CSE12 - Lecture 18 - C00

Friday, November 4, 2022 11:00 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) {       // private
    if (node == null) { //throw error }  ] not found      (base case)
    if (node.key.equals(key)) {          ] found
      return node.value;
    }
    if (node.key > key) {                ] node greater left   (recursive case)
      return get(node.left, key);
    }
    else {
      return get(node.right, key);       ] node smaller right
    }
  }

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

```
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 on Objects

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

Interface → compare( ) 0, -1, 1
- Comparator / also an Object
  - (2) ↳ pass to the constructor, save as 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 ____
    if (comp.compareTo(__) == 0) {
      return true
    }
  }
  return false;
}
```

run time? 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:

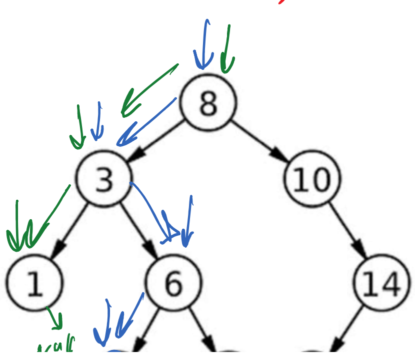
Trace the path for get(4)
How many nodes does it touch?

4 nodes

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

3 nodes

What happens when the node isn't found?



<
key smaller node.key
- go left

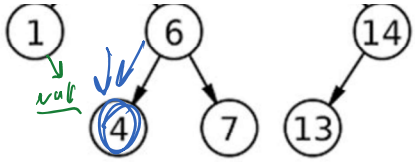
>
key larger node.key
- go right

==
key equal

3 nodes

What happens when the node isn't found?

throw exception



- go right

Use equal ==

- return value

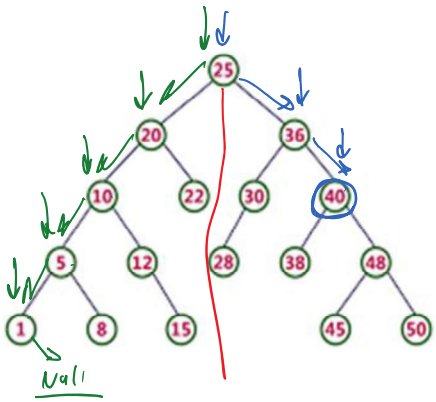
Assume the key and value are identical for this example:

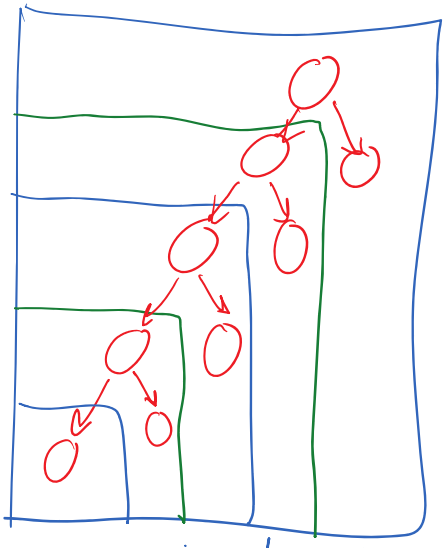
Trace the path for get(40)
How many nodes does it touch?

3 nodes

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

5 nodes





recursive data structure

Binary Search

$\log_2(N)$