University of Waterloo
CS240, Spring 2021
Midterm Post Mortem

Problem 1 [3+1+1 = 5 marks]

- For part a), quite a few students came up with a $o()$ relation
- For b), many students gave an answer in the form $n \log n$, others had the right expression but forgot the ceiling.
- Some students left an asymptotic bound.

Problem 2 [3+3+3+3=12 marks]

- For a), several students didn't reorder the entries at all (i.e. calculated the average cost with the given order a,b,c,d)
- For b), storing every other index in S_1 was a very common answer
- For c), many students tried and failed to solve the running time with a recurrence relation.
- Many students observed that running insertion sort on a list of size $\sqrt{n}$ takes time $O(n)$, but did not further observe that insertion sort would be run $O(\sqrt{n})$ times.
- For d), $O(n \log n)$ and $O(n^2)$ were very common answers

Problem 3 [4+4 = 8 marks]

- For a), some students tried to work with a fixed c like 100 or 500.
- A few students had solutions that involved multiplying/dividing potentially negative numbers across inequalities without addressing the consequences.

- A few students did not use first principles in their proof.
- For b), many students had a sloppy proof without treating the floor function carefully.
- Many students got $\Theta(n \log n)$ or $\Theta((\log n)^2)$

Problem 4 [6 marks]

- Some students thought the question was asking for $O(n \log n)$, not little-o, and solved the question with that assumption. In general, the most common error was violating the $o(n \log n)$ runtime.
- Some students used "standard" sorting algorithms or "k-way-merge-esque" algorithms that don't meet the runtime constraints.
- Some students realized it was little-o but didn't really understand what that means, stating things like $(n-1)\log n$ or $n(\lfloor \log n \rfloor)$ is $o(n \log n)$ because it's "less than" $n \log n$.
- Many students created a heap with $\log n$ elements and forgot the $+1$.
- Many students forgot to use the floor function on $\log n$.

Problem 5 [2+2+4 = 8 marks]

- For a) ii), several students did not use the original heap but rather than the result of a) i)
- For b), the answers were about a 50/50 split between yes and no.

Problem 6 [2+3 = 5 marks]

- For a), students gave an expression instead of a number for either array size or iteration count.
- Students assume a radix of 10, leading to array size of 10 and 4 iterations.

- For b), a lot of students conclude (usually without proper support) that the running time is $\Theta(mnR)$

Problem 7 [2+3 = 5 marks]

- Quite a few students were not familiar with the procedure of self-balancing, and made some errors trying to construct "balanced" trees using rotation.
- Some students put node 72 in the wrong place as they forget to compare with the ancestor node after balancing.
- For b), some students forgot to include node 60 or 65 after balancing.

Problem 8 [4 marks]

- Many students did not realize that the best case is when all the distinct keys we search for are in front of the list, and identical keys are 'grouped together'. Some students thought searching for $\sqrt(n)$ distinct items keys in front of the array takes $O(1)$ per key, some students did not add up the sum $1 + 2 + ... + \sqrt(n)$ correctly.
- Many students left out m in the final asymptotic notation.

Problem 9 [4 marks]

- Many students think the probabilities are $n/3$ and $2n/3$.
- Many students think the worst case runtime is infinite or $O(n)$.
- Some students think the worst case runtime is $O(\log n)$ or $O(n^2)$.

Problem 10 [3+3+4 = 10 marks]

- For a), some students got the order incorrect or didn't know how insertion worked.

- For b), many students used an exponent of 100. Many students forgot to incorporate n, or used it incorrectly.
- For c), many students created their own search algorithm instead of simply using the "default" skiplist search. This takes extra time, and has the chance of being invalid by introducing illegal operations (moving left or up in a skiplist).

Problem 11 [8 marks]

- Many students tried to use heaps or skip-lists, but skip-lists do not guarantee O(log n) worst-case runtime for any operation (they are only logarithmic in expectation), while heaps cannot guarantee deletion by key in O(log n) time, due to having to search for the key (worst-case is linear!).
- Many students maintained linked lists to handle duplicates, but its deletion would be linear to its size (which can be as bad as n). Many students did not even bother with handling duplicates, whose answers implicitly assumed that keys are distinct.
- Several students correctly defined two variables to store the current minimum or maximum, but did not update (or did not elaborate on how to update) them during insert/delete. Some students set the new min/max to the parent of the previous min/max when it is deleted, not realizing that it should instead be set to the right/left child of the deleted node if it exists.