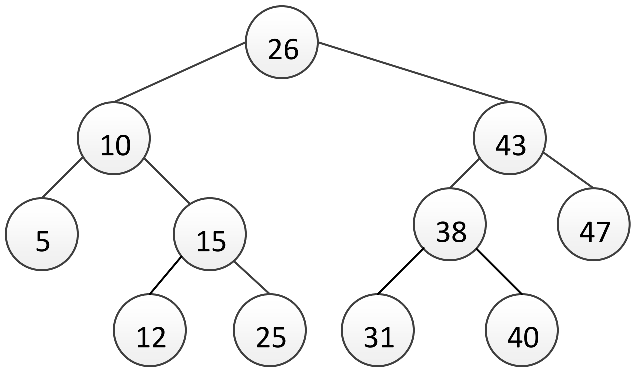
CS526 O2
Homework Assignment 5

Problem 1 (10 points). Consider the following binary search tree:

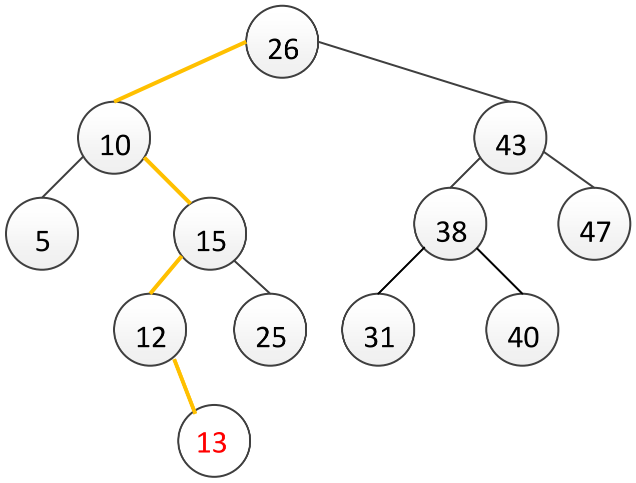


Show the resulting tree if you add the entry with key = 13 to the above tree. You need to describe, step by step, how the resulting tree is generated.

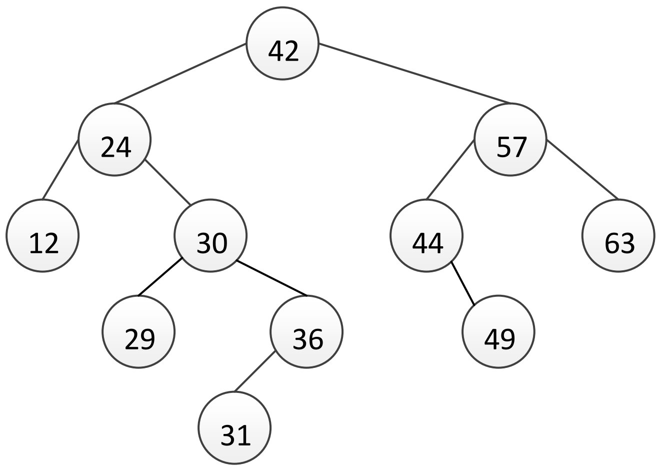
Answer:

1. Perform a search operation to find the entry with key = 13.
2. The entry with key = 13 is not found until it reaches the leaf node of 12 (search route highlighted below)
3. Add this entry at the leaf node at 12's right subtree where the unsuccessful search ends up.

Resulting tree:



Problem 2 (10 points). Consider the following binary search tree:

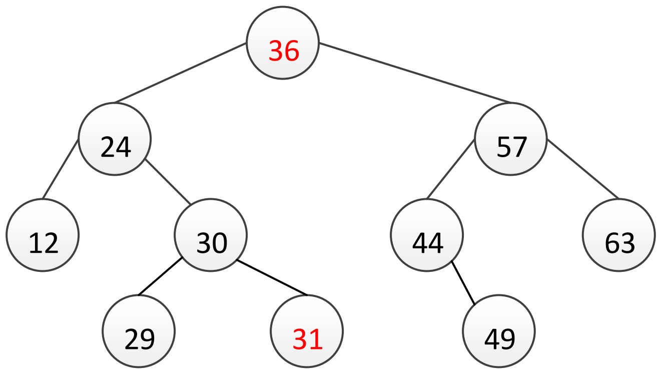


Show the resulting tree if you delete the entry with key = 42 from the above tree. You need to describe, step by step, how the resulting tree is generated.

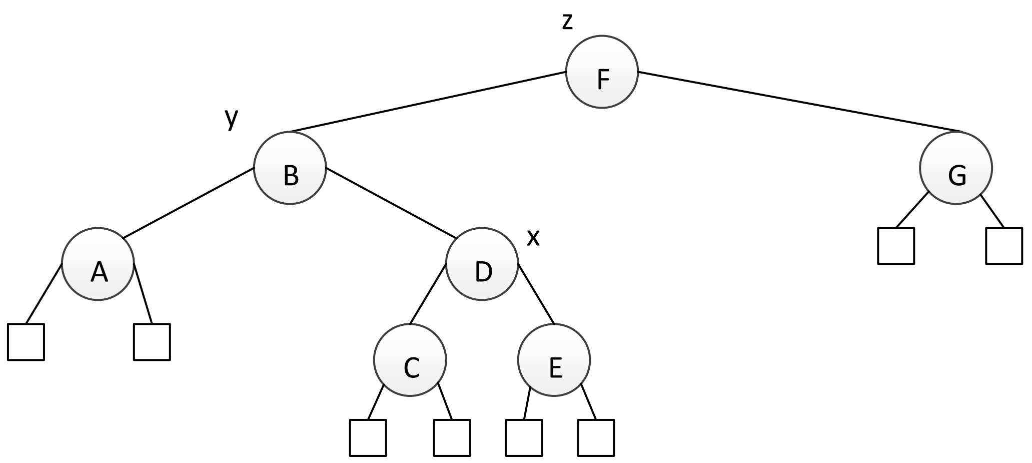
Answer:

1. Perform a search operation to find the entry with key = 42.
2. The entry with key = 42 is found and it has two children, both of which are internal.
3. Find predecessor node of 42 which is 36.
4. Replace 42 with 36.
5. 31 as previous left child of 36 is promoted to the previous position of 36.

Resulting tree:

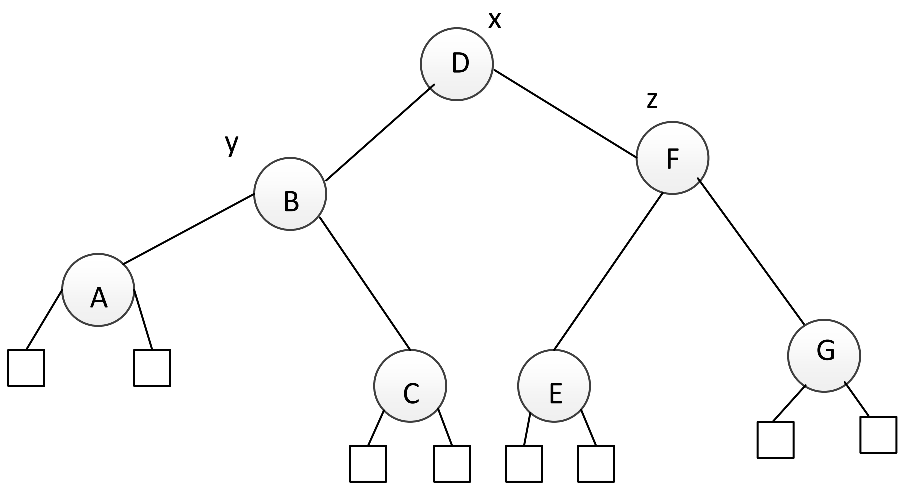


Problem 3 (10 points). Consider the following AVL tree, which is unbalanced:

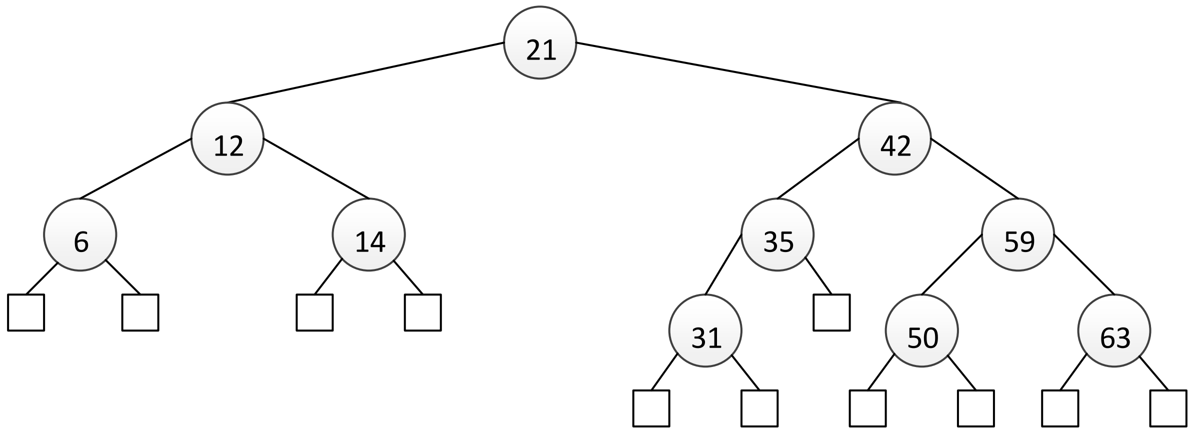


Note that the nodes *F*, *B* and *D* are labeled *z*, *y*, and *x*, respectively, following the notational convention used in the textbook. Apply a trinode restructuring on the tree and show the resulting, balanced tree.

Answer:



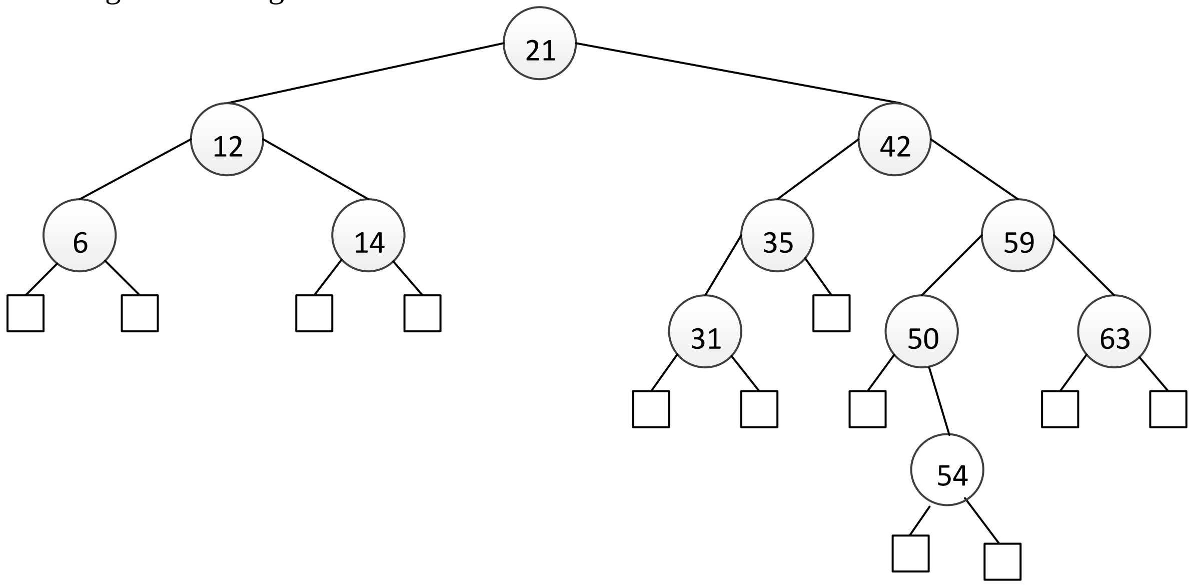
Problem 4 (10 points). Consider the following AVL tree.



Show the resulting, balanced tree after inserting an entry with key = 54 to the above tree. You must describe, step by step, how the resulting tree is obtained.

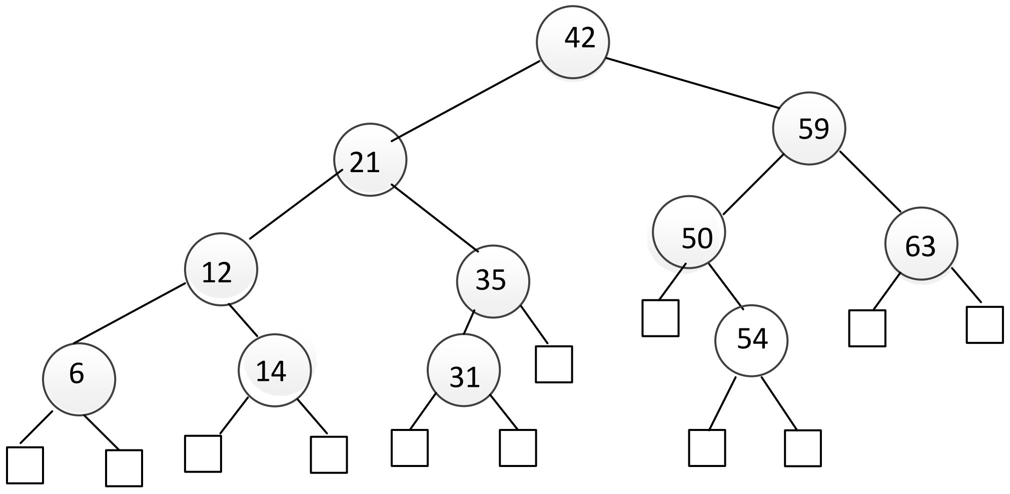
Answer:

1. Inserting 54 to the right child node of 50.

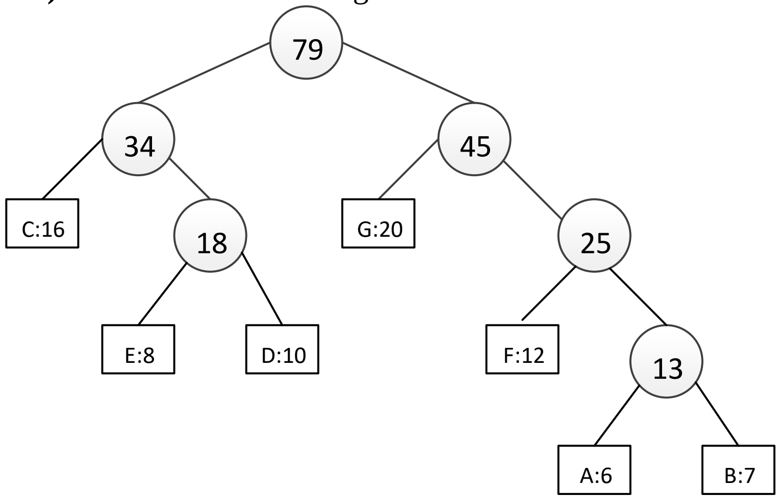


2. This tree is unbalanced as heights of 21’s children differ by 2 (42 has height of 4, 12 has height of 2). So this tree needs to be rebalanced using post-processing.
3. Perform search-and-repair strategy: search a node that is the lowest ancestor of 54 that is unbalanced, which is 21 (z).
4. Identify 21’s child with the greater height, which is 42 (y).
5. Identify 59’s child with the greater height, which is 59 (x).
6. Perform double rotation to rebalance the tree.

Resulting tree:



Problem 5 (10 points). Consider the following Huffman tree:



(1) Encode the string “BDEGC” to a bit pattern using the Huffman tree.

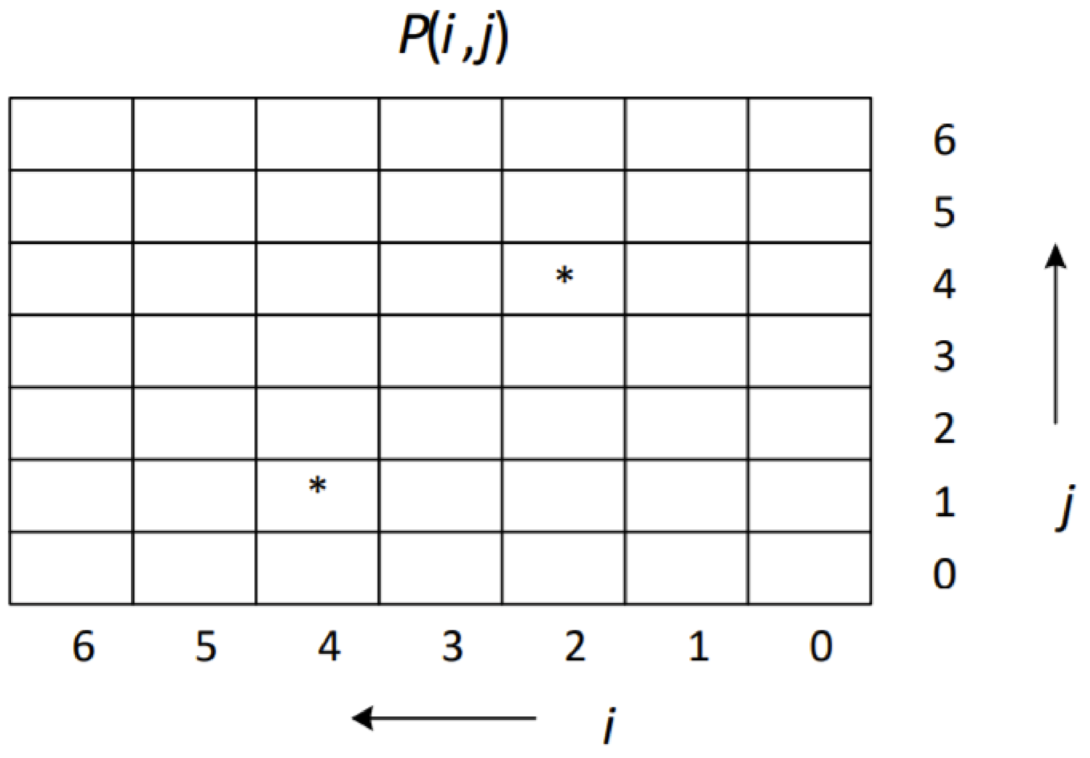
Answer:

Because B is 1111, D is 011, E is 010, G is 10, C is 00.

The final result is 11110110101000.

(2) Decode the bit pattern “010111010011” to the original string using the Huffman tree.
Answer:
Because 010 is E, 1110 is A, 10 is G, 011 is D.
The final result is EAGD.

Problem 6 (10 points). This question is about the *World Series* problem that we discussed in the class. The following is the probability matrix for the problem.



Calculate the probabilities of *P*(4, 1) and *P*(2, 4), which are marked with *. You must show all intermediate steps and calculations.

Answer:
P(1, 1) = (P(0, 1) + P(1, 0)) / 2 = (1 + 0) / 2 = ½
P(2, 1) = (P(1, 1) + P(2, 0)) / 2 = (1/2 + 0) / 2 = ¼
P(3, 1) = (P(2, 1) + P(3, 0)) / 2 = (1/4 + 0) / 2 = 1/8
P(4, 1) = (P(3, 1) + P(4, 0)) / 2 = (1/8 + 0) / 2 = 1/16

P(1, 2) = (P(1, 1) + P(0, 2)) / 2 = (1 + ½) / 2 = 3/4
P(2, 2) = (P(1, 2) + P(2, 1)) / 2 = (3/4 + ¼) / 2 = ½
P(1, 3) = (P(0, 3) + P(1, 2)) / 2 = (1 + ¾) / 2 = 7/8
P(2, 3) = (P(1, 3) + P(2, 2)) / 2 = (7/8 + ½) / 2 = 11/16
P(1, 4) = (P(0, 4) + P(1, 3)) / 2 = (1 + 7/8) / 2 = 15/16
P(2, 4) = (P(1, 4) + P(2, 3)) / 2 = (15/16 + 11/16) / 2 = 13/16

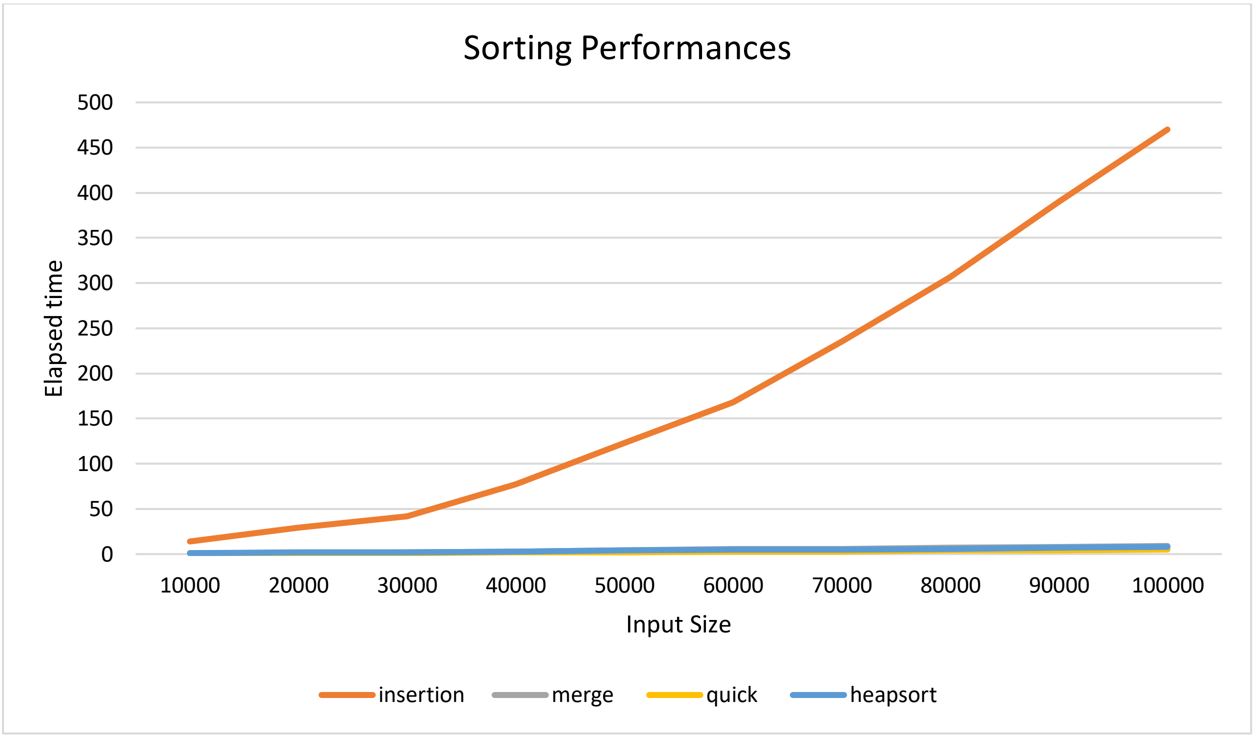
Problem 7 (40 points).

Result table:

n / *Algorithm*	10000	20000	30000	40000	50000	60000	70000	80000	90000	100000
insertion	14	29	42	77	123	168	235	307	390	470
merge	1	1	2	3	4	6	6	7	8	9
quick	1	1	1	2	2	3	3	4	4	5
heapsort	1	2	2	3	4	5	5	6	7	8

Entries in the table are elapsed times in milliseconds.

Result chart:



You must provide a 1-2 paragraph conclusion including the observations you collected, analysis of the results, and discussion of what you learned.

conclusion/observation/discussion:

1. Insertion sort performance gets worse with larger input sizes, whereas other 3 sorting algorithms remain relatively quicker regardless of input sizes. This is expected as insertion sort takes O(n^2) but the other 3 take O(nlogn).

2. Quick sort is the quickest among the quicker group of 3 sorting algorithms. This is also expected because:
    a. experience shows heap sort performance on large sequences is poorer than quick sort and merge sort, also heap sort is not a stable sorting algorithm.
    b. experimental studies showed quick sort outperformed heap sort and merge sort. The key advantage of quick sort is that it has a smaller constant factor than merge sort and heap sort,
    c. in contrast, merge sort and heap sort have larger constant factors due to the additional memory allocation required for their auxiliary data structures. Merge sort uses extra memory for the merge step, while heap sort requires an additional heap data structure.