CS 225

Data Structures

March 6 – kd-Tree and Btrees Intro

Wade Fagen-Ulmschneider, Craig Zilles

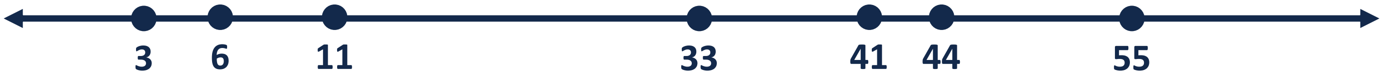
Range-based Searches

Balanced BSTs are useful structures for range-based and nearest-neighbor searches.

Q: Consider points in 1D: **p = {p_1, p_2, …, p_n}.**

…what points fall in [11, 42]?

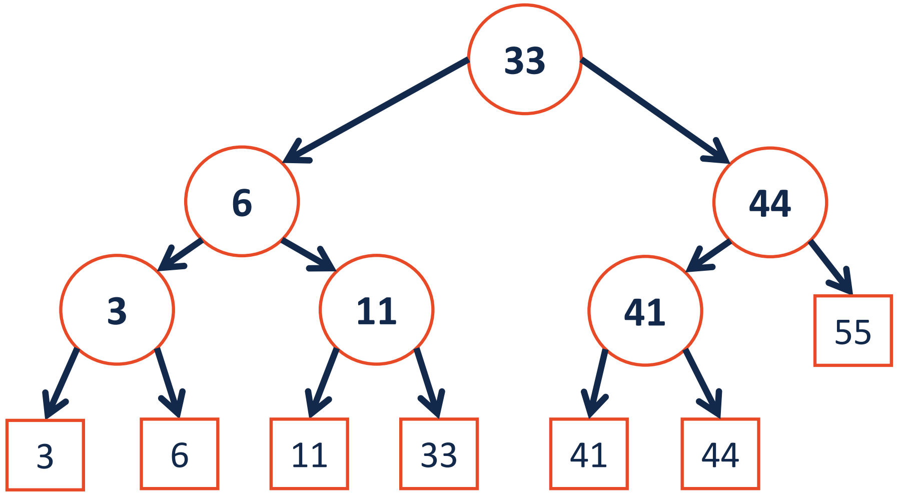
Ex:



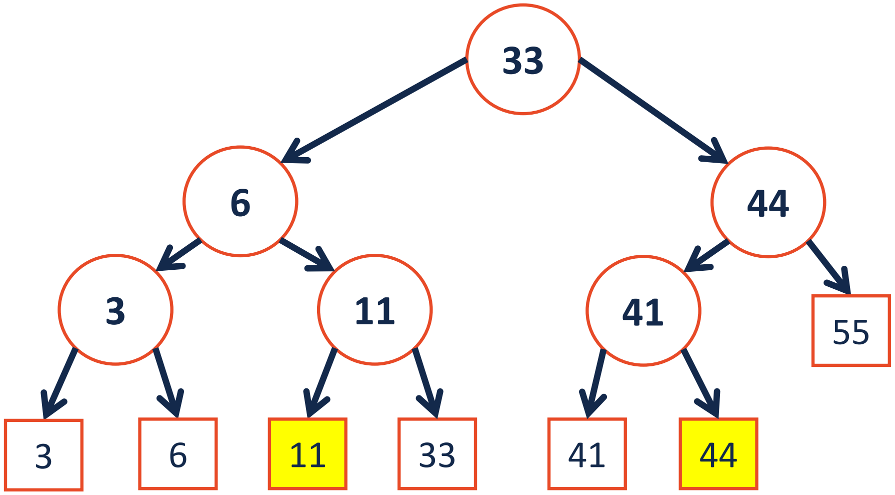
Range-based Searches

Q: Consider points in 1D: **$\mathbf{p} = \{\mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_n}\}$.**

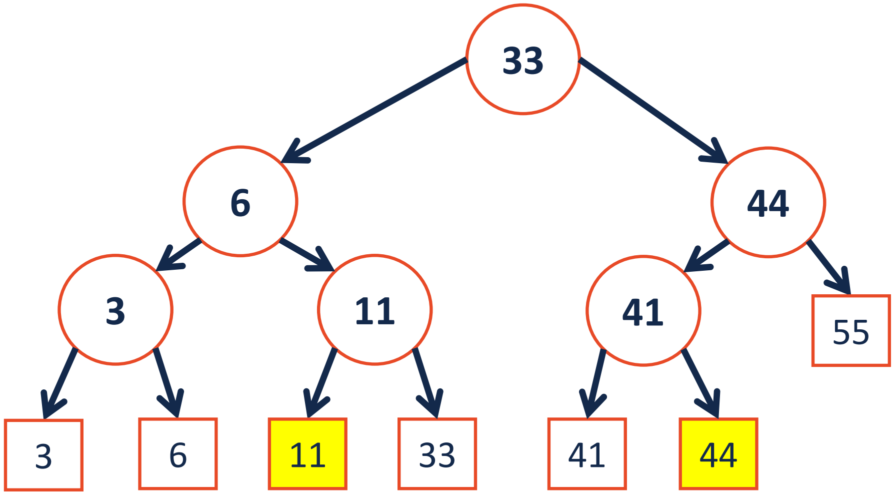
…what points fall in [11, 42]?



Range-based Searches



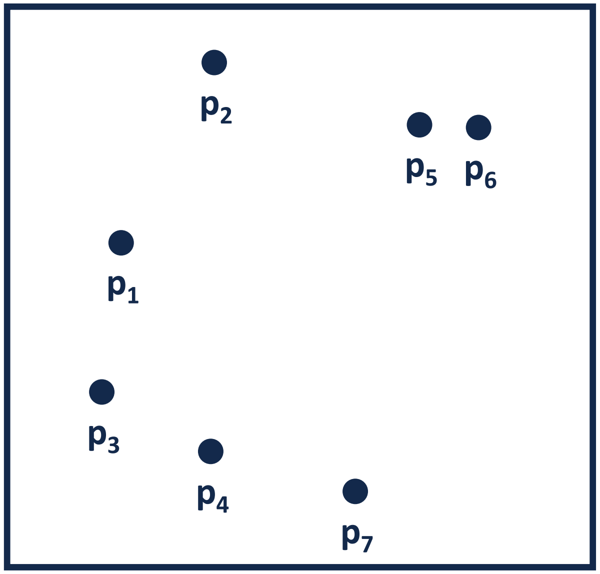
Running Time



Range-based Searches

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

Q: What points are in the rectangle:
$\mathbf{[\ (x_1, y_1),\ (x_2, y_2)\ ]?}$

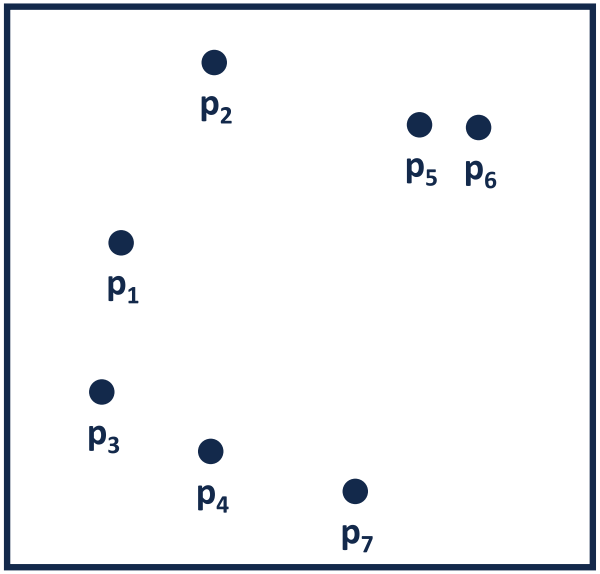


Q: What is the nearest point to $\mathbf{(x_1, y_1)}$?

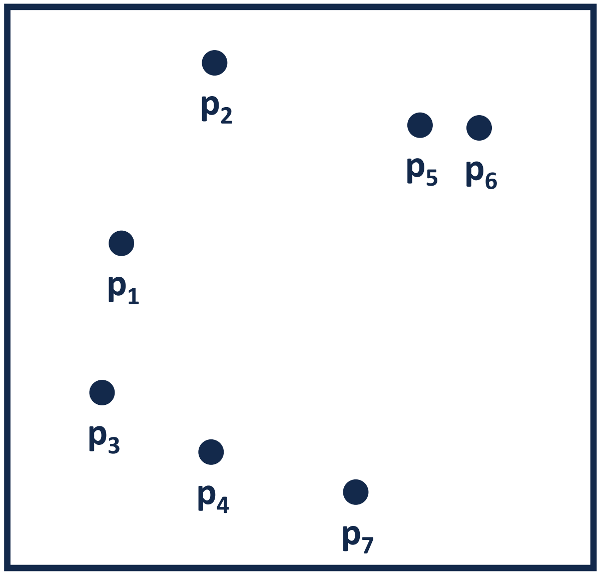
Range-based Searches

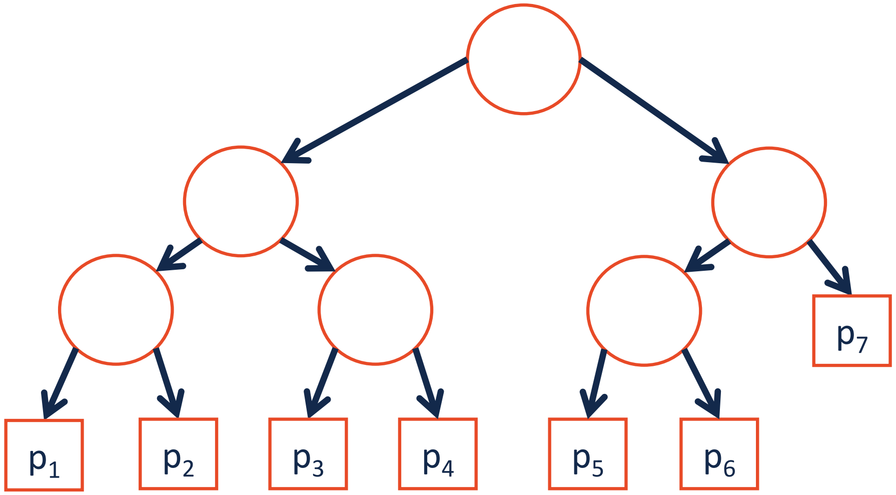
Consider points in 2D: $\mathbf{p = \{p_1, p_2, \ldots, p_n\}}$.

Space divisions:

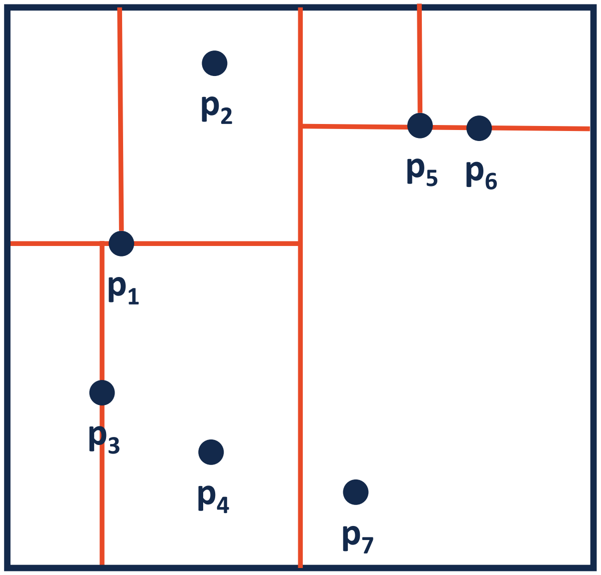


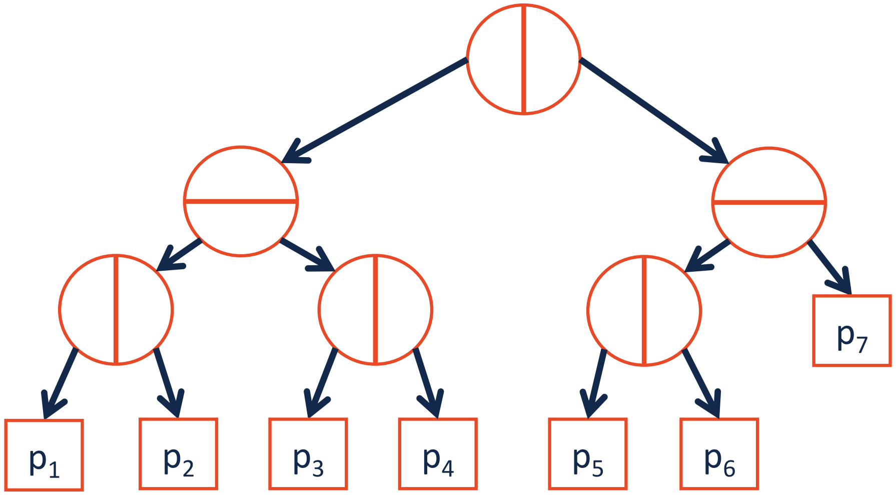
Range-based Searches



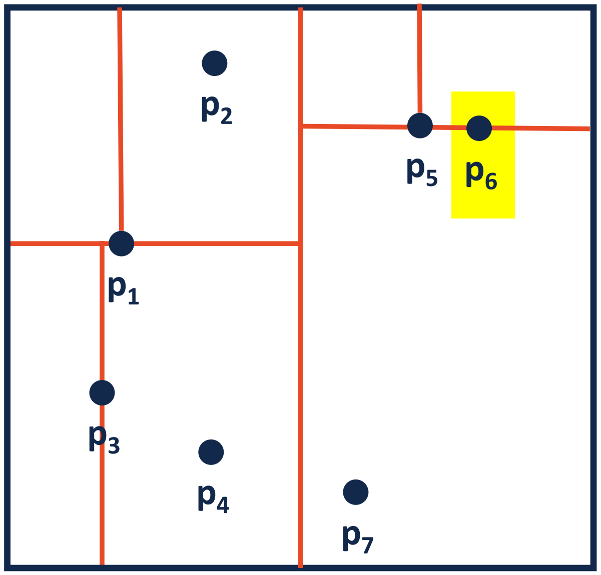


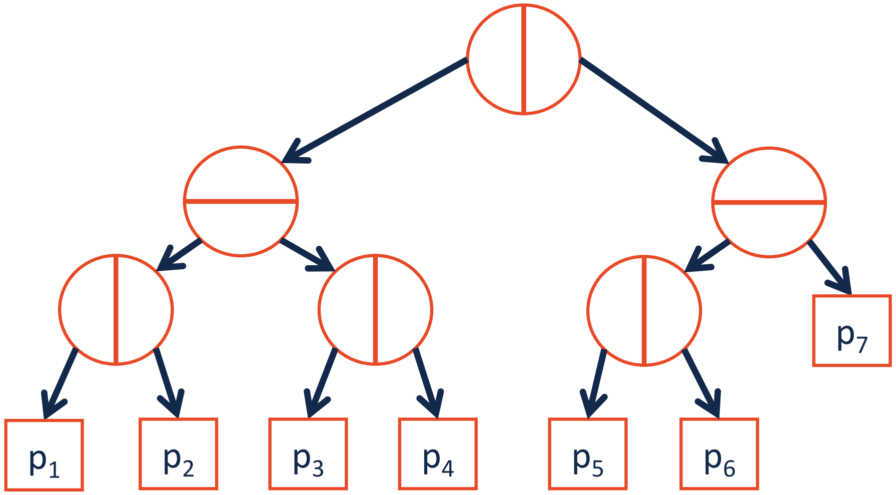
kD-Trees



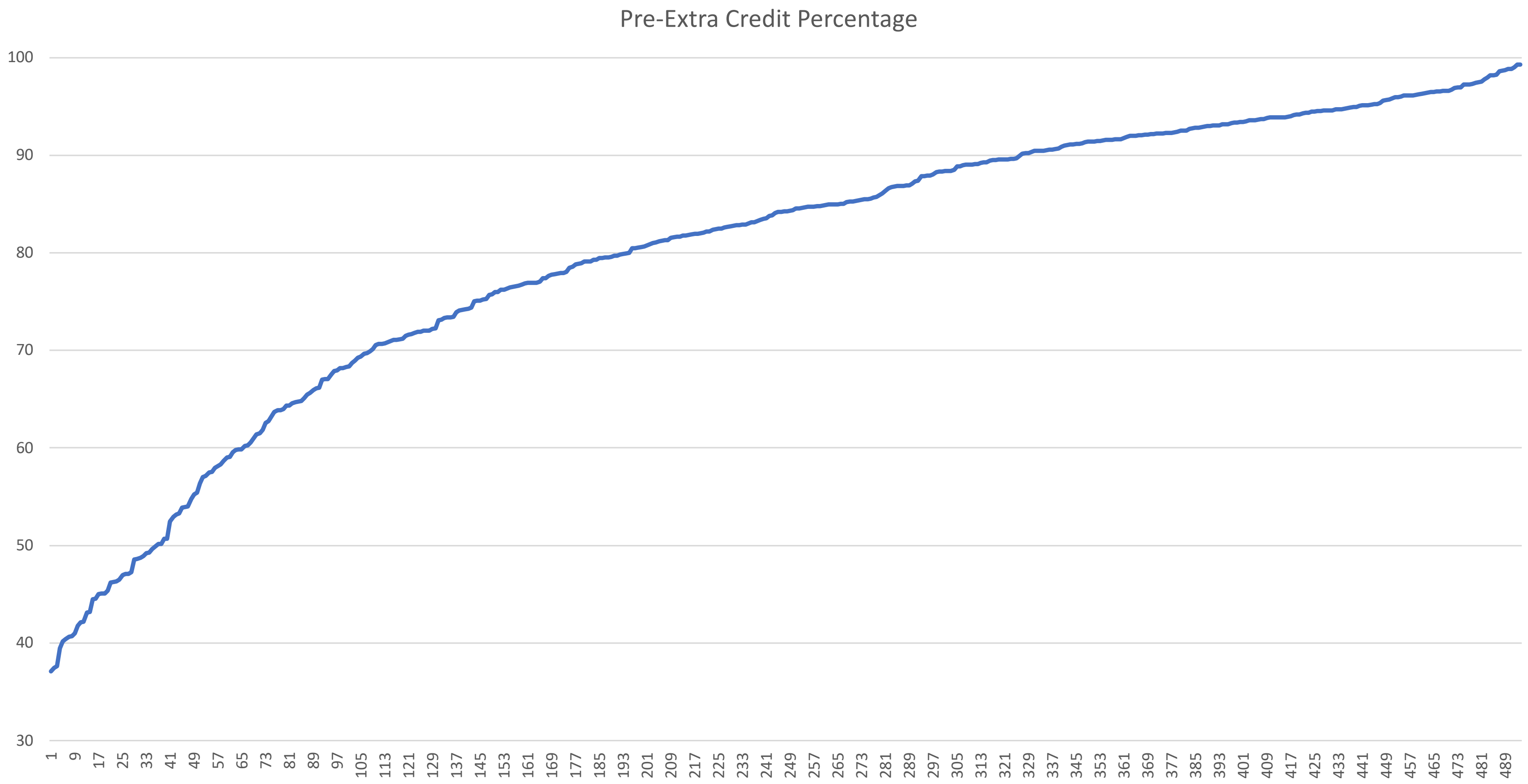


kD-Trees





CS 225 – Midpoint Grade Update



B-Trees

B-Trees

Q: Can we always fit our data in main memory?

Q: Where else can we keep our data?

However, big-O assumes uniform time for all operations.

Vast Differences in Time

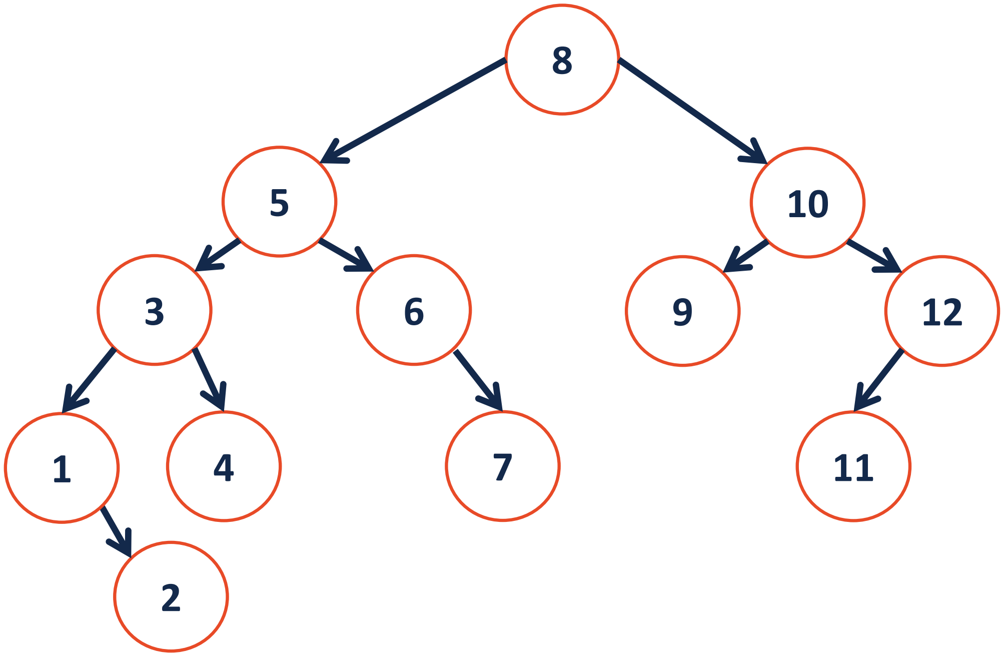
A **3GHz** CPU performs >3 billion operations/sec.

Comparatively: “Disk Storage is Slow”

- Large Disks (25 TB+) still have slow throughout:

Also: “Distributed Systems!”

AVLs on Disk



Real Application

Imagine storing driving records for everyone in the US:

How many records?

How much data in total?

How deep is the AVL tree?

BTree Motivations

Knowing that we have large seek times for data, we want to:

BTree (of order m)

-3	8	23	25	31	42	43	55

m=9

Goal: Minimize the number of reads!

Build a tree that uses ___________________________ / node
[1 network packet]
[1 disk block]

BTree Insertion

A **BTrees** of order **m** is an m-way tree:

- All keys within a node are ordered

- All leaves hold no more than **m-1** keys.

BTree Insertion

When a BTree node reaches **m** keys:

m=5