

Analysis of Algorithms  
CS 375, Spring 2020

Homework 9

Due **AT THE BEGINNING OF CLASS** Wednesday, March 11

- From your textbook (CLRS), please read all of Chapters 2 and 3. Although not all of it will show up on CS375 exercises, it is all good material to know.
- Please recall the guidelines for algorithms given on previous HW assignment sheets. They continue to apply to all HWs in CS375.
- In general, there may be multiple correct ways of presenting an algorithm, although excessively inefficient or inelegant solutions may not receive full credit. If you have questions about whether your proposed solution is excessively inefficient or inelegant, please ask your Prof.!
- *A general note for CS375:* 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 full credit to incomplete or illegible solutions. Clear communication *is* the point, on every assignment.

In general in CS375, unless explicitly specified otherwise, answers should be accompanied by explanations. Answers without explanations may not receive full credit. Please feel free to ask me any questions about explanations that might come up!

## Exercises

1. In class today, we discussed Insertion Sort and a loop invariant for it, and we discussed Bubble Sort and a loop invariant for it. (See the March 9 lecture notes.)

For this exercise, you'll extend what we did with Bubble Sort in class—you'll 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).

Recall that to show the maintenance step, you'll 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 to do so.)

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

```
SELECTIONSORT( $A[1 \dots n]$ )
  for  $i = 1$  to  $\text{length}[A] - 1$ 
    min =  $i$ 
    for  $j = i + 1$  to  $\text{length}[A]$ 
      if  $A[j] < A[\text{min}]$ 
        min =  $j$ 
    // the next 3 lines swap  $A[i]$  and  $A[\text{min}]$ , using a temporary variable
    temp =  $A[i]$ 
     $A[i] = A[\text{min}]$ 
     $A[\text{min}] = \text{temp}$ 
```

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, given the specification of the sorting problem (as given in class). 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).

These do not need to be lengthy answers—a few sentences each (i.e., some for initialization, and some for termination) could be sufficient, as long as those sentences contain the key details.