

### AVL – Proof of Runtime

On Friday, we proved an upper-bound on the height of an AVL tree is  $2 * \lg(n)$  or  $O(\lg(n))$ .

AVL Trees	Red-Black Trees
Balanced BST	Balanced BST
Max height: $1.44 * \lg(n)$ <i>Q: Why is our proof <math>2 * \lg(n)</math>?</i>	<i>Functionally equivalent to AVL trees; all key operations runs in <math>O(h)</math> time.</i>
Rotations: - find:	Max height: $2 * \lg(n)$
- insert:	Rotations: - find:
- remove:	- insert:
	- remove:

In CS 225, we learned **AVL trees** because they're intuitive and I'm certain we could have derived them ourselves given enough time. A red-black tree is simply another form of a balanced BST that is also commonly used.

### Summary of Balanced BSTs:

(Includes both AVL and Red-Black Trees)

Advantages	Disadvantages

### Using a Red-Black Tree in C++

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

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

The map implements a dictionary ADT. Primary means of access is through the overloaded `operator[]`:

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

*This function can be used for both insert and find!*

Removing an element:

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

Range-based searching:

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

### Iterators and MP4

Three weeks ago, you saw that you can use an iterator to loop through data:

```
1 DFS dfs(...);
2 for ( ImageTraversal::Iterator it = dfs.begin();
3     it != dfs.end(); ++it ) {
4     std::cout << (*it) << std::endl;
5 }
```

You will use iterators extensively in MP4, creating them in Part 1 and then utilizing them in Part 2. Given the iterator, you can use the for-each syntax available to you in C++:

```
1 DFS dfs(...);
2 for ( const Point & p : dfs ) {
3     std::cout << p << std::endl;
4 }
```

The exact code you might use will have a generic `ImageTraversal`:

```
1 ImageTraversal & traversal = /* ... */;
2 for ( const Point & p : traversal ) {
3     std::cout << p << std::endl;
4 }
```

### Running Time of Every Data Structure So Far:

	Unsorted Array	Sorted Array	Unsorted List	Sorted List
Find				
Insert				
Remove				
Traverse				

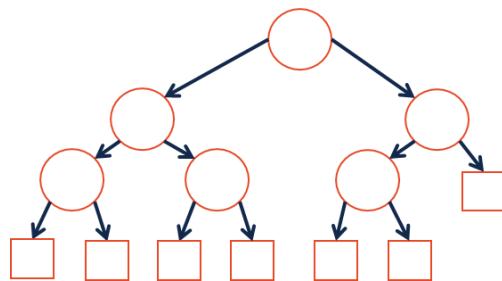
	Binary Tree	BST	AVL
Find			
Insert			
Remove			
Traverse			

### Range-based Searches:

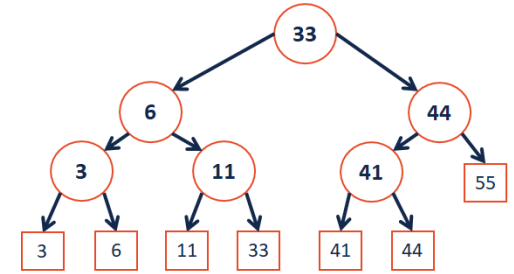
Q: Consider points in 1D:  $p = \{p_1, p_2, \dots, p_n\}$ .  
 ...what points fall in  $[11, 42]$ ?



### Tree Construction:



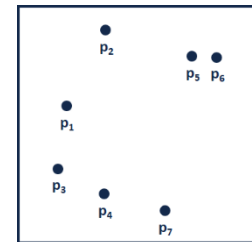
### Range-based Searches:



### Running Time:

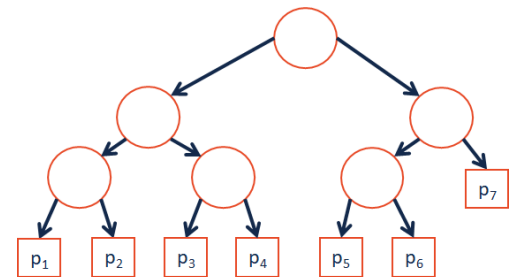
### Extending to k-dimensions:

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



...what points are inside a range (rectangle)?  
 ...what is the nearest point to a query point  $q$ ?

### Tree Construction:



### CS 225 – Things To Be Doing:

1. Quiz 5 ends today (Next quiz after break!)
2. Mp\_traversal extra credit +7 due tonight
3. lab\_avl released this week; details on Wednesday
4. Daily POTDs are ongoing!