Approximate Nearest Neighbor Search (ANN)

Store $n$ pts $P \subseteq \mathbb{R}^d$ ($d$ const)

St. given query pt $q \in \mathbb{R}^d$, find $p \in P$ with

$$d(p, q) \leq (1+\varepsilon) \min_{p' \in P} d(p', q)$$

Exact problem ($\varepsilon = 0$): by halfspace range searching in $d$ H dms

$O(n)$ space, $O^*(n^{1-\frac{1}{\sqrt{d}}})$ time

or $O^*(n^{\frac{d}{2}})$ space, $O((\log n)^{\frac{d}{2}})$ time

Approx decision problem: given $r$,

return some pt of distance $\leq (1+\varepsilon) r$

or declare all pts have distance $> r$.

Method 0: for decision with fixed $r$

form grid of side length $\frac{\varepsilon_r}{\sqrt{d}}$

store $S =$ all nonempty grid cells

query($q$):

check if any grid cell intersecting ball $(q, r)$

is in $S$ by hashing

Space $O(n)$

time $O(\# \text{grid cells intersecting ball}(q, r))$

$$= O\left( \frac{\text{volume}(\text{ball}(q, (1+\varepsilon)r))}{(\varepsilon_r/\sqrt{d})^d} \right)$$

$$= O\left( \frac{2T_{d/2}}{d^{d/2}} \left( \frac{(1+\varepsilon)r}{\varepsilon_r/\sqrt{d}} \right)^d \right)$$

$$= O\left( \frac{(1+\varepsilon)^d}{\varepsilon_r^d} \right)$$

$$= O\left( \frac{C^d}{\varepsilon_r^d} \right)$$

RMK - alternatively, $O\left( \frac{(1+\varepsilon)^d}{\varepsilon_r^d} \right)$ space, $O(1)$ time.

What if $r$ is not fixed? (Store $S =$ all grid cells intersecting ball $(p, r)$ for some $p \in P$)
Method 1: Quadtree
idea - hierarchy of grids

Def a quadtree cell \( B \) is a grid cell of side length \( 2^\ell \) for some \( \ell \in \{0, 1, \ldots, \log U\} \)

Eq

space \( O(n \log U) \)
reducible to \( O(n) \) by compressed quadtree (shortcutting deg-1 nodes)

height \( O(\log U) \)
decision query \((B,q,r)\):
if ball \((q,r)\) does not intersect \(B\) return
if \(B\) has side length \(< \frac{Er}{4}\) return any pt in \(B\)
for each child \(B_i\) of \(B\)

decision query \((B_i,q,r)\)

\[
\Rightarrow \text{time} = O\left(\sum_{l=0}^{\frac{2\log(l)}{d}} \frac{2^l}{2^l d}\right)
\]

\[
= O\left(\left[\frac{1}{2(l/q)}\right] + \left[\frac{1}{2E(x/q)}\right] + \ldots\right)
\]

\[
= O\left(\log U + \frac{1}{\varepsilon d}\right)
\]

How to adapt to ANN?

[Maintain curm min \(\varepsilon\), expand cells in increase order of dist]

Method 2...with Shifting (Bern '93) < first ann with const factor first approx.

name-ANN-query \((B,q)\):
find child \(B_i\) containing \(q\)
return name-ANN-query \((B_i,q)\)

Let \(p^*\) = nearest neighbor of \(q\)
\[r^* = d(p^*, q)\]

Def \(q\) is good if \(p^*\) and \(q\) lie in a quadtree cell
of side length \(< 2(d+1)r^*\)

Shifting Lemma
Shift all pts by random vector in \([0]^d\)
Then \(q\) is good with const prob.

Pf: Say \((d+1)r^* < 2\varepsilon^* < 2(d+1)r^*\)

\[
\text{Pr}[q \text{ bad}] = \text{Pr}\left[p^*q \text{ crosses boundary of quadtree cell of side length } 2\varepsilon^*\right]
\]

\[
2^\varepsilon^* < \frac{1}{2}\]
\[ \leq P[i \text{ shift lies in a bad interval of length } r^* \text{ along some dim}] \leq d \cdot \frac{r^*}{2^\ell} \leq \frac{d}{d+1} \cdot \frac{r^*}{2^\ell} \cdot 0 \]

Shifting Lemma (Deterministic version)
Say \( d \) even, \( U = 2^w \).
Shift all pts by vector \( v_i = \left( \frac{i \cdot 2^w}{d+1}, \ldots, \frac{i \cdot 2^w}{d+1} \right) \), \( i = 0, \ldots, d \).
Then \( q \) is good for some \( i \).

**Pf:** \( q \) bad for \( i \) \( \Rightarrow \) \( \frac{i \cdot 2^w}{d+1} \mod 2^\ell \) lies in a bad interval of length \( r^* \)

\[
\frac{i \cdot 2^w}{d+1} \mod (d+1) \text{ lies in a bad interval of length } \frac{(d+1)r^*}{2^\ell} < 1
\]

\[
\Rightarrow \frac{i \cdot 2^w}{d+1} \mod (d+1) \text{ is equal to a specific bad integer}
\]

\[
\Rightarrow \text{at most one bad } i \text{ per dim}
\]

\[
\Rightarrow \text{so by pigeonhole, } i \text{ good } i. \quad \square
\]

Just run naive ANN-query for each of these \( d+1 \) shifts.

**Query time:** \( O(\log U) \)

**Analysis:**
Let \( p \) be returned pt at leaf.
Suppose \( q \) is good,
\( p^* \) and \( q \) are in quadtree
cell of side length \( < 2(d+1)r^* \)
\[
\Rightarrow \quad d(p, q) < 2(d+1) \sqrt{d} r^*
\]
\[
\Rightarrow \quad \text{approx factor } O(1)
\]

Reduce query time to \( O(\log \log n) \)? [Arya, Mount et al. '94]
Method 3: Balanced Quadtree

Lemma: $J$ quadtree cell $B$

St.: $|P \cap B|, |P - B| \leq \frac{2^d}{2^{d+1}} n$

Pf.: Let $B$ be smallest quadtree cell with $|P \cap B| \geq \frac{1}{2^{d+1}} n$

Then $|P - B| \leq \frac{2^d}{2^{d+1}} n$

and for each child $B_i$ of $B$,

$|P \cap B_i| \leq \frac{1}{2^{d+1}} n \Rightarrow |P \cap B| \leq \frac{2^d}{2^{d+1}} n$.

Recursively in $P \cap B$ and $P - B$

$\Rightarrow$ binary tree of height $\log \frac{2^{d+1}}{2^d} n = O(\log n)$

$\Rightarrow$ query time $O(\log n)$

approx-factor $O(1)$ by shifting

Other variants:
Arora, Mauniot, et al.'s
BDD trees,
Goyal et al.'s
BAR trees, ...

Method 4: Z-Ordering (C.'02)

Idea: don't use any trees!

directly reduce to 1D, by sorting?

Space-filling Curve
(other exs: Hilbert Curve, etc.)
Def: Given point \( p = (x, y) \in [U]^2 \)
write \( x = a_{w-1}a_{w-2} \ldots a_0 \) in binary
\( y = b_{w-1}b_{w-2} \ldots b_0 \)
define \( \text{shuffle}(p) = a_{w-1}b_{w-1}a_{w-2}b_{w-2} \ldots a_0b_0 \)

\[ \text{eg} \quad x = 2, \ y = 1 \rightarrow 010 \quad \text{shuffle} \quad 001001 \rightarrow 9 \quad 001 \]

**Notes**
- \( \text{shuffle}(p) \) corresponds to position in \( z \)-order
- \( \text{shuffle} \) maps a quadtree cell to an interval of contiguous integers in \( 1D \).

**Lemma:** Can compare \( \text{shuffle}(p) \) with \( \text{shuffle}(q) \) in \( O(1) \) time.

**Proof:** \( p = (x, y), \ q = (x', y') \)
\[ \text{eg.} \quad x = 1101, \ y = 1011 \]
\[ x' = 1011, \ y' = 1001 \]

if \( \text{msb}(x \oplus x') > \text{msb}(y \oplus y') \)
compare \( x \) with \( x' \)
else compare \( y \) with \( y' \)

To check \( \text{msb}(a) > \text{msb}(b) \):
- check \( a > b \) and \( a \oplus b > b \)

\[ \text{eg.} \quad a = 01 \times \times \times, \ a = 0001 \times, \ a = 01 \times \times \times \]
\[ b = 0001 \times, \ b = 01 \times \times \times, \ b = 01 \times \times \times \]
\[ a \oplus b = 01 \times \times \times, \ a \oplus b = 01 \times \times \times, \ a \oplus b = 00 \times \times \times \]

Preproc align:
- for each of \( d+1 \) shifts
- store pts sorted along \( z \)-order \( \Rightarrow O(n \log n) \) time
- \( O(n) \) space
query(q):
  for each of d+1 shifts
  find pred & succ of q
  return closest pt found

Analysis:
Suppose q is good.
p^k, q are in quadtree
  cell of side length ≤ 4(d+1) r^k
⇒ so is succ or pred of q.
⇒ approx factor \( O(1) \).

Remark: immediately improvable to \( O(\log \log U) \) or \( O(\sqrt{\log n}) \)
by 1D data structures
can be made dynamic!

Refining approx factor:
  Orphan:
  let \( r = \frac{1}{2} \) factor-c approx
  divide ball(\( p^k, cr \)) into \( O(\frac{1}{c^d}) \) quadtree cells
  of side length ≤ \( 2r \)
  test if each subcell is empty
  \( \Rightarrow \) maps to 1D search.

⇒ query time \( O(\frac{1}{c^d} \log n) \) or \( O(\frac{1}{c^d} \log \log U) \)

Best \( q \)-dependency? Still open.
Arya, Mount et al., 94 / \( O(n) \) space, \( O(\frac{1}{c^d} \log n) \) time
Clarkson'94 / C,'97 \( O(\frac{1}{c^d} \log n) \) space, \( O(\frac{1}{c^d} \log \log n) \) time
Arya, de Fousner, Mount '12-16 \( O(\frac{1}{c^d} \log n) \) space, \( O(\frac{1}{c^d} \log \log n) \) time
17 / C,'17, \( O(\frac{1}{c^d} \log n) \) time
Option 2: (C.-Har-Peled-Jones '18)

Idea - keep query algm same
but use multiple orderings

Increase fan-out to \( \frac{1}{\varepsilon} \)
(reorder children)

\[
\begin{array}{cc}
0 & 1 & 2 & 3 \\
4 & 5 & 6 & 7 \\
8 & 9 & 10 & 11 \\
12 & 13 & 14 & 15 \\
\end{array}
\]

\[
\begin{array}{cc}
0 & 1 & 2 & 3 \\
4 & 5 & 6 & 7 \\
8 & 10 & 9 & 11 \\
12 & 14 & 13 & 15 \\
\end{array}
\]

\[
\begin{array}{cc}
0 & 1 & 2 & 4 & 6 \\
5 & 7 & 1 & 3 \\
8 & 10 & 12 & 14 \\
13 & 15 & 9 & 11 \\
\end{array}
\]

\( \forall \text{ children } u_i, u_j, \)

\( \exists \text{ ordering s.t. } u_i, u_j \text{ adjacent} \)

\# orderings \( O\left(\frac{1}{\varepsilon}^{4} \log \frac{1}{\varepsilon}\right) \)

\# levels \( \text{compressed} \)

[many appl'ns...]