Announcements

1. MP Lists due Monday
2. Exam 2 Next Week (M-W)
3. Exam 1 Grades Released ~2 weeks
   a. Exam Regrade Form Online
4. Extra Credit Survey Form (2pts, 70% of class must respond)

Extra Credit Survey!

Join Code: **225**

Warm-Up Question: Using a BST, how could I figure out what points fall between a range? Ex. Find all the values in the tree between (20, 40).

ILLINOIS

KD Trees

K- Dimensional

Learning Objectives

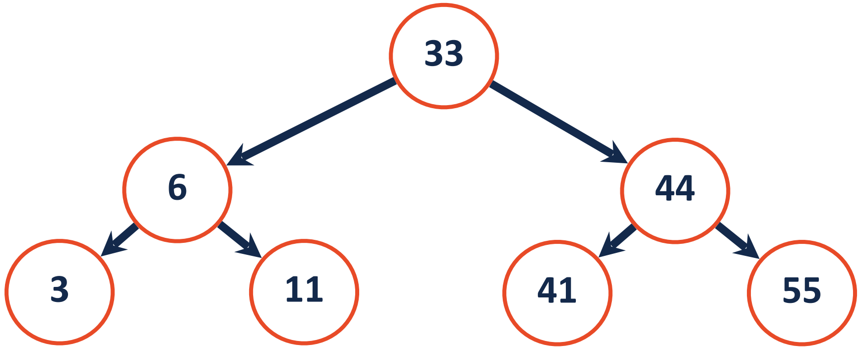
1. Articulate what questions a KD-Tree helps answer
2. Understand the KD-Tree Construction Algorithm
3. Implement the Quick Select Algorithm

Nearest Neighbor and Range Search

Join Code: **225**

What are situations where you would want the nearest point in your database to your search query?

What are situations where you would like a range?

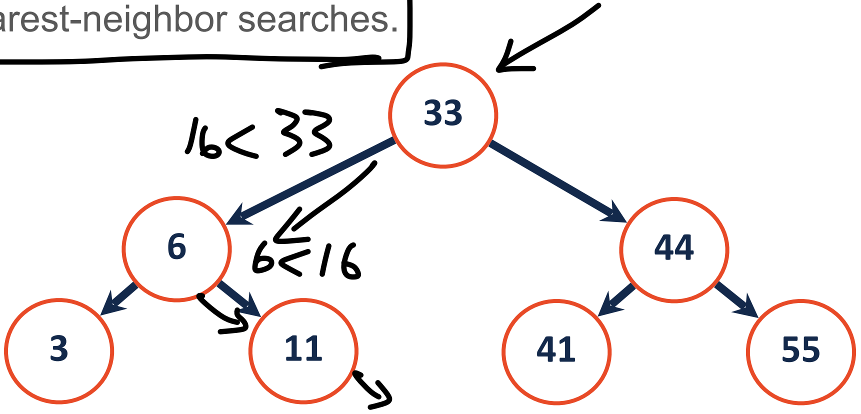


Closest Element

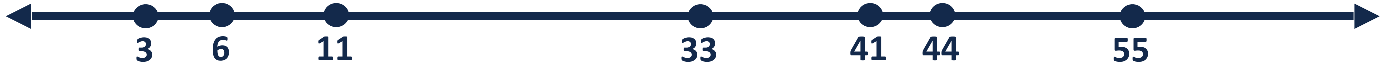
BSTs are useful structures for range-based and $\boxed{\text{nearest-neighbor searches.}}$

Q: Consider points in 1D: **p = {p$_1$, p$_2$, …, p$_n$}.**

…what point is closest to 16?



Ex:



closest_point = ~~33~~ ~~6~~ 11

smallest_distance = |33 − 16| = 17 , |11 − 16| = 5
|6 − 16| = 10 ,

next_largest
next_smallest

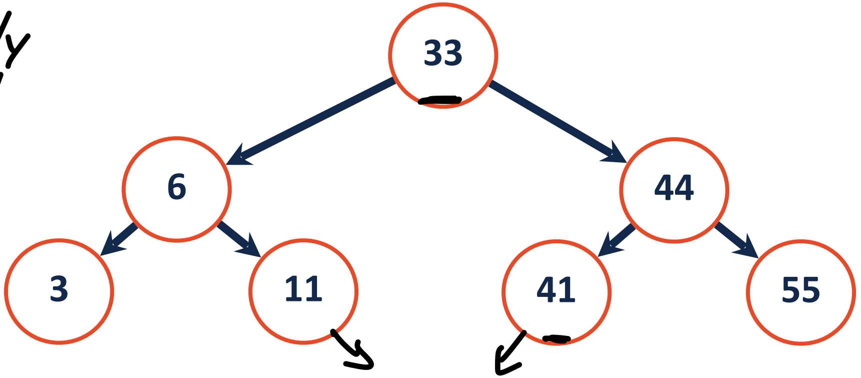
Range-based Searches

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

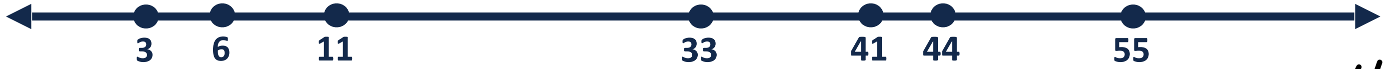
isn't necessarily sorted

Q: Consider points in 1D: **p = {p$_1$, p$_2$, …, p$_n$}.**

…what points fall in [11, 42]?



Ex:



Brute Force – print everything + check if it's in the range

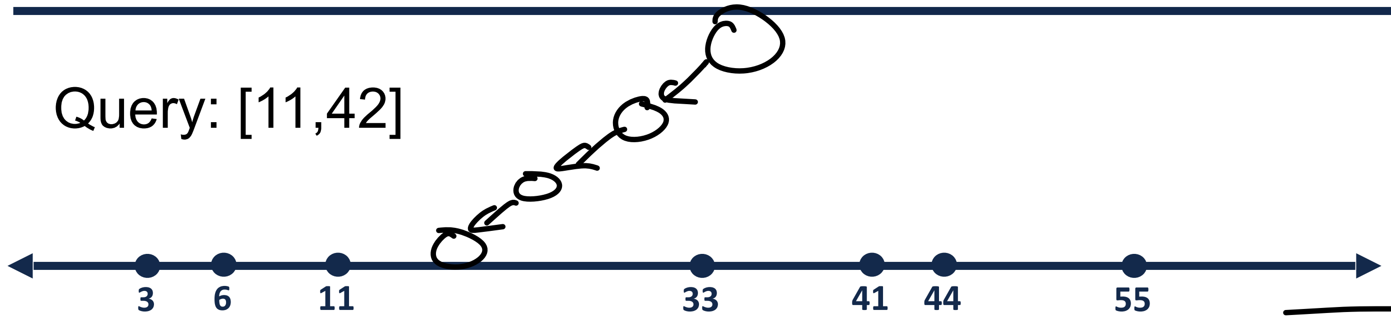
↳ O(n)

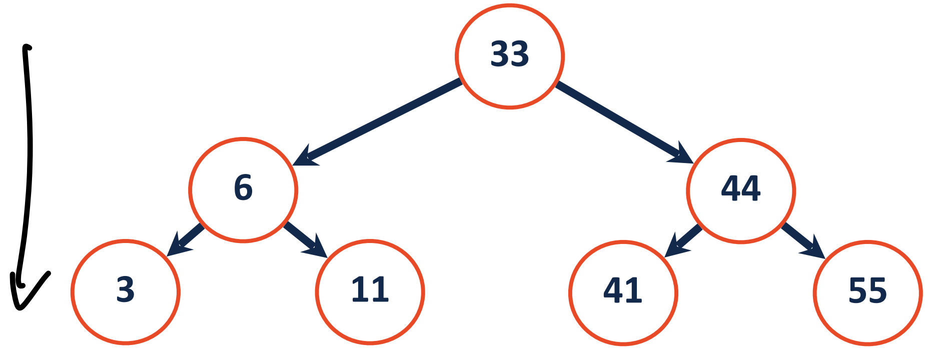
→ begin - smallest element in the range

→ end - smallest value larger than the range

Range-based Searches

Query: [11,42]





m - size of range

	BST	Perfect
1. Find Begin	$O(n) = O(h)$	$O(\log n)$
2. Find End	$O(n) = O(h)$	$O(\log n)$
3. Iterate from Begin to End	$O(m) = O(n)$	$O(m)$
Total	$O(n)$	$O(\log n + m)$

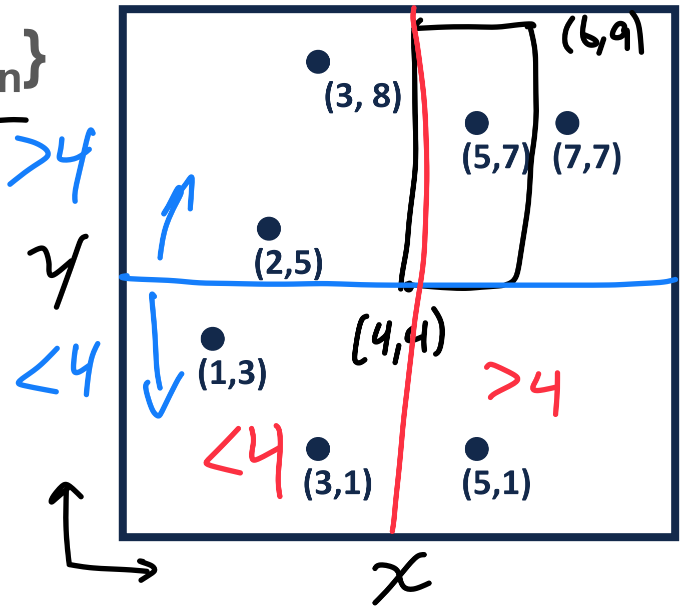
2D Range Based Searches

Consider points in 2D: **p = {p_1, p_2, …, p_n}**

Q: What points are in the rectangle:
[(4, 4), (6, 9)]?

x(4,6) , y[4,9)

Q: What is the nearest point to **(4,3)**?



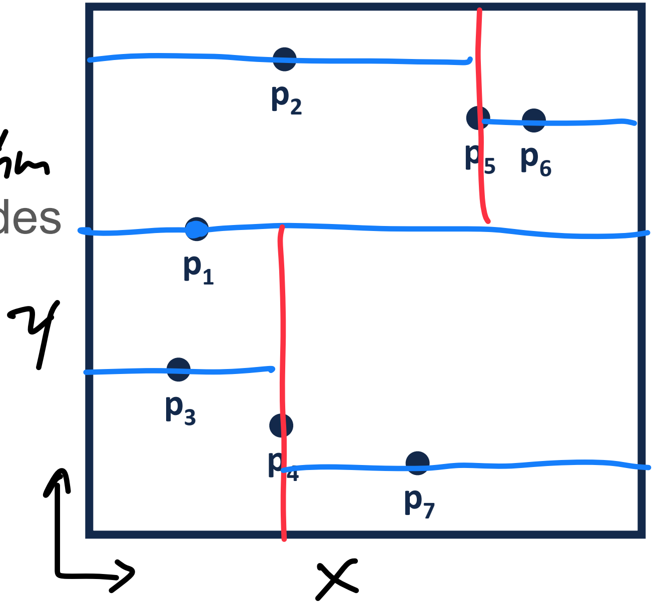
2D Range Based Searches

Consider points in 2D: **p = {p_1, p_2, …, p_n}**

Tree Construction:

Quick Select Algorithm

1. Find median point along a dimension and partition nodes
2. Go to next dimension
3. Recursively build left subtree
4. Recursively build right subtree



Quick Select Algorithm

Partitions elements about the median

Smaller Values are on the left of the median

Larger Values are on the right of the median

☆ Faster than sorting O(n log n)

Ex. Input: [11, 6, 44, 41, 33, 57, 2]

Output: [11, 6, 2, $\boxed{33}$, 41, 57, 44]

← left
less than 33

→ right
greater than 33

Quick Select Algorithm

1. Select a random pivot
2. Swap the pivot value to the end
3. Partition elements
4. Swap pivot value to proper place

Quick Select Algorithm

0 1 2 3 4 5 6

Ex. [11, 6, 44, 41, 33, 57, 2] ← considering 1dimension at a time

For a random pivot index, k = 3

[11, 6, 44, 2, 33, 57, 41], i = 0, small = 0

↑ small ↑ i

i – iterator to help traverse the array

small – track the swap index for values smaller than my pivot

[11, 6, 2, 44, 33, 57, 41]

↑ small ↑ i

Quick Select Algorithm

Ex. [11, 6, 44, 41, 33, 57, 2]

For a random pivot index, k = 3

[11, 6, 44, 2, 33, 57, 41], i = 0, small = 0

```
// swap pivot and last value
swap(arr[k], arr[length-1]);

for (int i = 0; i < length; i++){
    if (arr[i] < arr[length-1]){
        swap(arr[i], arr[small]);
        small++;
}
```

// swap pivot to proper place

Quick Select Algorithm

Join Code: **225**

Ex. [11, 6, 44, 41, 33, 57, 2], For a random pivot index, k = 3

arr	i	small
[11, 6, 44, 2, 33, 57, 41]	0	0
[11, 6, 44, 2, 33, 57, 41]	1	1
[11, 6, $\underline{44}$, 2, 33, 57, 41]	2	2
[11, 6, 44, $\underline{2}$, 33, 57, $\underline{41}$]	3	2
[11, 6, 2, 44, $\underline{33}$, 57, 41]	4	3
[11, 6, 2, 33, 44, $\underline{57}$, 41]	5	4
[11, 6, 2, 33, 44, 57, 41]	6	4

```
// swap pivot and last value
swap(arr[k], arr[length-1]);

for (int i = 0; i < length; i++){
        if (arr[i] < arr[length-1]){
                swap(arr[i], arr[small]);
                small++;
        }
}
```

Quick Select Algorithm

Ex. [11, 6, 44, 41, 33, 57, 2], For a random pivot index, k = 3

arr	i	small
[11, 6, 44, 2, 33, 57, 41]	0	0
[11, 6, 44, 2, 33, 57, 41]	1	1
[11, 6, 44, 2, 33, 57, 41]	2	2
[11, 6, 44, 2, 33, 57, 41]	3	2
[11, 6, 2, 44, 33, 57, 41]	4	3
[11, 6, 2, 33, 44, 57, 41]	5	4
[11, 6, 2, 33, 44, 57, 41]	6	4

Quick Select Algorithm

[11, 6, 2, 33, 44, 57, 41], i = 6, small = 4

[11, 6, 2, 33, 41, 57, 44], Swap pivot value back to proper place

median=3

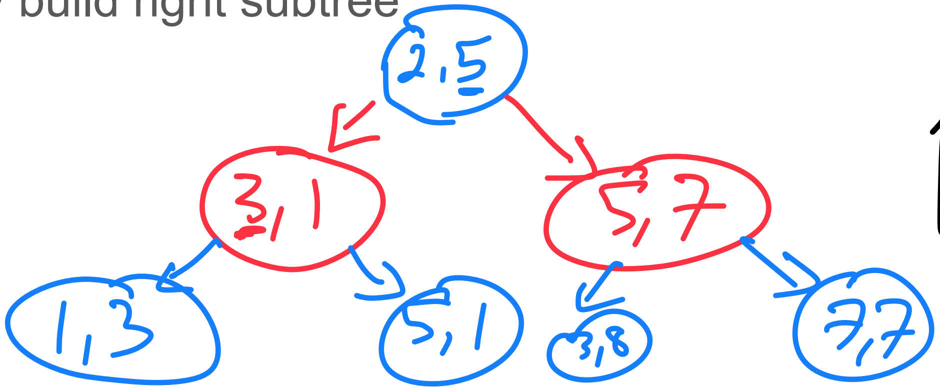
Recurse down either left or right subtree

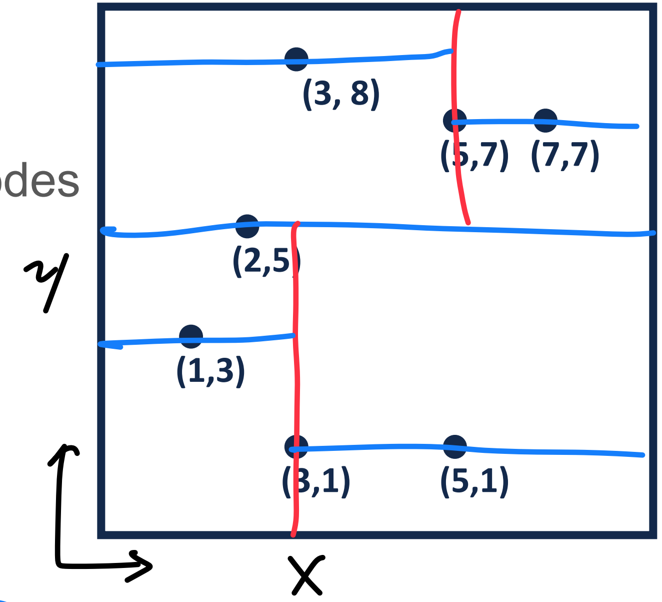
Constructing a KD Tree

Consider points in 2D: **p = {p$_1$, p$_2$, …, p$_n$}**

Tree Construction:

1. Find median point along a dimension and partition nodes
2. Go to next dimension
3. Recursively build left subtree
4. Recursively build right subtree





Worst Case Analysis

[11, 6, 44, 41, 33, 57, 2]

[11, 6, 44, 41, 33, 2, 57]

Total Work = $n + (n-1) + (n-2) + \dots + 1$

$= \frac{n(n+1)}{2}$

$= O(n^2)$

Average Case Analysis