CSE12 - Lecture 18 - B00

Friday, November 4, 2022 9:00 AM

PA6 due Tuesday
PA7 Late/Resubmit due Tuesday

Binary Search Tree (BST)

implements Map<K,V>

```
class BST<K, V> {
  Node<K, V> root;
  BST() (this.root = null);
  BST(Node<K, V> root) { this.root = root; }

  V get(Node<K, V> node, K key) {
    if (node == null) { //throw error }   ] Not Found
    if (node.key.equals(key)) {           ] found
      return node.value;
    }
    if (node.key > key) {                 ] greater
      return get(node.left, key);
    }
    else {                                ] lesser
      return get(node.right, key);
    }
  }

  public V get(Key key) {
    return this.get(root, key);
  }
}
```

private — base case — recursive case — public

```
class Node<K,V> {
  K key;
  V value;
  Node<K,V> left;
  Node<K,V> right;
  public Node(K key, V value,
              Node<K,V> left,
              Node<K,V> right) {
    this.key = key;
    this.value = value;
    this.left = left;
    this.right = right;
  }
}
```

Where is the get() method broken?

> does not work for Objects

How can we fix the get() method to work with Objects?

Interface → compare() 0, 1, -1
- Comparator / also an Object
  - pass to the constructor
  - (c) save in a field
- Comparable
  - compareTo()
    - < 0 less than
    - 0 equal
    - > 0 greater than

public String implements Comparable

What error should we throw in get() if the key isn't found?

NoSuchElementException() / ElementNotFoundException()

What would the code that uses get() look like to prevent the program crashing if the key is missing?

```
BST<> tree ____
try {
    tree.get(?);
}
catch (NoSuchElementException e) {
    // what to do?
    // print error message
}
catch (Exception e) {
}
```

```
boolean find(E toFind) {
  Comparable comp = (Comparable) toFind;
  while (comp
    if (comp.compareTo( ) == 0)
      return true
  }
  return false;
```

runtime error? bad

(1)

```
<E extends Comparable>
class Test<E extends Comparable> {
    boolean find(E toFind) {
        while (
            if (toFind.compareTo(___) > 0
        }
    }
```

Assume the key and value are identical for this example:

Trace the path for get(4)
How many nodes does it touch?

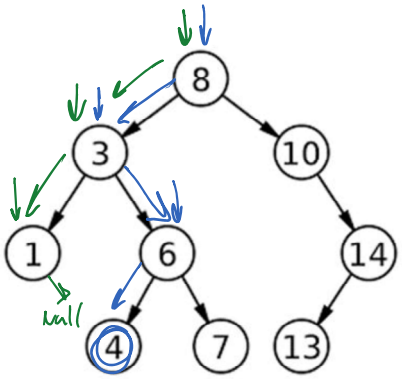
4 nodes

Trace the path for get(2)
How many nodes does it touch?

3 nodes

What happens when the node isn't found?

throw exception



key less than node's key
→ go left

key greater than node's key
→ go right

key equal to node's key
→ found it, return value

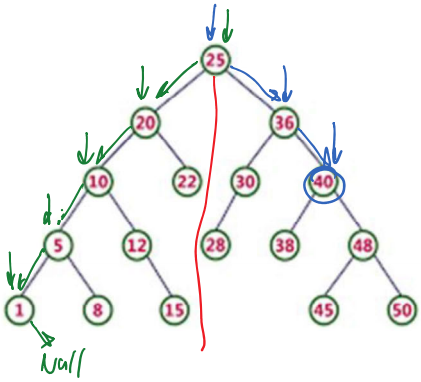
Assume the key and value are identical for this example:

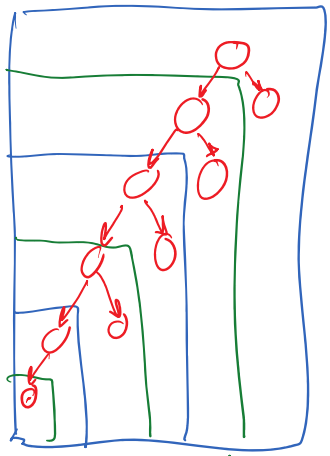
Trace the path for get(40)
How many nodes does it touch?

3 nodes

Trace the path for get(4)
How many nodes does it touch?

5 nodes





recursive data structure

Binary Search

$\log_2(N)$