# **CSc 130 Midterm Exam**

## **Spring 2021**

(Questions 1-18, 5 points each; Question 19, 30 points)

1. If f(n) = Ω(g(n), then
2. f(n) is less complex than g(n)
3. f(n) grows at least as fast as g(n)
4. The complexity of f(n) is no greater than that of g(n)
5. The complexity of f(n) is greater than that of g(n)
6. If f(n) = O(g(n), then
7. f(n) is more complex than g(n)
8. the growth rate of f(n) is no greater than that of g(n)
9. f(n) is the upper bound of g(n) in terms of complexity
10. g(n) is the lower bound of f(n) in terms of complexity
11. According to the big O notation, which of the following is false?
12. n1/2 = O(n/ log n)
13. (log n)1000 = O( n1/1000 )
14. (log n)n = O(2n)
15. 21,000,000 = O((log n)1/1000)
16. What is the maximum possible height of a binary search tree with 19 nodes?
17. 17
18. 18
19. 19
20. 20
21. Which statement is always true for a binary search tree of size n?
22. The value of the parent node is greater than that of its children
23. The height of the tree is n/2
24. For any node in the tree, the value of its left child is less than that of its right child
25. None of above
26. What is the minimum possible height of a binary search tree with 45 nodes?
27. 3
28. 4
29. 5
30. 6
31. The average case time complexity of inserting a node into a red-black tree of size n is
32. O(1)
33. O(log n)
34. O(n)
35. O(nlog n)
36. The worst case time complexity of inserting a node into a red-black tree of size n is
37. O(1)
38. O(log n)
39. O(n)
40. O(n log n)
41. The best case time complexity of inserting a node into a red-black tree of size n is
42. O(1)
43. O(log n)
44. O(n)
45. O(nlog n)
46. The time complexity of performing a double rotation in an AVL tree is
47. O(1)
48. O(log n)
49. O(n)
50. O(n2)
51. The time complexity of deleting a node from an AVL tree of size n is
52. O(1)
53. O(log n)
54. O(n)
55. O(n2)
56. The time complexity of performing a *preorder traversal* against a binary tree of size n is
57. O(1)
58. O(log n)
59. O(n)
60. O(n2)

Shape

Description automatically generated

1. The result of postorder traversal is
2. A, B, C, D, E, F, G, H, I
3. A, B, D, E, H, I, C, F, G
4. D, B, H, E, I, A, F, C, G
5. D, H, I, E, B, F, G, C, A
6. The result of preorder traversal is
7. A, B, C, D, E, F, G, H, I
8. A, B, D, E, H, I, C, F, G
9. D, B, H, E, I, A, F, C, G
10. D, H, I, E, B, F, G, C, A
11. The result of inorder traversal is
12. A, B, C, D, E, F, G, H, I
13. A, B, D, E, H, I, C, F, G
14. D, B, H, E, I, A, F, C, G
15. D, H, I, E, B, F, G, C, A
16. With an initially empty tree, the resulting red-black tree, after sequentially inserting 1, 2, 3, 4, 5, 6 looks like

A picture containing shape

Description automatically generated

Diagram

Description automatically generated

A picture containing diagram

Description automatically generated

* 1. None of above

Diagram

Description automatically generated

1. After placing 14 just according to BST property, which of the following is true?
2. It is still an AVL tree
3. It is not an AVL tree, and we need to rotate node 6 to convert it back to an AVL tree
4. It is not an AVL tree, and we need to rotate node 4 to convert it back to an AVL tree
5. It is not an AVL tree, and we need to rotate both nodes 4 and 6 to convert it back to an AVL tree
6. The resulting AVL tree after inserting 14 looks like

Diagram

Description automatically generated

Shape

Description automatically generated

Diagram

Description automatically generated

Logo

Description automatically generated

1. AVL Tree Programming (30 points)
   1. There is a ‘height’ data field in the AvlNode class definition. To get the height of an AVL node, which of the following two methods should you use? Why? (5 points)

// Method 1

private static int height(AvlNode t) {

if (t == null) return -1;

return t.height;

}

// Method 2

private static int height(AvlNode t) {

if (t == null) return -1;

return 1+ max(height(t.left), height(t.right));

}

* 1. Identify the errors in the code below for the single and double rotation to the right; then write the code accordingly for the single and double rotation to the left. (10 points)

private static AvlNode rotateToRight(AvlNode k2) {

AvlNode k1 = k2.left;

k2.left = k1.right;

k1.right = k2;

k1.height = max(height(k1.left), height(k2.height)) + 1;

k2.height = max(height(k2.left), height(k2.right)) + 1;

return k2;

}

private static AvlNode doubleRotateToRight(AvlNode k3) {

k3.left = rotateToRight(k3.left);

k3 = rotateToRight(k3)

k3.height = max(height(k3.left), height(k3.right)) + 1;

return k3;

}

* 1. Identify the errors and/or missing code in the “insert” and “remove” methods for the AvlNode class; then rewrite the code for the two methods. ( 15 points)

private AvlNode insert(AvlNode t, int x) {

if (t == null) return new AvlNode(x, null, null, 0);

if (x < t.val) {

t.left = insert(t.left, x);

if (height(t.left) – height (t.right) > 1)

if (x > t.left.val)

t = doubleRotateToRight(t.left);

else

t = rotateToRight(t.left);

} else if (x > t.val) {

t.right = insert(t.right), x);

if (height(t.right) – height(t.left) == 2)

if (x > t.right.val)

t.right = doubleRotateToLeft(t);

else

t.right = rotateToLeft(t);

} else ; // duplicate; do nothing

Return t;

}

private AvlNode remove(AvlNode t, int x) {

if (t == null) return null;

if (x < t.val) {

t.left = remove(t.left, x);

if (height(t.right) - height(t.left) == 2) {

if (height(t.right.right) < height(t.right.left)

t = rotateToLeft(t.right);

else

t = doubleRotateToLeft(t.right);

}

} else if (x > t.val) {

t.right = remove(t.right, x);

if (height(t.left) - height(t.right) == 2) {

if (height(t.left.left) >= height(t.left.right)

t.left = rotateToRight(t);

else

t.left = doubleRotateToRight(t);

}

} else if (t.left != null && t.right != null) {

t.val = findMin(t.right).val;

t.right = remove(t.right, t.val);

} else {

If (t.left != null)

t = t.left;

else

t = t.right;

}

return t;

}