CS 225

Data Structures

March 7 – kd-Tree and BTrees

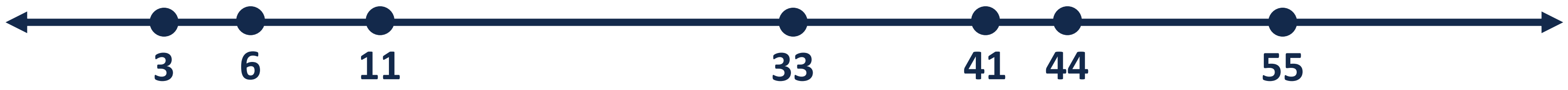
Wade Fagen-Ulmschneider

Range-based Searches

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

Q: Consider points in 1D: $\mathbf{p = \{p_1, p_2, \ldots, p_n\}}$.
…what points fall in [11, 42]?

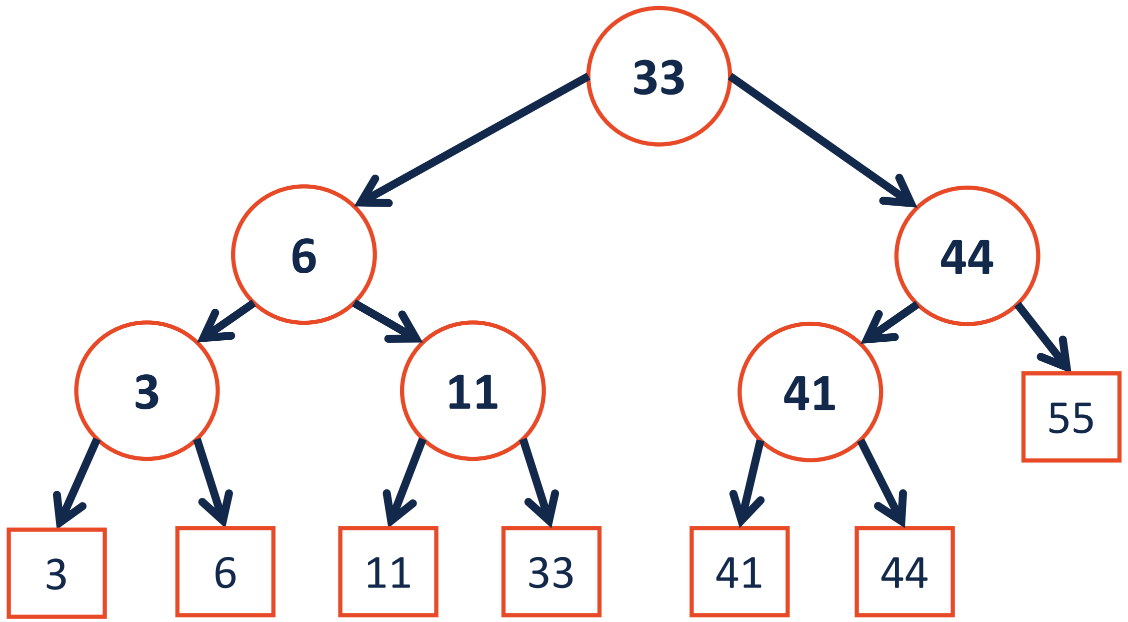
Ex:



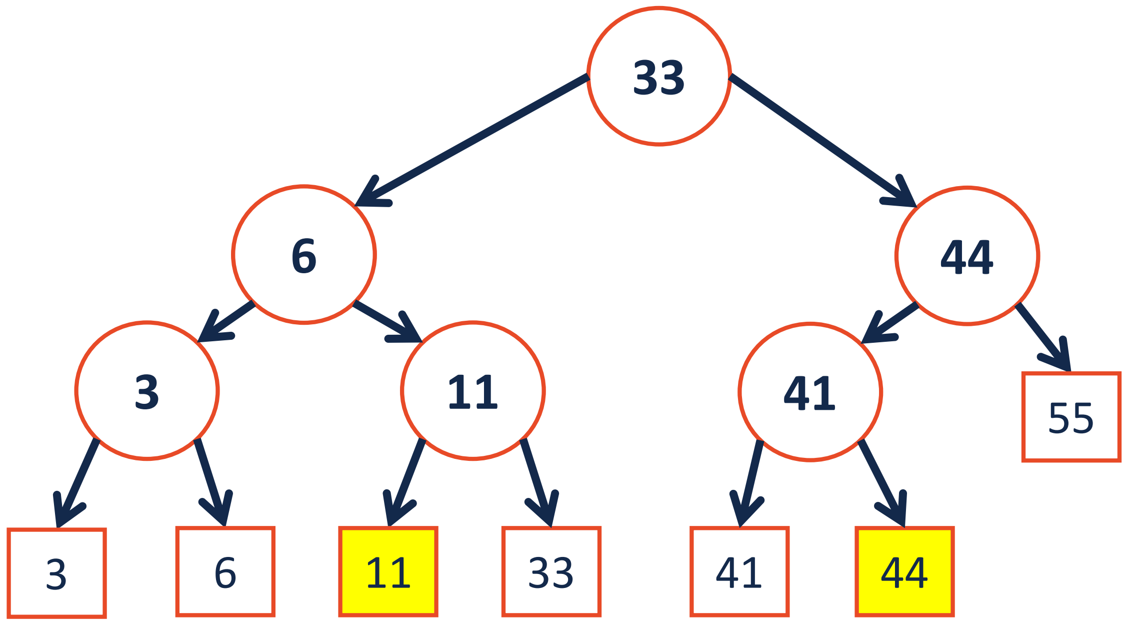
Range-based Searches

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

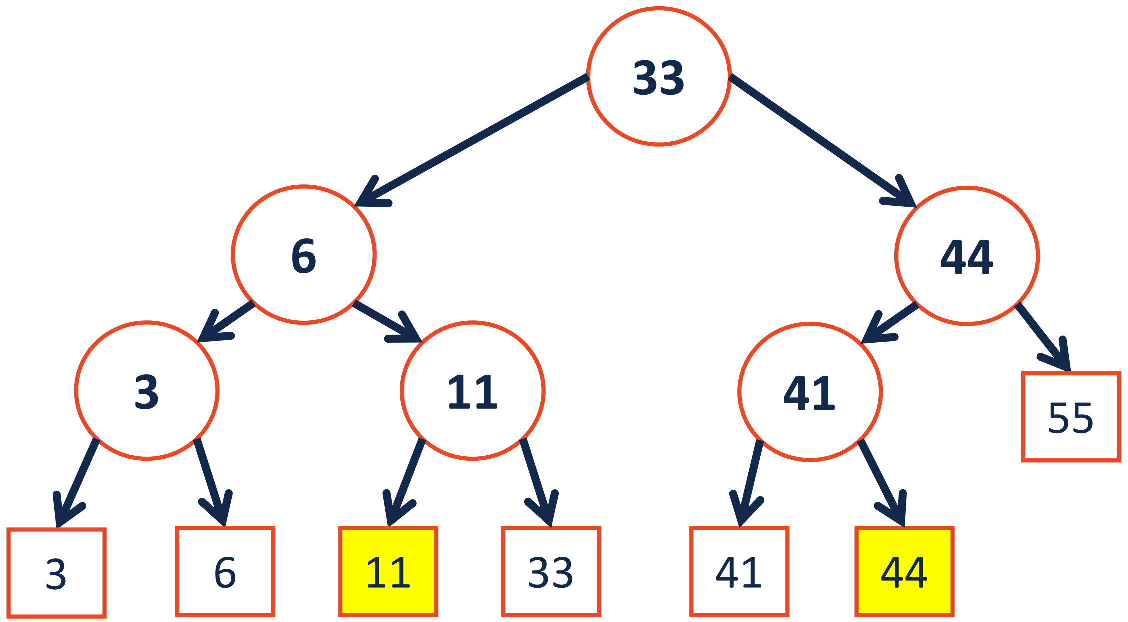
…what points fall in [11, 42]?



Range-based Searches



Running Time

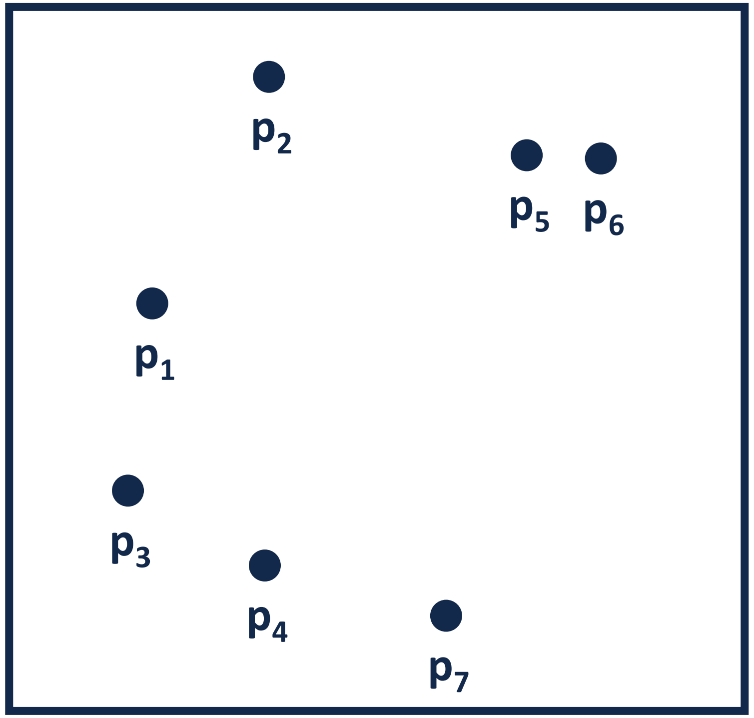


Range-based Searches

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

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

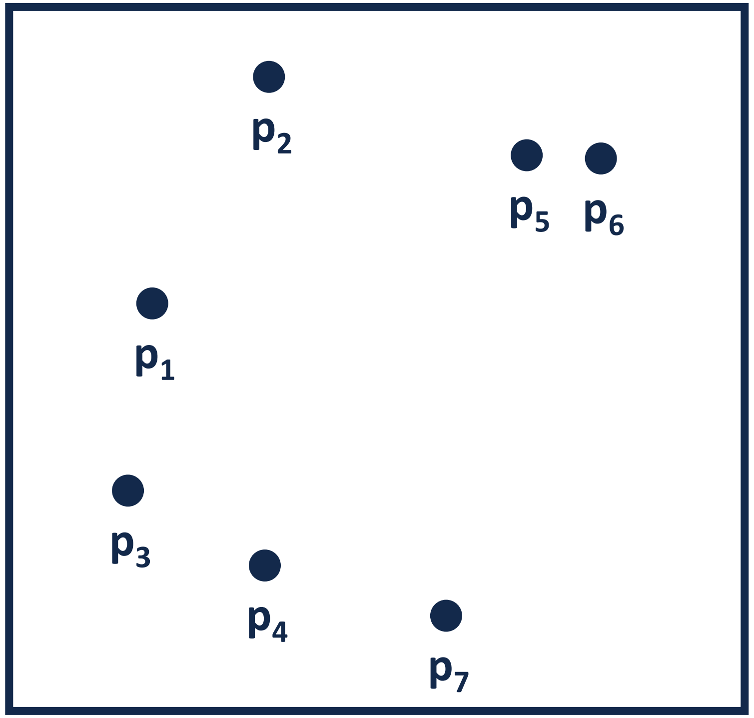
Q: What is the nearest point to (x_1, y_1)?



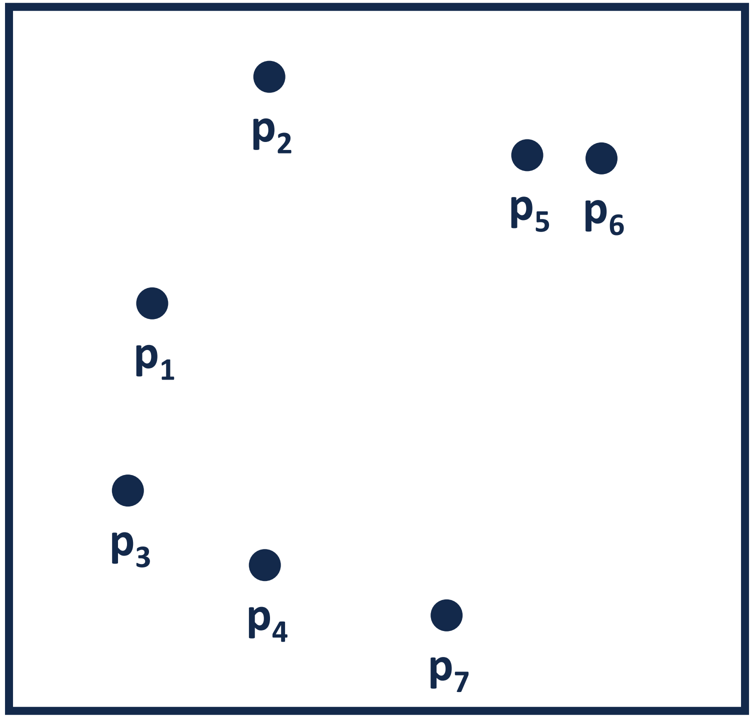
Range-based Searches

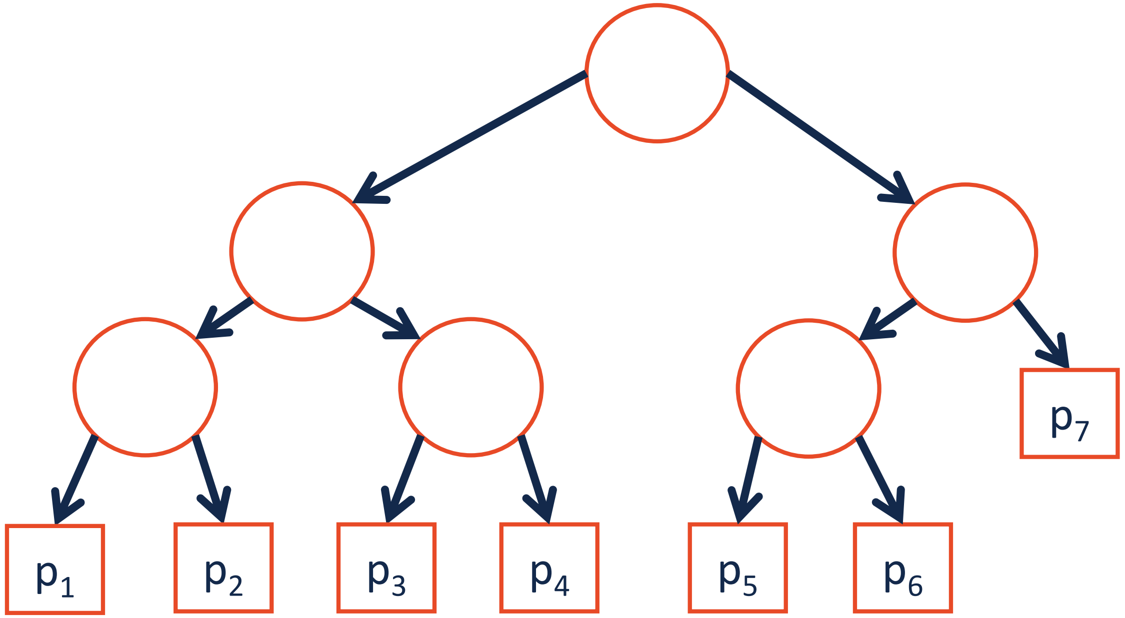
Consider points in 2D: $\mathbf{p} = \{p_1, p_2, \ldots, p_n\}$.

Space divisions:

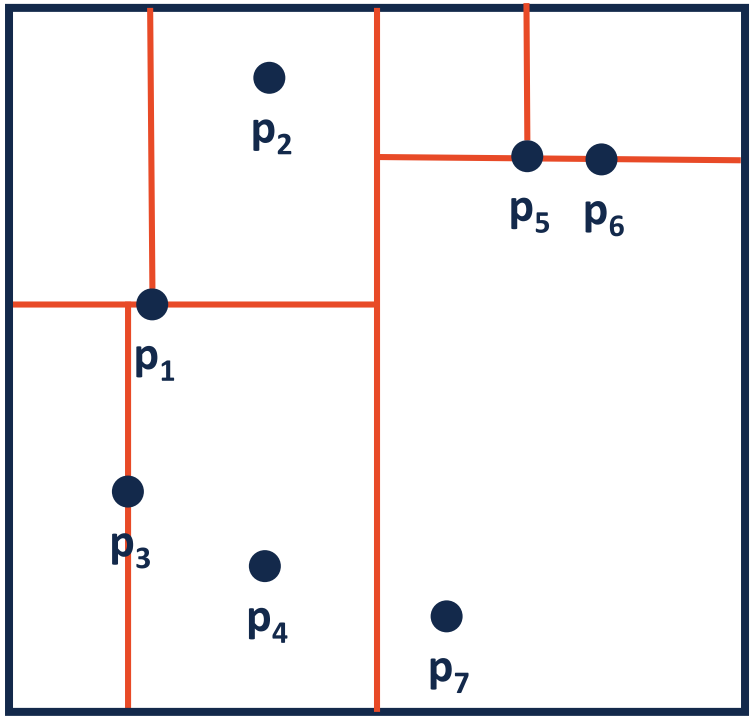


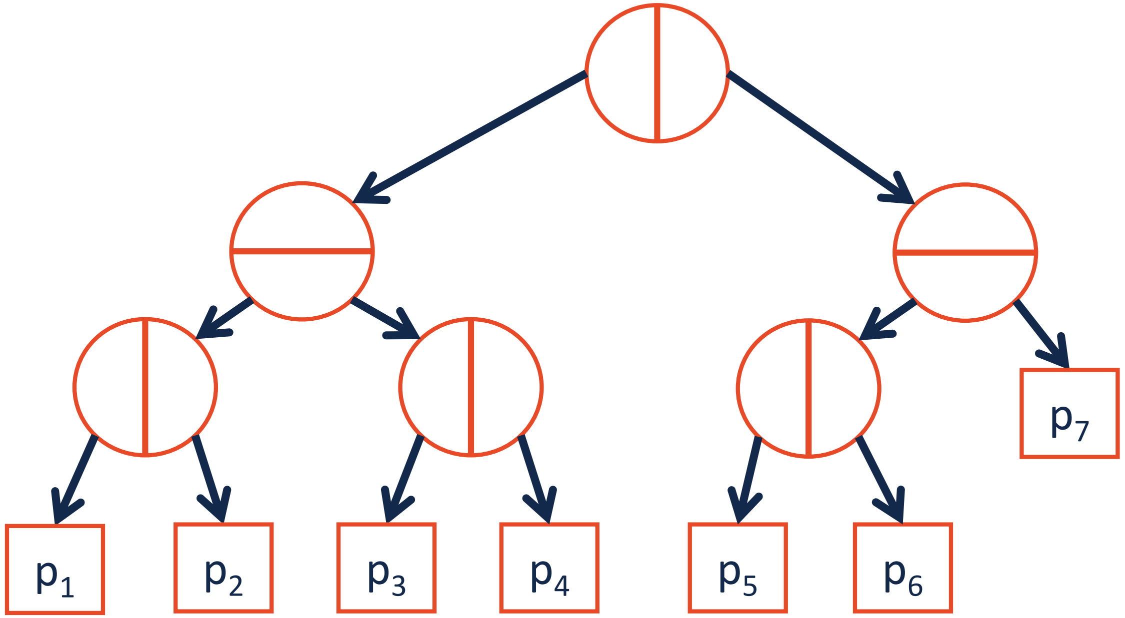
Range-based Searches



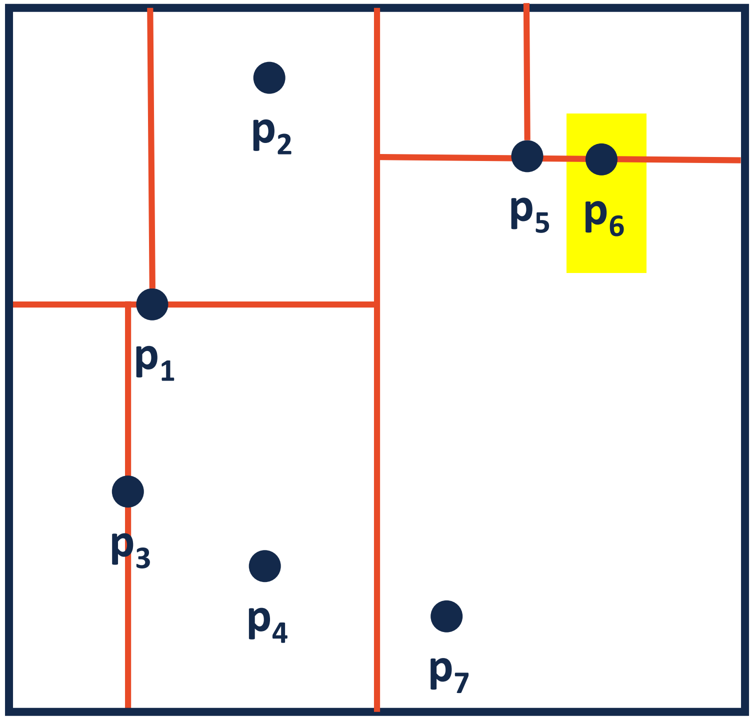


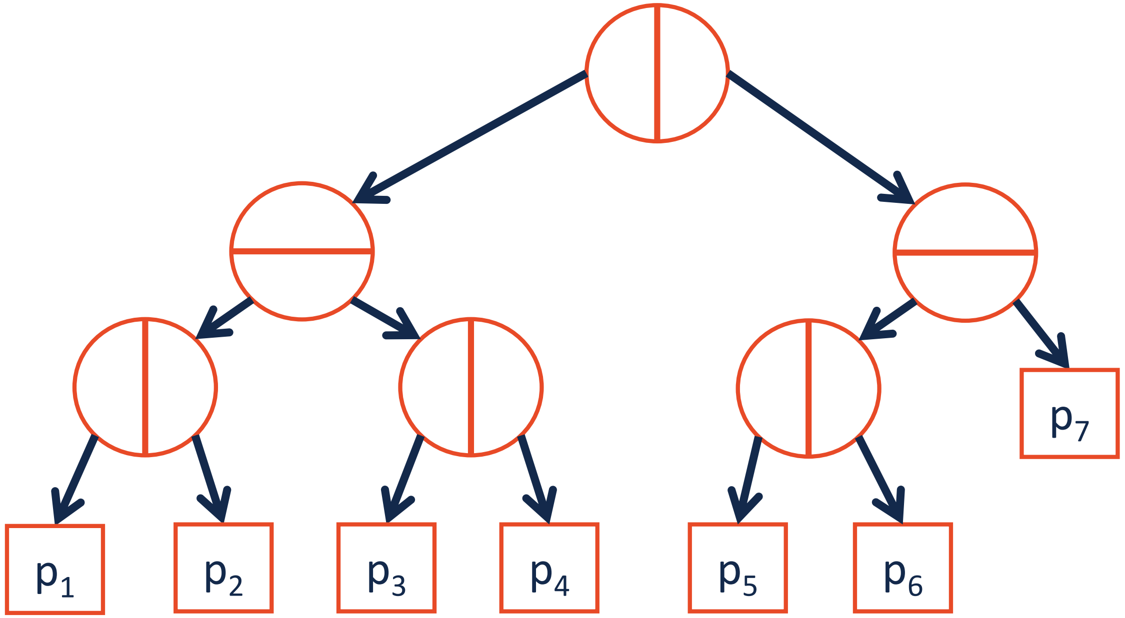
kD-Trees



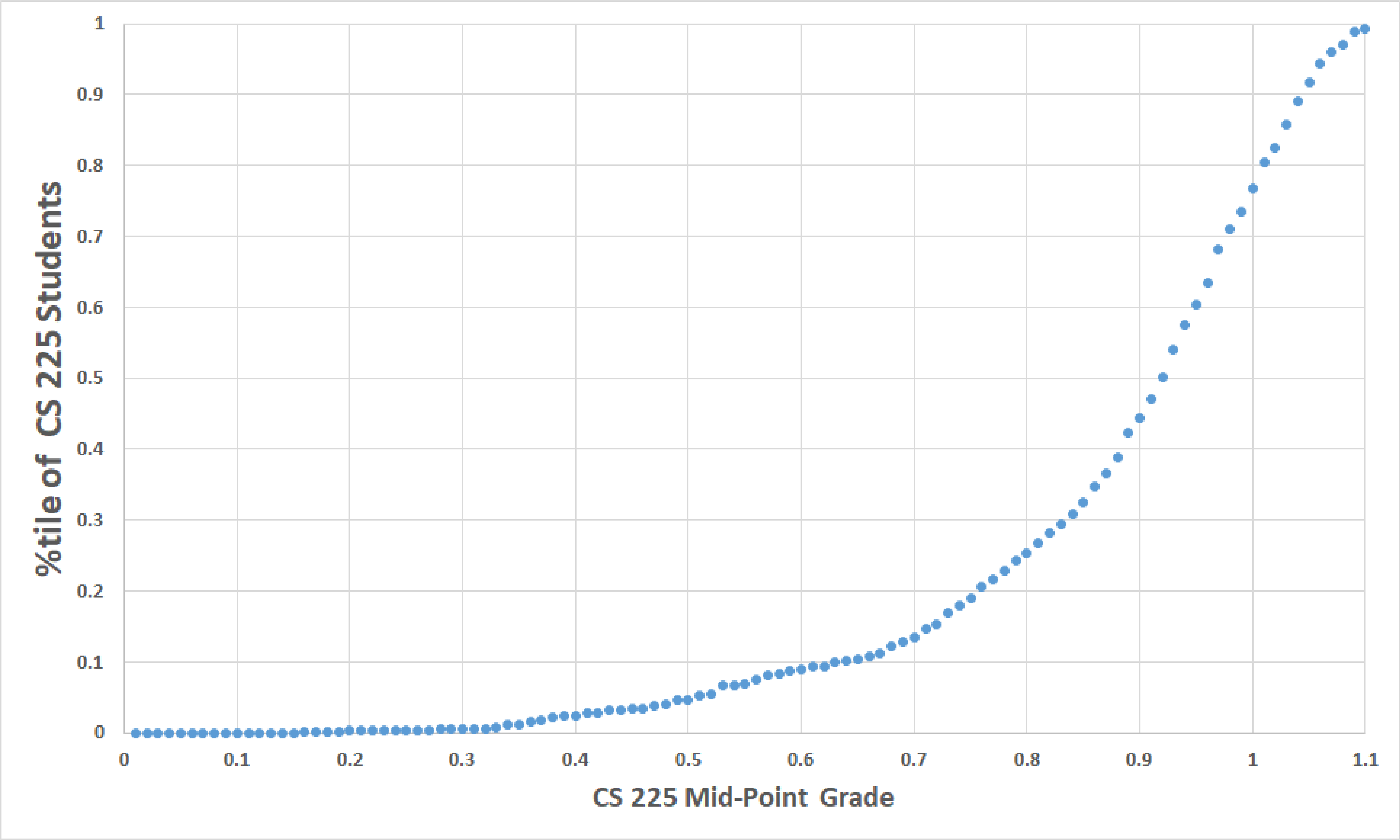


kD-Trees





CS 225 – Midpoint Grade Update



Share Your #cs225 MP4 animation

On Facebook/Twitter/Instagram:

#cs225

…I'll search this tag every few days and like/heart your work!

On Piazza:

See pinned post: "MP4 Animation Sharing"

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 3m operations in ________.

Old Argument: "Disk Storage is Slow"

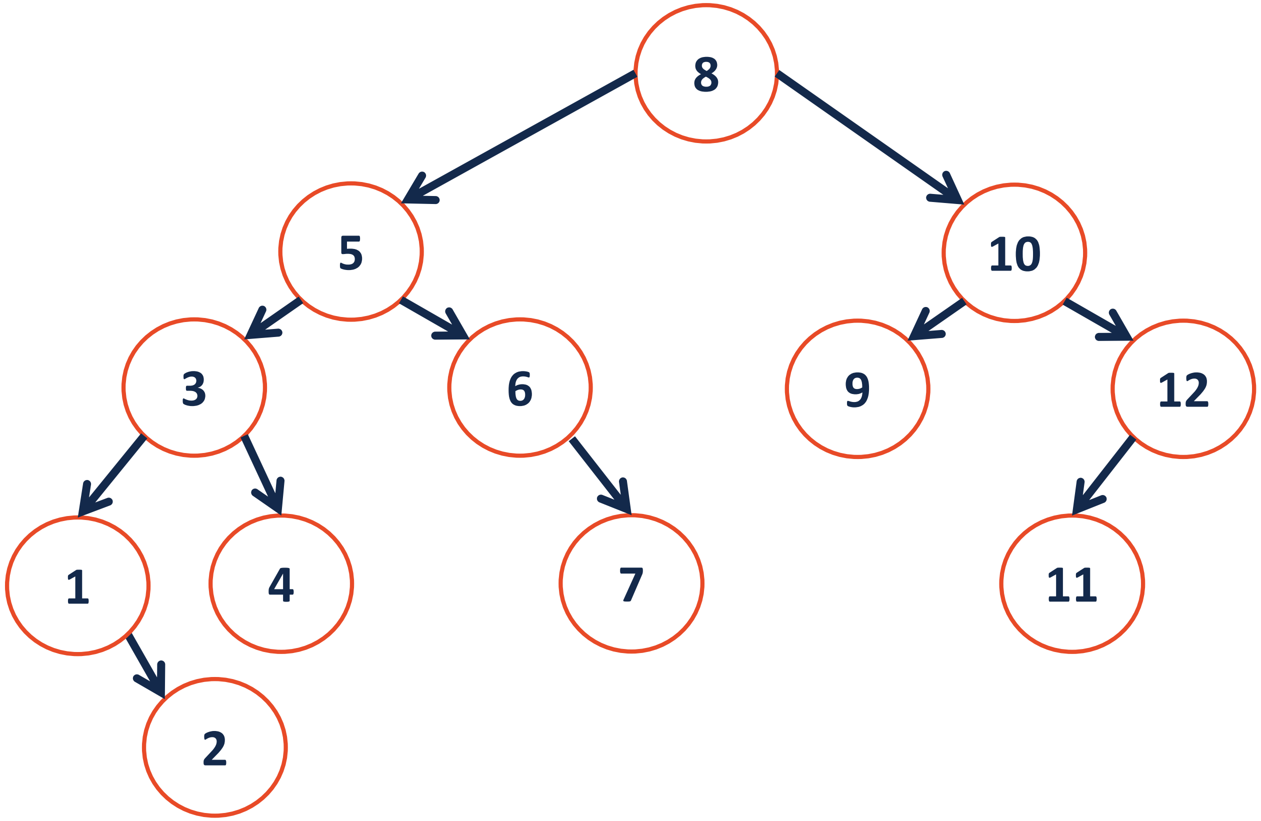
- Bleeding-edge storage is pretty fast:

  NVMe (M.2, PCIe 3.0 x4):

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

New Argument: "The Cloud is Slow!"

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 contain hold no more than **m-1** nodes.

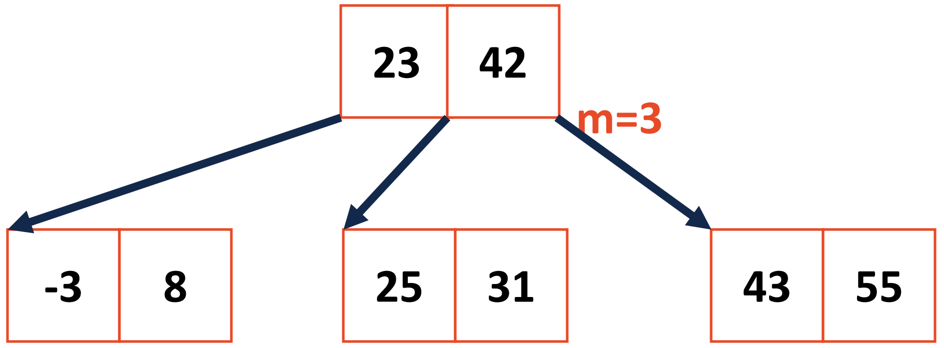
m=5

BTree Insertion

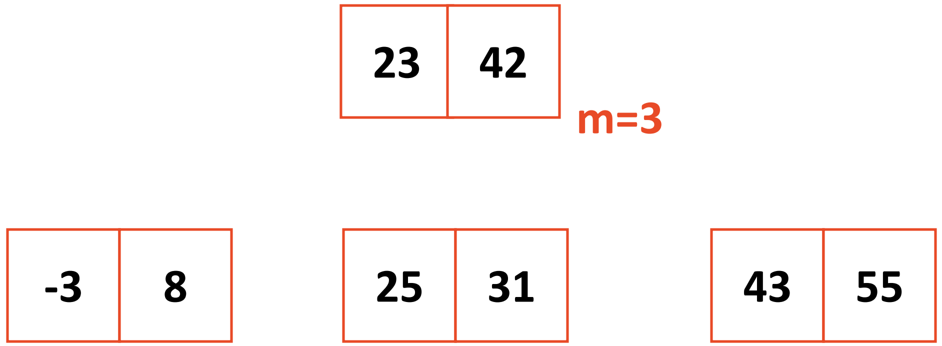
When a BTree node reaches **m** keys:

m=5

BTree Recursive Insert



BTree Recursive Insert



BTree Visualization/Tool

https://www.cs.usfca.edu/~galles/visualization/BTree.html

Btree Properties

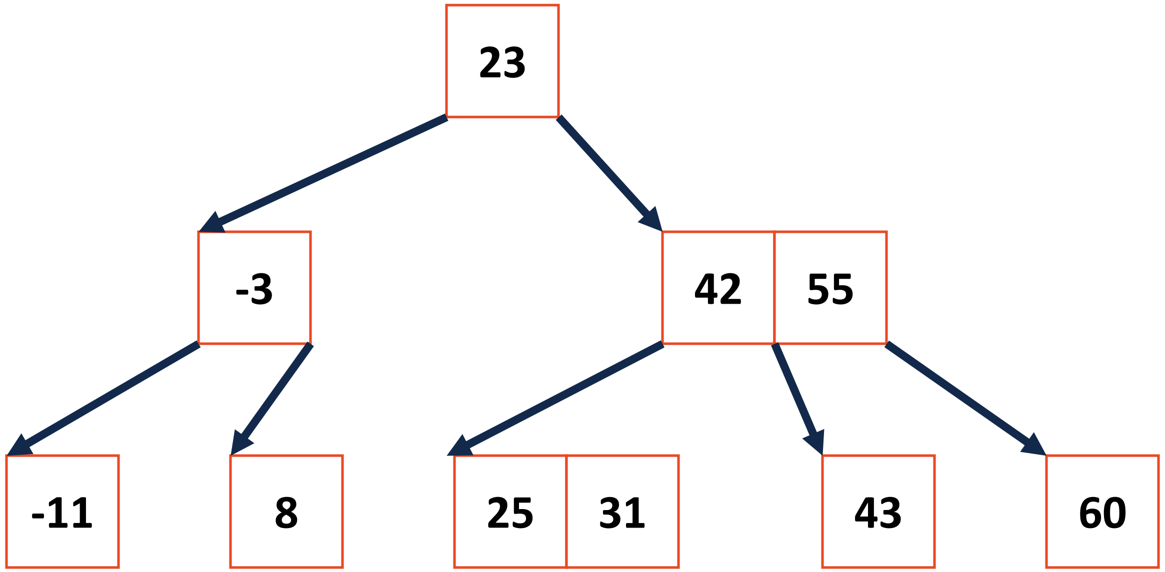
A **BTrees** of order **m** is an m-way tree:

- All keys within a node are ordered
- All leaves contain hold no more than **m-1** nodes.

- All internal nodes have exactly **one more key than children**
- Root nodes can be a leaf or have **[2, m]** children.
- All non-root, internal nodes have **[ceil(m/2), m]** children.

- All leaves are on the same level

BTree Search



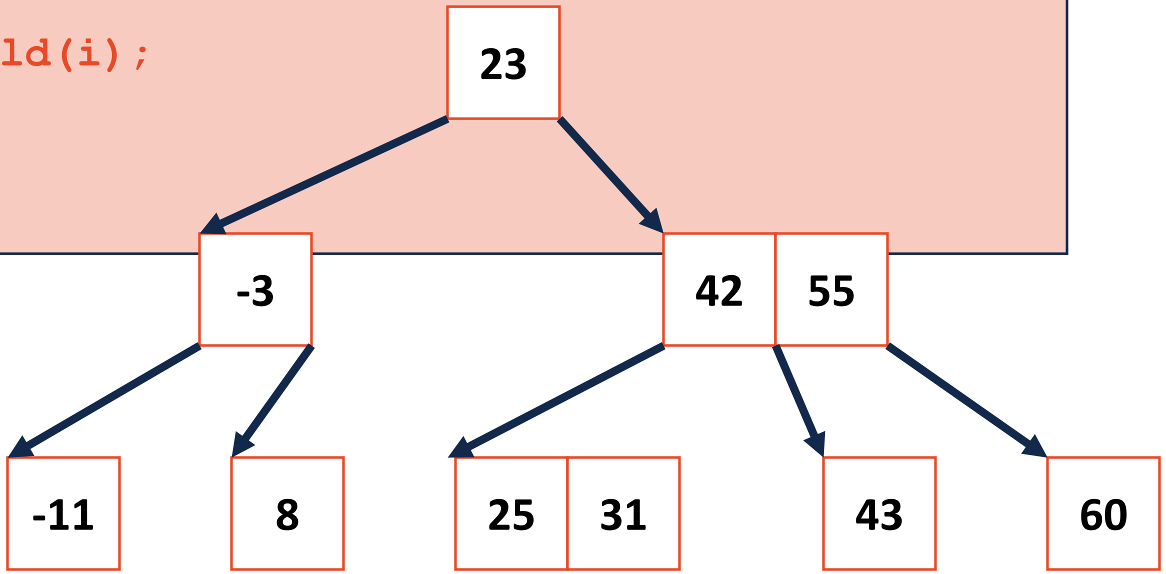
BTree Search

```
bool Btree::_exists(BTreeNode & node, const K & key) {

  unsigned i;
  for ( i = 0; i < node.keys_ct_ && key < node.keys_[i]; i++) { }

  if ( i < node.keys_ct_ && key == node.keys_[i] ) {
    return true;
  }

  if ( node.isLeaf() ) {
    return false;
  } else {
    BTreeNode nextChild = node._fetchChild(i);
    return _exists(nextChild, key);
  }
}
```



BTree Analysis

The height of the BTree determines maximum number of ______________ possible in search data.

…and the height of the structure is: ________________.

Therefore: The number of seeks is no more than ___________.

…suppose we want to prove this!

BTree Analysis

In our AVL Analysis, we saw finding an upper bound on the height (given **n**) is the same as finding a lower bound on the nodes (given **h**).

We want to find a relationship for BTrees between the number of keys (**n**) and the height (**h**).