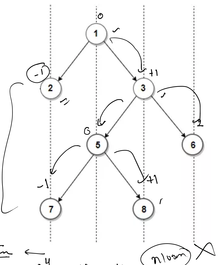
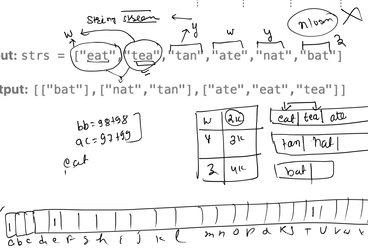
Lec-46

28/08/25 10:22 AM

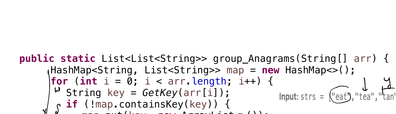
Tree - Top view



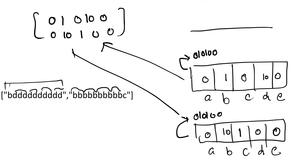


Input: strs = ["eat","tea","tan","ate","nat","bat"]

Output: [["bat"],["nat","tan"],["ate","eat","tea"]]



```
public static List<List<String>> group_Anagrams(String[] arr) {
    HashMap<String, List<String>> map = new HashMap<>();
    for (int i = 0; i < arr.length; i++) {
      String key = GetKey(arr[i]);
      if (!map.containsKey(key)) {
          map.put(key, new ArrayList<>());
      }
      map.get(key).add(arr[i]);
    }
    List<List<String>> ll = new ArrayList<>();
    for (String key : map.keySet()) {
       ll.add(map.get(key));
    }
    return ll;
  }
```

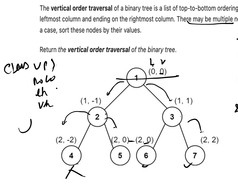


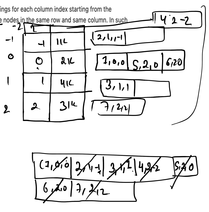
Given the `root` of a binary tree, calculate the **vertical order traversal** of the binary tree.

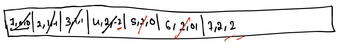
For each node at position `(row, col)`, its left and right children will be at positions `(row + 1, col - 1)` and `(row + 1, col + 1)` respectively. The root of the tree is at `(0, 0)`.

The **vertical order traversal** of a binary tree is a list of top-to-bottom orderings for each column index starting from the leftmost column and ending on the rightmost column. There may be multiple nodes in the same row and same column. In such a case, sort these nodes by their values.

Return the **vertical order traversal** of the binary tree.

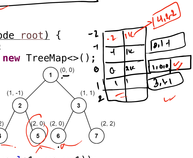


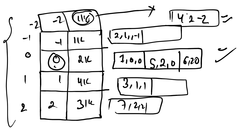


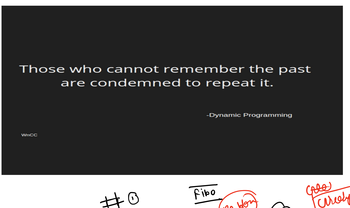


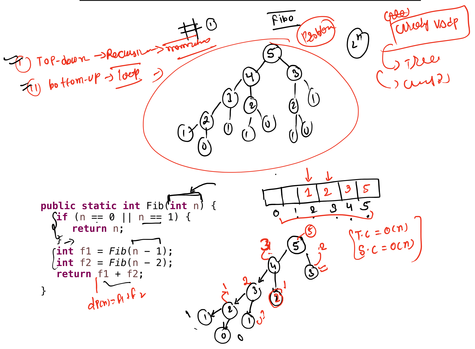
```
public List<List<Integer>> verticalTraversal(TreeNode root) {
    Queue<VerticalPair> q = new LinkedList<>();
    TreeMap<Integer, List<VerticalPair>> map = new TreeMap<>();
    q.add(new VerticalPair(root, 0, 0));
    while(!q.isEmpty()) {
      VerticalPair vp = q.poll();
      if(!map.containsKey(vp.v)) {
        map.put(vp.v, new ArrayList<>());
      }
      map.get(vp.v).add(vp);
      if(vp.node.left!=null) {
        q.add(new VerticalPair(vp.node.left, vp.l+1, vp.v-1));
      }
      if(vp.node.right!=null) {
        q.add(new VerticalPair(vp.node.right, vp.l+1, vp.v+1));
      }
    }

  }
```









```
public static int Fib(int n) {
    if (n == 0 || n == 1) {
      return n;
    }
    int f1 = Fib(n - 1);
    int f2 = Fib(n - 2);
    return f1 + f2;
  }
```



```
public static int FibTD(int n, int[] dp) {
    if (n == 0 || n == 1) {
      return n;
    }
    if (dp[n] != 0) {// dp apply kra hai
      return dp[n];
    }
    int f1 = FibTD(n - 1, dp);
    int f2 = FibTD(n - 2, dp);
    return dp[n] = f1 + f2;// yaad kra hai
  }
```