Approximate Nearest Neighbor Search (ANN)

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

s.t. 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 $d+1$ dims

$O(n)$ space, $O^*(n^{1-1/\lceil d/2 \rceil})$ time

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

Approx decision problem: given r,

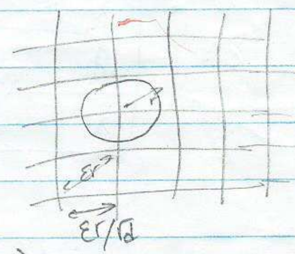
return some pt of distance $\leq (1+\varepsilon) r$

or declare all pts have distance $> r$.

Method 0: for decision with fixed r

form grid of side length $\varepsilon r/\sqrt{d}$

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

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

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

$$= O\left(\left(\frac{c}{\varepsilon}\right)^d\right)$$

naively: $O\left(\frac{r^d}{(\varepsilon r/\sqrt{d})^d}\right) = O\left(\left(\frac{\sqrt{d}}{\varepsilon}\right)^d\right)$

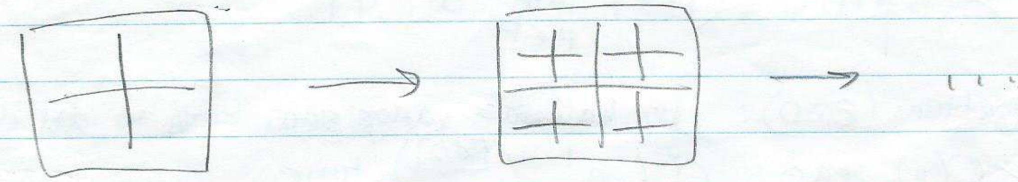
Rmk - alternatively, $O\left(\left(\frac{c}{\varepsilon}\right)^d n\right)$ space, $O(1)$ time

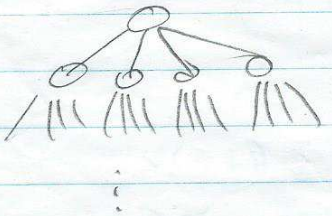
(store S = all grid cells intersecting ball(p, r) for some $p \in P$)

what if r is not fixed?

Method 1: Quadtree

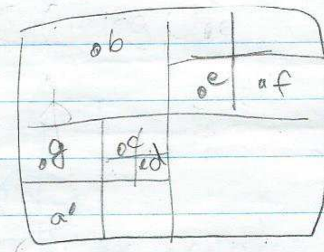
idea - hierarchy of grids

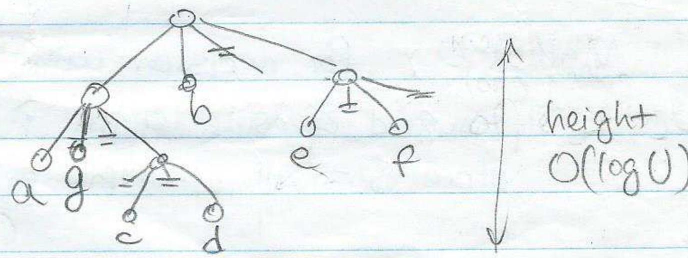




Def A quadtree cell B is a grid cell of side length 2^{ℓ} for some $\ell \in \{0, 1, \ldots, \log U\}$

e.g.



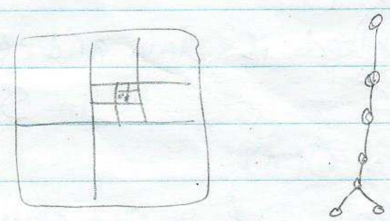


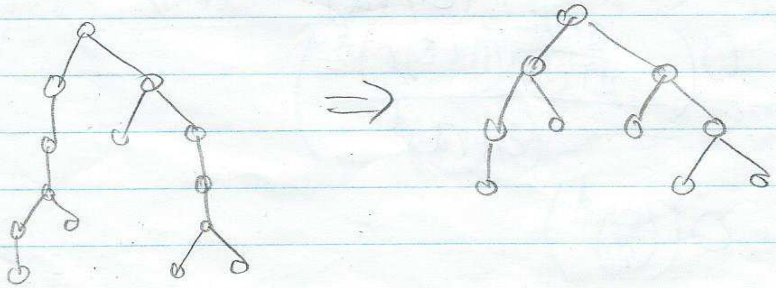
space $O(n \log U)$

reducible to $\boxed{O(n)}$

by compressed quadtree

(shortcutting deg-1 nodes)





decis-query (B, q, r):

    if ball(q, r) does not intersect B return
    if B has side length $< \varepsilon r/\sqrt{d}$ return any pt in B
    for each child B_i of B
        decis-query (B_i, q, r)

$\Rightarrow$ time $= O(\#$ quadtree cells of side length $> \frac{\varepsilon r}{\sqrt{d}}$ (intersecting B))

$$= O\left(\sum_{\ell : 2^\ell > \frac{\varepsilon r}{\sqrt{d}}} \left\lceil \frac{r^d}{(2^\ell)^d} \right\rceil \right)$$

(ignore const factors depending on d)

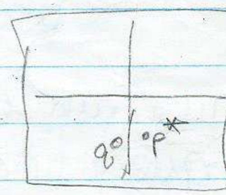
$$= O\left(\left\lceil \frac{1}{(\varepsilon/\sqrt{d})^d} \right\rceil + \left\lceil \frac{1}{(2\varepsilon/\sqrt{d})^d} \right\rceil + \cdots \right)$$

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

how to adapt to ANN? [maintain curr. min r, expand cells in increas order of dist]

Method 2 ... with Shifting (Bern'93) $\leftarrow$ first aim for const factor first approx.

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

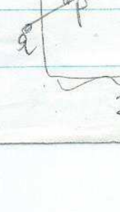


Let $p^* =$ nearest neighbor of q
$r^* = d(p^*, q)$

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 $[U]^d$.
    Then q is good with const prob.

Pf: Say $(d+1)r^* < 2^\ell \le 2(d+1)r^*$
    $\Pr[q \text{ bad}] \le \Pr[\, \overline{p^*q}$ crosses boundary of quadtree cell of side length $2^\ell\,]$



$\leq \Pr[$ shift lies in a bad interval of length r^* along some dim $]$

$$\leq d \cdot \frac{r^*}{2^\ell} \leq \frac{d}{d+1} \quad \square$$

Shifting Lemma (Deterministic version)

Say d even, $U = 2^w$.

Shift all pts by vector $v_i = \left(\frac{i2^w}{d+1}, \ldots, \frac{i2^w}{d+1} \right)$, $i = 0, \ldots, d$.

Then q is good for some i.

Pf: q bad for i along one dim. $\Rightarrow$ $\frac{i2^w}{d+1} \bmod 2^\ell$ lies in a bad interval of length r^*

$\Rightarrow$ (multiply by $\frac{d+1}{2^\ell}$) $i2^{w-\ell} \bmod (d+1)$ lies in a bad interval of length $\frac{(d+1)r^*}{2^\ell} < 1$

$\Rightarrow$ $i2^{w-\ell} \bmod (d+1)$ is equal to a specific bad integer.

Fact: $[ax \equiv b \bmod n]$ has unique soln if a, n are relatively prime

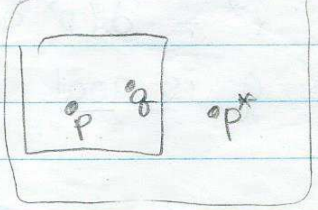
$\Rightarrow$ since $2^{w-\ell}$ and $d+1$ are rel. prime: at most one bad i per dim, so by pigeonhole, $\exists$ good i. $\square$

just run naive-ANN-query for each of these $d+1$ shifts

query time: $\boxed{O(\log U)}$

Analysis:

let p be returned pt at leaf.



Suppose q is good, p^* and q are in quadtree cell of side length $< 2(d+1)r^*$

$\Rightarrow$ so are p and q

$\Rightarrow$ $d(p,q) < 2(d+1)\sqrt{d}\, r^*$

$\Rightarrow$ approx factor $O(1)$.

reduce query time to $O(\log n)$? [Arya, Mount et al. '94]

Method 3: Balanced Quadtree

Lemma $\exists$ quadtree cell B ← tree centroid

s.t. $|P \cap B|, |P - B| \le \frac{2^d}{2^d+1} n$

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

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

and for each child B_i of B,

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

recurse in $P \cap B$ & $P - B$

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

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

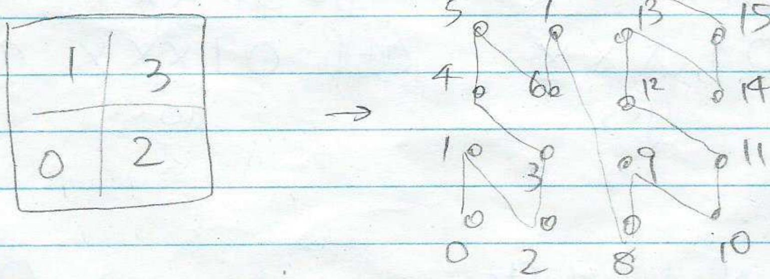
approx factor $O(1)$ by shifting

[other variants: Arya, Mount et al.'s BBD trees, Goodrich 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 $x = a_{w-1}a_{w-2} \cdots a_0$ in binary
$y = b_{w-1}b_{w-2} \cdots b_0$
define $\text{shuffle}(p) = a_{w-1}b_{w-1}a_{w-2}b_{w-2} \cdots a_0b_0$

e.g. $x=2, y=1 \rightarrow \begin{matrix}010\\001\end{matrix} \xrightarrow{\text{shuffle}} 001001 \rightarrow 9$

Notes
- shuffle(p) corresponds to position in Z-order
- shuffle maps a quadtree cell to an interval of contiguous integers in 1D.

Lemma can compare shuffle(p) with shuffle(q) in $O(1)$ time.
Pf: $p=(x,y)$, $q=(x',y')$

e.g. $x=1101$ $y=1011$
$x'=1011$ $y'=1001$
$x \oplus x' = 0110$ $y \oplus y' = 00100$

$\oplus$ bitwise or
msb most significant 1 bit

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 $a \oplus b > b$.

e.g.

$a = 01XXX$	$a = 001X$	$a = 01XXX$
$b = 0001X$	$b = 01XXX$	$b = 01XXX$
$a \oplus b = 01XXX$	$a \oplus b = 01XXX$	$a \oplus b = 00XXX$
yes	no	no

$\square$

Preproc alg'm:
for each of $d+1$ shifts $\Rightarrow$ $O(n \log n)$ time
store pts sorted along Z-order $O(n)$ space

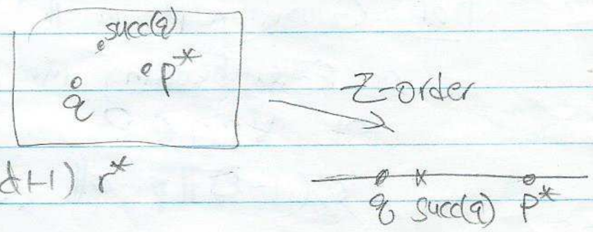
query(q):

for each of d+1 shifts

find pred & succ of q $\Rightarrow$ $\boxed{O(\log n)}$ time

return closest pt found

Analysis:



Suppose q is good.

p^*, q are in quadtree cell of side length $\le 4(d+1)\, r^*$

$\Rightarrow$ so is succ or pred of q.

$\Rightarrow$ approx factor $O(1)$.

Rmk: immediately improvable to $O(\log\log U)$ or $O(\sqrt{\log n})$ by 1D data structures)

can be made dynamic!

Refining approx factor:

Option 1: let r = factor-c approx.

divide ball(q, cr) into $O(\frac{1}{\varepsilon^d})$ quadtree cells of side length $\le \frac{\varepsilon r}{\sqrt{d}}$

test if each subcell is empty

maps to 1D search.

$\Rightarrow$ query time $\boxed{O(\frac{1}{\varepsilon^d} \log n)}$ or $O(\frac{1}{\varepsilon^d} \log\log U)$ ...

Best ε-dependence? still open

Arya, Mount et al. '94 $O(n)$ space, $O(\frac{1}{\varepsilon^d} \log n)$ time

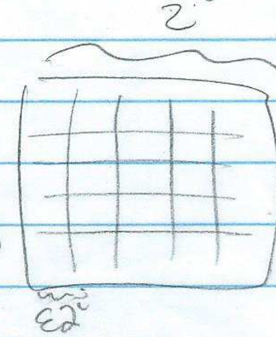
Clarkson '94 / C. '97 $\tilde{O}(\frac{1}{\varepsilon^{d/2}} n)$ space, $O(\frac{1}{\varepsilon^{d/2}} \log n)$ time

Arya, da Fonseca, Mount '12–'16 $\tilde{O}(\frac{1}{\varepsilon^{d/3}} n)$ space, $O(\frac{1}{\varepsilon^{d/3}} \log n)$ time

" " " '17 / C. '17 $\tilde{O}(\frac{1}{\varepsilon^{d/4}} n)$ " $O(\frac{1}{\varepsilon^{d/4}} \log n)$ "

<u>Option 2:</u> (C.-Har-Peled-Jones '18)

idea - keep query alg'm same
but use multiple orderings



increase fan-out to $\left(\frac{1}{\varepsilon}\right)^d$ (i.e. compress $\log\frac{1}{\varepsilon}$ levels into one)

reorder children

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

0	2	1	3
4	6	5	7
8	10	9	11
12	14	13	15

0	2	4	6
5	7	1	3
8	10	12	14
13	15	9	11

...

$\forall$ children u_i, u_j,
$\exists$ ordering s.t. u_i, u_j adjacent

orderings $O\left(\left(\frac{1}{\varepsilon}\right)^d \underbrace{\log\frac{1}{\varepsilon}}_{\#\text{ levels compressed}}\right)$

[many appl'ns ...]