CSE 21 – Winter 2018 – Midterm 1 Solution

1. Asymptotic Notation

a. (8 points) Each of the expression below gives the processing time $T(n)$ spent by an algorithm for solving a problem of size n. Give the *lowest* big-O class in simplest form for the complexity of each algorithm.

$T(n)$	$O(\dots)$
$n\sqrt{n} + n^{1.2} + n^{1.7}$	$n^{1.7}$
$\left(10n + n^2 + n\log(n)\right)^3$	n^6
$n^3 + n^2\log(n)$	n^3
$2^n + n^2\log(n) + n!$	$n!$
$n^2\log(n) + n\left(\log(n)\right)^2$	$n^2\log(n)$
$5^n + n^5$	5^n
$3^{2n} + 2^{3n}$	9^n
$\log\left(\log(n)\right) + \left(\log n\right)^2$	$(\log n)^2$

b. (2 points) Circle True or False, and give a short explanation for your answer.

Let f, g, and h be functions from the natural numbers to the non-negative real numbers with $f(n) \geq g(n)$ for all $n \geq 1$. If $f(n) \in \Theta(h(n))$ and $g(n) \in \Theta(h(n))$ then $f(n) - g(n) \in \Theta(h(n))$.

Solution. The statement is false. Counter example: $f(n) = n^2 + n, g(n) = n^2, h(n) = n^2$.

2. Solving Recursion

(5 points) Suppose f is a function defined by the following recursive formula, where n is a positive integer,

$$f(n) = f(n-1) + n \text{ and } f(1) = 1.$$

Find a closed-form formula for $f(n)$. You may use any method discussed in the lecture.

Solution. By unraveling,

$$\begin{aligned}
f(n) &= f(n-1) + n \\
&= f(n-2) + (n-1) + n \\
&= f(n-3) + (n-2) + (n-1) + n \\
&\ \ \vdots \\
&= f(n-k) + (n-k+1) + \cdots + (n-1) + n \\
&\ \ \vdots \\
&= f(1) + 2 + 3 + \cdots + (n-1) + n \quad (\ \textit{let } k = n-1) \\
&= 1 + 2 + \cdots + n = \frac{n(n+1)}{2}
\end{aligned}$$

Final answer: $f(n) = 1 + 2 + \cdots + n = \dfrac{n(n+1)}{2}$

3. Iterative Algorithms and Loop Invariant

Given a list of distinct integers $A = (a_1, a_2, \ldots, a_n)$. A *left-to-right minimum* is an entry of the list that is smaller than all entries to its left. Formally, we say that a_j is a left-to-right minimum of A if $a_i > a_j$ for all $i < j$. By default, we also consider the first element of the list, a_1, to be a left-to-right minimum.

For example, if $A = (9, 3, 8, 4, 7, 1, 6, 2, 5)$ then the left-to-right minima of A are $9, 3$, and 1.

Let $n \geq 1$, the following iterative algorithm takes as input a list $a_1, \ldots, a_n$ of distinct integers and returns the number of left-to-right minima of the input. For instance, on using the input from the previous example, the algorithm would return 3, as there are three left-to-right minima.

procedure $LRMin$ ($a_1, \ldots, a_n$: list of $n \geq 1$ distinct integers)

1. $lrm := 1$
2. $min := a_1$
3. **for** $i := 2$ **to** n
4.   **if** $a_i < min$ **then**
5. $lrm = lrm + 1$
6. $min = a_i$
7. **return** lrm

a. (7 points) Prove the following loop invariant for $LRMin$

Loop Invariant After t iteration of the **for** loop, min is the value of the smallest element in the list $a_1, \ldots, a_{t+1}$ and lrm is the number of left-to-right minima of the list $a_1, \ldots, a_{t+1}$.

Solution. The base case is when $t = 0$. In this case, we have not entered the **for** loop in line 3, so $min = a_1$ is the smallest element in the list a_1 and $lrm = 1$ is the correct number of left-to-right minima of the list containing only a_1.

For the inductive step, suppose that the loop invariant holds for $t = k - 1$. We shall prove that it also holds for $t = k$. That is, we want to show that after k iterations of the **for** loop, min is the value of the smallest element in the list $a_1, \ldots, a_{k+1}$ and lrm is the number of left-to-right minima of the list $a_1, \ldots, a_{k+1}$.

By the induction hypothesis, before the k-th iteration, min is the value of the smallest element in the list $a_1, \ldots, a_k$ and lrm is the number of left-to-right minima of the list $a_1, \ldots, a_k$. During the k-th iteration (the index $i = k + 1$), we compare a_{k+1} with min in line 4. There are two cases

- Case 1: $a_{k+1} > min$. In this case, min is still the smallest element in $a_1, \ldots, a_{k+1}$ and a_{k+1} is not a left-to-right minimum so the number of left-to-right minima in $a_1, \cdots, a_{k+1}$ is the same as those in $a_1, \cdots, a_k$. In the algorithm, we do not change the value of min and lrm so the algorithm gives the correct output in this case.
- Case 2: $a_{k+1} < min$. In this case, a_{k+1} is a left-to-right minimum. This means the number of left-to-right minima in $a_1, \cdots, a_k, a_{k+1}$ is one more than the number of left-to-right minima in $a_1, \cdots, a_k, a_k$. Line 5 of the algorithm increases this count by 1 so this is the correct output. In addition, a_{k+1} is the smallest element in $a_1, \cdots, a_k, a_{k+1}$. Line 6 of the algorithm updates this value to be a_{k+1} so this is the correct output in this case.

This shows that the loop invariant holds for $t = k$. By induction hypothesis, the loop invariant holds for any value of t.

b. (3 points) Conclude from the loop invariant that the algorithm $LRMin$ is correct.

Solution. Since the loop invariant in part (b) holds for any nonnegative integer value t, it is true for $t = n - 1$. After $n - 1$ iterations of the **for** loop, the algorithm terminates and the output lrm is the correct number of left-to-right minima in $a_1, \cdots, a_n$.

4. Recursive Algorithms

Here is a recursive algorithm that finds the number of left-to-right minima in a list of distinct integers $a_1, \ldots, a_n$, with $n \geq 1$.

procedure *LRMinRec* ($a_1, \ldots, a_n$: list of $n \geq 1$ distinct integers)

1. **if** $n = 1$ **then return** a_1
2. **for** $i := 1$ **to** $n - 1$
3.   **if** $a_i < a_n$ **then return** $LRMinRec(a_1, \ldots, a_{n-1})$
4. **return** $1 + LRMinRec(a_1, \ldots, a_{n-1})$

a. (4 points) For a fixed n, a best-case input to this algorithm is an input list of size n where the number of comparisons (involving list elements) is as small as possible. How many comparisons does *LRMinRec* do on a best-case input of size n? Give your answer as a closed-form (non-recursive) formula in terms of n.

Solution. $n - 1$.

b. (3 points) For a fixed n, a worst-case input to this algorithm is an input list of size n where the number of comparisons (involving list elements) is as large as possible. Find the recurrence for the exact number of comparisons done by *LRMinRec* on an input of size n, in the worst case. Do not solve the recursion.

Solution. $T(n) = T(n-1) + (n-1)$. Base case: $T(1) = 0$.

c. (3 points) Give the worst-case runtime of *LRMinRec* in Θ notation.

Solution. $\Theta(n^2)$.

5. Best and Worst Case for Sorting Algorithm

Below is the pseudocode for a revised BubbleSort algorithm. Here, the **break** command on line 7 will terminate the execution of the loop in which it occurs.

procedure *RevisedBubbleSort* ($a_1, \ldots, a_n$: list of integers)

1. **for** $i := 1$ **to** $n - 1$
2.   *done* := true
3.   **for** $j := 1$ **to** $n - i$
4.   **if** $a_j > a_{j+1}$ **then**
5.   Interchange a_j and a_{j+1}
6.   *done* := false
7.   **if** (*done* == true) **then break**

a. (5 points) For the input $(1, 2, 3, 4, 5, 6)$, the algorithm *RevisedBubbleSort* does **5** comparisons. Give a different input with the same entries for which *RevisedBubbleSort* also does 5 comparisons, or explain why no such input exists.

Solution. There is no other case.

After one iteration of the **for** loop, we have used exactly 5 comparisons between list elements. Thus, in order or the algorithm to do exactly 5 comparison, it must terminate after the first iteration which means *done* must be *true* at the end of the iteration. Observe that once we set *done* to $false$, we cannot turn it back to *true* within the same iteration. Hence, it must be the case that *done* is never set to $false$ within the first iteration. Therefore, it must be the case that the input is an increasing sequence.

b. (5 points) For the input $(6, 5, 4, 3, 2, 1)$, the algorithm RevisedBubbleSort does **15** comparisons. Give a different input with the same entries for which RevisedBubbleSort also does 15 comparisons, or explain why no such input exists.

Solution. There are many answer. One possible answer is $(5, 6, 4, 3, 2, 1)$