University of Waterloo
CS240, Fall 2021
Midterm Post Mortem

Problem 1 [2+2+2=6 marks]

- For part a):
  - Many students used O notation or Ω notation
  - Many students got the worst case runtime wrong. Common incorrect answers were $\Theta(n)$ or $\Theta(n^2)$
  - A few students got best case runtime wrong. Common incorrect answer was $\Theta(n)$.

- For b):
  - Many students answered along the lines of "AVL uses a dynamically allocated linked node structure because restructuring of the tree happens a lot whereas restructuring doesn't happen often in heaps" or explains that linked lists are better structures to do rotations.
  - Some students mentions that using arrays for AVLs would waste space but doesn't explain in detail why

- For c), many students got either the total calls to BucketSort wrong, or the depth of recursion wrong or the order of elements in the resulting array wrong.

Problem 2 [2+3=5 marks]

- For a), many students only give the bound without any explanation

- For b):
  - Many students get 7 using the bound nlogn instead of decision tree
  - Some students try to manually construct an example that requires 7 comparisons

Problem 3 [4 marks]

- Many students gave each item in the heap an additional field that functioned as a index/order. However, many did not include enough details on how to modify the new comparison and/or fix-down.
- Many students did not specify how to use this index to decide which child would be promoted (if they had equal keys)
- Many students did not account for the fact that elements with the same key are removed from the heap in reverse order of what they appear as.
- Some students left out their runtime analysis or exceeded the requirement.

Problem 4 [3+3+5+2=13 marks]

- For a):
  - Many students did not cover all cases such as $0 < c < 1$ and $c >= 1$.
  - Some students provide a very large n_0 or a magical bound without deriving it.
  - Some students ignored the constant term and continued the proof without it.
  - A few students did not use first principles.
  - A few students did not understand the definition of little-o
- For b):
  - Some students did not use limit rule but used something not formal.
  - Some students did not write the computation steps and give infinity for the limit.
  - Some students did not simplify their equations.
- For c):
  - Most students didn't correctly derive of sufficiently explain the runtime for the first set of loops (see Module 1 slide 31 for a similar example).
  - Many students stated that it is $O(n^3)$ without enough justification.

- Several students tried to analyze the nested loops independently, stating for example that the inner loop by itself is O(n) when it should really depend on i and j.
- Forgetting to include the time to compute n^i in the else case.
- Some students thought the else case would never run at all.

- For d):
  - Lots of off-by-one errors.
  - Rounding: some students rounded fractions to integer values (losing the remainder) and underlining the integer as if that was their final answer.

Problem 5 [2+2=4 marks]

- Some students did not write the balance factors
- Some students did not draw part b correctly. Part a is generally well done.
- A few students wrote +1 when the left child is larger than the right child in terms of balance factors.

Problem 6 [4 marks]

- Some students forget to mention the total time for partition is now $O(n^2)$
- Some students solve the recurrence incorrectly

Problem 7 [3+1+2=6 marks]

- For a):
  - Some students did not have the keys in sorted order in the skiplist.
  - Some students did not draw the top level, had incorrect tower heights, did not draw the horizontal or vertical edges.
  - Some students had different tower heights in their skiplist that didn't match what they had in the table.
  - Some students didn't draw the skip list at all

- For b), some students gave “at least” height instead of “exactly” height
- For c):
  - Many students did not take ceiling of their log function.
  - Many students did not include a top level into their count.
  - Many students used log with base 2.

Problem 8 [6 marks]

- Several students tried to keep track of the latest inserted element only, but did not consider that the deletion of this element would require finding the next latest inserted element to track, which cannot be done without a larger structure.
- Many students stored the timestamp in the secondary structure, but the space required for the timestamp depends on the number of elements ever inserted in the history of the structure, and is not bounded with respect to n.
- Several students used the size of the structure during insertion to represent the order of insertion, but this doesn’t account for the scenarios in which elements are deleted from the structure, so a later insertion might occur with a smaller tree size than an earlier insertion.
- Many students did not provide any method of efficiently finding an element in the secondary (time) structure based on the key, which is necessary for delete(k).
- Many students used a skip-list instead of an AVL Tree, but a skip-list cannot guarantee O(log n) time for any operation. It is only the expected runtime which is bounded.
- A few students implemented an array for the secondary structure, which would not be able to maintain the required runtime and may also fail the space requirement. Deletion of an element would require either shifting all later elements (runtime Theta(n) worst-case), or keeping the indices empty (array size no longer bounded by n, runtime to find the last undeleted element is also unbounded).