CSC 212: Data Structures and Abstractions
University of Rhode Island
Problem Set #1 (Due on Feb 11th at 12p)

This problem set is worth a total of 100 points. Each ⭐ represents 5 points and each ♦ represents 10 points. Partial credit will be given when appropriate, provided that you show your work and not only the final solution. You must turn in your own write-up. Please clearly describe your answers in the order they are given. Solutions should be turned in using your account on Gradescope in PDF form.

1. ⭐ Algorithm algo1 uses $10 \cdot n \log n$ operations, while algorithm algo2 uses $n^2$ operations. What is the value of $n_0$, such that algo1 is better than algo2 for $n \geq n_0$?

2. ♦ Rank the following functions by their asymptotic growth rate in ascending order. In your solution, group those functions that are Big Theta of one another (all log functions are base 2):

   $\begin{align*}
   6 \cdot n \log n & \quad 2^{100} \quad \log \log n \quad \log^2 n \quad 2^{\log n} \\
   2^n & \quad \left\lceil \sqrt{n} \right\rceil \quad n^{0.01} \quad 1/n \quad 4n^{3/2} \\
   3n^{0.5} & \quad 5n \quad \left\lfloor 2 \cdot n \log^2 n \right\rfloor \quad 2^n \quad n^2 \\
   4^n & \quad n^3 \quad n^2 \log n \quad 4^{\log n} \quad \sqrt{\log n}
   \end{align*}$

3. ⭐ Suppose you are given an algorithm A to find an element $e$ in an $n \times n$ matrix $M$. The algorithm $A$ iterates over the rows of $M$ and calls another algorithm $B$ on each one, until $e$ is found or it has searched all rows of $M$. What is the worst-case running time of $A$ in terms of $n$? (B performs a linear search in $O(n)$ time)

4. ♦⭐ For each of the following pieces of code, give a Big O characterization in terms of $n$:

   (a) $s = 0$
   for $i = 1$ to $n$
   $s = s + 1$

   (b) $s = 1$
   for $i = 1$ to $2n$
   $s = s \cdot i$

   (c) $s = 1$
   for $i = 1$ to $n^2$
   $s = s \cdot i$

   (d) $s = 0$
   for $i = 1$ to $2n$
   for $j = 1$ to $i$
   $s = s + 1$
(c) 
\[
\begin{align*}
s &= 0 \\
\text{for } i &= 1 \text{ to } n \cdot n \\
&\text{for } j &= 1 \text{ to } i \\
&\text{do } s = s + i
\end{align*}
\]

5. * Show that \((n + 1)^5\) is \(O(n^5)\).

6. * Show that \(2^{n+1}\) is \(O(2^n)\).

7. * Show that \(n^3 \log n\) is \(\Omega(n^2)\).

8. ◊ Suppose you run two algorithms, \(P\) and \(Q\), on many randomly generated data sets. \(P\) is an \(O(n \log n)\)-time algorithm and \(Q\) is an \(O(n^2)\)-time algorithm. After your experiments you find that if \(n < 100\), \(Q\) actually runs faster, and only when \(n \geq 100\), \(P\) is faster. Explain why this scenario is possible, including numerical examples.

9. ◊ Describe a method for finding both the maximum and minimum of \(n\) numbers using fewer than \(3n/2\) comparisons.

10. ◊ An array \(A\) contains \(n - 1\) unique integers in the range \([0, n - 1]\); that is, there is one number from this range that is not in \(A\). Describe an \(O(n)\) time algorithm for finding that number. You are allowed to use only \(O(1)\) additional memory besides the array \(A\) itself.

11. ◊ Suppose that each row of an \(n \times n\) matrix \(M\) consists of only binary digits, such that, in any row of \(M\), all the 1’s come before any 0’s in that row. Assuming \(M\) is already in memory, describe an \(O(n)\)-time algorithm for finding the row of \(M\) that contains the most 1’s.

12. ◊ Given an array \(A\) of \(n - 2\) unique integers in the range \([1, n]\), describe an \(O(n)\)-time algorithm for finding the two integers in the range \([1, n]\) that are not in \(A\). You are allowed to use only \(O(1)\) additional memory besides the array \(A\) itself.