You're expected to work on the problems before coming to the lab. Discussion session is not meant to be a lecture. TAs will guide the discussion and correct your solutions if needed. We may not release solutions for all problems. If you're better prepared for discussion, you will learn more. TAs will record names of the students who actively engage in discussion and report them to the instructor. The instructor will factor in participation in final grade.

1. (Intermediate) Draw the decision tree for Merge sort operating on three element; so Merge-sort(A, 1, 3) where the input is $A[1...3] = \langle a_1, a_2, a_3 \rangle$. Use the decision tree for Insertion sort in Fig. 8.1 as a model. Note that each internal node must correspond to a comparison of two given elements from $A[1...3]$.
   Sol. See the handwritten pdf file.

2. (Basic) What is a stable sort?
   Sol. elements with the same key must appear in the same order as they did before the sorting.

3. (Basic) Suppose before sorting we had three vectors $(4, 2), (2, 5), (3, 2)$ in this order. Suppose we sorted them by their second coordinate value and we obtained $(3, 2), (4, 2), (2, 5)$. Is this a stable sort?
   Sol. No. As $(4, 2)$ and $(3, 2)$ both have the same value in the second coordinate, $(4, 2)$ must appear before $(3, 2)$ as it was the case before the sorting. So, the unique stable sort result should be $(4, 2), (3, 2), (2, 5)$.

4. (Basic) Explain why the decision tree for any comparison based sorting algorithm must have at least $n!$ leaves.
   Sol. There are $n!$ possible outcomes and each of them must appear in one of the leaves.

5. (Basic) Show $\log_2 n! = \Omega(n \log n)$.
   Sol. For simplicity, assume that n is even. $n! \geq (n/2 + 1)(n/2 + 2)...(n/2 + n/2) \geq (n/2)^{n/2}$.

6. (Basic) Illustrate the operation of Counting-Sort on $A = \langle 6, 0, 2, 0, 1, 3, 4, 6, 1, 3, 2 \rangle$.
   Sol. See the lecture slides with annotations.

7. (Basic) We want to sort n dates, where each date is of the form $\langle year(xxxx), month(xx), day(xx) \rangle$; for example, $\langle 1935, 10, 08 \rangle$. Explain how. What is the running time?
   Sol. We use Radix sort. More precisely, we can iteratively sort the dates using counting sort – first by date, then by month, and finally by year. For each run of counting sort, note that $k \leq 2022$. So, the run time is $O(3 * (n + 2022)) = O(n)$.

8. (Advanced) Describe an algorithm that, given n integers in the range from 0 to k, preprocesses the input and then answers any query on how many of the n integers fall into range $[a...b]$ in $O(1)$ time. Your algorithm should use $O(n + k)$ preprocessing time.
   Sol. Following the counting sort, we can compute $C[0...k]$ such that $C[i]$ is set to the number of elements no greater than i. Then, given a query, we just need to return $C[b]-C[a-1]$; here $C[-1] = 0$.

9. (Intermediate) Which of the following sorting algorithms are stable: insertion sort, merge sort, and quicksort (according to their implementations in the textbook)?
   Sol. Insertion, Merge. But not quicksort.

10. (Advanced) Suppose we are given n 2-dimensional vectors. Suppose we want to order them in the following order: For two distinct vectors $v_i = (a_{i1}, a_{i2})$ and $v_j = (a_{j1}, a_{j2})$, v_i must appear before v_j if $a_{i1} < a_{j1}$, or $a_{i1} = a_{j1}$ and $a_{i2} < a_{j2}$. Explain how. What is your running time? The faster, the better. **Sol.** We stable-sort the n vectors first by the second coordinate, and then by the first coordinate. If we use an $O(n \log n)$ time stable sorting algorithm, such as Merge sort, we can get this done in $2 * O(n \log n) = O(n \log n)$ time.

11. (basic) Using Figure 8.4 as a model, illustrate the operation of Bucket-Sort on the array $A = \langle .79, .13, .16, .64, .39, .20, .89., 53., .71, .42 \rangle$.
    Sol. See the textbook or lecture slides.

12. (intermediate) Explain why the worst-case running time for bucket sort is $\Theta(n^2)$? What simple change to the algorithm preserves its linear average-case running time and makes its worst-case running time $O(n \log n)$?
    Sol. All elements could be assigned to the same bucket. Then, sorting the linked list of n elements via insertion sort could take $\Omega(n^2)$ time. A simple fix is to sort the list using Merge sort or heap sort to, which will improve the running time to $O(n \log n)$.

13. (advanced) Describe a (worst-case) linear time algorithm that decides if a given sequence of integers $\langle a_1, a_2, ..., a_n \rangle$ is a permutation of $\langle 1, 2, 3, ..., n \rangle$. Your algorithm only needs to say Yes or No. (Hint: counting sort.)