ECE 606, Fall 2021, Assignment 3
Due: Thursday, September 30, 11:59pm

Submission: your solutions for the written problems to crowdmark. There are no **[python3]** problems in this assignment.

1. Consider IterativeInsertionSort from Lecture 3 of your textbook.

   (a) Alice says that a count of the number of times Line (6), "$A[k] \leftarrow A[k-1]$," is executed is a meaningful measure of the time-efficiency of the algorithm. Do you concur with Alice? Why or why not?

   (b) Notwithstanding your answer to (a) above, Alice observes that Line (6) is within three nested **foreach** loops: the ones in Lines (1), (2) and (5). As that as justification, she asserts that the number of times Line (6) is executed in the worst-case is $\Theta(n^3)$. Do you agree? Why or why not?

   (c) Suppose the number of items n in any array A that is input to IterativeInsertionSort is bounded by constants. That is, we know that it is always the case that, for example, $1 \leq n \leq 10^6$. However, each entry in the input array is unbounded, e.g., if we restrict the entries in A to be positive integers only, an entry may be any finite positive integer.

   Under this assumption, suppose the maximum size to encode any of the entries in the input array A is s. What is a meaningful characterization of the worst-case time-efficiency of IterativeInsertionSort as a function of s in $\Theta(\cdot)$ notation?

2. This problem refers to BinSearch from Lecture 3 of your textbook. Consider the following two alternative versions.

   BinSearchRecursive(A, lo, hi, i)

   1 **if** $lo \leq hi$ **then**
   2 $\quad mid \leftarrow \left\lfloor \frac{lo+hi}{2} \right\rfloor$
   3 $\quad$ **if** $A[mid] = i$ **then return** true
   4 $\quad$ **if** $A[mid] < i$ **then return** BinSearchRecursive($A, mid+1, hi, i$)
   5 $\quad$ **else return** BinSearchRecursive($A, lo, mid-1, i$)
   6 **else return** false

   BinSearchCeil(A, lo, hi, i)

   11 **while** $lo \leq hi$ **do**
   12 $\quad mid \leftarrow \left\lceil \frac{lo+hi}{2} \right\rceil$
   13 $\quad$ **if** $A[mid] = i$ **then return** true
   14 $\quad$ **if** $A[mid] < i$ **then** $lo \leftarrow mid + 1$
   15 $\quad$ **else** $hi \leftarrow mid - 1$
   16 **return** false

   BinSearchRecursive is a recursive version of BinSearch in Lecture 3 of your textbook, and BinSearchCeil changes Line (2) of BinSearch to use the ceiling instead of the floor.

(a) Is the worst-case time-efficiency in $\Theta(\cdot)$ notation of BinSearchRecursive the same, worse or better than BinSearch? Give a brief, but precise and technical justification.

(b) Is the worst-case space-efficiency in $\Theta(\cdot)$ notation of BinSearchRecursive the same, worse or better than BinSearch? Give a brief, but precise and technical justification.

(c) Prove that BinSearchCeil possesses the termination property.

(d) Suppose we want to adapt the proof for Claim 35 in Lecture 3 of your textbook to BinSearchCeil. Note that nothing changes for Case 1 because if $lo + hi$ is even, then $\left\lceil \frac{lo+hi}{2} \right\rceil = \left\lfloor \frac{lo+hi}{2} \right\rfloor$. Redo the proof for Case 2 for BinSearchCeil.

3. Consider the problem that we are given as input (i) an array $A[1, \ldots, n]$ of all distinct items, and, (ii) an item i that is guaranteed to be one of the items in the array, i.e., we know that $i \in \{A[1], A[2], \ldots, A[n]\}$. We want an algorithm that identifies and returns the index j in the array A whose entry is i.

Consider the following two randomized algorithms for this problem, RandFindIndex$_1$ and RandFindIndex$_2$.

```
RandFindIndex_1(A[1, ..., n], i)
ret ← 0
while ret = 0 do
    Uniformly pick j ∈ {1, ..., n}
    if i = A[j] then  ret ← j
return ret
```

```
RandFindIndex_2(A[1, ..., n], i)
ret ← 0
doneSet ← ∅
while ret = 0 do
    Uniformly pick j ∈ {1, ..., n} \ doneSet
    if i = A[j] then  ret ← j
    else doneSet ← doneSet ∪ {j}
return ret
```

(a) Life is all about trade-offs. That is, there is rarely one option that is unqualifiedly better than another. In this spirit state (i) one thing that is better with algorithm RandFindIndex$_1$ over RandFindIndex$_2$, and, (ii) one thing that is better with RandFindIndex$_2$ over RandFindIndex$_1$.

(b) What, in the expected-case, is the number of iterations of the **while** loop in RandFindIndex$_1$? (Your solution will presumably be a function of n.)

(c) What, in the expected-case, is the number of iterations of the **while** loop in RandFindIndex$_2$? (Your solution will presumably be a function of n.)