CS 225

Data Structures

February 24 – BBST Summary

G Carl Evans

Would you like to attend UIUC tuition-free?
What if you could even get paid to go to school?
Come and join us!

Illinois Cyber Security Scholars Program Info Session

Thursday, March 2, 2023

5:00 – 6:45 pm @ 2405 Siebel Center

PIZZA WILL BE PROVIDED!

To learn more, visit:
https://publish.illinois.edu/cybersecurityscholars

Applicants must be US citizens or legal permanent residents.

Would you like to attend UIUC tuition-free?

What if you could even get paid to go to school?

Come and join us!

Illinois Cyber Security Scholars Program Info Session

Thursday, March 2, 2023

5:00–6:45 pm @ 2405 Siebel Center

PIZZA WILL BE PROVIDED!

This NSF-funded CyberCorps Scholarship for Service program provides scholarships to admitted, full-time UIUC students interested in cybersecurity.

Benefits include:

- Full tuition and fee waiver
- Stipend, allowance, and professional development funds
- Research experience with security experts
- Expertise in a growing professional field

Application Deadline:

March 31, 2023

To learn more, visit:

publish.illinois.edu/cybersecurityscholars

Applicants must be US citizens or legal permanent residents.

Summary of Balanced BST

Pros:

- Running Time:

    - Improvement Over:

- Great for specific applications:

Summary of Balanced BST

Cons:

- Running Time:

- In-memory Requirement:

Range-based Searches

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

Tree construction:

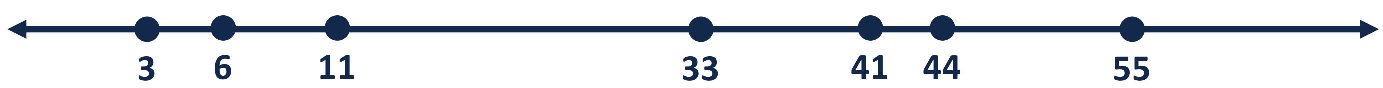
Range-based Searches

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]?

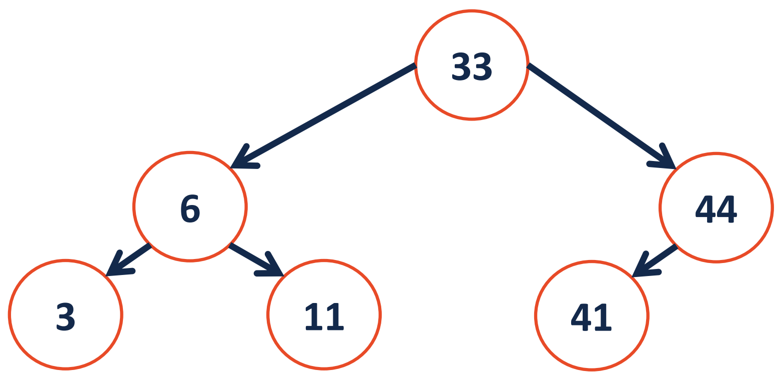
Ex:



Range-based Searches

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

…what points fall in [11, 42]?



Red-Black Trees in C++

C++ provides us a balanced BST as part of the standard library:

```
std::map<K, V> map;
```

Red-Black Trees in C++

```
V & std::map<K, V>::operator[]( const K & )
```

Red-Black Trees in C++

```
V & std::map<K, V>::operator[]( const K & )
```

```
std::map<K, V>::erase( const K & )
```

Red-Black Trees in C++

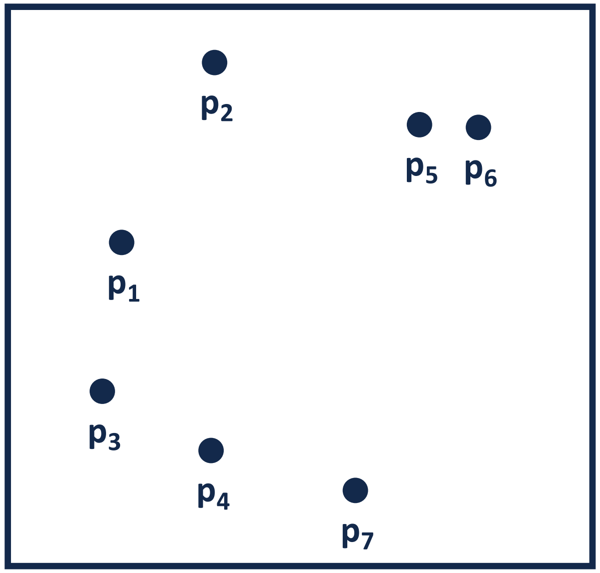
```
iterator std::map<K, V>::lower_bound( const K & );
iterator std::map<K, V>::upper_bound( const K & );
```

Range-based Searches

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

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

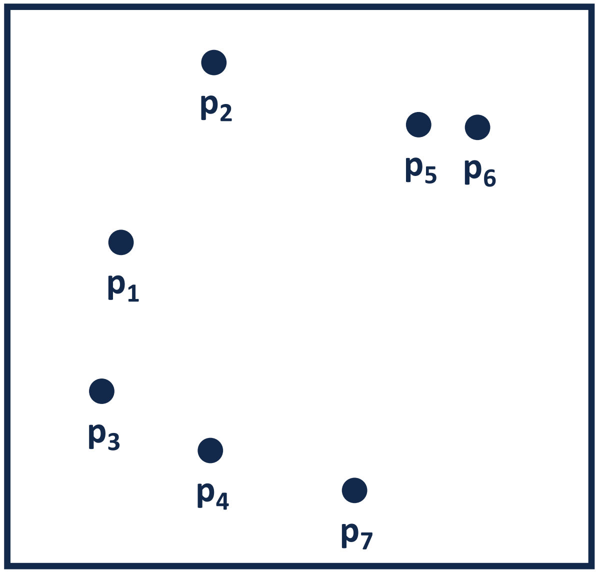
Q: What is the nearest point to **(x_1, y_1)**?



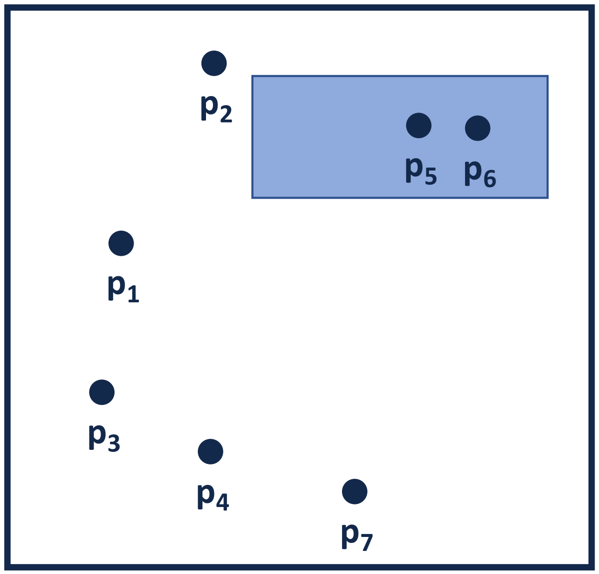
Range-based Searches

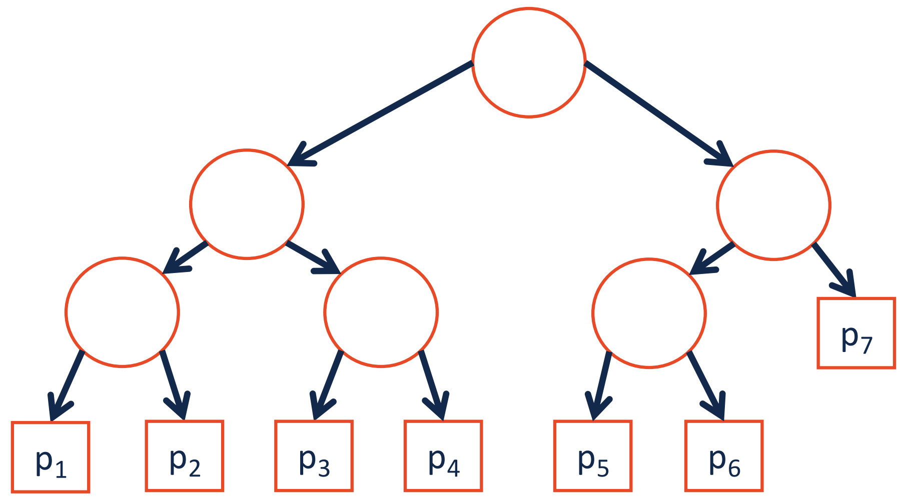
Consider points in 2D: $\mathbf{p} = \{\mathbf{p_1, p_2, \ldots, p_n}\}$.

Tree construction:

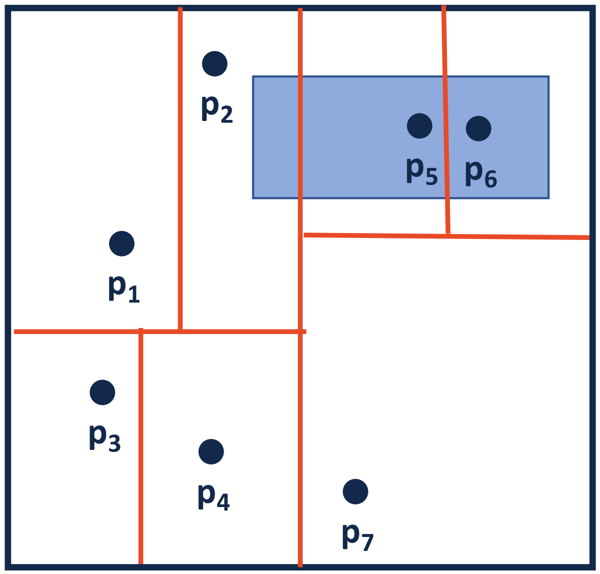


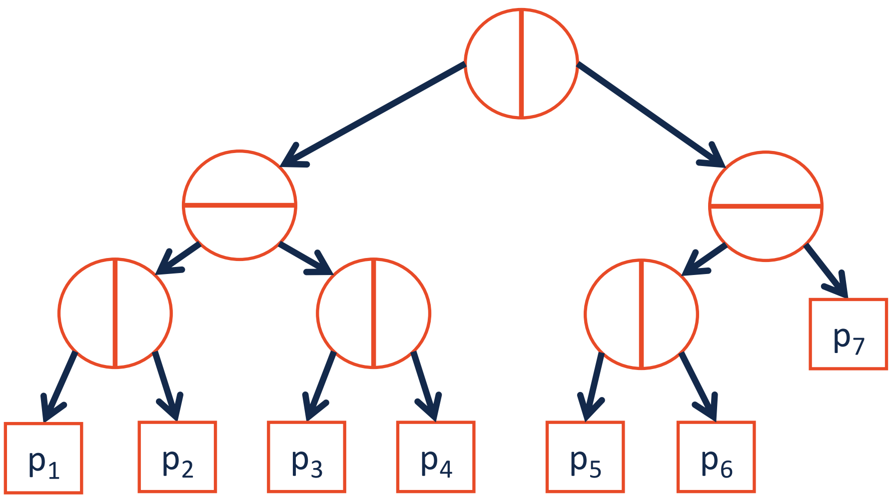
Range-based Searches



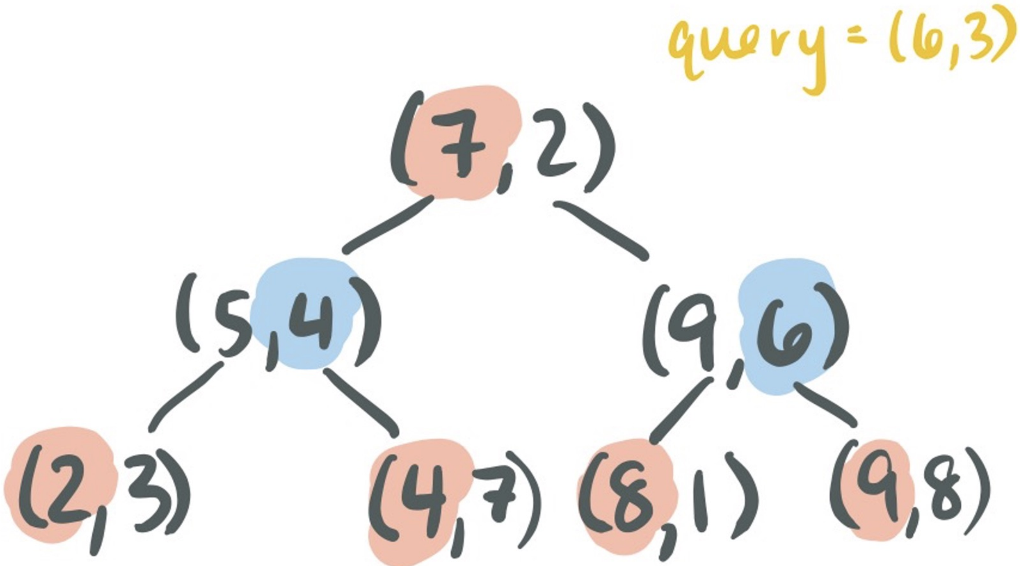


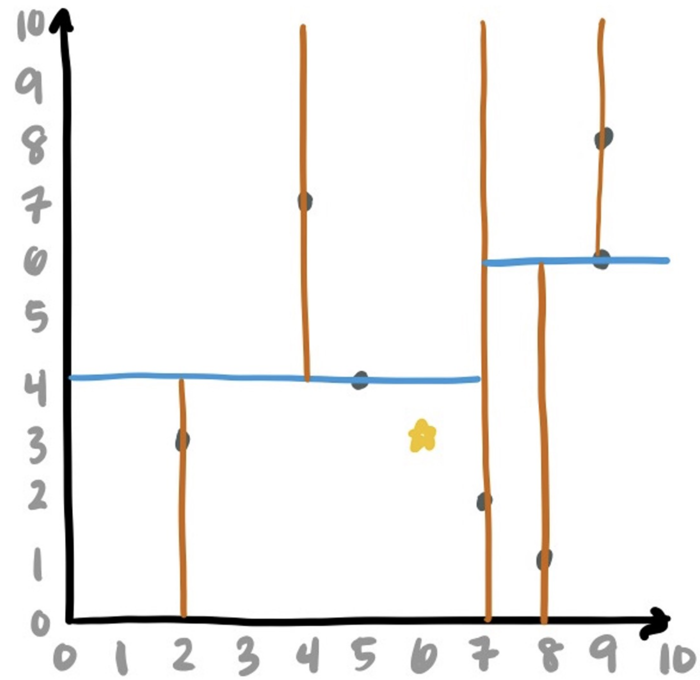
Range-based Searches



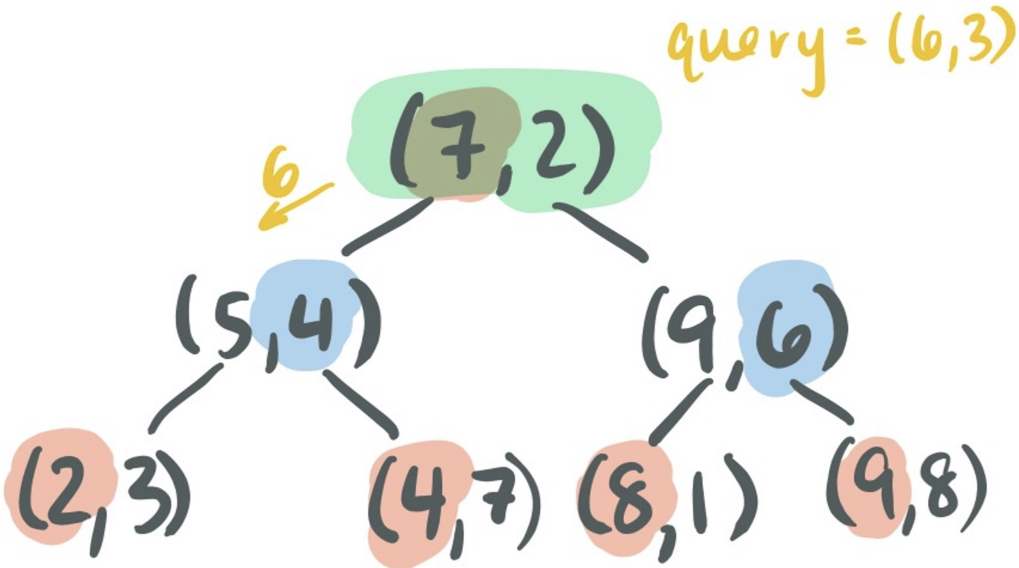


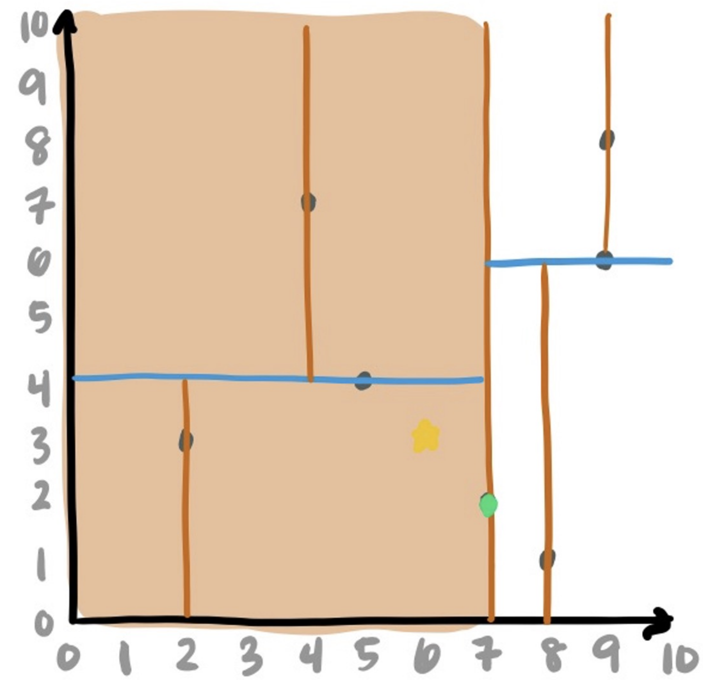
Nearest Neighbor - demo



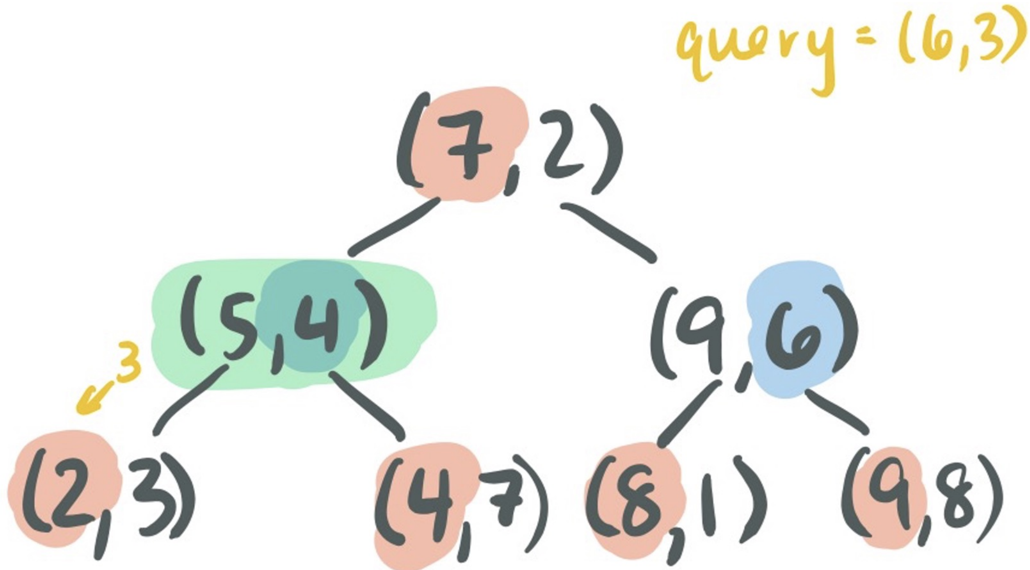


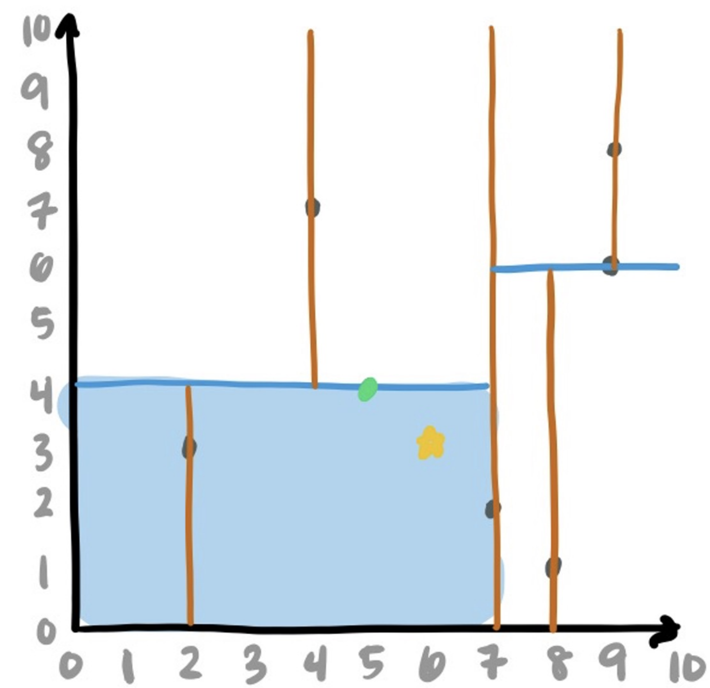
Nearest Neighbor - demo



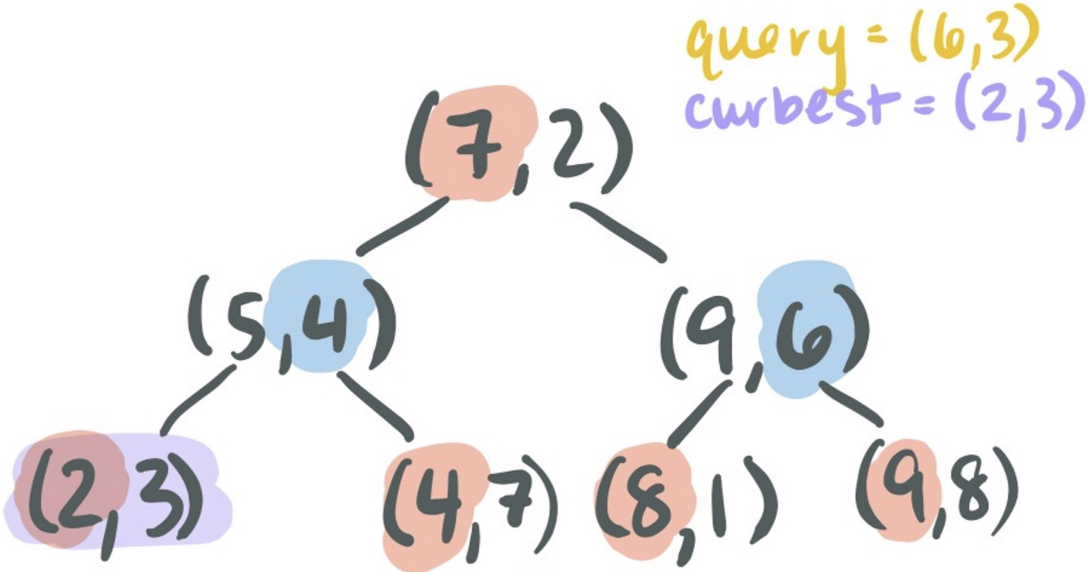


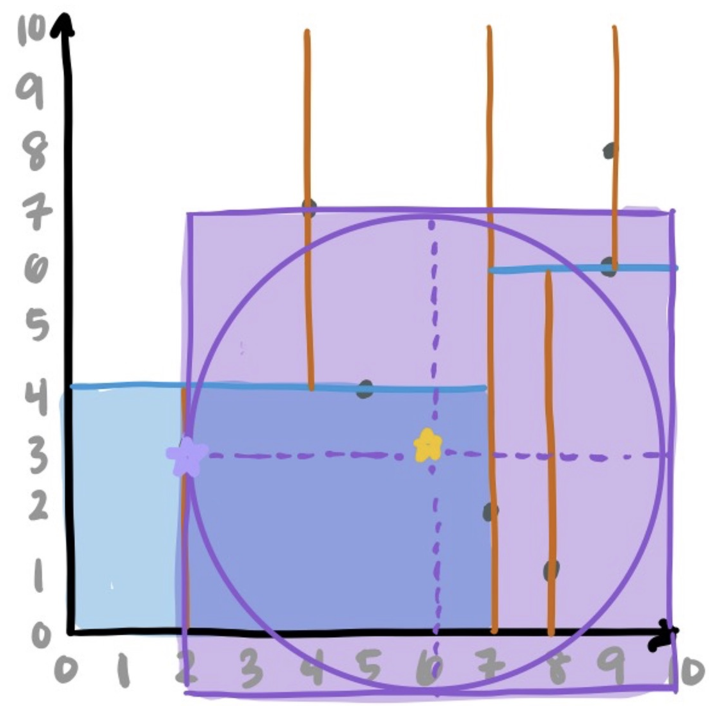
Nearest Neighbor - demo





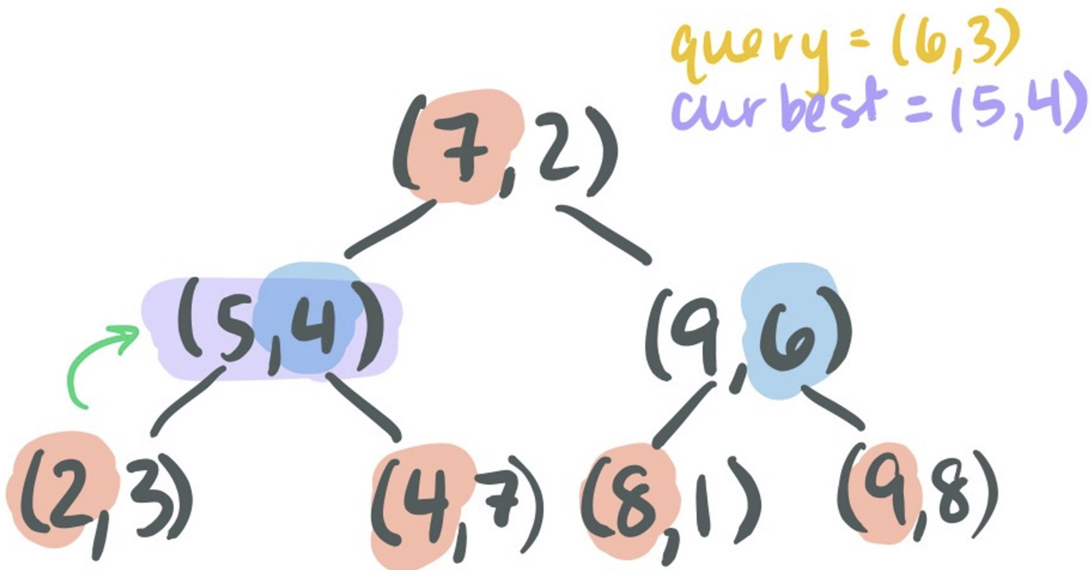
Nearest Neighbor - demo

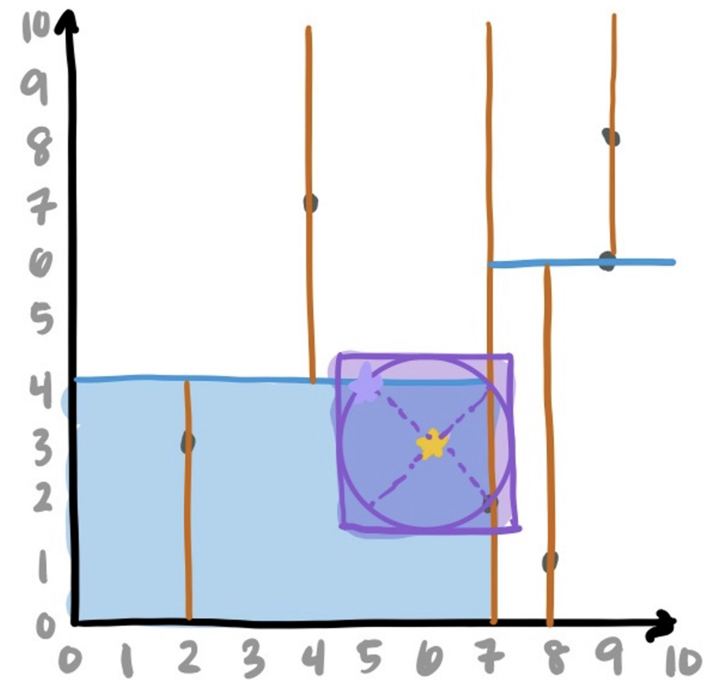




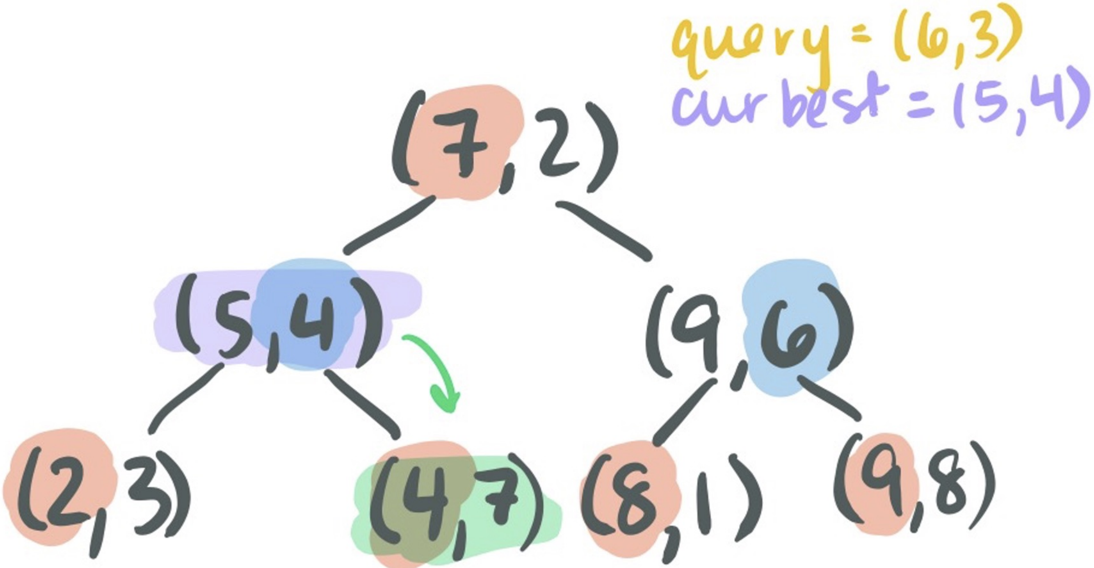
Nearest Neighbor - demo

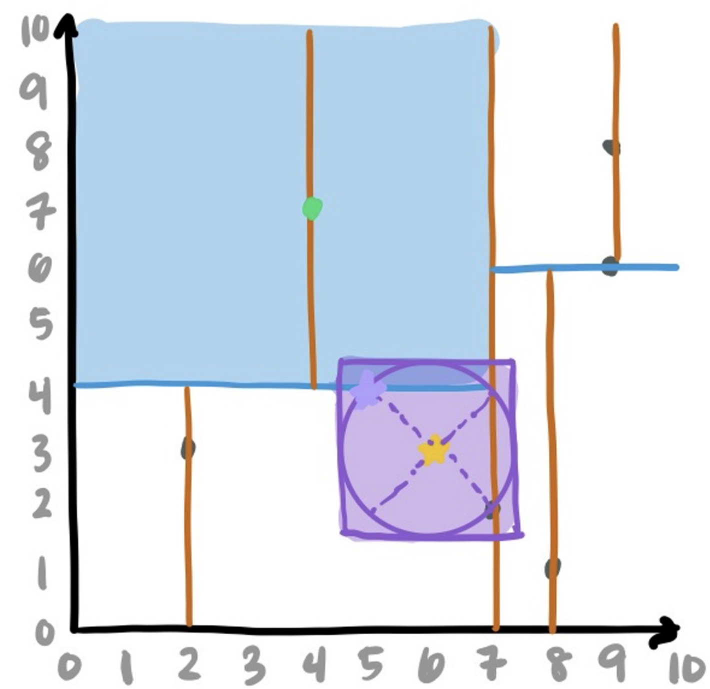
Backtracking: start recursing backwards -- store “best” possibility as you trace back





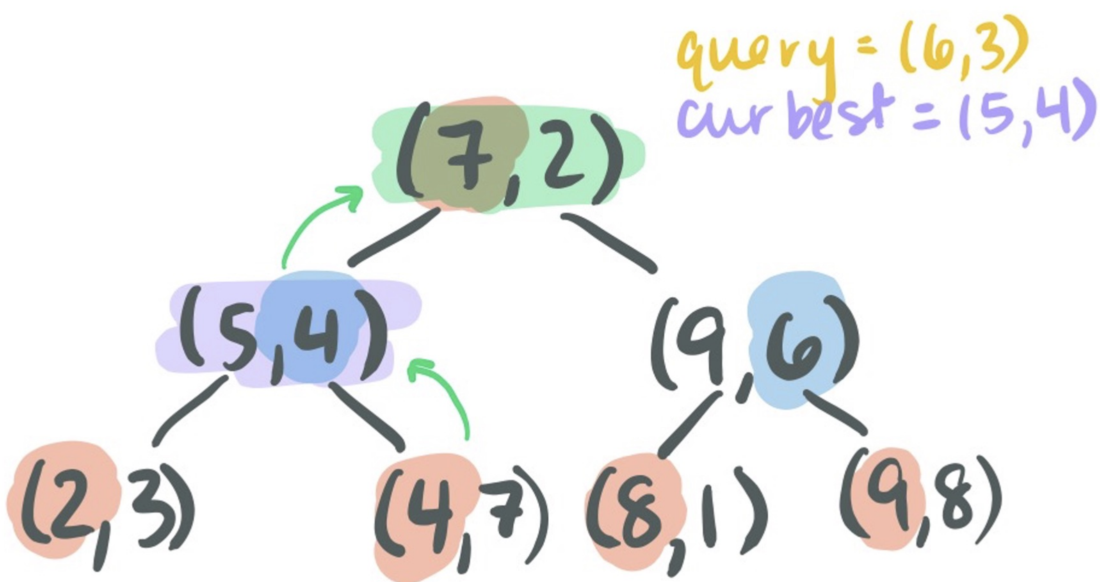
Nearest Neighbor - demo

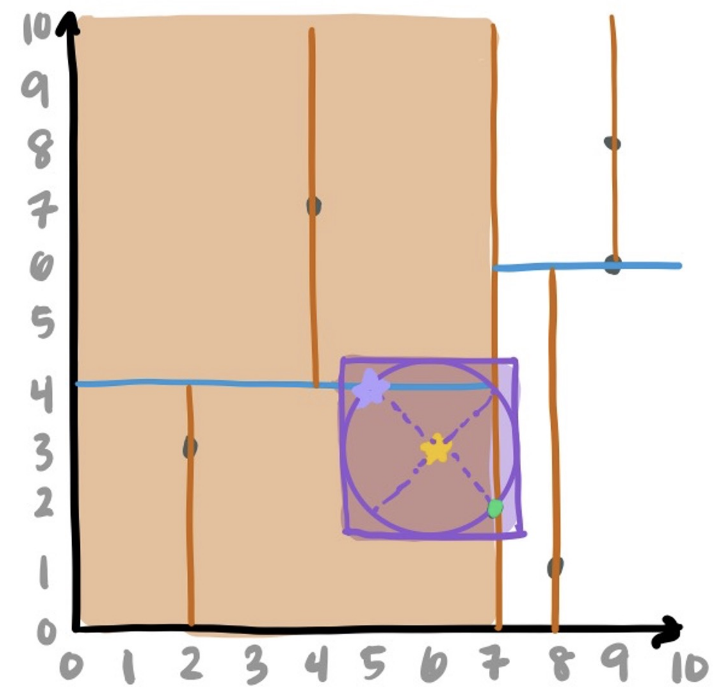




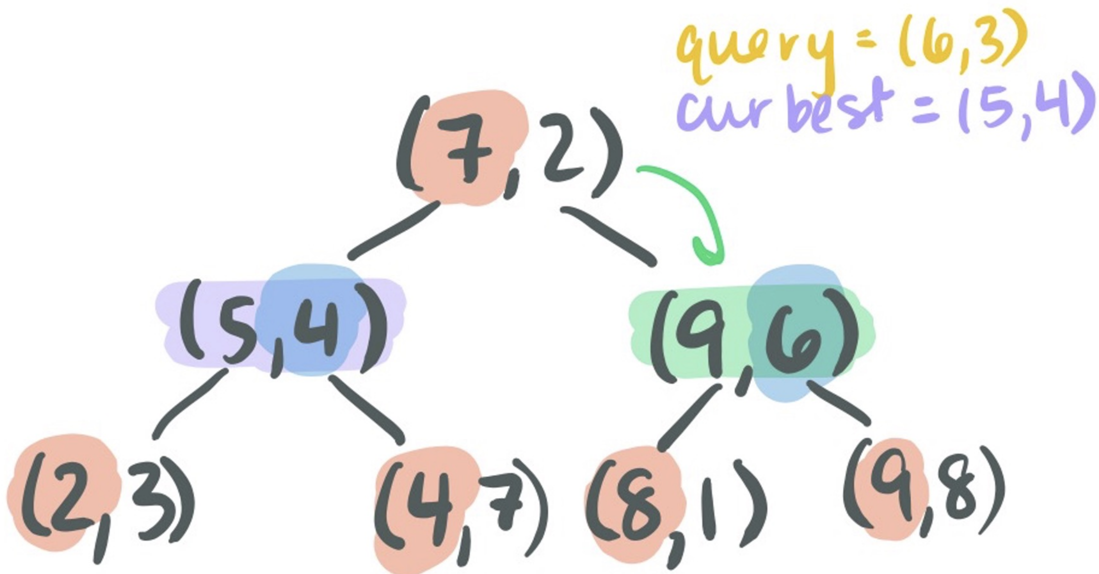
Nearest Neighbor - demo

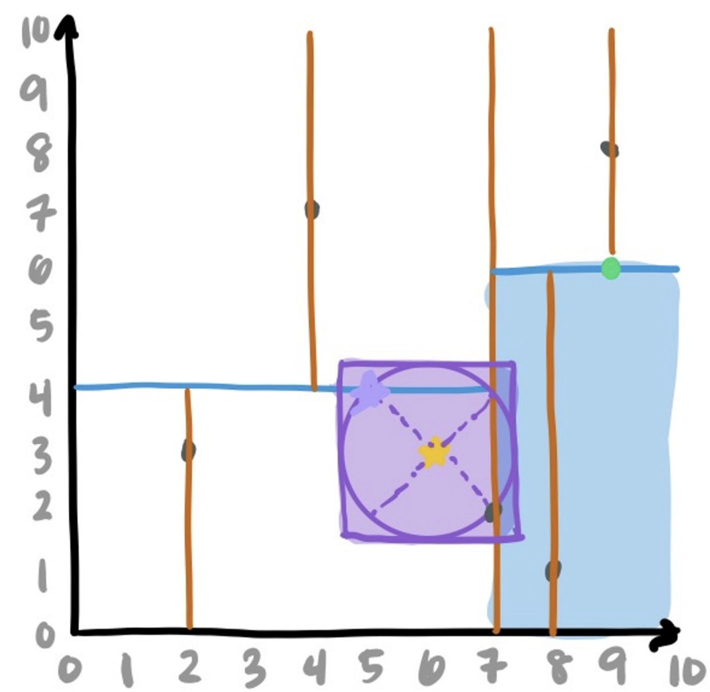
On ties, use `smallerDimVal` to determine which point remains `curBest`



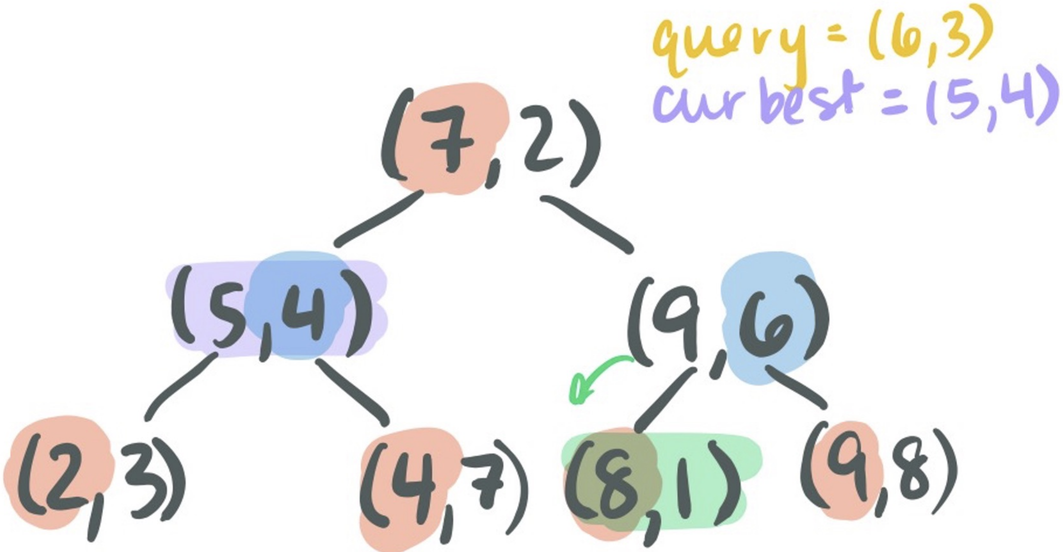


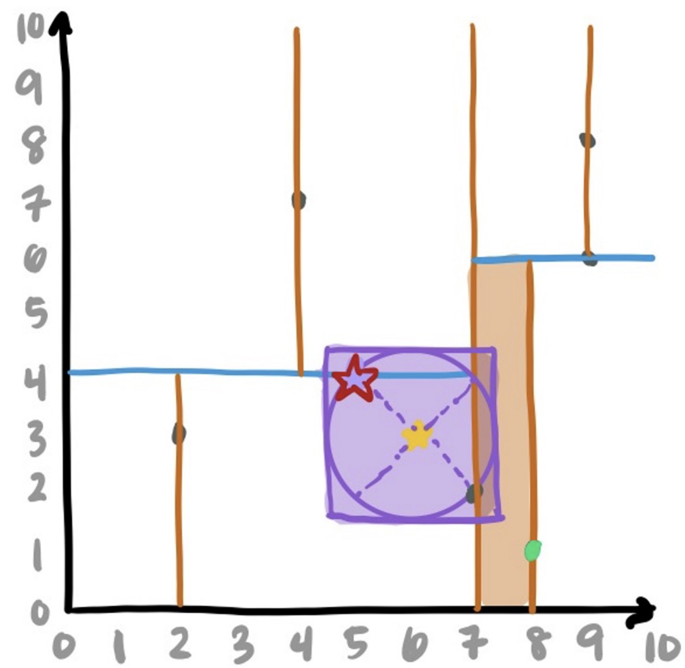
Nearest Neighbor - demo





Nearest Neighbor - demo





BEST: (5,4)

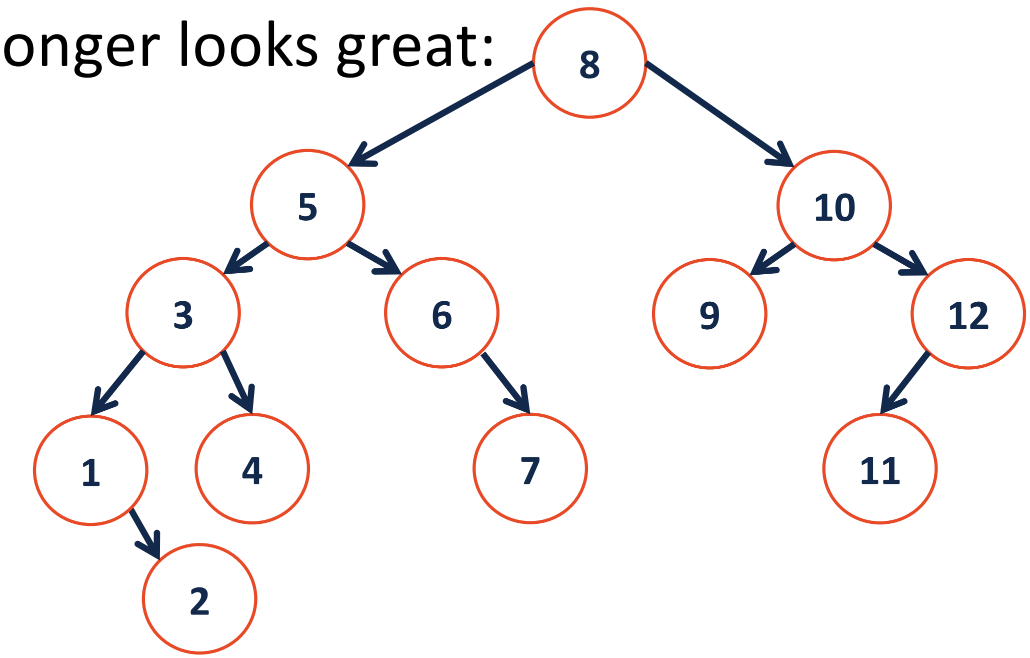
Every Data Structure So Far

	Unsorted Array	Sorted Array	Unsorted List	Sorted List	Binary Tree	BST	AVL
Find							
Insert							
Remove							
Traverse							

B-Tree Motivation

In Big-O we have assumed uniform time for all operations, but this isn't always true.

However, seeking data from the cloud may take 40ms+.
…an O(lg(n)) AVL tree no longer looks great:



BTree Motivations

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