University of Waterloo
CS240E, Spring 2026
Written Assignment 2 Post-Mortem

Question 1 [1+1+7=9 marks]

There were very few mistakes in this question. Note that some of you chose to use a recurrence, but made mistakes in either forming the recurrence or simplifying.

There are multiple ways to arrive at the limit for part (c). You can use both the bound provided in the question, or arrive at an exact sum via the telescoping technique seen in Tutorial 1.

Question 2 [6+6=12 marks]

Although this is a calculation-heavy question, you are expected to explain your steps. In the future, marks may be deducted for insufficiently explained proofs.

Part (a) has the $\leq$ sign due to having no need to recurse when the left subtree is empty. If you didn't end up with a $\leq$ sign, this is likely the issue.

Part (b) is generally well done.

Question 3 [6+3=9 marks]

Most of you got full marks.

In part (a), a common mistake is trying to "cut" the linked list in half, which takes $\frac{n}{2}$ time. A recurrence like $T(n) = 2T(\frac{n}{2}) + \Theta(n)$ is in $T(n) \in \Theta(n \log n)$, which is slower than necessary.

Part (b) has a few small issues. Note that by using median-of-medians, you can perform quick-select for the median in worst-case $O(n)$, so selection in $O(n)$ is not a contradiction.

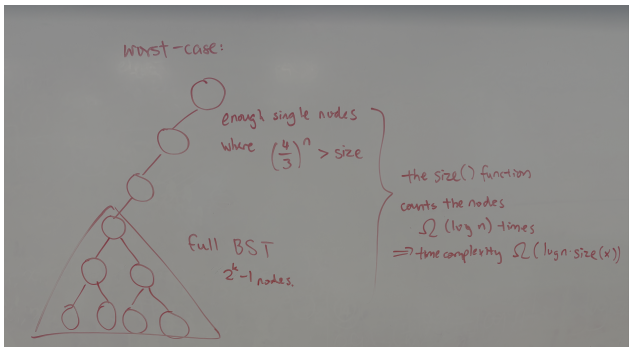
Question 4 [2+3+2+4+3+4=18 marks]

Parts (a), (c), and (f) are mostly problem-free.

Part (b). Log is not a primitive or constant-time operation. You can compute log by repeated division in $O(\log n)$, which is fast enough, but it's not $O(1)$ and should be mentioned in time complexity analysis / implemented in code.

Part (d). Exponentiation is not a primitive operation. If you use exponentiation (even fast exponentiation which takes $\log(\textit{exponent})$ time), you will end up with $\log n \log\log n \in \omega(\log n)$.

Also, we don't have parent links in this course. You can search(z) to find the newly inserted node and keep track of what nodes you encountered on the way.



Some of you repeatedly computed the size of the subtree, instead of only computing size() on the sibling and using previous work. This causes the time complexity to be too high.

Part (e). Some of you did not show that the height of the subtree decreased after rebuilding rigorously, or only proved that z has decreased in height.