CSE520 Computational Geometry
Lecture 25
Shifted Quadtrees

Antoine Vigneron

Ulsan National Institute of Science and Technology

June 15, 2020

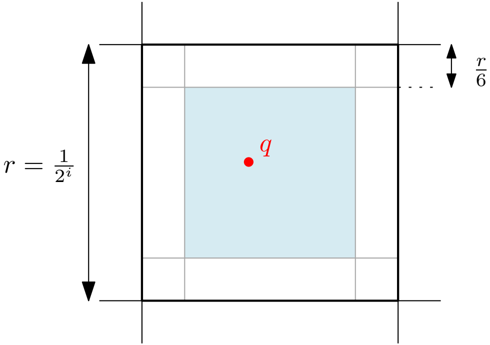
Course Organization

- Today, I present *shifted quadtrees*, which are a modified version of the compressed quadtrees presented in last lecture.
- I will also show how to apply them to near-neighbor searching.

References

- Sariel Har-Peled's book, chapters 2, 11 and 17.
- Timothy Chan's paper *Approximate Nearest Neighbor Queries Revisited.*

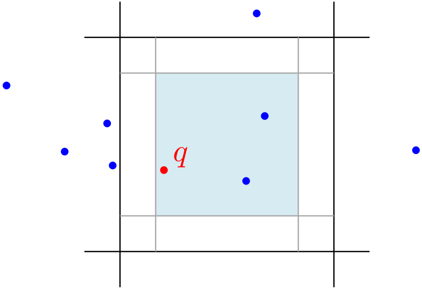
Central Point in a Quadtree Box



- Remember that a quadtree box in $\mathbb{R}^2$ is of the form $[rk_x, r(k_x+1)] \times [rk_y, r(k_y+1)]$ where $r = 1/2^i$ for some $i \in \mathbb{N}$ and, $k_x, k_y \in \mathbb{Z}$.
- A point q in a quadtree box b of size r in $\mathbb{R}^2$ is *central* if it is at distance at least $r/6$ from the boundary of b.

Central Point in a Quadtree Box

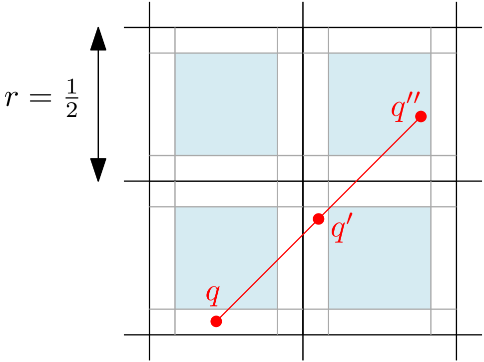
- Why is it a useful property?



Example

If this box contains at least one input point, then one of them is a $4\sqrt{2}$-ANN of q.

Central Point in a Quadtree Box



Lemma

Let $q = (x, y) \in \mathbb{R}^2$ *and let* $r = 1/2^i$ *for some* $i \in \mathbb{N}$. *Let* $q' = (x + \frac{1}{3}, y + \frac{1}{3})$ *and* $q'' = (x + \frac{2}{3}, y + \frac{2}{3})$. *Then at least one of these three points is central in the quadtree box of size* r *that contains it.*

Proof of the Lemma

- Euclidean division: $19 = 3 \times 5 + 4$.
- We have $19 \operatorname{div} 5 = 3$ and $19 \bmod 5 = 4$.

- More generally, we can apply Euclidean division to any positive divisor.
- In particular, we will apply it to $r = 1/2^i$.

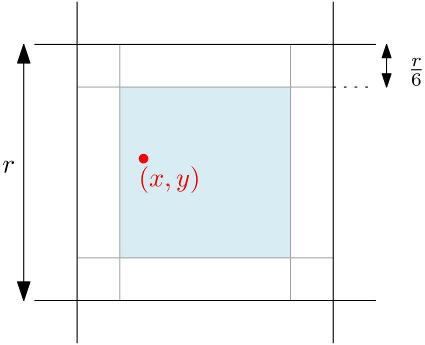
- So for any $z \in \mathbb{R}$, there is a unique pair $\delta \in \mathbb{Z}$, $\rho \in \mathbb{R}^+$ such that $z = \delta r + \rho$ and $0 \leqslant \rho < r$.
- We write $\delta = z \operatorname{div} r$ and $\rho = z \bmod r$.
- For instance, $\frac{5}{3} = 3 \times \frac{1}{2} + \frac{1}{6}$ and thus $\frac{5}{3} \bmod \frac{1}{2} = \frac{1}{6}$.

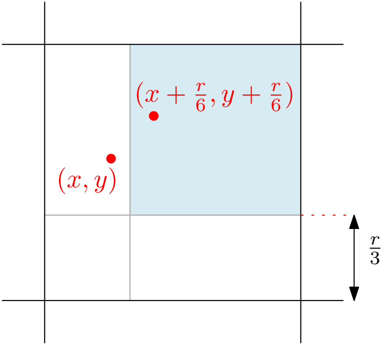
- Expressions: $\delta = \lfloor z/r \rfloor$ and $r = z - r\lfloor z/r \rfloor$.

Proof of the Lemma

- $q = (x, y)$ is central iff

$$\left(x + \frac{r}{6}\right) \bmod r \geqslant \frac{r}{3} \quad \text{and} \quad \left(y + \frac{r}{6}\right) \bmod r \geqslant \frac{r}{3}.$$





Proof of the Lemma

- Remember that $r = 2^i$.
- We multiply the inequalities above by $3 \cdot 2^i$, and we obtain that $q = (x, y)$ is central iff

$$\left(2^i \cdot 3x + \frac{1}{2}\right) \bmod 3 \geqslant 1 \quad \text{and} \quad \left(2^i \cdot 3y + \frac{1}{2}\right) \bmod 3 \geqslant 1.$$

- It means that q is *not* central iff

$$\left(2^i \cdot 3x + \frac{1}{2}\right) \bmod 3 < 1 \quad \text{or} \quad \left(2^i \cdot 3y + \frac{1}{2}\right) \bmod 3 < 1.$$

- Now suppose, for sake of contradiction, that none of the points q, q' and q'' are central.

Proof of the Lemma

- Then the three statements below are true:

$$\left(2^i \cdot (3x+0) + \frac{1}{2}\right) \bmod 3 < 1 \quad \text{or} \quad \left(2^i \cdot (3y+0) + \frac{1}{2}\right) \bmod 3 < 1$$

$$\left(2^i \cdot (3x+1) + \frac{1}{2}\right) \bmod 3 < 1 \quad \text{or} \quad \left(2^i \cdot (3y+1) + \frac{1}{2}\right) \bmod 3 < 1$$

$$\left(2^i \cdot (3x+2) + \frac{1}{2}\right) \bmod 3 < 1 \quad \text{or} \quad \left(2^i \cdot (3y+2) + \frac{1}{2}\right) \bmod 3 < 1$$

- So two of the LHS statements or two of the RHS statements are true.
- WLOG, we take the LHS.

Proof of the Lemma

- So there exist integers $0 \leqslant j < k \leqslant 2$ such that

$$\left(2^i \cdot (3x + j) + \frac{1}{2}\right) \bmod 3 < 1 \text{ and } \left(2^i \cdot (3x + k) + \frac{1}{2}\right) \bmod 3 < 1.$$

- Let $z = 2^i \cdot 3x + \frac{1}{2}$. Then we have

$$\left(z + 2^i j\right) \bmod 3 < 1 \text{ and } \left(z + 2^i k\right) \bmod 3 < 1.$$

- As $2^i j$ and $2^i k$ are integers, this is only possible if $\left(2^i j\right) \bmod 3 = \left(2^i k\right) \bmod 3$, and thus $\left(2^i(j-k)\right) \bmod 3 = 0$.
- Since 2^i and 3 are relatively prime, it implies that $(j - k) \bmod 3 = 0$. (See discrete mathematics course.)

- It contradicts the fact that $0 \leqslant j < k \leqslant 2$.
- So it completes our proof.

Generalization

- The lemma above generalizes to arbitrary *even* dimension d.
- We say that a point q in a quadtree box b of size r is central if it is at distance at least $r/(2d+2)$ from the boundary of b.
- Let $\tau_d = \frac{1}{d+1}(1, 1, \ldots, 1)$.

Lemma

Suppose that d is even. Let $q \in \mathbb{R}^d$ and let $r = 1/2^i$ for some $i \in \mathbb{N}$. Then at least one of the points $q + j\tau_d$, $j \in \{0, 1, \ldots, d\}$ is central in the quadtree box of size r that contains it.

(Proof in the paper by T. Chan.)

- When d is odd, we replace d with $d+1$. So we have $\tau_d = \frac{1}{d+2}$ and we consider the points $q + j\tau_d$ where $j \in \{0, \ldots, d+1\}$. Being central means that the distance from the boundary is at least $r/(2d+4)$.

Shifted Quadtrees

- We would like to construct a quadtree for a set P of n points such that for any box size $r = 1/2^i$, any query point q is central within the quadtree box of size r containing it.
- This is impossible, but we can do the following.

- Let $j\tau_d + P$ be the set of points $\{j\tau_d + p \mid p \in P\}$.
- Let $\mathcal{T}_j$ be the compressed quadtree recording $j\tau_d + P$, where $j \in \{0, \dots, d\}$ when j is even and $j \in \{0, \dots, d+1\}$ when j is odd.
- We construct all these quadtrees.
- For a query point q, we perform the query $q_j = j\tau_d + q$ in $\mathcal{T}_j$ for all j.
- For one of these queries, q is central in the quadtree box of size r that contains q. (Remark: this box may not be the region associated with a node of the quadtree.)

Low Quality ANN Queries

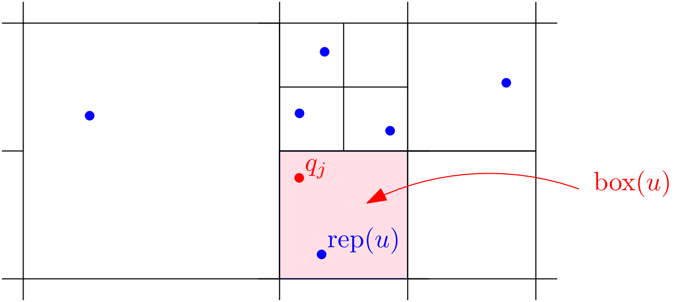
- We now show how to use the at most $d+2$ shifted quadtrees to report an approximate near neighbor (ANN).

Low quality ANN

procedure $\textsc{ShiftedANN}(\mathcal{T}_0, \ldots, \mathcal{T}_{d+1}, q)$
    $p_{\text{best}} \leftarrow \text{rep}(\text{root of } \mathcal{T}_0)$
    for $j \leftarrow 0, d+1$ **do**
        find the deepest node u of $\mathcal{T}_j$ containing $q_j = j\tau_d + q$
        if u is an empty leaf node **then**
            $p_{\text{temp}} \leftarrow \text{rep}(\text{parent}(u))$
        else
            $p_{\text{temp}} \leftarrow \text{rep}(u)$
        if $d(q, p_{\text{temp}}) < d(q, p_{\text{best}})$ **then**
            $p_{\text{best}} \leftarrow p_{\text{temp}}$
    return p_{best}

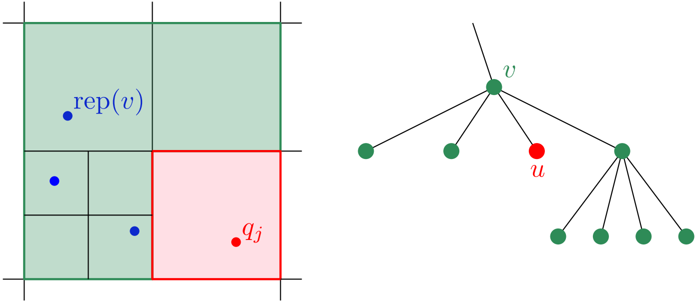
Low Quality ANN Queries

- We now explain what this algorithm does. There are 3 cases.



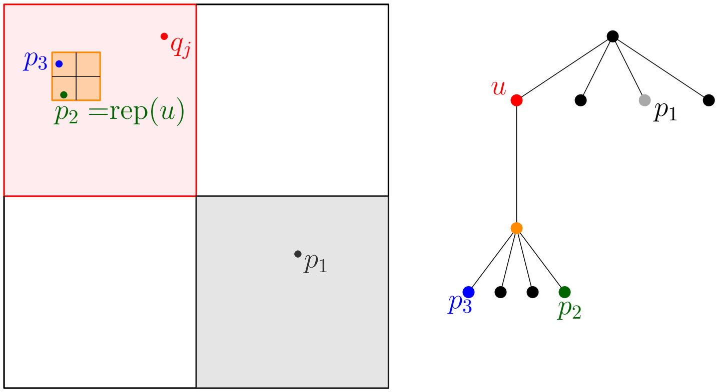
- **Case 1**. The node u is a non-empty leaf.
- So $\mathrm{box}(u)$ contains exactly one point of P, which is $\mathrm{rep}(u)$, and $d(q_j, \mathrm{rep}(u)) \leqslant \mathrm{diam}(\mathrm{box}(u))$.

Low Quality ANN Queries



- **Case 2.** The node u is an empty leaf.
- Then $v =$parent(u) must be a non-empty internal node of degree 2^d, and $d(q_j, \mathrm{rep}(v)) \leqslant 2\,\mathrm{diam}(\mathrm{box}(u))$.
- Remark: In this lecture we construct empty leaves. It increases the size of the quadtree by a factor less than 4.

Low Quality ANN Queries



- **Case 3.** The node u is a compressed node.
- Let $\mathrm{rep}(u)$ be any point in $\mathrm{box}(\mathsf{child}(u))$.
- Then $d(q_j, \mathrm{rep}(u)) \leqslant \mathsf{diam}(\mathrm{box}(u))$.

Low Quality ANN Queries

- We now want to bound the approximation factor.
- Let $r^* = d(q, \mathrm{nn}(q))$ and $r = 2^{\lceil \log((2d+4)r^*) \rceil}$, so we have

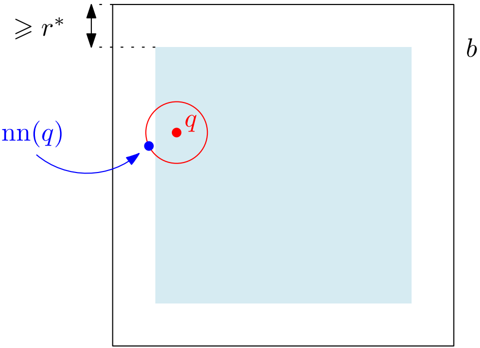
$$(2d+4)r^* \leqslant r < 2(2d+4)r^*.$$

- At least one of the points q_j is central in its quadtree box of size r.
- WLOG, suppose it is $q_0 = q$.

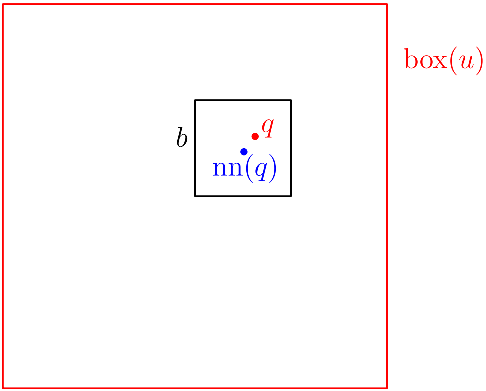
- Let u be the deepest node of $\mathcal{T}_0$ containing q, and let v be its parent.
- From the discussion above, our algorithm returns a point p_{best} such that $d(q, p_{\mathrm{best}}) \leqslant 2\,\mathsf{diam}(\mathrm{box}(u))$.

Low Quality ANN Queries

- Let b be the quadtree box of size r containing q.
- Since $r \geqslant (2d+4)r^*$, we know that $\mathrm{nn}(q) \in b$.

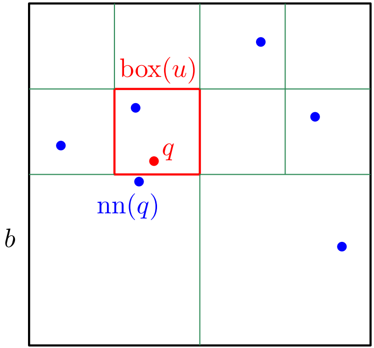


Low Quality ANN Queries



- If $\mathrm{nn}(q)$ is the only point in $b \cap P$, then u is the leaf node containing $\mathrm{nn}(q)$, and $b \subseteq \mathrm{box}(u)$. So the algorithm returns $p_{\mathrm{best}} = \mathrm{rep}(u) = \mathrm{nn}(q)$.

Low Quality ANN Queries



- If $\mathrm{nn}(q)$ is not the only point in $b \cap P$, then $\mathrm{box}(u) \subseteq b$, so $\mathrm{diam}(\mathrm{box}(u)) \leqslant \mathrm{diam}(b)$, and the algorithm returns a point p_{best} at distance at most $2\,\mathrm{diam}(\mathrm{box}(u)) \leqslant 2r^*(2d+4)\sqrt{d}$ from q.

Low Quality ANN Queries

- We conclude the following:

Theorem

The procedure $\textsc{ShiftedANN}$ *returns a* $(4d+8)\sqrt{d}$*-ANN in time* $O(\log n)$.

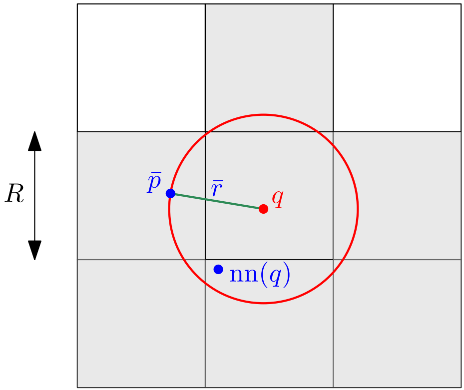
- The running time is $O(\log n)$ because we perform $O(d) = O(1)$ point location queries on compressed quadtree, and each query takes $O(\log n)$ time.

Answering $(1+\varepsilon)$-ANN Queries

- In previous section, we obtained an $O(d^{3/2})$-ANN in $O(\log n)$ time.
- We now show how to improve it to an $O(1+\varepsilon)$-ANN with a query time $O(1/\varepsilon^d + \log n)$.

- Let $\bar{p}$ be the ANN returned by the $\textsc{ShiftedANN}$, and let $\bar{r} = d(q, \bar{p})$. So we have $\bar{r} \leqslant (4d+8)\sqrt{d}r^*$.
- Let $R = 2^{\lceil \log \bar{r} \rceil}$, and thus $\bar{r} \leqslant R < 2\bar{r}$.

Answering $(1+\varepsilon)$-ANN Queries



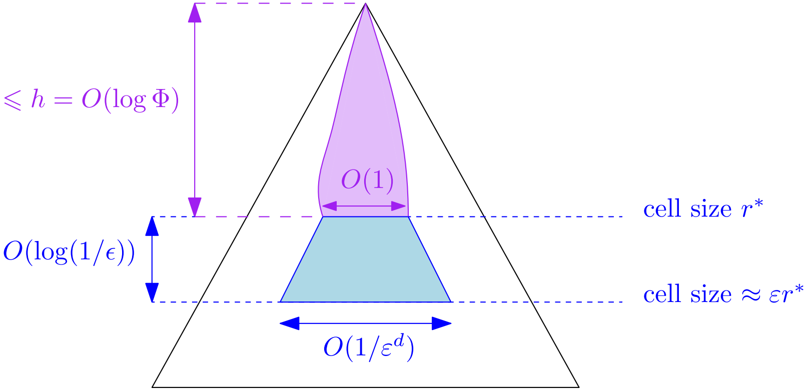
- The ball centered at q with radius $\bar{r}$ is contained in a collection $\mathcal{C}$ of at most 3^d quadtree boxes of size R.

Answering $(1+\varepsilon)$-ANN Queries

- Remember $\mathcal{T}_0$ is a compressed quadtree recording P.
- In order to answer ANN queries, we only need to consider points in $\bigcup \mathcal{C}$.
- So for each box b in $\mathcal{C}$, we find the highest node u_b of $\mathcal{T}_0$ such that $\mathrm{box}(u_b) \subseteq b$.
- It can be done in $O(\log n)$ time. (Similar to a point location query; left as an exercise.)
- Then for each node u_b, we perform a $(1+\varepsilon)$ ANN query for q, starting from node u_b, and using the algorithm from Lecture 23. (This algorithm also works for compressed quadtrees.)
- The nearest of the points returned by this algorithm is a $(1+\varepsilon)$-ANN.
- We still need to analyze this algorithm.

Answering $(1+\varepsilon)$-ANN Queries: Analysis

- From Lecture 23: Analysis of ANN queries.



Answering $(1+\varepsilon)$-ANN Queries: Analysis

- So there are two stages: Initially, when the boxes have size less than r^*, we visit $O(1)$ nodes, and we go down $O(\log \Phi)$.
- Then for smaller boxes, we visit a total of $O(1/\varepsilon^d)$ nodes.

- With the new algorithm, we start from nodes u_b whose boxes are contained in boxes of $\mathcal{C}$, whose size is $R \leqslant (8d+16)\sqrt{d}r^*$.
- So at the first stage, we only go down $\log(R/r^*) = O(1)$ levels.
- Then the first stage takes $O(1)$ time as we visit $O(1)$ node per level.

- So the total running time is:
  - $O(\log n)$ to find a low quality ANN.
  - $O(3^d \log n) = O(\log n)$ to find the 3^d nodes u_b for $b \in \mathcal{C}$.
  - $O(3^d 1/\varepsilon^d) = O(1/\varepsilon^d)$ to perform 3^d ANN queries in the subtrees of these nodes.

Conclusion

Theorem

Given a set of n points in dimension $d = O(1)$, we can construct in $O(n \log n)$ time a a data structure that allows us to answer $(1+\varepsilon)$-ANN queries in $O(1/\varepsilon^d + \log n)$ time.

- Compared with lecture 23, we replaced $\log \Phi$ with $\log n$.
- It took us some effort.
- It is often the case: In geometric approximation algorithms, we can replace $\log \Phi$ with $\log n$ in the time bounds.
- This can be significant in theory as this $\log n$ factor may be shown to be optimal in some sense.
- In practice it may not be so useful, because $\log \Phi$ can only be large for extremely skewed data sets.