

CSSE 230 Day 23

2D Trees

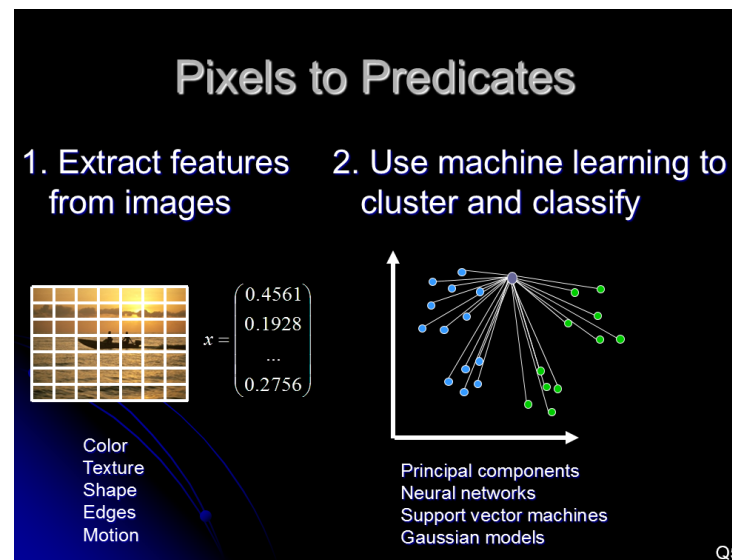
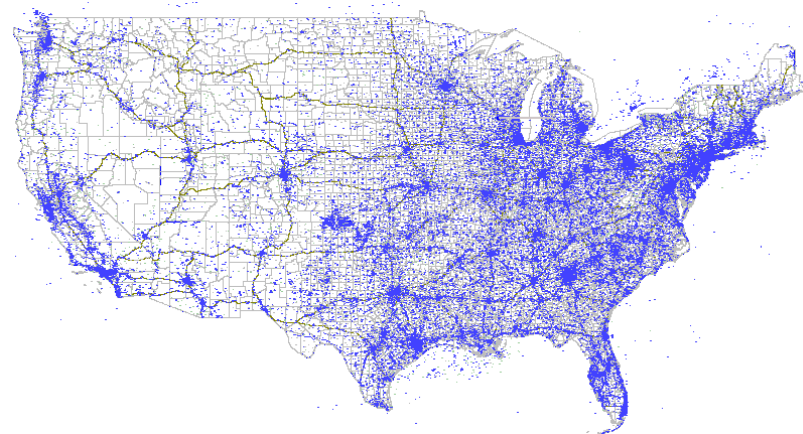
After today, you should be able to ...
... explain insert and nearest-neighbor in 2D trees
... implement these algorithms

Reminders / Announcements

- ▶ EditorTrees evals due tonight

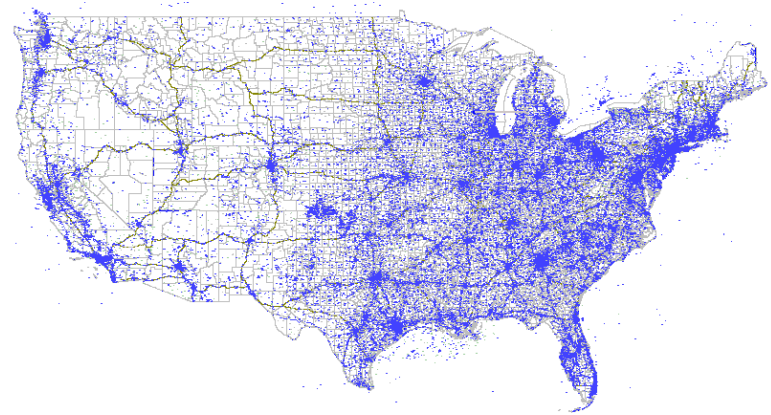
2D Data

- ▶ A large set of (x,y) points
- ▶ Which cell phone tower is closest to me?
- ▶ Which image is most like this one?
- ▶ In general:
 - Find the nearest neighbor of a query point (today).
 - Find or return all points in a certain range.



How to represent 2D data?

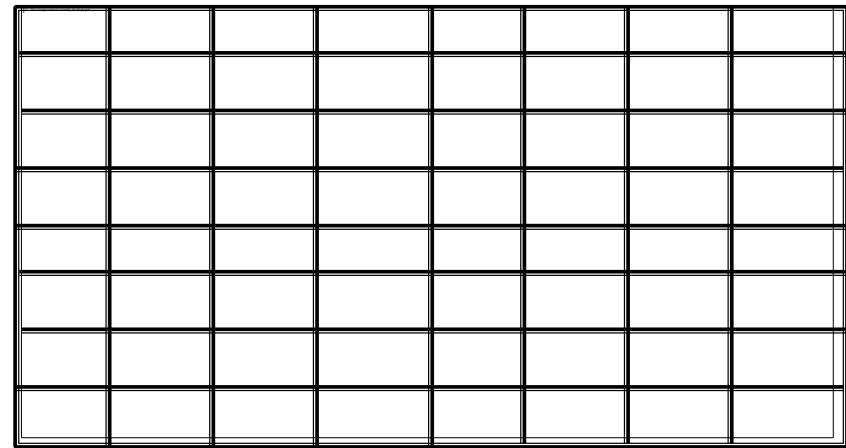
- ▶ List of points. Simple but slow
 - $[p_1, p_2, \dots, p_N]$
 - To find closest to q , find smallest of $\text{dist}(q, p_1), \text{dist}(q, p_2), \dots$



| Representation | Average nearest-nbr efficiency |
|----------------|--------------------------------|
| List of points | N |
| | |
| | |

How to represent 2D data?

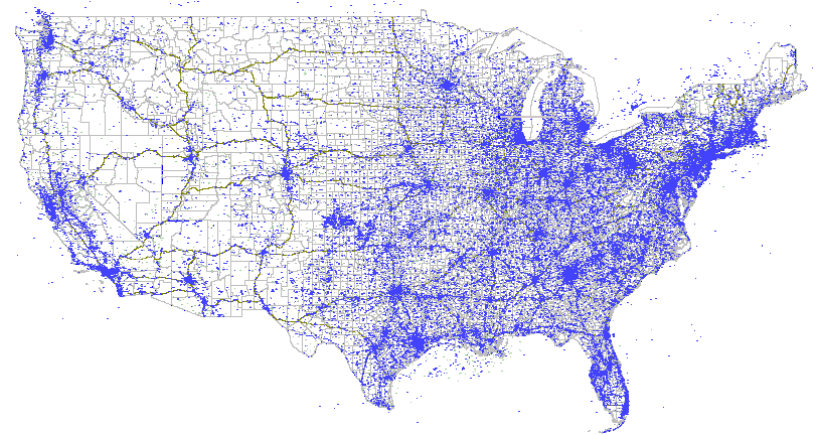
- ▶ List of points. Simple but slow
- ▶ Use a regular grid.
 - 2D array of lists
 - Faster, but which resolution?
 - Example, $M=8$



| Representation | Average nearest-nbr efficiency |
|----------------|--|
| List of points | N |
| Regular grid | $1 + N/M^2$ but space = $N + M^2$, clustering degrades |
| | |

How to represent 2D data?

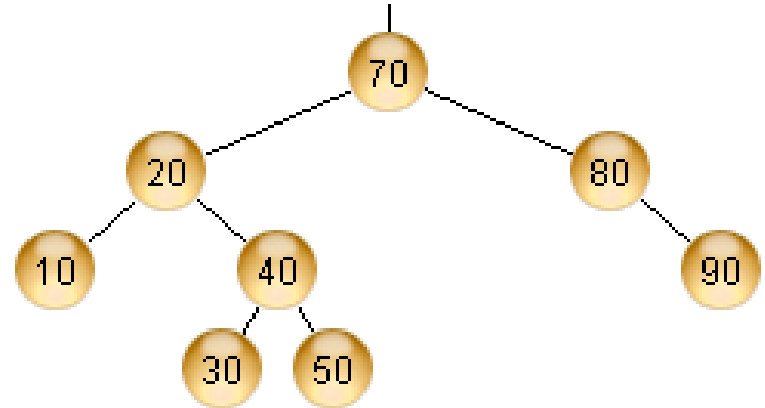
- ▶ List of points. Simple but slow
- ▶ Use a regular grid.
- ▶ ???



| Representation | Average nearest-nbr efficiency |
|----------------|--|
| List of points | N |
| Regular grid | $1 + N/M^2$ but space = $N + M^2$, clustering degrades |
| ??? | $\log N$ |

Binary search trees partition the number line

- ▶ Split at 70
- ▶ Split at 20
- ▶ etc



- ▶ Any value inserted to the left of 30 must be in what range?

You can partition the coordinate plane with a variation of BSTs

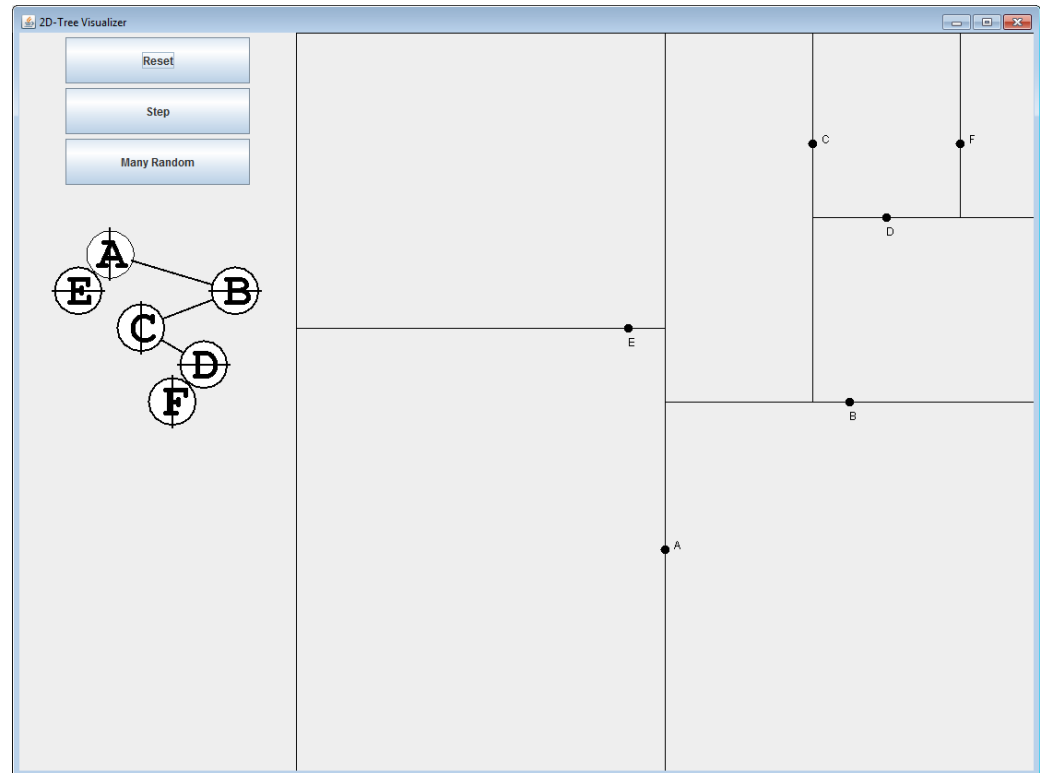
- ▶ Root splits plane using x-coordinate and each level splits the plane in one direction only.
- ▶ Use the insert algorithm to build a tree from points:
 - A (0.5, 0.7)
 - B (0.75, 0.5)
 - C (0.7, 0.15)
 - D (0.8, 0.25)
 - E (0.45, 0.4)
 - F (0.9, 0.15)

You can partition the coordinate plane with a variation of BSTs

- ▶ Root splits plane using x-coordinate and each level splits the plane in one direction only.

- ▶ Use the insert algorithm to build a tree from points:

A (0.5, 0.7)
B (0.75, 0.5)
C (0.7, 0.15)
D (0.8, 0.25)
E (0.45, 0.4)
F (0.9, 0.15)



Nearest neighbor using a 2D Tree

- ▶ Initialize the closest point as the root.
- ▶ Recursively go to each side if it could be closer:
 - To left/top and update closest if one found
 - To right/bottom and update closest if one found
 - When hit a null node, just return
- ▶ New idea: don't always recurse to left/top first. Instead, first recurse to the **same side** as the query point, and then **only** recurse to the other side *if it could* yield a closer point
 - To do this, each node also stores the bounds of rectangle it is part of
 - I give you a Rectangle class with a method to find closest distance from a point that that rectangle.

Nearest neighbor using a 2D Tree

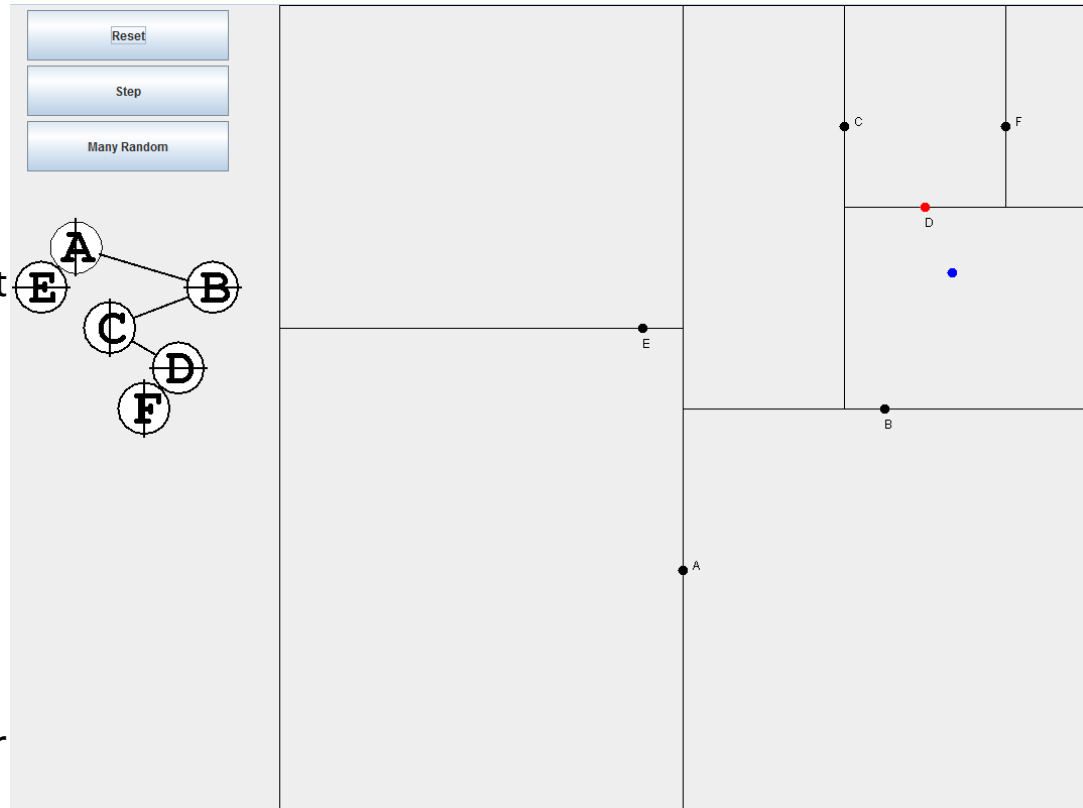
Initialize the closest point as the root.

Recursively go to each side if it could be closer:

- To left/top and update closest if one found
- To right/bottom and update closest if one found
- When hit a null node, just return

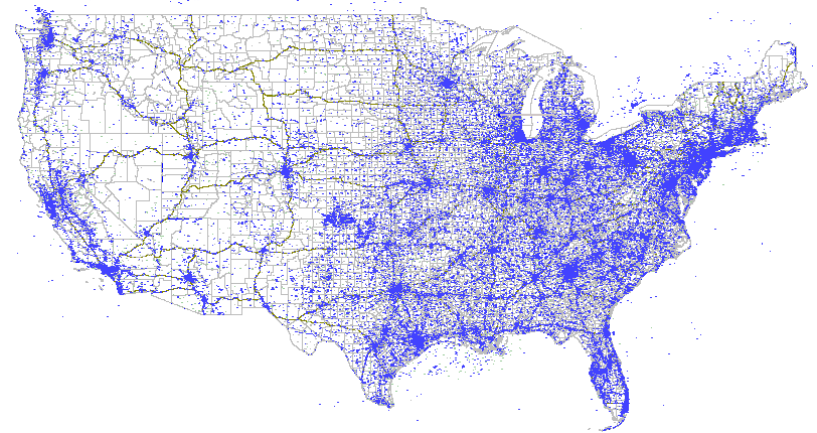
New idea: don't always recurse to left/top first. Instead, recurse to the same side as the query point, and then only recurse to the other side if it *could* yield a closer point

- To do this, each node will also store the bounds of rectangle it is part of



How to represent 2D data?

- ▶ List of points. Simple but slow
- ▶ Use a regular grid.
- ▶ Use a 2D tree
 - You can find the nearest neighbor efficiently



| Representation | Average nearest-nbr efficiency |
|----------------|---|
| List of points | N |
| Regular grid | $1 + N/M^2$ but space = $N/M^2 + 1$, clustering degrades |
| 2D tree | $\log N$ |

2D Trees are useful

- ▶ Questions for thought:
 - How would you build a 3D tree?
 - ... a kD tree for arbitrary dimension D ?
- ▶ Summarize now
- ▶ Assignment for this week:
 - Implement `insert(Point)`, `contains(Point)`, and `nearest(Point)` using a 2D tree