Analysis of Algorithms
CS 375, Spring 2019
Homework 5
Due **AT THE BEGINNING OF CLASS** Monday, March 4

- From your textbook (CLRS), please be sure to read Chapter 3, and begin reading Chapter 4—we'll start Ch. 4 in our next class meeting.
- *A general note:* When writing up your homework, please write neatly and **explain your answers clearly**, giving all details needed to make your answers easy to understand. Graders may not award credit to incomplete or illegible solutions. Clear communication *is* the point, on every assignment.

Exercises

1. Prof. E. Nigma of the Portland Institute of Technology hired you to analyze the algorithm given here in pseudocode, but as usual, Prof. Nigma neglected to explain what the algorithm does.

```
// Input: A matrix A[0..n-1, 0..n-1] of integers
for i = 0 to n-2 do
    for j = i+1 to n-1 do
        if A[i,j] != A[j,i]
            return False
return True
```

   In the above, recall that a matrix is essentially just a two-dimensional array, so $A[i, j]$ might in some languages be written as $A[i][j]$.

   (a) What does this algorithm do? Give an English description of what inputs lead to it returning True and what inputs lead to it returning False. (You do not need to give examples as part of your answer, but you are welcome to include example 2D arrays along with the English description, if it would make your answer clearer.)

   (b) Give an *exact* count of the number of array accesses (count $A[x, y]$ as a single array access) done by this algorithm in the worst case on input of size n, and based on that, give the Θ complexity class for this algorithm. Be sure to explain the details behind your answer: show all work that you did to count those operations and arrive at a summation that expresses the running time; and briefly explain how you got from that summation to the Θ complexity bound.

2. In class, we discussed Bubble Sort and a loop invariant for it. (See the last slides of the Feb. 25 lecture notes.) For this exercise, show the *maintenance* part of a correctness argument using the invariant. (You do not need to show the initialization or termination parts.)

   To do this, consider the loop invariant, presented here for convenience:

   > Subarray $A[1..i-1]$ consists of the $i-1$ smallest values of A, in sorted order, and $A[i..n]$ consists of the remaining values of A (no constraint on order).

Then, to show the maintenance step, show that for the algorithm as given in lecture notes, if the invariant is true at the beginning of an iteration, then it's true at the end of the iteration / the beginning of the next iteration. That is, for any iteration m, if it's true when $i = m$, then it's true when the iteration is over and i becomes $m+1$. Use the definition of the algorithm (i.e., the pseudocode) in the explanation. (Diagrams or specific examples are not sufficient for an explanation, but if you'd like to include them along with a textual explanation, feel free do so.)

3. Consider this pseudocode algorithm for the sorting method *Selection Sort*:

```
SelectionSort(A[1 ... n])
    for i = 1 to length[A] − 1
        min = i
        for j = i + 1 to length[A]
            if A[j] < A[min]
                min = j
        // the next 3 lines swap A[i] and A[min], using a temporary variable
        temp = A[i]
        A[i] = A[min]
        A[min] = temp
```

   (a) Give an *exact* count of the number of array accesses done by this algorithm in the worst case on input of size n, and based on that, give the Θ complexity class for this algorithm. Be sure to explain the details behind your answer: show all work that you did to count those operations and arrive at a summation that expresses the running time; and briefly explain how you got from that summation to the Θ complexity bound.

   (b) Given the following proposed loop invariant for the outer for loop of SelectionSort, explain the *initialization* and *termination* steps of a correctness argument. That is, explain how the invariant is true when that loop is first initialized, but before anything in the body of the loop has executed, and explain how the truth of the invariant after the final iteration of the loop leads to the correctness of the algorithm. (You do not need to show the maintenance step for this exercise.)

   Proposed loop invariant for SelectionSort:

   > Subarray $A[1..i-1]$ contains the $i-1$ smallest elements of A in sorted order, and $A[i..n]$ consists of the remaining values of A (no constraint on order).