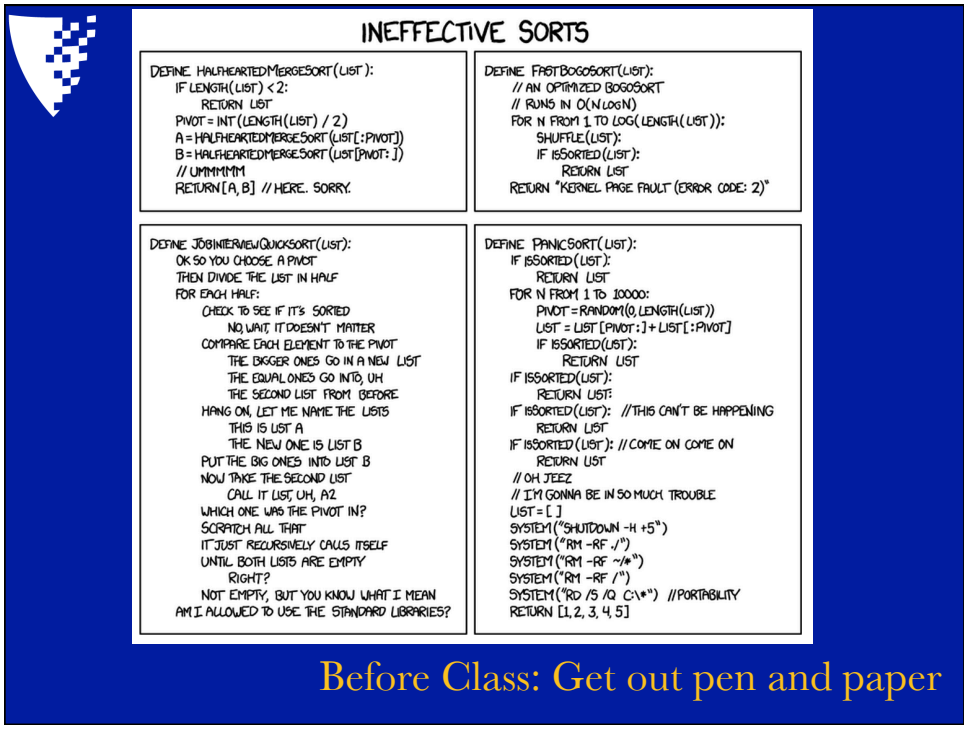
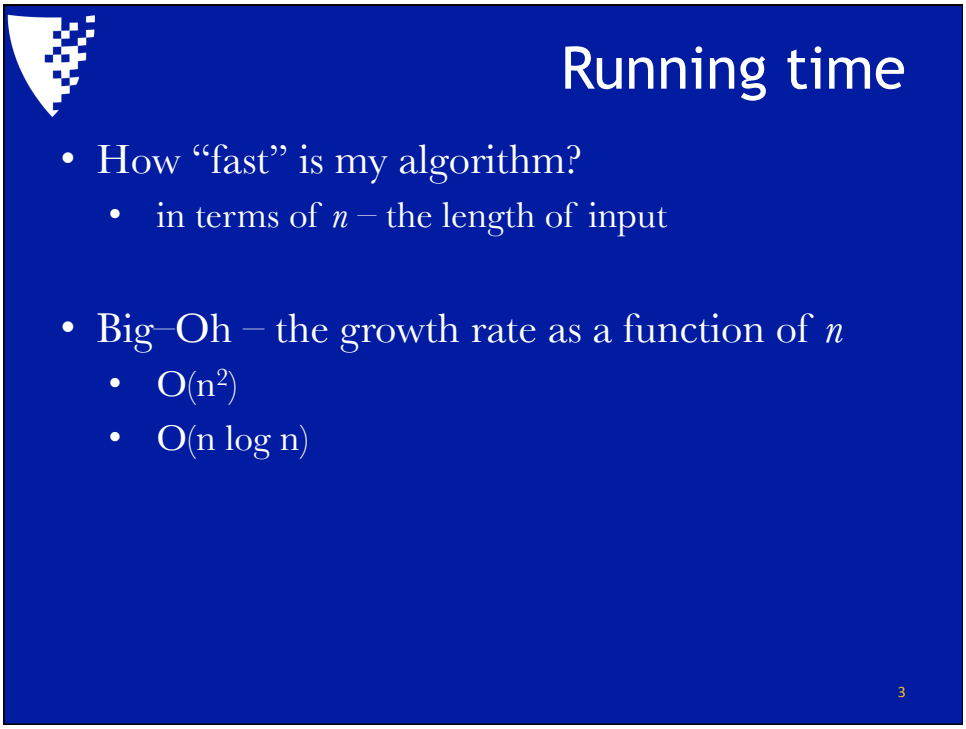
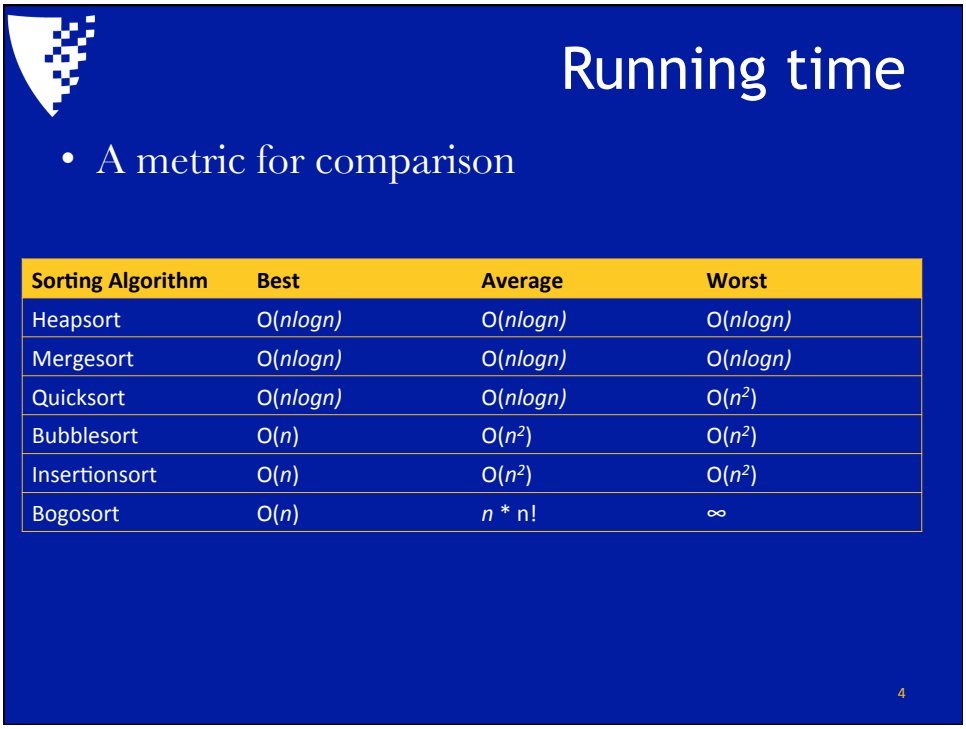


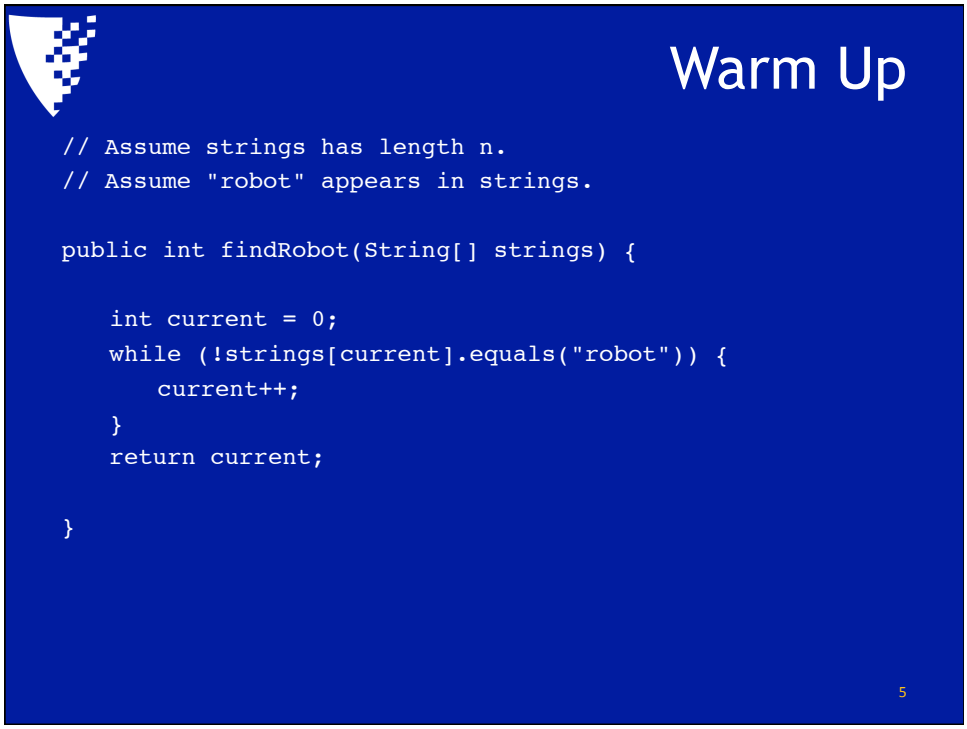

Before Class: Get out pen and paper

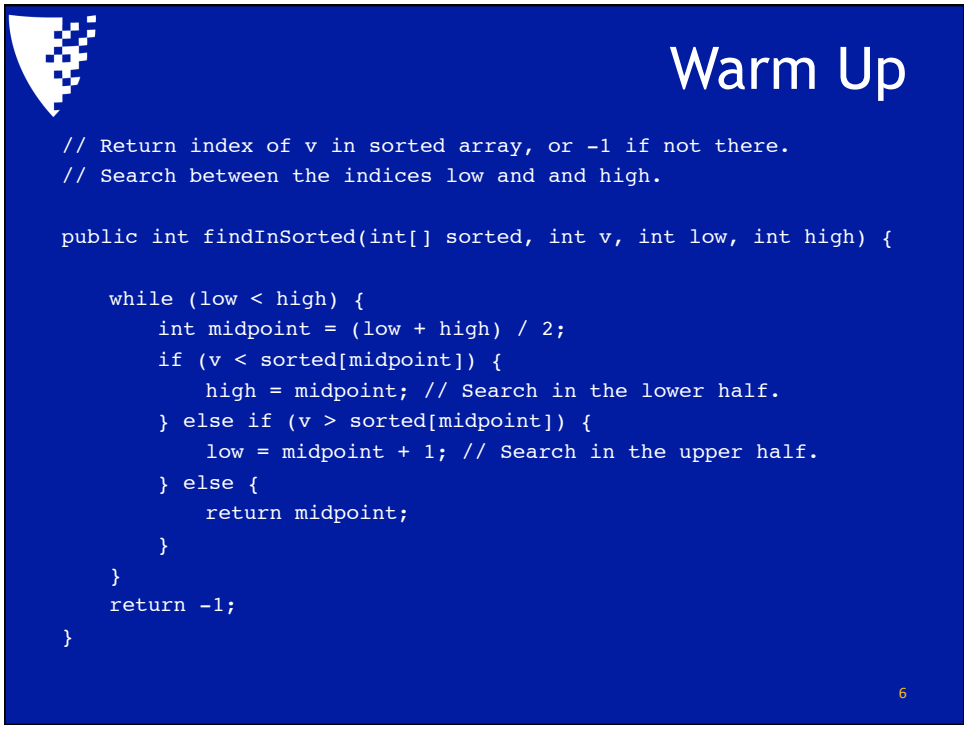
# Announcements

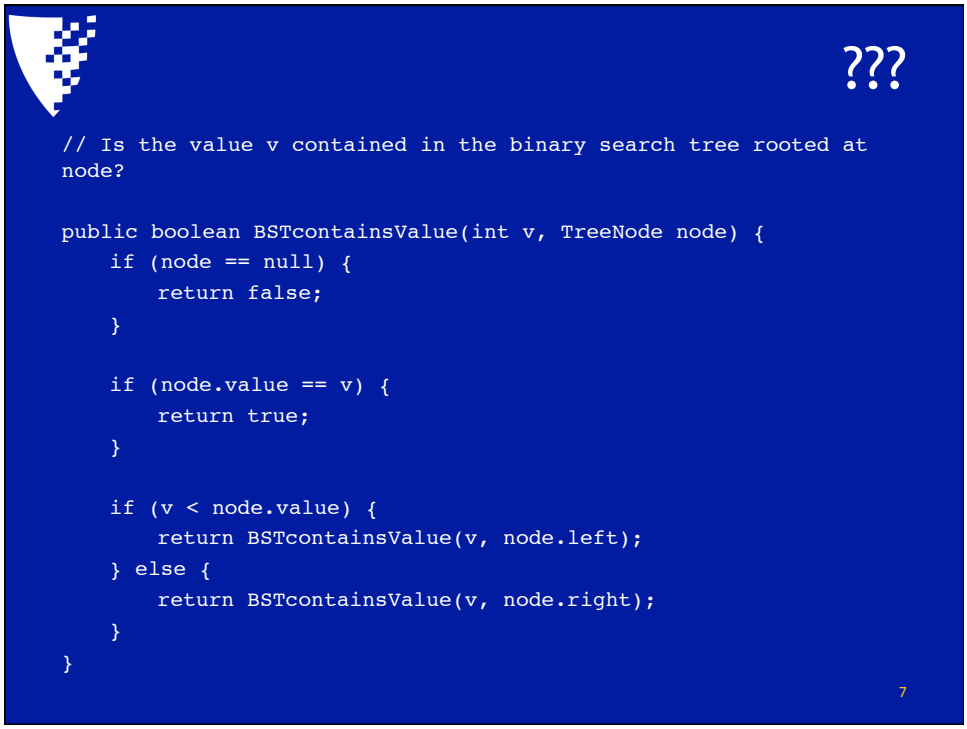
- Exams
  - will be returned on Wednesday
- Boggle
  - due April 4
- Apt Set 6 – Posted tomorrow
  - due April 9

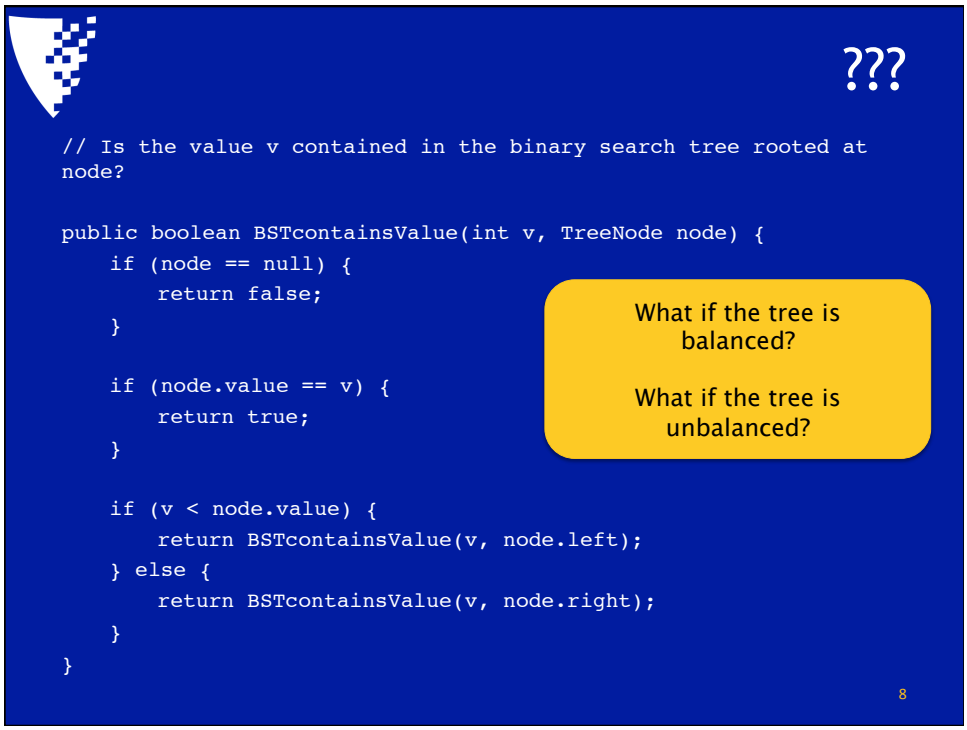

Running time
How "fast" is my algorithm?
in terms of n – the length of input
Big–Oh – the growth rate as a function of n
O(n²)
O(n log n)
3

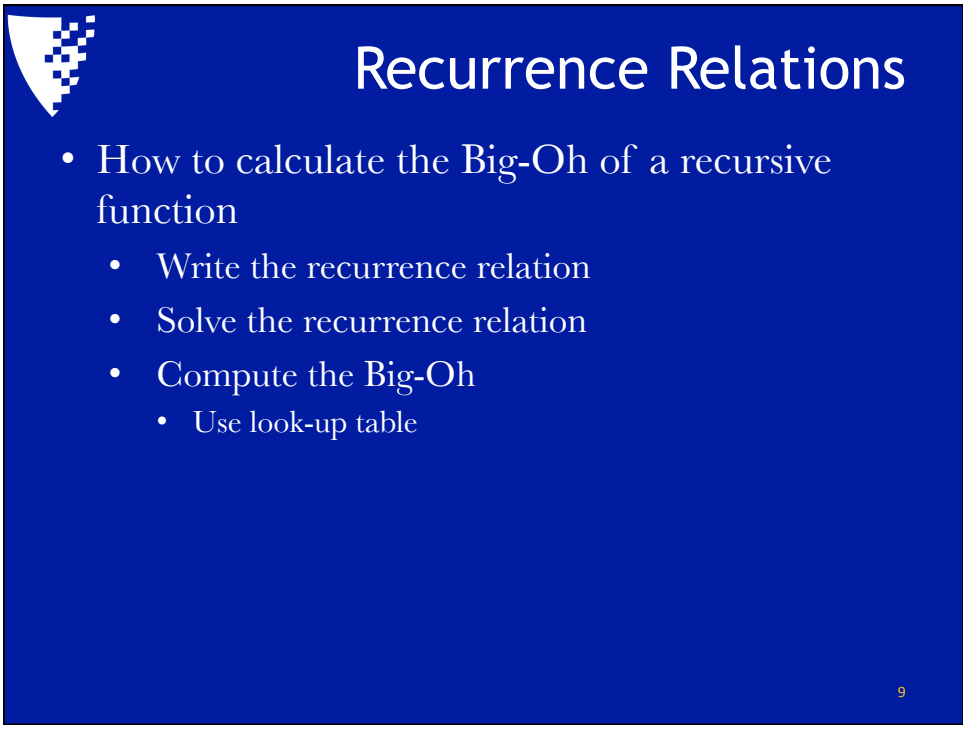

Running time
A metric for comparison
Sorting Algorithm
Best
Average
Worst
Heapsort
O(nlogn)
O(nlogn)
O(nlogn)
Mergesort
O(nlogn)
O(nlogn)
O(nlogn)
Quicksort
O(nlogn)
O(nlogn)
O(n²)
Bubblesort
O(n)
O(n²)
O(n²)
Insertionsort
O(n)
O(n²)
O(n²)
Bogosort
O(n)
n * n!
∞
4

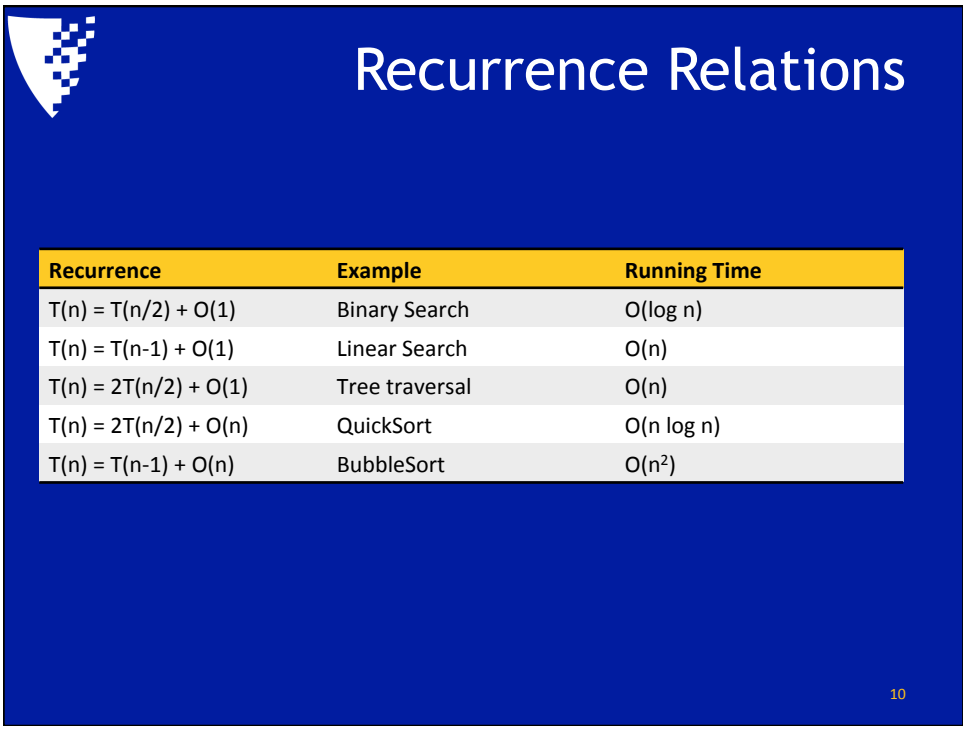

Warm Up
// Assume strings has length n.
// Assume "robot" appears in strings.
public int findRobot(String[] strings) {
int current = 0;
while (!strings[current].equals("robot")) {
current++;
}
return current;
}
5

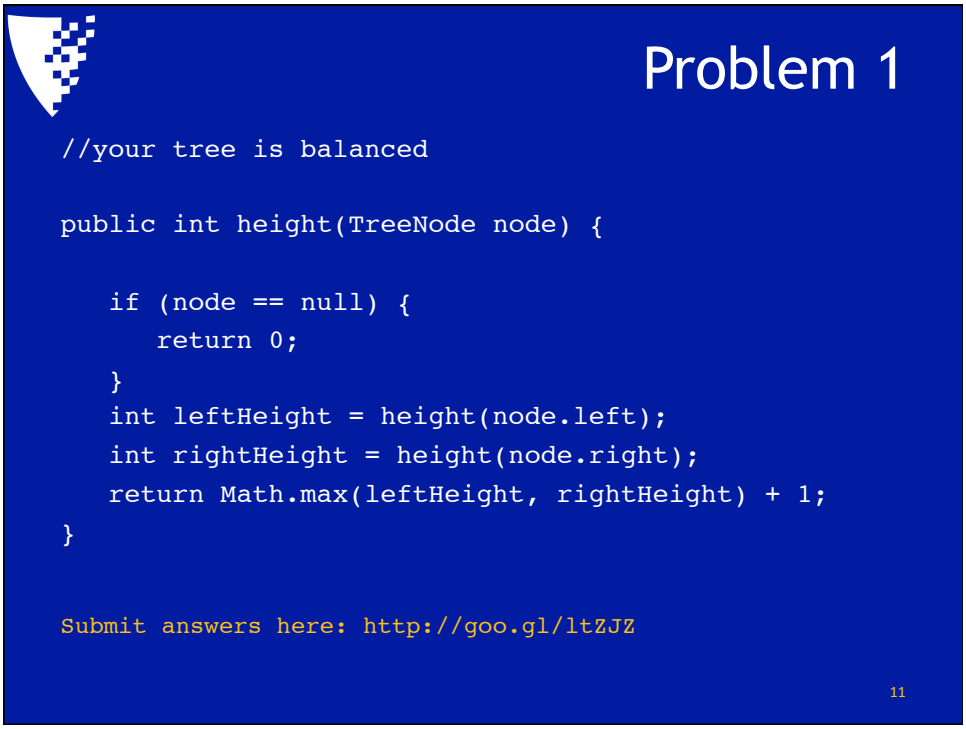

Warm Up
// Return index of v in sorted array, or -1 if not there.
// Search between the indices low and and high.
public int findInSorted(int[] sorted, int v, int low, int high) {
while (low < high) {
int midpoint = (low + high) / 2;
if (v < sorted[midpoint]) {
high = midpoint; // Search in the lower half.
} else if (v > sorted[midpoint]) {
low = midpoint + 1; // Search in the upper half.
} else {
return midpoint;
}
}
return -1;
}
6

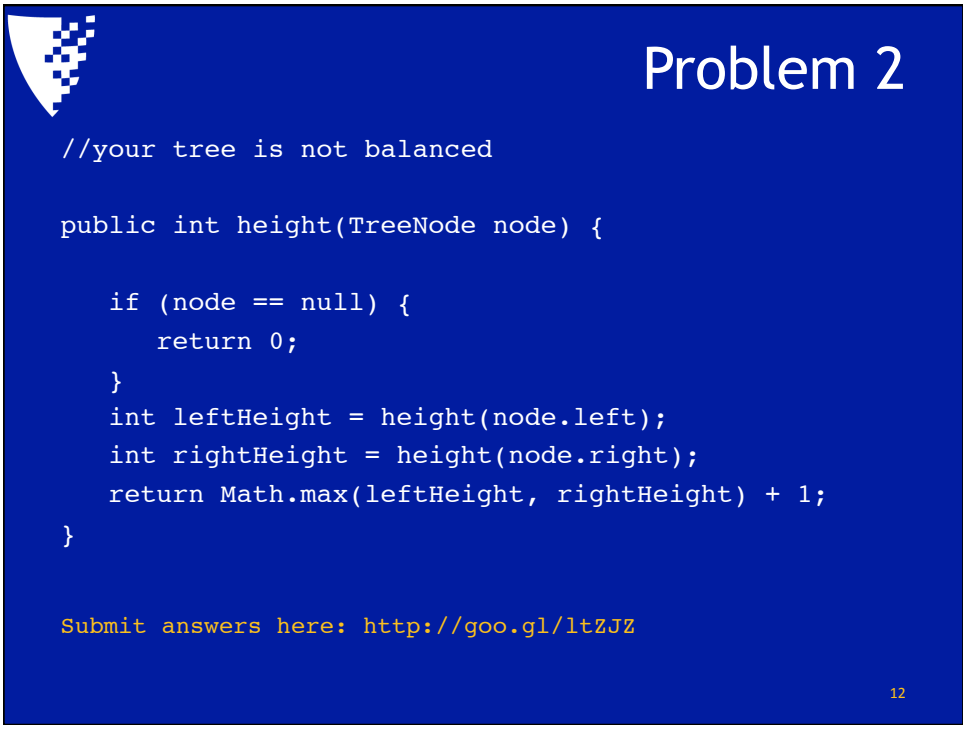

???
// Is the value v contained in the binary search tree rooted at node?
public boolean BSTcontainsValue(int v, TreeNode node) {
if (node == null) {
return false;
}
if (node.value == v) {
return true;
}
if (v < node.value) {
return BSTcontainsValue(v, node.left);
} else {
return BSTcontainsValue(v, node.right);
}
}
7

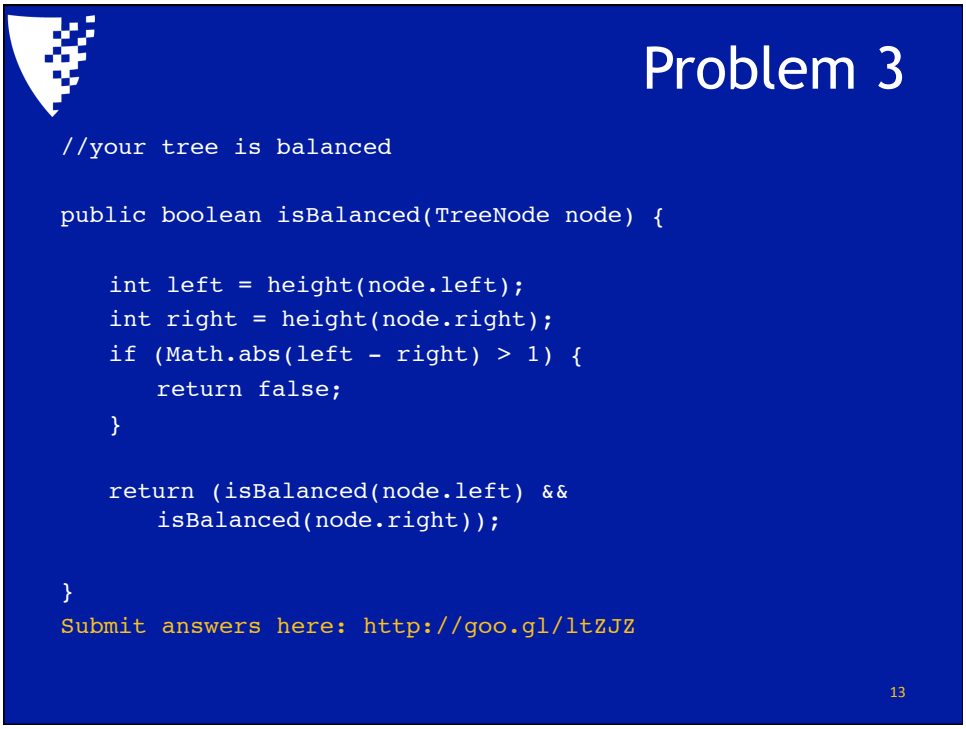

???
// Is the value v contained in the binary search tree rooted at node?
public boolean BSTcontainsValue(int v, TreeNode node) {
if (node == null) {
return false;
}
if (node.value == v) {
return true;
}
if (v < node.value) {
return BSTcontainsValue(v, node.left);
} else {
return BSTcontainsValue(v, node.right);
}
}
What if the tree is balanced?
What if the tree is unbalanced?
8

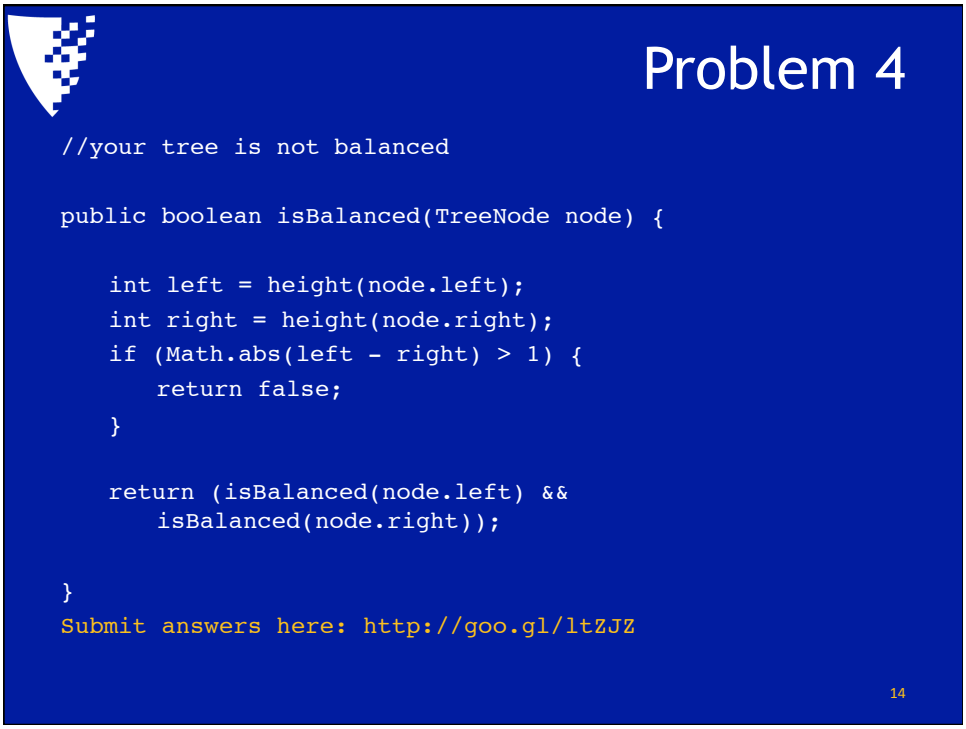

Recurrence Relations
How to calculate the Big-Oh of a recursive function
Write the recurrence relation
Solve the recurrence relation
Compute the Big-Oh
Use look-up table
9

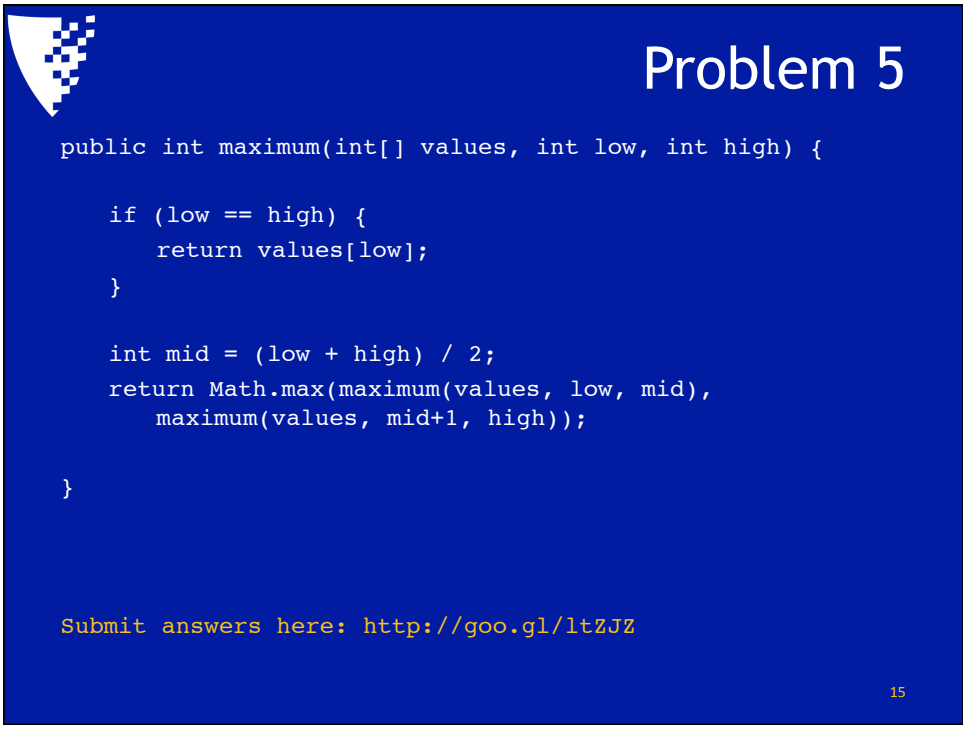

Recurrence Relations
Recurrence
Example
Running Time
T(n) = T(n/2) + O(1)
Binary Search
O(log n)
T(n) = T(n-1) + O(1)
Linear Search
O(n)
T(n) = 2T(n/2) + O(1)
Tree traversal
O(n)
T(n) = 2T(n/2) + O(n)
QuickSort
O(n log n)
T(n) = T(n-1) + O(n)
BubbleSort
O(n2)
10

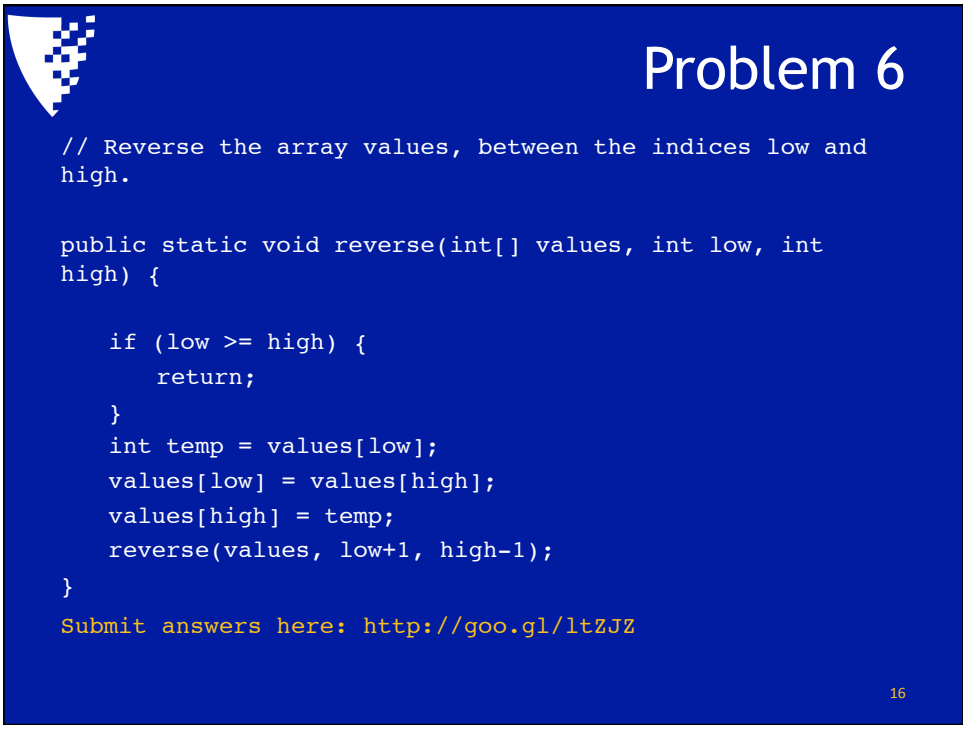

Problem 1
//your tree is balanced
public int height(TreeNode node) {
if (node == null) {
return 0;
}
int leftHeight = height(node.left);
int rightHeight = height(node.right);
return Math.max(leftHeight, rightHeight) + 1;
}
Submit answers here: http://goo.gl/ltZJZ
11

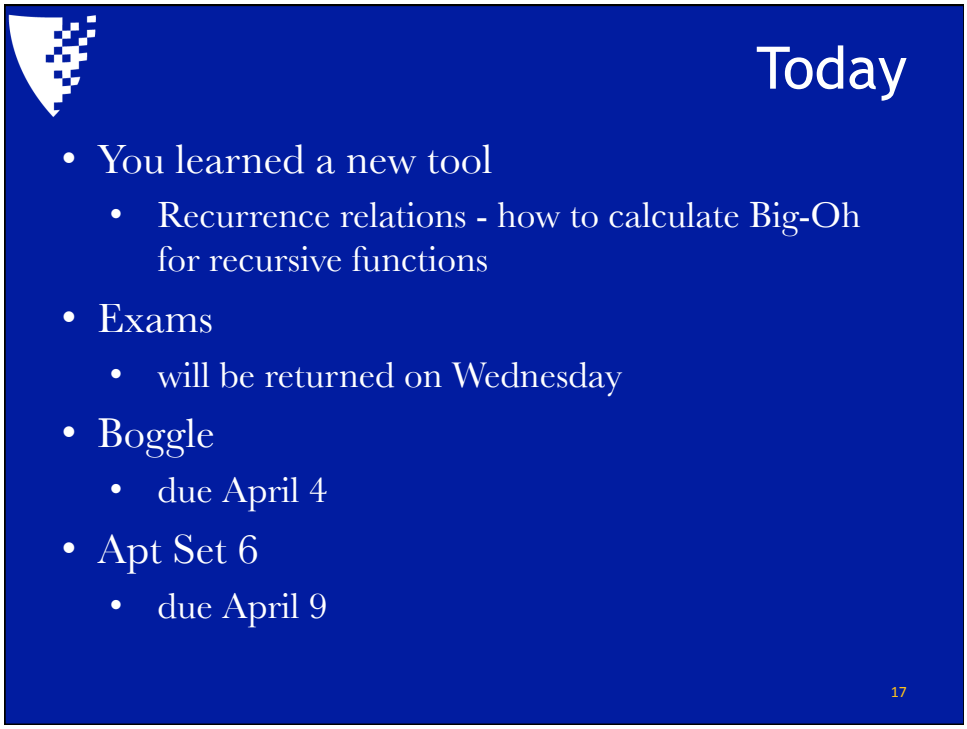

Problem 2
//your tree is not balanced
public int height(TreeNode node) {
if (node == null) {
return 0;
}
int leftHeight = height(node.left);
int rightHeight = height(node.right);
return Math.max(leftHeight, rightHeight) + 1;
}
Submit answers here: http://goo.gl/ltZJZ
12


Problem 3
//your tree is balanced
public boolean isBalanced(TreeNode node) {
int left = height(node.left);
int right = height(node.right);
if (Math.abs(left - right) > 1) {
return false;
}
return (isBalanced(node.left) &&
isBalanced(node.right));
}
Submit answers here: http://goo.gl/ltZJZ
13


Problem 4
//your tree is not balanced
public boolean isBalanced(TreeNode node) {
int left = height(node.left);
int right = height(node.right);
if (Math.abs(left - right) > 1) {
return false;
}
return (isBalanced(node.left) &&
isBalanced(node.right));
}
Submit answers here: http://goo.gl/ltZJZ
14


Problem 5
public int maximum(int[] values, int low, int high) {
if (low == high) {
return values[low];
}
int mid = (low + high) / 2;
return Math.max(maximum(values, low, mid),
maximum(values, mid+1, high));
}
Submit answers here: http://goo.gl/ltZJZ
15


Problem 6
// Reverse the array values, between the indices low and high.
public static void reverse(int[] values, int low, int high) {
if (low >= high) {
return;
}
int temp = values[low];
values[low] = values[high];
values[high] = temp;
reverse(values, low+1, high-1);
}
Submit answers here: http://goo.gl/ltZJZ
16


Today
• You learned a new tool
• Recurrence relations - how to calculate Big-Oh for recursive functions
• Exams
• will be returned on Wednesday
• Boggle
• due April 4
• Apt Set 6
• due April 9
17