CSE12 - Lecture 18

Monday, November 13, 2023 8:00 AM

PA6 and PA4 Late/Resubmit - due Wednesday @ 8am
Exam2 - Wed - covers from last exam up to and including last Wednesday (hashmaps)

Lecture 18

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 }
    if (node.key.equals(key)) {
      return node.value;
    }
    if (node.key > key) {
      return get(node.left, key);
    }
    else {
      return get(node.right, key);
    }
  }

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

private

base case

Not found

found

recursive case

key is smaller

key is greater

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

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

Interface → compare() method 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<String> {
}
```

```
boolean find ( E toFind, E key)
  Comparable<E> comp = (Comparable<E>) toFind;
  loop { comp.Compareto ( key)
}
```

runtime error? ↓ bad!

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

more appropriate ↓

NoSuchElementException() / ElementNotFoundException()

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

```
BSTree tree ______
try {
    tree.get (x);
}
catch (ElementNotFoundException e) {
    // what do we do?
    // print error?
}
catch (Exception e) {
    // bigger error / other error
}
```

<E extends Comparable>

```
class Test < E extends Comparable > {
(1) boolean find ( E toFind) {
        if ( toFind.compareTo (...) ) {

}
```

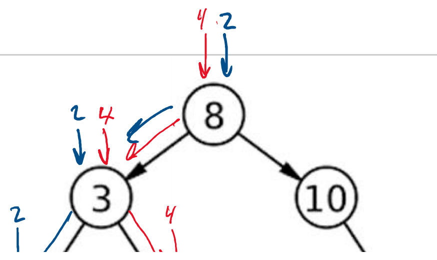
Name: ______________________ PID: ______________________ Code: 2208

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)



Key smaller Node.key?

How many nodes does it touch?

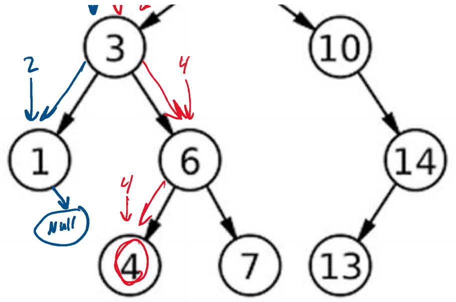
4 Nodes

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

3 nodes, 4 comparisons

What happens when the node isn't found?

throws exception

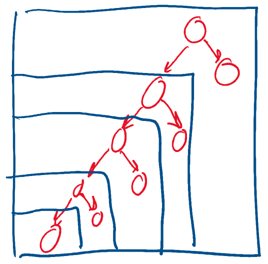


Key smaller node.key?
→ go left

Key greater node.key?
→ go right

key equal node.key
→ found it
→ return value

BST → recursive data structure



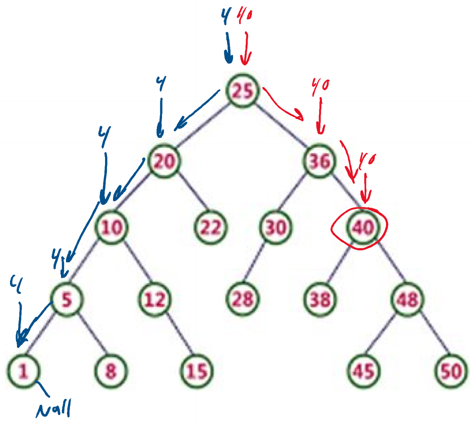
Assume the key and value are identical for this example:

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/ 6 comparisons



binary search

log₂(N)