Question 1  True or False

Many students made some mistakes from part (i) to (m), especially part (i).

Question 2  Short Answer

(a)  • It is not enough to prove that fix-up or fix-down requires at least $O(\log n)$ comparisons since there may be an algorithm that doesn’t use these functions.
    • The proven lower bound for comparison-based sorting ($\Omega(n \log n)$) need to be used to get a contradiction.

(b)  • Some students said an optimal choice of $S_1$ has $\frac{n}{2}$ keys where the worst-case run-time was $\Theta(n)$.
    • Some students obtained the expression $\Theta(k + \frac{n}{k})$ without giving the optimal value of $k$.

(c)  • Many students gave the correct search sequence but incorrectly thought that the run-time for Transpose was $O(n^2)$ or $O(n^k)$.
    • Some students did not give the run-time for Transpose.
    • Some students defined their own $L$.
    • Some students said that each search ”goes to the end” instead of saying it is in $O(n)$.

(d)  • Some students forgot to mention that finding the min/max of 3 elements takes constant time (No marks deducted).
    • Some students tried to use the $\Omega(n \log n)$ bound on comparison-based sorting algorithms; however, it does not hold here since we are given 3 sorted arrays to work with.
    • Some students only found a solution for a specific case (not general).
    • Some students simply claimed yes without any justification.
    • Some students used insertion sort which has $O(n^2)$ run-time.
    • Some students used Count sort or Radix sort, assuming that $A_1$, $A_2$, and $A_3$ only contain bounded integers.
    • Some students claimed that if merge can be done in $\Theta(n)$ time for 2 lists, then it can also be done in $\Theta(n)$ time for 3 without any justification.
Some students made some simple mistakes when adding up the number of "n" it takes in the run-time.

- Some students just added up frequencies multiplied by the position instead of the probability; i.e. $1 \times 4 + 2 \times 2 + 3 \times 1 = 11$ (without the denominator 7).
- Some students used 3 as the denominator instead of 7.
- Some students calculated the cost of search using 0, 1, and 2 (or even left them in variables like $a$, $b$, and $c$) instead of 1, 2, and 3.

**Question 3  Asymptotic Analysis**

(a) Some students proved $O$ and $\Omega$ separately and concluded $\Theta$ without stating $c_1$, $c_2$, and $n_0$.
- Some students simply stated the value of $n_0$ without any justification.

(b) Some students left out part of the definition (there exists $n_0$ such that it holds for $n \geq n_0$).
- Some students only prove that the inequality holds for some specific $c$ instead of any $c$.
- Some students chose $n_0 = c$ instead of choosing some $n_0 > c$.
- Some students used limit rule or little-o definition instead of little-omega definition.

(c) Many students got the incorrect run-time $O(n^2 \log n)$.
- Some students missed some steps.
- Some students wrote $2n^2$ in the second iteration instead of $(\frac{n}{3})^2 + n^2$.
- Some students got a run-time with $k$ in it where they assumed $n = 2^k$.
- Some students did not simplify $O(4^{\log n})$ which is equivalent to $O(n^2)$.

(d) Some students found $\log n$ for the inner loop instead of $\log i$ and assumed $\sqrt{n \log(n)}$ instead of doing the analysis of the sum.
- Some students got the incorrect range of $i$.
- Some students bounded the inner loop to $\frac{i}{3}$ instead of $\log_3(i)$.

**Question 4  Priority Queues**

(b) Some students performed deleteMin $k$ times from the original heap, which has run-time $O(k \log n)$.
- Some students gave $O(k^2)$ algorithms.
- Some students gave incorrect algorithms.
• Some students used a max-heap as the auxiliary instead of a min-heap.
• Some students missed some details in their analysis of run-time.

(c) Some students incorrectly assumed the last layer of the heap is full.

Question 5  AVL Trees

(a)  • Some students calculated the balance backwards (height(left) - height(right)).
• Some students labelled nodes with height instead of balance.
• Some students got the balance of leaf nodes wrong.

(b) Some students incorrectly inserted 8 as the right child of 10.

(c)  • Some students gave the answer $M(h) = M(h - 1) + 2^h$ which is logically sound
      but not the standard answer.
• Some students misunderstood the question and tried to give an algorithm to
      compute $M(h)$ and $m(h)$.
• Some students gave a simplified equation instead of a recursive formula.

Question 6  Skip Lists

(a)  • Many students did not draw the vertical links between nodes in the skip-list, while
      some students missed the horizontal links as well.
• Many students did not draw the topmost level which only contains the sentinels.
• Some students gave an unsorted skip-list.

(b)  • Some students did not give enough details for their justification.
• Some students did not calculate the result of the summation.
• Some students incorrectly said that the series converges since the $n$-th term
      approaches 0 when $n$ approaches infinity (harmonic series is a counterexample).

Question 7  Algorithm Design and Analysis

• Many students failed to ensure that the indices that result in the desired sum are
      distinct. That is, the algorithm may return "Yes" on a "No"-instance where M is even
      due to adding M/2 with itself.

• Many students used MergeSort or QuickSort, which are not in-place. The non-recursive
      variant of QuickSort still uses $\Omega(\log n)$ space (in the form of an explicit stack, as
      opposed to the recursive call stack). QuickSort also fails to guarantee $O(n \log n)$ worst-
      case run-time. The only in-place $O(n \log n)$ sorting algorithm covered in this course is
      HeapSort.
• Several students tried to use a hash table (not in-place) without specifying how collisions are resolved. Although hashing is not included in the midterm coverage, students can still design data structures if they clearly specify the implementation details.

• A few students referred to programming data types like C++ map or Python dictionary. These kinds of data types cannot be used to represent data structures for algorithm design without demonstrating the details on how they work.