COSC 240, Fall 2018: Problem Set $\#1$

Assigned: 9/12
Due: Wed., 9/19, at the beginning of class (hand in hard copy)
Lectures Covered: 2 - 4
Academic Integrity: You can work with other people in the class but you must write up your own answers in your own words. You can also use the textbook and talk to the professor. You may not use any other resources (e.g., material found online) or talk to people outside the class about these problems. See the syllabus for details on the academic integrity policy for problem sets.

Problems

1. Use a loop invariant to prove the following subroutine returns the smallest value in array A between positions i and j (assuming it is called with parameters $i < j$):

```
findMin($A, i, j$)
 $min \leftarrow i$
 $k \leftarrow i + 1$
  while $k \leq j$
    if $A[k] < A[min]$ then $min \leftarrow k$
    $k \leftarrow k + 1$
  return $min$
```

2. Use a loop invariant to prove the follow algorithm, which uses **findMin** as a subroutine, correctly sorts an array A indexed from 1 to n, for some $n > 1$:

```
SelSort($A, 1, n$):
 $i \leftarrow 1$
  while $i \leq n$
    $m \leftarrow$ findMin($A, i, n$)
    swap values in $A[i]$ and $A[m]$
    $i \leftarrow i + 1$
```

3. Prove the following statements using the formal definitions of asymptotic notation we learned in class (i.e., actually identify positive constants c and k for which the stated claims hold):

   (a) $5\sqrt{n} = O(n)$
   (b) $(1/2)n^2 = \Omega(n^2 + 10n + 8)$
   (c) $n^3 + n^2 + n = \Theta(n^3)$

4. Draw a recursion tree for the following recurrence (you can assume that n is a power of 3), and label it with the following: (a) the number of levels; (b) number of nodes per level; (c) cost per non-leaf level; and (d) cost of leaf level.

$$T(n) = \begin{cases} 2T(n/3) + n & \text{if } n \geq 3 \\ 1 & \text{if } n < 3 \end{cases}$$

5. Using your answers from the previous problem, find a big-Θ bound for the above recurrence and argue why it is correct. You may find it useful in your argument to use the infinite decreasing geometric series bound, which says that for $0 < r < 1$: $\sum_{i=0}^{\infty} r^k \leq \frac{1}{1-r}$.

   (Recall: to show $T(n) = \Theta(f(n))$, for some $f(n)$, you must show both that $T(n) = O(f(n))$ *and* that $T(n) = \Omega(f(n))$.)

6. Now use the master theorem to bound the recurrence $T(n) = 2T(n/3) + n$.

   (If you end up applying Case 3, make sure you also show the regularity condition; i.e., that there exists a constant $c < 1$ such that $af(n/b) \leq cf(n)$.)

7. Calculate a big-Θ bound on the average case complexity for the following algorithm under the assumption that the input A is an array containing the integer values $\{1, 2, ..., n\}$, for some $n > 1$, that are randomly permuted with uniform randomness (i.e., every ordering of the values is equally possible). To receive full credit you must show the relevant expectation calculation.

```
**FindOne**($A$)
 $i \leftarrow 1$
 **while** $A[i] \neq 1$
   $i \leftarrow i + 1$
 **return** $i$
```

8. Calculate a big-Θ bound on the randomized step complexity for the following random algorithm. Assume the input A to the algorithm is an array that holds the integers $\{1, 2, ..., n\}$ in some arbitrary order. To receive full credit you must show the relevant expectation calculation.

```
**FindLargeHalf**($A$)
 $i \leftarrow \mathbf{random - integer}(1, n)$
 **while** $A[i] \leq n/2$
   $i \leftarrow \mathbf{random - integer}(1, n)$
 **return** $i$
```