University of Waterloo
CS240E, Winter 2025
Written Assignment 1 Post-Mortem

This document goes over common errors and general performance on the assignment. We create it using feedback from the graders, and it is meant to be used as a resource to understand common areas that we can improve in.

Question 1 [9 marks]

- This question was done well.

Question 2 [3+6=9 marks]

- Part (a) was done well.
- For (b), many solutions gave creative constructions for counterexample functions, and most were correct. However, some solutions had incorrect negations of the definition of big-O.

Question 3 [5 marks]

- To our surprise, very few solutions used the intended approach of relating the mystery algorithm to heapify. It is much shorter than the direct approach, hence why this question was worth relatively few marks.
- Most solutions instead derived the answer using some combination of integral bounds, logarithm formulas, or Stirling's approximation, and this was done well for the most part.
- Some solutions only showed that $f(n) \in O(n)$ instead of $f(n) \in \Theta(n)$ (the latter is what we mean by "tight bound").

Question 4 [2+3+4=9 marks]

- Some solutions to (a) forgot to check if a node was NULL before accessing its data.
- Many solutions to (b) cleverly realized that reusing the algorithm from (a), if implemented a certain way, will also achieve the required worst-case runtime. This was not intended, but goes to show that there are often different correct solutions to a given problem. (Nonetheless, we will endeavour to not be outsmarted in the next offering of this course.)

- Some solutions to (b) mixed up the expected $O(\log n)$ and worst-case $O(h)$ run time for `meldableHeap::merge`, resulting in incorrect algorithms or run time proofs.
- In all parts, some solutions forgot to show the correctness and/or run time of their algorithms. Unless otherwise stated, this is always necessary to receive full marks.

Question 5 [3+7+3=13 marks]

- Many solutions to (a) gave a tree that was not a flagged tree, usually because the left subtree of the root was not full.
- Some solutions to (b) failed to check for binomial heap-order violations above the immediate parent.
- Some solutions to (c) used a particular priority queue implementation, rather than the abstract data type, for their reduction.