# CS2040S Tutorial 4

AY 24/25 Sem 2 — github/omgeta

Q1. (a.)
1. Find the successor of node 70
2. Since the right subtree is available, search there
3. Successor found is node 72, swap values and delete the node

(b.) No; we rotate right on node 72 which is left-heavy.

(c.) $39, 52, 57, 58, 65, 68, 70, 72$

(d.) Use the 2 bits to store balance factor. We can use 2 bits excess-1 to store values $-1, 0, 1$ for balanced AVL trees. If an operation were to make the tree imbalanced, we just balance it and set the new balance factor instead.

(e.)
1. Take first element to be the root and partition around it
2. All elements less than the root form the new left subtree
3. All elements more than the root from the new right subtree
4. Recursively apply the steps 1-3 to the left and right subtrees

Q2. Pseudocode:

```
Node[] inOrder(Node root) {
    Node[] res = new Node[];
    Stack s = new Stack();
    Node curr = root;

    while (curr || !s.isEmpty()) {
      // scan left-wise
      while (curr != null) {
        s.push(curr);
        curr = curr.left;
      }

      // curr is null now
      curr = s.pop();
      res.push(curr);

      // add right node now
      curr = curr.right;
    }

    return res;
}
```

Q3. (a.) Bites left on winning: 1

(b.) Bites left on winning: $O(\log n)$,Total bites: $O(n)$,

1. Tournament-style, compare every 2 and continue with winners

(c.) Bites left on median: $O(\log n)$,Total bites: $O(n)$,

1. QuickSelect-style, choosing pivot plates at random and recursing on the side with less than $\frac{n}{2}$ elements

Q4.

Q5.

1. Calculate the height and depth of each node using in-order traversal and DFS respectively, each $O(n)$
2. Store the results in an array of Pairs, grouping by depth
3. When a node is removed with depth $d$, find the node with the maximum height $h_d$ with the same depth $d$
4. Our final height will just be $d + h_d$