

Data Structures

KD Tree

CS 225

September 23, 2026

Brad Solomon



UNIVERSITY OF
ILLINOIS
URBANA - CHAMPAIGN

Department of Computer Science

Informal Early Feedback Released!

A larger anonymous survey designed to give feedback to staff

Collective extra credit opportunity!

Studying what aspects of class are most / least helpful

Exam 2 (9/30 — 10/02)

Autograded MC and one coding question

Manually graded short answer prompt

Practice exam will be released on PL

Topics covered can be found on website

Register now

<https://courses.engr.illinois.edu/cs225/exams/>

Exam 2 (9/30 — 10/02)

Lab for this week is **EXAM 2 REVIEW SESSION**

Strongly encouraged to attend!

Extra credit lab releases this week (lab_trees)!

Learning Objectives

Explore the need and use of range search in 1D and 2D+

Introduce the KD Tree

Extensions to Tree Search

What is a situation where you want to find an item...

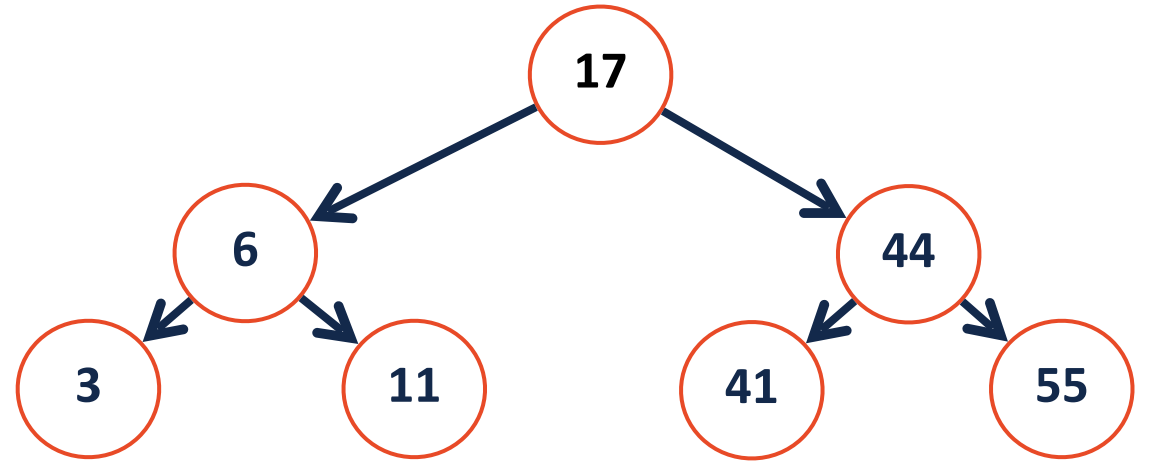
...or its nearest neighbor?

What is a situation where you want to find all items in a range?

Nearest Neighbor Tree Search

Consider a collection of points on a 1D line: $\mathbf{p} = \{p_1, p_2, \dots, p_n\}$

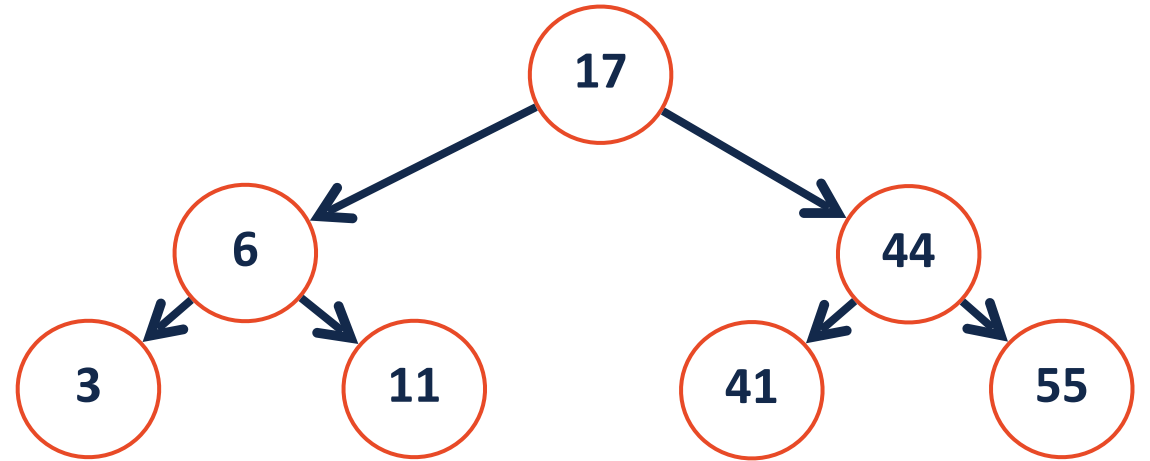
If I wanted to find the closest item to 16, what would I do?



Range-based Tree Search

Consider a collection of points on a 1D line: $\mathbf{p} = \{p_1, p_2, \dots, p_n\}$

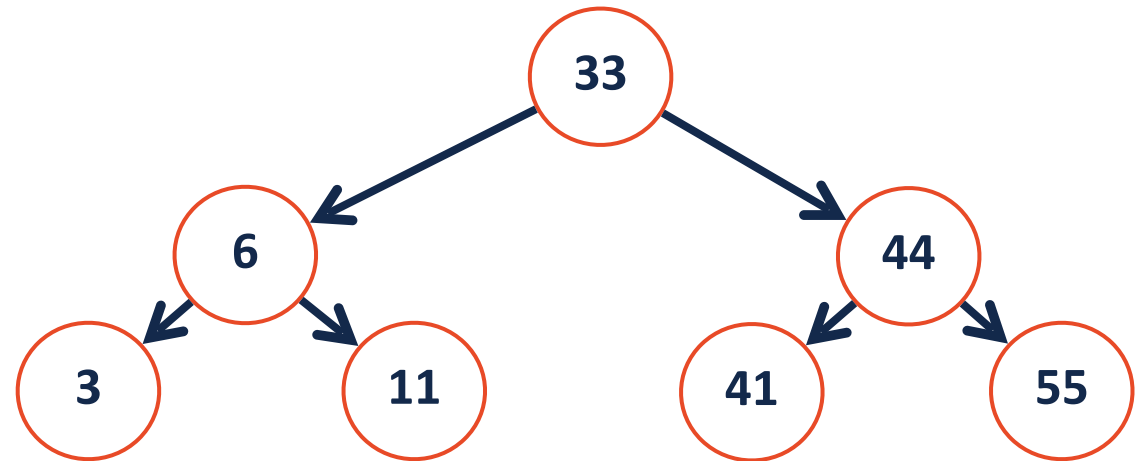
If I want to find all values between $[A, B]$, how could I implement this?



Range-based Tree Search

Consider a collection of points on a 1D line: $\mathbf{p} = \{p_1, p_2, \dots, p_n\}$

If I want to find all values between $[A, B]$, how could I implement this?

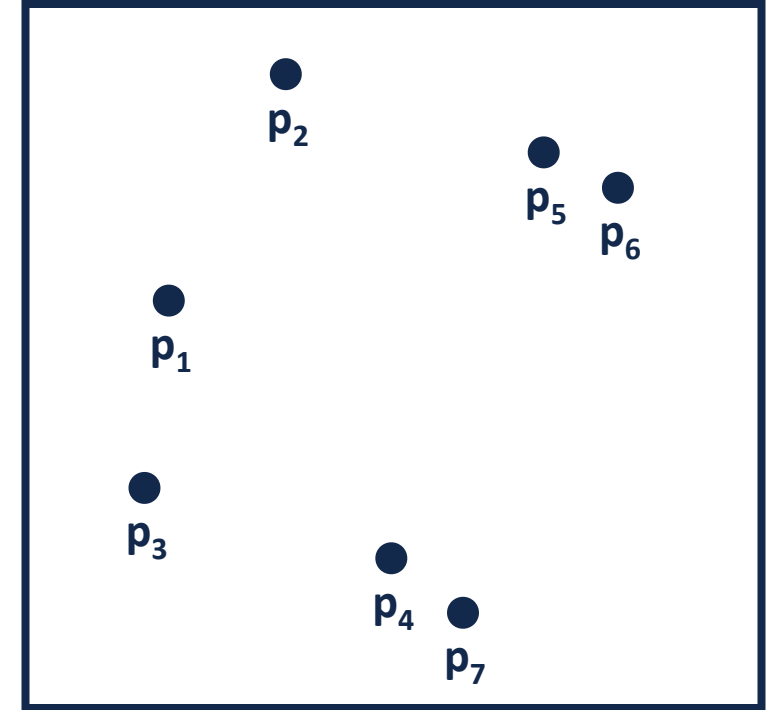


```
1  
2 for(auto it = myMap.lower_bound(A); it != myMap.upper_bound(B); ++it) {  
3  
4 // Do Stuff  
5 }
```

Nearest-Neighbor 2D Search

Consider points in 2D: $\mathbf{p} = \{p_1, p_2, \dots, p_n\}$

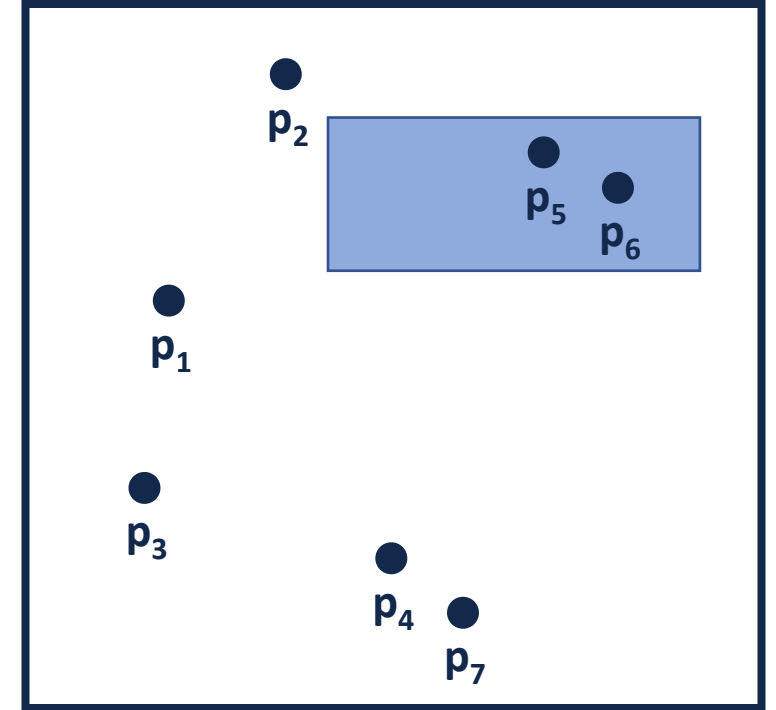
What is nearest point to (x_1, y_1) ?



Range-based 2D Search

Consider points in 2D: $\mathbf{p} = \{p_1, p_2, \dots, p_n\}$

What points in rectangle $[(x_1, y_1), (x_2, y_2)]$?



Constructing a multi-dimensional tree



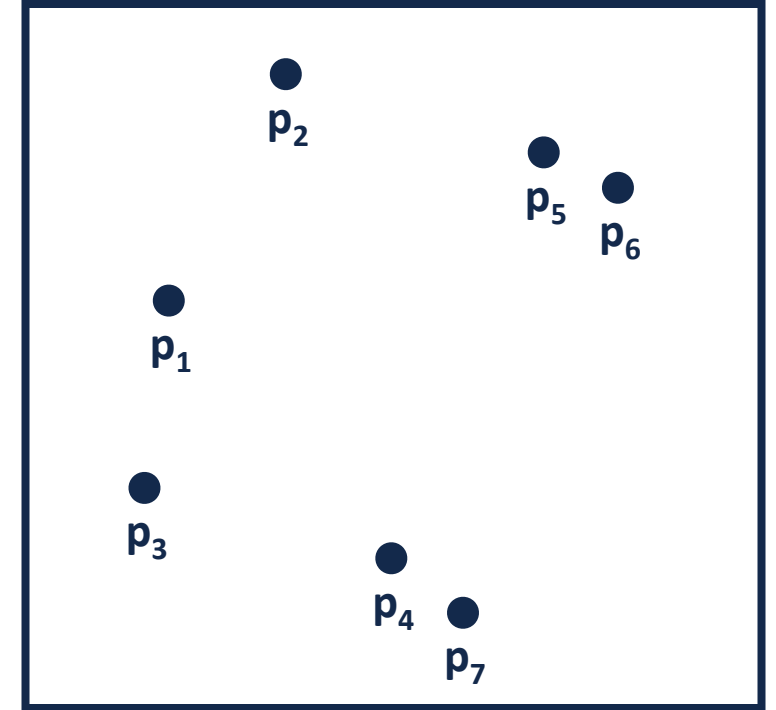
Naive walks or 1D trees can't help us here! Need something that...

1. Starts with a binary search in k dimensions
2. Tries to enforce a balanced tree
3. At each node, can make a branch decision in $O(1)$ time

Constructing a multi-dimensional tree

Consider points in 2D: $\mathbf{p} = \{p_1, p_2, \dots, p_n\}$

Tree Construction:

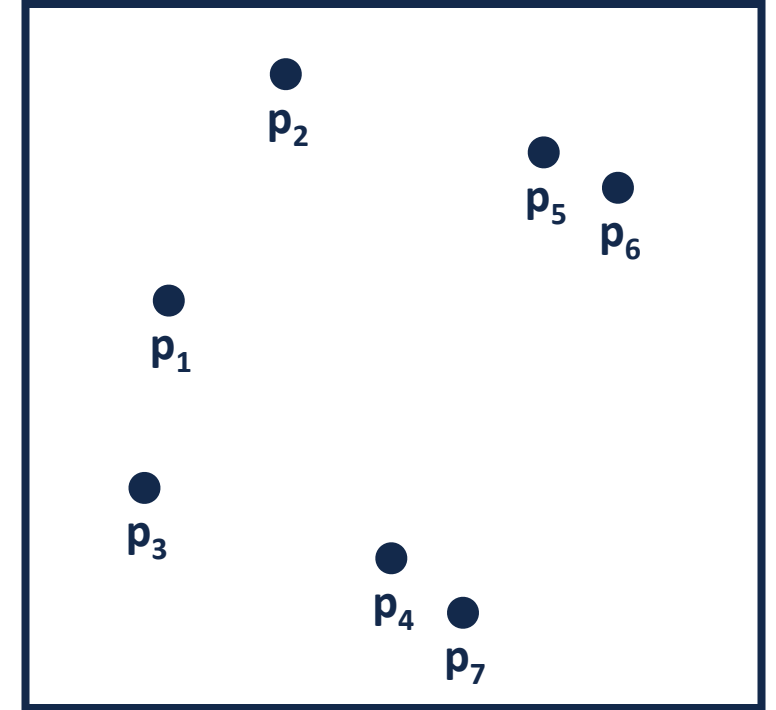


Constructing a multi-dimensional tree

Consider points in 2D: $\mathbf{p} = \{p_1, p_2, \dots, p_n\}$

Tree Construction:

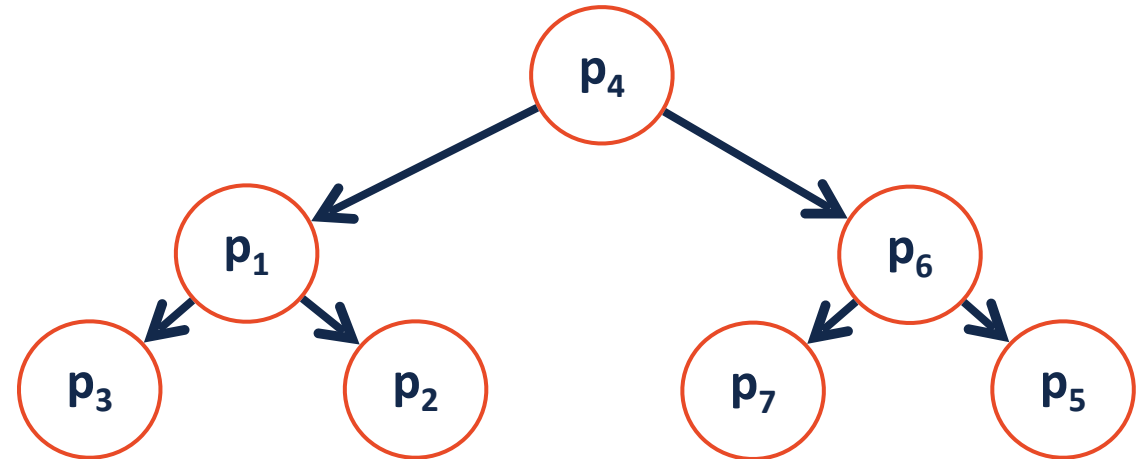
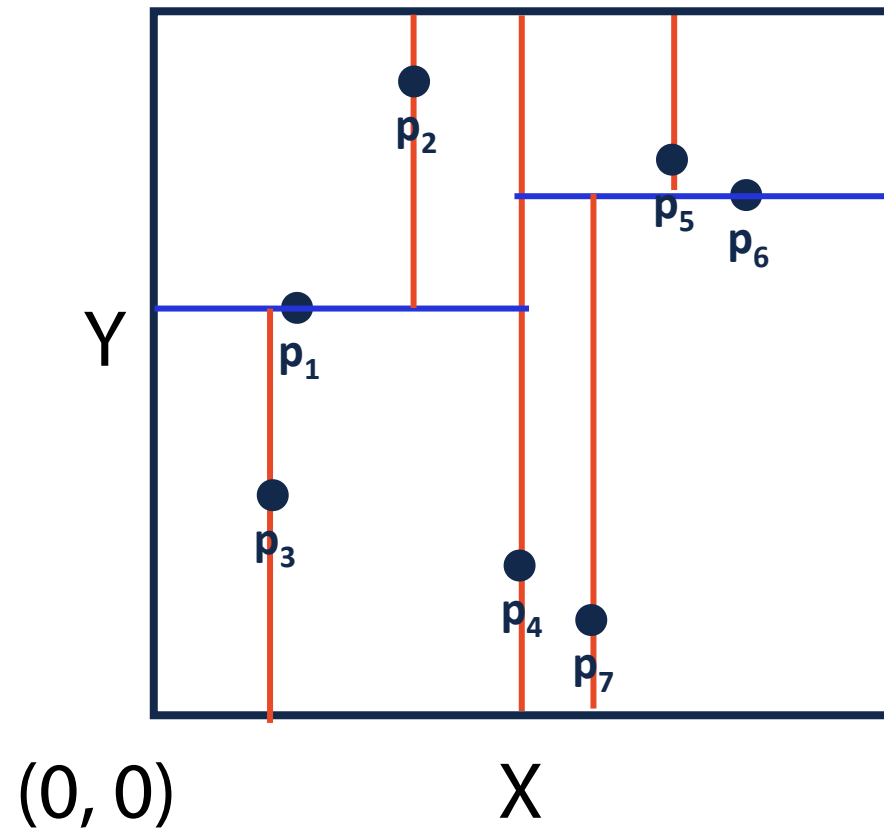
1. Find **median point** along a particular dim
2. Recurse left and right, **swapping** dimension



Constructing a multi-dimensional tree

We now have a tree which partitions a space into 'cells'

... Why did we choose to split on median point?

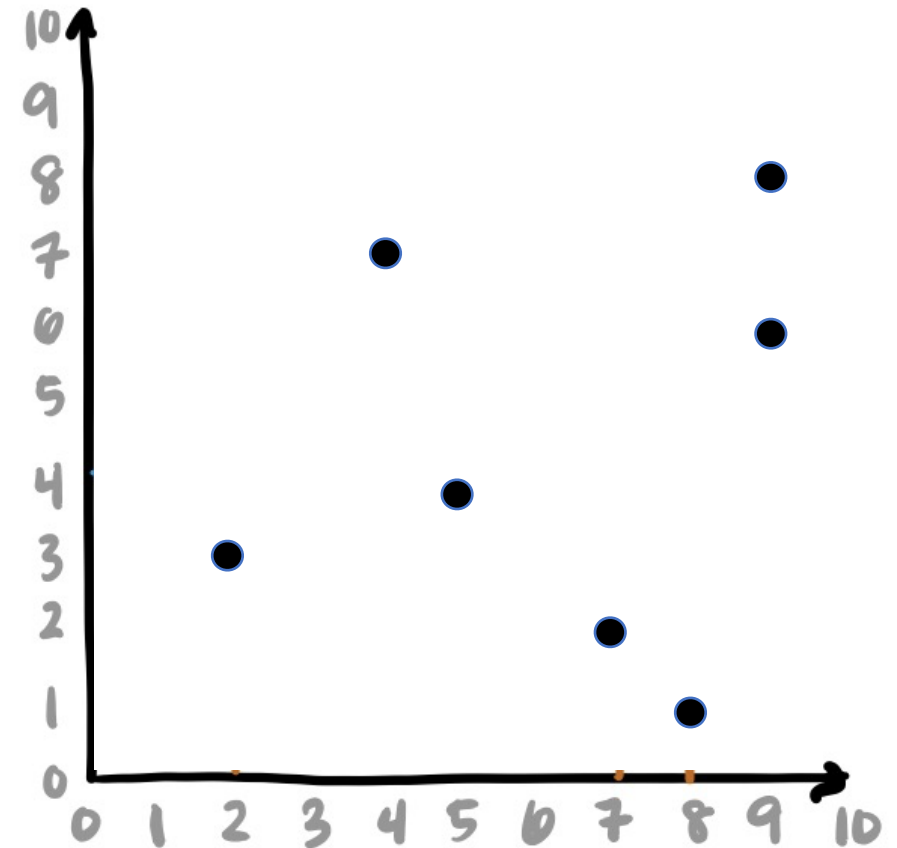


Nearest Neighbor: k-d tree



This construction strategy builds a **balanced** k-dimension (k-d tree):

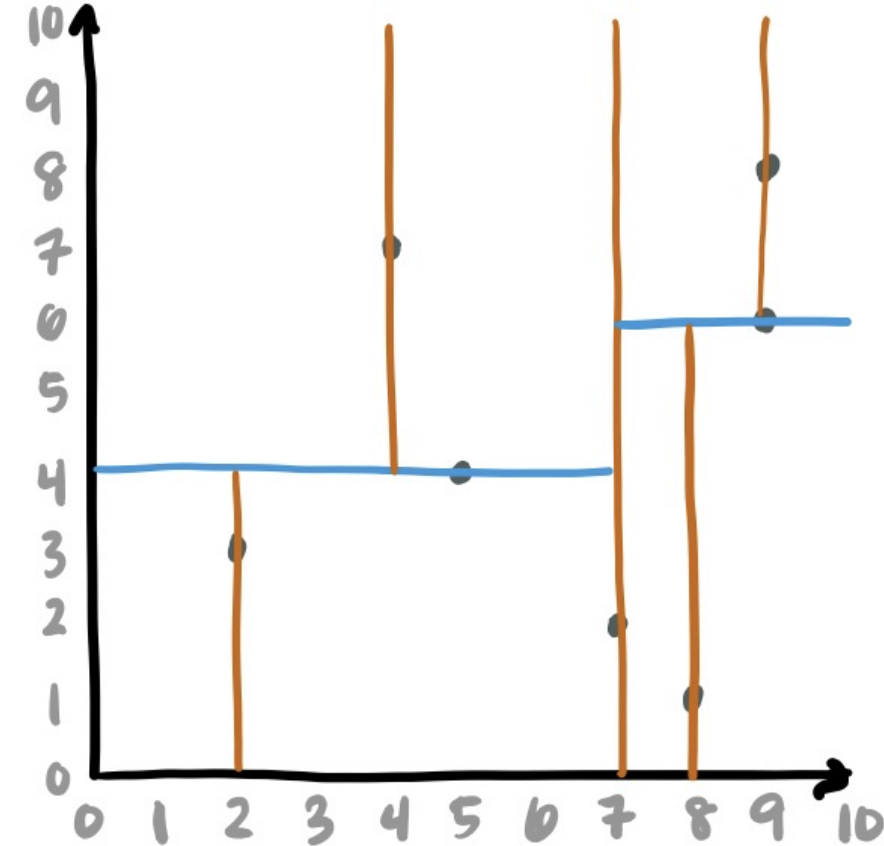
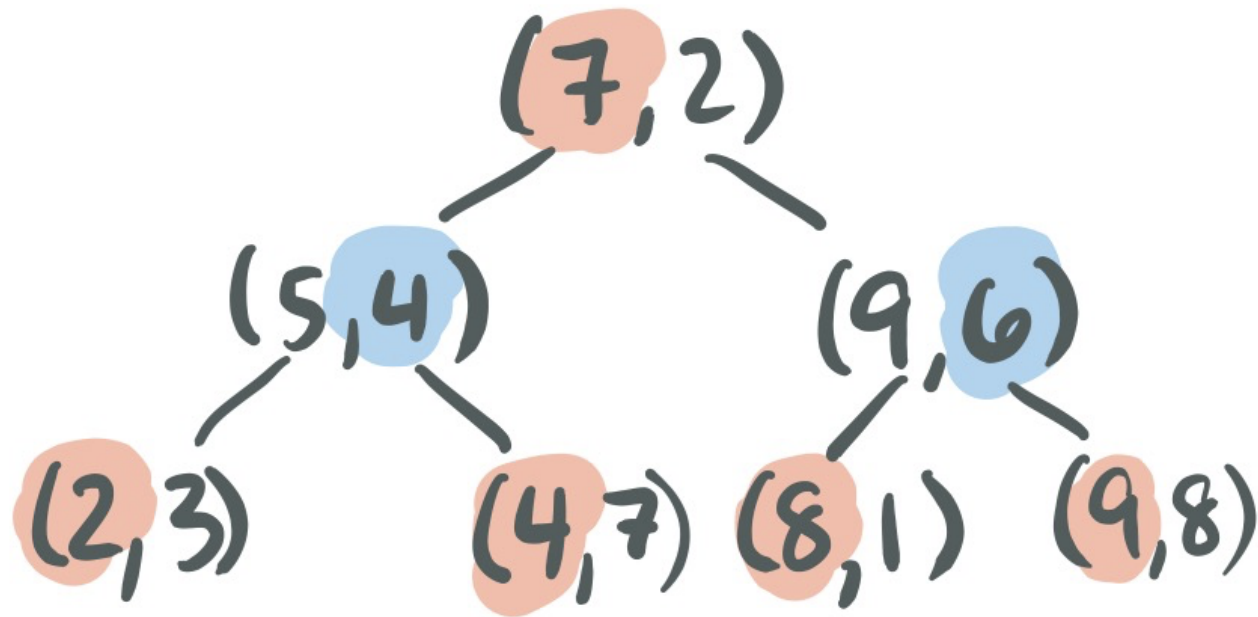
$(7, 2)$, $(5, 4)$, $(9, 6)$, $(4, 7)$, $(2, 3)$, $(8, 1)$, $(9, 8)$



Nearest Neighbor: k-d tree

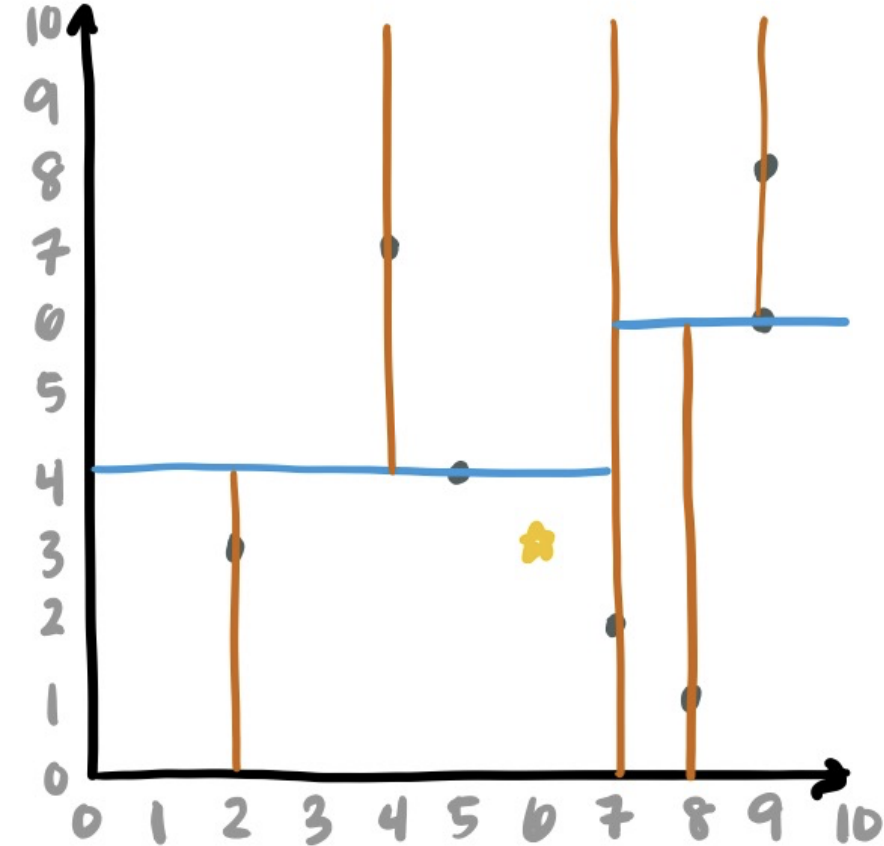
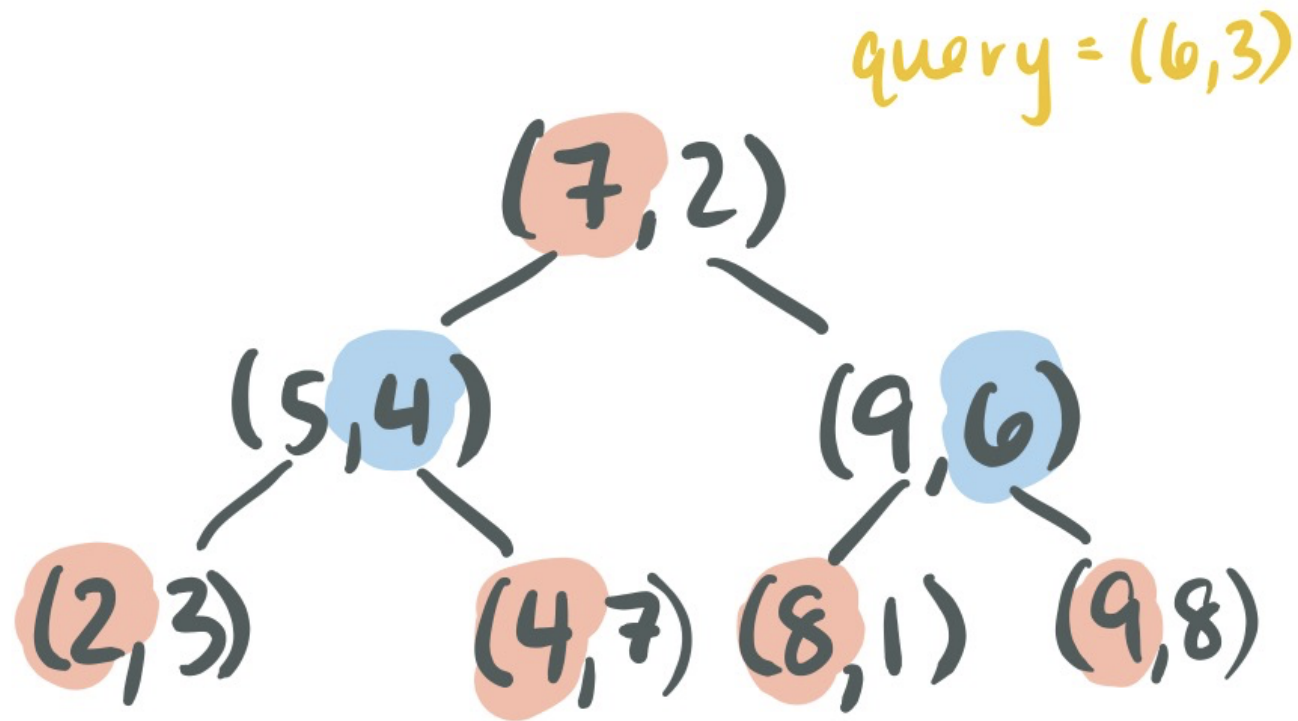


Claim: This tree allows sorted search in multiple dimensions!



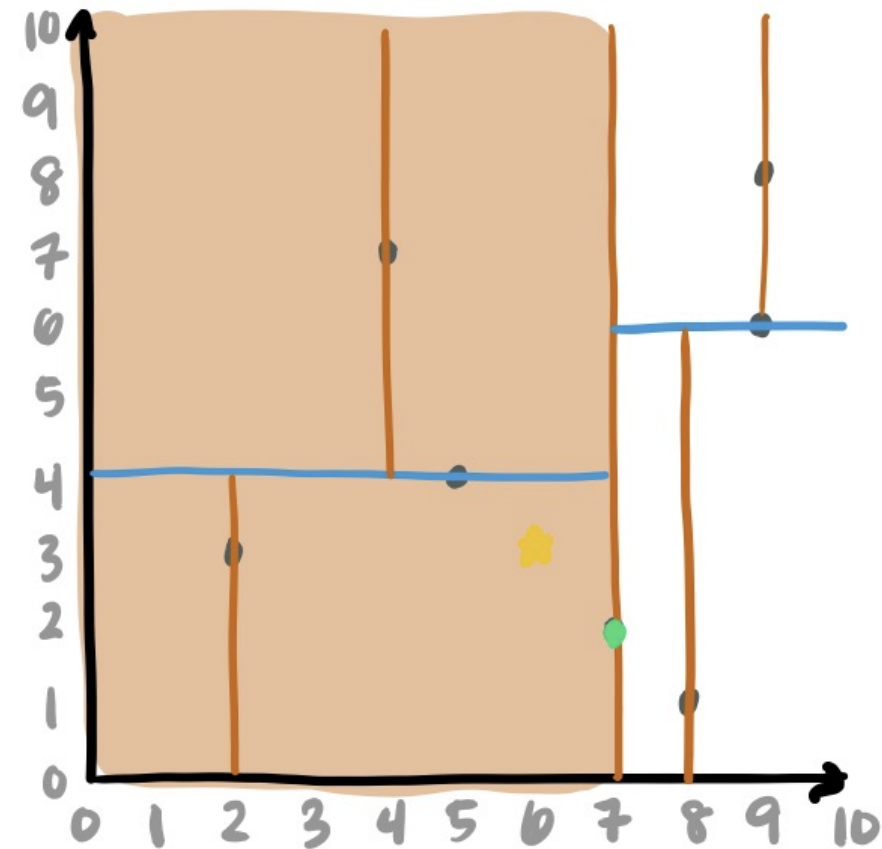
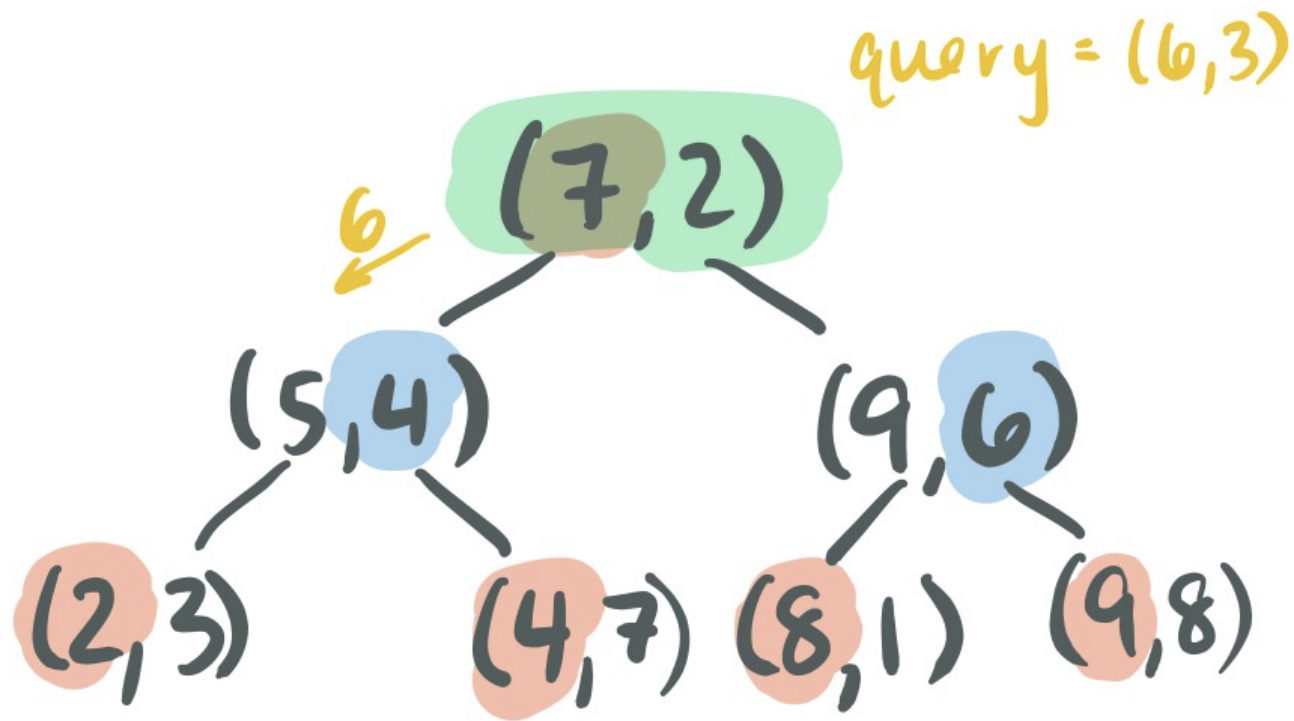
Nearest Neighbor: k-d tree

When querying a k-d tree, it acts like a BST* at first...



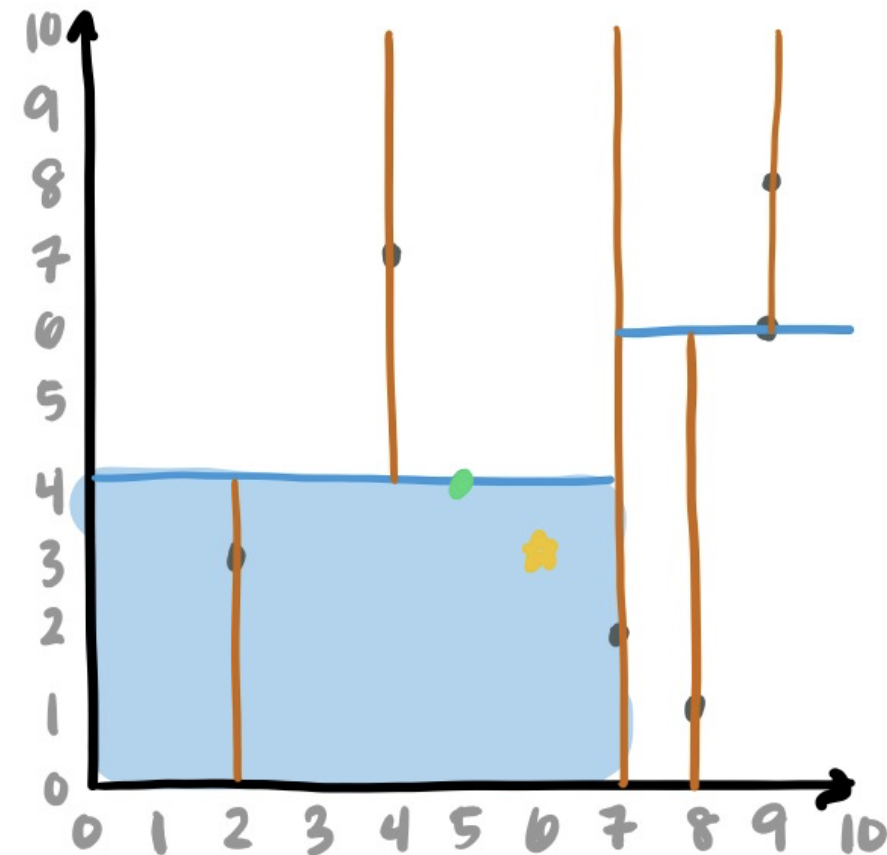
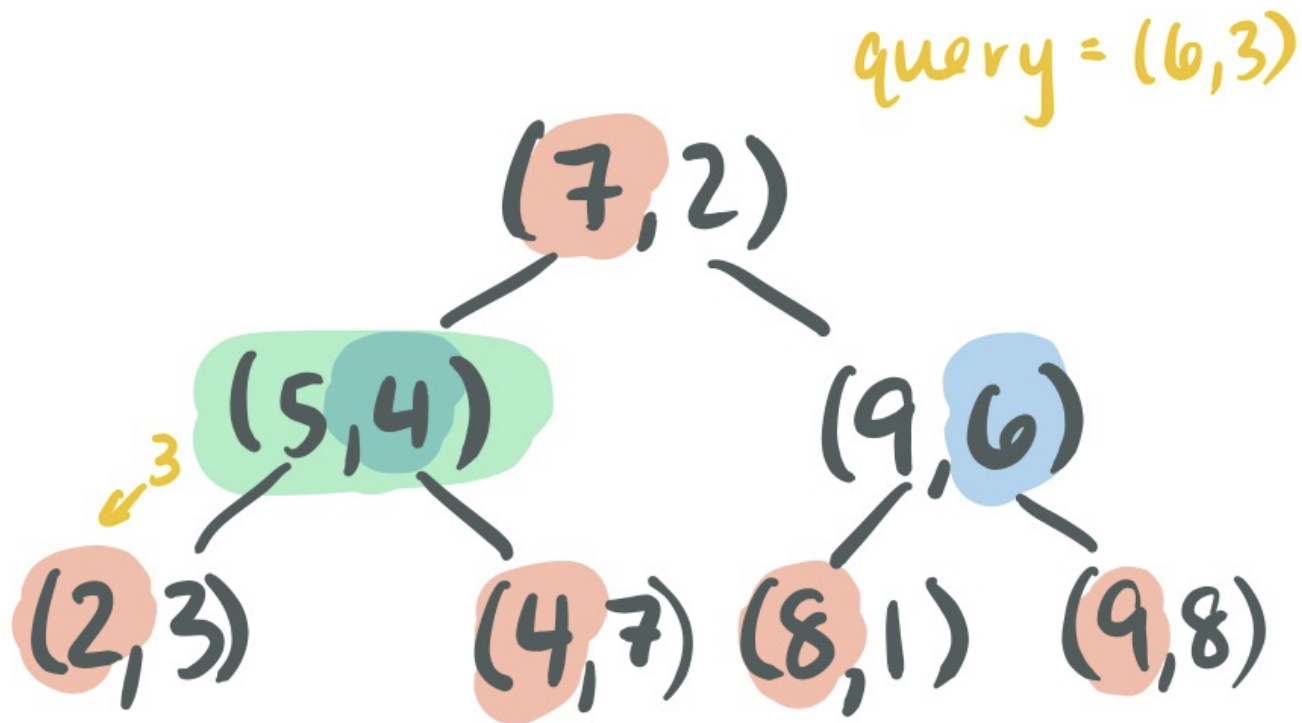
Nearest Neighbor: k-d tree

When querying a k-d tree, it acts like a BST* at first...



Nearest Neighbor: k-d tree

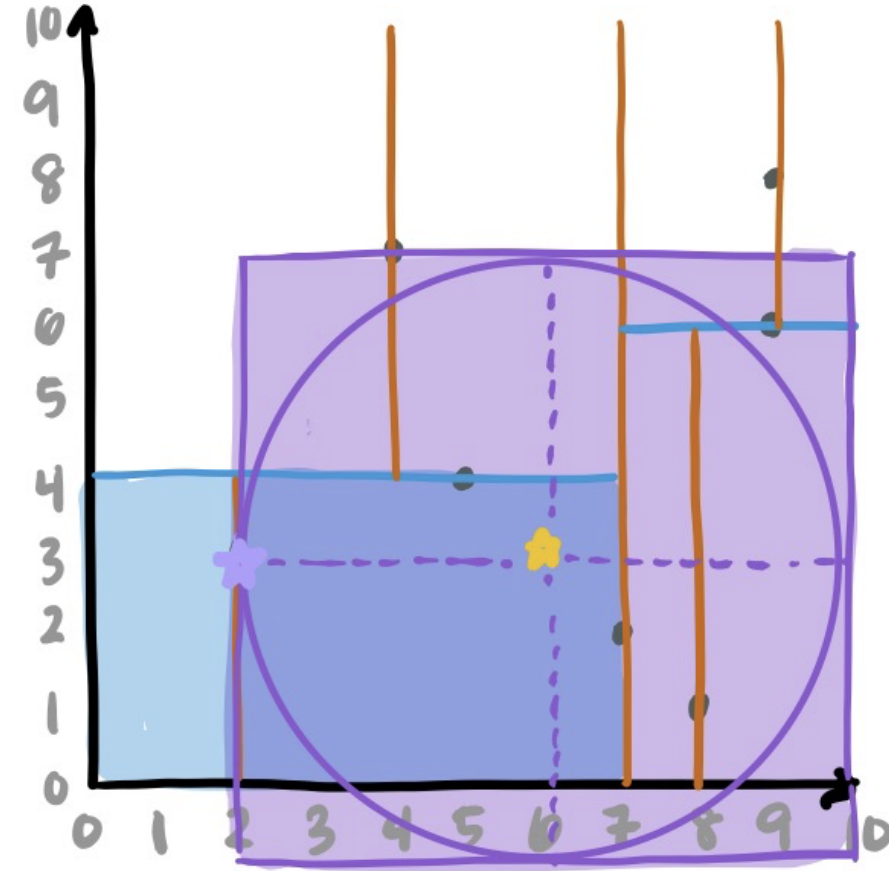
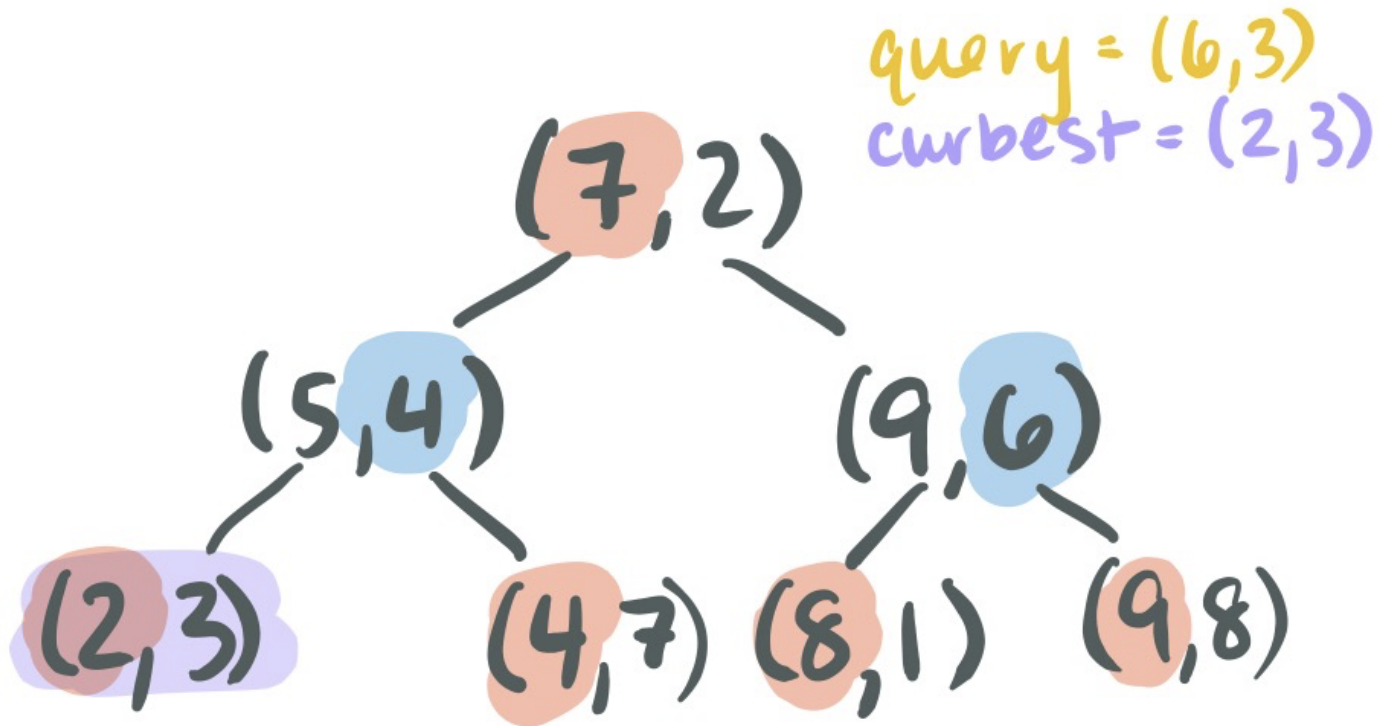
When querying a k-d tree, it acts like a BST* at first...



Nearest Neighbor: k-d tree

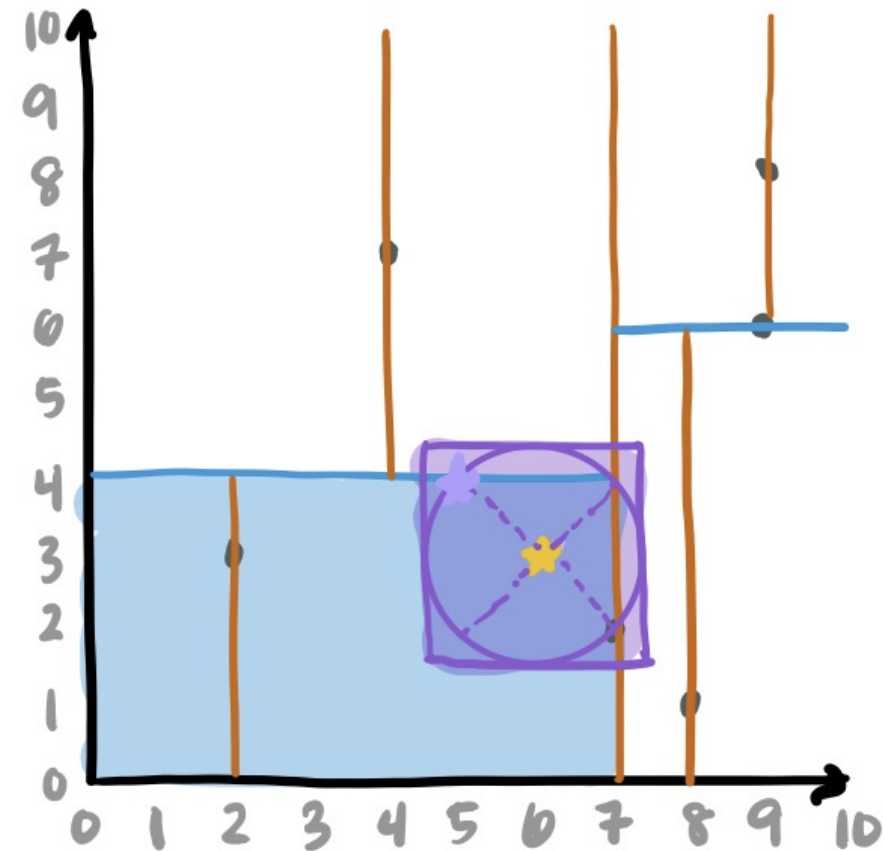
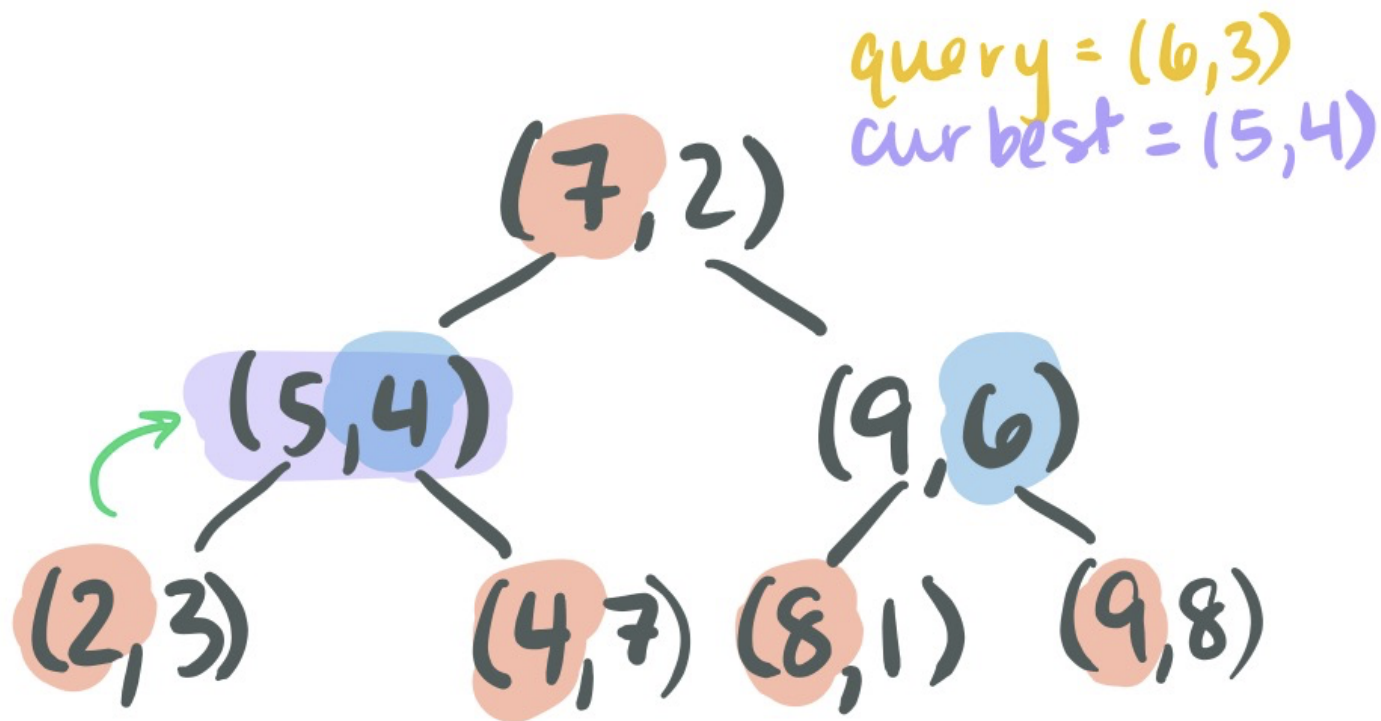
When querying a k-d tree, it acts like a BST* at first...

... But if we don't find exact match, have to find nearest neighbor

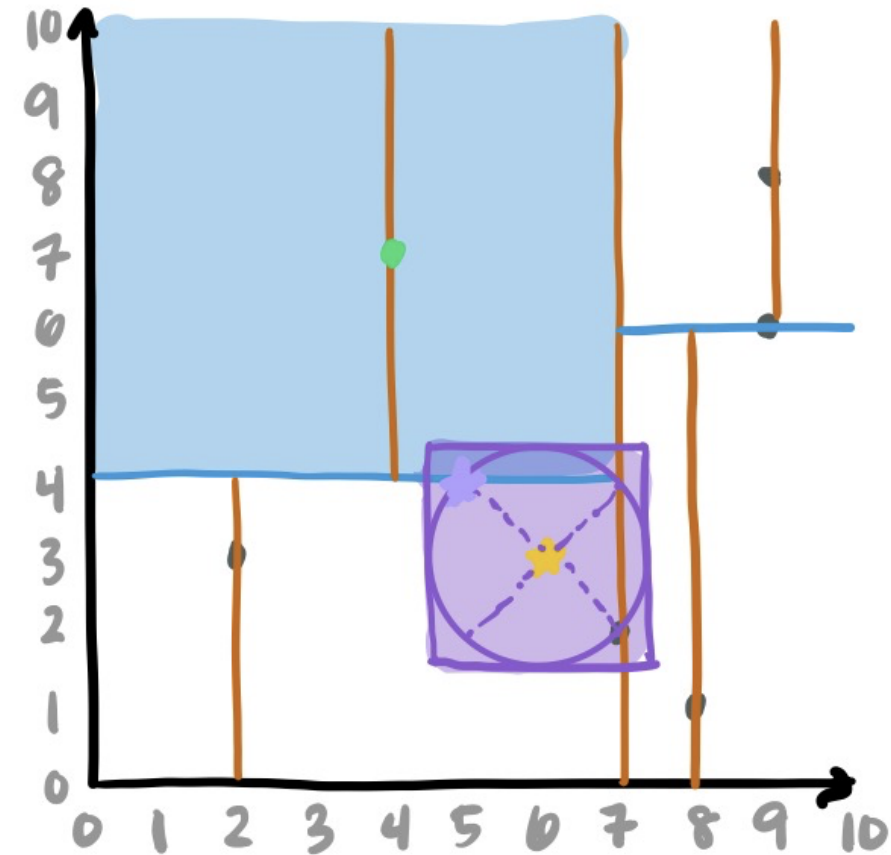
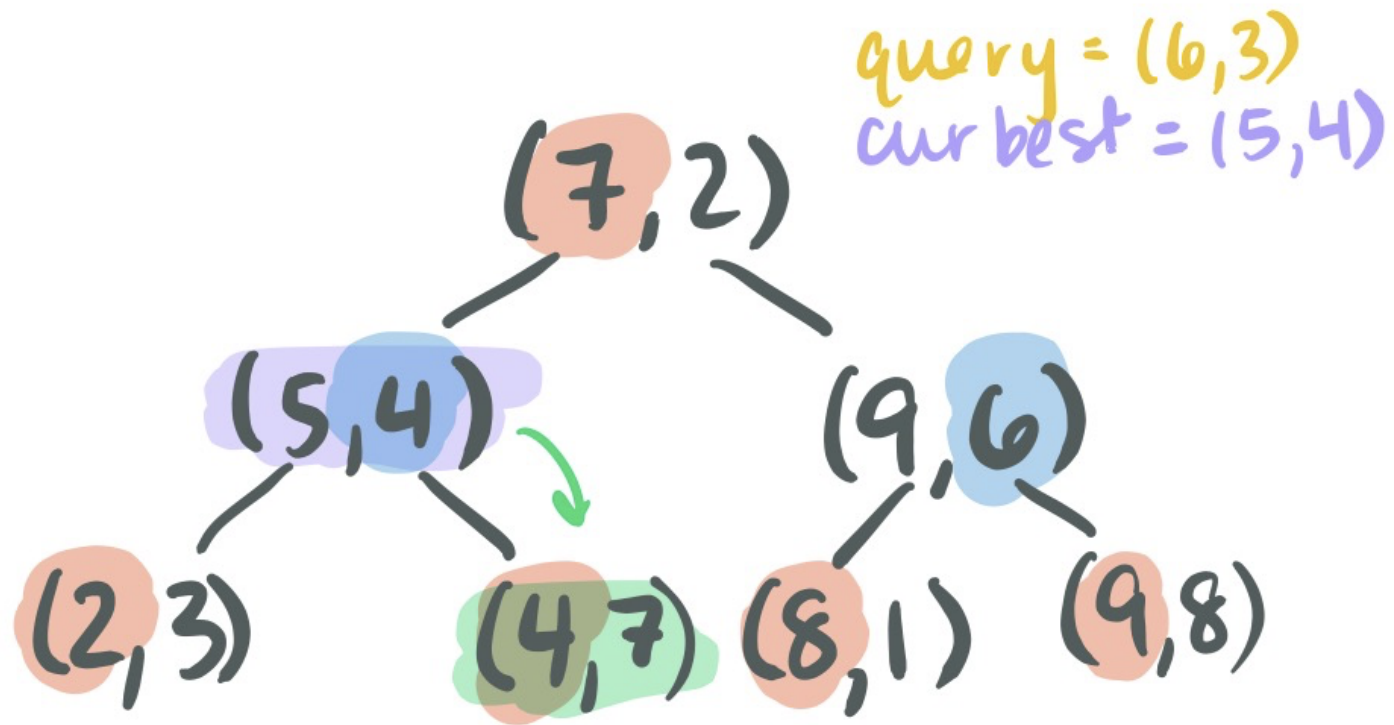


Nearest Neighbor: k-d tree

Backtracking: start recursing backwards -- store "best" possibility as you trace back

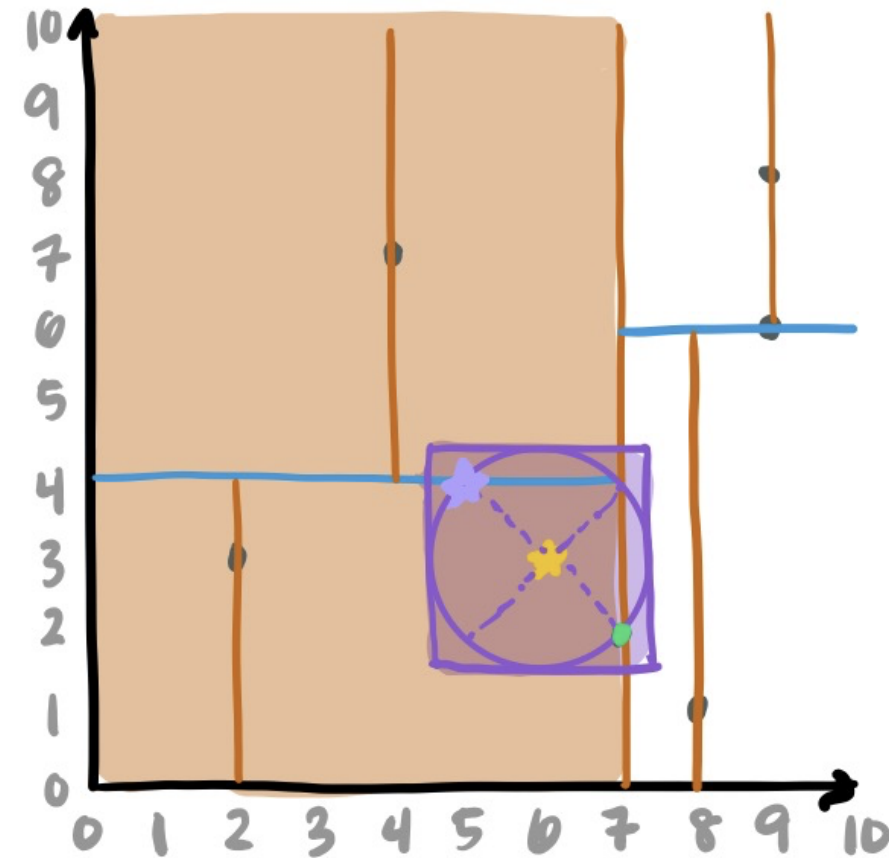
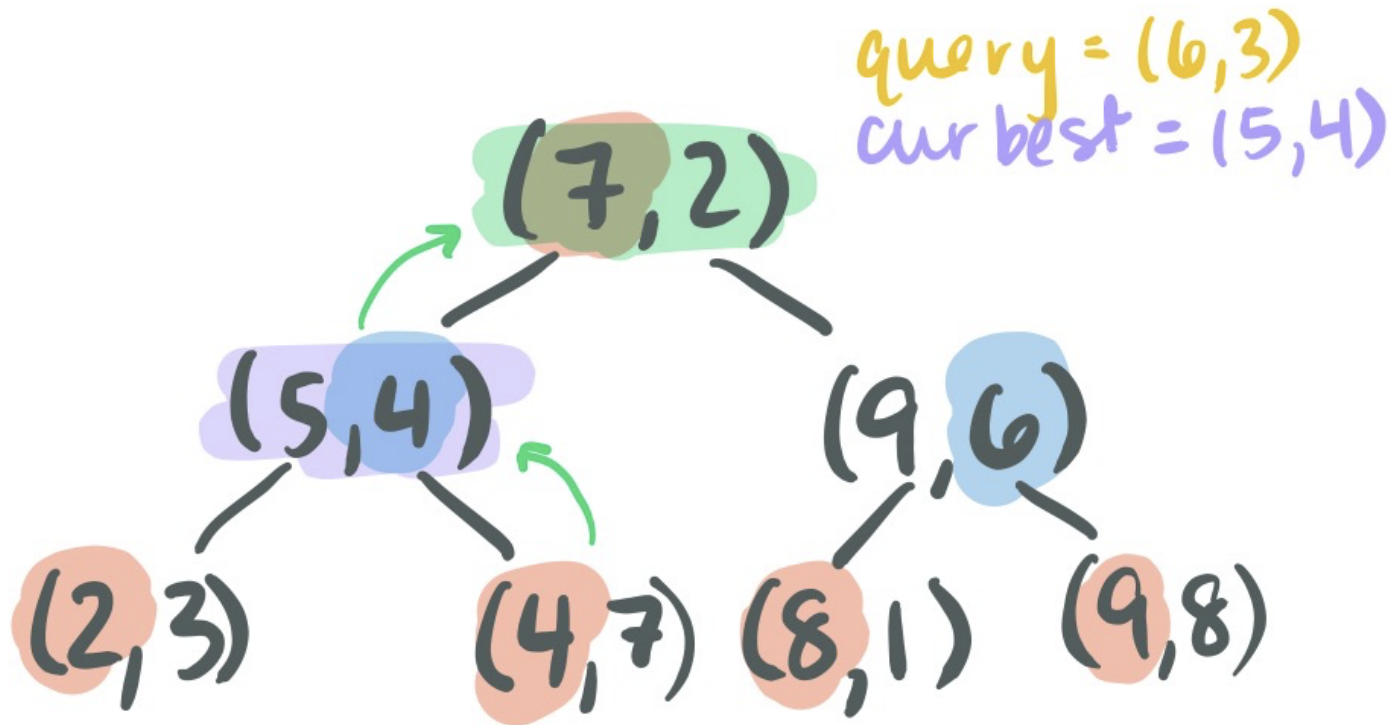


Nearest Neighbor: k-d tree



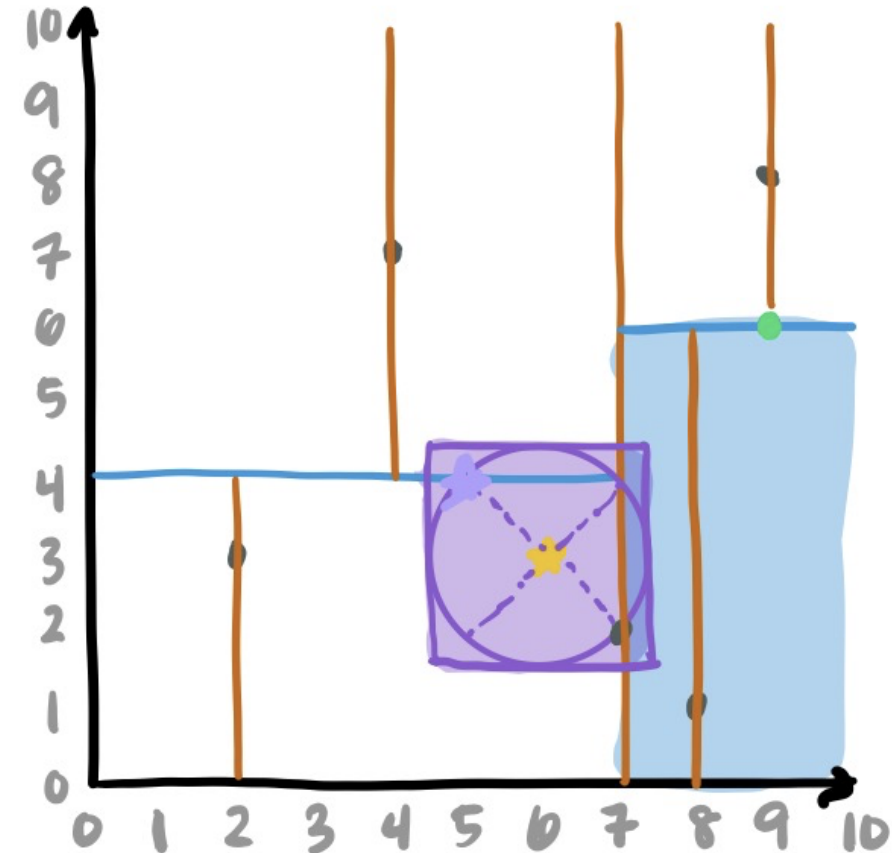
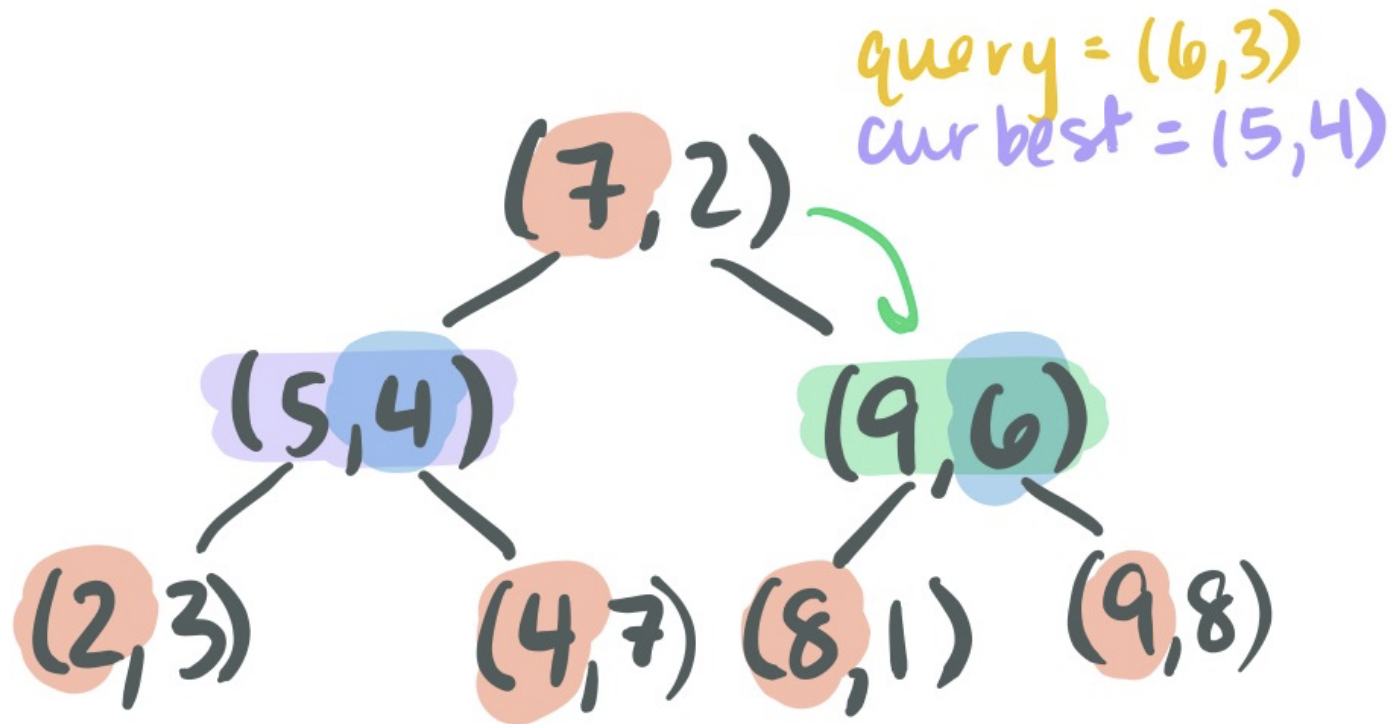
Nearest Neighbor: k-d tree

On ties, use smallerDimVal to determine which point remains curBest

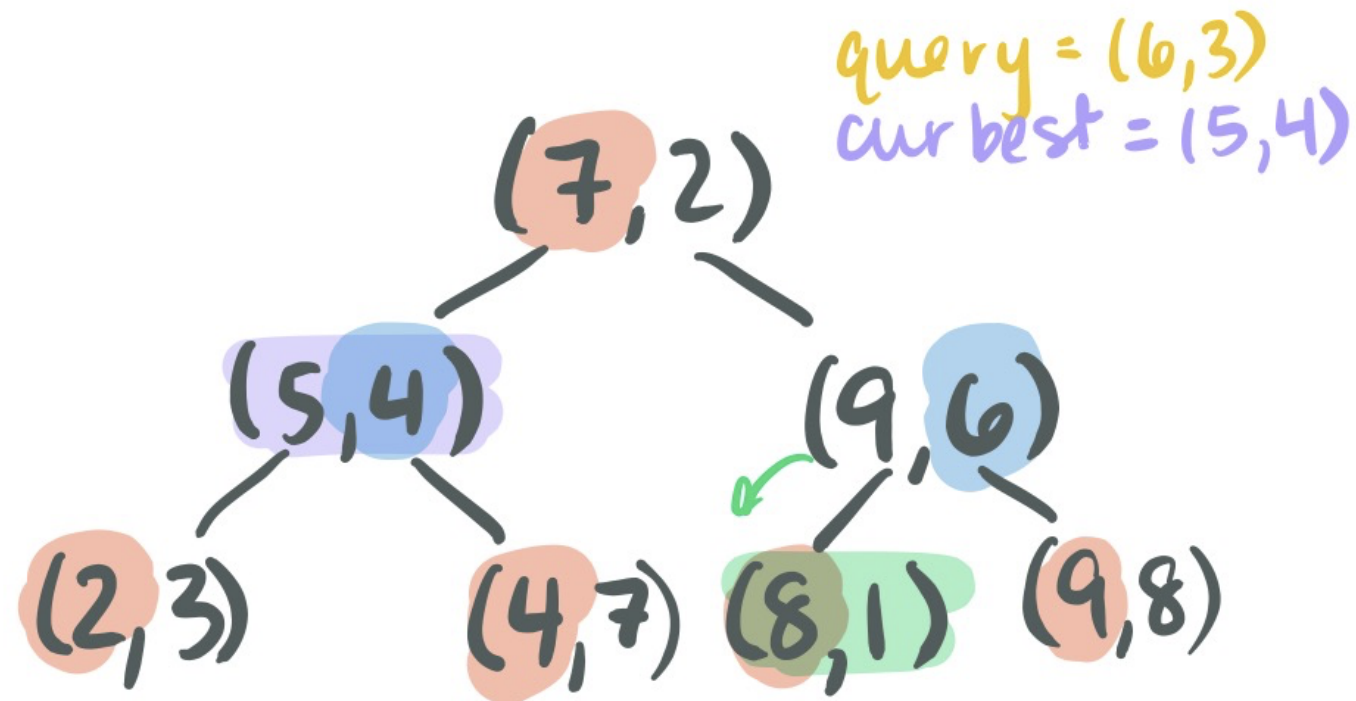


Nearest Neighbor: k-d tree

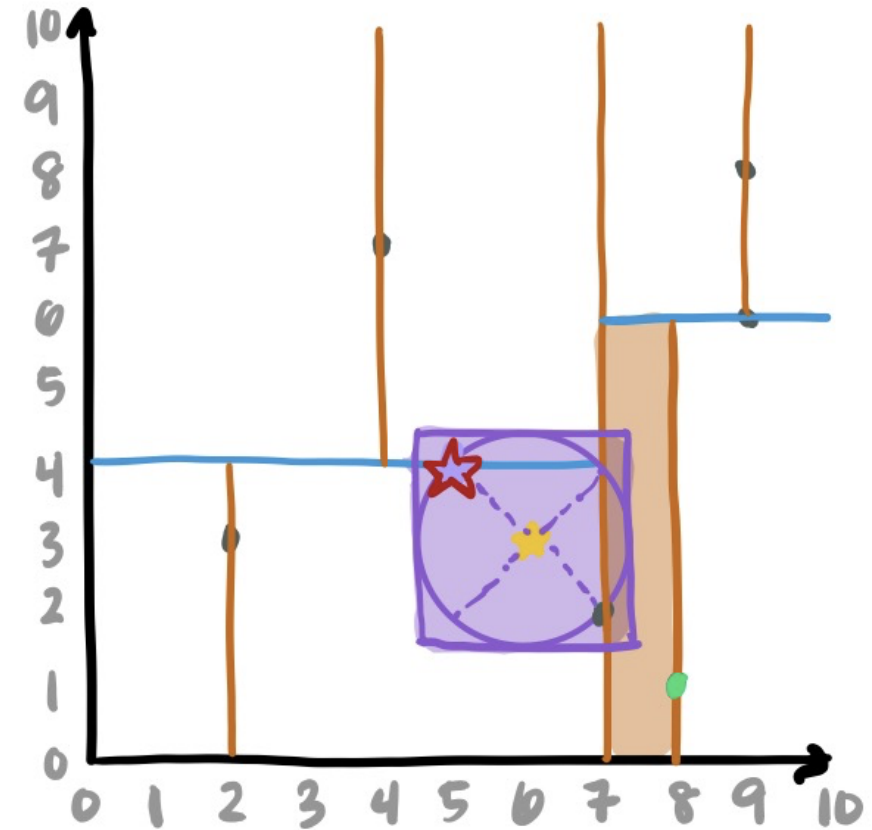
Why do we need to explore this subtree?



Nearest Neighbor: k-d tree



BEST: (5,4)



K-D Tree Runtime Efficiency

Because our search radius can touch every point, search is $O(N)$!

... But if my data is well-formed (uniform distribution), we can make an argument that the average-case search is actually $O(\log n)$

Intuition: More points == more partitions == smaller search radius

MP_Mosaics shows off this theory vs practice!

Next Class...

There's an important algorithm we are neglecting...

Consider points in 2D: $\mathbf{p} = \{p_1, p_2, \dots, p_n\}$

Tree Construction:

1. Find **median point** along a particular dim
2. Recurse left and right, swapping dimension

We will also need to understand a few subtle C++ concepts!