$p = \begin{bmatrix} x \\ y \end{bmatrix}$ $\quad x = \underbrace{x_0 x_1 \cdots x_k}_{\text{bits}}$ $\quad y = \underbrace{y_0 y_1 \cdots y_k}_{\text{bits}}$

$$z(p) = \underbrace{y_0 x_0 y_1 x_1 \cdots y_k x_k}_{\text{(in binary)}}$$

Computing ANN quickly:

Store the points $P \subset \mathbb{R}^2$ in a balanced binary search tree T where point $p \in P$ has key $z(p)$.

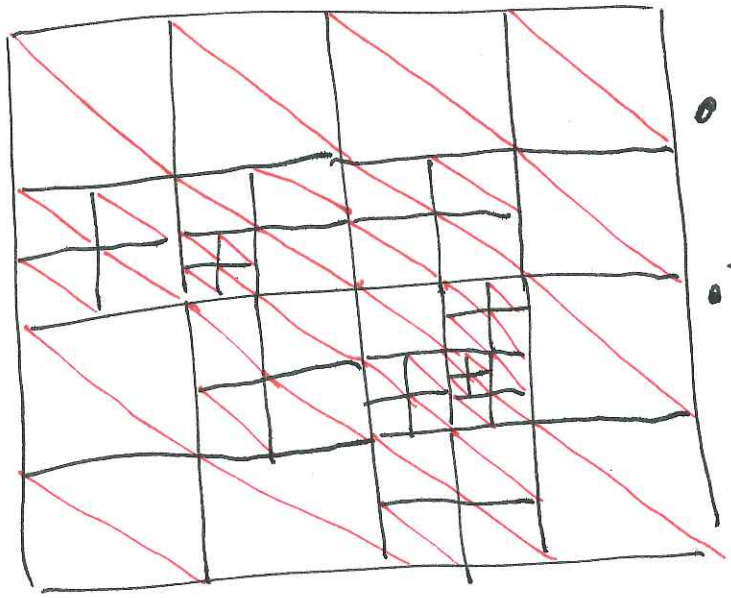
To process a query q, find the predecessor and successor of $z(q)$ in T. The closer of these two points to q is our guess.

To get a guarantee, we use shifts. Store one BBST for each shift. Find the pred. and succ of $z(q)$ in each tree. Return the best (closest to q) among these.

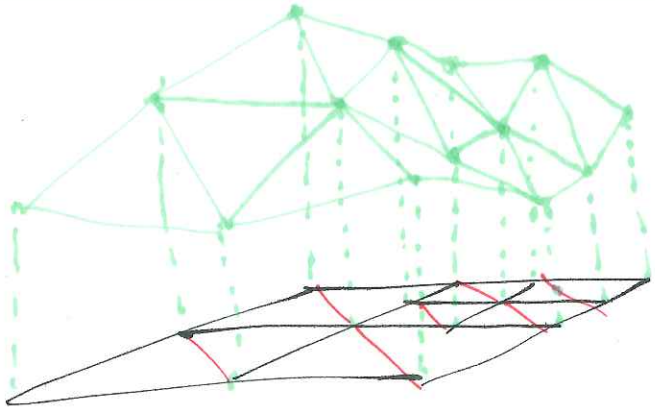
Overview: Geometric Search data structures covered in this course.

Point location in PSLGs (History DAG)
Halfspace Range Counting (Duality, Arrangements, Zone Thm)
Orthogonal Range Search (kd-trees)
Generic Range Search in binary decompositions
Quadtrees (and compressed QTs)
Nearest Neighbor search (also ANN)
Bit twiddling to search a compressed QT.

A QT application: Mesh Generation



- The Quadtree is an adaptive grid.
- It's easy to subdivide into triangles.
- Triangulations can be used to represent functions.



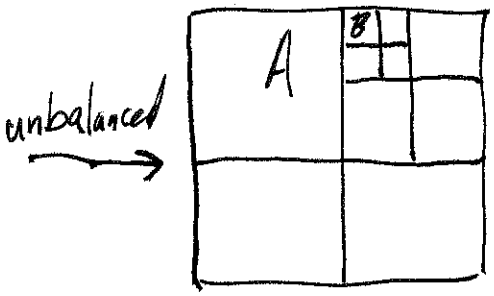
Think of the function as a lifting of the vertices.

It suffices to store f^n at the vertices.

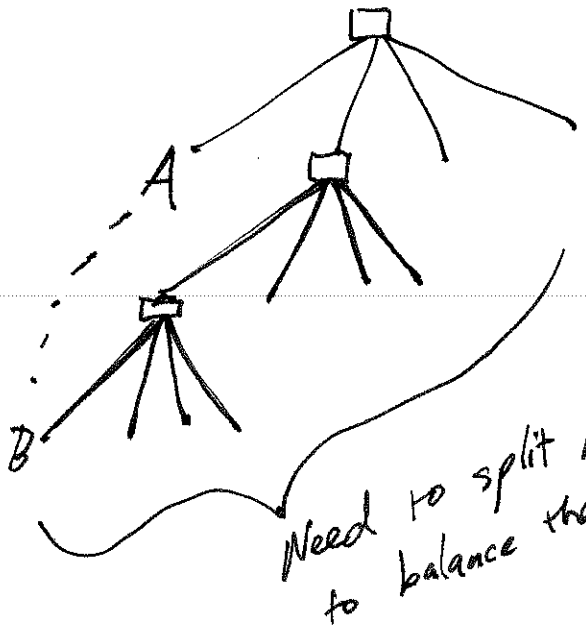
Meshes are essential in physical simulation.

Balancing a Quadtree

* This is nothing like balancing a quadtree



Def A quadtree is balanced if every pair of adjacent leaves have sidelengths that differ by a factor of at most 2.



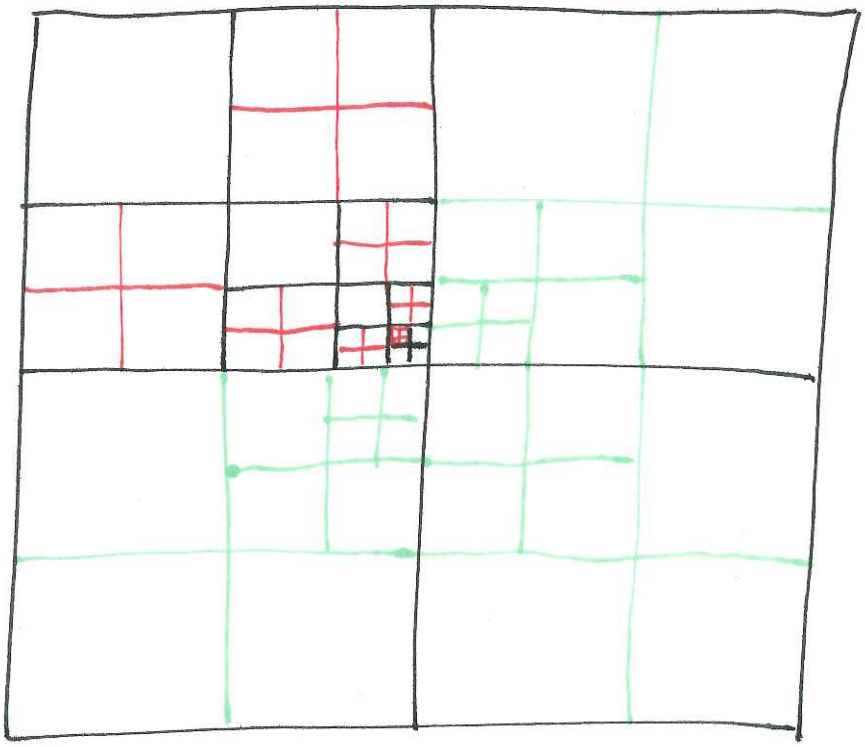
The algorithmic problem:
Split a minimum number of cells to make a QT balanced.

The Big Question

How many splits are required to balance a QT with n nodes.

One new split can cause O(depth) new splits to balance.

[depth = O(n)]



Before Balancing QT T ⟶ After Balancing QT T'

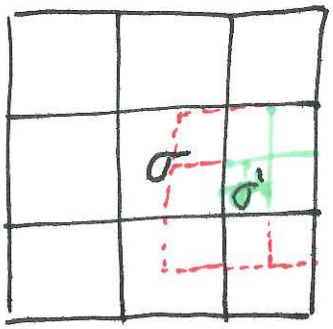
nodes of T are "old" nodes

nodes of $T' \setminus T$ are "new" nodes

Let m be the number of old nodes.

Claim: T' has $O(m)$ nodes.

pf It suffices to show that at most $8m$ splits happen during balancing.



We will show that if σ is split, then one of the eight neighbors of the same size is old. So, each old node will be "charged" for at most 8 splits.

Suppose for contradiction that σ will be split, its same-size neighbors are all new, and σ is the smallest such node. Some other (smaller) node σ' must be split to make σ unbalanced. However, none of σ''s same-size neighbors are old, a contradiction.