# Lecture 6 - Sep 22

## *Self-Balancing Binary Search Trees*

***Implementing BST in Java***
***BST Operations: Search & Insert***
***Tree Rotation, In-Order Traversal***

## Announcements/Reminders

- First Class (Syllabus) recording & notes posted
- Today's class: notes template posted
- Exercises:
  - + Tutorial Week 1 (2D arrays)
  - + Tutorial Week 2 (2D arrays, Proving Big-O)
  - + Tutorial Week 3 (avg case analysis on doubling strategy)
- This Wednesday's class will have a **later** start: **4:10 PM**

# Generic, Binary Tree Nodes

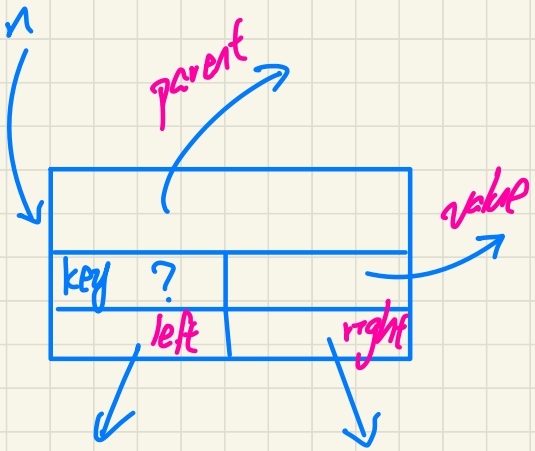
```
public class BSTNode<E> {
  private int key; /* key */
  private E value; /* value */
  private BSTNode<E> parent; /* unique parent node */
  private BSTNode<E> left; /* left child node */
  private BSTNode<E> right; /* right child node */

  public BSTNode() { ... }
  public BSTNode(int key, E value) { ... }

  public boolean isExternal() {
    return this.getLeft() == null && this.getRight() == null;
  }
  public boolean isInternal() {
    return !this.isExternal();
  }
  public int getKey() { ... }
  public void setKey(int key) { ... }
  public E getValue() { ... }
  public void setValue(E value) { ... }
  public BSTNode<E> getParent() { ... }
  public void setParent(BSTNode<E> parent) { ... }
  public BSTNode<E> getLeft() { ... }
  public void setLeft(BSTNode<E> left) { ... }
  public BSTNode<E> getRight() { ... }
  public void setRight(BSTNode<E> right) { ... }
}
```

searching

root of left-subtree



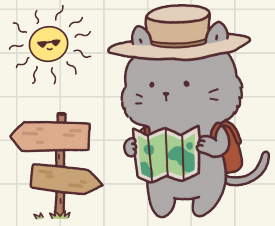

**Compare**:

+ prev ref.
+ next ref.

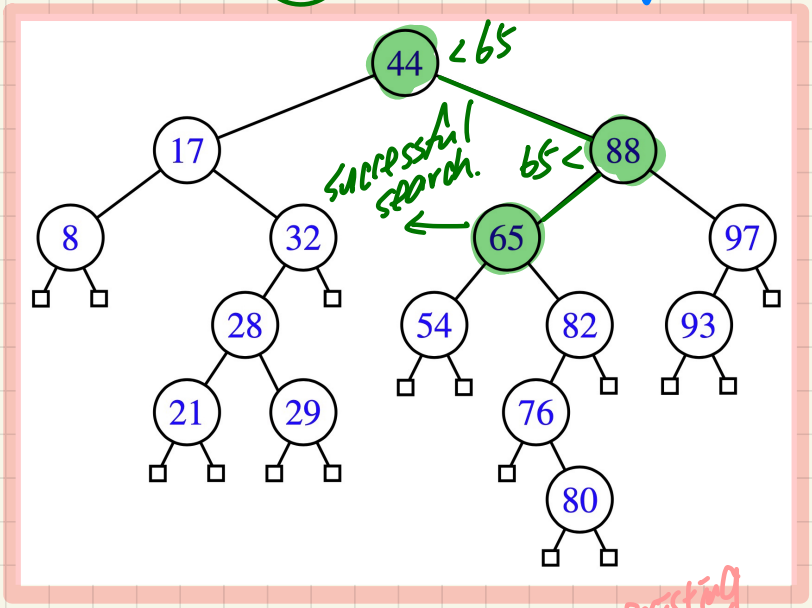
in a DLN.

# BST Operation: Searching a Key

BST ⇒ in-order traversal gives a sorted seq. of keys.

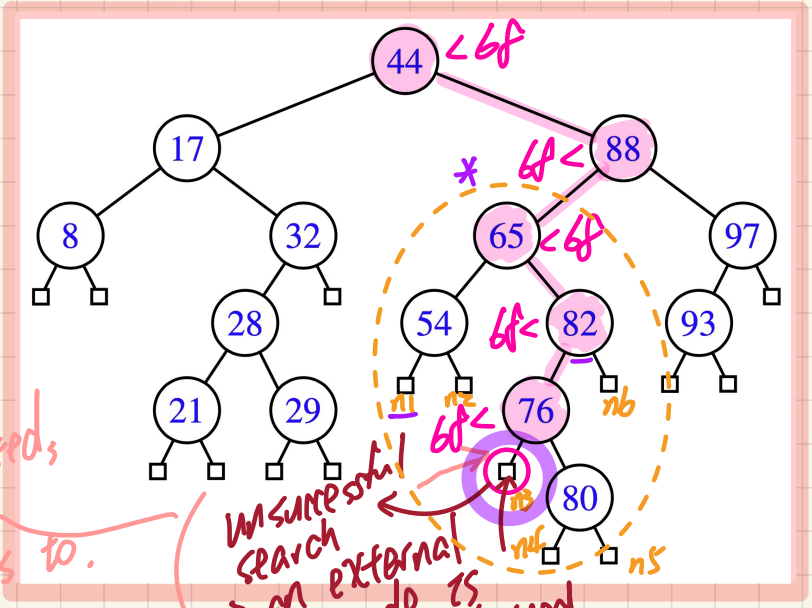


Search key 65




successful search.

Search key 68




unsuccessful search → an external node is returned

if the non-existing key 68 is to be inserted, this ext. node is where it belongs to.

# Tracing: Searching through a BST

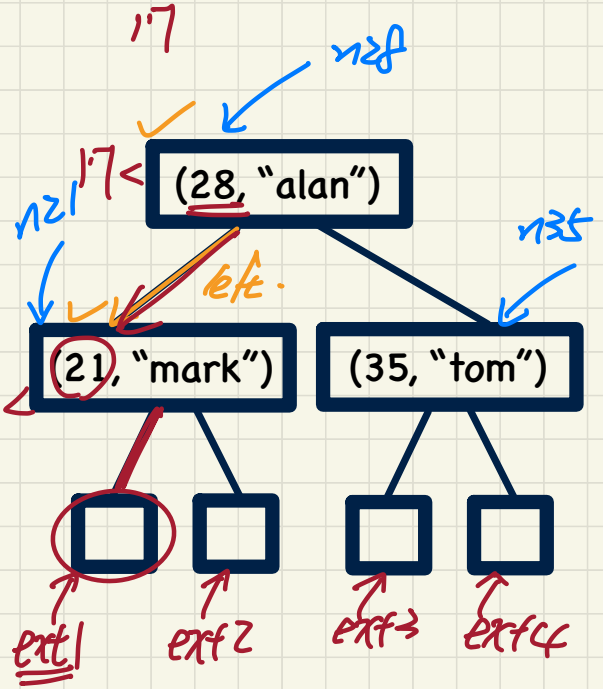
```
@Test
public void test_binary_search_trees_search() {
  BSTNode<String> n28 = new BSTNode<>(28, "alan");
  BSTNode<String> n21 = new BSTNode<>(21, "mark");
  BSTNode<String> n35 = new BSTNode<>(35, "tom");
  BSTNode<String> extN1 = new BSTNode<>();
  BSTNode<String> extN2 = new BSTNode<>();
  BSTNode<String> extN3 = new BSTNode<>();
  BSTNode<String> extN4 = new BSTNode<>();
  n28.setLeft(n21); n21.setParent(n28);
  n28.setRight(n35); n35.setParent(n28);
  n21.setLeft(extN1); extN1.setParent(n21);
  n21.setRight(extN2); extN2.setParent(n21);
  n35.setLeft(extN3); extN3.setParent(n35);
  n35.setRight(extN4); extN4.setParent(n35);

  BSTUtilities<String> u = new BSTUtilities<>();
  /* search existing keys */
  assertTrue(n28 == u.search(n28, 28));
  assertTrue(n21 == u.search(n28, 21));
  assertTrue(n35 == u.search(n28, 35));
  /* search non-existing keys */
  assertTrue(extN1 == u.search(n28, 17)); /* *17* < 21 */
  assertTrue(extN2 == u.search(n28, 23)); /* 21 < *23* < 28 */
  assertTrue(extN3 == u.search(n28, 33)); /* 28 < *33* < 35 */
  assertTrue(extN4 == u.search(n28, 38)); /* 35 < *38* */
}
```

node creations

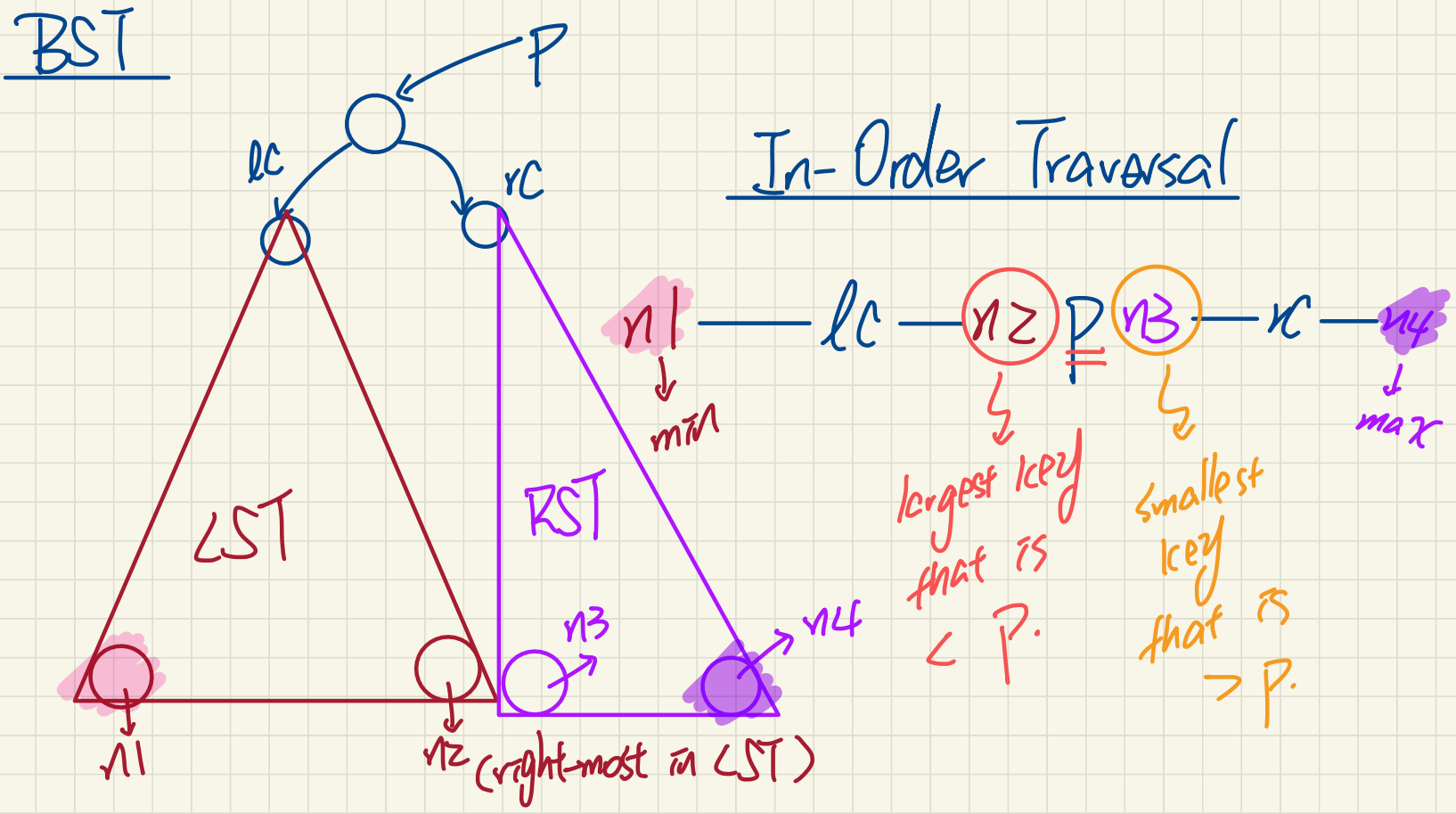
comparing nodes.

internal nodes (successful search).



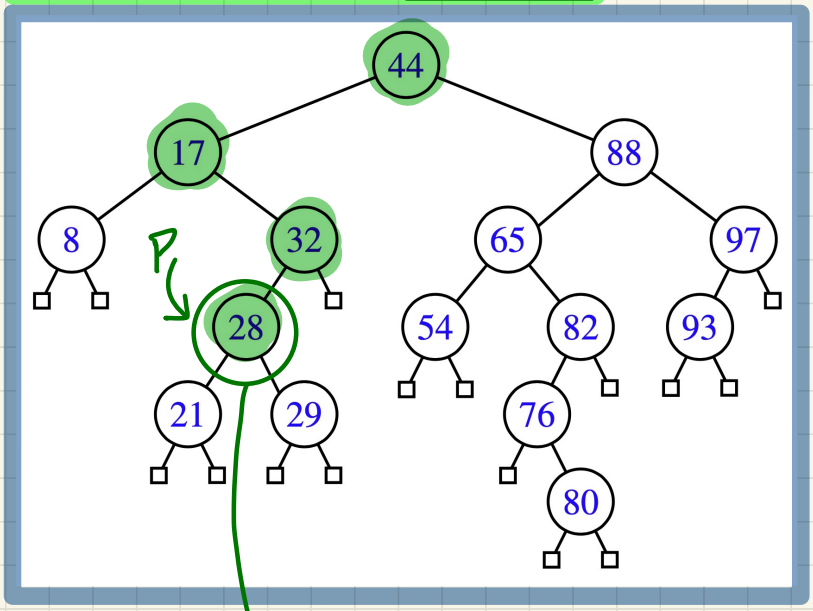

exercise: why a particular ext. node is returned.

# Visual Summary: In-Order Traversal on BST



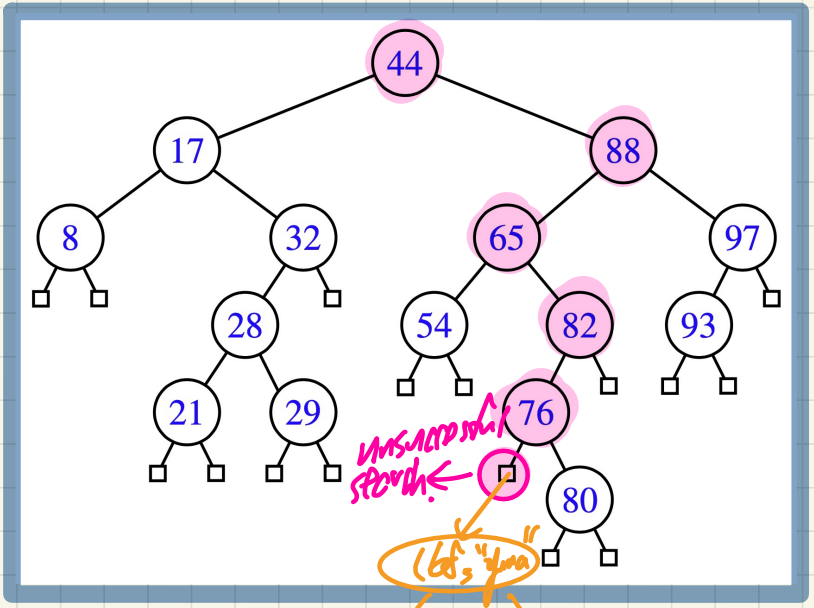
# Visualizing BST Operation: Insertion

Insert Entry (28, "suyeon")

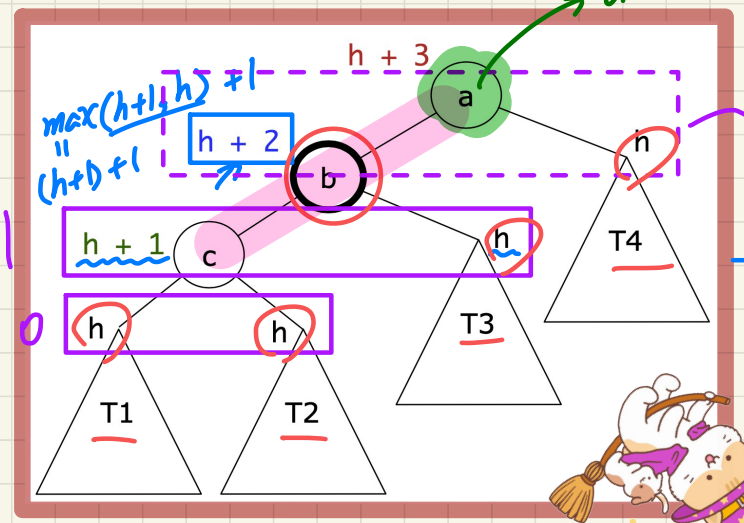



replace the value by the argument

Insert Entry (68, "yuna")

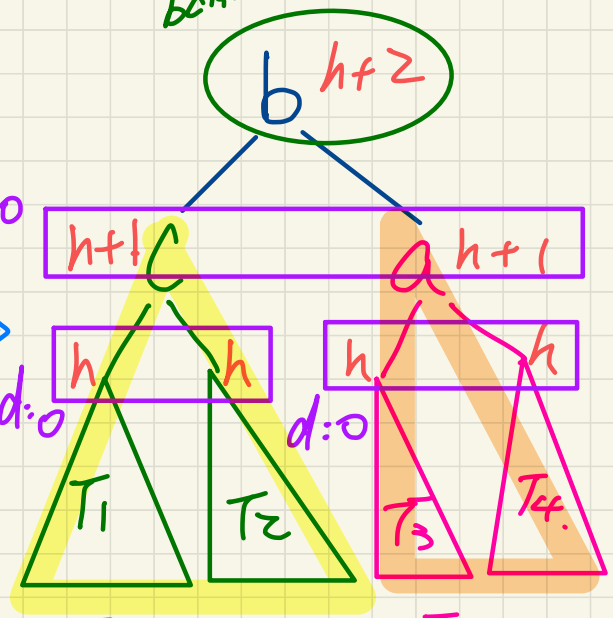


# Restoring Balance via Rotations




Rotate on the middle node b to the right

balance!




Before Rotation, I.O.T:

$\langle T_1, c, T_2, b, T_3, a, T_4 \rangle$

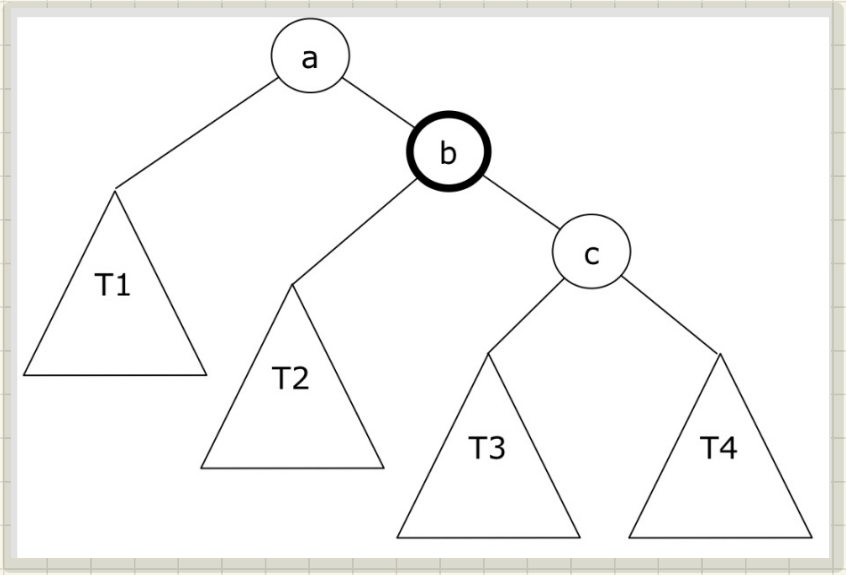
After Rotation, I.O.T:

$\langle T_1, c, T_2, b, T_3, a, T_4 \rangle$

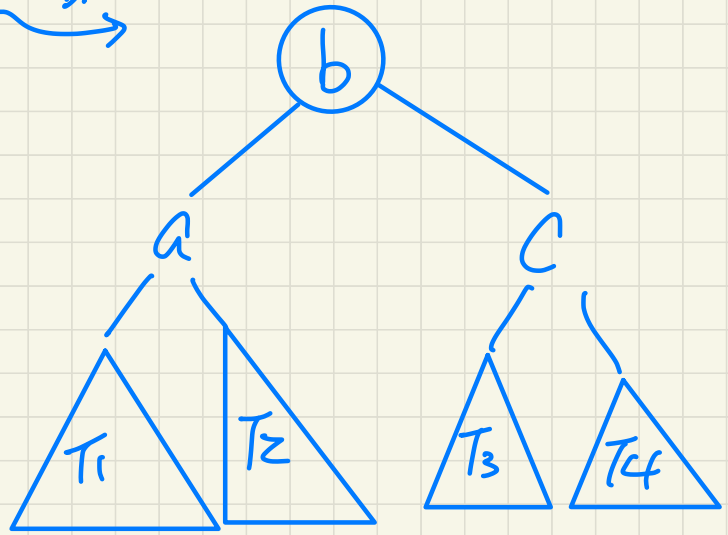
Q. Is the above tree balanced? YES

Q. After a right-rotation on node b, is the resulting tree still a BST? YES.

# Trinode Restructuring: Single, Left Rotation




left rotation on b




I.O.T.

$\langle T_1, a, T_2, b, T_3, c, T_4 \rangle$

# Trinode Restructuring after Insertion: Left Rotation

- Insert the following sequence of keys into an empty BST:

  <44, 17, 78, 32, 50, 88, 95>

- Insert 100 into the BST.