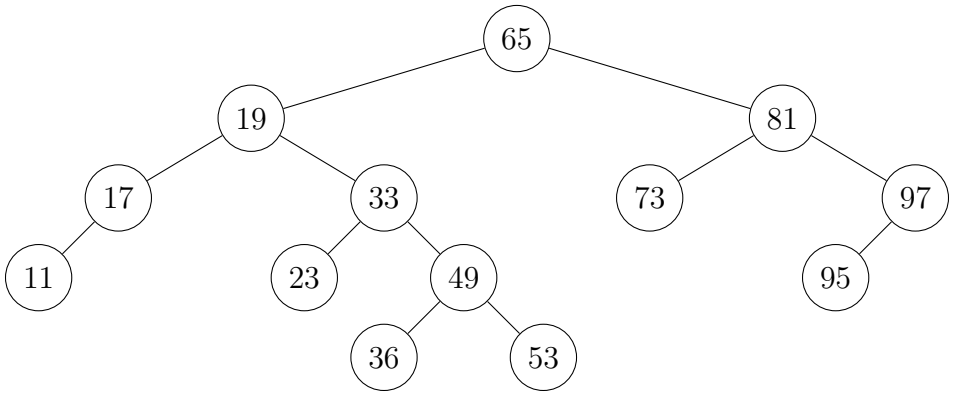
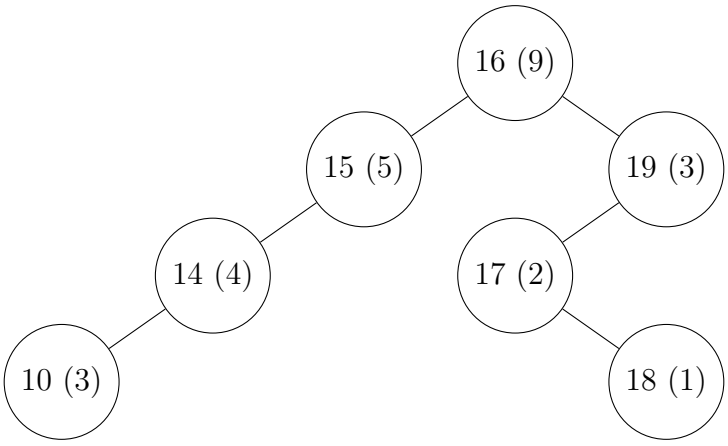
CS240E S25	**Tutorial 4**	June 5

Q1. AVL-tree height. Define a 2-AVL tree to be a binary search tree where for every node, the difference of heights of its left and right subtree is at most 2. Prove that a 2-AVL tree has height at most $3 \log n$ where n is the number of nodes in the tree.

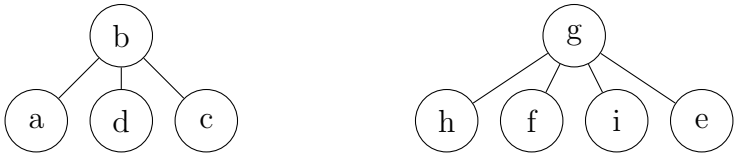
Q2. AVL-tree operations. Consider the AVL Tree shown below and perform `insert(61)`, then `delete(73)`.



Q3. Scapegoat tree operations. Insert the key 11 in the following scapegoat (2/3)-tree. The numbers in the brackets are the number of nodes in that subtree.



Q4. Stars (amortized analysis). We have a data structure to maintain collection of stars (height-1 trees).



Every child knows its parent. It supports three operations:

- *new-star*(x) : creates a new star whose only member is x
- *find-star*(x) : returns a handle to the root of the star containing x
- *merge*(x, y) : merges the stars that contain x and y

new-star(x) is implemented in constant worst-case time by simply creating a new star with x as its only element. Similarly, *find-star*(x) is implemented in constant worst-case time by returning x's parent pointer.

The operation *merge*(x, y), however, can be slow: it sets the parent pointer of all elements of y's star to *find-star*(x), in time proportional to the size of y's star (i.e. the number of element's in y's star).

Let n be the number of objects currently stored.

(a) Construct a sequence of $\Theta(n)$ operations that requires $\Theta(n^2)$ time.

Hence, conclude that the amortized cost of all operations using the aggregate method is $\Theta(n)$.

(b) We may *augment* this data structure with a *size* field at the root: now every root knows the size of its star. Now rather than breaking ties arbitrarily during *merge*, we always set the parent pointers of a smaller star.

Show using the aggregate method that the amortized runtime of all operations is $O(\log n)$.

Hint: argue that any sequence of m *new-star*, *find-star*, and *merge* operations, n of which are *new-star* operations take $O(m + n \log n)$ time.

Q5. van Emde Boas tree tutorial. See `veb_note.pdf`.