Exercise Group of 2 or alone

Important Preprocessing

- Domain discretization (decompose)
- Do grid over it → Computational inefficient
  → Uniform Grid not good
- Non-uniform, conforming mesh that respects the input.
- Long & thin triangles, always bad
- Quadtree quite nice

Used in simulation to e.g. flow (air) around a vehicle or crashed test.
Quadtrees

store geometry data

Coincidence, Incidence, adjacency

\[ e \uparrow \hspace{2em} \uparrow \hspace{2em} q \]

e is incidence to point \( v \)

they are neighbors

Points/vectors: \( p, q, v, w \ldots \)

set of points, polygons: \( P, Q, S \ldots \)

Segments: \( \overrightarrow{P} \)

Quadtree = Tree, with
inner nodes corresponding to squares;
children of a node partition the node into four quadrants

Children Direction
Note: quadtree induces partitioning of the domain

Complete quadtree is like a normal grid, called multi-level grid

**Terminology:**

**Def:** nodes are adjacent: \( \iff \) their squares share an edge

**Def:** square of a node \( v \)

\[
q(v) = [x_v, x_v'] \times [y_v, y_v']
\]

Given: set of points \( P \subseteq \mathbb{R}^2 \)

**Def:** quadtree \( Q \) over point set \( P \)
Proof: \( s = 4 \)
side length of nodes \( v \) on level \( i = \frac{1}{2^k} \).
max distance inside \( v = \frac{\sqrt{2}}{2^k} \).

If \( v \) is inner node \( \Rightarrow c = \frac{\sqrt{2}}{2} \).

\[ i \leq \log \frac{\sqrt{2}}{2} = \log \frac{1}{2} + \frac{1}{2} \Rightarrow \text{Lemma} \]

for leaves: \( i \leq \frac{\log \frac{1}{2} + \frac{1}{2} + 1}{\text{level of parent}} \)

Lemma: Complexity of quadtrees

A quadtree of depth \( d \) over \( n \) points, takes \( O(n(d+1)) \) nodes and takes \( O(n(d+1)) \) to construct.

Proof:

number of leaves = \((\#\text{inner nodes}) \cdot 3 + 1 \) (by induction)

\( \Rightarrow \) upper bounds on inner nodes suffice.
Part 1

- # inner nodes on one level
  \[ \leq n \] (in each inner node there are at least 2 points)

- \[ \Rightarrow \] # inner nodes over all levels \[ \leq n \cdot (d-1) \]

- \[ \Rightarrow \] # nodes \[ \leq n \cdot (d-1) + 2n \]
  because in each quadtree, 2 leaves must contain a point.

Part 2

For each node \( v \), we have time \( T(v) \).

\[ T(v) = O(m), \quad m = \# \text{points in } v \]

Sum of all points on level \( i \) \[ \leq n \]

\[ \sum_{v \text{ is node on level } i} T(v) \in O(n) \]

\[ \Rightarrow \text{ time } O(n \cdot d) \text{ or } (n \cdot d + n) \]
Find north neighbor

Given: node v

Sought: v' - north neighbor of v, such that depth(v') \leq depth(v)

Algorithm get North Neighbor(v)

If \( v \) is root \( \rightarrow \) return nil

let \( p := \text{parent}(v) \)

(1) If \( v \) is lower(left) child of \( p \) \( \rightarrow \) return \( \text{LC child of } p \)

(2) If \( v \) is LR child of \( p \) \( \rightarrow \) return \( \text{UR child of } p \)

Case 1 & 2
\( p' = \text{getNorthNeighbor}(p) \)

If \( p' \) is nil or \( p' \) is leaf \( \rightarrow \) return \( p' \)

(3) If \( u \) is UL child of \( p \) \( \rightarrow \) return UL child of \( p' \)

(4) If \( u \) is UR \( \rightarrow \) return CR

\begin{align*}
\text{Running Time: } & O(d) \\
\text{Worst case to get parent go all the way up in the hierarchy and all the way down again.}
\end{align*}

\( \text{Exam: Sketch this for get west neighbors and why is it so complex in worst case?} \)
**Balanced Quadtrees**

**Def:** A quadtree is "balanced" if

\[ \forall \text{neighbors } v, v': |\text{depth}(v) - \text{depth}(v')| \leq 1 \]

**Corollary**

If \( Q \) is balanced \( \Rightarrow \) size of neighbors differs by factor 2 at most.

---

**Balanced Quadtree**

---

**Algo for constructing balanced quadtrees:**

**Maintain:** Cist \( \mathcal{L} \) of leaves

while there are still nodes \( v \) in \( \mathcal{L} \):

1. check whether \( v \) needs to split (neighbor finding algo)

2. If \( v \) had to split, check whether neighbors need splitting, too
Lemma:
cet Q be a quadtree with m nodes, 
\( \delta = \) balanced quadtree from Q.

Then \( \hat{Q} \) has \( O(m) \) nodes, and it can be constructed in time \( O(m \log m) \).

Proof

Part 1: We prove that there are \( O(m) \) splitting operations
\( \implies \) lemma follows, b/c each split
generate 4 additional nodes.

Define split counter
- Only for old nodes (from origin quadtree):=
  how many times did the old node cause split

- Split counter at end of balancing \( \leq 8 \)

- Each old node generated at most \( 8 \times 4 \) new nodes

\( \hat{Q} \) has \( O(m) \) nodes, and it can be constructed in time \( O(m \log m) \).
**Assertion (to be proven):**

No matter how deep subtree under $v_1$ is, $v_3$ never has to split because of $v_1$.

**Def:** $D(v) =$ depth of subtree under $v$.

**Base case:**

$D(v_2) = D(v_3) = 0$
$D(v_1) = 2$

**Inductive step:** Lemma is true for $D < d$

$D(v_1) = d > 2$

$D(\text{ur child of } v) = d-1$. $v_2$ is split at least once.

**Situation for which Lemma holds, b/c depth of ur child of $v \leq d$**

$\Rightarrow$ ur child of $v_2$ will not be split.

**Part 2:**

Time per node $\in O(d+1)$, b/c of const number of neighbor finding operations (ops).

Each node will be considered only once $\Rightarrow$ lemma