Q2b  • Many students wrote “delete the value”, without explaining how to delete it, i.e., without explaining the process of swap and remove.
    • Some students called both fix-up and fix-down without any condition checks.

Q2c  • Many students had incorrect order in one or both of the rounds because of incorrect radix calculation.

Q2e  • Students directly tried to use the sorting lower-bound (i.e. we know that sorting has an $\Omega(n \log n)$ lower-bound, and so, this gives an $\Omega(5 \log 5)$ bound for the case when $n = 5$ (in most cases, students got $0/3$ for only saying this with no other details). There is another issue here that $\Omega(5 \log 5) = \Omega(1)$ since everything is constant. Marks were not deducted for the second part if students were explicit about $\log(5!)$ being a lower bound.
    • Other students tried proving a lower bound by giving an algorithm (either directly or indirectly). These students also got $0/3$ for such reasoning.

Q2f  • The question asks to do initial insertions super fast ($O(1)$ time amortized would be sufficient). The next priority was that searching is fast in the future. This should have suggested the MTF heuristic be used.
    • Many students gave solutions using AVL trees where all operations were $O(\log n)$. While things like “superfast” may sound subjective and $O(\log n)$ does sound fast, the lack of emphasis for the speed of the other operations should have indicated that those operations need not be $O(\log n)$ and a data structure that supports $O(1)$ insertions should have been used.

Q2g  • Several students did not correctly handle the base-case, usually setting the height to be greater by 1. A tree with only a single node should have a height of 0, while an empty tree should have a height of $-1$.
    • Most students who correctly asserted that the number of iterations is equal to the height of the tree failed to note that each iteration would run in constant-time.
    • Several students tried to represent the runtime as a recurrence relation, claiming there is a recursive call of $T(n/2)$, which is not justified since AVL Trees do not enforce any constraints on the number of nodes in each subtree.
    • Most students traversed the longer subtree when the balance was not 0, even though it would be faster to traverse the shorter subtree and add 2 to the height instead of 1. This was not penalized, since the worst-case asymptotic runtime is the same for both algorithms.
A few students designed the algorithm to traverse both subtrees for either all cases, or only for the balance = 0 case. This leads to a worst-case runtime of $\Theta(n)$, which is very inefficient compared to the optimal $O(\log n)$ algorithms. Note that for the balance = 0 case, both subtrees would have the same height, so it would suffice to pick either one to traverse through.

Q3a

- Some common errors were providing proof for only $O$ bound, incorrect values of the constraints, missing constraint values, steps and/or justification.
- Some students had a negative constant $c$ for $\Omega$ proof and incorrect inequalities.

Q3b

- Many students incorrectly analysed the first two while loops
- Some students did not justify why the second loop runs $\log n^n$ times. A correct justification would be the index starts from $n^n$ and repeatedly divided the quantity by 2 so it runs for $\log n^n = n \log n$ iterations.
- Some students did not plug in $i = n$ in their analysis of the second loop and said the runtime of it was $i \log n$.

Q3d

- Students incorrectly set up expectation for even numbers (they mostly got the odd part right). Some replaced all constants with asymptotic notation which wouldn’t give a tight recurrence.
- Many students did not solve in terms of $n$ all the way: leaving $T(n)$ on both right and left side of the equation.

Q4

- Several students used too much space by keeping track of all items ever inserted into the data-structure. This would usually happen if a student used a linked list or an array to keep track of the insertion times of items.
- Many students used unbounded space by forgetting to delete from both AVL trees
- Some used a heap or linked list for second data structure, which leads to linear time search (some students thought that heaps have $\log n$ search).
- Some students used regular binary search tree instead of an AVL tree
- Many attempted to solve the problem with a single AVL tree, usually by storing both a key and a counter in a single node (assuming binary search order holds for both fields)

Q6ab

- Many students got 6(b) wrong. Some of the common mistakes were: forgetting $-\infty$ on level 4 and not returning a stack, instead just returning TRUE

Q7abc

- Many students drew an uncompressed trie, also some students did not see 7th key in the second row, so they just drew a trie with 6 keys
- In part (c) many students said that number of nodes was dependent on keys themselves. Also, those who got it right rarely stated a lower-bound on number of internal nodes.

Rest of the questions were generally well done.