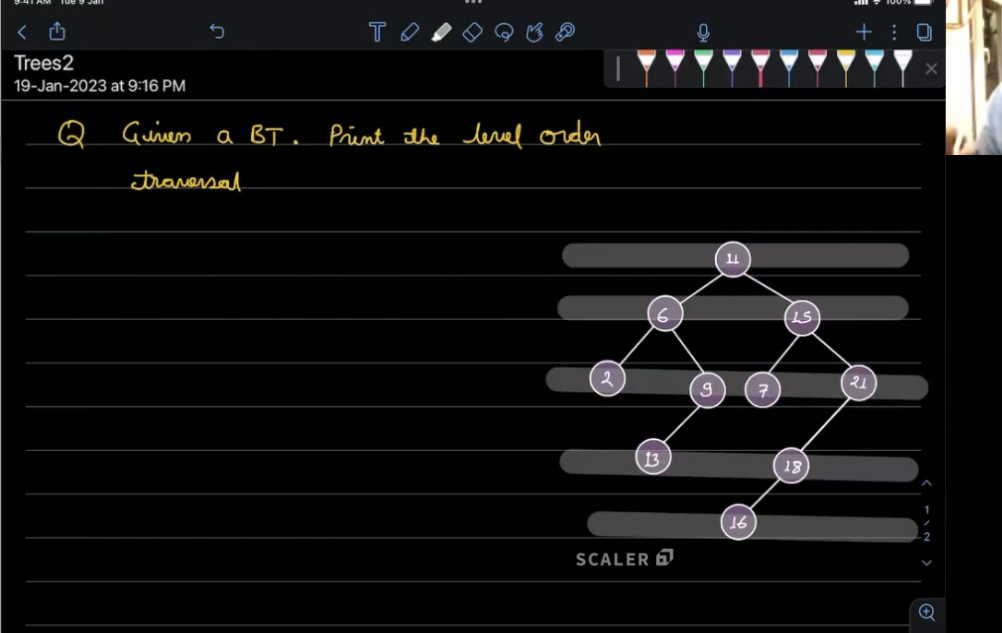
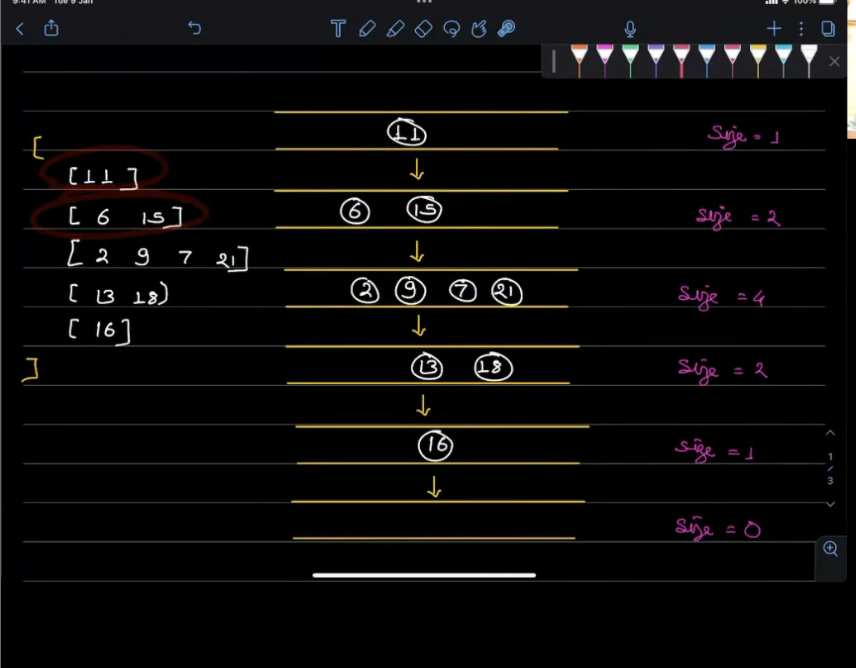
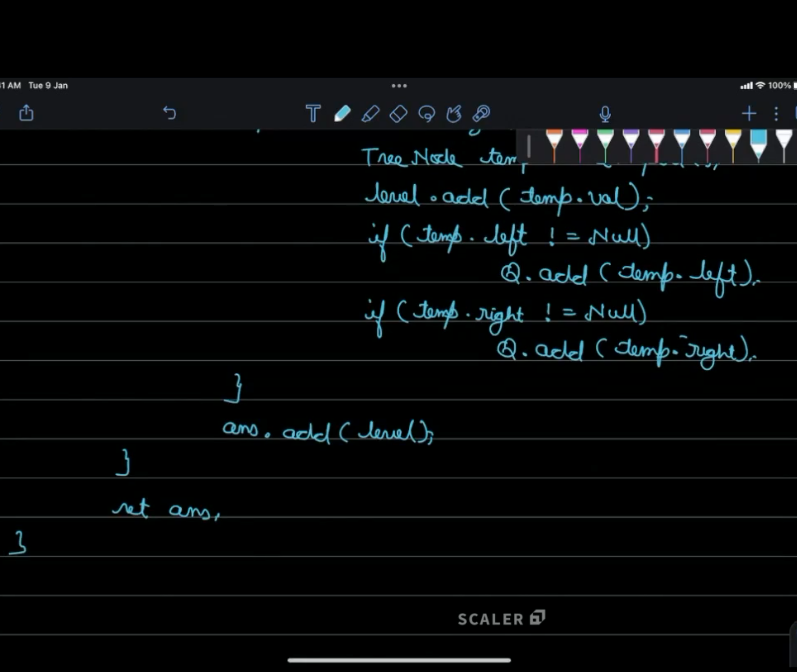
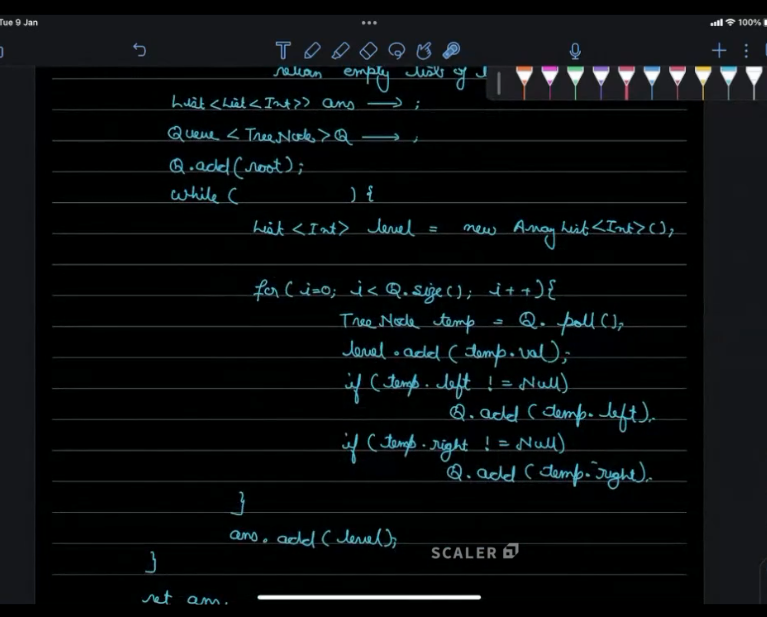
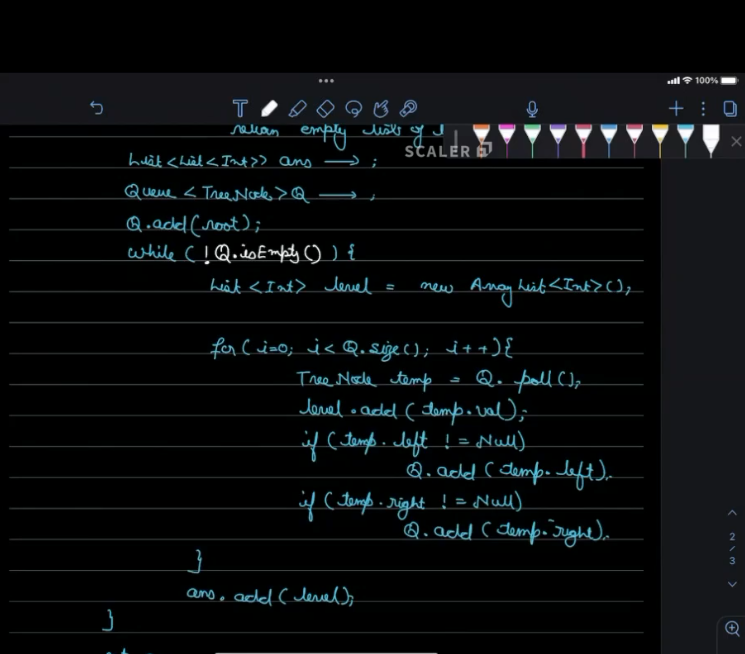
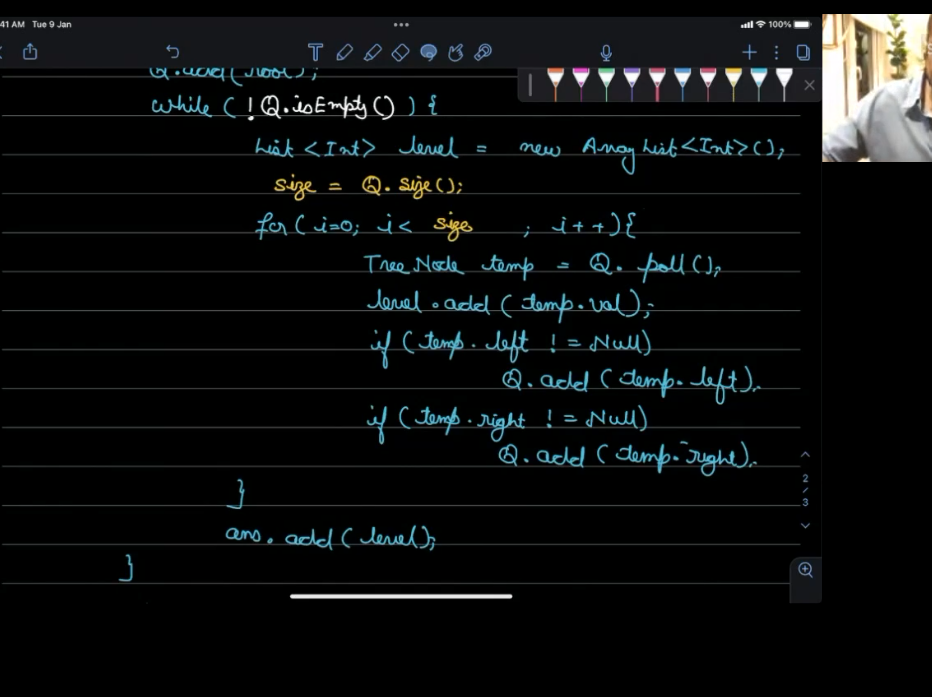
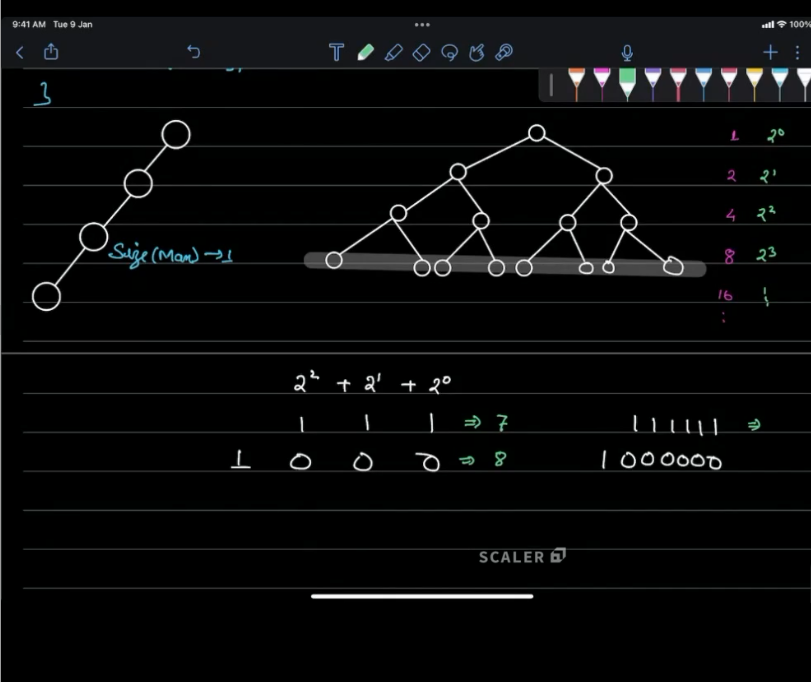
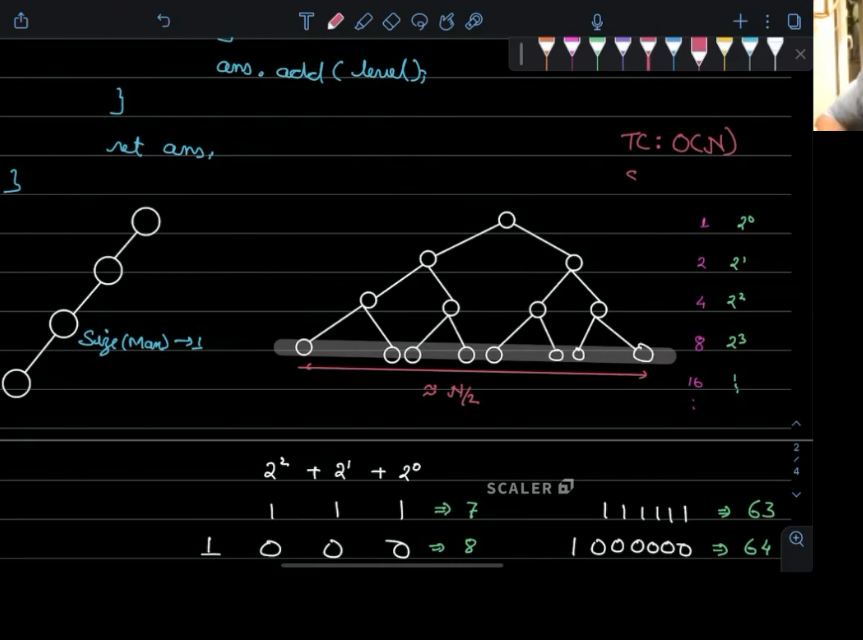
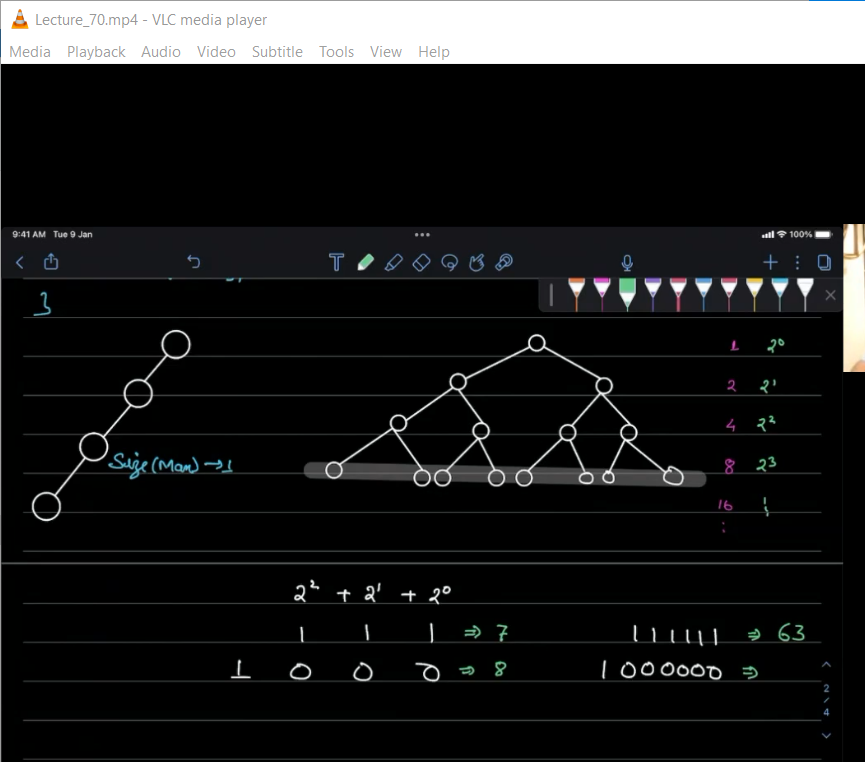
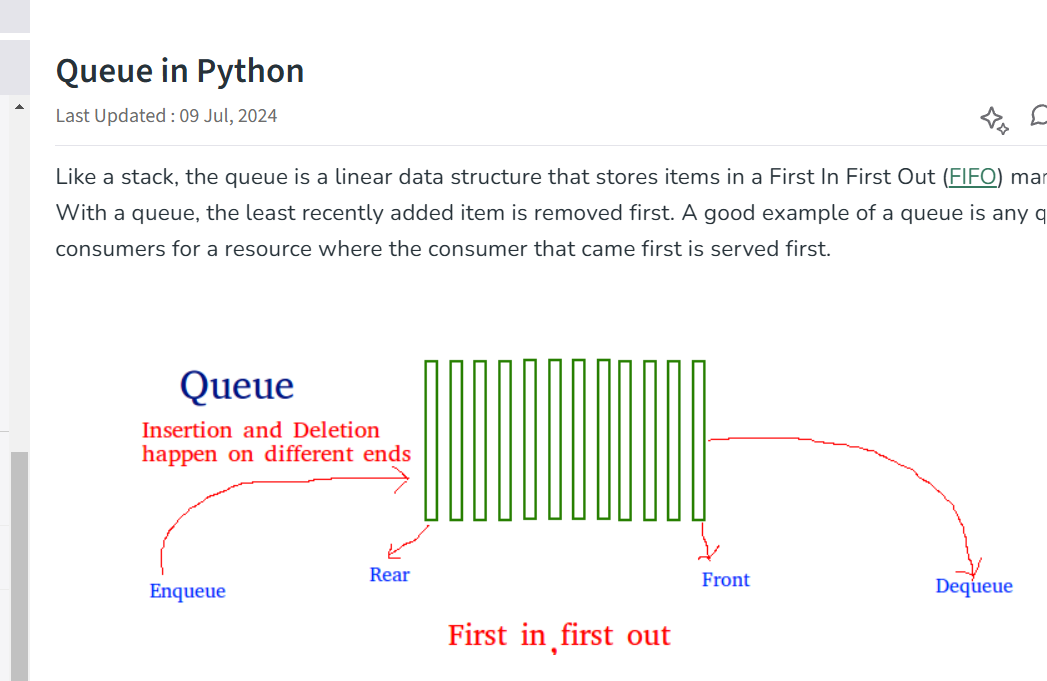
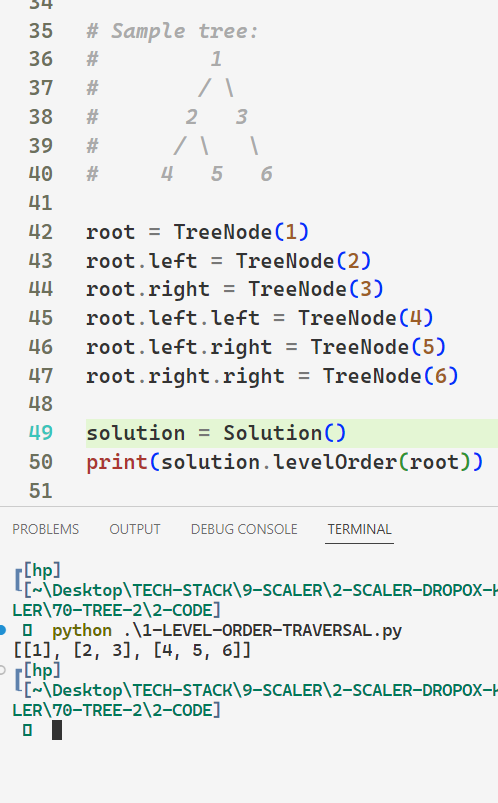
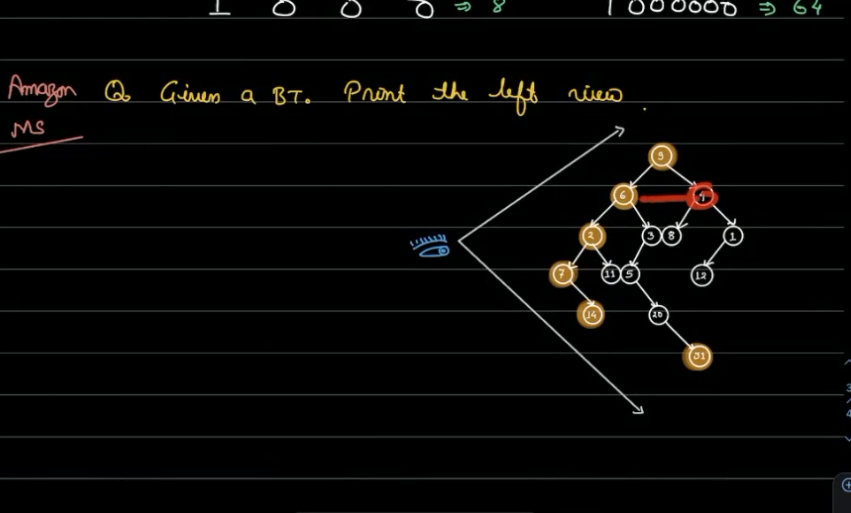
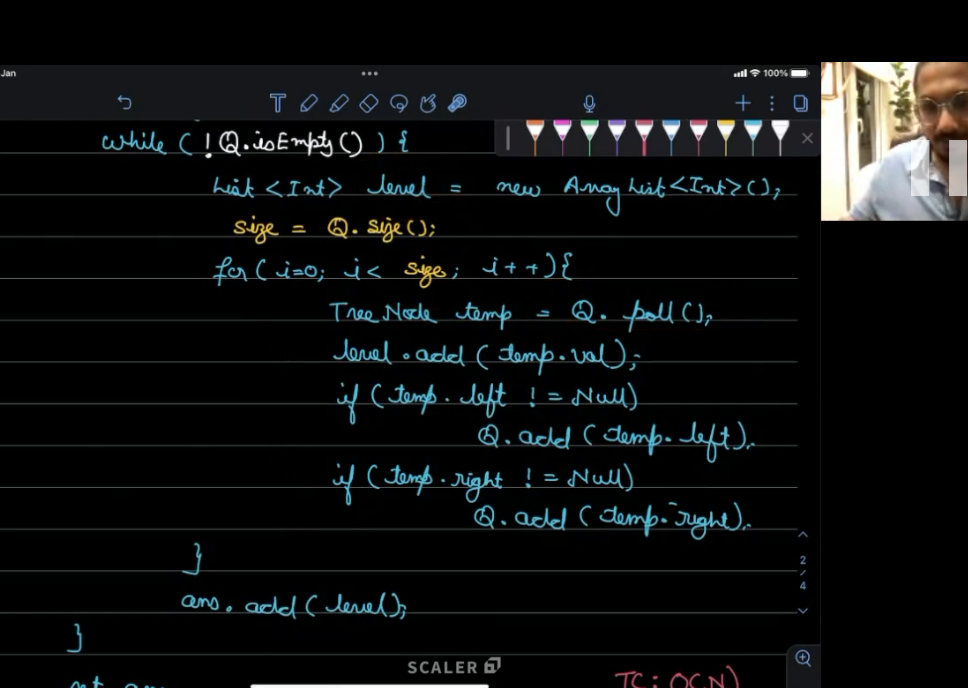
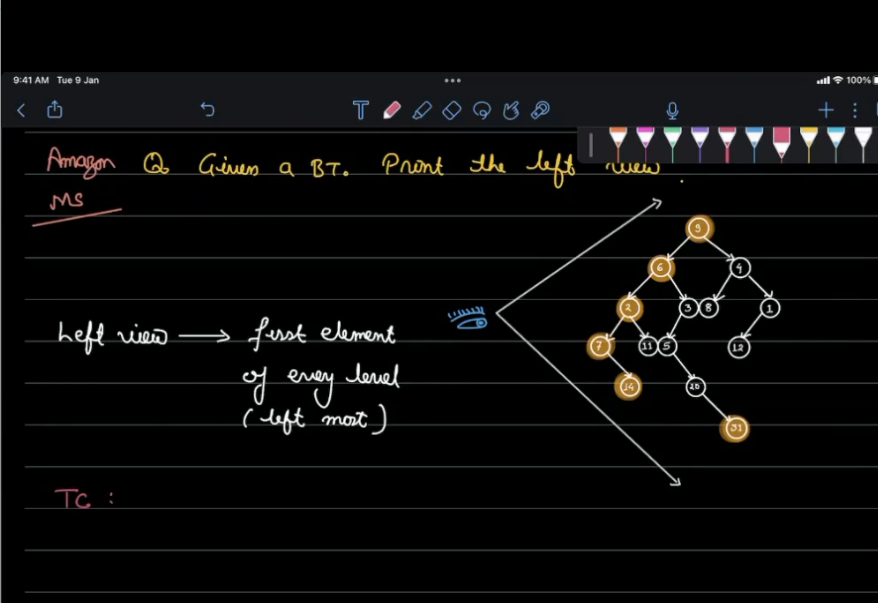
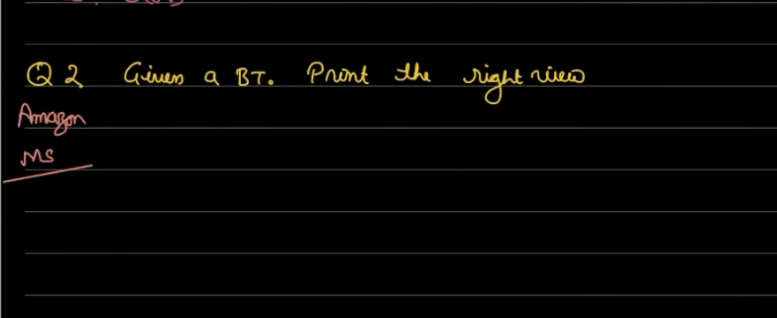
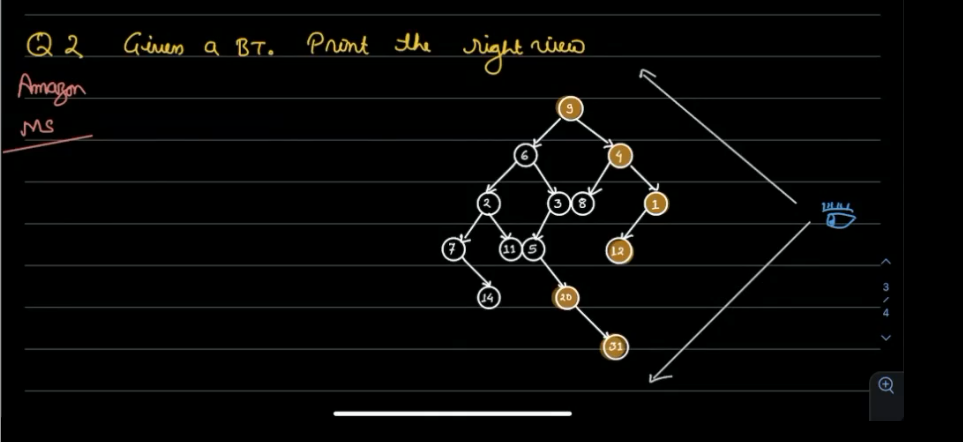
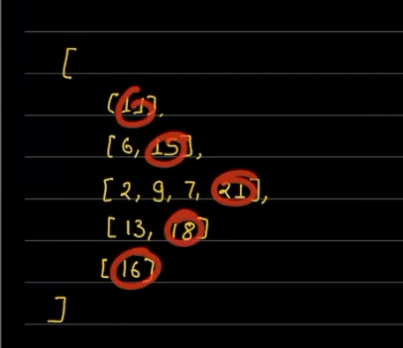
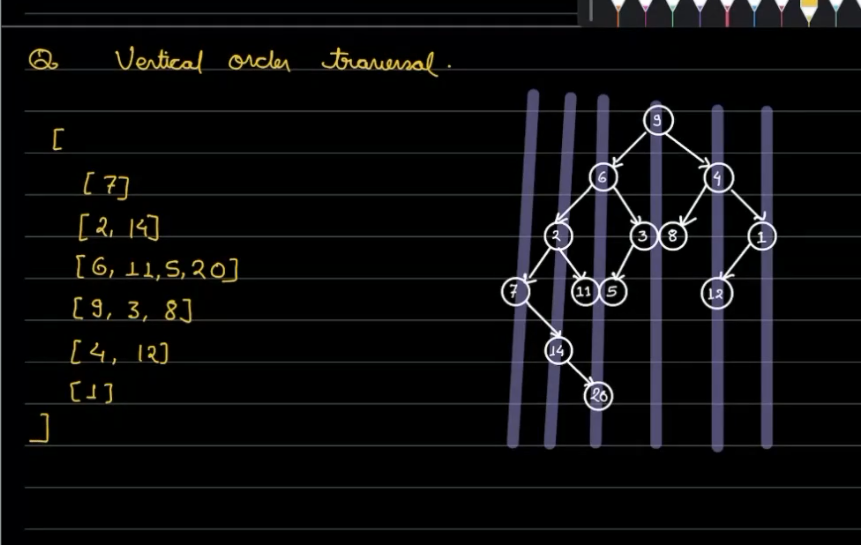
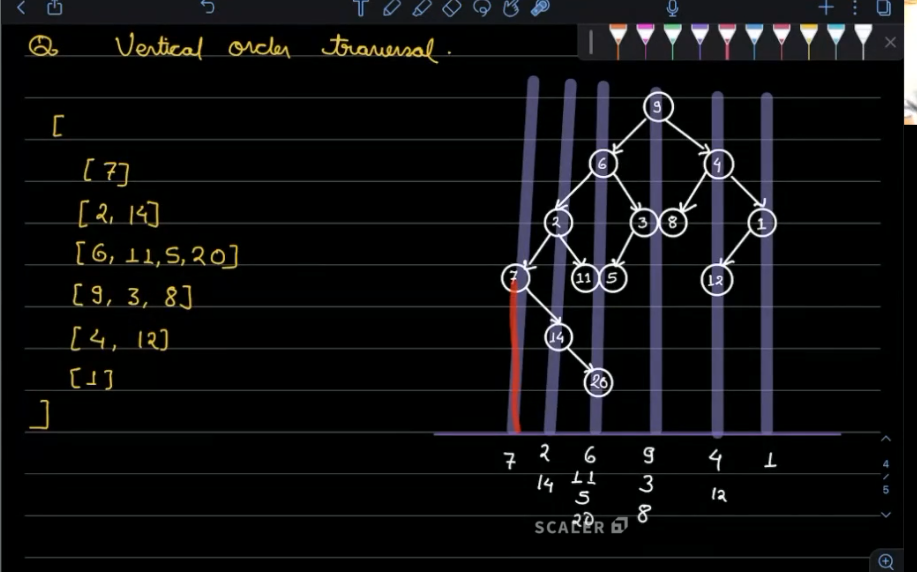
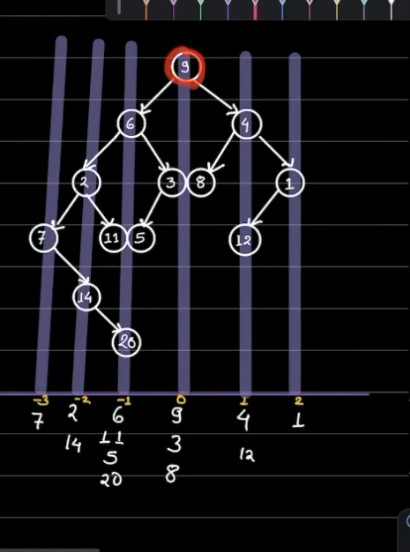
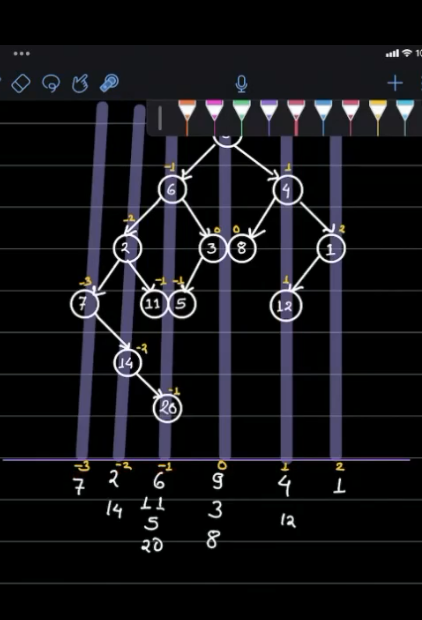
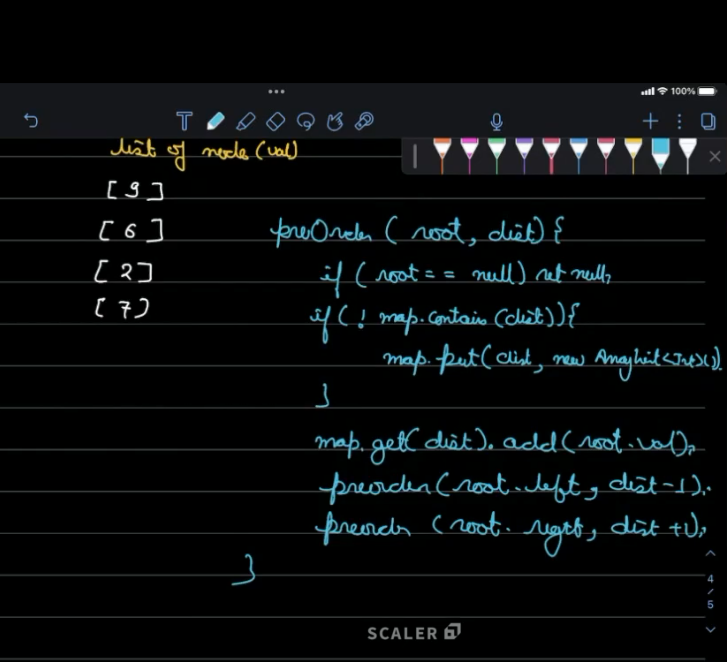
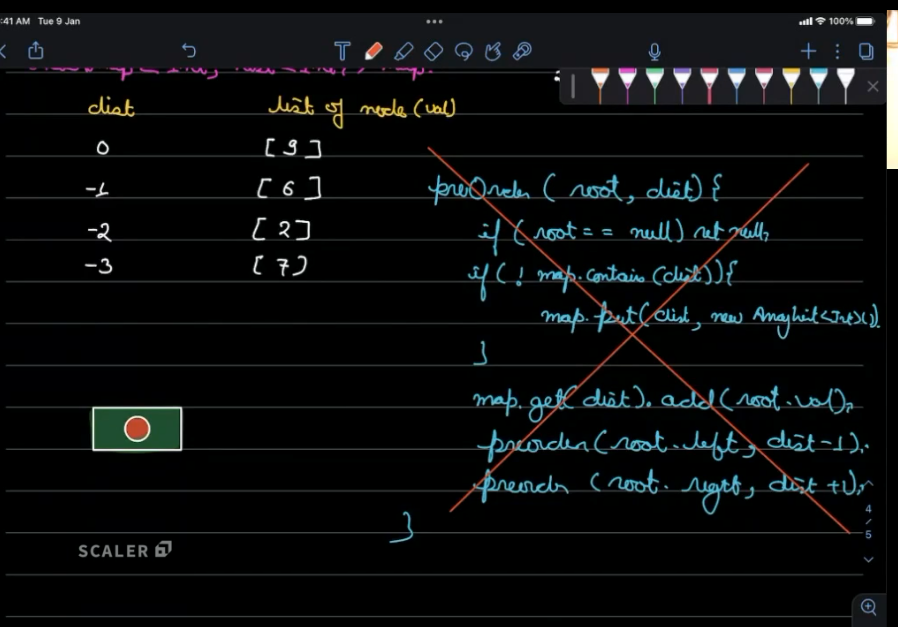
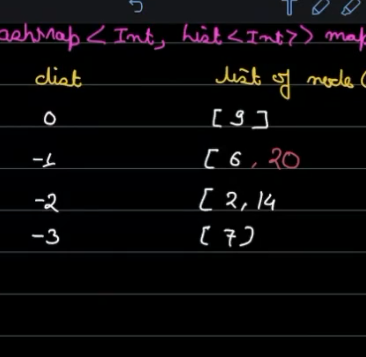
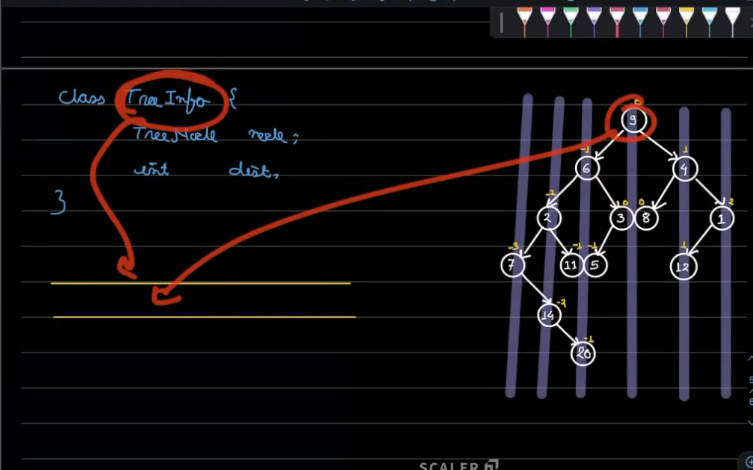
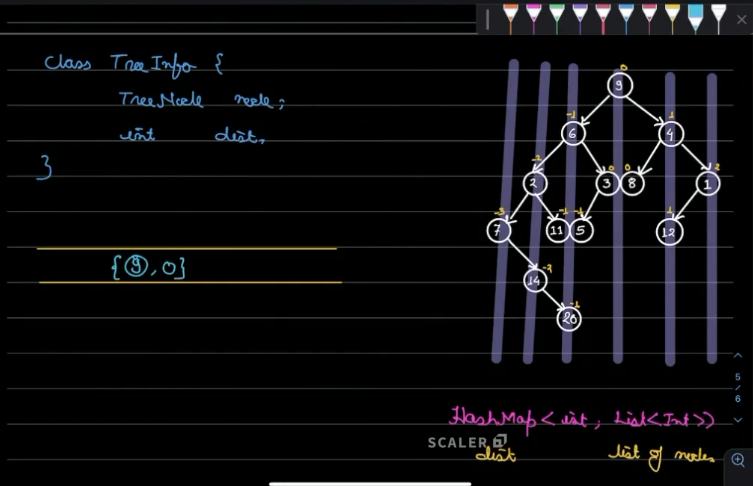
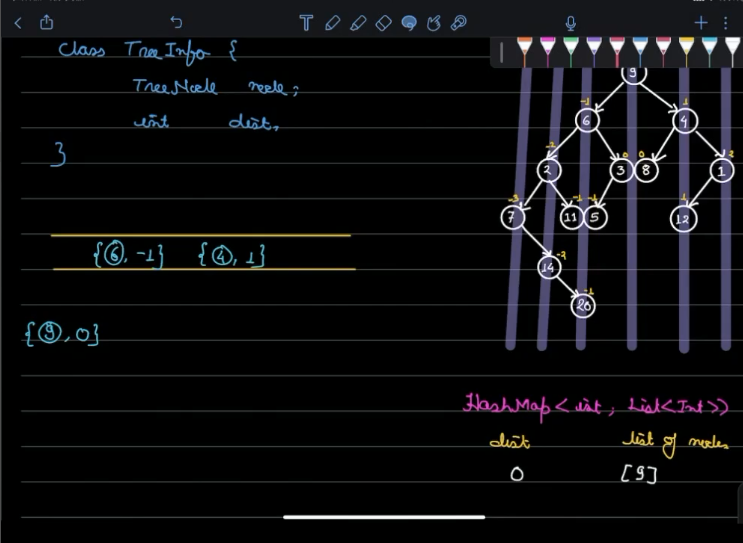
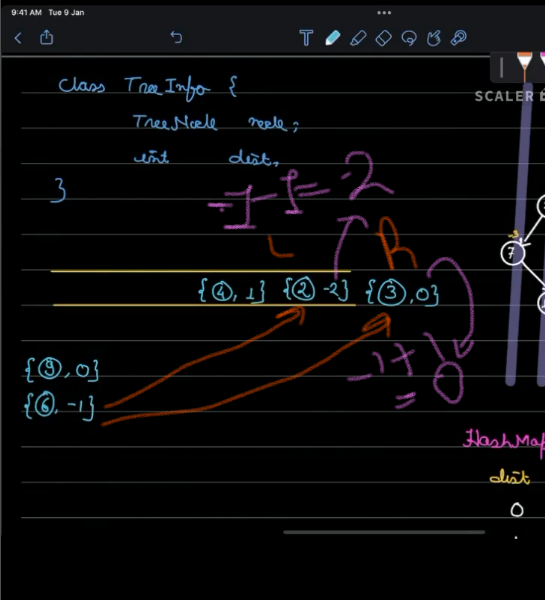
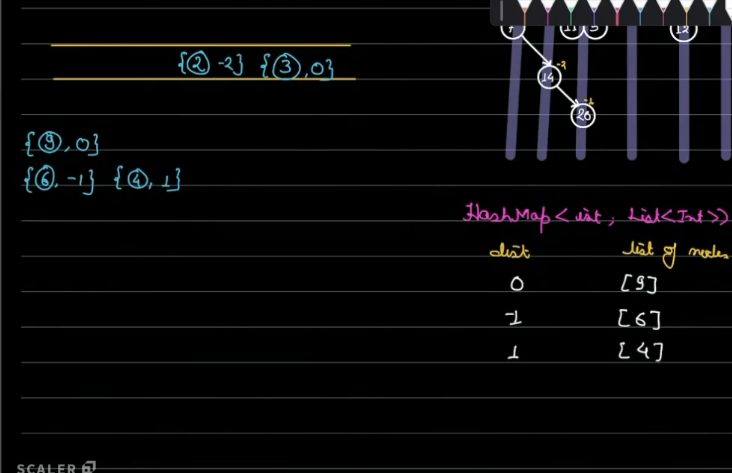
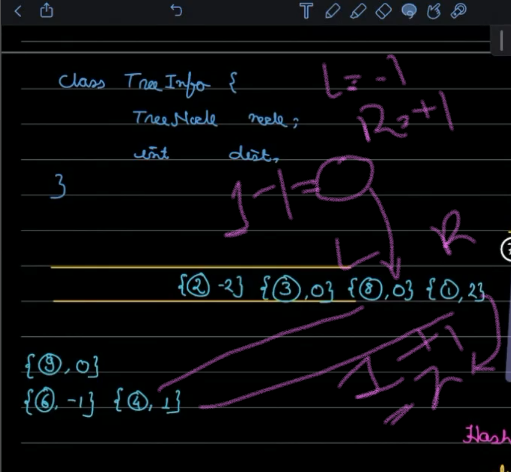
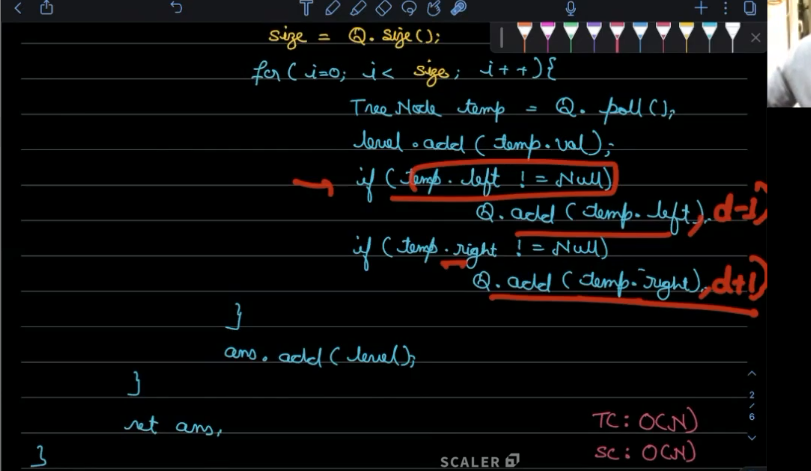
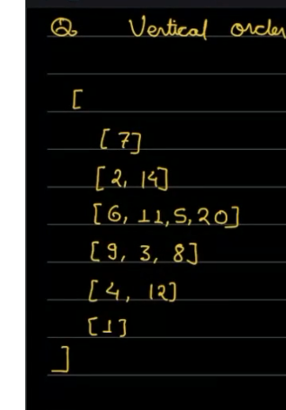
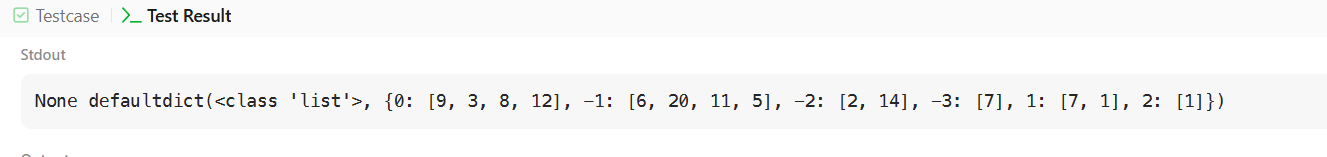
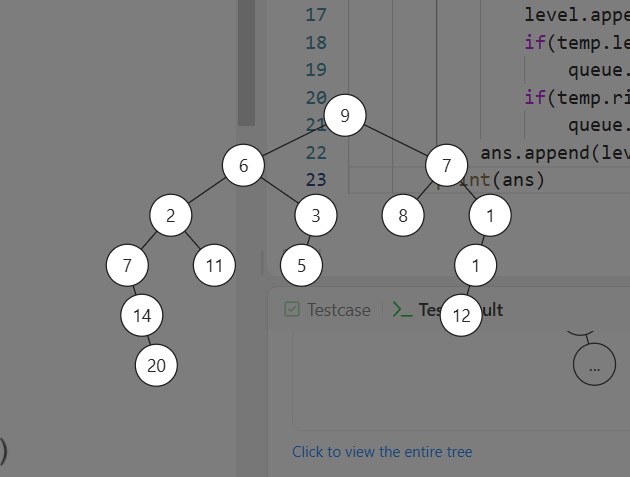
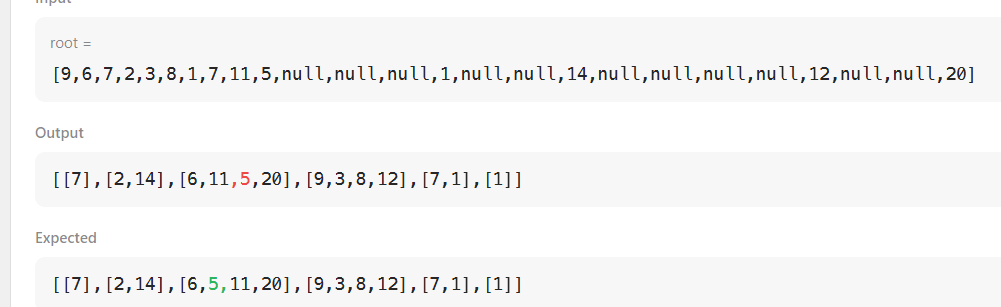
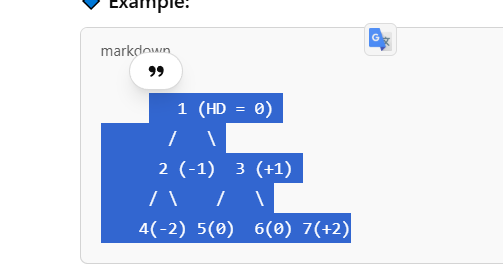
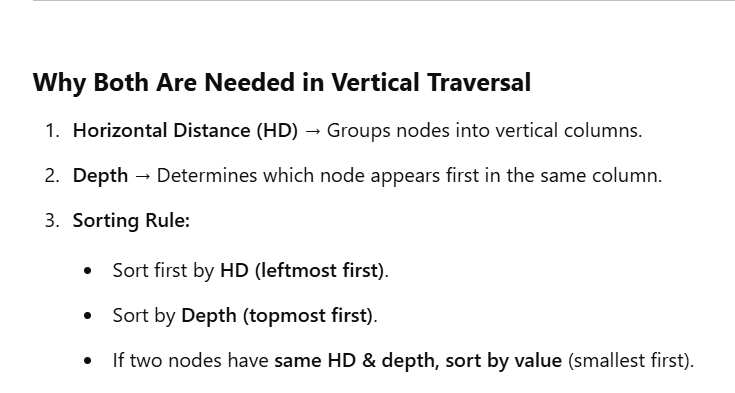
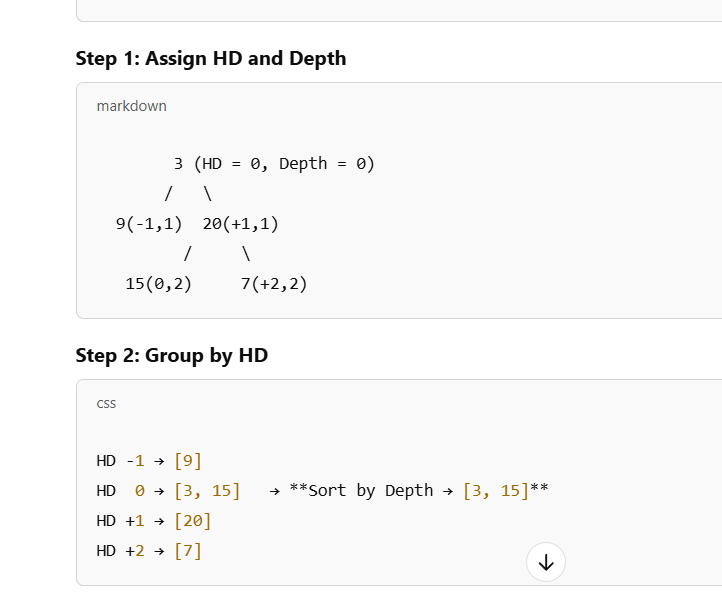
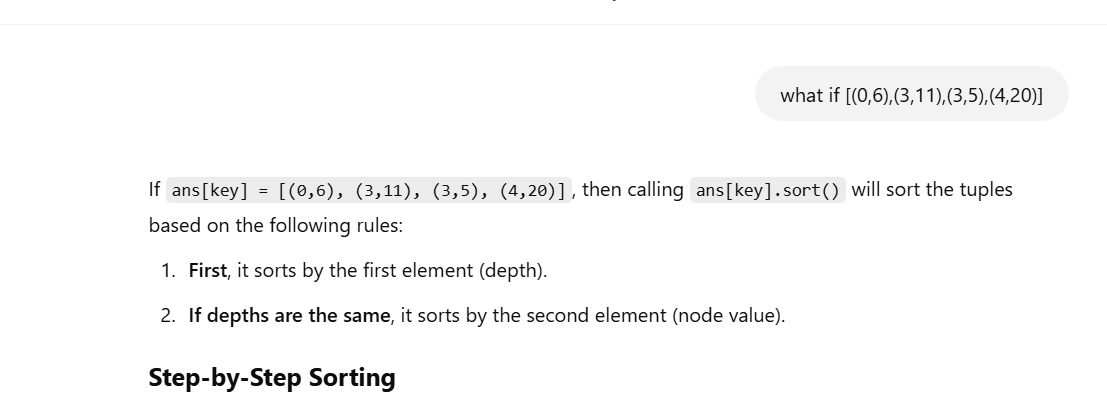
1. ****
2. [**https://chatgpt.com/c/67d1697c-a810-800d-a7cd-73b6f358aa3b**](https://chatgpt.com/c/67d1697c-a810-800d-a7cd-73b6f358aa3b)
3. [**https://chatgpt.com/c/67d1cc2e-85e4-800d-b360-1a11add4ab73**](https://chatgpt.com/c/67d1cc2e-85e4-800d-b360-1a11add4ab73)
4. **In brave**
5. **IN BRAVE**
6. **Queue=[11]**
7. **Queue=[15,6]**
8. **Print([6,15]) pop from front which is last not the rear which is wehere is enter the eleemtn**
9. **Queue.pop(0)🡺6 first 6 first come first serve**
10. **Queue.pop(0)🡺15**
11. **Queue.appen(6.left,6.right)**
12. **Queue=[9,2]**
13. **Queue.append(15.left,15.right)**
14. **Queue=[21,7,9,2]**
15. **Print(2,9,7,21)**
16. **Because pop from the last position first come first serve which is from the front position not the rear**
17. ****
18. ****
19. **4 pop**
20. ****
21. ****
22. **NUMBER OF POPS IS EQUAL TO NUMBER OF CHILDREN**
23. **USING THE QUEUE**
24. **FIRST IN FIRST OUT**
25. ****
26. ****
27. ****
28. **The FOR LOOP Q.SIZE KEE[ CHANGING DYNAMICALLY SO AS THE ACTUAL RUNNING WE DON’T EXPECT THAT CODE RUN WELL**
29. ****
30. **SIZE IS GOING TO KEEP CHANGING**
31. **QUEUE IS SKEWED**
32. **HOW MANY TIME AT MAX VISITED A NODE**
33. **TOUCHING 2 TIME, TOIUCH AND REMOVING**
34. **O(N)**
35. **SC🡺SKEWED TREE IS THE BEST TREE FOR SC AS ONE NODE IN ONE LEVEL**
36. ****
37. ****
38. **O(N/2) AS THE LAST NODE LAYER WHICH IS HIGH LIGHTED SO O(N)**
39. ****
40. ****
41. **2-GIVEN A BT PRINT ALL THE ELFT VIEW**
42. [**https://leetcode.com/problems/binary-tree-right-side-view/description/**](https://leetcode.com/problems/binary-tree-right-side-view/description/)
43. ****
44. **IF WE SEE FROM LEFT**
45. **Print the first node of evert level**
46. ****
47. **Only the FIRST NODE WE NEED**
48. ****
49. ****
50. **WILL BECOME THE LEFT VIEW**
51. ****
52. **RIGHT VIEW**
53. ****
54. ****
55. **EACH VERTICAL LINE NODE HAS EQUAL DISTANCE FROM THE NODE**
56. ****
57. ****
58. **TAKING HASHMAP WITH THE X AXIS**
59. **DISTANCE OF NODES AS KEY**
60. ****
61. ****
62. **FIRST RESOLVE WHOLE ROOT’S LEFT SUBTREE AND THEN RIGHT SUBTREE**
63. **RELATIVELY ORDERING IS WRONG**
64. **PUT THE ROOT NODE AND MAKE THE LEFT SUBTREE CHILD CALL**
65. **20 COMES BEFORE 11 AND 5 WORNG**
66. ****
67. ****
68. **We are not going to insert the node in the queue but The TREEINFO CLASS**
69. ****
70. ****
71. ****
72. ****
73. ****
74. ****
75. ****
76. **{0: [9, 3, 8, 12], -1: [6, 20, 11, 5], -2: [2, 14], -3: [7], 1: [7, 1], 2: [1]})**
77. **ORIGINAL 🡺6,11**
78. **MINE WITH THIS CODE🡺 6,20**
79. ****
80. ****
81. **[9,6,7,2,3,8,1,7,11,5,null,null,null,1,null,null,14,null,null,null,null,12,null,null,20]**
82. ****
83. ****
84. ****
85. **HORIZONTAL DISTANCE**
86. ****
87. ****
88. [**https://chatgpt.com/c/67d1697c-a810-800d-a7cd-73b6f358aa3b**](https://chatgpt.com/c/67d1697c-a810-800d-a7cd-73b6f358aa3b)
89. **5 AND 11 HAVE BOTH DEPTH 2**
90. **AND HAVE DISTANCE -1 SO 5<11 WE PUT 5 FIRST THEN 11**
91. **The distance of a node from the root along the vertical axis (tree height).**
92. **IF TWP NODES HAVE SAME DEPTH AND DISTANCE WE SORT THEM WITH THE VALUE SO 5 < 11**
93. **5 COMES FIRST**
94. ****