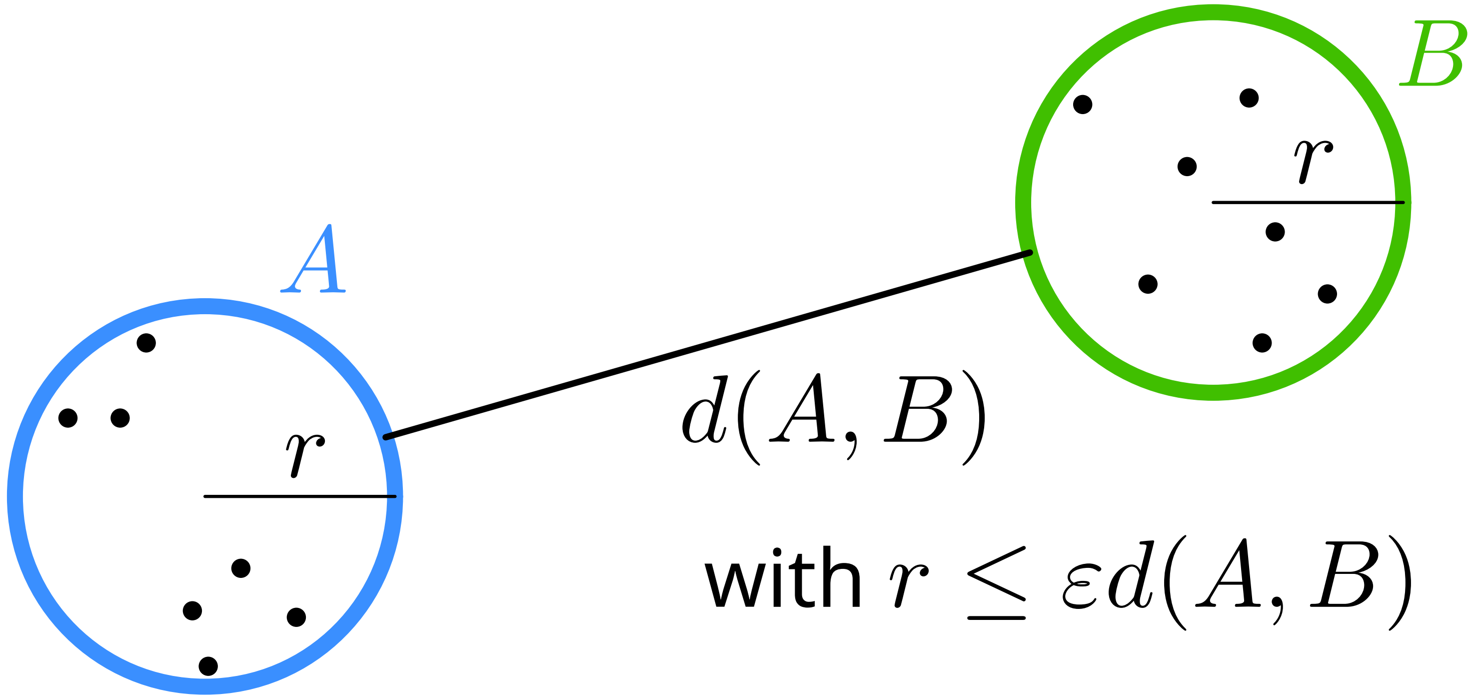
2. If we can compute an $O(n)$-ANN p, then we can
   (a) Find the $O(1)$ cells of G_α that could contain $nn(q)$,
   where $\alpha = \|p - q\|$ rounded down to the next power 2^{-i}
   (b) Use (1.), starting from these cells

Now, $1/r = O(n)$ relative to $\alpha = O(nr)$

This requires $O(n)$-ANN $\rightarrow$ today

Motivation 2: The weight of the WSPD

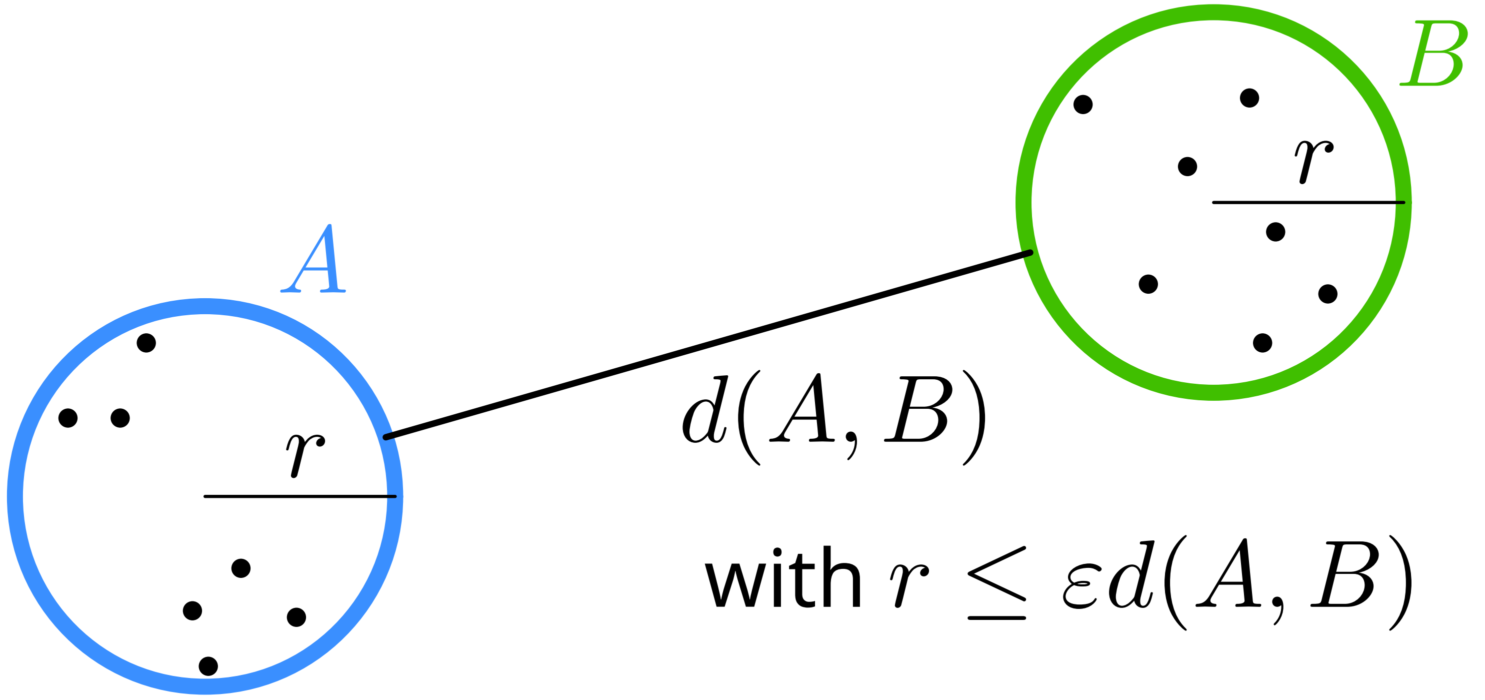
Well-separated pair decomposition: Cover all pairs of points by $1/\varepsilon$-well-separated pairs of point sets $\{A, B\}$:



with $r \leq \varepsilon d(A, B)$

Motivation 2: The weight of the WSPD

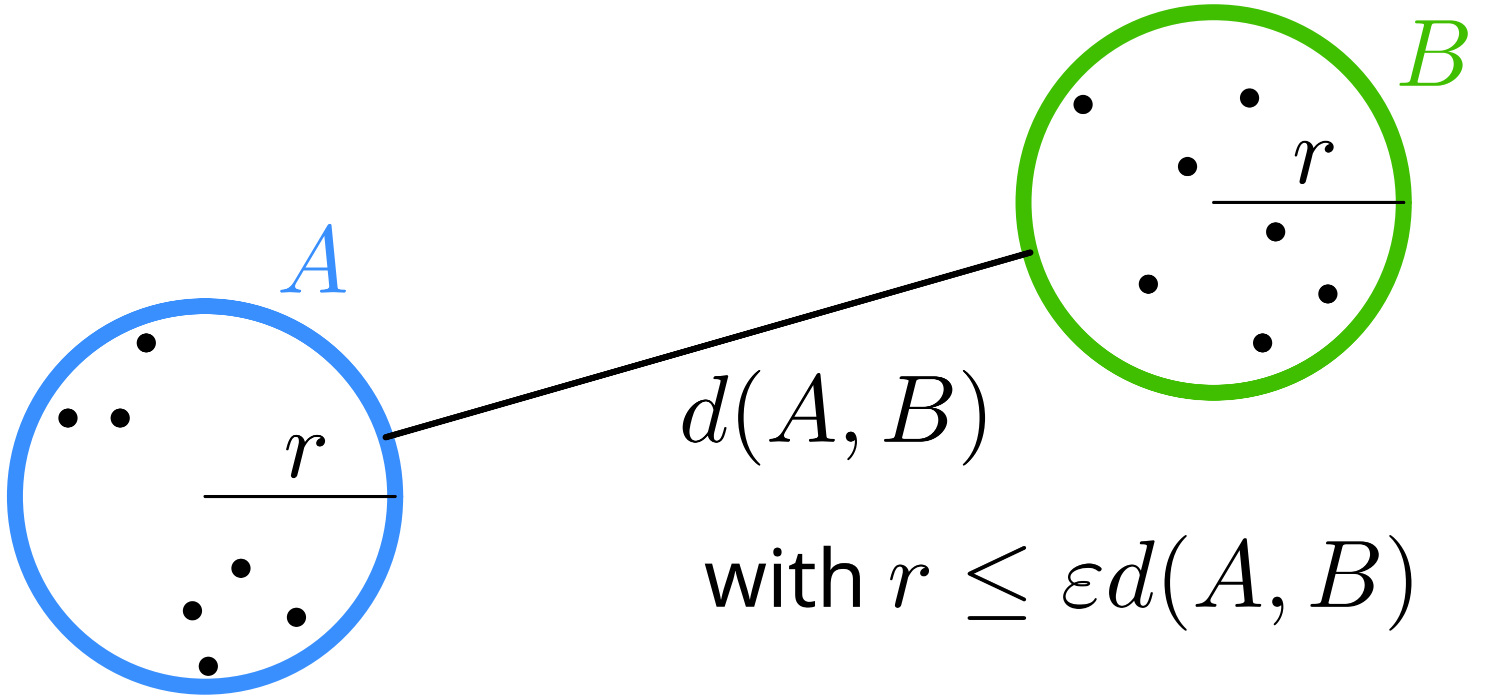
Well-separated pair decomposition: Cover all pairs of points by $1/\varepsilon$-well-separated pairs of point sets $\{A, B\}$:



good: $O(n)$ pairs are enough (size of WSPD)

Motivation 2: The weight of the WSPD

Well-separated pair decomposition: Cover all pairs of points by $1/\varepsilon$-well-separated pairs of point sets $\{A, B\}$:



with $r \leq \varepsilon d(A, B)$

good: $O(n)$ pairs are enough (size of WSPD)

but: might require $\sum_i |A_i| + |B_i| = \Theta(n^2)$ (weight of WSPD)

Overview

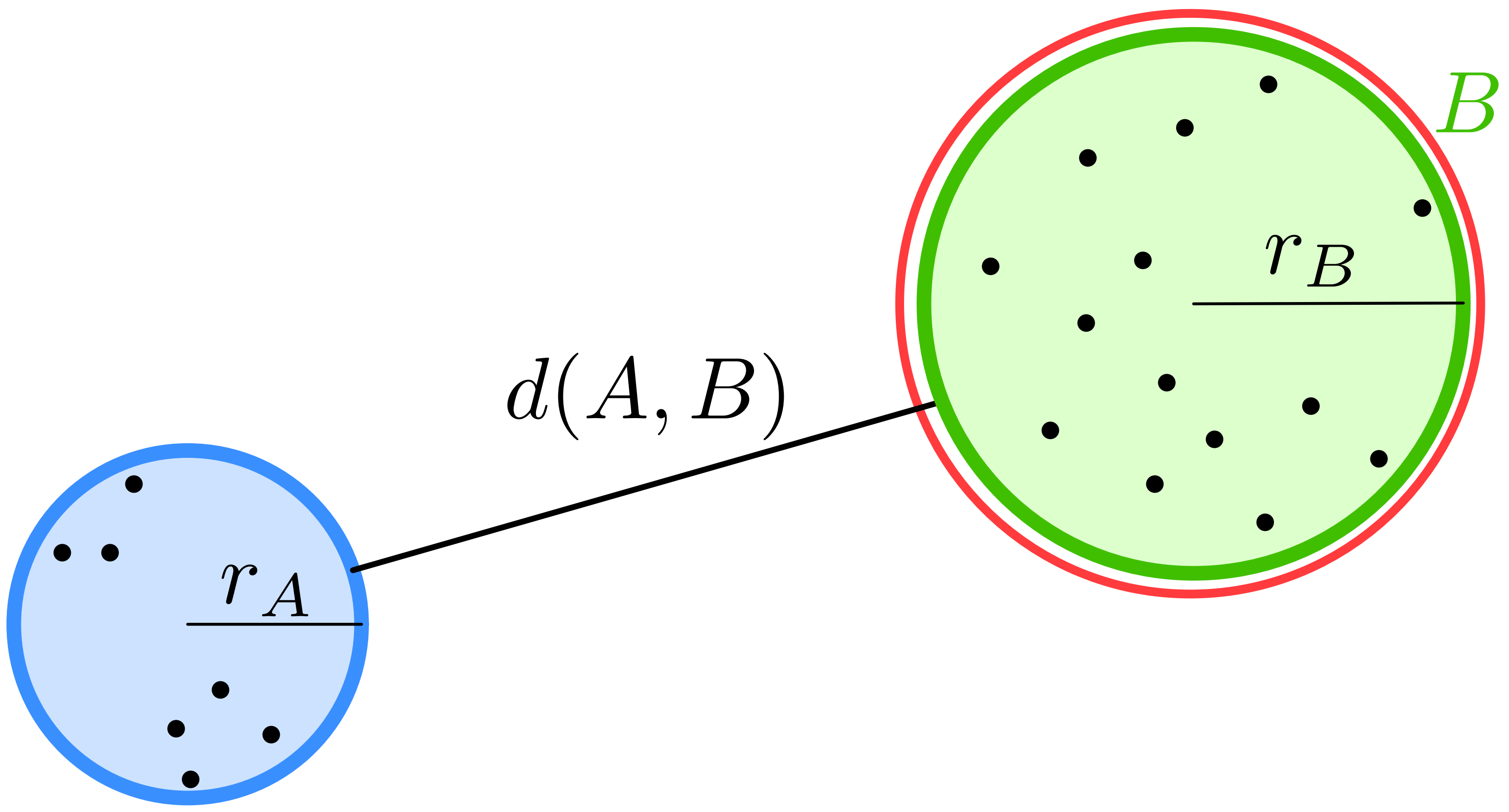
Semi-separated pair decomposition (SSPD)

Ring separator tree: n-semi-separated pair decomposition

Ring separator tree: $O(n)$-ANN

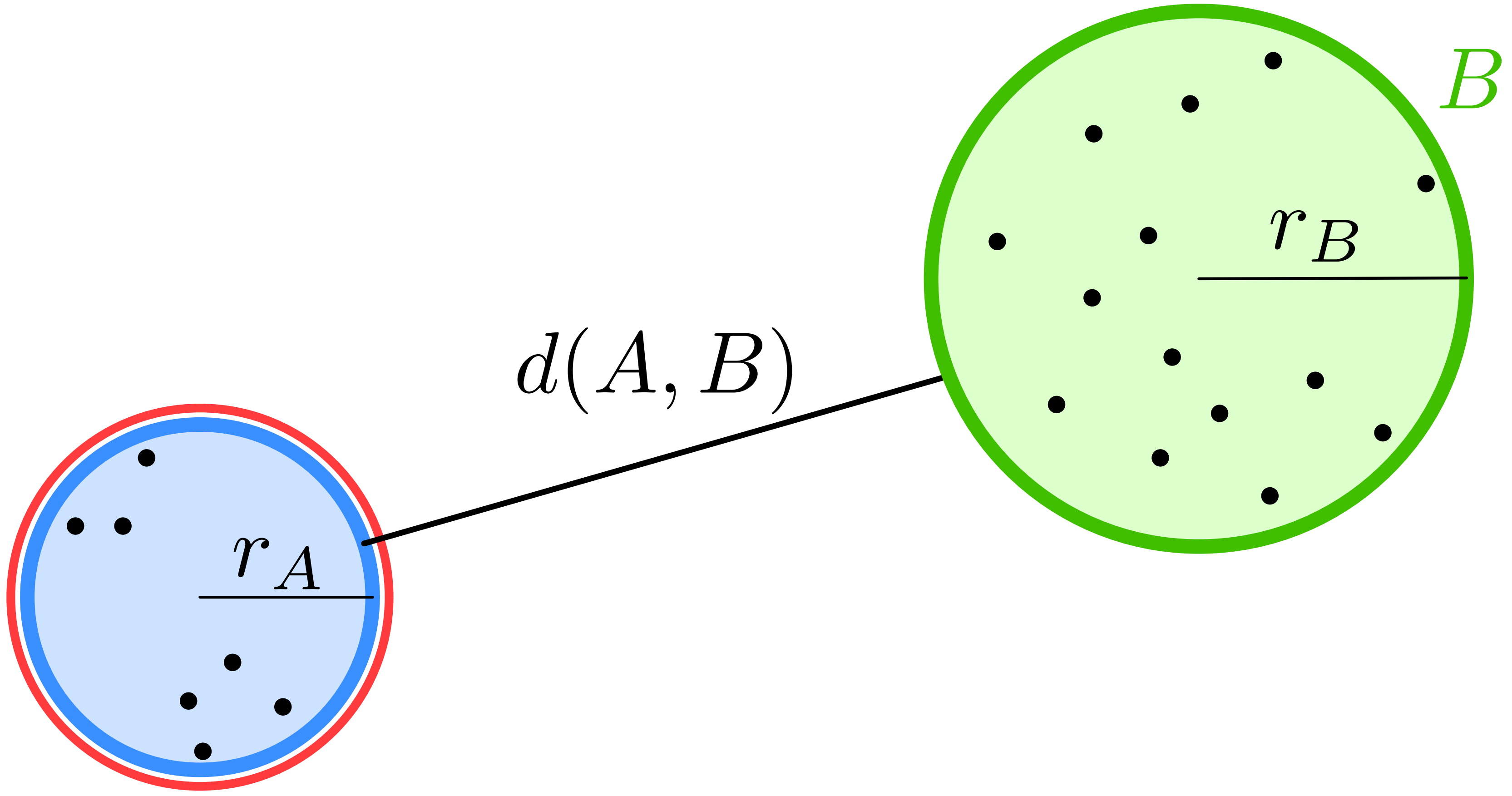
$(1/\varepsilon)$-semi-separated pair decomposition

Semi-Separated Pairs



$1/\varepsilon$-well-separated pair: $\max(r_A, r_B) \leq \varepsilon d(A, B)$

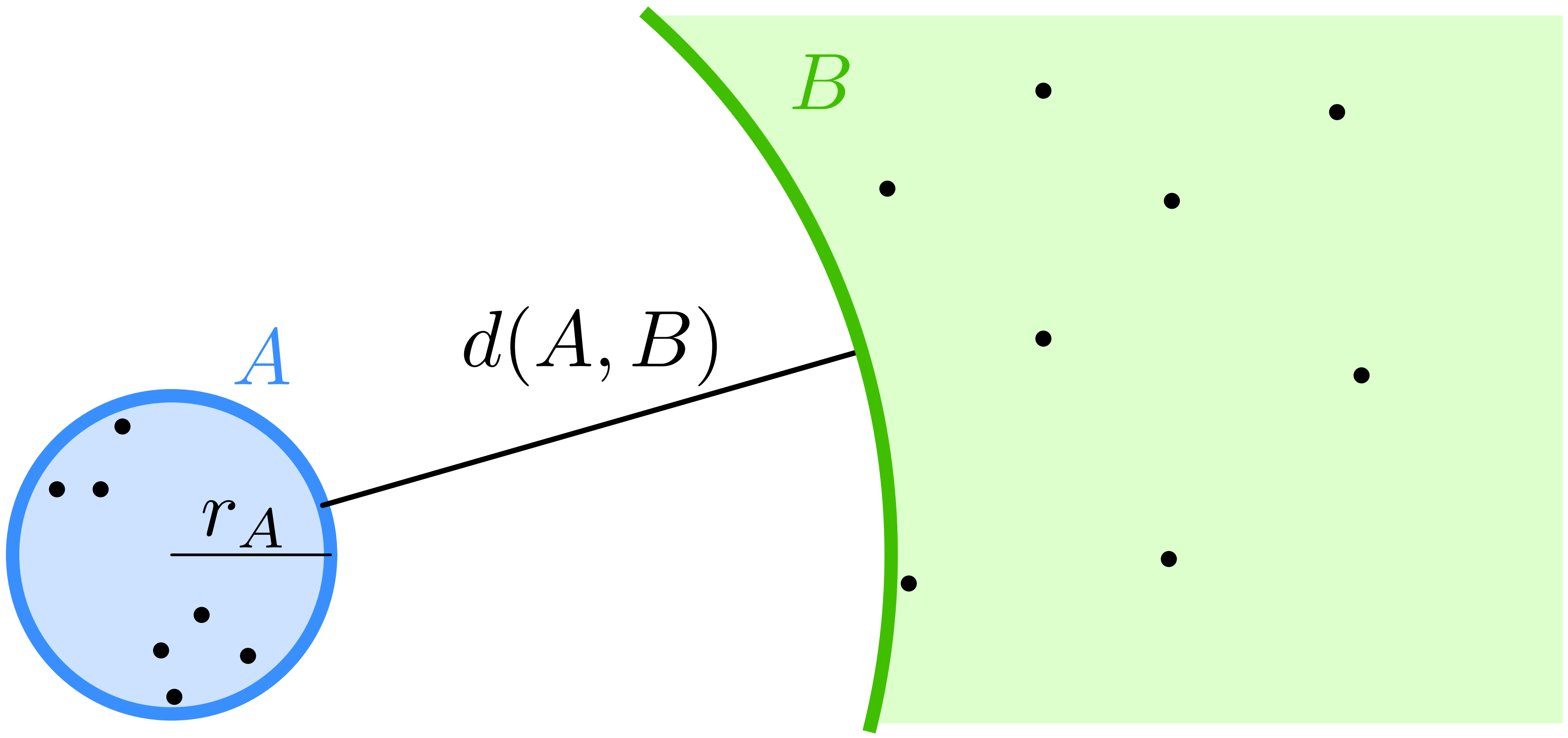
Semi-Separated Pairs



$1/\varepsilon$-well-separated pair: $\max(r_A, r_B) \leq \varepsilon d(A, B)$

$1/\varepsilon$-semi-separated pair: $\min(r_A, r_B) \leq \varepsilon d(A, B)$

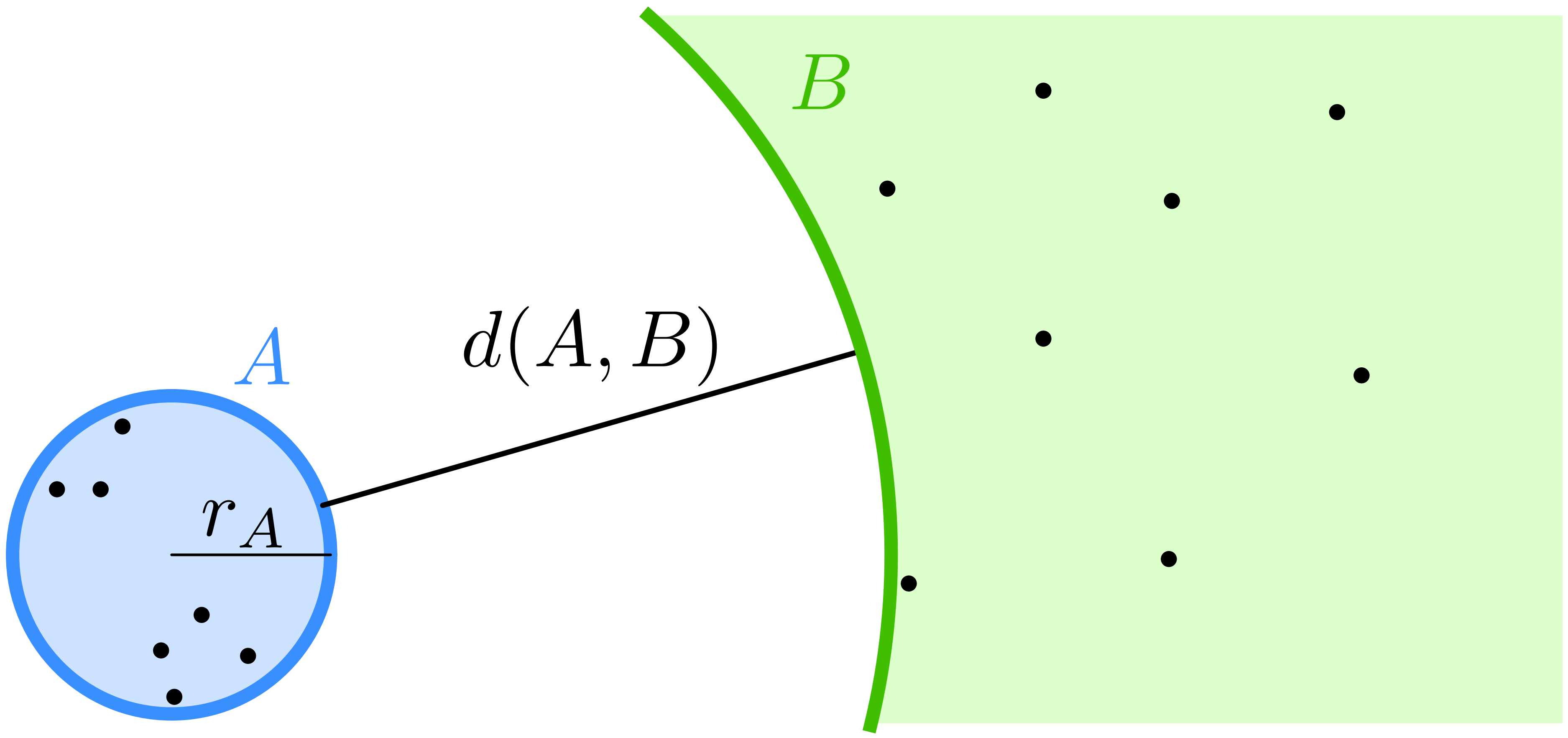
Semi-Separated Pairs



$1/\varepsilon$-well-separated pair: $\max(r_A, r_B) \leq \varepsilon d(A, B)$

$1/\varepsilon$-semi-separated pair: $\min(r_A, r_B) \leq \varepsilon d(A, B)$

Semi-Separated Pairs



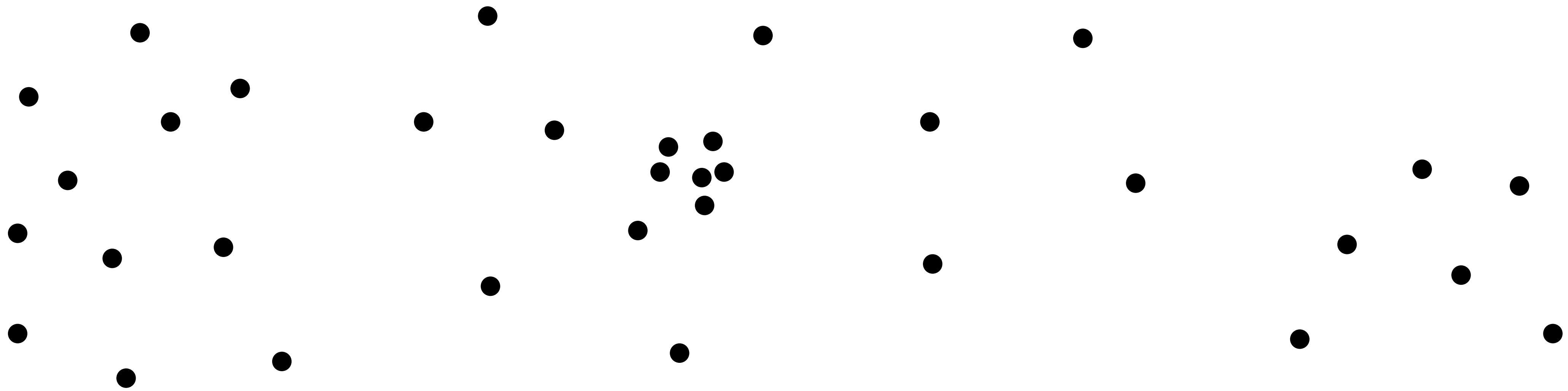
ring separator:

- ball $b = b(p, r)$ containing $\geq n/c_1$ points
- no point in $b(p, r(1 + 1/n)) \setminus b$
- $\geq n/c_2$ points outside $b(p, 2r)$

Computing a ring separator

1. Compute $b = b(p, \alpha)$: 2-approximation of smallest ball containing n/c_1 points

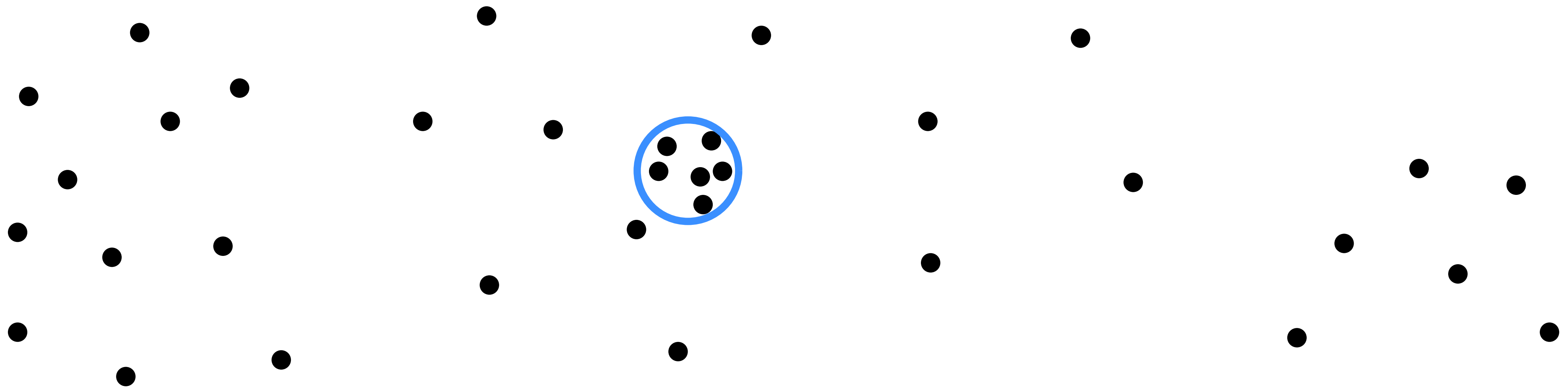
(c_1 to be determined later)



Computing a ring separator

1. Compute $b = b(p, \alpha)$: 2-approximation of smallest ball containing n/c_1 points

(c_1 to be determined later)

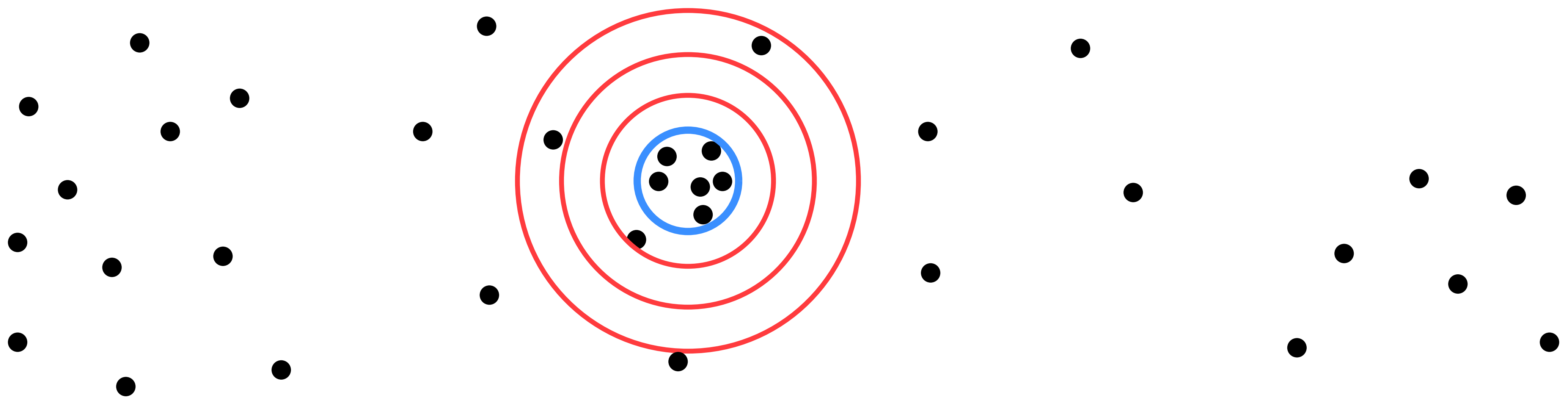


Computing a ring separator

1. Compute $b = b(p, \alpha)$: 2-approximation of smallest ball containing n/c_1 points

(c_1 to be determined later)

2. Hash the points in $b(p, e \cdot \alpha) \setminus b$ into rings $r_i := b_i \setminus b_{i-1}$, where $b_i := b(p, r(1+1/n)^i$, $b_0 = b$

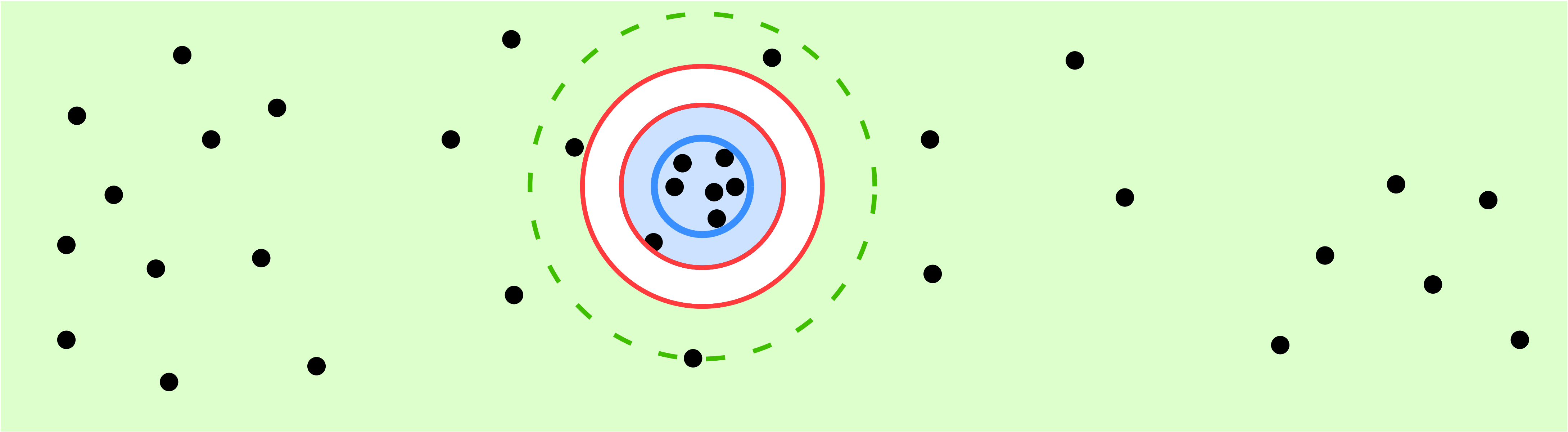


Computing a ring separator

1. Compute $b = b(p, \alpha)$: 2-approximation of smallest ball containing n/c_1 points (c_1 to be determined later)

2. Hash the points in $b(p, e \cdot \alpha) \setminus b$ into rings $r_i := b_i \setminus b_{i-1}$, where $b_i := b(p, r(1 + 1/n)^i$, $b_0 = b$

3. Find empty ring r_i and return b_{i-1}



Computing a ring separator

1. Compute $b = b(p, \alpha)$: 2-approximation of smallest ball containing n/c_1 points

(c_1 to be determined later)

2. Hash the points in $b(p, e \cdot \alpha) \setminus b$ into rings $r_i := b_i \setminus b_{i-1}$, where $b_i := b(p, r(1+1/n)^i$, $b_0 = b$

3. Find empty ring r_i and return b_{i-1}

Quiz What is the (expected) worst-case running time of the algorithm?

A $\Theta(n)$

B $\Theta(n \log n)$

C $\Theta(n^2)$

Computing a ring separator

1. Compute $b = b(p, \alpha)$: 2-approximation of smallest ball containing n/c_1 points

(c_1 to be determined later)

2. Hash the points in $b(p, e \cdot \alpha) \setminus b$ into rings $r_i := b_i \setminus b_{i-1}$, where $b_i := b(p, r(1 + 1/n)^i$, $b_0 = b$

3. Find empty ring r_i and return b_{i-1}

Quiz What is the (expected) worst-case running time of the algorithm?

$\boxed{\text{A} \quad \Theta(n)}$

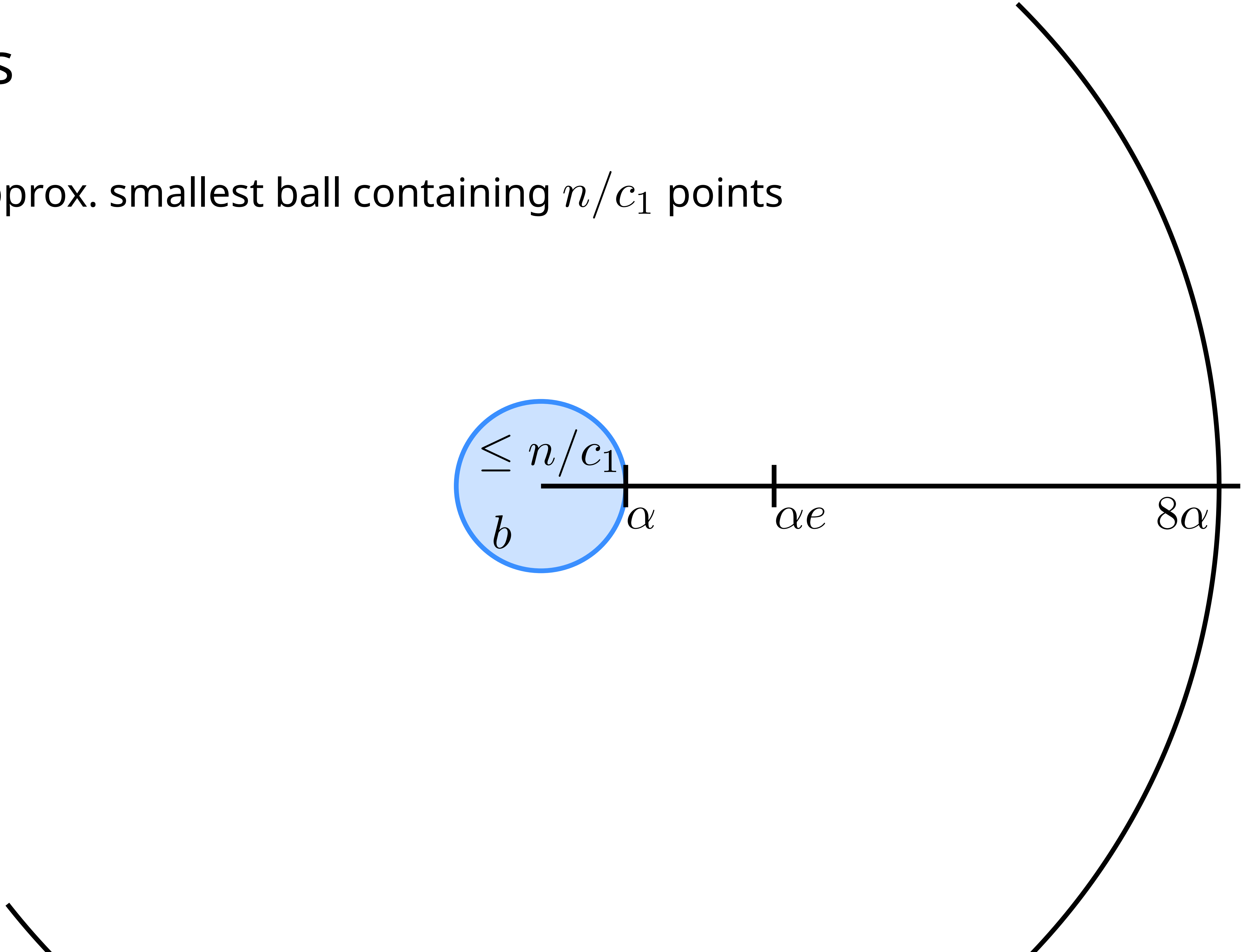
B $\Theta(n \log n)$

C $\Theta(n^2)$

Correctness

Correctness

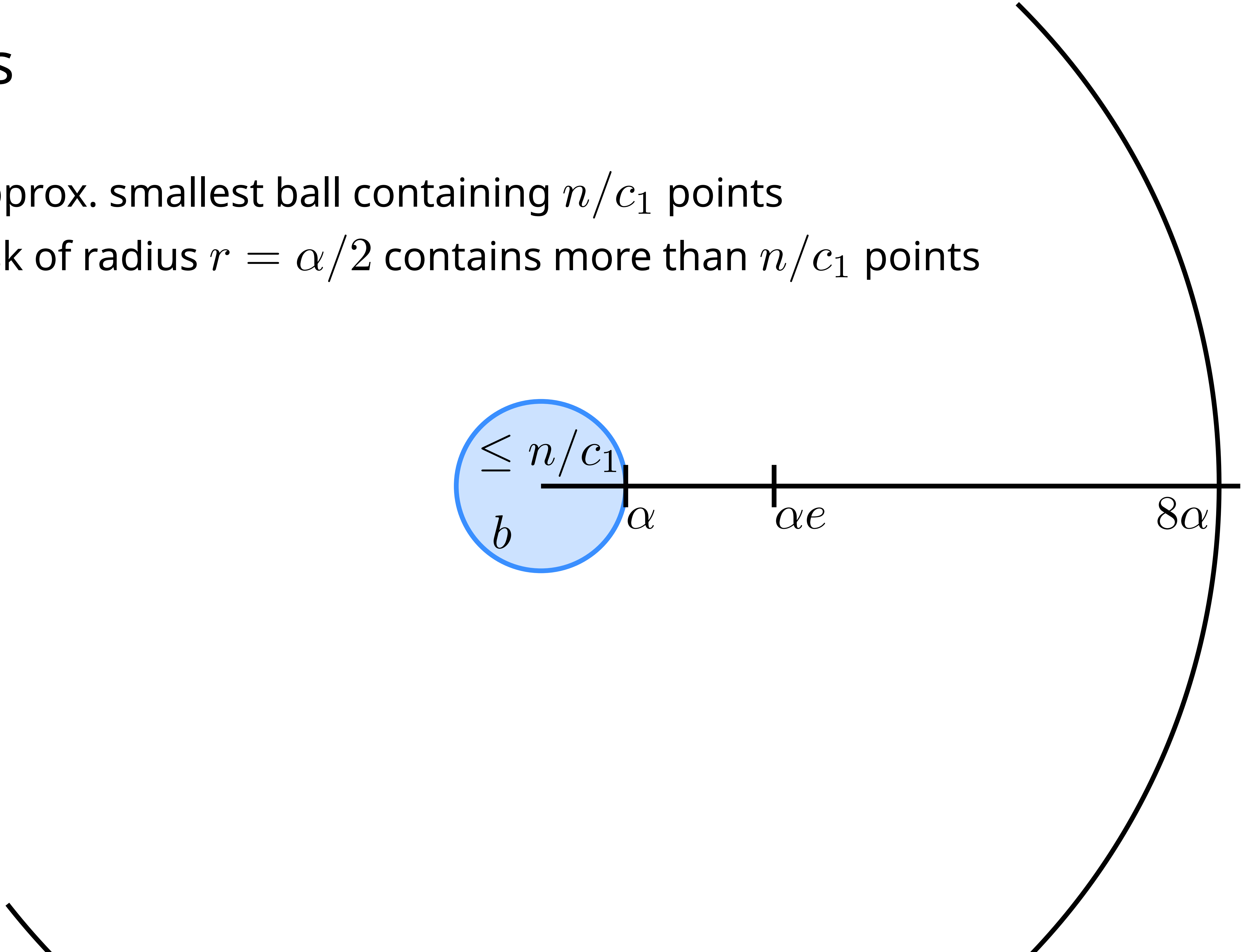
$b = b(p, \alpha)$ 2-approx. smallest ball containing n/c_1 points



Correctness

$b = b(p, \alpha)$ 2-approx. smallest ball containing n/c_1 points

$\Rightarrow$ no disk of radius $r = \alpha/2$ contains more than n/c_1 points

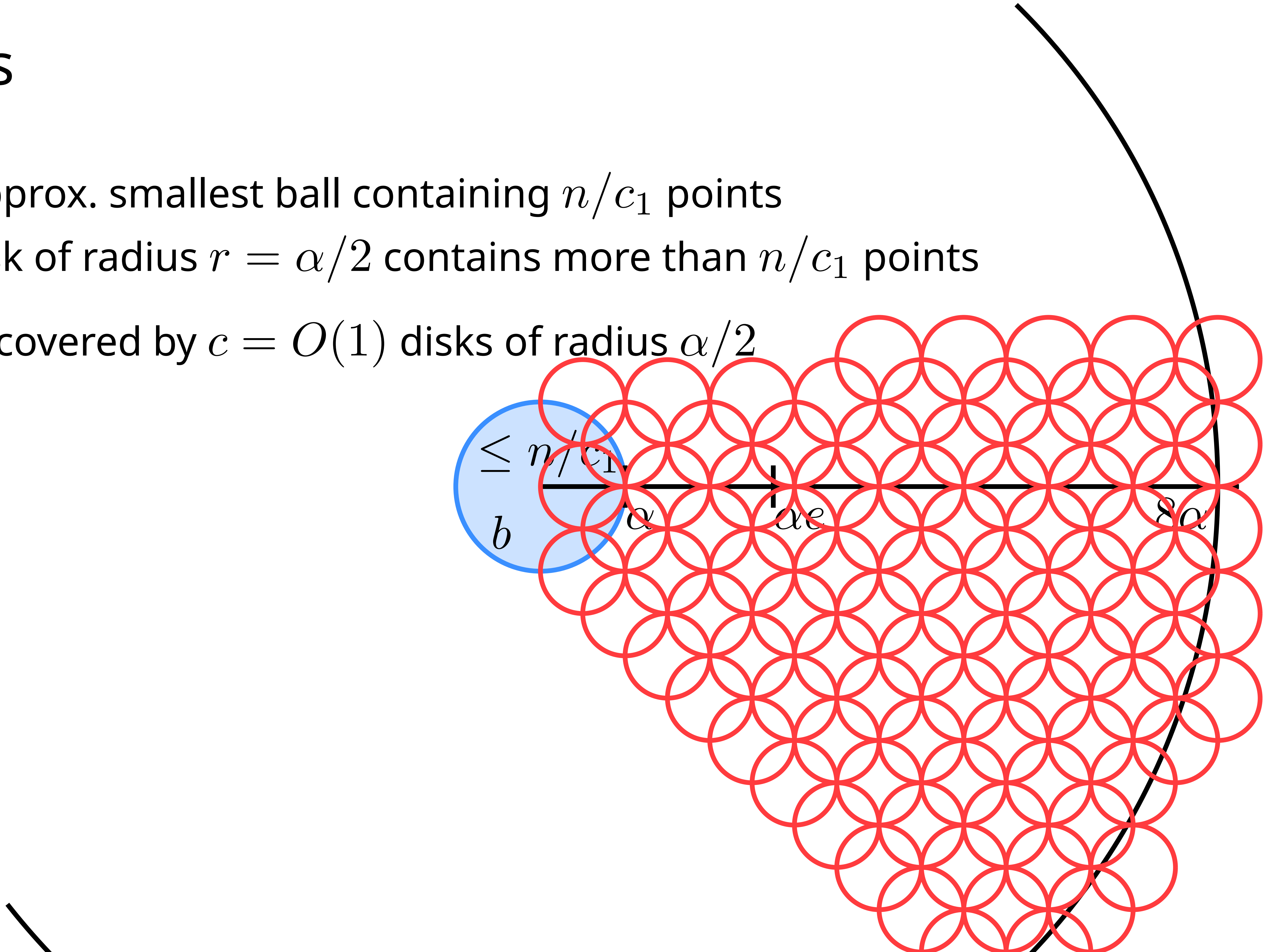


Correctness

$b = b(p, \alpha)$ 2-approx. smallest ball containing n/c_1 points

$\Rightarrow$ no disk of radius $r = \alpha/2$ contains more than n/c_1 points

$b(p, 8\alpha)$ can be covered by $c = O(1)$ disks of radius $\alpha/2$



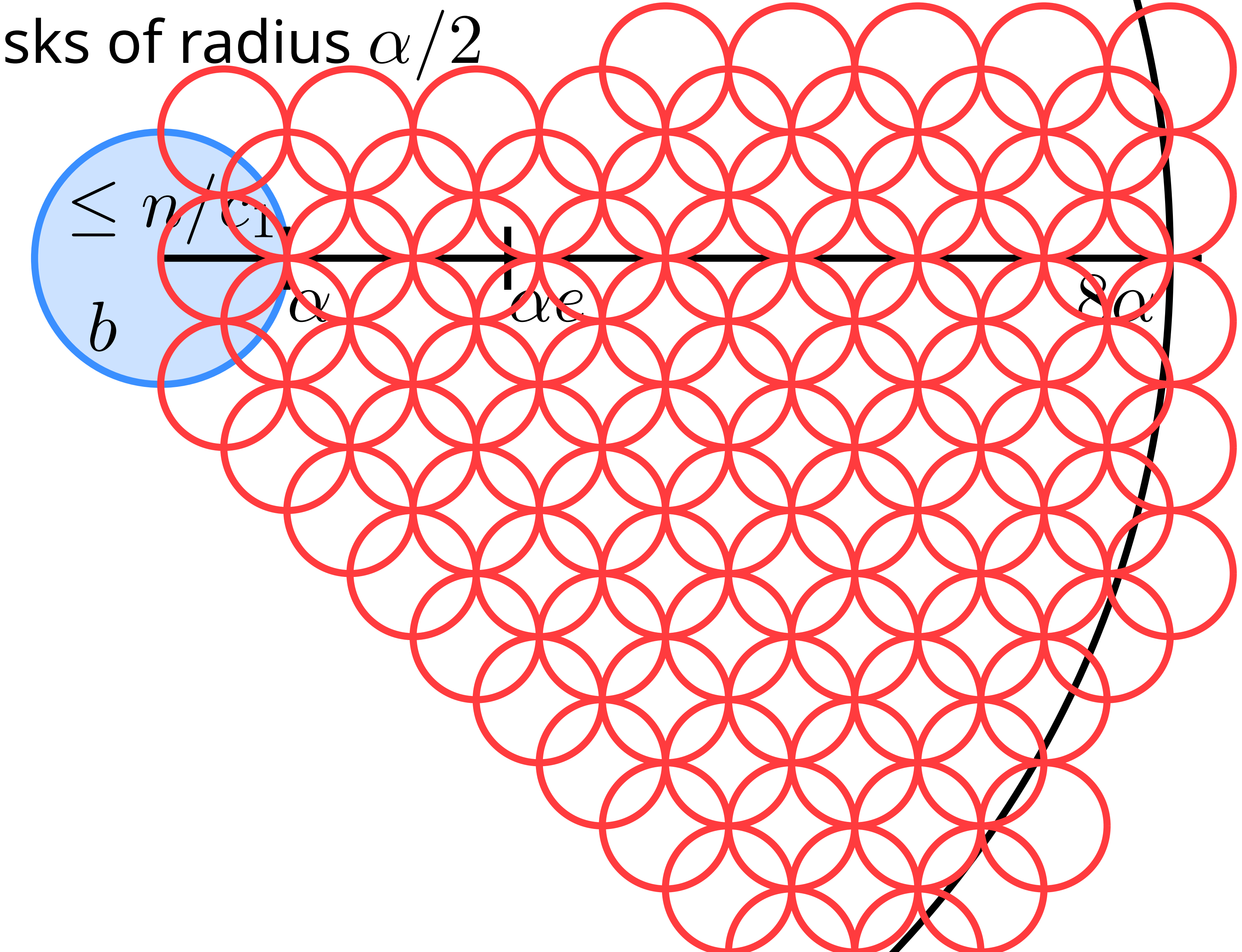
Correctness

$b = b(p, \alpha)$ 2-approx. smallest ball containing n/c_1 points

$\Rightarrow$ no disk of radius $r = \alpha/2$ contains more than n/c_1 points

$b(p, 8\alpha)$ can be covered by $c = O(1)$ disks of radius $\alpha/2$

Choose: $c_1 := 3c$



Correctness

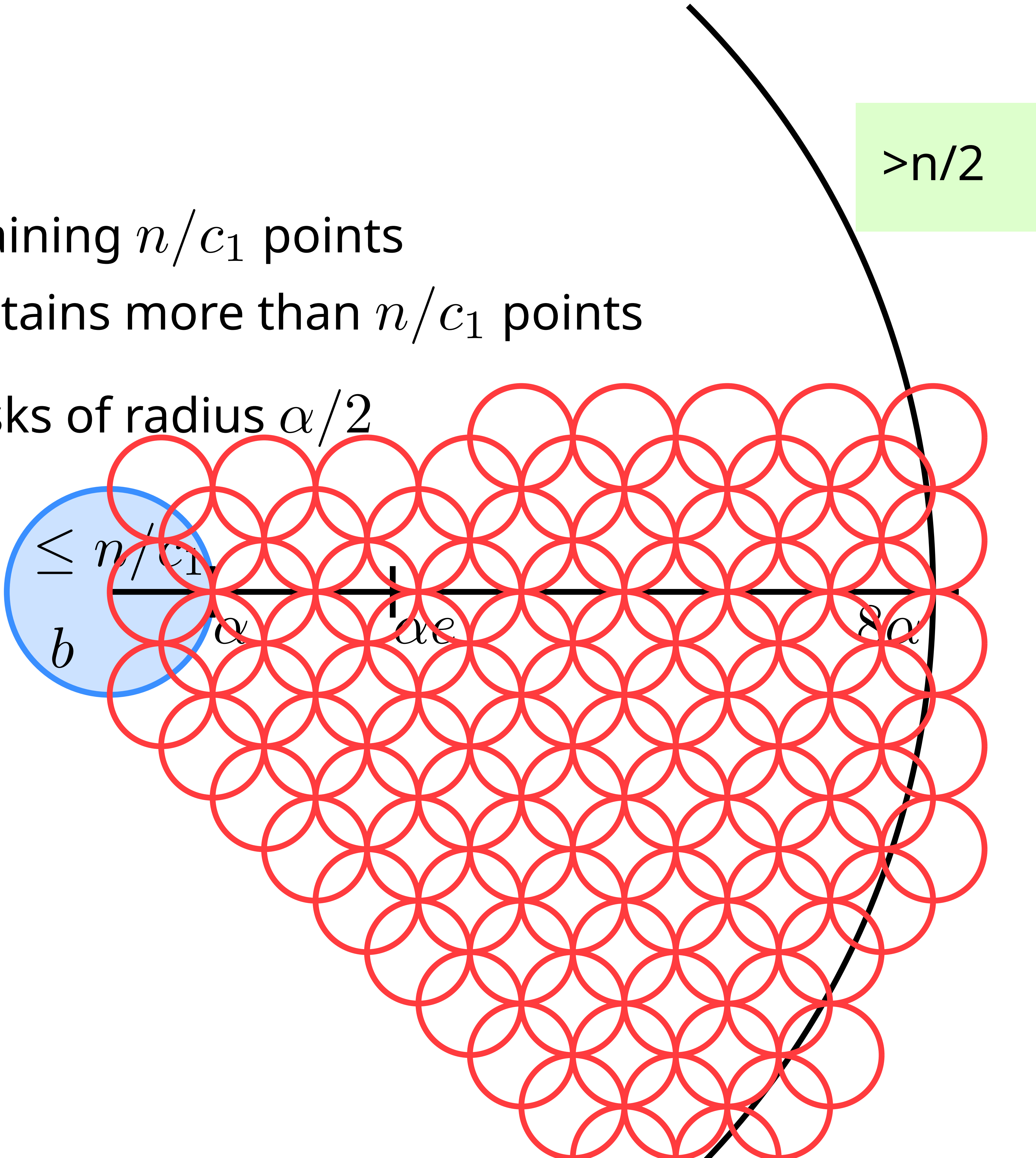
$b = b(p, \alpha)$ 2-approx. smallest ball containing n/c_1 points

$\Rightarrow$ no disk of radius $r = \alpha/2$ contains more than n/c_1 points

$b(p, 8\alpha)$ can be covered by $c = O(1)$ disks of radius $\alpha/2$

Choose: $c_1 := 3c$

$b(p, 8\alpha)$ contains $\leq c\frac{n}{3c} < n/2$ points



Correctness

$b = b(p, \alpha)$ 2-approx. smallest ball containing n/c_1 points

$\Rightarrow$ no disk of radius $r = \alpha/2$ contains more than n/c_1 points

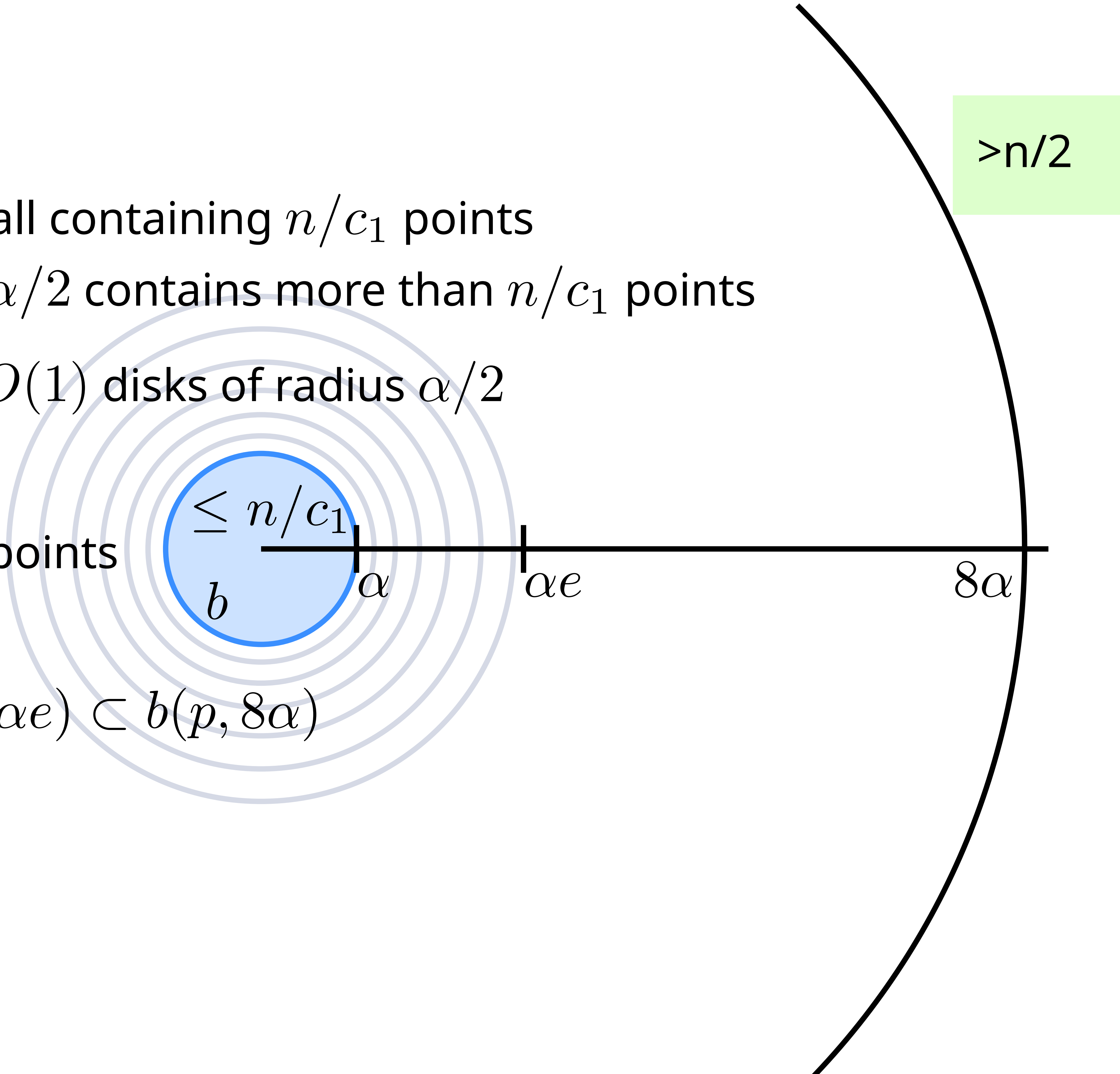
$b(p, 8\alpha)$ can be covered by $c = O(1)$ disks of radius $\alpha/2$

Choose: $c_1 := 3c$

$b(p, 8\alpha)$ contains $\leq c\frac{n}{3c} < n/2$ points

$r_i := b_i \setminus b_{i-1}$, are n ranges in

$b_n = b(p, \alpha(1 + 1/n)^n) \subset b(p, \alpha e) \subset b(p, 8\alpha)$



Correctness

$b = b(p, \alpha)$ 2-approx. smallest ball containing n/c_1 points

$\Rightarrow$ no disk of radius $r = \alpha/2$ contains more than n/c_1 points

$b(p, 8\alpha)$ can be covered by $c = O(1)$ disks of radius $\alpha/2$

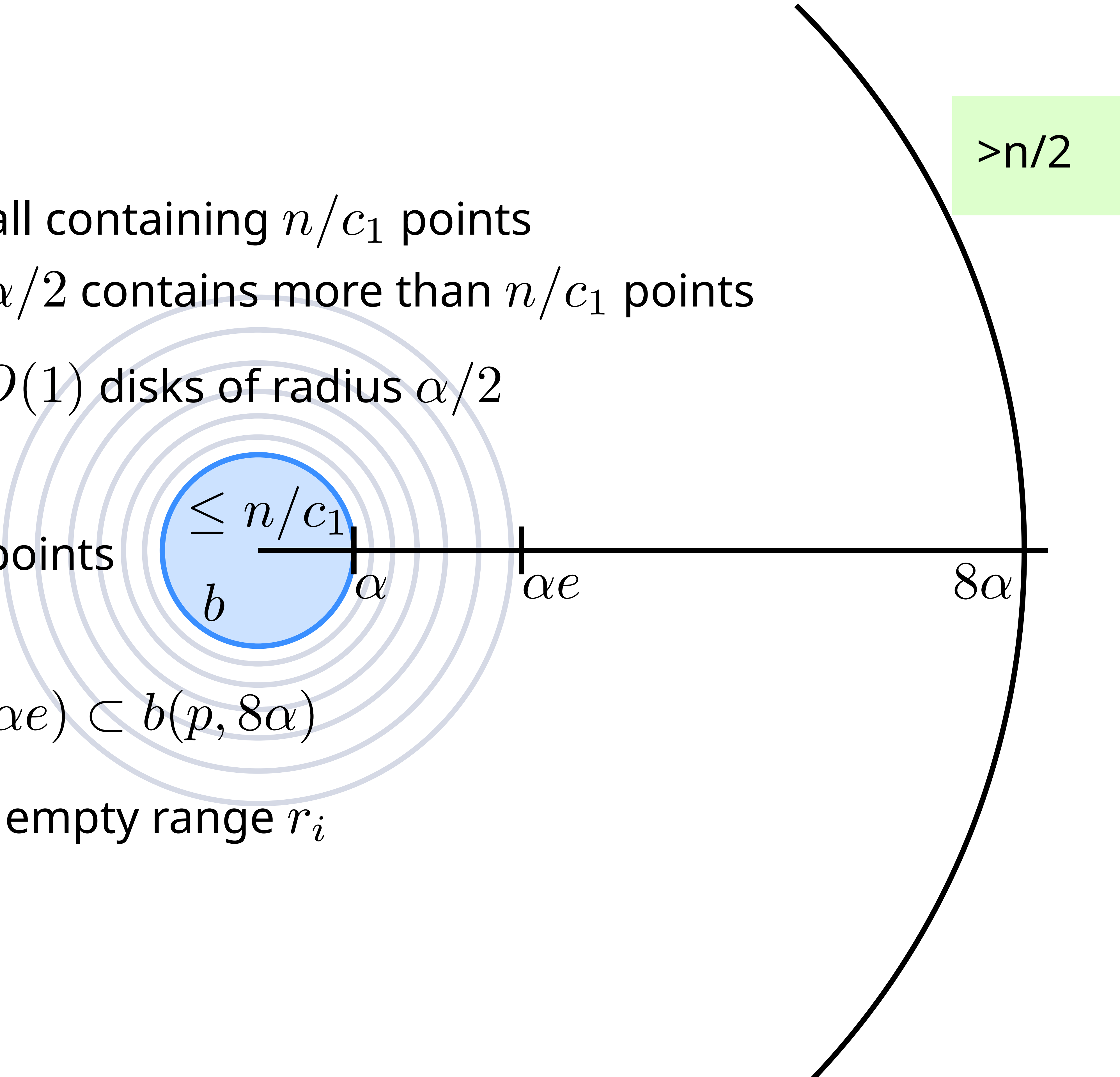
Choose: $c_1 := 3c$

$b(p, 8\alpha)$ contains $\leq c\frac{n}{3c} < n/2$ points

$r_i := b_i \setminus b_{i-1}$, are n ranges in

$b_n = b(p, \alpha(1+1/n)^n) \subset b(p, \alpha e) \subset b(p, 8\alpha)$

pigeonhole principle: There is an empty range r_i



Correctness

$b = b(p, \alpha)$ 2-approx. smallest ball containing n/c_1 points

$\Rightarrow$ no disk of radius $r = \alpha/2$ contains more than n/c_1 points

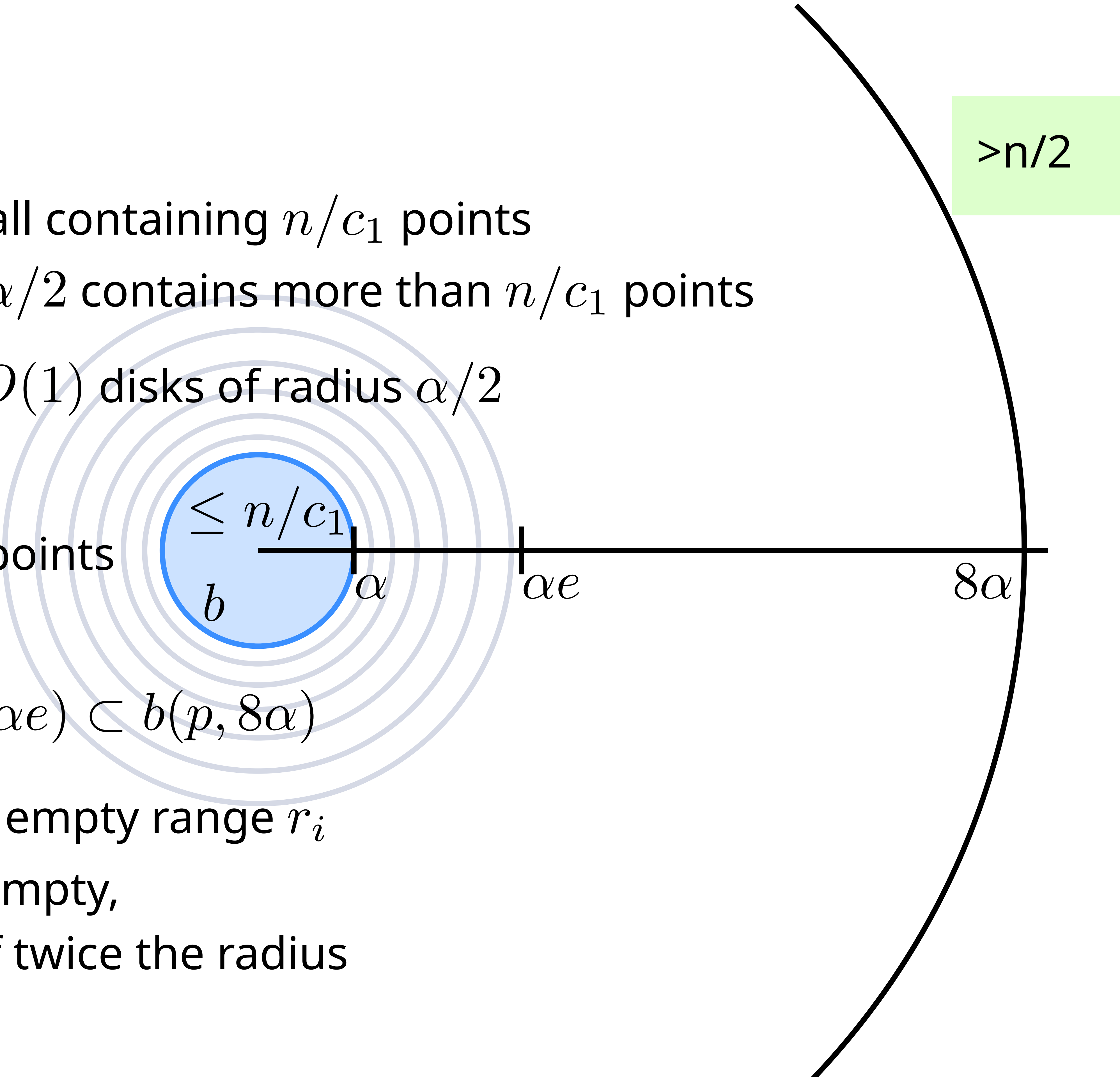
$b(p, 8\alpha)$ can be covered by $c = O(1)$ disks of radius $\alpha/2$

Choose: $c_1 := 3c$

$b(p, 8\alpha)$ contains $\leq c\frac{n}{3c} < n/2$ points

$r_i := b_i \setminus b_{i-1}$, are n ranges in

$b_n = b(p, \alpha(1 + 1/n)^n) \subset b(p, \alpha e) \subset b(p, 8\alpha)$



pigeonhole principle: There is an empty range r_i

b_{i-1} contains n/c_1 points, r_i empty,

$> n/2$ points outside of ball of twice the radius

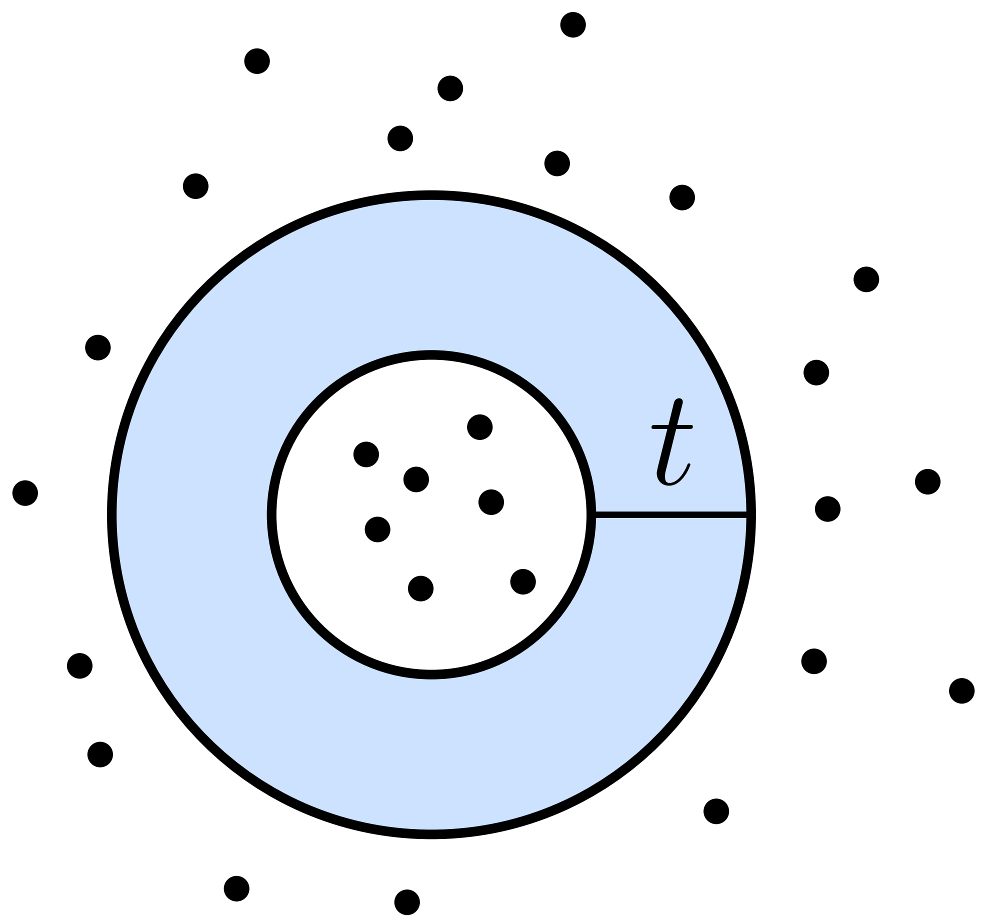
Ring Separator tree

A binary tree T having the points of P as leaves is a t-ring tree for P iff:

Ring Separator tree

A binary tree T having the points of P as leaves is a t-ring tree for P iff:

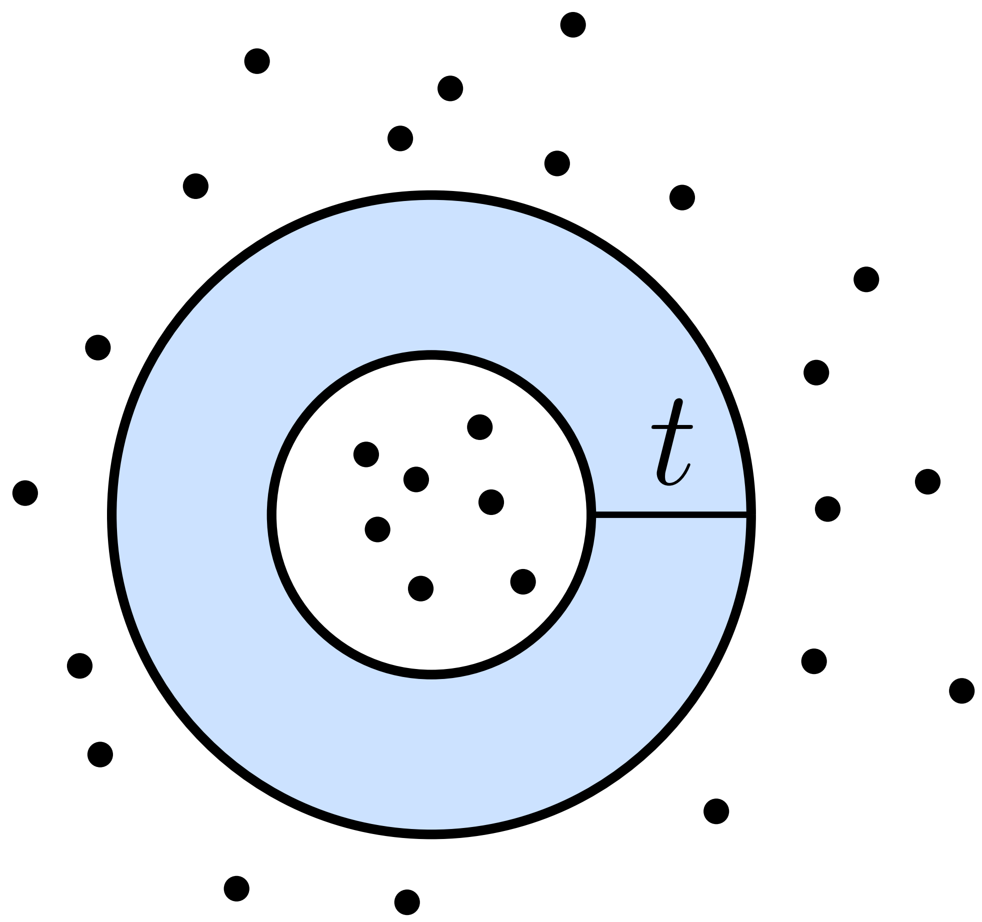
- Every node $v \in T$ with corresponding subset $P_v \subset P$ is associated with a ‘ring’ that separates the points of P_v into two sets



Ring Separator tree

A binary tree T having the points of P as leaves is a t-ring tree for P iff:

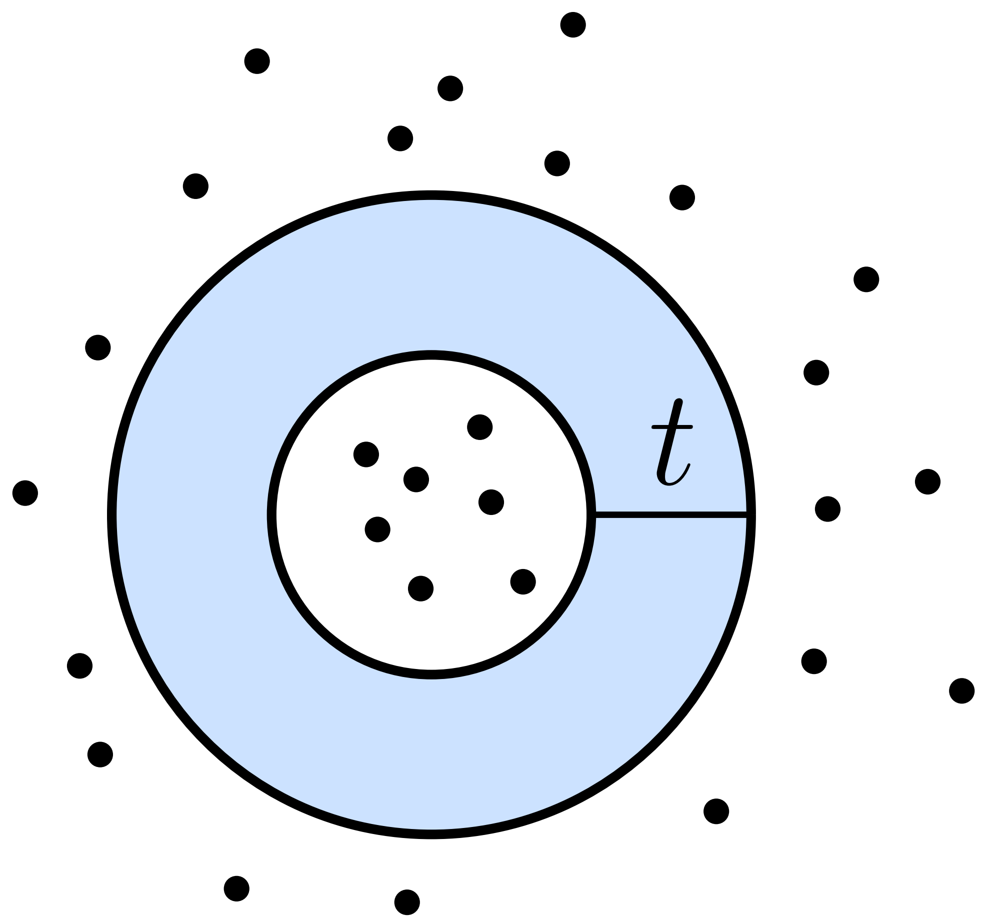
- Every node $v \in T$ with corresponding subset $P_v \subset P$ is associated with a 'ring' that separates the points of P_v into two sets
- The interior of the ring has no points inside it



Ring Separator tree

A binary tree T having the points of P as leaves is a t-ring tree for P iff:

- Every node $v \in T$ with corresponding subset $P_v \subset P$ is associated with a 'ring' that separates the points of P_v into two sets
- The interior of the ring has no points inside it
- The interior of the ring is of width t

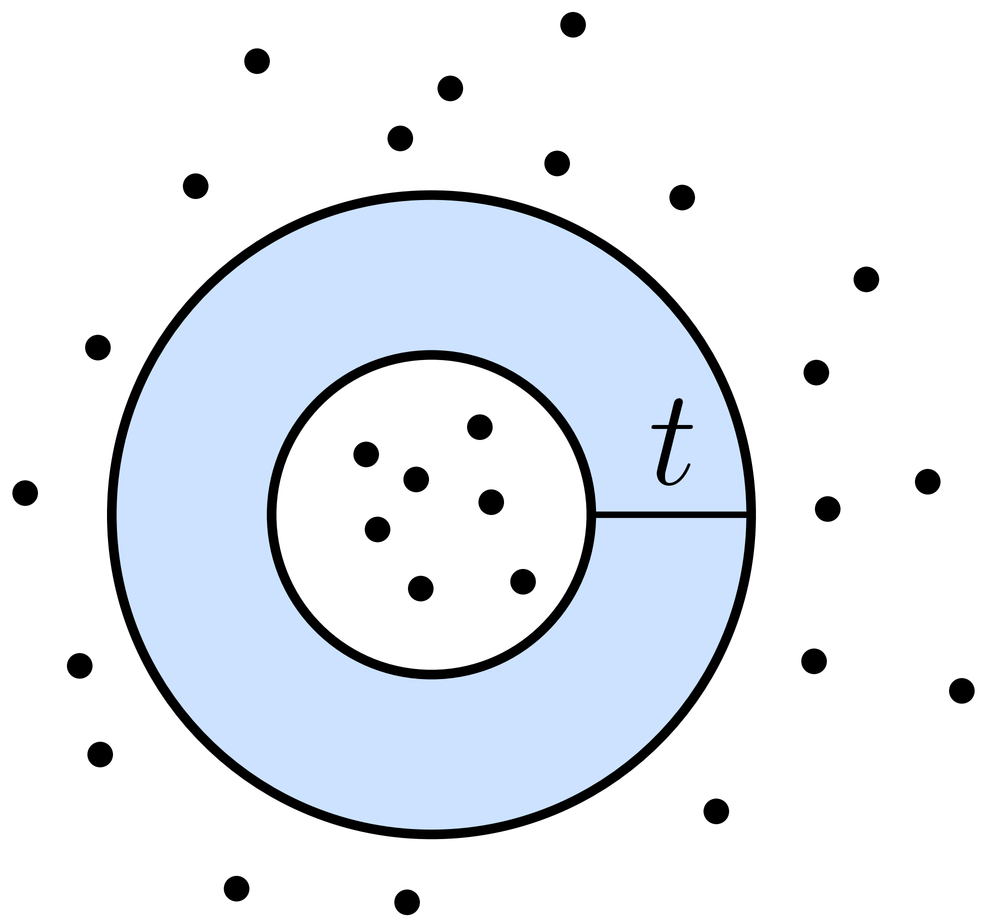


Ring Separator tree

A binary tree T having the points of P as leaves is a t-ring tree for P iff:

- Every node $v \in T$ with corresponding subset $P_v \subset P$ is associated with a 'ring' that separates the points of P_v into two sets
- The interior of the ring has no points inside it
- The interior of the ring is of width t

Apply algorithm for ring separator recursively



Ring Separator tree

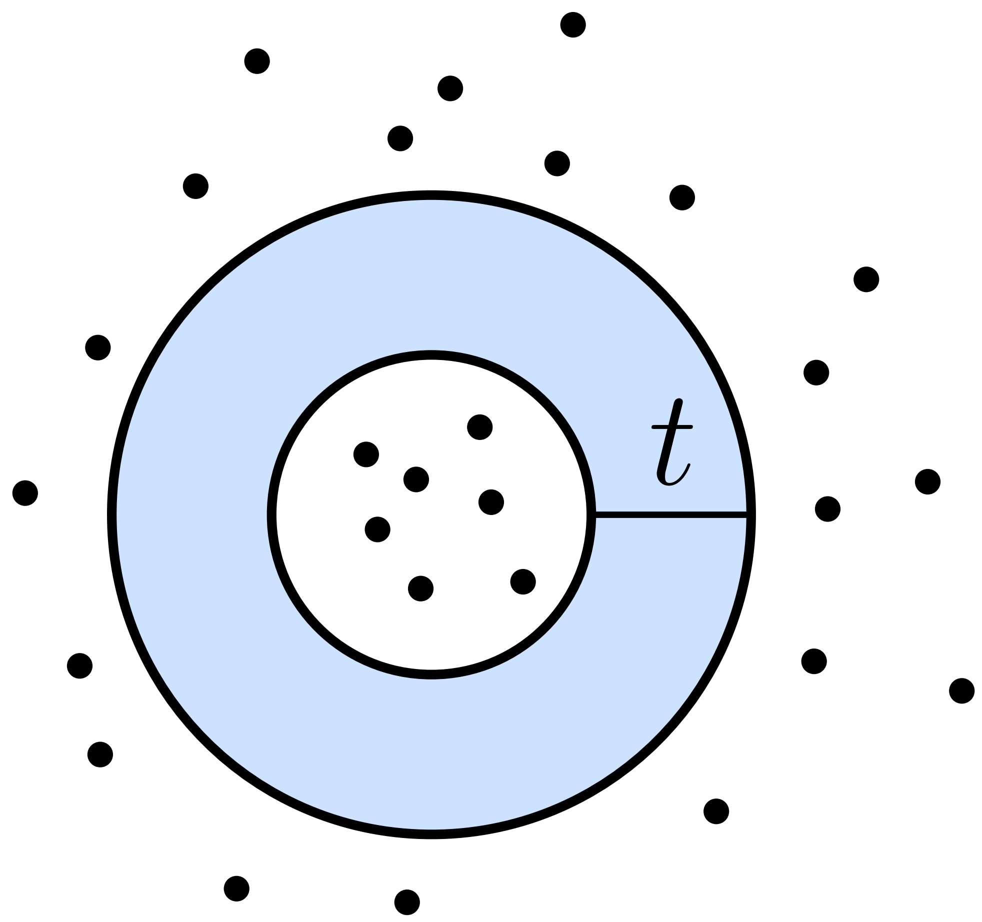
A binary tree T having the points of P as leaves is a t-ring tree for P iff:

- Every node $v \in T$ with corresponding subset $P_v \subset P$ is associated with a 'ring' that separates the points of P_v into two sets
- The interior of the ring has no points inside it
- The interior of the ring is of width t

Apply algorithm for ring separator recursively

Result: A $\frac{1}{n}$-ring separator tree

A n-semi-separated pair decomposition of weight $\Theta(n \log n)$



Overview

Semi-separated pair decomposition (SSPD) ✓

Ring separator tree: n-semi-separated pair decomposition ✓

Ring separator: $O(n)$-ANN

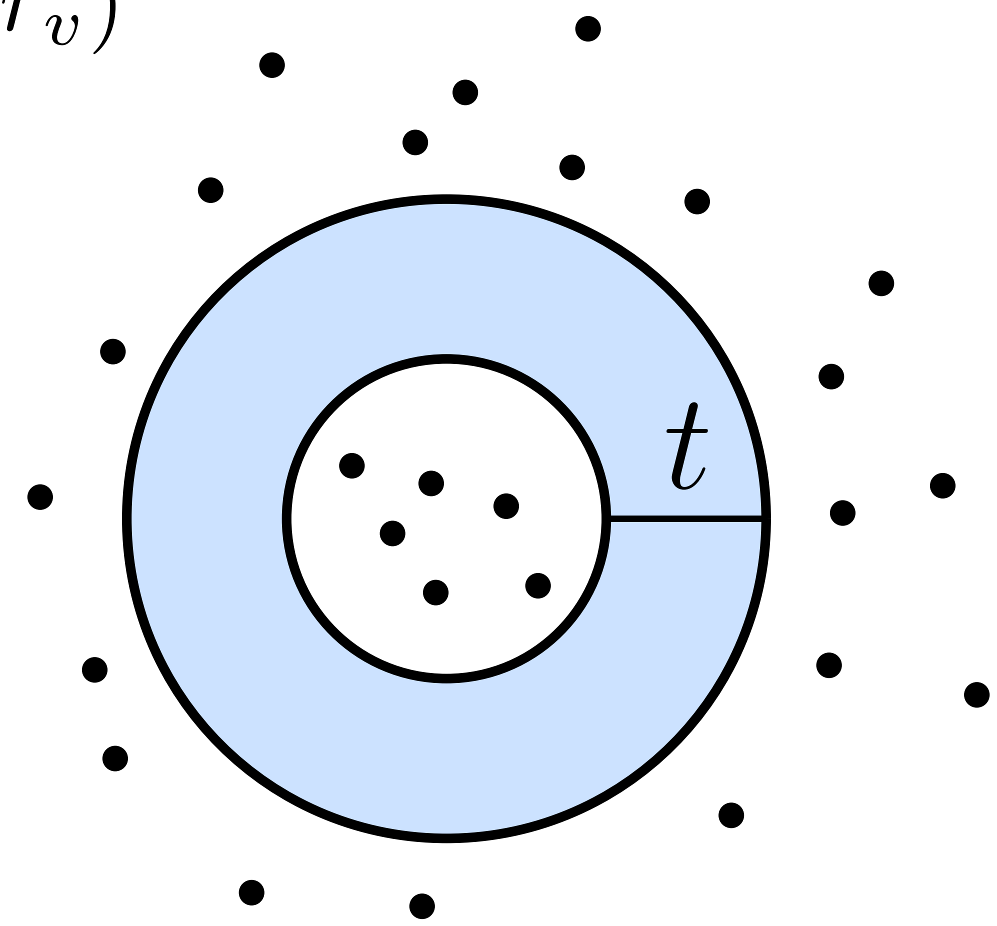
$(1/\varepsilon)$-semi-separated pair decomposition

Ring Separator Tree

For every node v we ensure:

There is a ball $b_v = b(c_v, r_v)$ s.t. all points of $P_v^{\text{in}} = P_v \cap b_v$ are in one child of v (the inner child)

All other points of P_v are outside $b(c_v, (1+t)r_v)$ and stored in the other child (the outer child)



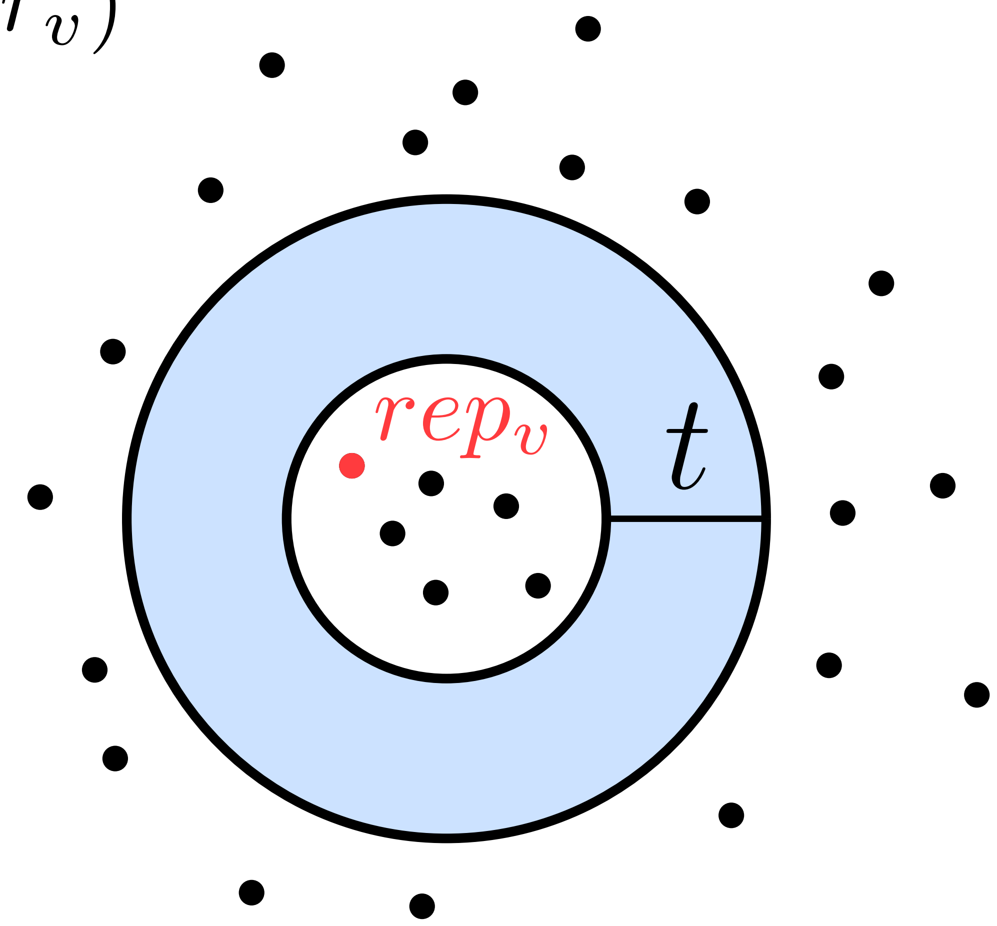
Ring Separator Tree

For every node v we ensure:

There is a ball $b_v = b(c_v, r_v)$ s.t. all points of $P_v^{\text{in}} = P_v \cap b_v$ are in one child of v (the inner child)

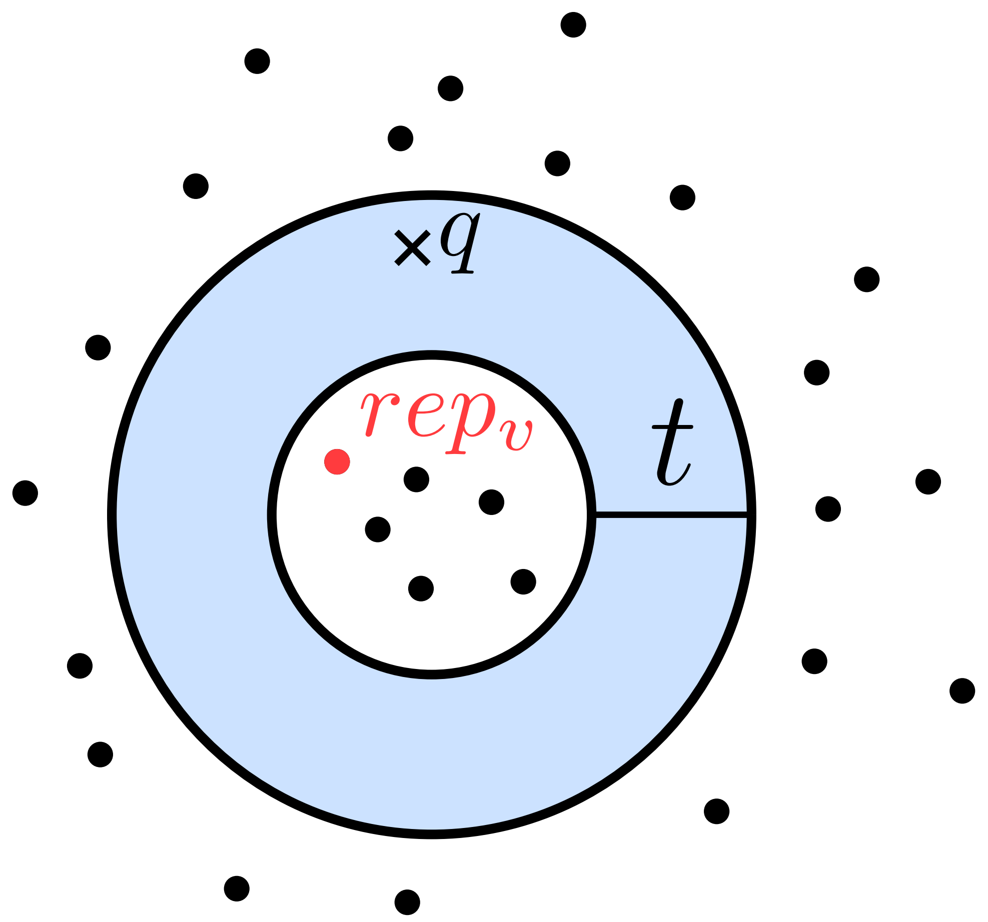
All other points of P_v are outside $b(c_v, (1+t)r_v)$ and stored in the other child (the outer child)

Store an arbitrary $rep_v \in P_v^{\text{in}}$ in v



ANN search procedure

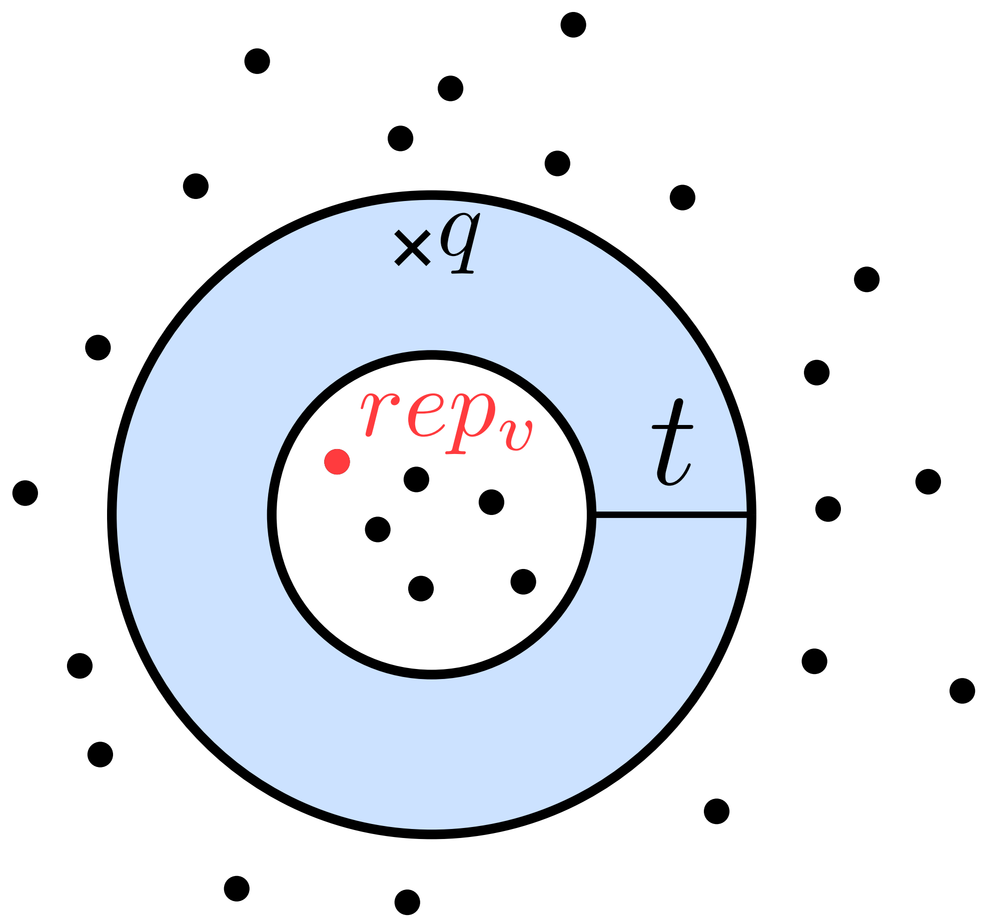
Given query point q:



ANN search procedure

Given query point q:

$v =$ root of T, $r = \infty$



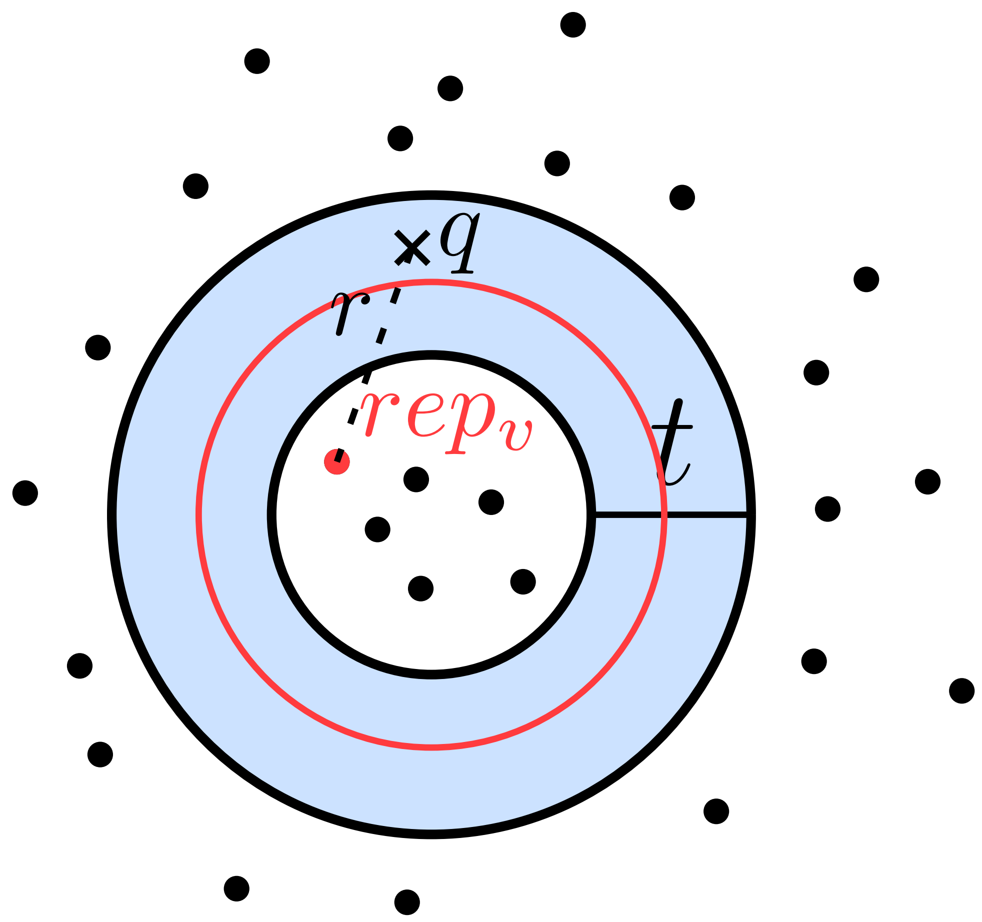
ANN search procedure

Given query point q:

$v =$ root of T, $r = \infty$

while v is not a leaf:

$r = \min(r, \|q - rep_v\|)$



ANN search procedure

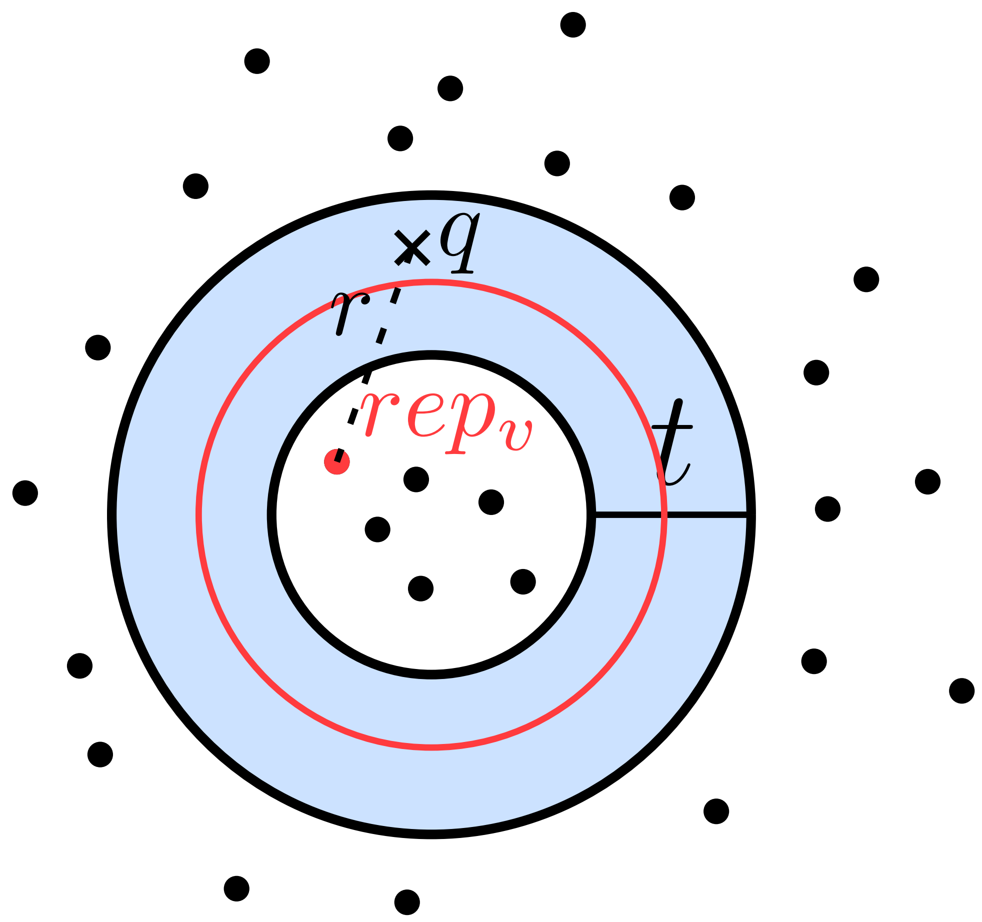
Given query point q:

v = root of T, $r = \infty$

while v is not a leaf:

$r = \min(r, \|q - rep_v\|)$

$r_{mid} = (1 + t/2) r_v$



ANN search procedure

Given query point q:

$v =$ root of T, $r = \infty$

while v is not a leaf:

$r = \min(r, \|q - rep_v\|)$

$r_{mid} = (1 + t/2) r_v$

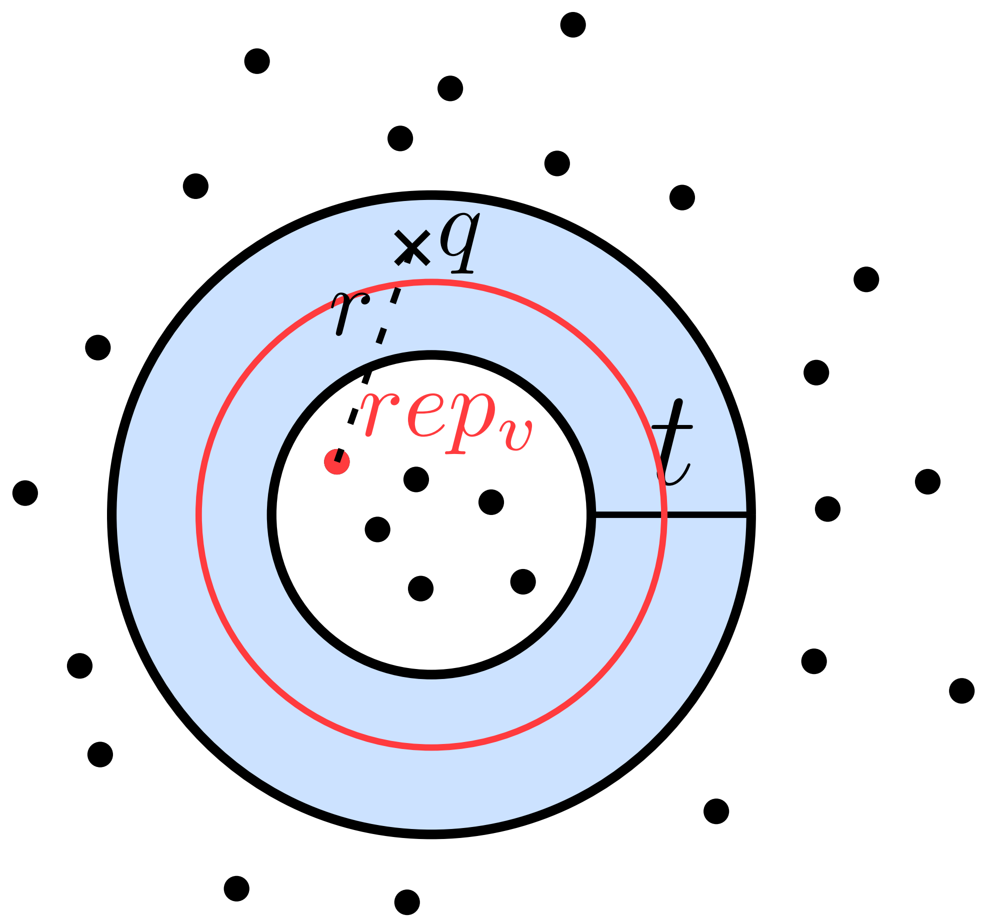
if $\|q - c_v\| \leq r_{mid}$ then

$v =$ inner child of v

else

$v =$ outer child of v

return r



ANN search procedure

Given query point q:

$v =$ root of T, $r = \infty$

while v is not a leaf:

$r = \min(r, \|q - rep_v\|)$

$r_{mid} = (1 + t/2)r_v$

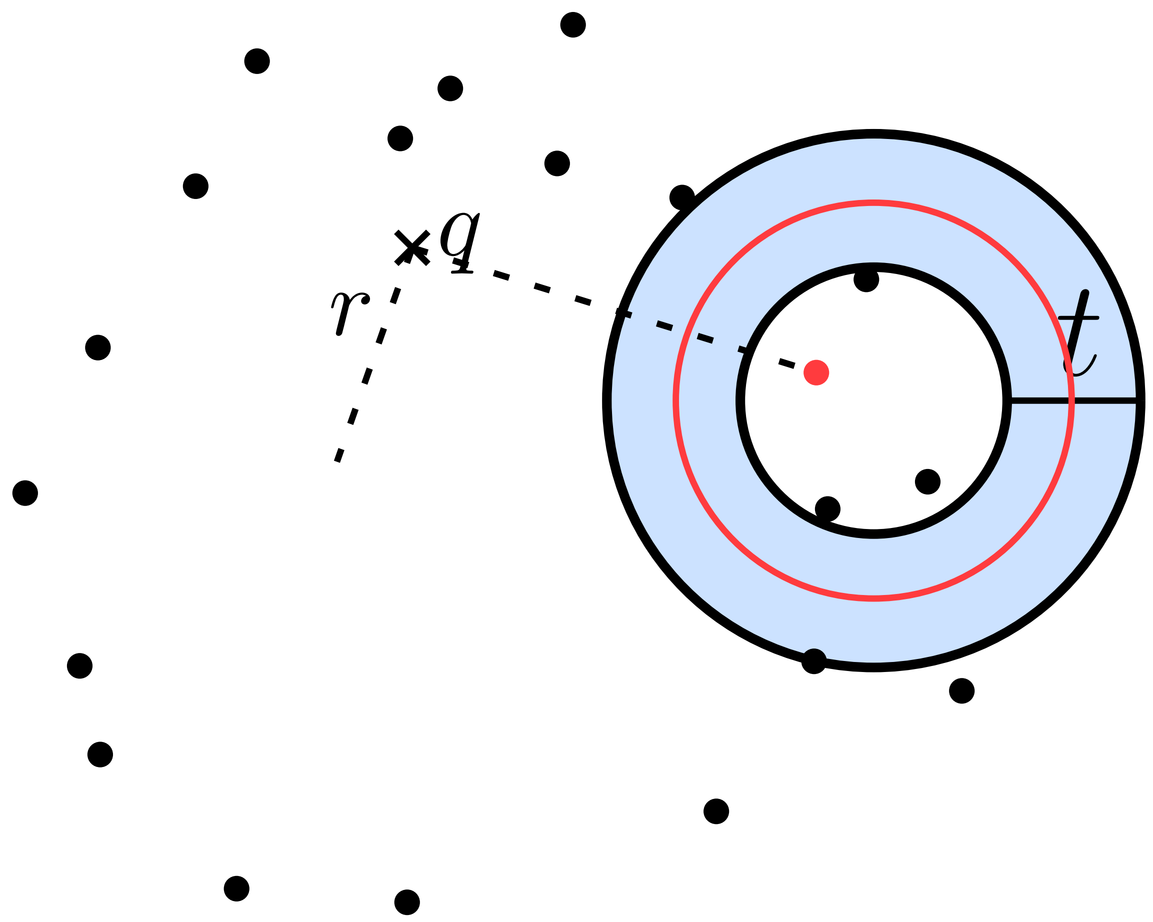
if $\|q - c_v\| \leq r_{mid}$ then

$v =$ inner child of v

else

$v =$ outer child of v

return r



ANN search procedure

Given query point q:

$v =$ root of T, $r = \infty$

while v is not a leaf:

$r = \min(r, \|q - rep_v\|)$

$r_{mid} = (1 + t/2)r_v$

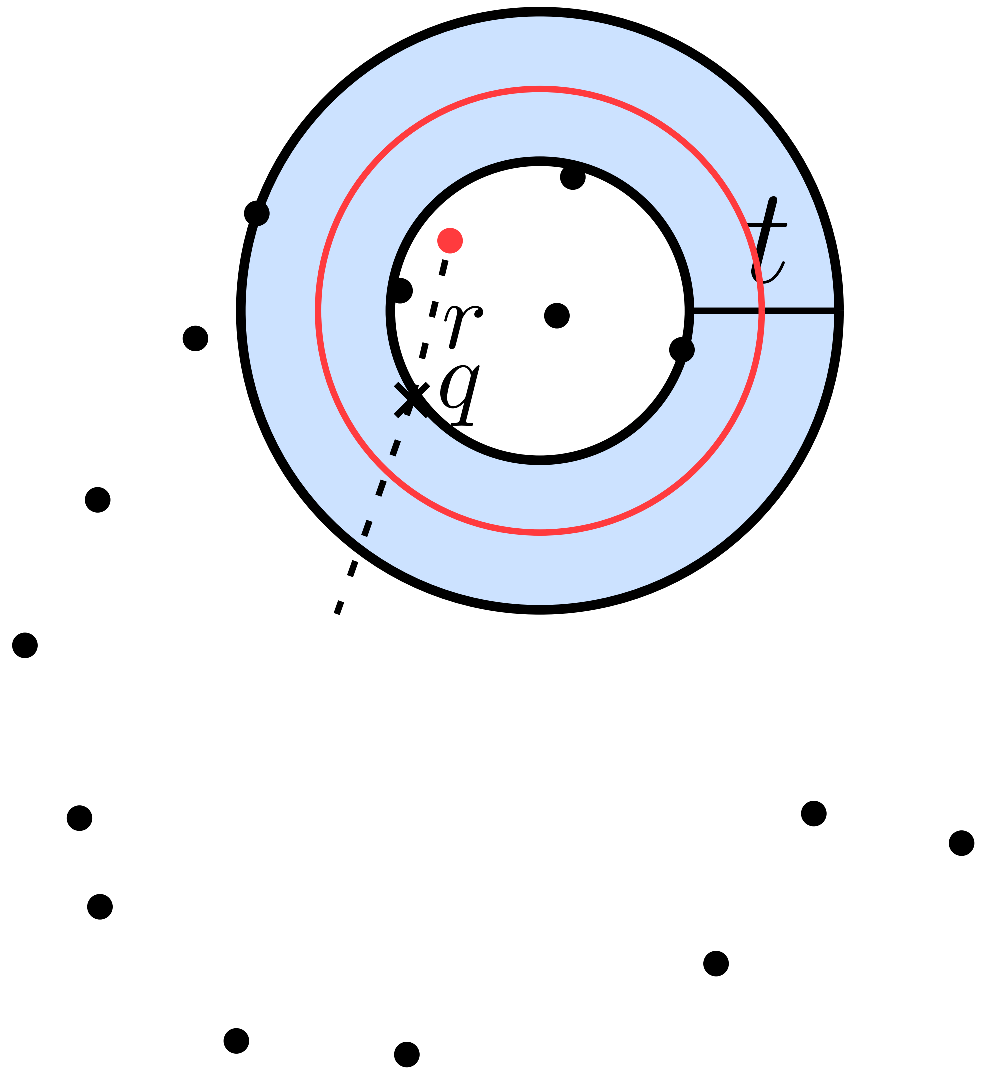
if $\|q - c_v\| \leq r_{mid}$ then

$v =$ inner child of v

else

$v =$ outer child of v

return r



ANN search procedure

Given query point q:

$v =$ root of T, $r = \infty$

while v is not a leaf:

$r = \min(r, \|q - rep_v\|)$

$r_{mid} = (1 + t/2) r_v$

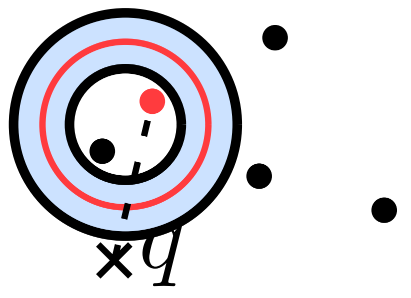
if $\|q - c_v\| \leq r_{mid}$ then

$v =$ inner child of v

else

$v =$ outer child of v

return r



ANN search procedure

Given query point q:

$v =$ root of T, $r = \infty$

while v is not a leaf:

$r = \min(r, \|q - rep_v\|)$

$r_{mid} = (1 + t/2) r_v$

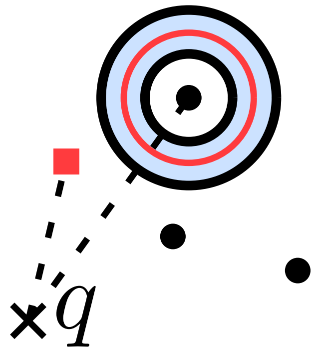
if $\|q - c_v\| \leq r_{mid}$ then

$v =$ inner child of v

else

$v =$ outer child of v

return r



ANN search procedure

Given query point q:

$v =$ root of T, $r = \infty$

while v is not a leaf:

$r = \min(r, \|q - rep_v\|)$

$r_{mid} = (1 + t/2)r_v$

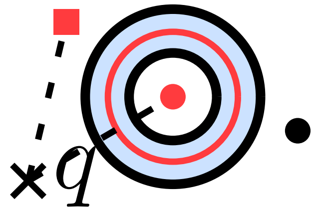
if $\|q - c_v\| \leq r_{mid}$ then

$v =$ inner child of v

else

$v =$ outer child of v

return r



ANN search procedure

Given query point q:

$v =$ root of T, $r = \infty$

while v is not a leaf:

$r = \min(r, \|q - rep_v\|)$

$r_{mid} = (1 + t/2)r_v$

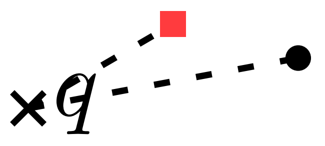
if $\|q - c_v\| \leq r_{mid}$ then

$v =$ inner child of v

else

$v =$ outer child of v

return r



ANN search procedure

Given query point q:

$v =$ root of T, $r = \infty$

while v is not a leaf:

$r = \min(r, \|q - rep_v\|)$

$r_{mid} = (1 + t/2) r_v$

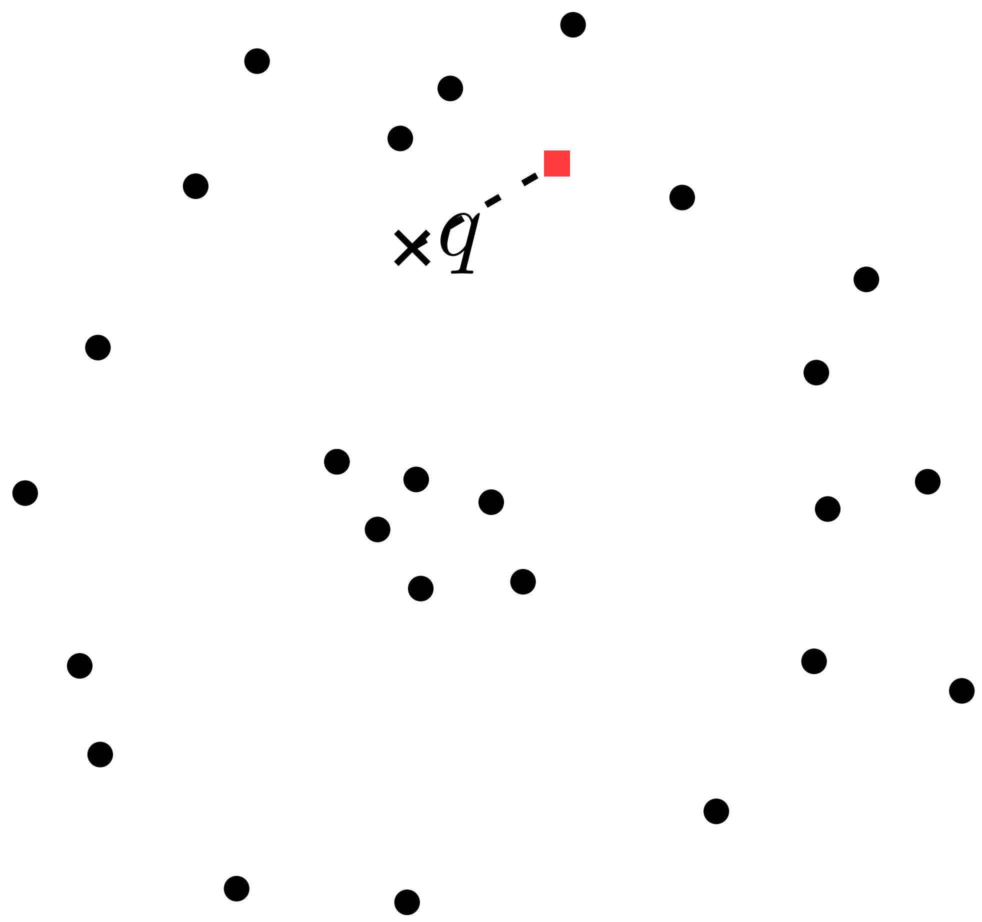
if $\|q - c_v\| \leq r_{mid}$ then

$v =$ inner child of v

else

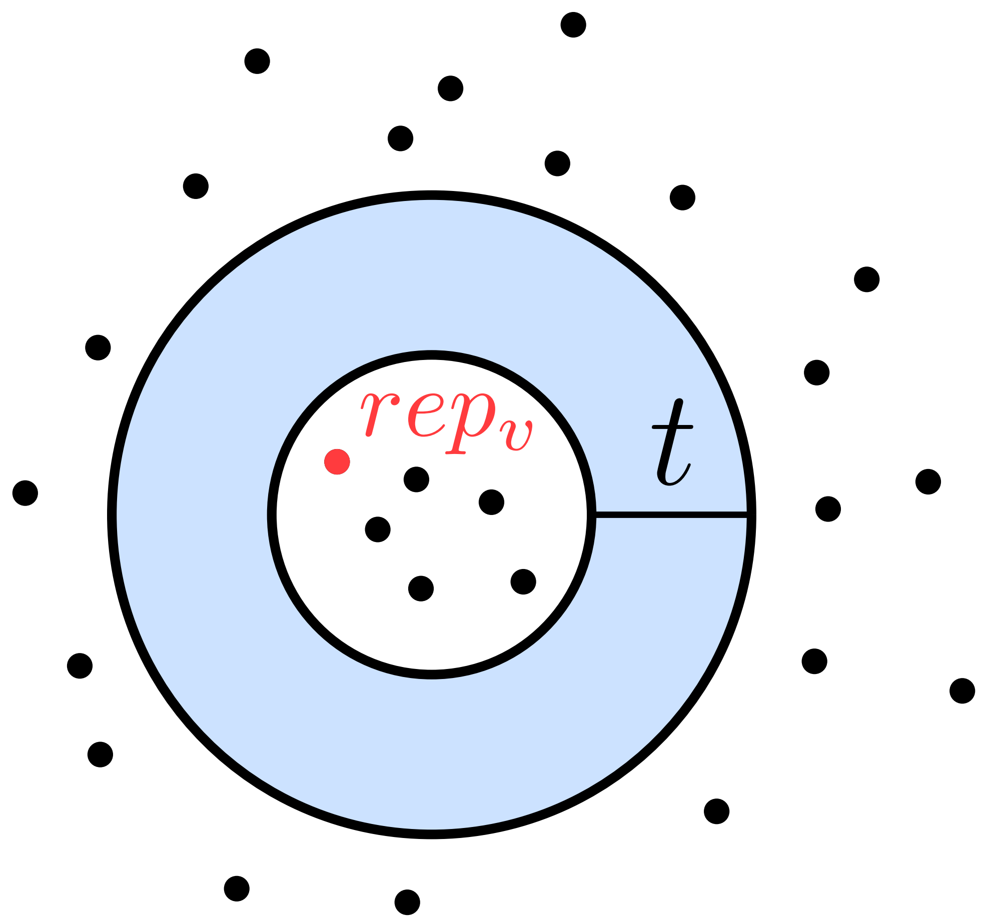
$v =$ outer child of v

return r



Intuition of correctness

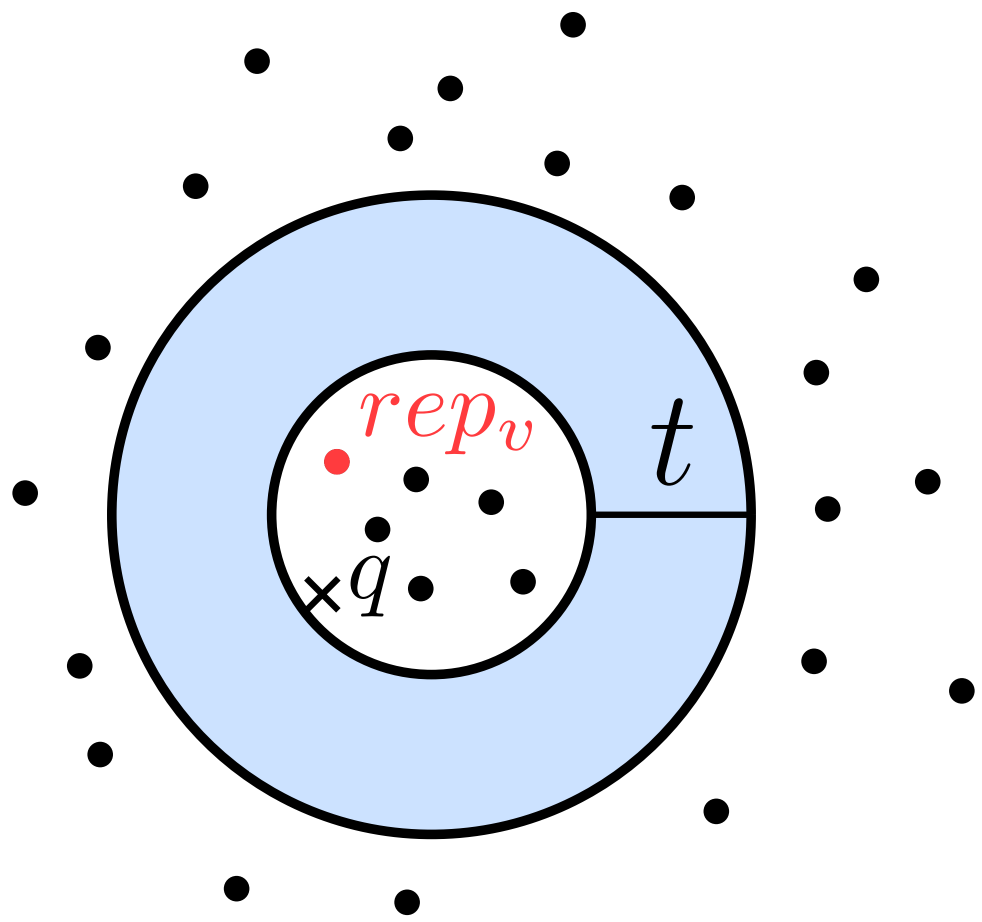
Case distinction:



Intuition of correctness

Case distinction:

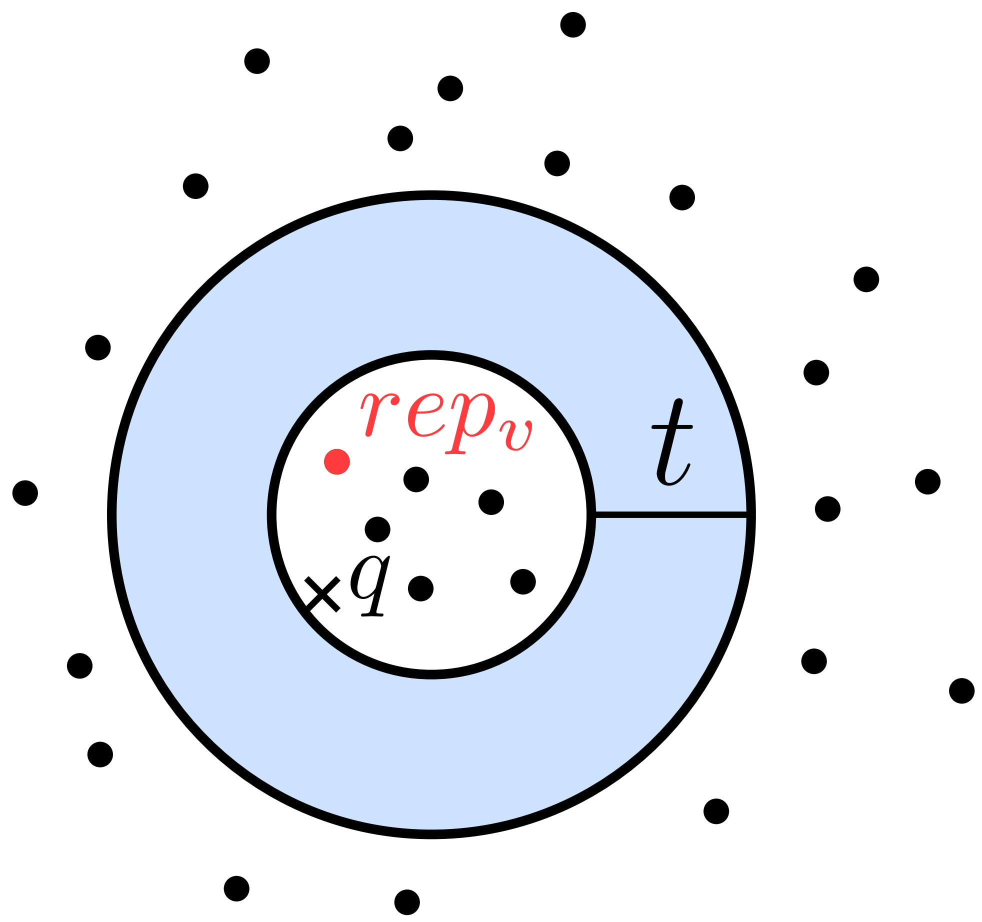
1.) q in b_v



Intuition of correctness

Case distinction:

1.) q in b_v $\rightarrow$ points outside of b_v too far away (to invalidate rep_v as $1/t$-ANN)

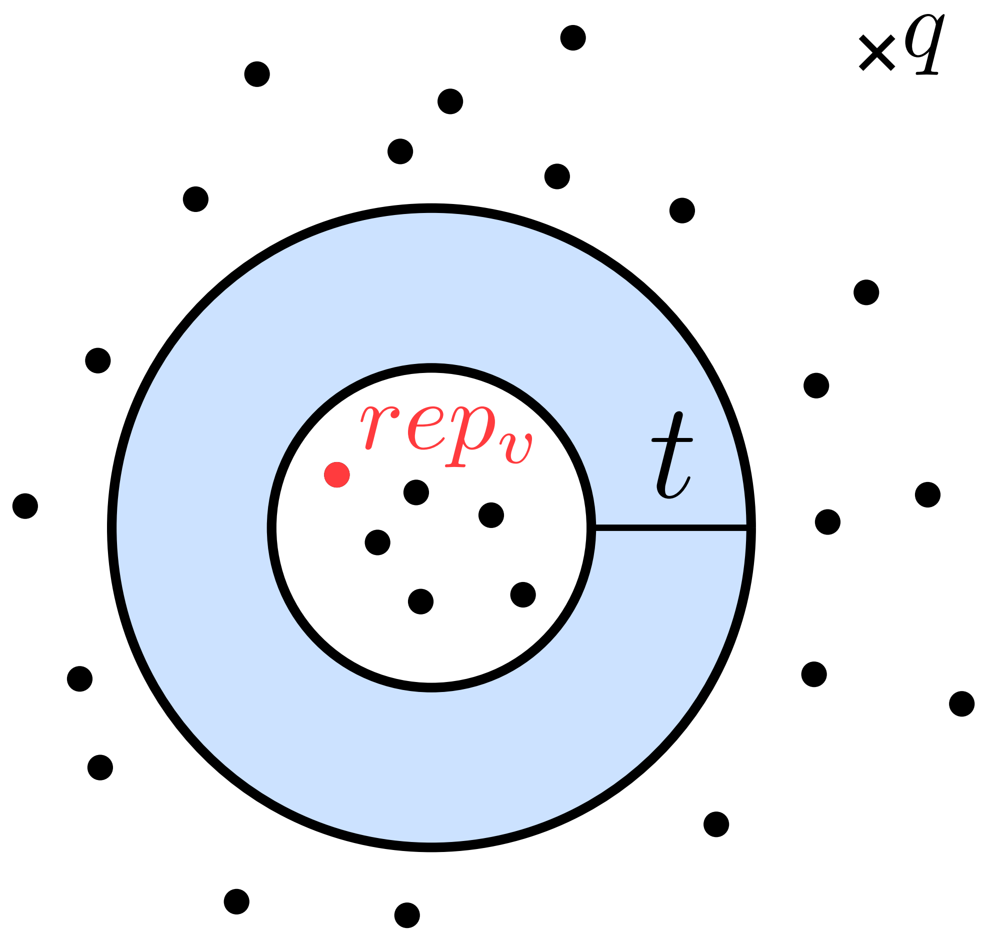


Intuition of correctness

Case distinction:

1.) q in b_v $\rightarrow$ points outside of b_v too far away (to invalidate rep_v as $1/t$-ANN)

2.) q outside enlarged b_v



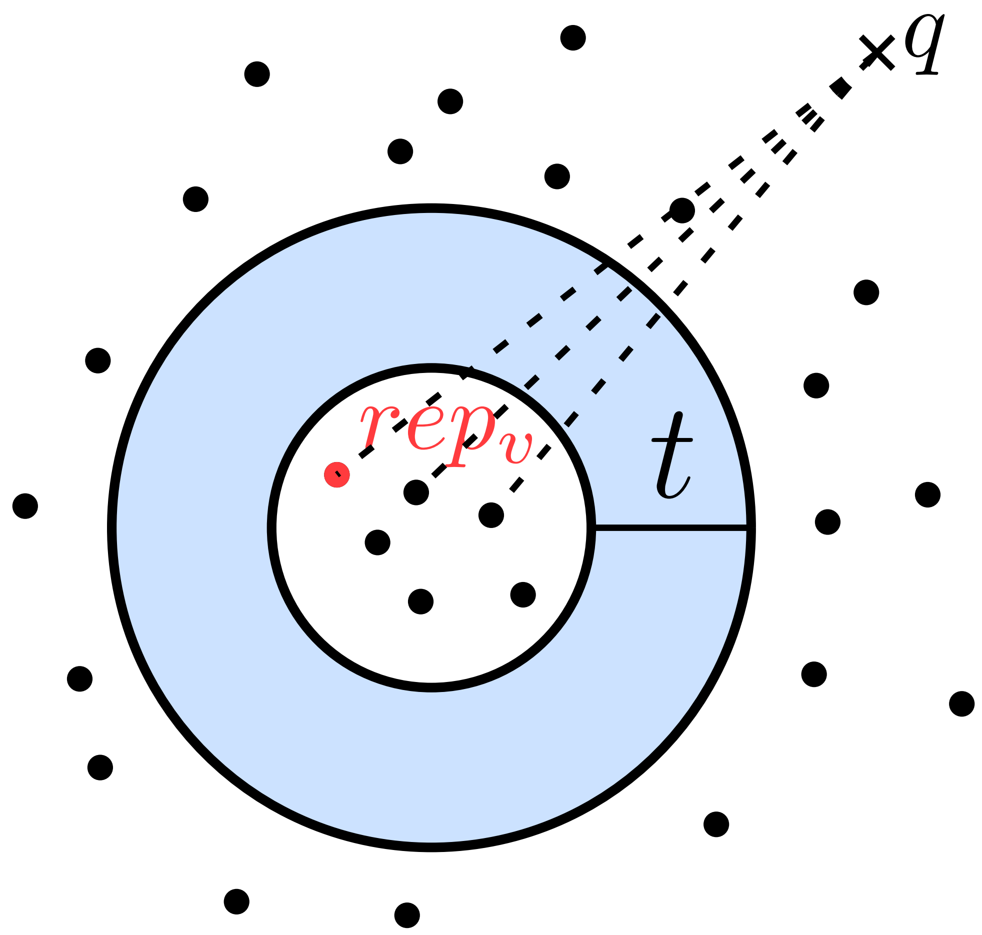
Intuition of correctness

Case distinction:

1.) q in b_v $\quad \rightarrow$ points outside of b_v too far away (to invalidate rep_v as $1/t$-ANN)

2.) q outside enlarged b_v

$\rightarrow$ points inside b_v all have approximately the same distance to q



Intuition of correctness

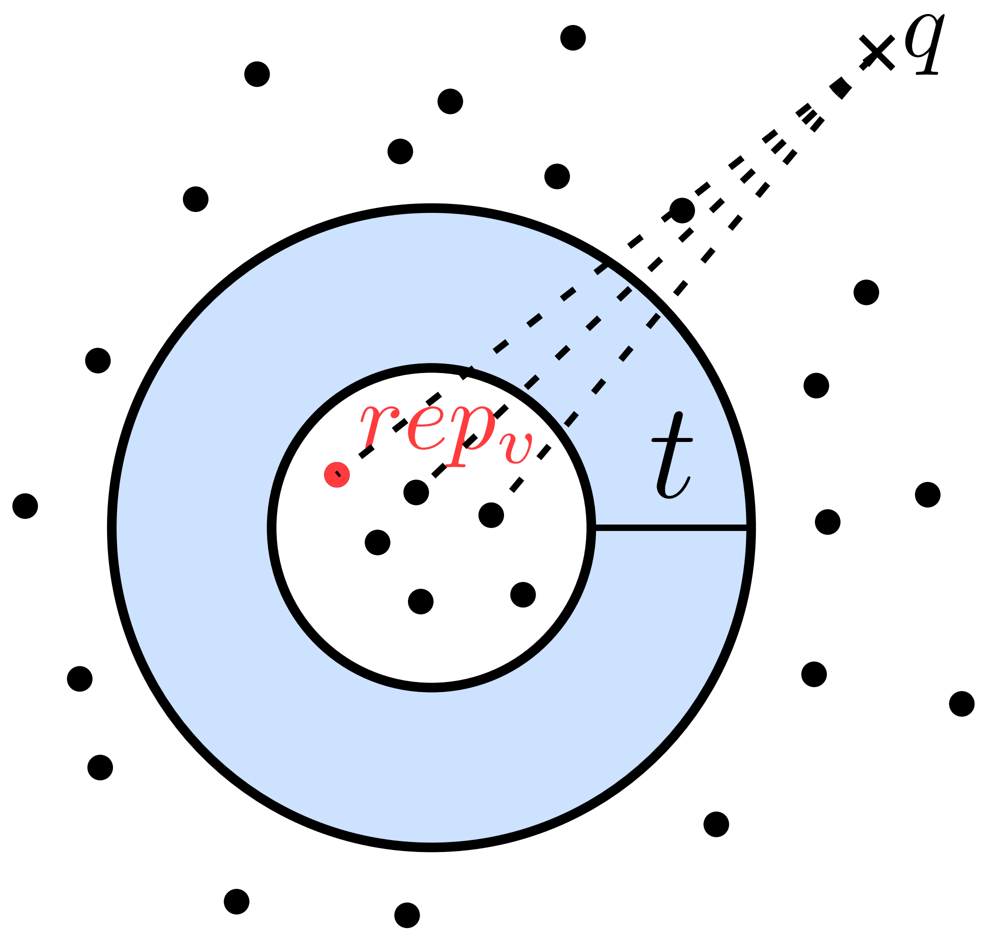
Case distinction:

1.) q in b_v $\rightarrow$ points outside of b_v too far away
(to invalidate rep_v as $1/t$-ANN)

2.) q outside enlarged b_v

$\rightarrow$ points inside b_v all have approximately the same distance to q

$\rightarrow$ sufficient to test 1 point in b_v: rep_v



Intuition of correctness

Case distinction:

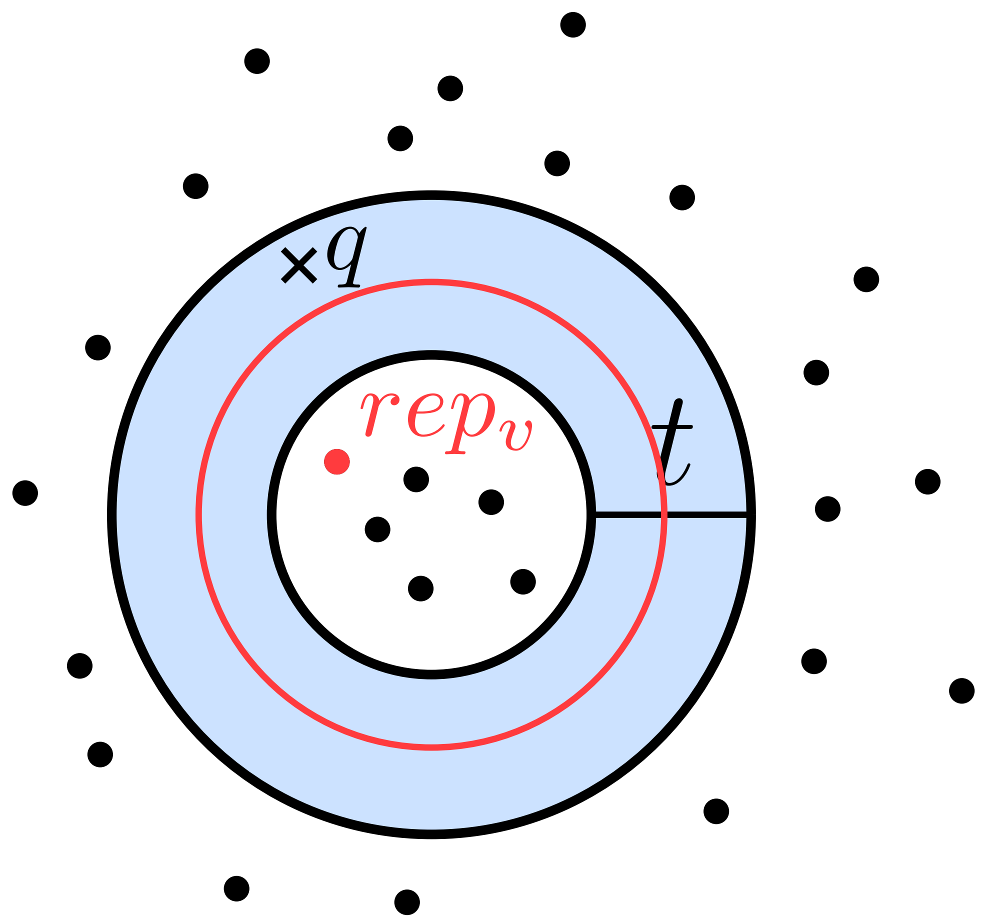
1.) q in b_v $\rightarrow$ points outside of b_v too far away (to invalidate rep_v as $1/t$-ANN)

2.) q outside enlarged b_v

$\rightarrow$ points inside b_v all have approximately the same distance to q

$\rightarrow$ sufficient to test 1 point in b_v: rep_v

3.) q in ring



Intuition of correctness

Case distinction:

1.) q in b_v $\rightarrow$ points outside of b_v too far away (to invalidate rep_v as $1/t$-ANN)

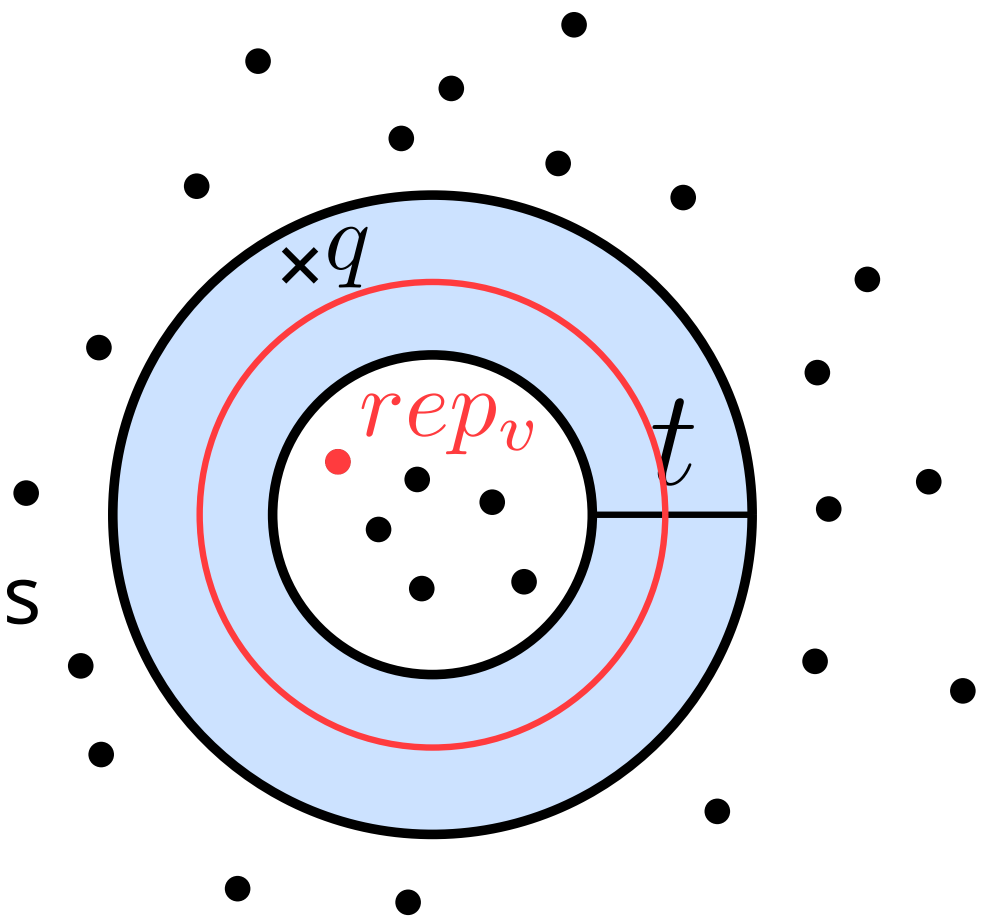
2.) q outside enlarged b_v

$\rightarrow$ points inside b_v all have approximately the same distance to q

$\rightarrow$ sufficient to test 1 point in b_v: rep_v

3.) q in ring

1.) or 2.) applies with slightly worse bounds



Correctness

The algorithm finds a $(1 + 4/t)$-ANN.

Correctness

The algorithm finds a $(1 + 4/t)$-ANN.

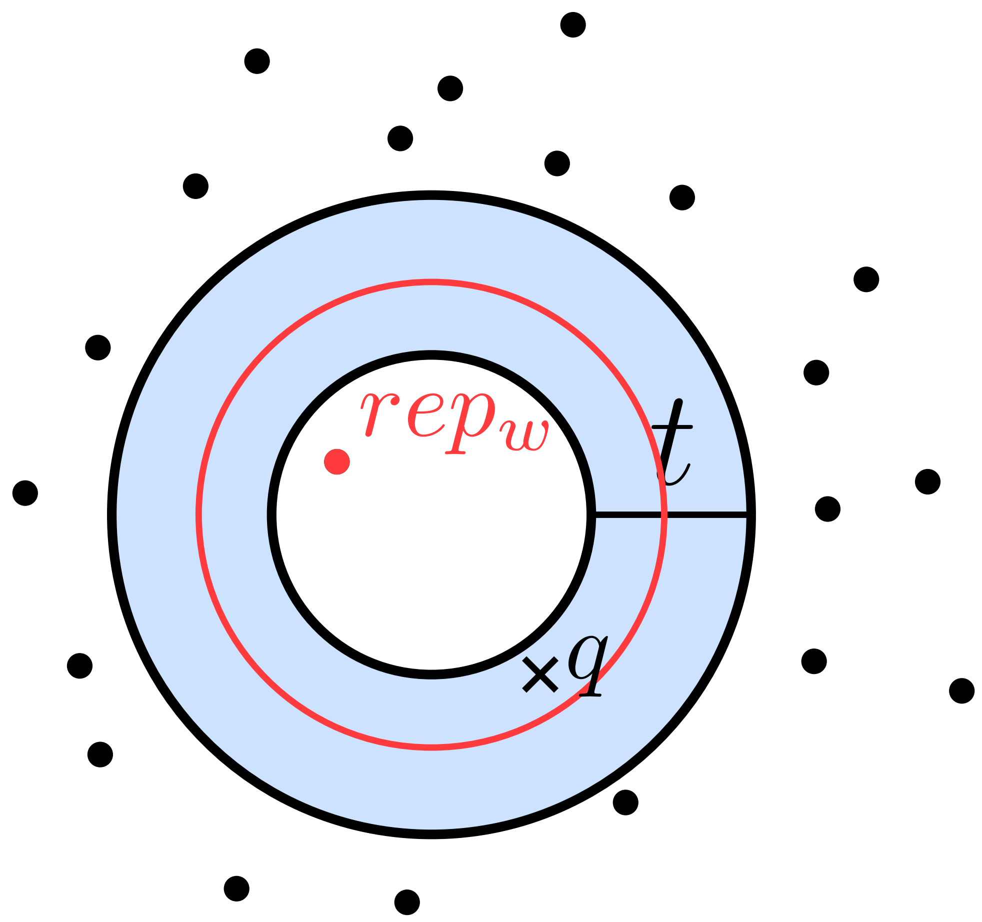
Node w of T: Last node on search path such that $nn(q) \in P_w$.

Correctness

The algorithm finds a $(1 + 4/t)$-ANN.

Node w of T: Last node on search path such that $nn(q) \in P_w$.

Case 1: $nn(q) \in P^w_{out}$ but $\|q - c_w\| \leq r_w(1 + 1/t)$



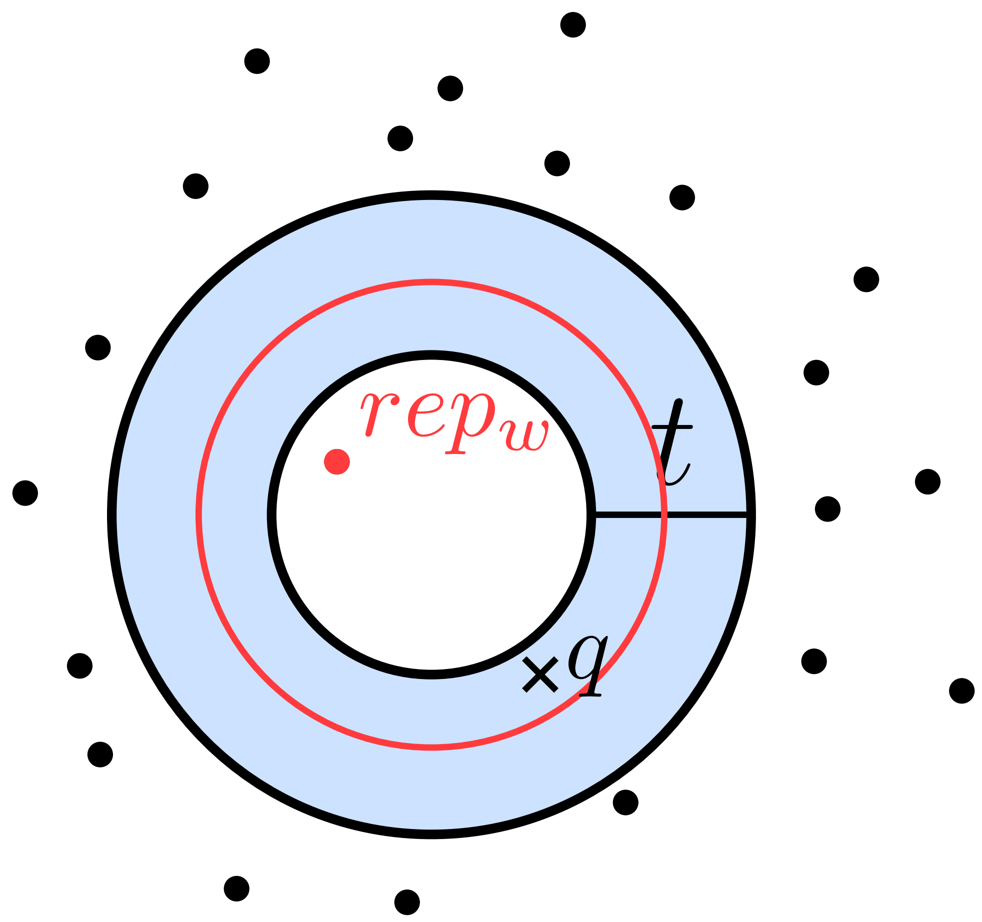
Correctness

The algorithm finds a $(1+4/t)$-ANN.

Node w of T: Last node on search path such that $nn(q) \in P_w$.

Case 1: $nn(q) \in P^w_{out}$ but $\|q - c_w\| \leq r_w(1+1/t)$

$$\frac{\|q - rep_w\|}{\|q - nn(q)\|} \leq \frac{(2+t/2)r_w}{(t/2)r_w} \leq 1 + 4/t$$



Correctness

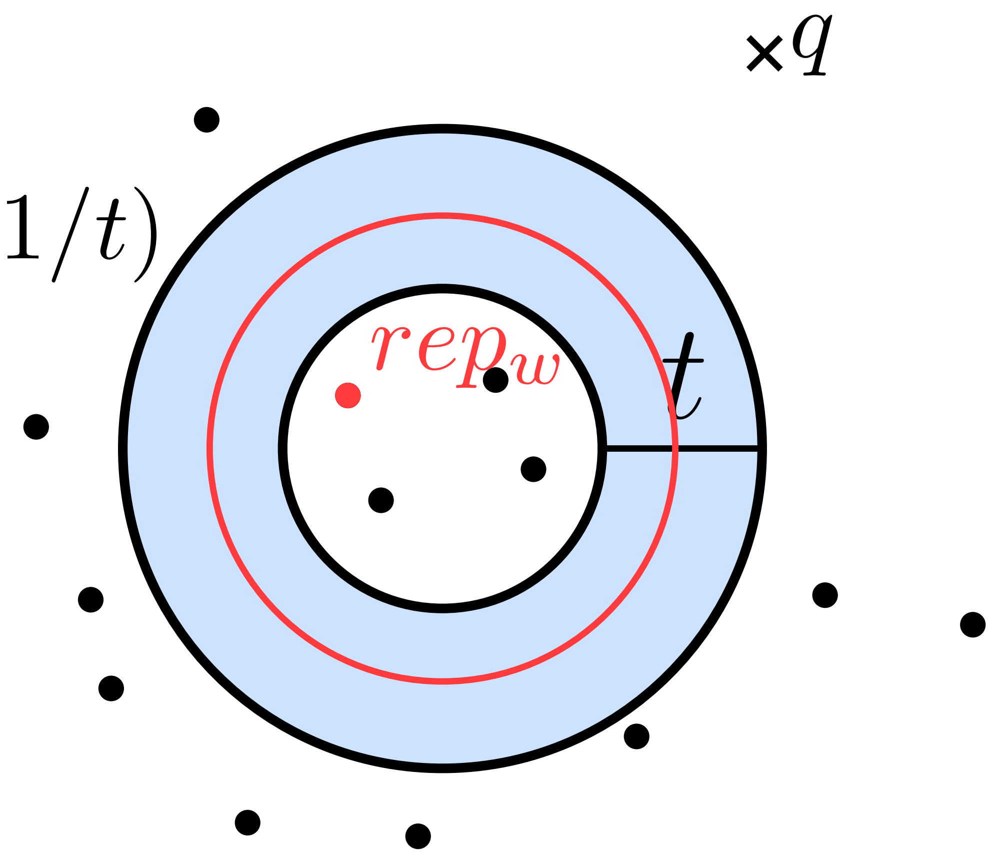
The algorithm finds a $(1+4/t)$-ANN.

Node w of T: Last node on search path such that $nn(q) \in P_w$.

Case 1: $nn(q) \in P_{out}^w$ but $\|q - c_w\| \leq r_w(1+1/t)$

$$\frac{\|q - rep_w\|}{\|q - nn(q)\|} \leq \frac{(2+t/2)r_w}{(t/2)r_w} \leq 1 + 4/t$$

Case 2: $nn(q) \in P_{in}^w$ but $\|q - c_w\| \geq r_w(1+1/t)$



Correctness

The algorithm finds a $(1+4/t)$-ANN.

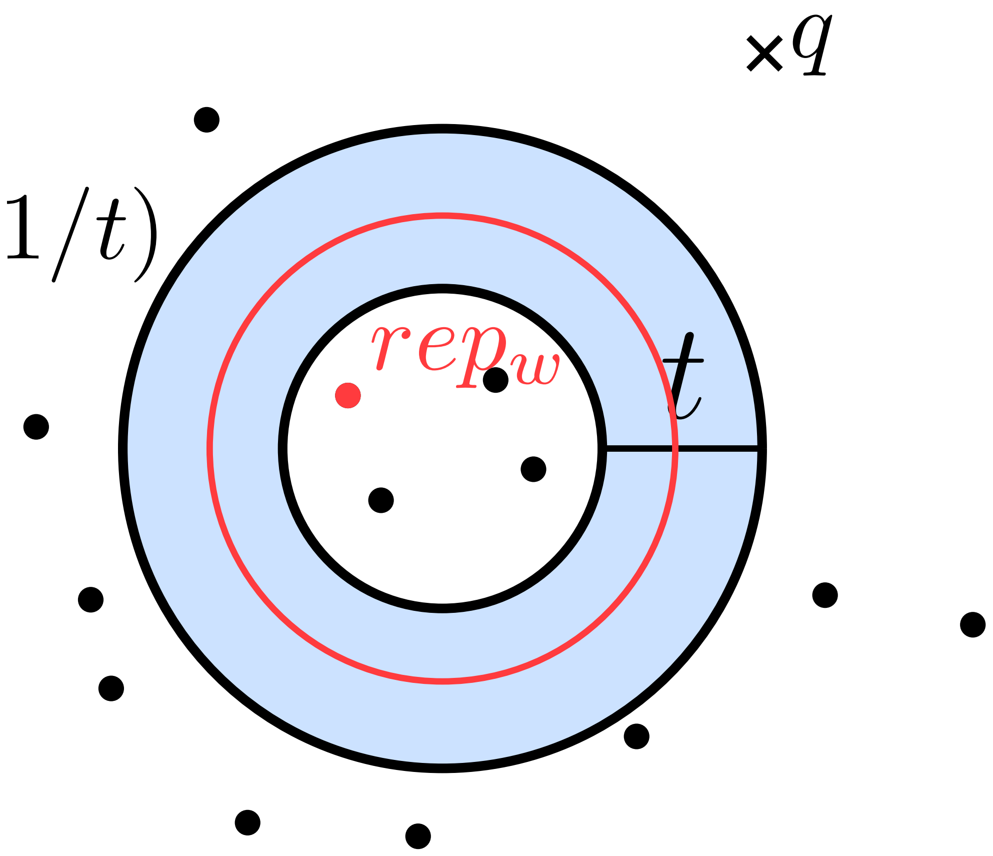
Node w of T: Last node on search path such that $nn(q) \in P_w$.

Case 1: $nn(q) \in P^w_{out}$ but $\|q - c_w\| \le r_w(1+1/t)$

$$\frac{\|q-rep_w\|}{\|q-nn(q)\|} \le \frac{(2+t/2)r_w}{(t/2)r_w} \le 1 + 4/t$$

Case 2: $nn(q) \in P^w_{in}$ but $\|q - c_w\| \ge r_w(1+1/t)$

$$\frac{\|q-rep_w\|}{\|q-nn(q)\|} \le \frac{\|q-nn(q)\|+\|nn(q)-rep_w\|}{\|q-nn(q)\|}$$



Correctness

The algorithm finds a $(1+4/t)$-ANN.

Node w of T: Last node on search path such that $nn(q) \in P_w$.

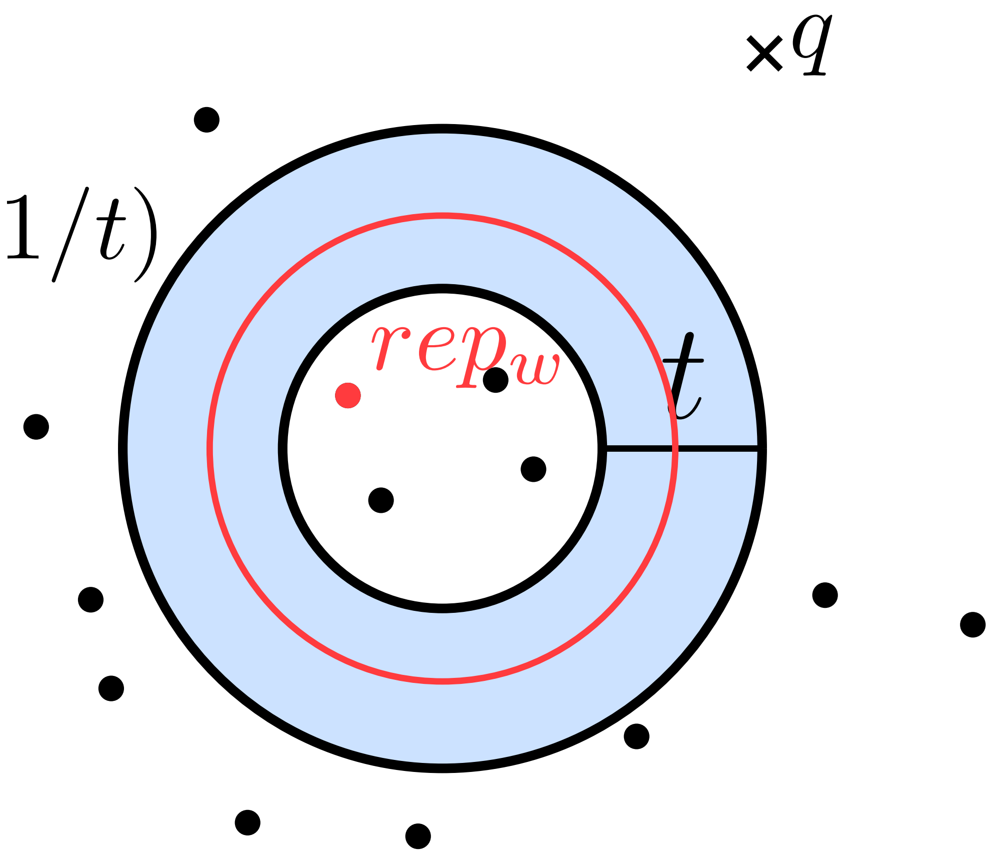
Case 1: $nn(q) \in P_{out}^{w}$ but $\|q - c_w\| \leq r_w(1+1/t)$

$$\frac{\|q-rep_w\|}{\|q-nn(q)\|} \leq \frac{(2+t/2)r_w}{(t/2)r_w} \leq 1+4/t$$

Case 2: $nn(q) \in P_{in}^{w}$ but $\|q - c_w\| \geq r_w(1+1/t)$

$$\frac{\|q-rep_w\|}{\|q-nn(q)\|} \leq \frac{\|q-nn(q)\|+\|nn(q)-rep_w\|}{\|q-nn(q)\|}$$

$$\leq 1+\frac{2r_w}{(t/2)r_w} = 1+4/t$$



Overview

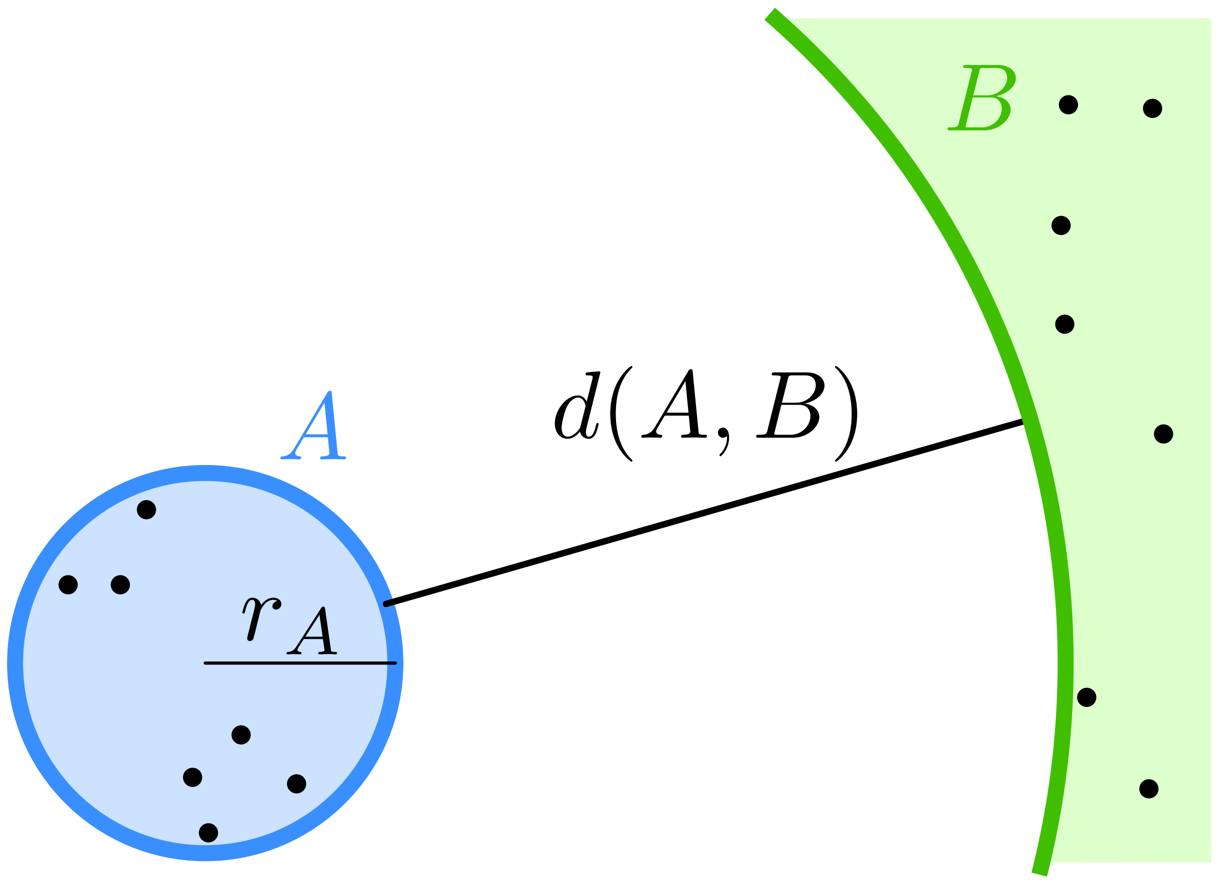
Semi-separated pair decomposition (SSPD)

Ring separator: n-semi-separated pair decomposition

Ring separator tree: $O(n)$-ANN

$(1/\varepsilon)$-semi-separated pair decomposition – brief sketch

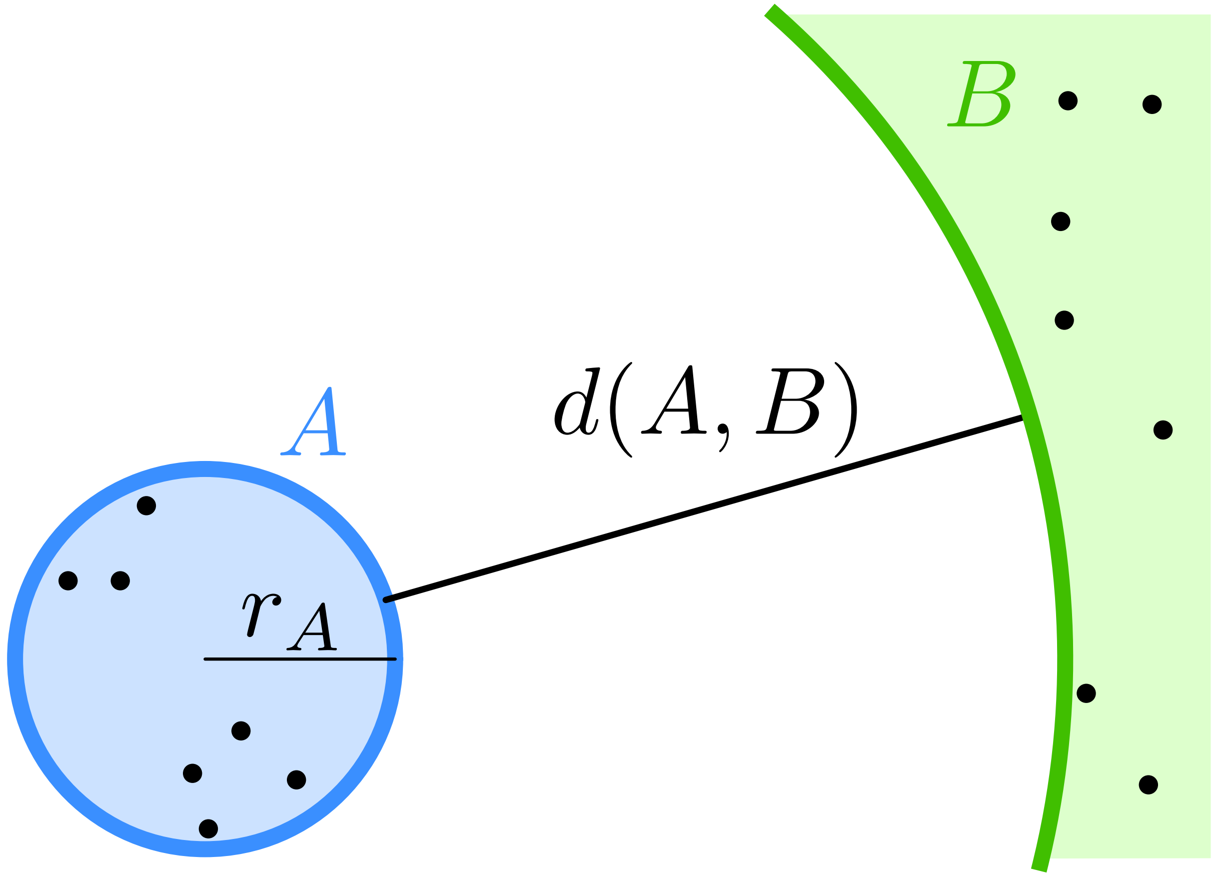
$(1/\varepsilon)$-Semi-Separated Pairs



ring separator:

- ball $b = b(p, r)$ containing $\geq n/c_1$ points
- no point in $b(p, r(1 + 1/n)) \setminus b$
- $\geq n/c_2$ points outside $b(p, 2r)$

$(1/\varepsilon)$-Semi-Separated Pairs

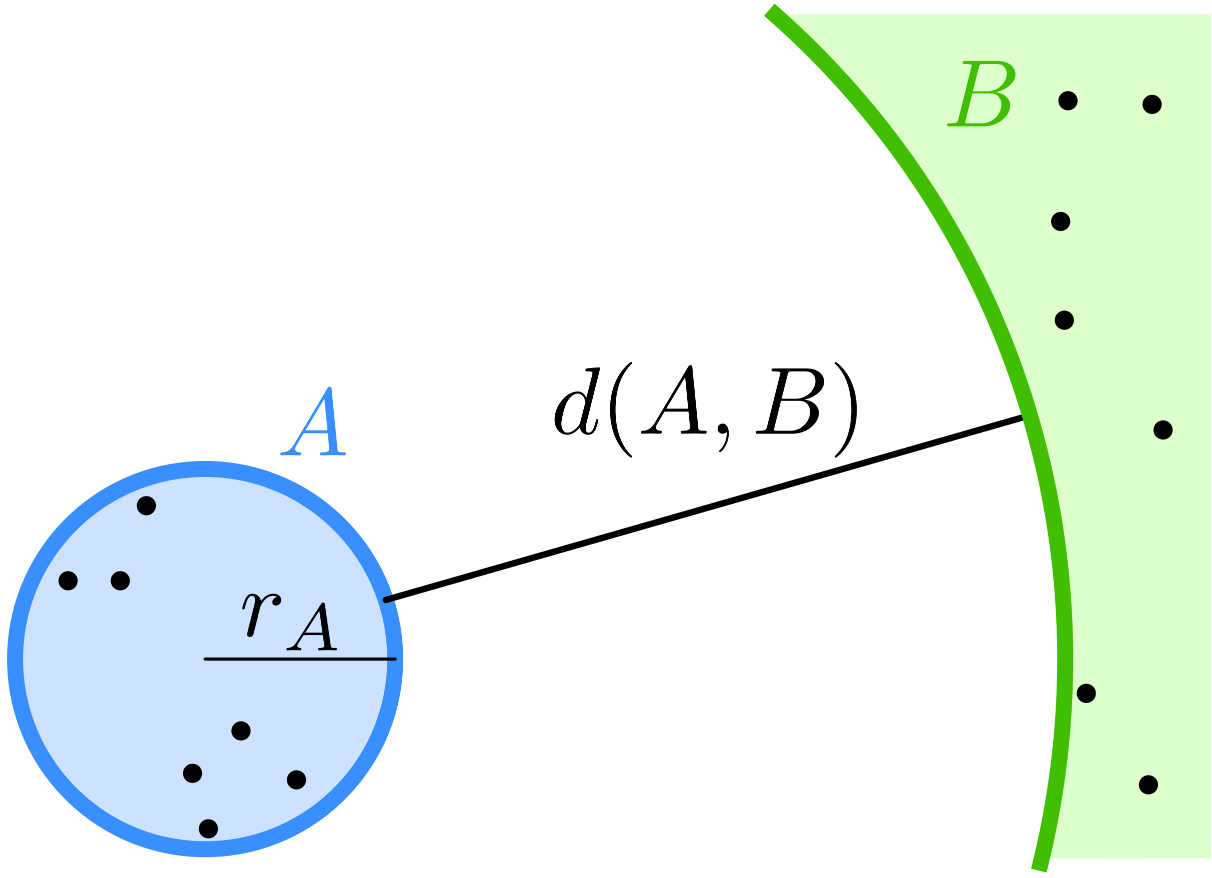


$$P_{in} = P \cap b, P_{far} = P \setminus b(p, 2r/\varepsilon)$$
$$P_{out} = P \setminus (P_{in} \cup P_{far})$$

ring separator:

- ball $b = b(p, r)$ containing $\geq n/c_1$ points
- no point in $b(p, r(1 + 1/n)) \setminus b$
- $\geq n/c_2$ points outside $b(p, 2r)$

$(1/\varepsilon)$-Semi-Separated Pairs



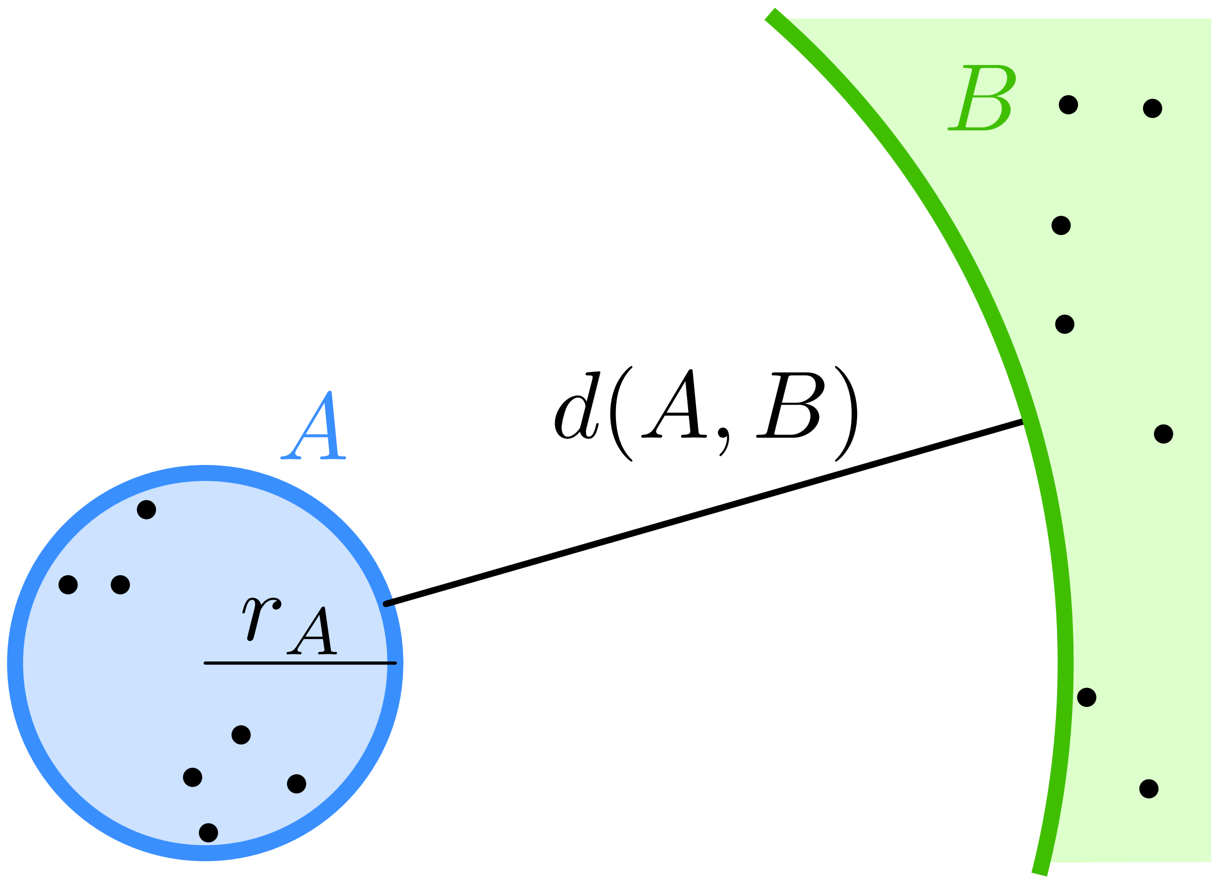
$P_{in} = P \cap b, P_{far} = P \setminus b(p, 2r/\varepsilon)$
$P_{out} = P \setminus (P_{in} \cup P_{far})$

1. P_{in}, P_{far} semi-separated
2. recurse on P_{in}, P_{in}

ring separator:

- ball $b = b(p, r)$ containing $\geq n/c_1$ points
- no point in $b(p, r(1 + 1/n)) \setminus b$
- $\geq n/c_2$ points outside $b(p, 2r)$

$(1/\varepsilon)$-Semi-Separated Pairs



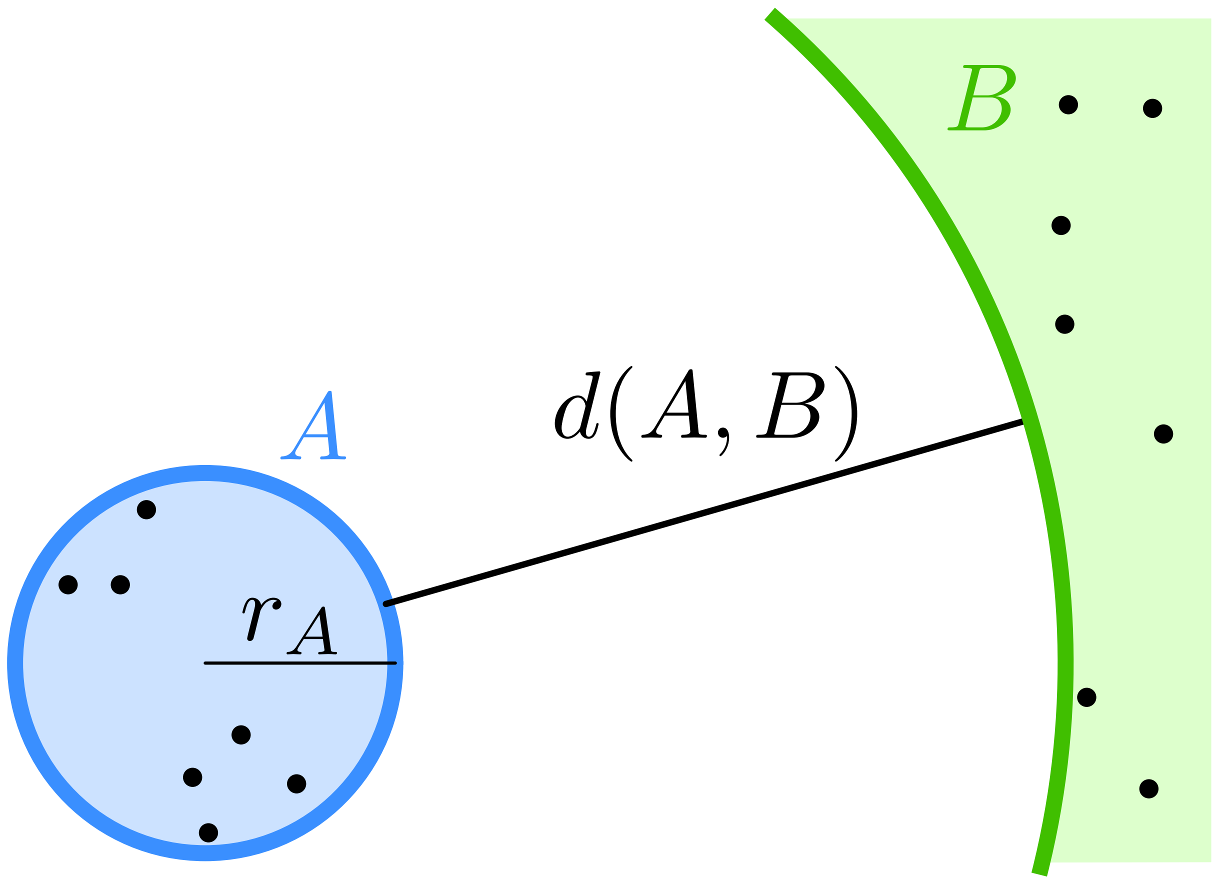
$$P_{in} = P \cap b, P_{far} = P \setminus b(p, 2r/\varepsilon)$$
$$P_{out} = P \setminus (P_{in} \cup P_{far})$$

1. P_{in}, P_{far} semi-separated
2. recurse on P_{in}, P_{in}
3. recurse on $P \setminus P_{in}, P \setminus P_{in}$

ring separator:

- ball $b = b(p, r)$ containing $\geq n/c_1$ points
- no point in $b(p, r(1 + 1/n)) \setminus b$
- $\geq n/c_2$ points outside $b(p, 2r)$

$(1/\varepsilon)$-Semi-Separated Pairs



$P_{in} = P \cap b, P_{far} = P \setminus b(p, 2r/\varepsilon)$
$P_{out} = P \setminus (P_{in} \cup P_{far})$

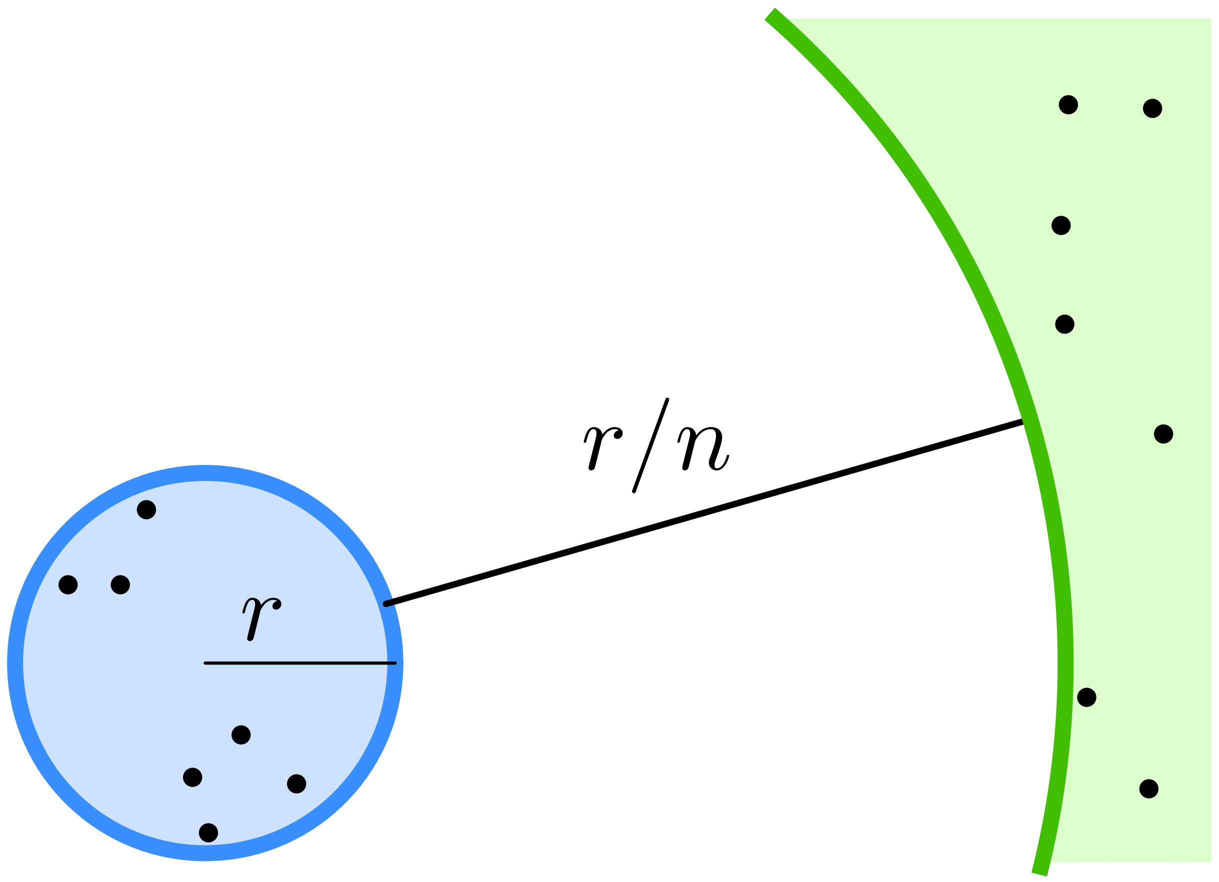
1. P_{in}, P_{far} semi-separated
2. recurse on P_{in}, P_{in}
3. recurse on $P \setminus P_{in}, P \setminus P_{in}$

difficult: 4. P_{in}, P_{out}

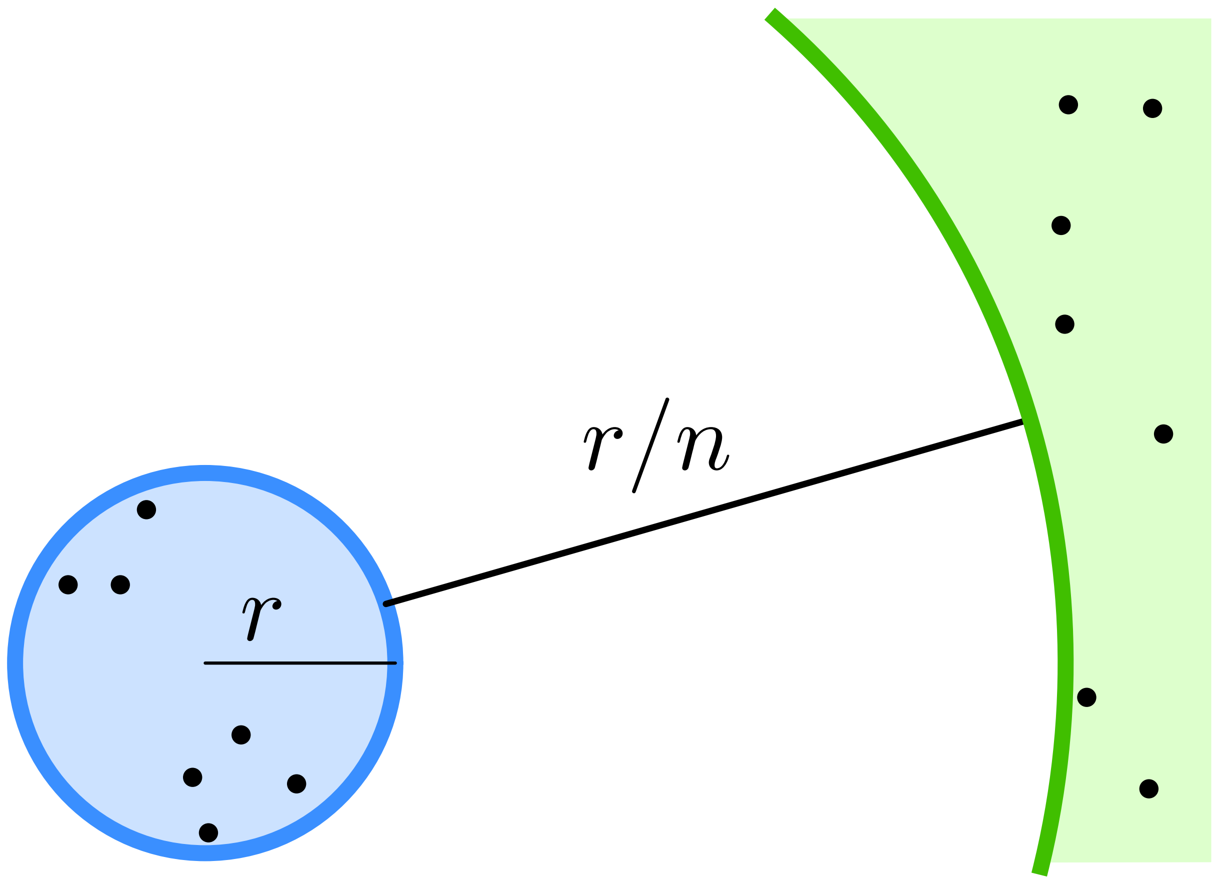
ring separator:

- ball $b = b(p, r)$ containing $\geq n/c_1$ points
- no point in $b(p, r(1 + 1/n)) \setminus b$
- $\geq n/c_2$ points outside $b(p, 2r)$

Dealing with P_{in}, P_{out} (sketch)

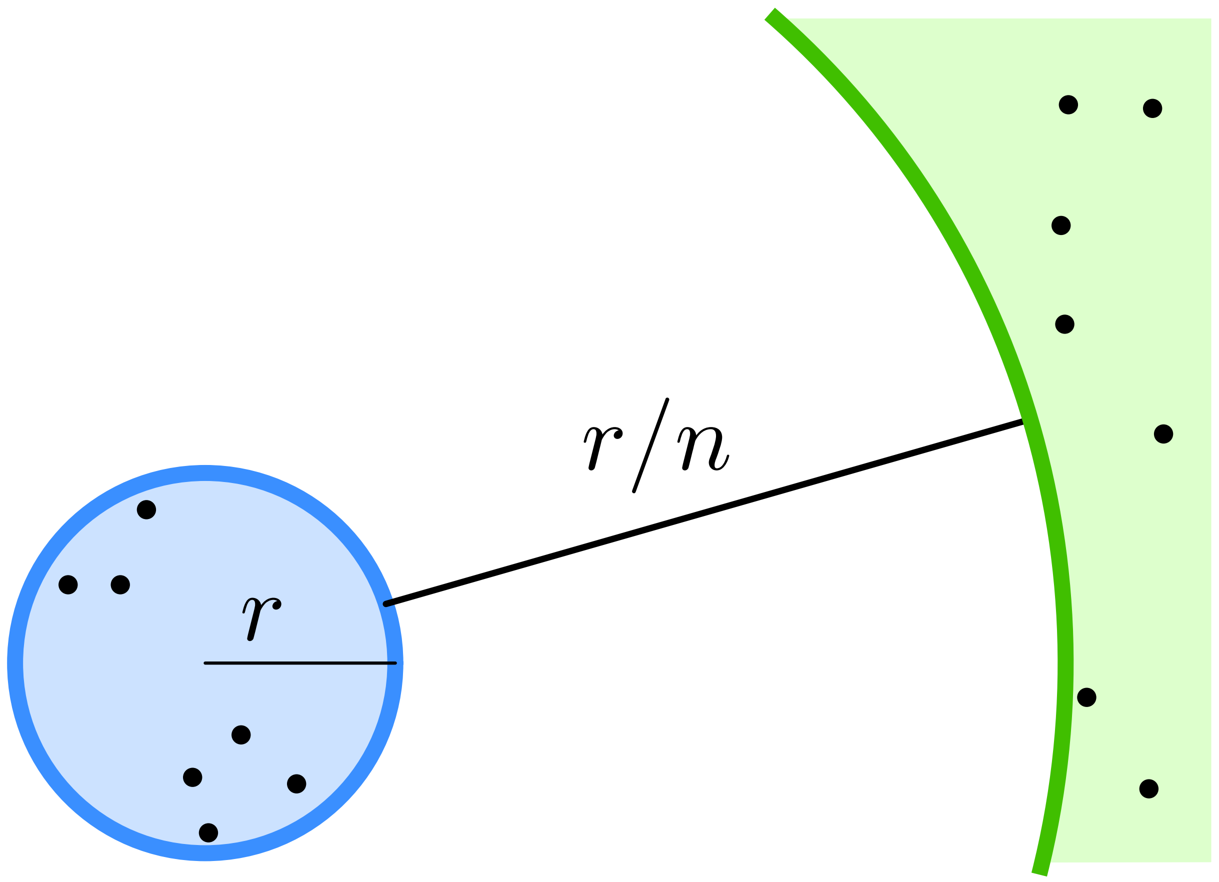


Dealing with P_{in}, P_{out} (sketch)



$diam(P_{in} \cup P_{out}) \leq 2r/\varepsilon$

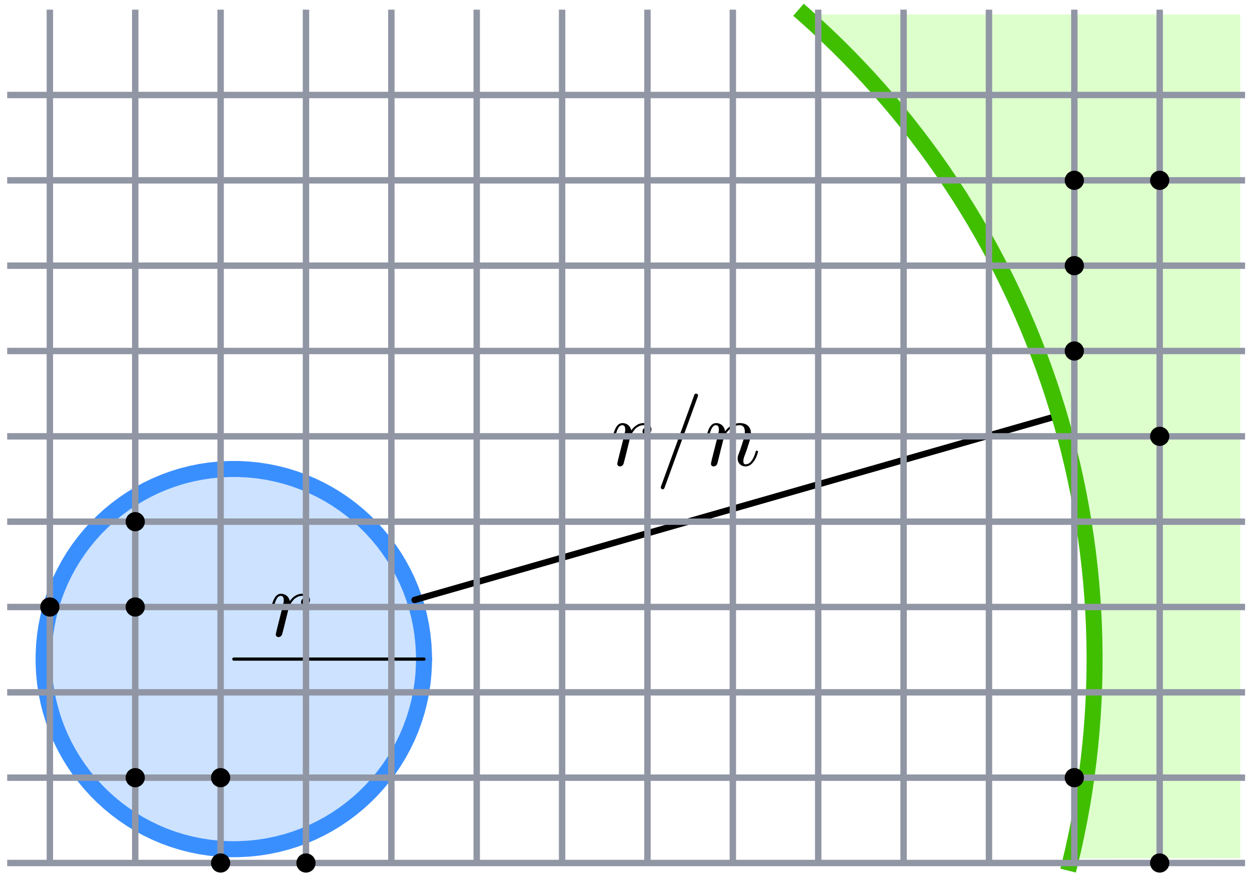
Dealing with P_{in}, P_{out} (sketch)



$diam(P_{in} \cup P_{out}) \leq 2r/\varepsilon$

$\ell := \min_{p \in P_{in}, q \in P_{out}} \|p - q\| \geq r/n$

Dealing with P_{in}, P_{out} (sketch)

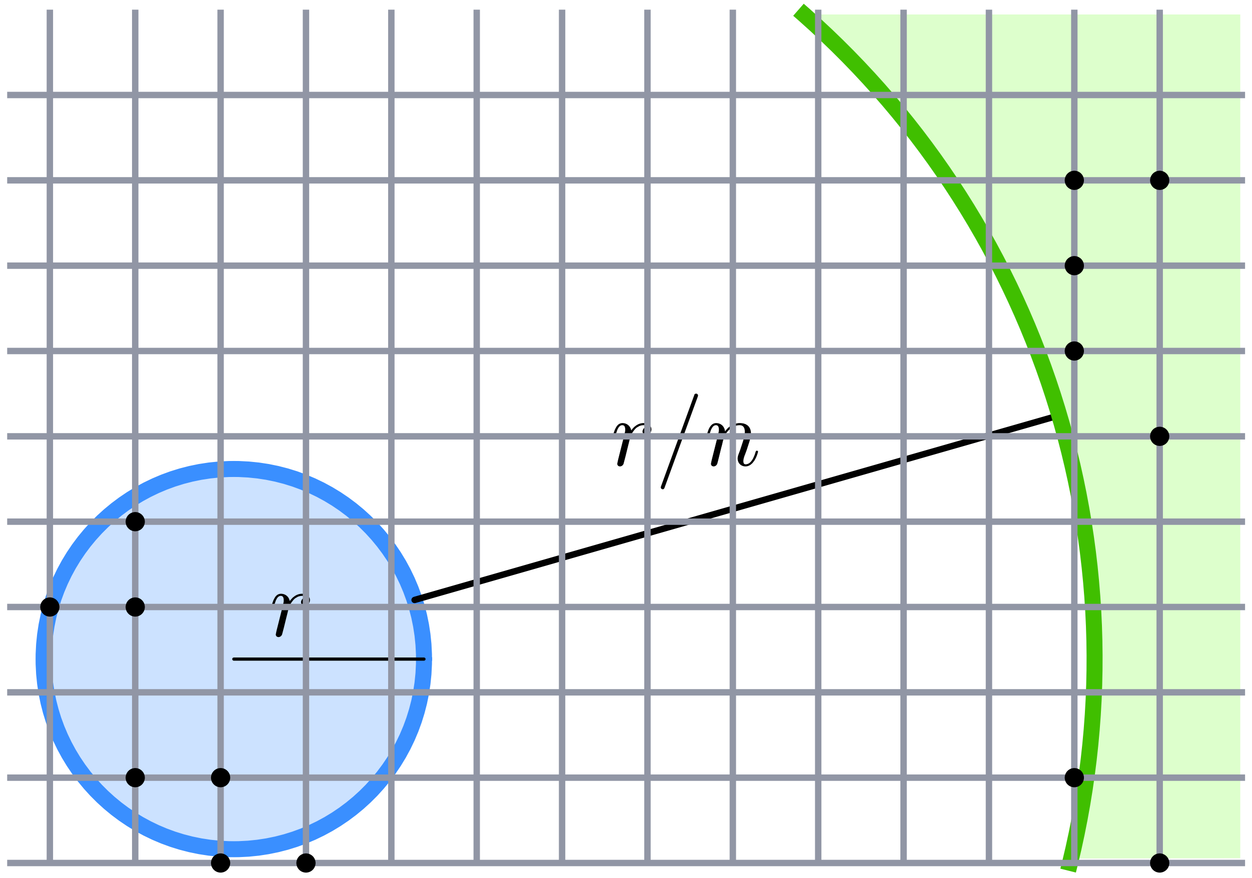


$diam(P_{in} \cup P_{out}) \leq 2r/\varepsilon$

$\ell := \min_{p \in P_{in}, q \in P_{out}} \|p - q\| \geq r/n$

snap points to a grid G_α with $\alpha = \varepsilon\ell/10$

Dealing with P_{in}, P_{out} (sketch)



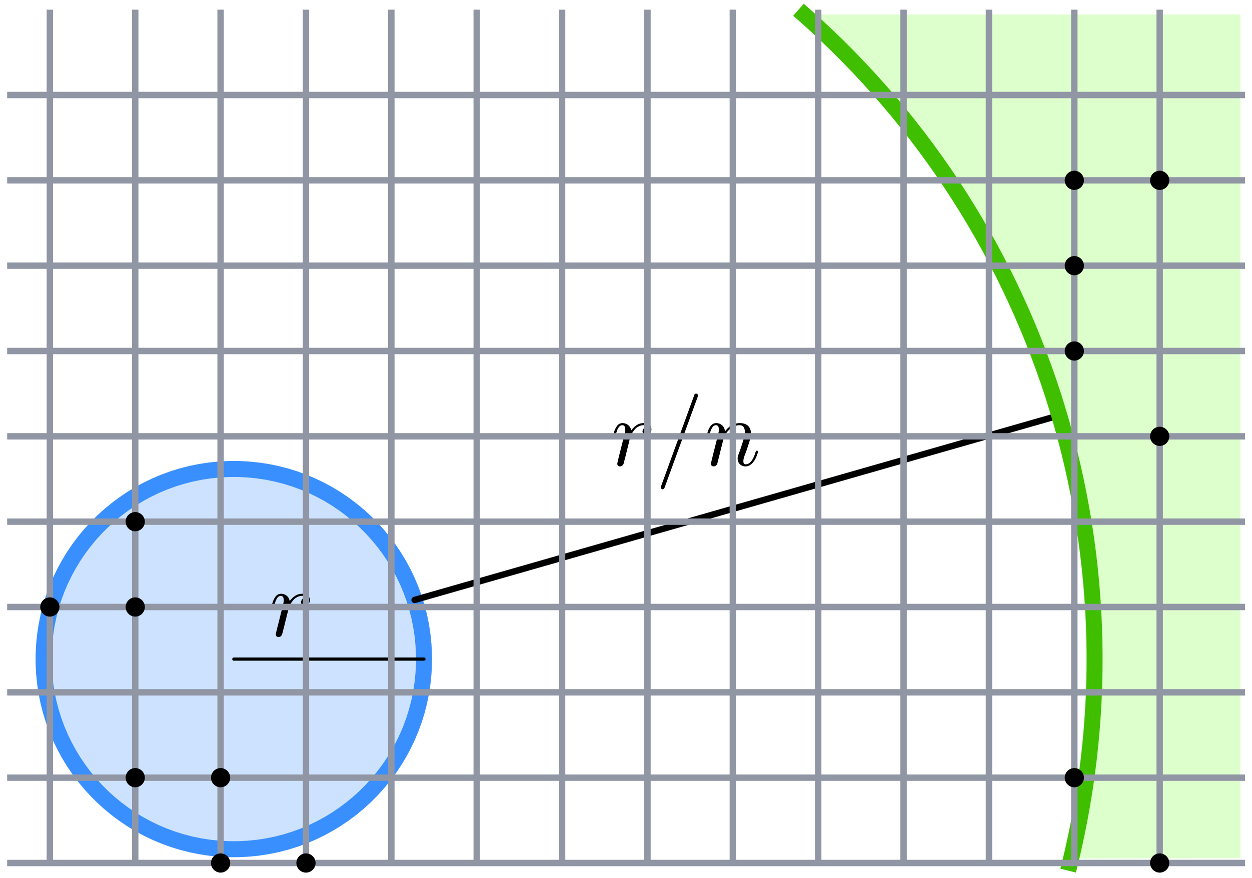
$diam(P_{in} \cup P_{out}) \leq 2r/\varepsilon$

$\ell := \min_{p \in P_{in}, q \in P_{out}} \|p - q\| \geq r/n$

snap points to a grid G_α with $\alpha = \varepsilon\ell/10$

Use WSPD algorithm for bounded spread to compute WSPs with $A \subset P_{in}$ and $B \subset P_{out}$

Dealing with P_{in}, P_{out} (sketch)



$diam(P_{in} \cup P_{out}) \leq 2r/\varepsilon$

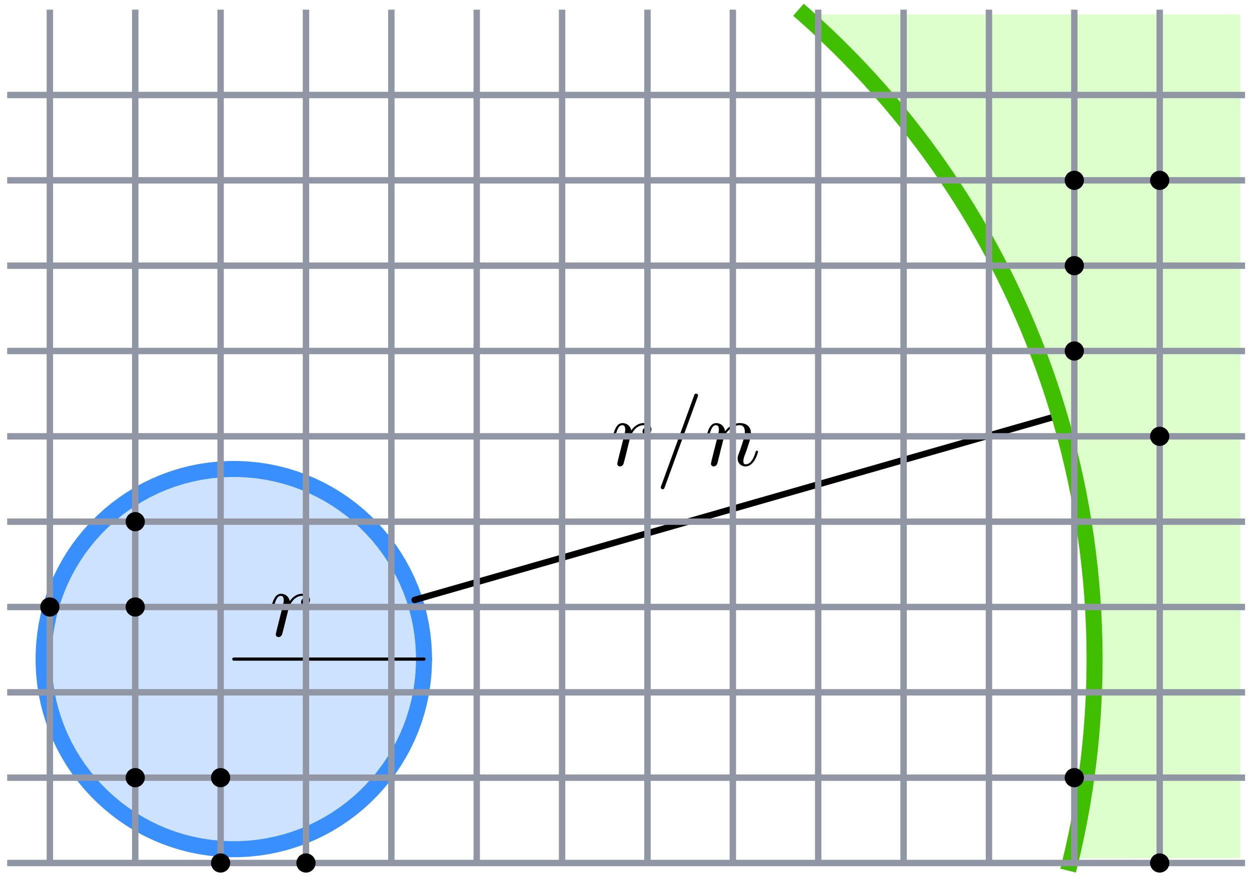
$\ell := \min_{p \in P_{in}, q \in P_{out}} \|p - q\| \geq r/n$

snap points to a grid G_α with $\alpha = \varepsilon\ell/10$

Use WSPD algorithm for bounded spread to compute WSPs with $A \subset P_{in}$ and $B \subset P_{out}$

- grid size: $O(n/\varepsilon^2)$

Dealing with P_{in}, P_{out} (sketch)



$diam(P_{in} \cup P_{out}) \leq 2r/\varepsilon$

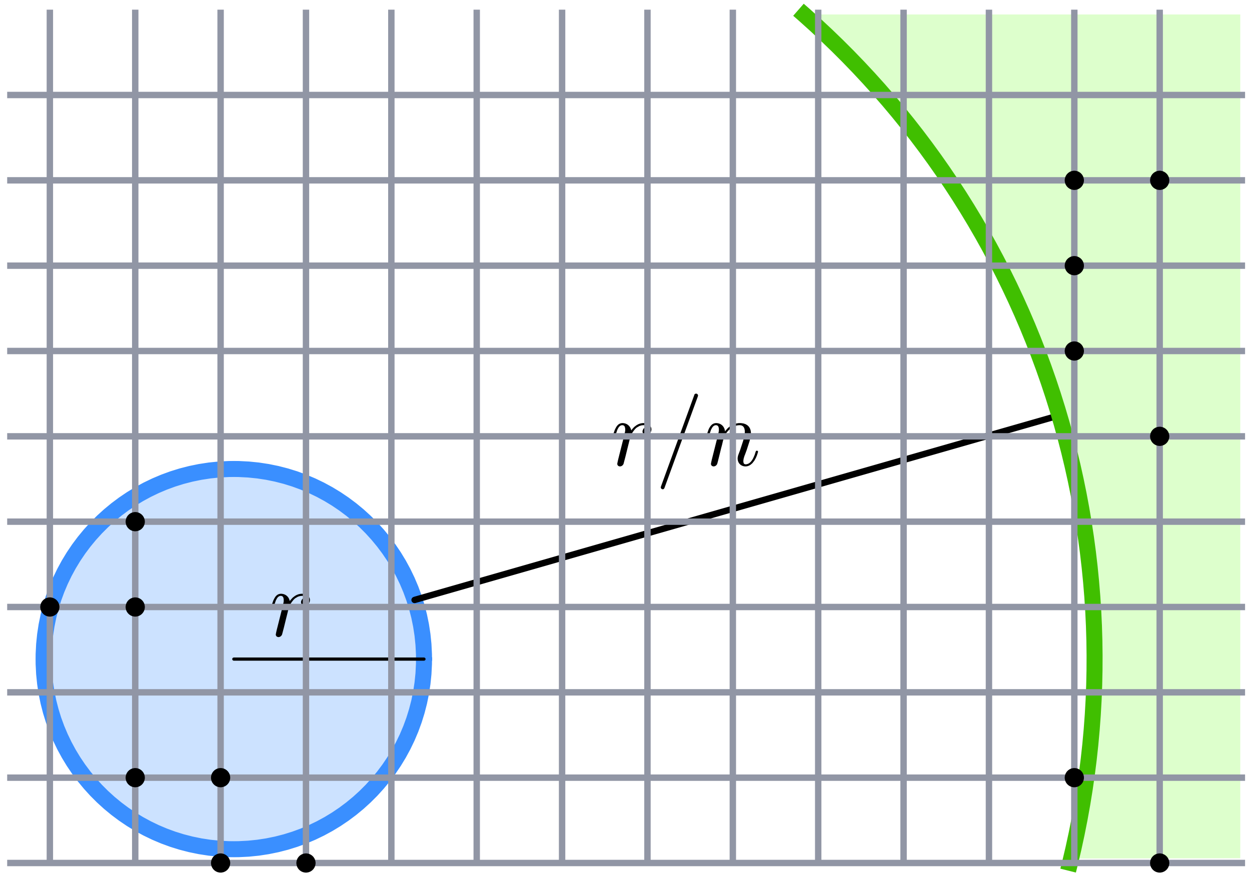
$\ell := \min_{p \in P_{in}, q \in P_{out}} \|p - q\| \geq r/n$

snap points to a grid G_α with $\alpha = \varepsilon\ell/10$

Use WSPD algorithm for bounded spread to compute WSPs with $A \subset P_{in}$ and $B \subset P_{out}$

- grid size: $O(n/\varepsilon^2)$
- levels in quadtree: $O(\log(n/\varepsilon^2))$
  $\rightarrow$ # pairs = $O(n \log n)$ (low weight)

Dealing with P_{in}, P_{out} (sketch)



$diam(P_{in} \cup P_{out}) \leq 2r/\varepsilon$

$\ell := \min_{p \in P_{in}, q \in P_{out}} \|p - q\| \geq r/n$

snap points to a grid G_α with $\alpha = \varepsilon\ell/10$

Use WSPD algorithm for bounded spread to compute WSPs with $A \subset P_{in}$ and $B \subset P_{out}$

- grid size: $O(n/\varepsilon^2)$
- levels in quadtree: $O(\log(n/\varepsilon^2))$
  $\rightarrow$ # pairs = $O(n \log n)$ (low weight)
- snapping: since $\ell \geq r/n$ distances between $p \in P_{in}$ and $q \in P_{out}$ approximately stay the same.

Summary

$(1/\varepsilon)$-Semi-separated pair decomposition (SSPD)

$$\min(r_A, r_B) \leq \varepsilon d(A, B)$$

Summary

$(1/\varepsilon)$-Semi-separated pair decomposition (SSPD)

$$\min(r_A, r_B) \leq \varepsilon d(A, B)$$

Ring separator tree: n-semi-separated pair decomposition

enclose constant fraction A of P by ball $b(c, r)$ s.t. $dist(A, P \setminus A) \geq r/n$

Summary

$(1/\varepsilon)$-Semi-separated pair decomposition (SSPD)

$\min(r_A, r_B) \leq \varepsilon d(A, B)$

Ring separator tree: n-semi-separated pair decomposition

enclose constant fraction A of P by ball $b(c, r)$ s.t. $dist(A, P \setminus A) \geq r/n$

Ring separator tree: $O(n)$-ANN

data structure of size $O(n)$ computed in $O(n \log n)$ time,
which gives n-ANN in $O(\log n)$ time

Summary

$(1/\varepsilon)$-Semi-separated pair decomposition (SSPD)

$\min(r_A, r_B) \leq \varepsilon d(A, B)$

Ring separator tree: n-semi-separated pair decomposition

enclose constant fraction A of P by ball $b(c, r)$ s.t. $dist(A, P \setminus A) \geq r/n$

Ring separator tree: $O(n)$-ANN

data structure of size $O(n)$ computed in $O(n \log n)$ time,
which gives n-ANN in $O(\log n)$ time

$(1/\varepsilon)$-semi-separated pair decomposition

n-ring separator + snap to grid + WSPD