University of Waterloo
CS240 Spring 2022
Assignment 3 Post-Mortem

This document goes over common errors and general student performance on the assignment questions. We put this together using feedback from the graders once they are done marking. It is meant to be used as a resource to understand what we look for while marking, as well as some common areas that students can improve in.

Question 1 [5 Marks]

- Some students did not show sufficient justification for their final running times. In particular, some steps were skipped and/or some claims were introduced without enough explanation.
- Some students did not write the correct expression for the permutations of the array which would lead to (at least) k comparisons.
- Some students used the uniform distribution over pairwise sortedness instead of the uniform distribution over permutations without justification.
- A few students forgot to multiply the number of comparisons with the number of instances which would lead to those comparisons in their summation.
- A few students found the expected running time of the algorithm instead of the average case. Since $HalfSort$ is not a randomized algorithm, its expected running time is not defined.

Question 2 [2+2+4=8 Marks]

- Most students did parts (a) and (b) very well.
- Many students did not attempt part (c) at all.
- For part (c), many students wrote that the array size decreases by 2; it actually decreases by 1 at each recursive call. ($A.\text{size} - 2 = n - 2 = (n - 1) - 1$ where $n - 1$ is the maximum index in the original array.)
- For part (c), some students derived the incorrect recurrence relation – recurrence relations are generally of the format specified in the course slides and the assignment solutions (writing a recurrence relation in terms of a single instance or using T^{exp} are both valid)

- For part (c), a few students did not fully explain the difference between the expected runtime of a single instance (i.e. $T(A, \langle x, R' \rangle))$ versus the expected runtime for all instances of a certain size (i.e. $T^{exp}(n)$). This difference is highlighted in both the course notes and course slides, so students are encouraged to review the definition of expected runtime.

- For part (c), a few students found the average running time of the algorithm instead of the expected running time.

- In general, many submitted proofs were too informal and/or skipped a few steps. When answering these types of questions, it doesn't hurt to include as many steps as you feel are necessary – this helps the graders better assess your knowledge of the material.

Question 3 [0+3+3+3=9 Marks]

- For parts (b) and (d), some students forgot to include balance factors in their AVL trees.

- For part (c), some students did not elaborate enough on how their idea would work on a set of numbers to insert into the tree (for instance, using QuickSelect to get the median of an array) and instead proposed an idea for inserting elements into an AVL tree from an existing AVL tree.

- For part (d), many students swapped the deleted node with the in-order predecessor, instead of the in-order successor. Students are advised to read the question very carefully to avoid losing marks over small mistakes.

- Overall, this question was done quite well.

Question 4 [10 Marks]

- Some solutions only considered the base case (where the height difference between the trees is 1) – the same approach may not work for height differences greater than 1.

- Quite a few solutions involved simply attaching $T2$ to $T1$ (or vice versa) and performing rotations until the resulting tree is balanced. While this approach could produce a valid AVL tree, the number of rotations required would be in $O(\log^2 n)$, which exceeds the allowed $O(\log n)$ runtime.

- Some solutions involved attaching a subtree of $T1$ of height h_2 or $h_2 + 1$ (where h_2 is the height of $T2$) as a left child of the smallest node of $T2$, with $T2$ (minus the smallest node) as the right child. This approach may leave out many elements of $T1$ since only one of its subtrees is added to the final tree.

- Some answers were missing justification of correctness and/or running time. As stated in the assignment guidelines section of the course webpage, questions that ask to give an algorithm or data structure require a description of the main idea, all the details of the algorithm, and justifications of correctness (i.e. why the algorithm produces the correct answer for all valid inputs) and running time.

Question 5 [0+3+3+3+3=12 Marks]

- For part (b), some submitted skip lists either had a tower that did not have the correct height or were missing the topmost sentinel-only level.
- Some students forgot to include the final equality comparison that is present in search() in part (b). No marks were deducted this time.
- For part (b), some students forgot to include the skip list after insertion and only included the table with the number of comparisons required for searching each key in their answer.
- For part (b), a few answers for the number of comparisons used $\leq$ comparisons instead of strict $<$ comparisons. In a skip list, we cannot reach the node we are searching for unless we are in the lowest level (S_0). Students are encouraged to step through the pseudocode in the course slides with some examples to better understand how searching in a skip list works.
- For part (d), many students forgot to account for the sentinel nodes in their proof.
- For part (d), some students forgot to start their summation from 1 instead of 0. Since the question asked about the expected number of extra nodes, the nodes at level S_0 should not be considered.
- For part (d), some stduents forgot that the probability of adding a level to a tower is p instead of $\frac{1}{2}$.
- Although many students did not attempt part (e), most submitted proofs were done quite well.