University of Waterloo
CS240 Fall 2022
Midterm Post-Mortem

This document goes over common errors and general student performance on the midterm 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 at while marking and some common areas where students can improve in.

Question 1 [10 marks]

- No common errors

Question 2 [8 marks]

- For parts d) and e), many students incorrectly answered "True".
- For parts e) and f), some students gave incorrect examples.
- For part f): some students incorrectly answered "False".

Question 3 [6 marks]

- For part a):
  - Some students gave the exact bound without mentioning Θ.
  - Some students gave incorrect runtime bound.
  - Some students gave a O-bound instead of a Θ-bound.
  - Some students did not simplify their calculations.
- For part b):
  - Some students did not specify an n_0.
  - Some students let $n_0 = 0$ instead of setting n_0 to a value greater than 0.
  - Some students set c to values that would not work (e.g. a value greater than $\frac{1}{5}$).
  - Some students proved a O-bound instead of a Ω-bound.
- For part c):
  - Many students incorrectly set $n_0 = c$.
  - Many students did not specify n_0.
  - Some students used the limit rule. But the question asked to prove from definition.

- Some students stated n_0 in terms of c at the beginning of their solutions instead of deriving n_0 at the end.
- Some incorrect answers used $\geq$ instead of a strict $>$, so they produced a final answer of $n_0 = c$ instead of $c + 1$.

Question 4 [6 marks]

- Many students answered parts a), c), and d) incorrectly or left them blank.

Question 5 [6 marks]

- For part e), many students tried to prove the statement by showing that `insert` and `deleteMax` should be $O(\log n)$ instead of proving by contradiction, that assuming an insert and deleteMax operations with running time $O(\log \log n)$ gives a PQ-sort algorithm with running time $O(n \log \log n)$, thus violating the $\Omega(n \log n)$ lower bound on comparison based sorting.

Question 6 [4 marks]

- Many students did not use a decision tree to justify their answer.
- Many students considered only the best-case scenario of 3 weighings.
- Many students did not present the total number of outcomes: $\binom{7}{3} = 35$ in their solutions.
- Many students attempted to consider all the different permutations of 3 coins, 2 coins, and/or 1 coin weighings (on a single side of the scale).
- Some students attempted to prove the statement by providing an algorithm. Providing an algorithm can be used to establish an upper bound, but not a lower bound.
- Some students mistakenly claimed the number of children nodes at each parent of the decision tree is 2 instead of 3.

Question 7 [4 marks]

- For part b):
  - Many students incorrectly calculated the AVL balance factor as [height of left-subtree] subtracting [height of right subtree] instead of the other way around.
  - Many students mistakenly calculated the height of the subtree at each node instead of the balance factor.
- For part d):

  - Many students did not understand the meaning of "inorder predecessor" and selected the wrong nodes to delete.
  - Many students incorrectly selected "True" due to incorrectly positioning elements in part a).

Question 8 [4 marks]

- Many students did not answer infinite for part b), subpart A.
- Some students used O-notation instead of Θ-notation.

Question 9 [4 marks]

- Many students did not have $O(1)$ expected time, likely due to confusing the difference between expected- and average-case runtime.
- A common incorrect algorithm provided was to check a single random index, then perform a linear scan (either from the beginning of the array or from the random index).
- Another common incorrect algorithm is to provide only a non-randomized linear scan.
- Many students' algorithms did not have $O(n)$ as the worst-case, some of these students re-wrote `foo2` from the previous question.
- Some students re-wrote `foo2` or an algorithm similar to `foo2` but halted the recursive calls after n times, so the algorithm had a chance of failure.
- Some students tried to randomly choose an index that has not already been picked.
- Some students' algorithm required initializing an $O(n)$ data structure, and they often had a loop to pick a random number unseen before, but ignore this in their runtime analysis.
- Some students incorrectly assumed that picking a new number unseen before can be done in $O(1)$ time without justification.