CSE100 Algorithm Design and Analysis Discussion Ch. 2

You're expected to work on the problems before coming to the lab. Discussion session is not meant to be a one-way lecture. The TA will lead the discussion and correct your solutions if needed. For many problems, we will not release 'official' solutions. If you're better prepared for discussion, you will learn more. The TA is allowed to give some bonus points to students who actively engage in discussion and report them to the instructor. The bonus points earned will be factored in the final grade.

1. (basic) Run Insertion Sort on input array $A = \langle 5, 8, 4, 2, 3, 1 \rangle$, show how the array looks before each iteration of the for loop in line 1; see page 18 for the pseudo-code.

2. (basic) Show the operation of Merge sort on the array $A = \langle 7, 4, 2, 8, 3, 1, 5, 6, 9 \rangle$ as shown in Fig. 2.4.

3. (Intermediate) The following is a pseudo-code of Selection-Sort. Describe Selection-Sort in plain English.

```
Selection-Sort(A)
1.  n = A.length
2.  for j = 1 to n-1
3.   smallest = j
4.    for i = j+1 to n
5.      if A[i] < A[smallest]
6.        smallest = i
7.    exchange A[j] with A[smallest]
```

   State the loop invariant and prove that the algorithm is correct. What is the running time?

4. (basic) Suppose there are 2^n inputs to a certain problem. The running time of algorithm A is exactly 2^n for exactly one of the inputs, and 1 for any other input. Then, the running time is $O(1)$ since $2^n * \frac{1}{2^n} + 1 * (1 - \frac{1}{2^n}) \leq 2$. Is this statement correct?

5. (basic) Suppose the input sequence is already sorted. For this specific input, what is the running time of Insertion-sort and Merge-sort? Assume that they are implemented as in the textbook.

6. (basic) Answer the same question as above when the input is $\langle n, n-1, \ldots, 1 \rangle$.

7. (basic) Briefly explain the computational model, RAM (Random Access Model).

8. (intermediate) Consider the pseudocode of $Merge(A, p, q, r)$ in the textbook. Given that $A[p...q]$ and $A[q+1...r]$ are both sorted, the function call merges the two sorted subarrays into a sorted subarray $A[p...r]$. Prove the correctness of Merge. For simplicity, you can assume that $L[1...n_1] = A[p...q]$ and $R[1...n_2] = A[q+1...r]$., and $L[n_1+1] = R[n_2+1] = \infty$. (So you only need to consider from lines 10). You can assume that all elements stored in the array have distinct values.

9. (intermediate/advanced) Consider the searching problem: The input is an array $A[1 \cdots n]$ of n numbers and a value v. You're asked to find an index i such that $A[i] = v$. If there's no such index, return NIL. The following is a pseudocode of linear search, which scans through the sequence, looking for v. Using a loop invariant, prove that your algorithm is correct. Make sure that your loop invariant fulfills the three necessary properties. What is the asymptotic worst case running time?

```
1. For i = 1 to n
2.   If A[i] == v then return i
3. return NIL.
```

10. (advanced) You're given an array $A[1 \cdots n]$ of n numbers, and would like to know if the array includes two equal numbers. Describe an algorithm that does this job for you. The running time must be $O(n \log n)$.

11. (advanced) We're given three sorted arrays $A[1 \cdots x]$, $B[1 \cdots y]$, $C[1 \cdots z]$. Describe an algorithm that merges the given three sorted (in increasing order) arrays into one (sorted array). What is the running time?

12. (basic) Merge sort is always faster than insertion sort. True or false?. Justify your answer.

13. (intermediate) Observe that if the sequence A is sorted, we can check the midpoint of the sequence against v (the element we're looking for) and eliminate half of the sequence from further consideration. The binary search algorithm repeats this procedure, halving the size of the remaining portion of the sequence each time. The following is a recursion-based pseudo-code is supposed the following: returns an index i such that $p \leq i \leq r$ and $A[i] = v$ if such an index exists, and returns -1 otherwise. Prove the correctness via induction.

```
BSearch(A, p, r, v)
   If r < p, return -1
   q = the floor of (p + r)/2
   if A[q] == v return q
   else if v < A[q] return BSearch(A, p, q-1, v)
   else return BSearch(A, q+1, r, v)
```

14. (intermediate) Can you use binary search to improve the running time of insertion sort to $O(n \log n)$?