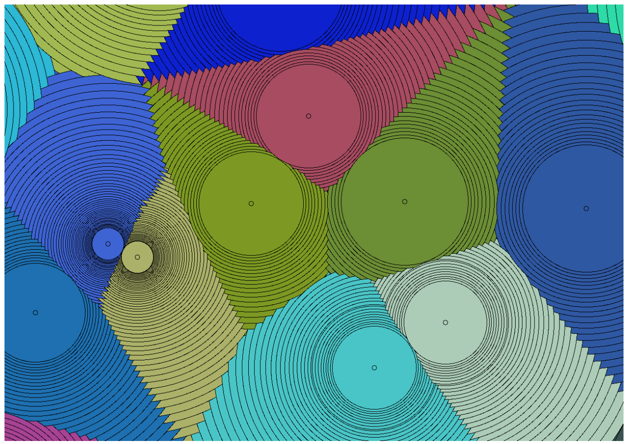
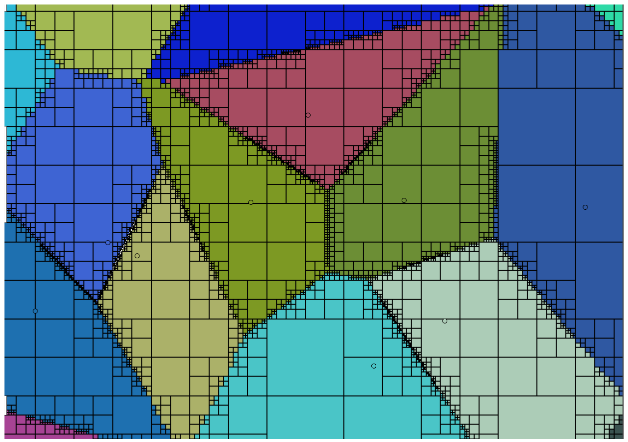
Approximate Voronoi Diagrams

Presentation by Maks Ovsjanikov



S. Har-Peled's notes, **Chapters 6** and **7**

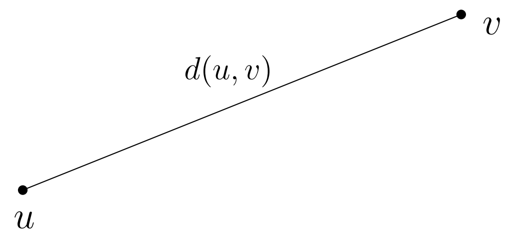


Outline

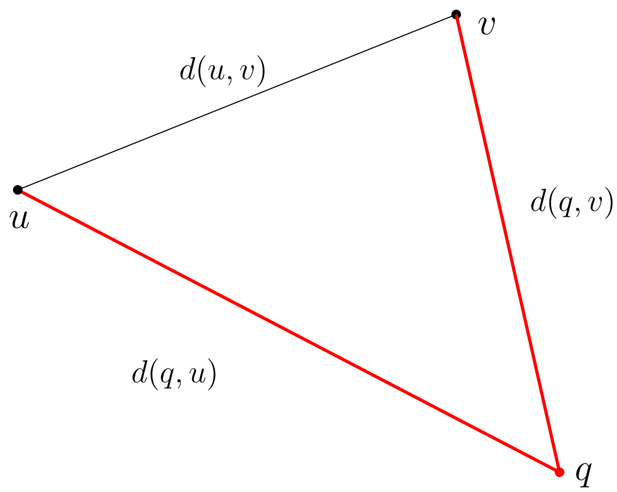
- Preliminaries
- Problem Statement
- ANN using PLEB
  } (Previous Lecture)
- Bounds and Improvements
  - Near Linear Space
  - Linear Space
- ANN in $\mathbb{R}^d$ using compressed quad-trees

Preliminaries

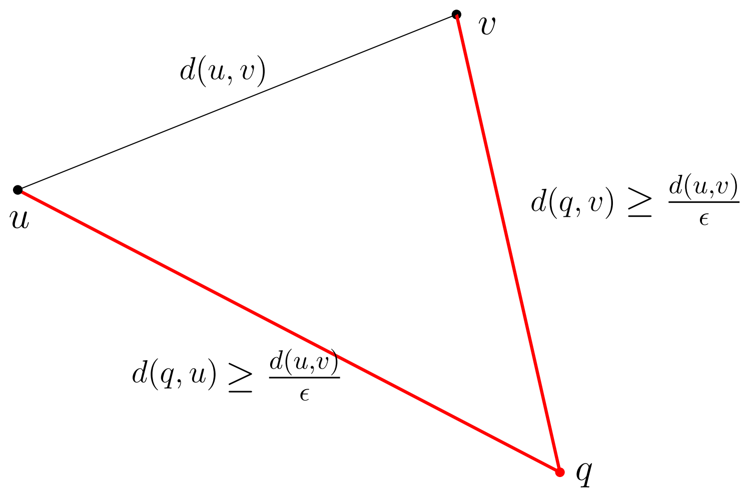
Preliminaries



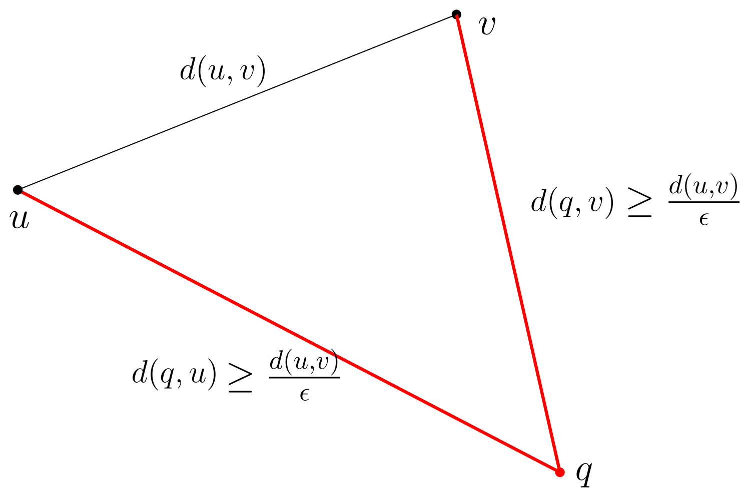
Preliminaries



Preliminaries

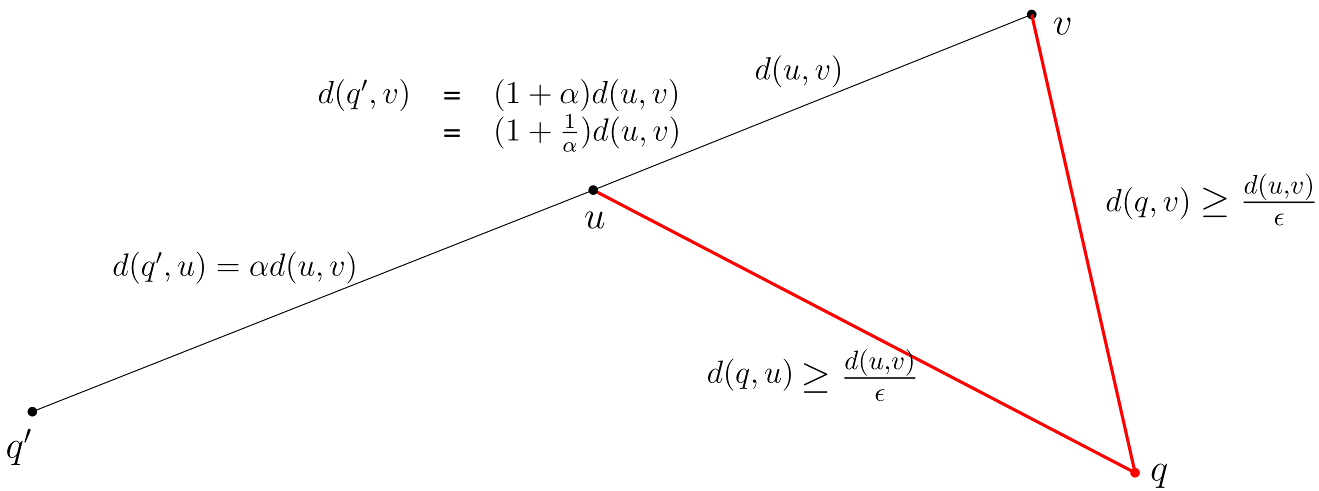


Preliminaries



$$\begin{cases} d(q, u) \geq \dfrac{d(u,v)}{\epsilon} \\ d(q, v) \geq \dfrac{d(u,v)}{\epsilon} \end{cases} \Longrightarrow \boxed{\dfrac{d(q, v)}{d(q, u)} \leq 1 + \epsilon}$$

Preliminaries



$$\begin{cases} d(q,u) \geq \frac{d(u,v)}{\epsilon} \\ d(q,v) \geq \frac{d(u,v)}{\epsilon} \end{cases} \implies \boxed{\frac{d(q,v)}{d(q,u)} \leq 1+\epsilon}$$

Preliminaries

$$\begin{cases} d(q,u) \geq \frac{d(u,v)}{\epsilon} \\ d(q,v) \geq \frac{d(u,v)}{\epsilon} \end{cases} \Longrightarrow \boxed{\frac{d(q,v)}{d(q,u)} \leq 1+\epsilon}$$

Holds in any metric space:

Preliminaries

$$\begin{cases} d(q,u) \geq \frac{d(u,v)}{\epsilon} \\ d(q,v) \geq \frac{d(u,v)}{\epsilon} \end{cases} \Longrightarrow \boxed{\frac{d(q,v)}{d(q,u)} \leq 1+\epsilon}$$

Holds in any metric space:

$$d(q,u) = \alpha d(u,v)$$

$$d(q,v) \leq d(q,u) + d(u,v) = (1+\tfrac{1}{\alpha})d(q,u)$$

$\Longrightarrow \frac{d(q,v)}{d(q,u)} \leq (1+\frac{1}{\alpha}) \leq (1+\epsilon)$ if $\alpha \geq \frac{1}{\epsilon}$

Preliminaries

$$\begin{cases} d(q,u) \geq \frac{d(u,v)}{\epsilon} \\ d(q,v) \geq \frac{d(u,v)}{\epsilon} \end{cases} \Longrightarrow \boxed{\frac{d(q,v)}{d(q,u)} \leq 1+\epsilon}$$

Holds in any metric space:

$$d(q,u) = \alpha d(u,v)$$

$$d(q,v) \leq d(q,u) + d(u,v) = (1+\tfrac{1}{\alpha})d(q,u)$$

$$\Longrightarrow \tfrac{d(q,v)}{d(q,u)} \leq (1+\tfrac{1}{\alpha}) \leq (1+\epsilon) \text{ if } \alpha \geq \tfrac{1}{\epsilon}$$

Similarly:

$$d(q,v) = \alpha d(u,v)$$

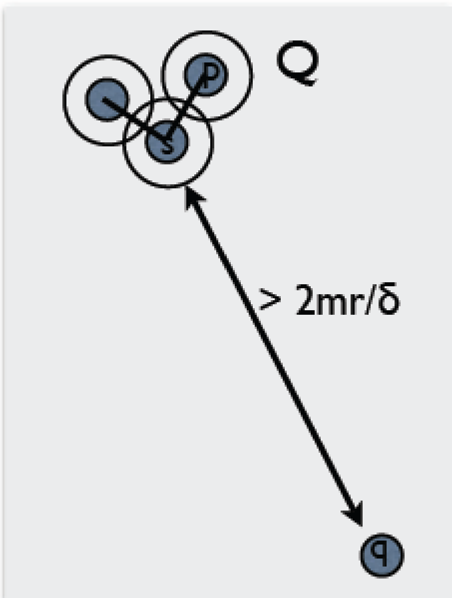
$$\Longrightarrow \tfrac{d(q,u)}{d(q,v)} \leq (1+\tfrac{1}{\alpha}) \leq (1+\epsilon) \text{ if } \alpha \geq \tfrac{1}{\epsilon}$$

Preliminaries

$$\begin{cases} d(q,u) \geq \frac{d(u,v)}{\epsilon} \\ d(q,v) \geq \frac{d(u,v)}{\epsilon} \end{cases} \Longrightarrow \boxed{\frac{d(q,v)}{d(q,u)} \leq 1+\epsilon}$$

Moral:

Any of the far away points is a $(1+\epsilon)$ closest neighbor



Problem Statement:

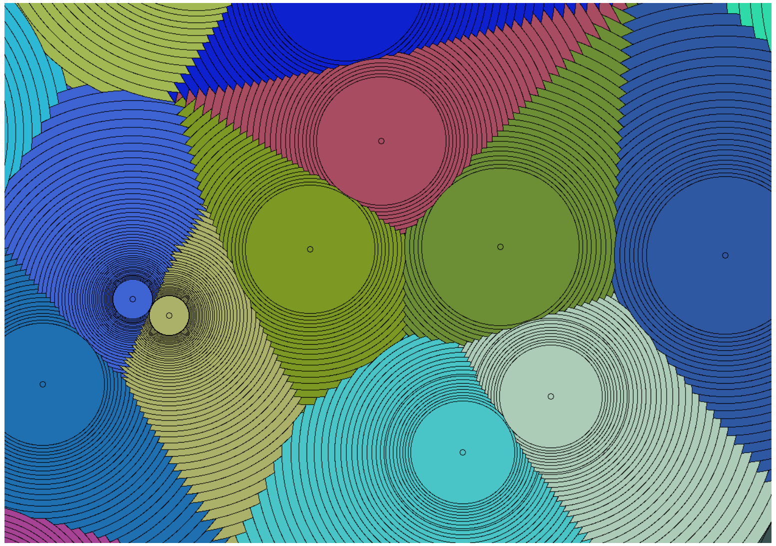
For a given ϵ, find a $(1+\epsilon)$ Aproximate Voronoi Diagram:

Partition of space into regions with one representative r_i per region, such that for any point q in region i, r_i is a $(1+\epsilon)$ nearest neighbor of q

Problem Statement:

For a given ϵ, find a $(1 + \epsilon)$ Aproximate Voronoi Diagram:

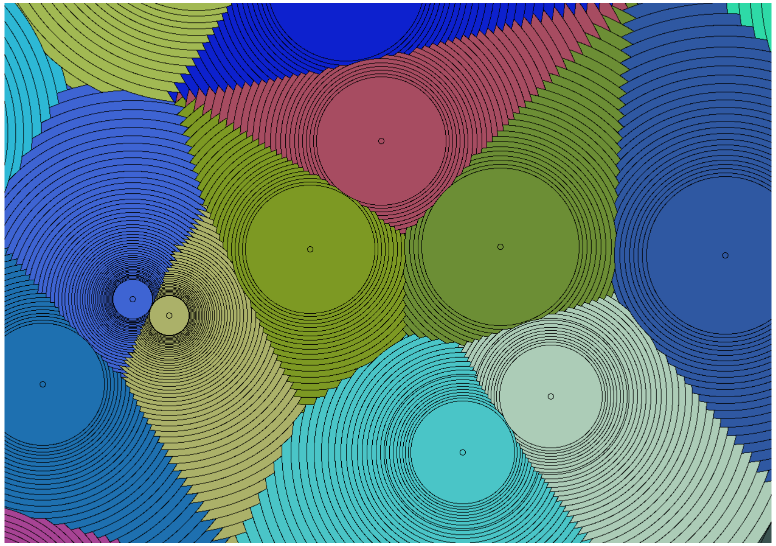
Partition of space into regions with one representative r_i per region, such that for any point q in region i, r_i is a $(1 + \epsilon)$ nearest neighbor of q



Problem Statement:

For a given ϵ, find a $(1+\epsilon)$ Aproximate Voronoi Diagram:

Partition of space into regions with one representative r_i per region, such that for any point q in region i, r_i is a $(1+\epsilon)$ nearest neighbor of q



Constraints:

- bounded construction time and space (complexity)
- Cover all space
- sub-linear (1+ϵ) NN queries

ANN using PLEB

Reduce $(1+\epsilon)$-ANN queries to target ball queries

ANN using PLEB

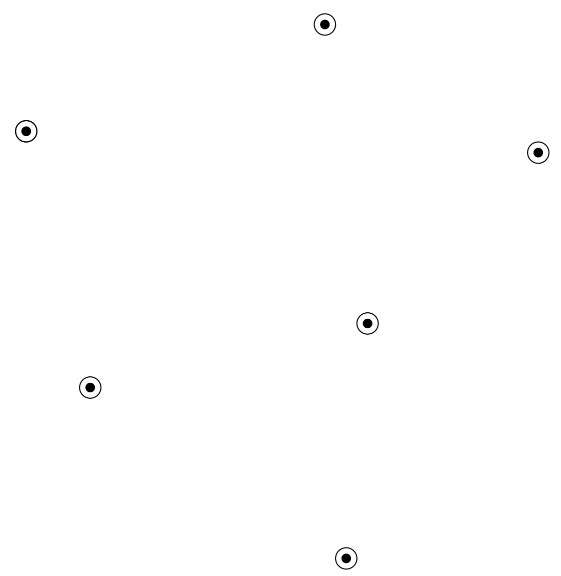
Reduce $(1+\epsilon)$-ANN queries to target ball queries

1) Construct balls of radius $(1+\epsilon)^i$ around each point, for $i = 1..\infty$

ANN using PLEB

Reduce $(1+\epsilon)$-ANN queries to target ball queries

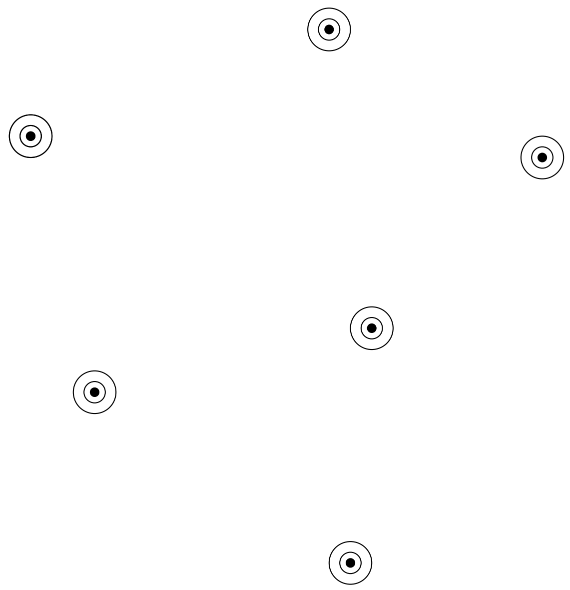
1) Construct balls of radius $(1+\epsilon)^i$ around each point, for $i = 1..\infty$



ANN using PLEB

Reduce $(1+\epsilon)$-ANN queries to target ball queries

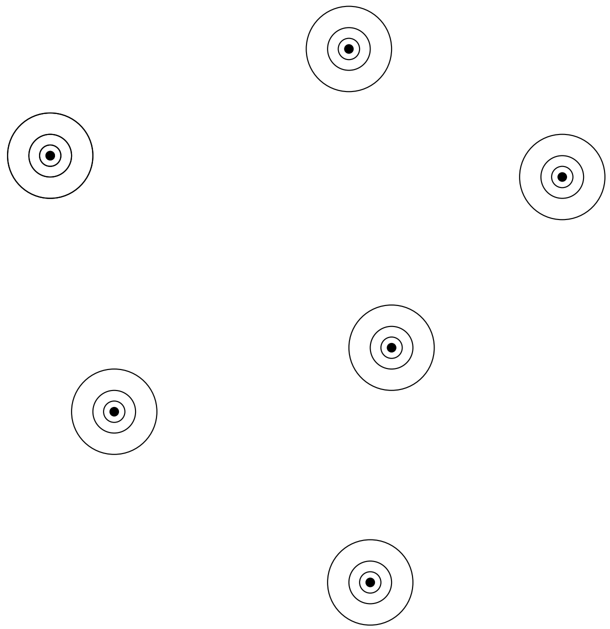
1) Construct balls of radius $(1+\epsilon)^i$ around each point, for $i = 1..\infty$



ANN using PLEB

Reduce $(1+\epsilon)$-ANN queries to target ball queries

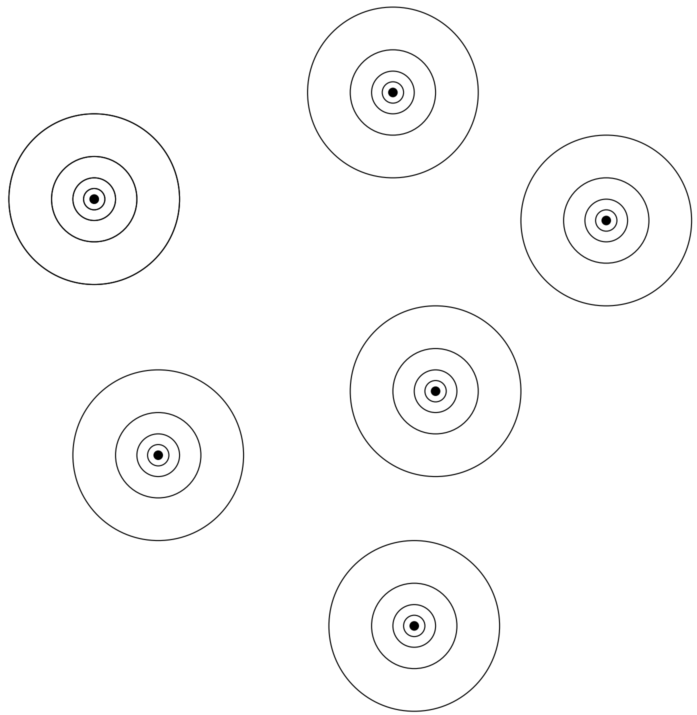
1) Construct balls of radius $(1+\epsilon)^i$ around each point, for $i = 1..\infty$



ANN using PLEB

Reduce $(1+\epsilon)$-ANN queries to target ball queries

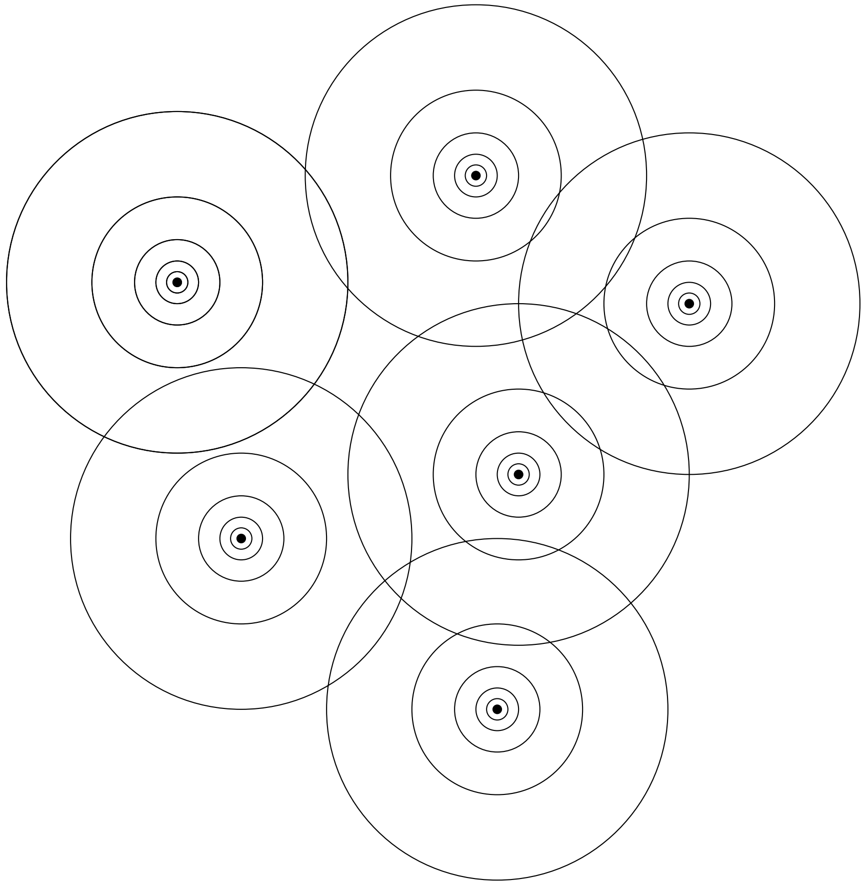
1) Construct balls of radius $(1+\epsilon)^i$ around each point, for $i = 1..\infty$



ANN using PLEB

Reduce $(1+\epsilon)$-ANN queries to target ball queries

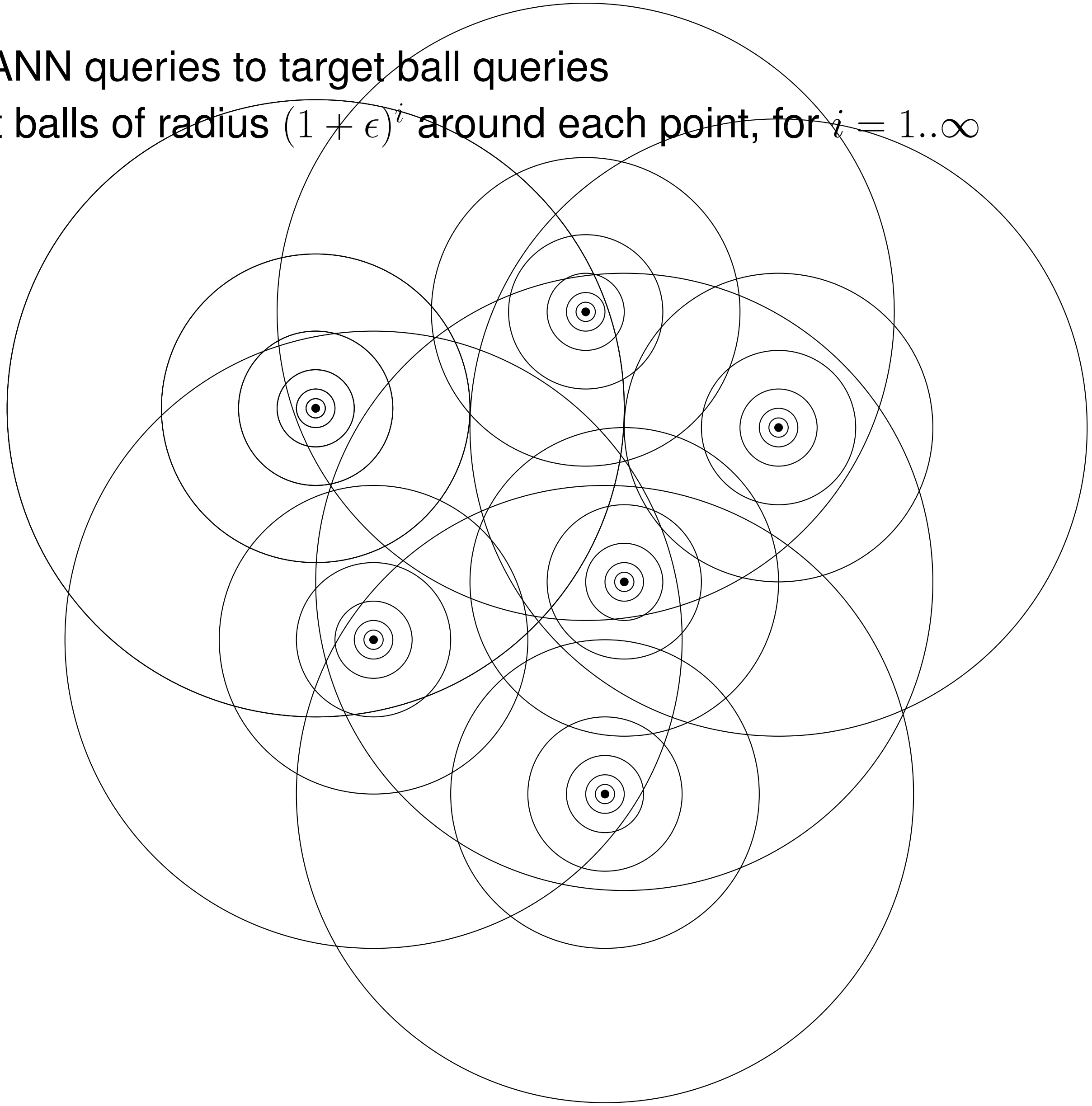
1) Construct balls of radius $(1+\epsilon)^i$ around each point, for $i = 1..\infty$



ANN using PLEB

Reduce $(1+\epsilon)$-ANN queries to target ball queries

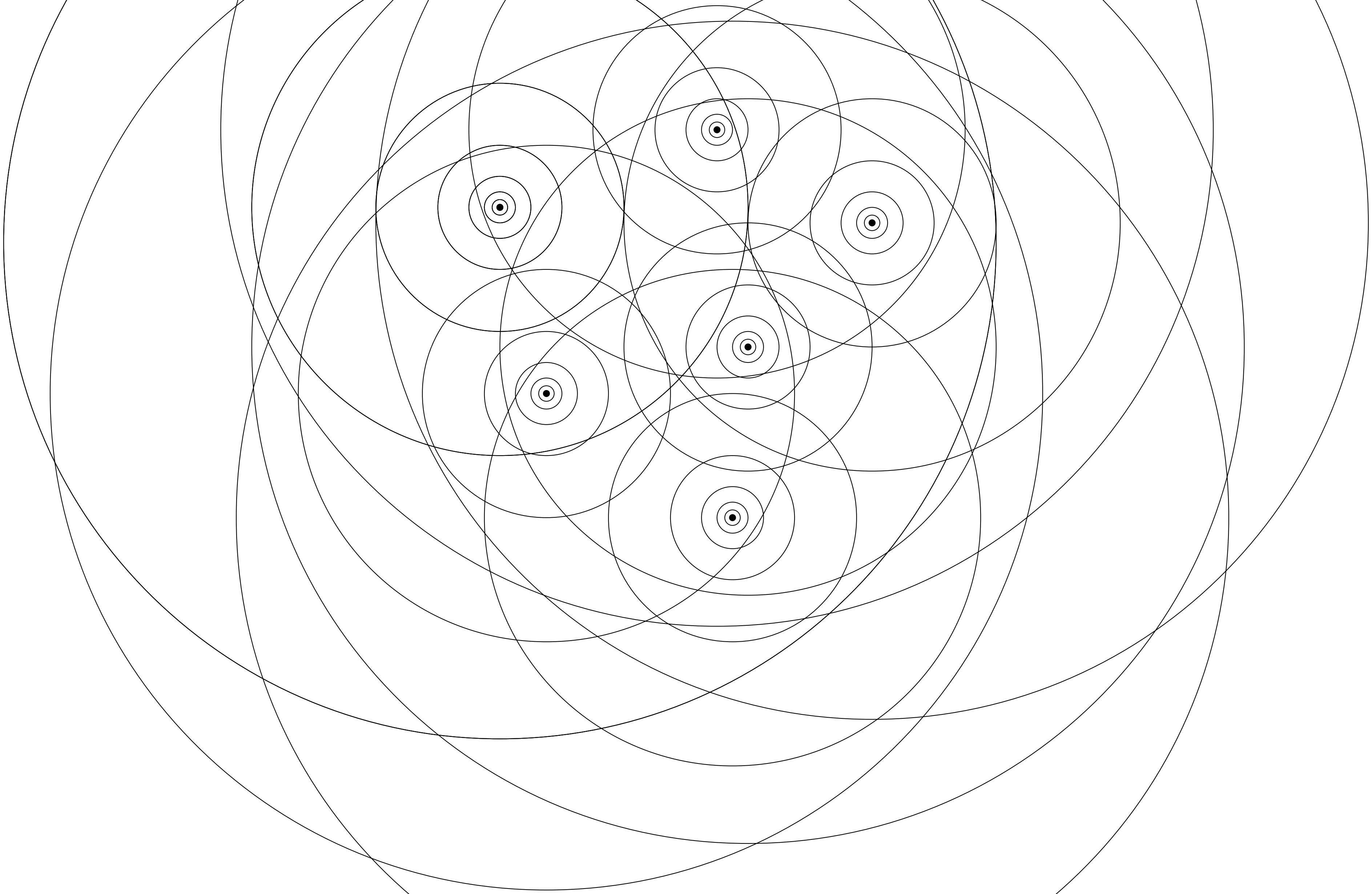
1) Construct balls of radius $(1+\epsilon)^i$ around each point, for $i = 1..\infty$



ANN using PLEB

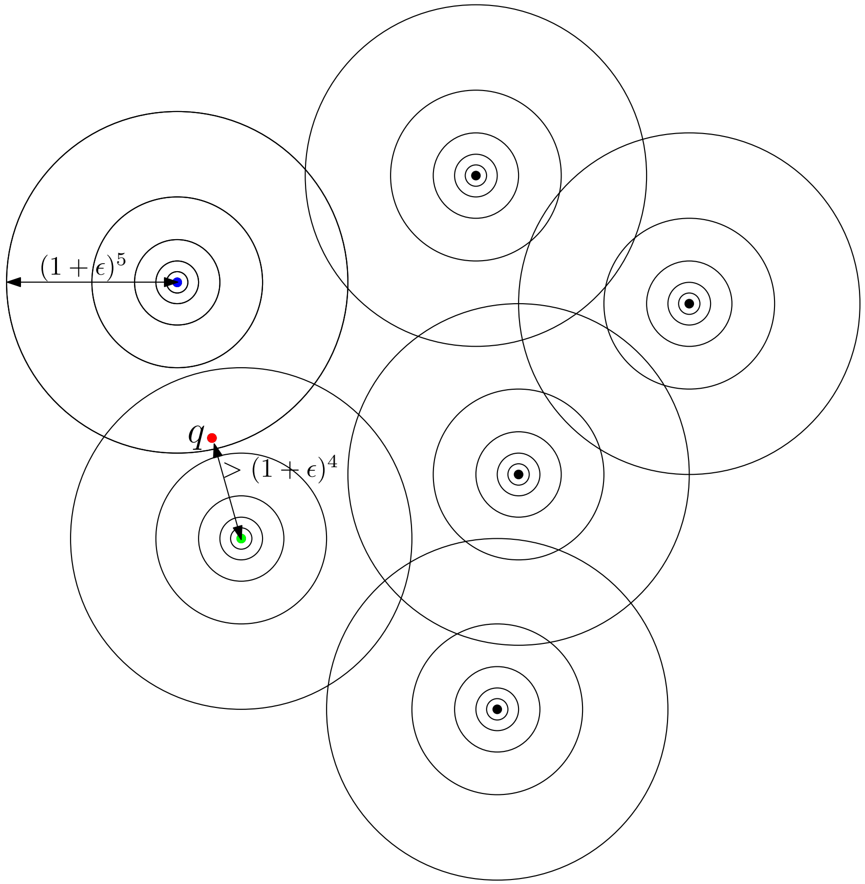
Reduce $(1+\epsilon)$-ANN queries to target ball queries

1) Construct balls of radius $(1+\epsilon)^i$ around each point, for $i = 1..\infty$



ANN using PLEB

Reduce $(1+\epsilon)$-ANN queries to target ball queries



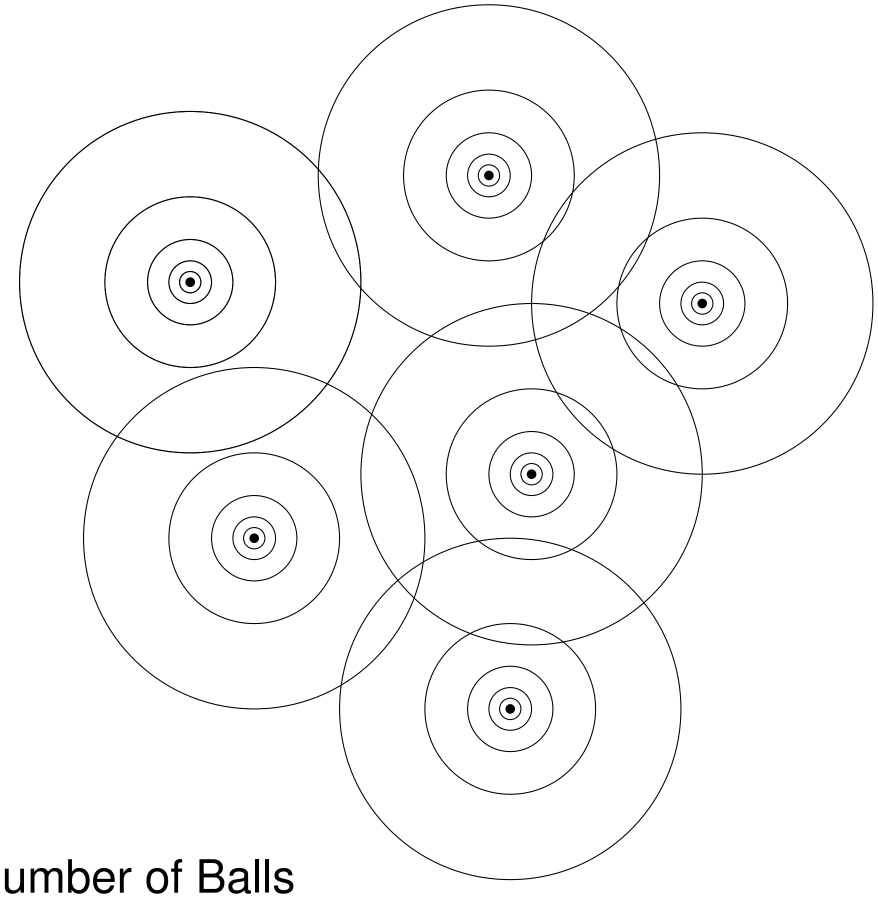
For any query point q, return the center p of the smallest ball that contains it:

$d(q,n) > (1+\epsilon)^{i-1}$, and $d(q,p) \leq (1+\epsilon)^i < (1+\epsilon) \cdot d(q,n)$

$\Longrightarrow$ always get a $(1+\epsilon)$-Nearest Neighbor

ANN using PLEB

Reduce $(1+\epsilon)$-ANN queries to target ball queries



Problems:

- Unbounded Number of Balls
- Not clear how to preform target ball queries efficiently
  - Partition the space into regions of influence

Bounding the number of balls

Intuition:

* For a given pair u and v, we only care if $\min d(q, \{u, v\}) \in [\frac{d(u,v)}{\epsilon+2}, \frac{d(u,v)}{\epsilon}]$

Bounding the number of balls

Intuition:

* For a given pair u and v, we only care if $\min d(q, \{u, v\}) \in [\frac{d(u,v)}{\epsilon+2}, \frac{d(u,v)}{\epsilon}]$

    - if $\min d(q, \{u, v\}) > \frac{d(u,v)}{\epsilon} \Longrightarrow$ either u or v are $(1 + \epsilon)$ NN

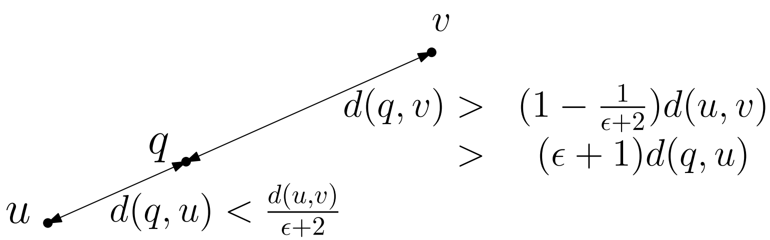
Bounding the number of balls

Intuition:

* For a given pair u and v, we only care if $\min d(q, \{u, v\}) \in [\frac{d(u,v)}{\epsilon+2}, \frac{d(u,v)}{\epsilon}]$

  - if $\min d(q, \{u, v\}) > \frac{d(u,v)}{\epsilon} \implies$ either u or v are $(1+\epsilon)$ NN

  - if $\min d(q, \{u, v\}) < \frac{d(u,v)}{\epsilon+2} \implies q$ has a unique $(1+\epsilon)$ NN



* Do not need to grow balls of radius smaller than $\frac{d(u,v)}{4}$ or larger than $\frac{2d(u,v)}{\epsilon}$

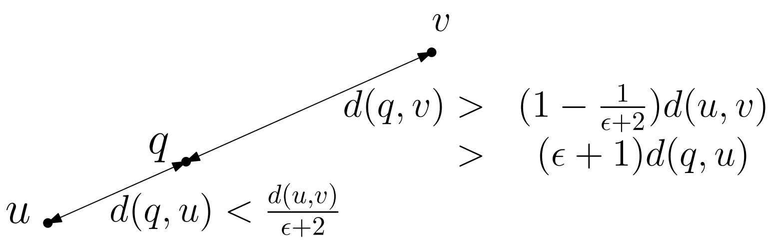
Bounding the number of balls

Intuition:

* For a given pair u and v, we only care if $\min d(q, \{u, v\}) \in [\frac{d(u,v)}{\epsilon+2}, \frac{d(u,v)}{\epsilon}]$

  - if $\min d(q, \{u, v\}) > \frac{d(u,v)}{\epsilon} \implies$ either u or v are $(1+\epsilon)$ NN

  - if $\min d(q, \{u, v\}) < \frac{d(u,v)}{\epsilon+2} \implies q$ has a unique $(1+\epsilon)$ NN



* Do not need to grow balls of radius smaller than $\frac{d(u,v)}{4}$ or larger than $\frac{2d(u,v)}{\epsilon}$

Method 1:

for every pair of points $\{u, v\}$, construct enough balls to cover $[\frac{d(u,v)}{4}, \frac{2d(u,v)}{\epsilon}]$ on u, v

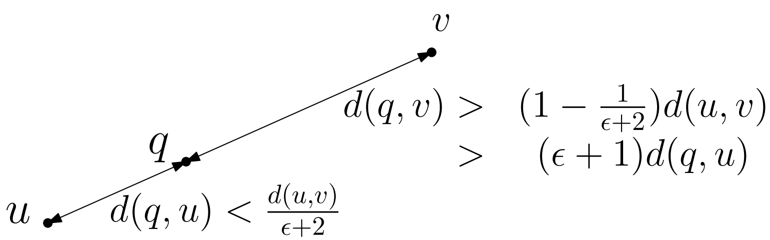
Bounding the number of balls

Intuition:

* For a given pair u and v, we only care if $\min d(q,\{u,v\}) \in [\frac{d(u,v)}{\epsilon+2}, \frac{d(u,v)}{\epsilon}]$

  - if $\min d(q,\{u,v\}) > \frac{d(u,v)}{\epsilon} \implies$ either u or v are $(1+\epsilon)$ NN

  - if $\min d(q,\{u,v\}) < \frac{d(u,v)}{\epsilon+2} \implies q$ has a unique $(1+\epsilon)$ NN



* Do not need to grow balls of radius smaller than $\frac{d(u,v)}{4}$ or larger than $\frac{2d(u,v)}{\epsilon}$

Method 1:

for every pair of points $\{u,v\}$, construct enough balls to cover $[\frac{d(u,v)}{4}, \frac{2d(u,v)}{\epsilon}]$ on u, v

Overall: $O(n^2 \log_{\epsilon+1}(\frac{2C}{\epsilon} - \frac{C}{4})) = O(n^2 \frac{\log(\frac{7C}{\epsilon})}{\log(\epsilon+1)}) = O(n^2 \frac{1}{\epsilon}\log(\frac{1}{\epsilon}))$ balls

Note: $\log(1+\epsilon) = \epsilon - \epsilon^2/2 + \epsilon^3/3 - .... = O(\epsilon)$ in most cases

Bounding the number of balls

Interval Near-Neighbor data structure

given a range of distances $[a, b]$, and a set of points P, answers:

1. $d_P(q) > b$
2. $d_P(q) < a$ with a witness
3. otherwise, finds a point $p \in P$, s.t. $d_P(q) \leq d(p, q) \leq (1 + \epsilon)d_P(q)$

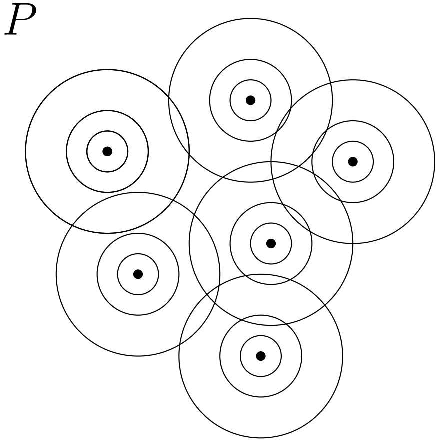
Bounding the number of balls

Interval Near-Neighbor data structure

given a range of distances $[a, b]$, and a set of points P, answers:

1. $d_P(q) > b$
2. $d_P(q) < a$ with a witness
3. otherwise, finds a point $p \in P$, s.t. $d_P(q) \leq d(p, q) \leq (1 + \epsilon)d_P(q)$

Can be realized by a set of balls of radius $a(1 + \epsilon)^i$ for $i = 0...M - 1$, where $M = \lceil \log_{1+\epsilon}(b/a) \rceil$ and a ball of radius b around every point in P



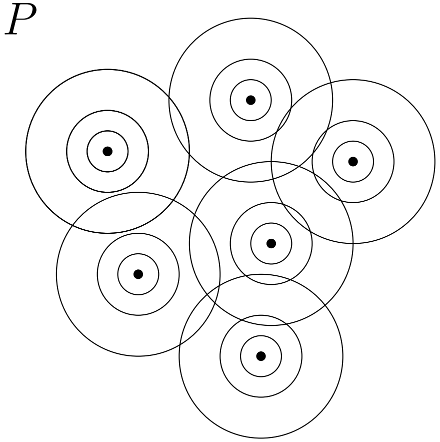
Bounding the number of balls

Interval Near-Neighbor data structure

given a range of distances $[a, b]$, and a set of points P, answers:

1. $d_P(q) > b$
2. $d_P(q) < a$ with a witness
3. otherwise, finds a point $p \in P$, s.t. $d_P(q) \leq d(p, q) \leq (1+\epsilon)d_P(q)$

Can be realized by a set of balls of radius $a(1+\epsilon)^i$ for $i = 0...M-1$, where $M = \lceil \log_{1+\epsilon}(b/a) \rceil$ and a ball of radius b around every point in P



Contains $O(n\frac{1}{\epsilon}\log(b/a))$ balls. Takes at most 2 target ball queries if 1 or 2 hold, and

* $O(\log(M)) = O(\log \frac{\log(b/a)}{\epsilon})$ otherwise

Bounding the number of balls

A data structure to answer $(1+\epsilon)$-ANN queries on general points

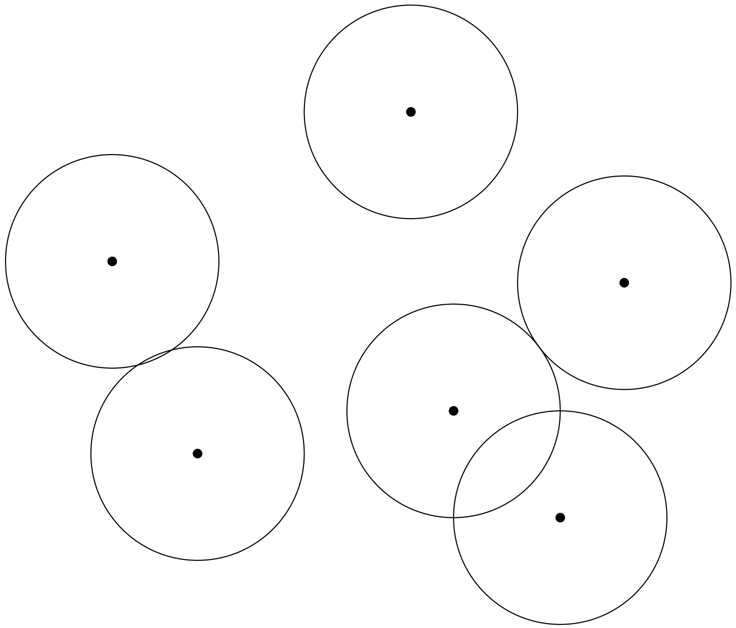
Build a tree, with an Interval Near Neighbor structure associated with each node

(Sariel Har-Peled: *A Replacement for Voronoi Diagrams of Near Linear Size*. FOCS 2001: 94-103)

Bounding the number of balls

A data structure to answer $(1 + \epsilon)$-ANN queries on general points

Build a tree, with an Interval Near Neighbor structure associated with each node



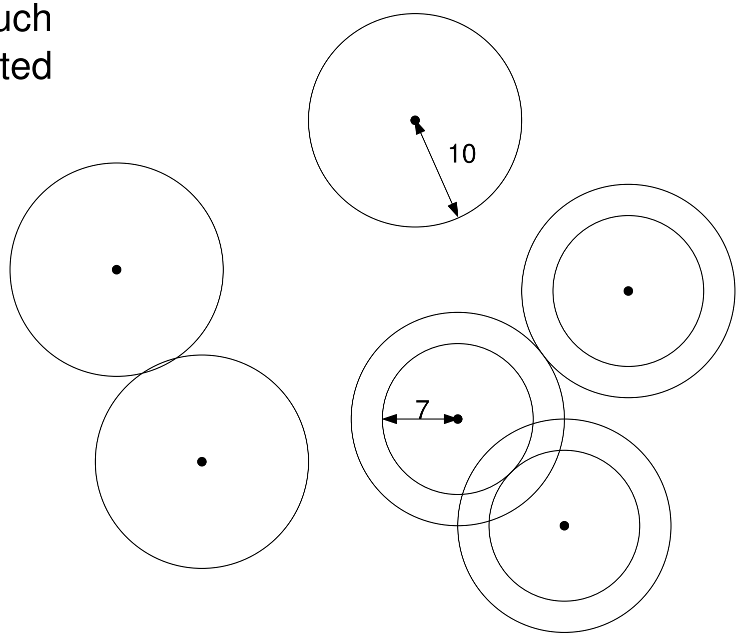
(Sariel Har-Peled: *A Replacement for Voronoi Diagrams of Near Linear Size*. FOCS 2001: 94-103)

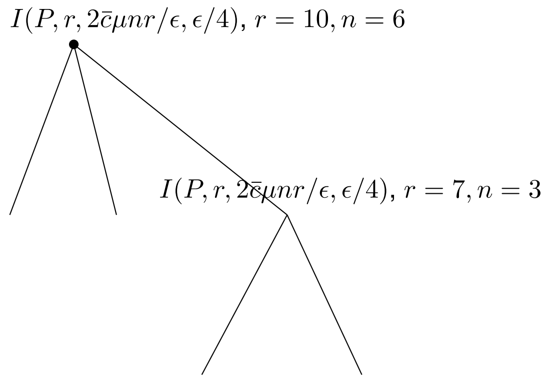
Bounding the number of balls

A data structure to answer $(1+\epsilon)$-ANN queries on general points

Build a tree, with an Interval Near Neighbor structure associated with each node

Recursively find $\min r$ such that there are $\lceil n/2 \rceil$ connected components





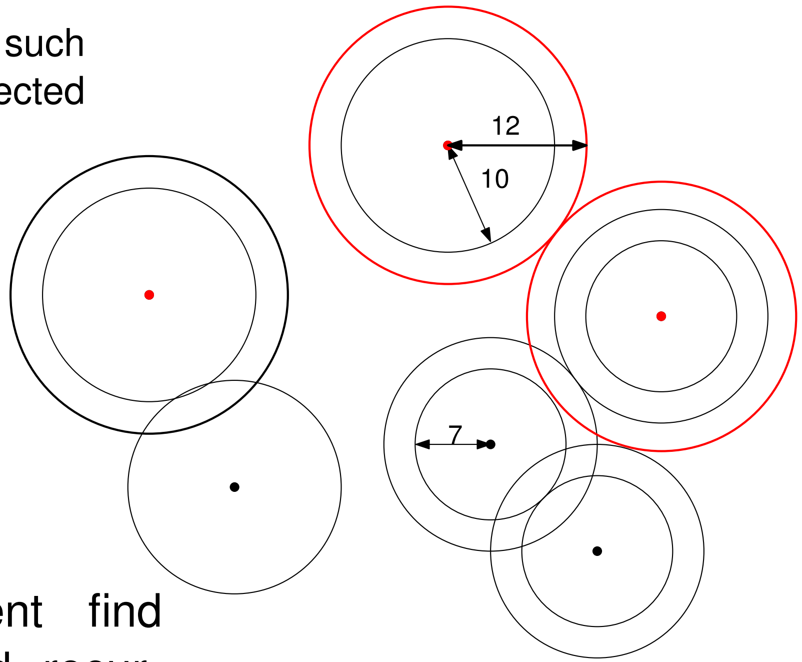
(Sariel Har-Peled: *A Replacement for Voronoi Diagrams of Near Linear Size*. FOCS 2001: 94-103)

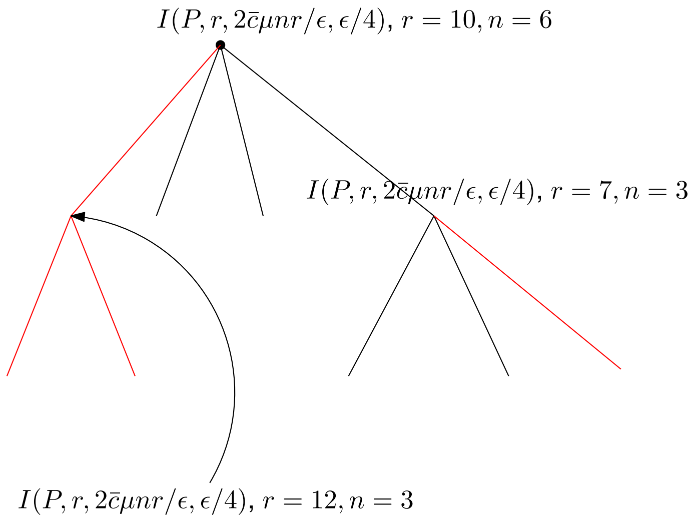
Bounding the number of balls

A data structure to answer $(1+\epsilon)$-ANN queries on general points

Build a tree, with an Interval Near Neighbor structure associated with each node

Recursively find $\min r$ such that there are $\lceil n/2 \rceil$ connected components





For each component find a representative and recursively build the outer tree

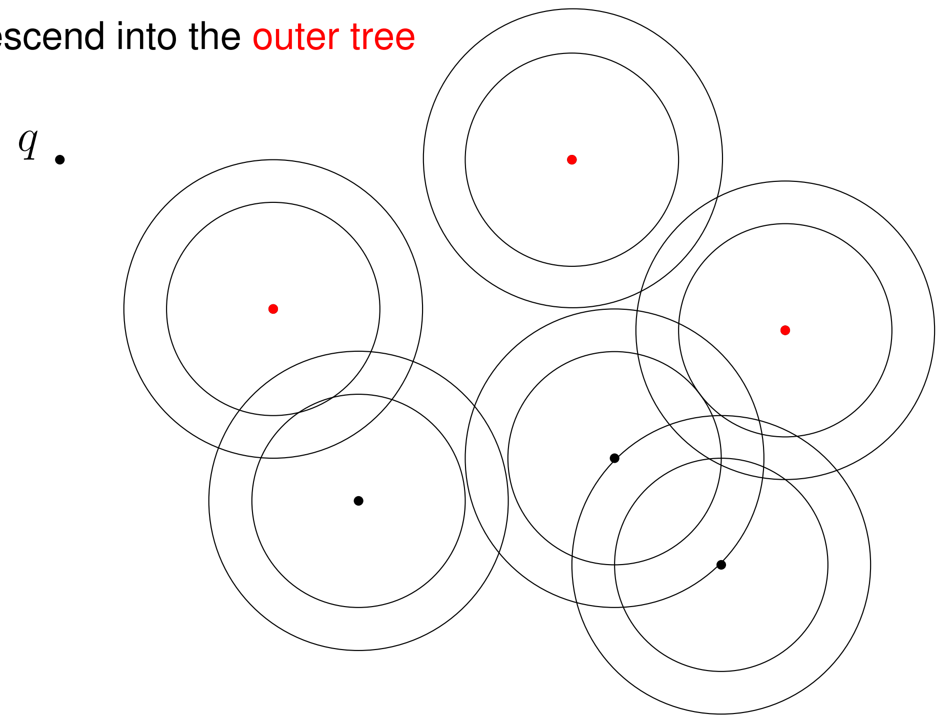
(Sariel Har-Peled: *A Replacement for Voronoi Diagrams of Near Linear Size*. FOCS 2001: 94-103)

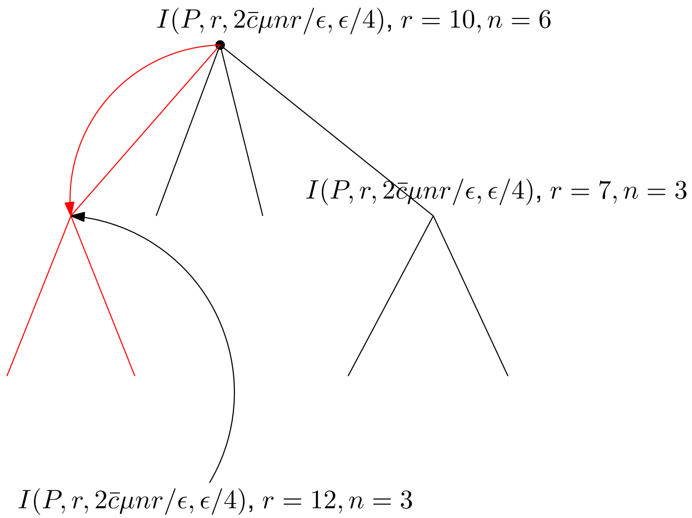
Bounding the number of balls

A data structure to answer $(1+\epsilon)$-ANN queries on general points

Given a query point q:

1) q is outside R descend into the outer tree





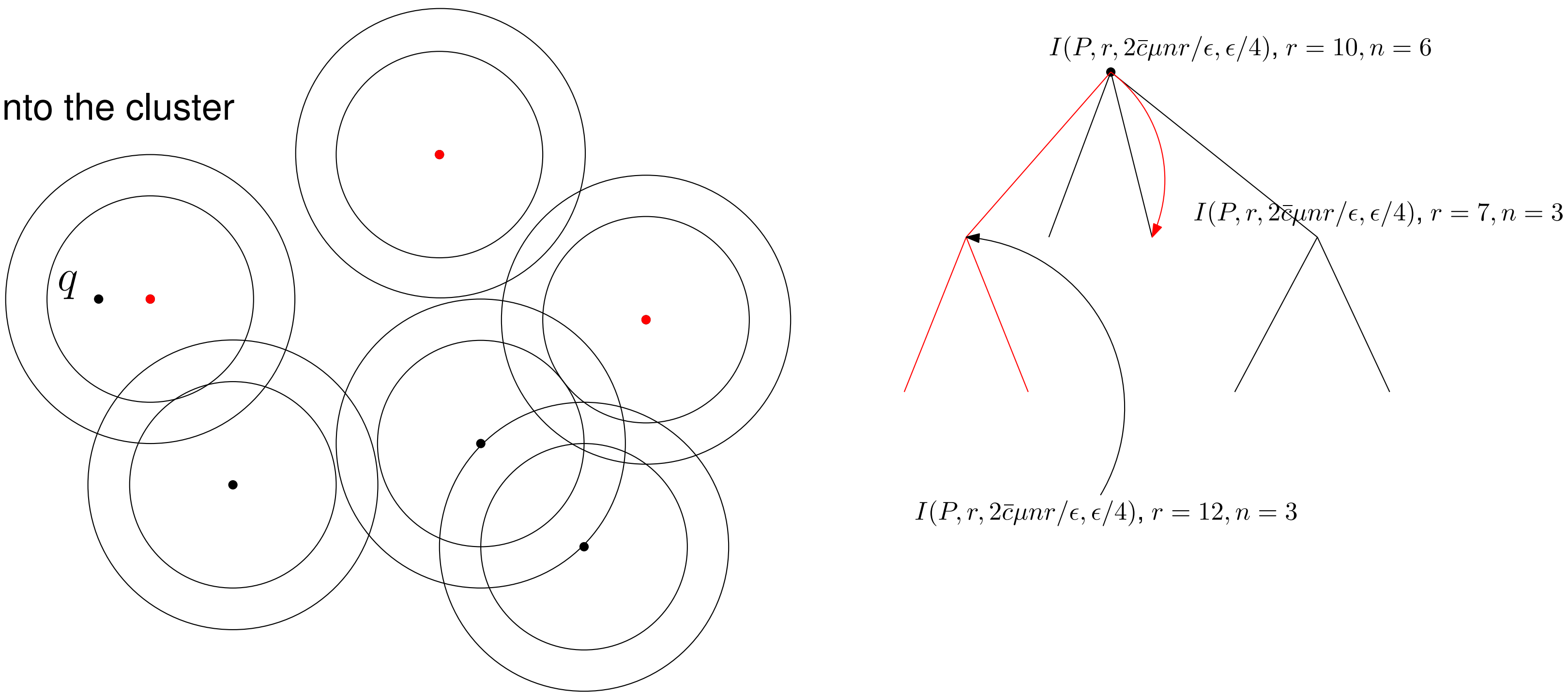
(Sariel Har-Peled: *A Replacement for Voronoi Diagrams of Near Linear Size*. FOCS 2001: 94-103)

Bounding the number of balls

A data structure to answer $(1+\epsilon)$-ANN queries on general points

Given a query point q:

2) if q is inside r descend into the cluster



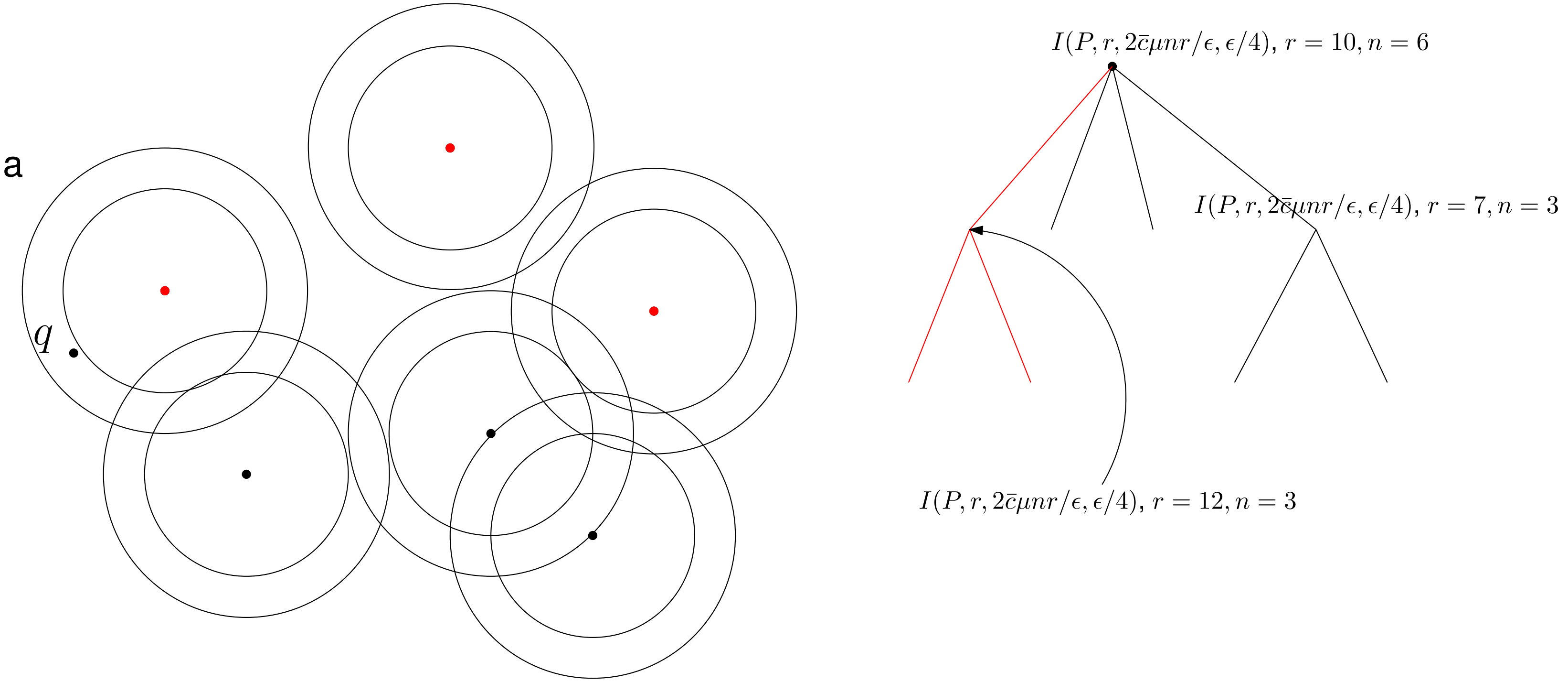
(Sariel Har-Peled: *A Replacement for Voronoi Diagrams of Near Linear Size*. FOCS 2001: 94-103)

Bounding the number of balls

A data structure to answer $(1+\epsilon)$-ANN queries on general points

Given a query point q:

3) otherwise I will return a $(1+\frac{\epsilon}{4})$-NN

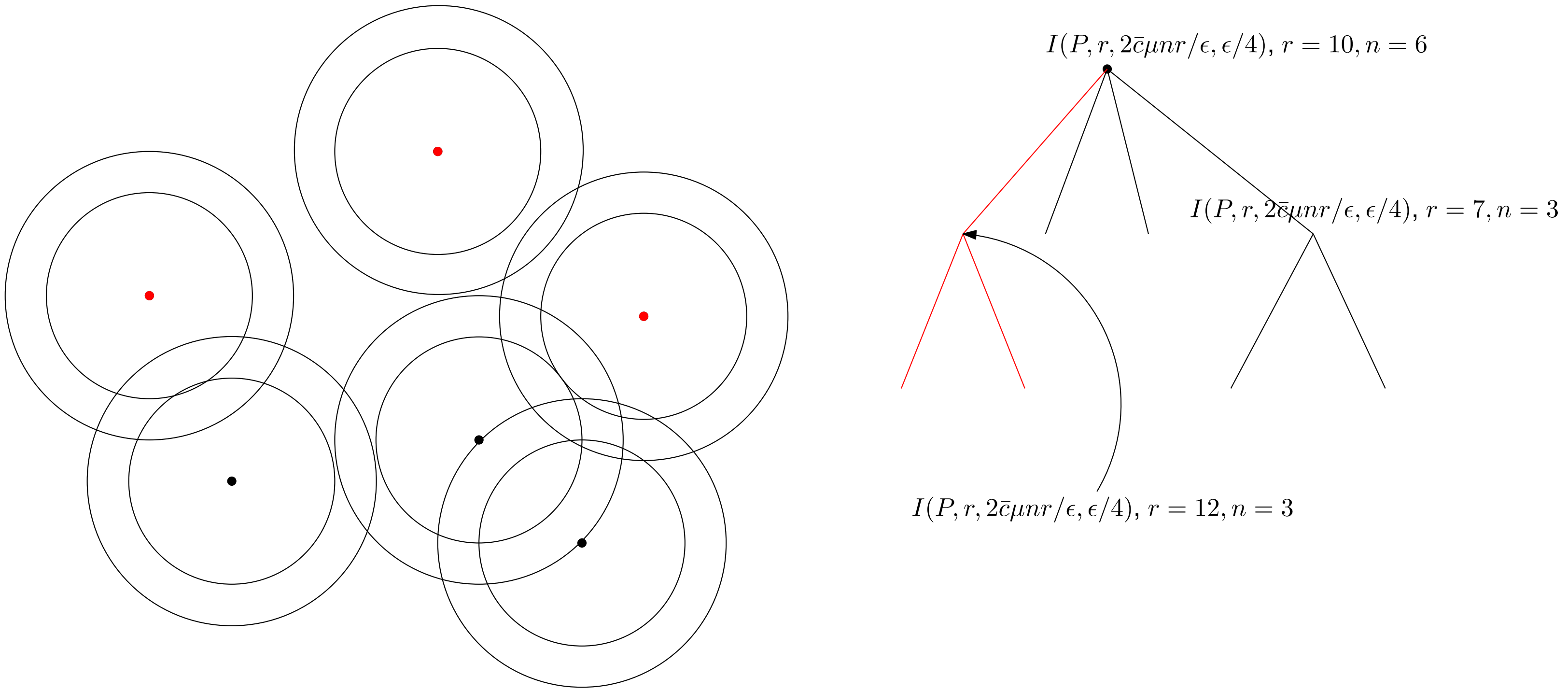


(Sariel Har-Peled: *A Replacement for Voronoi Diagrams of Near Linear Size*. FOCS 2001: 94-103)

Bounding the number of balls

A data structure to answer $(1+\epsilon)$-ANN queries on general points

Given a query point q:



Because of rounding up, after each step, continue on set containing $\leq n/2 + 1$ points
$\Longrightarrow$ number of steps $\leq \log_{3/2} n$

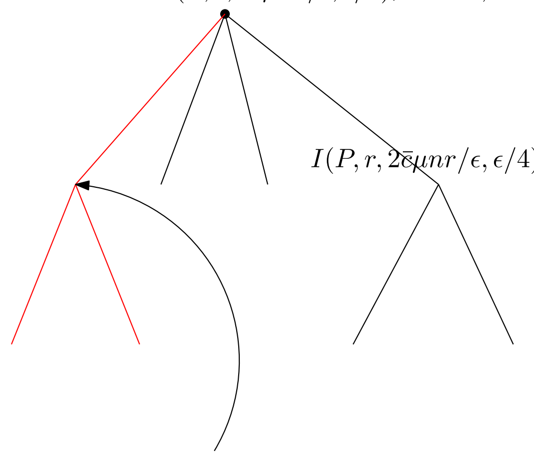
(Sariel Har-Peled: *A Replacement for Voronoi Diagrams of Near Linear Size*. FOCS 2001: 94-103)

Bounding the number of balls

1) q is outside R descend into the outer tree

2) if q is inside r descend into the cluster

3) otherwise I will return a $(1+\frac{\epsilon}{4})$-NN



Note that:

- last step is always 3)
- no error is incurred in 2)
- diameter of a cluster $\leq 2nr \implies$ error in 1) is at most $(1+\frac{\epsilon}{\bar{c}\mu})$

Thus, overall error is bounded by:

$$(1+\frac{\epsilon}{4})\prod_{i=1}^{\log_{3/2} n}(1+\frac{\epsilon}{\bar{c}\mu}) \leq \exp(\frac{\epsilon}{4})\prod_{i=1}^{\log_{3/2} n}\exp(\frac{\epsilon}{\bar{c}\mu}) \leq \exp\left(\frac{\epsilon}{4}+\sum_{i=1}^{\log_{3/2} n}\frac{\epsilon}{\bar{c}\mu}\right) \leq \exp(\epsilon/2) \leq (1+\epsilon)$$

if $\mu = \lceil \log_{3/2} n \rceil$, $\bar{c} = 4$ and $\epsilon < 1$

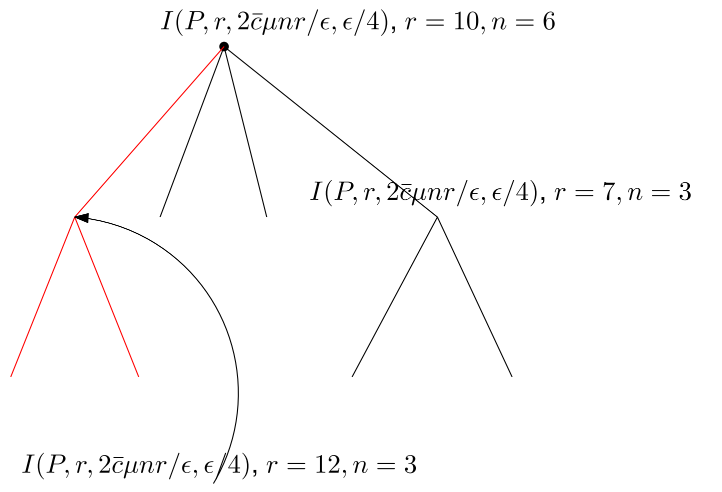
Bounding the number of balls

Overall Number of Balls:

Since

- the depth of the tree is at most $\log_{3/2} n$
- each node ν has $I(P_\nu, r, 2\bar{c}\mu nr/\epsilon, \epsilon/4)$ with $M = n \log n$ balls

we get an immediate bound of
$O(M \log M) = O(n \log(n) \log(n \log n)) = O(n \log^2 n)$



(Sariel Har-Peled: *A Replacement for Voronoi Diagrams of Near Linear Size*. FOCS 2001: 94-103)

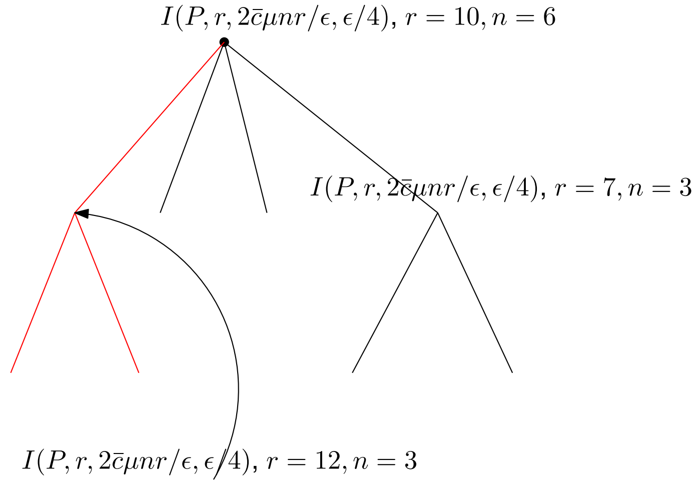
Bounding the number of balls

Overall Number of Balls:

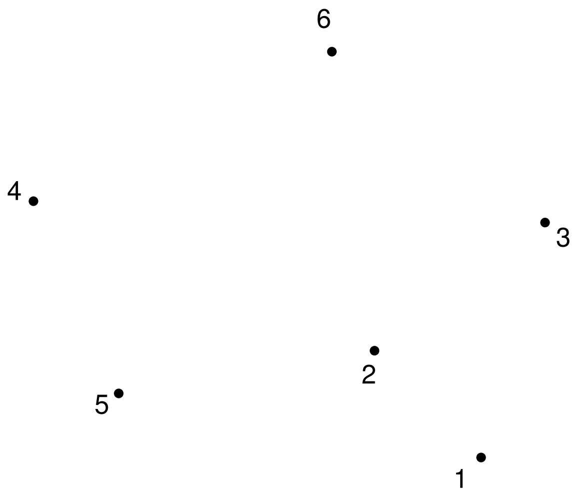
Since

- the depth of the tree is at most $\log_{3/2} n$
- each node ν has $I(P_\nu, r, 2\bar{c}\mu nr/\epsilon, \epsilon/4)$ with $M = n \log n$ balls

we get an immediate bound of
$O(M \log M) = O(n \log(n) \log(n \log n)) = O(n \log^2 n)$



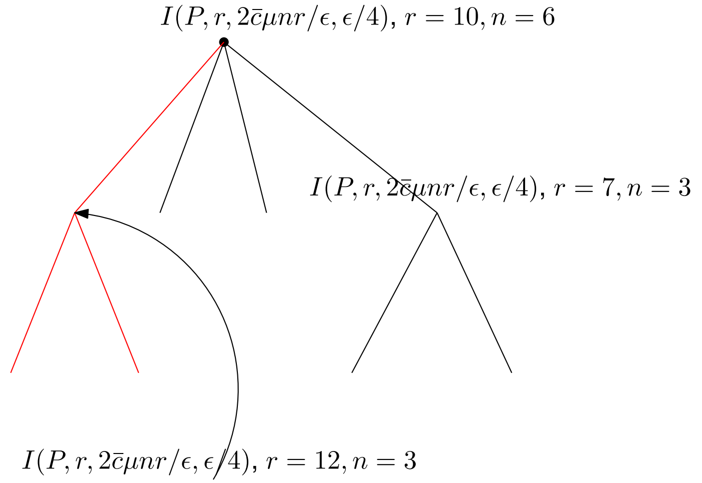
However, can achieve $O(n \log n)$ by considering the connection with the *Cluster Tree*



S. Sen, N. Sharma, Y. Sabharwal: *Nearest Neighbors Search using Point Location in Balls with applications to approximate Voronoi Decompositions* Journal of Computer and System Sciences, Volume 72(6) , September 2006, Pages 955-977.

Bounding the number of balls

Overall Number of Balls:

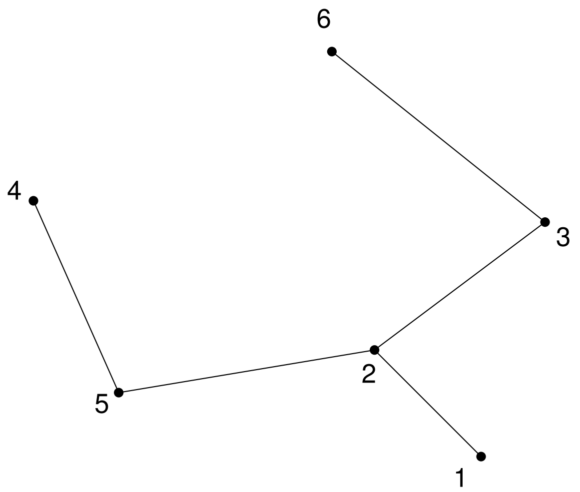


Since

- the depth of the tree is at most $\log_{3/2} n$
- each node ν has $I(P_\nu, r, 2\bar{c}\mu nr/\epsilon, \epsilon/4)$ with $M = n \log n$ balls

we get an immediate bound of
$O(M \log M) = O(n \log(n) \log(n \log n)) = O(n \log^2 n)$

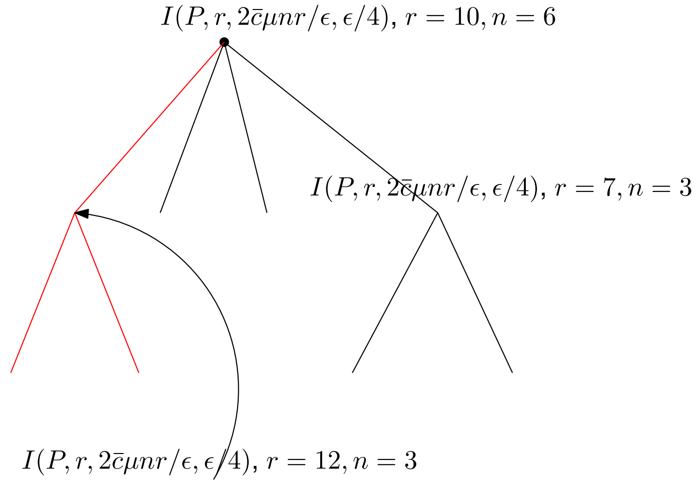
However, can achieve $O(n \log n)$ by considering the connection with the *Cluster Tree*



S. Sen, N. Sharma, Y. Sabharwal: *Nearest Neighbors Search using Point Location in Balls with applications to approximate Voronoi Decompositions* Journal of Computer and System Sciences, Volume 72(6) , September 2006, Pages 955-977.

Bounding the number of balls

Overall Number of Balls:

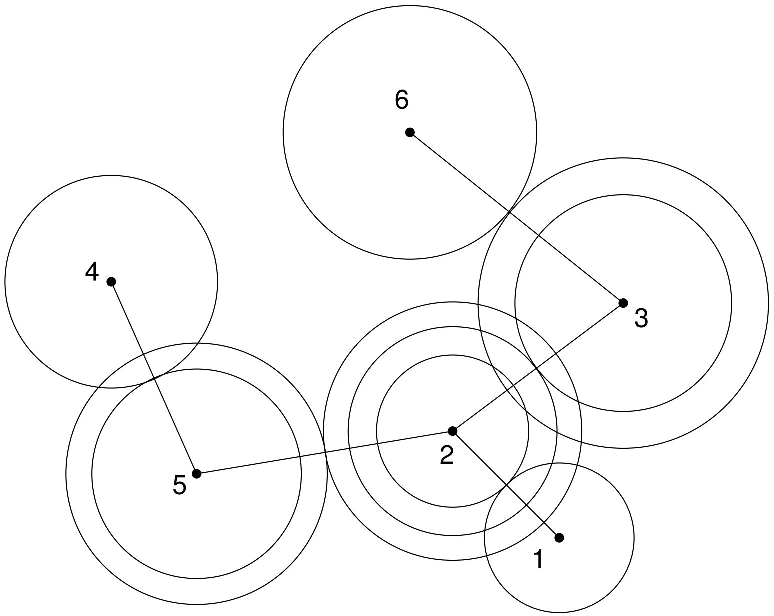


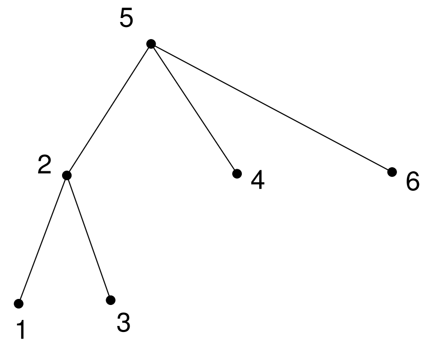
Since

- the depth of the tree is at most $\log_{3/2} n$
- each node ν has $I(P_\nu, r, 2\bar{c}\mu nr/\epsilon, \epsilon/4)$ with $M = n \log n$ balls

we get an immediate bound of
$O(M \log M) = O(n \log(n) \log(n \log n)) = O(n \log^2 n)$

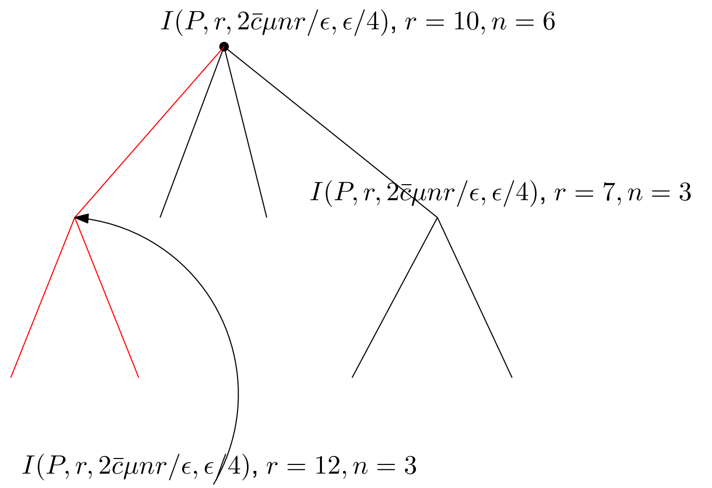
However, can achieve $O(n \log n)$ by considering the connection with the *Cluster Tree*





Bounding the number of balls

Overall Number of Balls:

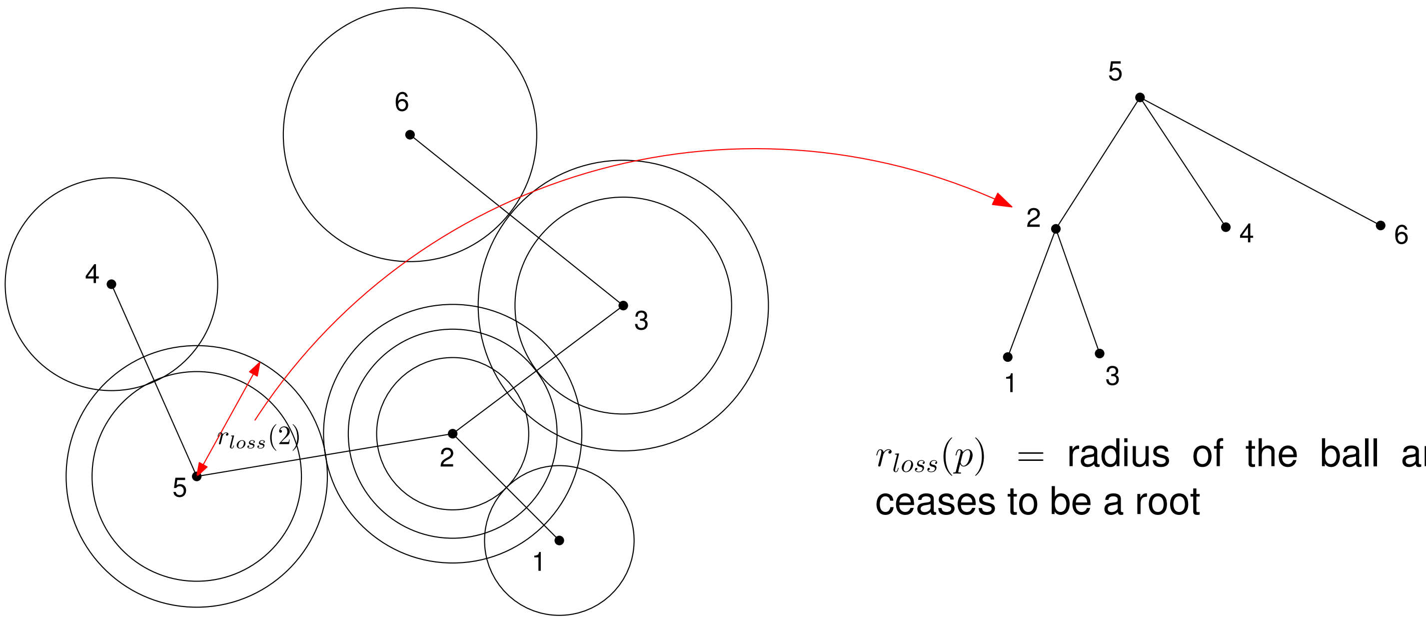


Since

- the depth of the tree is at most $\log_{3/2} n$
- each node ν has $I(P_\nu, r, 2\bar{c}\mu nr/\epsilon, \epsilon/4)$ with $M = n \log n$ balls

we get an immediate bound of
$O(M \log M) = O(n \log(n) \log(n \log n)) = O(n \log^2 n)$

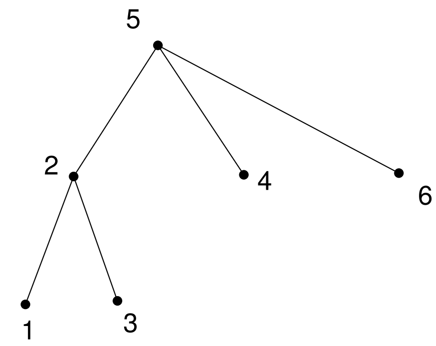
However, can achieve $O(n \log n)$ by considering the connection with the *Cluster Tree*

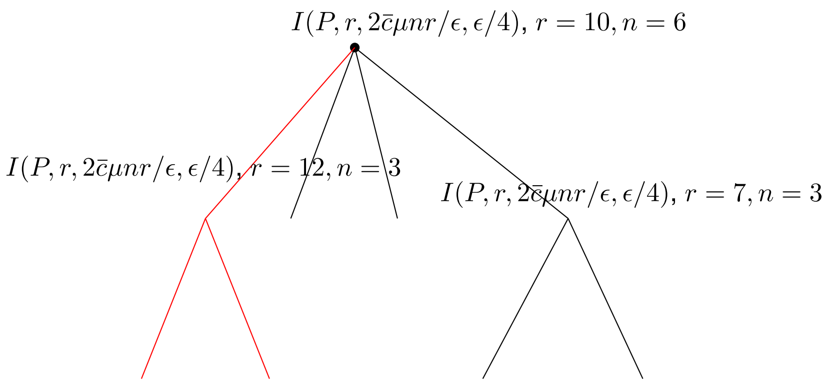


$r_{loss}(p)$ = radius of the ball around p, when p ceases to be a root

Bounding the number of balls

Apart from the outer trees, going down the $(1+\epsilon)$ ANN tree is equivalent to disconnecting edges of the MST tree





The subtrees of a node are disjoint in edges

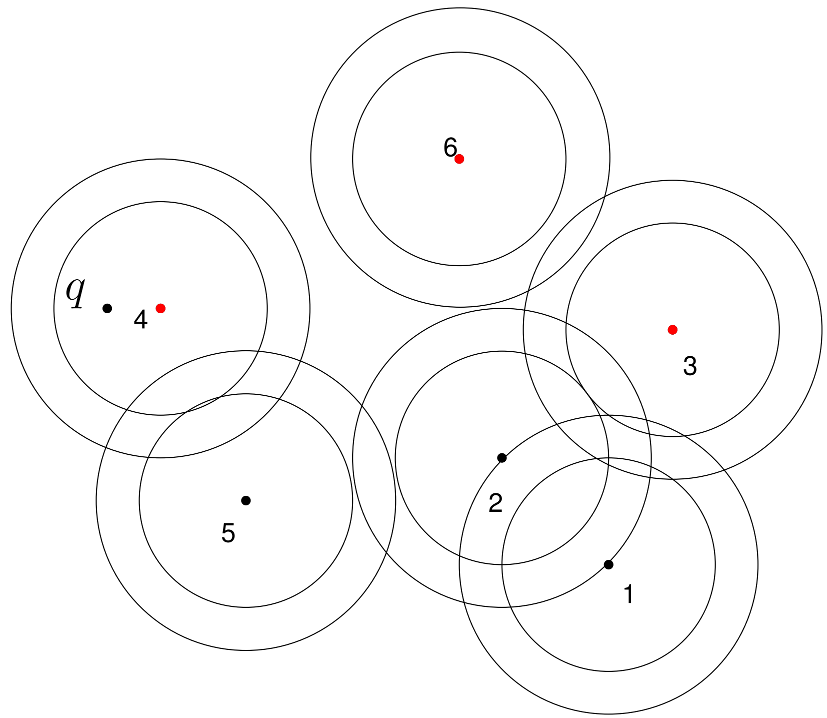
$\Longrightarrow$ can charge at least 1 edge to each child.

Namely: if n_ν is the number of children of ν

$|P_\nu| = O(n_\nu)$ and $\sum_{\nu \in D} n_\nu = O(n)$

Thus, total number of balls:

$$\sum_{\nu \in D} O\left(\frac{n_\nu}{\epsilon} \log \frac{\mu n_\nu}{\epsilon}\right) = O\left(\frac{n}{\epsilon} \log \frac{n \log n}{\epsilon}\right) = O\left(\frac{n}{\epsilon} \log \frac{n}{\epsilon}\right)$$



Construction Time

Construction time will be dominated by constructing the tree D

Can be constructed directly from the cluster tree but this takes time $O(n^2)$ time

Construction Time

Construction time will be dominated by constructing the tree D

Can be constructed directly from the cluster tree but this takes time $O(n^2)$ time

In $\mathbb{R}^d$ the cluster tree can be $(2n-2)$-approximated by a HST in $O(n \log n)$ time:

1. construct a 2-spanner of P of size $O(n)$ in $O(n \log n)$ time
2. construct an HST that $(n-1)$ approximates the spanner in $O(n \log n)$ time

Construction Time

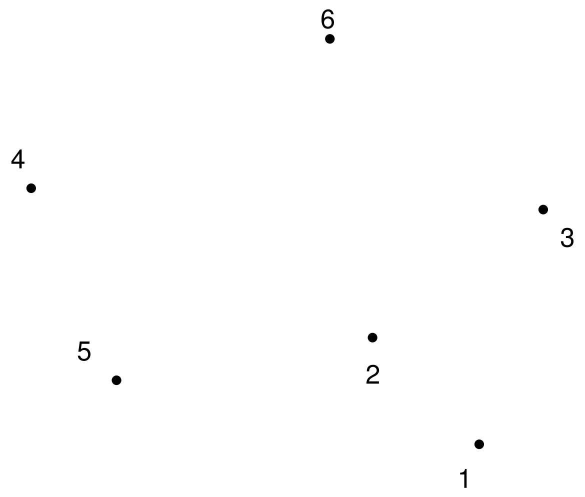
Construction time will be dominated by constructing the tree D

Can be constructed directly from the cluster tree but this takes time $O(n^2)$ time

In $\mathbb{R}^d$ the cluster tree can be $(2n-2)$-approximated by a HST in $O(n \log n)$ time:

1. construct a 2-spanner of P of size $O(n)$ in $O(n \log n)$ time
2. construct an HST that $(n-1)$ approximates the spanner in $O(n \log n)$ time

Only possible in $\mathbb{R}^d$, in general **no** HST can be computed in subquadratic time



Construction Time

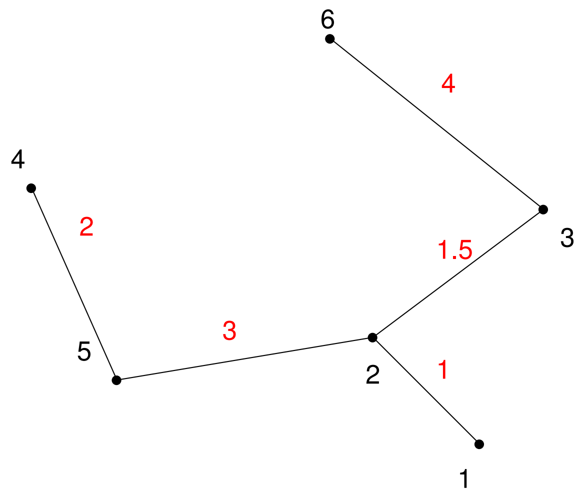
Construction time will be dominated by constructing the tree D

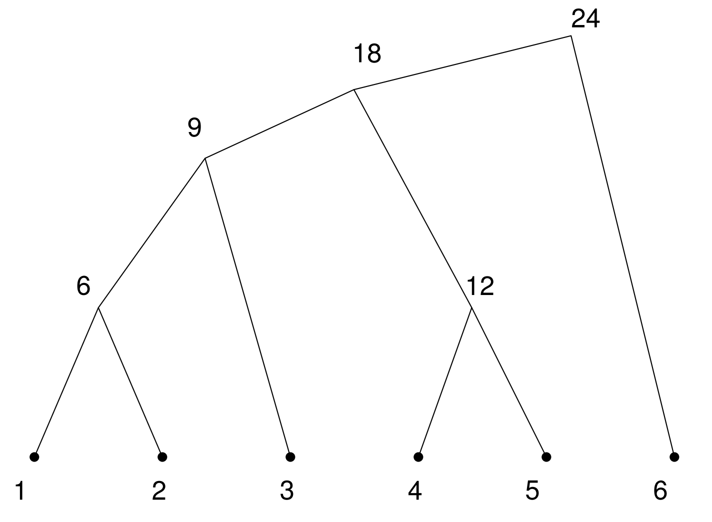
Can be constructed directly from the cluster tree but this takes time $O(n^2)$ time

In $\mathbb{R}^d$ the cluster tree can be $(2n-2)$-approximated by a HST in $O(n \log n)$ time:

1. construct a 2-spanner of P of size $O(n)$ in $O(n \log n)$ time
2. construct an HST that $(n-1)$ approximates the spanner in $O(n \log n)$ time

Only possible in $\mathbb{R}^d$, in general **no** HST can be computed in subquadratic time



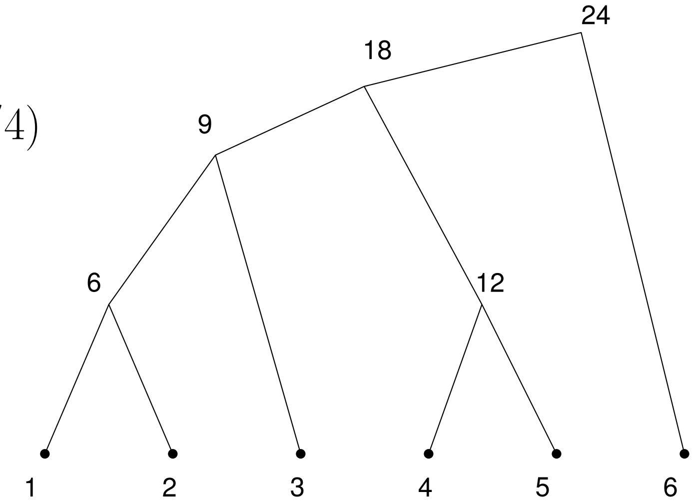


Construction Time

In $\mathbb{R}^d$ the cluster tree can be $(2n-2)$-approximated by a HST in $O(n \log n)$ time:

1. construct a 2-spanner of P of size $O(n)$ in $O(n \log n)$ time
2. construct an HST that $(n-1)$ approximates the spanner in $O(n \log n)$ time

To compensate for the approximation factor, grow more balls:
Instead of $I(P, r, 2\bar{c}\mu nr/\epsilon, \epsilon/4)$ construct $I(P, r/(2n), 2\bar{c}\mu nr/\epsilon, \epsilon/4)$

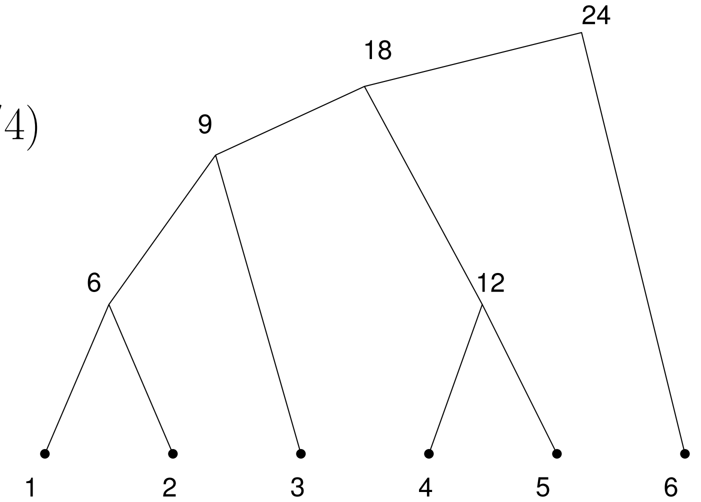


Construction Time

In $\mathbb{R}^d$ the cluster tree can be $(2n-2)$-approximated by a HST in $O(n \log n)$ time:

1. construct a 2-spanner of P of size $O(n)$ in $O(n \log n)$ time
2. construct an HST that $(n-1)$ approximates the spanner in $O(n \log n)$ time

To compensate for the approximation factor, grow more balls:
Instead of $I(P, r, 2\bar{c}\mu nr/\epsilon, \epsilon/4)$ construct $I(P, r/(2n), 2\bar{c}\mu nr/\epsilon, \epsilon/4)$



Instead of $O(\frac{n}{\epsilon} \log \frac{b}{a}) = O(\frac{n}{\epsilon} \log n)$ will have:

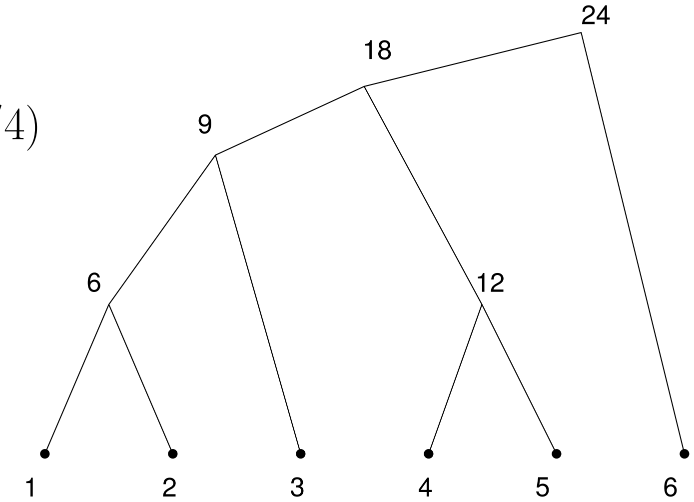
$O(\frac{n}{\epsilon} \log \frac{nr}{\frac{r}{n}}) = O(\frac{n}{\epsilon} \log n^2) = O(\frac{n}{\epsilon} \log n)$ balls at every node

Construction Time

In $\mathbb{R}^d$ the cluster tree can be $(2n-2)$-approximated by a HST in $O(n \log n)$ time:

1. construct a 2-spanner of P of size $O(n)$ in $O(n \log n)$ time
2. construct an HST that $(n-1)$ approximates the spanner in $O(n \log n)$ time

To compensate for the approximation factor, grow more balls:
Instead of $I(P, r, 2\bar{c}\mu nr/\epsilon, \epsilon/4)$ construct $I(P, r/(2n), 2\bar{c}\mu nr/\epsilon, \epsilon/4)$



Instead of $O(\frac{n}{\epsilon} \log \frac{b}{a}) = O(\frac{n}{\epsilon} \log n)$ will have:
$O(\frac{n}{\epsilon} \log \frac{nr}{\frac{r}{n}}) = O(\frac{n}{\epsilon} \log n^2) = O(\frac{n}{\epsilon} \log n)$ balls at every node

Same asymptotic space and time complexity

Answering ANN queries

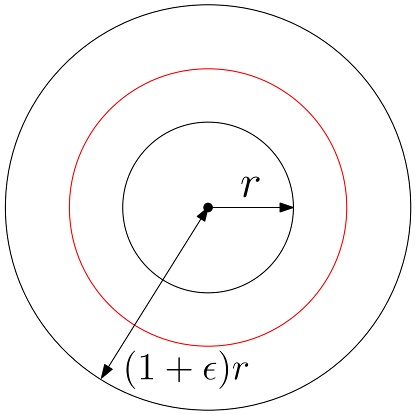
Haven't made our life easier, since answering target ball queries is a difficult problem

Answering ANN queries

Haven't made our life easier, since answering target ball queries is a difficult problem

Don't need exact balls

$(1+\epsilon)$ ball



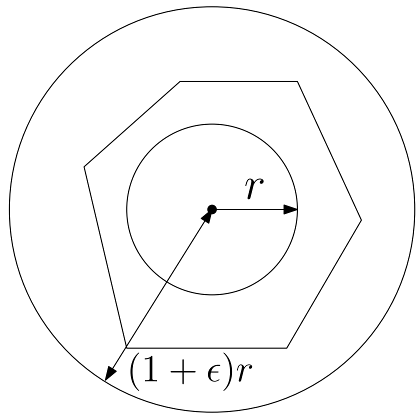
$\mathbf{b}_{\approx}$ is $(1+\epsilon)$ approximation of $\mathbf{b} = \mathbf{b}(p, r)$, if
$\mathbf{b} \subseteq \mathbf{b}_{\approx} \subseteq \mathbf{b}(p, r(1+\epsilon)$

Answering ANN queries

Haven't made our life easier, since answering target ball queries is a difficult problem

Don't need exact balls

$(1+\epsilon)$ ball



b$_\approx$ is $(1+\epsilon)$ approximation of **b** $=$ **b**(p, r), if

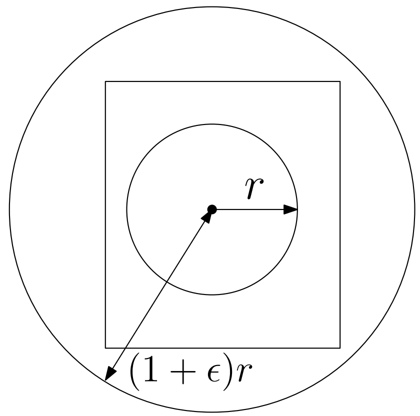
b $\subseteq$ **b**$_\approx$ $\subseteq$ **b**$(p, r(1+\epsilon))$

Answering ANN queries

Haven't made our life easier, since answering target ball queries is a difficult problem

Don't need exact balls

$(1+\epsilon)$ ball



$\mathbf{b}_{\approx}$ is $(1+\epsilon)$ approximation of $\mathbf{b} = \mathbf{b}(p, r)$, if

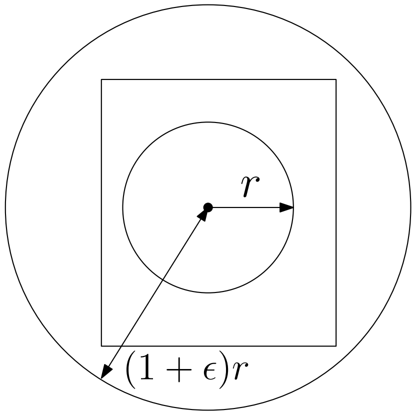
$\mathbf{b} \subseteq \mathbf{b}_{\approx} \subseteq \mathbf{b}(p, r(1+\epsilon))$

Answering ANN queries

Haven't made our life easier, since answering target ball queries is a difficult problem

Don't need exact balls

$(1+\epsilon)$ ball



b$_\approx$ is $(1+\epsilon)$ approximation of **b** $=$ **b**(p, r), if

b $\subseteq$ **b**$_\approx$ $\subseteq$ **b**$(p, r(1+\epsilon)$

Consider *Interval Near Neighbor* structure on approximate balls:

If $I_\approx(P, r, R, \epsilon/16)$ is a $(1+\epsilon/16)$ approximation to $I(P, r, R, \epsilon/16)$

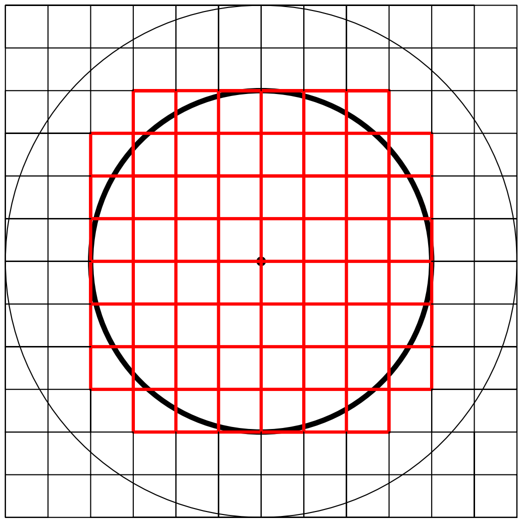
If for point q, $I_\approx(P, r, R, \epsilon/16)$ returns a ball $(p, \alpha), \alpha \in [r, R] \implies p$ is $(1+\epsilon/4)$-ANN to q:

$$r(1+\epsilon/16)^i \leq d_P(q) \leq d(p,q) \leq r(1+\epsilon/16)^{i+1}(1+\epsilon/16) \leq (1+\epsilon/4)r$$

Fast ANN in $\mathbb{R}^d$

The distance between 2 points in a d-dimensional cell of size α is at most $\sqrt{\sum_{i=1}^{d} \alpha^2} = \sqrt{d}\alpha$

For a given ball, **b**(p, r), construct a grid centered at p, with cell-size 2^i, s.t. $\sqrt{d}2^i \leq \frac{(\epsilon r)}{16}$
Call, **b**$_{\approx}$ the set of cells that intersect **b**(p, r)

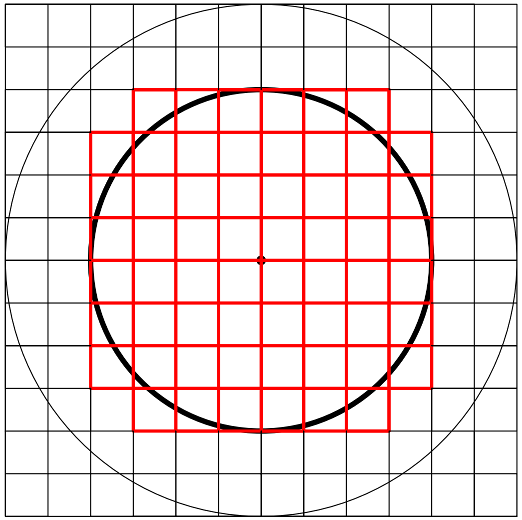


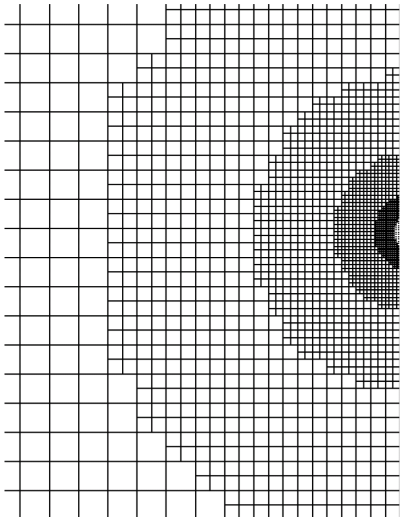
b$_{\approx}$ is a $(1 + \epsilon/16)$ approximate ball, and contains $O\left(\frac{r^d}{(\epsilon r)^d}\right) = O\left(\frac{1}{\epsilon}^d\right)$ cells

Fast ANN in $\mathbb{R}^d$

The distance between 2 points in a d-dimensional cell of size α is at most $\sqrt{\sum_{i=1}^{d} \alpha^2} = \sqrt{d}\alpha$

For a given ball, **b**(p, r), construct a grid centered at p, with cell-size 2^i, s.t. $\sqrt{d}2^i \leq \frac{(\epsilon r)}{16}$
Call, **b**$_\approx$ the set of cells that intersect **b**(p, r)





b$_\approx$ is a $(1 + \epsilon/16)$ approximate ball, and contains $O\left(\frac{r^d}{(\epsilon r)^d}\right) = O\left(\frac{1}{\epsilon}^d\right)$ cells

Fast ANN in $\mathbb{R}^d$

- Fix the origin, and construct grid-cells from there
- If there are 2 cells with the same size, pick the one, corresponding to the smallest ball
- Thus construct an approximate I-$(1 + \epsilon/16)$ data structure C

Fast ANN in $\mathbb{R}^d$

- Fix the origin, and construct grid-cells from there
- If there are 2 cells with the same size, pick the one, corresponding to the smallest ball
- Thus construct an approximate I-$(1 + \epsilon/16)$ data structure C

Finding the smallest ball containing $q \Longleftrightarrow$ finding the smallest grid-cell containing q

Fast ANN in $\mathbb{R}^d$

- Fix the origin, and construct grid-cells from there
- If there are 2 cells with the same size, pick the one, corresponding to the smallest ball
- Thus construct an approximate I-$(1+\epsilon/16)$ data structure C

Finding the smallest ball containing $q \iff$ finding the smallest grid-cell containing q

Encode all the cells of C into a compressed quad-tree, such that each cell appears as a node

- Construction takes $O(|C| \log |C|)$ time
- Finding the appropriate node in C takes $O(\log |C|)$ time
- If information about smallest ball is propagated down the tree, answering a query takes $O(\log |C|)$

Fast ANN in $\mathbb{R}^d$

- Fix the origin, and construct grid-cells from there
- If there are 2 cells with the same size, pick the one, corresponding to the smallest ball
- Thus construct an approximate I-$(1+\epsilon/16)$ data structure C

Finding the smallest ball containing $q \iff$ finding the smallest grid-cell containing q

Encode all the cells of C into a compressed quad-tree, such that each cell appears as a node

- Construction takes $O(|C| \log |C|)$ time
- Finding the appropriate node in C takes $O(\log |C|)$ time
- If information about smallest ball is propagated down the tree, answering a query takes $O(\log |C|)$

Recall that we had a data structure with $O(\frac{n}{\epsilon} \log \frac{n}{\epsilon})$ balls. Each ball is approximated by $O(\frac{1}{\epsilon^d})$ cells
$\Rightarrow$ The overall complexity of the quad-tree is $O(N)$, where $N = O(\frac{n}{\epsilon^{d+1}} \log \frac{n}{\epsilon})$.
By noticing that there are many balls of similar sizes, we reduce the complexity to:

- Construction: $O(n\epsilon^{-d} \log^2(n/\epsilon)$ time
- Storage: $O(n\epsilon^{-d} \log(n/\epsilon)$ space
- Point location query: $O(\log(n/\epsilon))$