Analysis of Algorithms
CS 375, Fall 2022

Small Assignment 3

Due **BY THE BEGINNING OF CLASS** Monday, October 17

- The purpose of this Smaller Assignment is to practice explaining algorithm correctness using loop invariants. Please see the lecture notes of Oct. 12 for many related details!

  We will go over this Smaller Assignment (and those lecture notes) more fully in class on Monday. Because of that, on-time submission of this assignment is essential—please do this work to prepare for Monday's class! It's okay if you're not fully sure how to do these exercises at this point—we'll go over them soon, and in terms of your grade on this SA, see the note immediately below!

- For this Smaller Assignment, as long as you submit an answer with a demonstrable, strong effort to solve the problems and explain your answers thoroughly—including, if you were not able to solve an exercise, what progress you made, what was left unsolved, and what made it hard to solve—if you do not receive full credit, you will be able to revise your answer after receiving feedback, to bring your grade up to full credit!

- For this Smaller Assignment, the standard file naming conventions apply: Please submit your typewritten answers in a PDF file named

  `CS375_SA3_<userid>.pdf`

  where `<userid>` is replaced by your Colby userid (your full userid, including class year) and submit it to your `SubmittedWork` folder in your Google drive space for this course.

- *A general note for CS375:* As always, please present answers cleanly and **explain them clearly and thoroughly**, giving all details needed to make your answers easy to understand; typed-up (rather than handwritten) answers are especially appreciated. (Feel free to talk with your Prof. or TA's about using LaTeX to typeset your answers!) 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. Here is the pseudocode for Bubble Sort, as seen in class:

```
BubbleSort(A[1...n])
1.  for $i = 1$ to length[A] $- 1$
2.      for $j =$ length[A] downto $i + 1$
3.          if A[$j$] < A[$j - 1$]
4.              swap A[$j$] with A[$j - 1$]
```

For this exercise, you'll extend what we did with Bubble Sort in class—you'll give a correctness argument using our loop invariant. 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 the three parts of an explanation using loop invariants, presented here in slightly different wording than in the lecture notes:

(a) Give a very short and convincing explanation of how the invariant is true before the first iteration of the loop. As always, be sure to explain how every part of the invariant is true, not just one part of it.

Hint: In this case, note that before the first iteration, i is 1. What can be said about subarray $A[1..i-1]$? Recall what was said in lecture about vacuous truth!

(b) Give a concise and convincing explanation of how your pseudocode ensures the invariant is true after each successive iteration. Refer directly to the pseudocode, citing specific lines of pseudocode in your explanation.

Hint: Before the i'th iteration starts, we know that $A[1..i-1]$ contains the $i-1$ smallest values of A, in sorted order. What element of A will be in $A[i]$ when the i'th iteration is over? And how does that help you establish, when that iteration is over, that $A[1..i]$ are then the i smallest elements of A **and** in sorted order?

(c) Give a concise and convincing explanation of how the algorithm meets its specifications. Refer specifically to both the invariant property and the specifications as part of this explanation—referring to the specifications is essential for establishing algorithm correctness!

Hint: At the end of the loop, $i = n$. What elements, then, are in $A[i..n-1]$? And what does that say about the element that's then in $A[n]$, and where it belongs in sorted order? Then, how does knowing that $A[1..i-1]$ is the $i-1$ smallest values of A in sorted order fit with the specification for the sorting problem, to help you explain correctness?

Please write all three parts in your answer for this exercise. And please be sure each part of your explanation explains *both components of the invariant*—not just the part about $A[1..i-1]$ but also the part about $A[i..n]$. (It is a common error to sometimes overlook one part or the other!)

These do not need to be three lengthy answers—a few sentences each could be enough, as long as those sentences contain the key details.

Refer specifically to the pseudocode and to the specifications for the sorting problem in your explanation. Diagrams or specific examples are not sufficient on their own, but if you'd like to include them along with a textual explanation, feel free do so.

The specification for the *sorting problem* is repeated here for convenience:

- *Input*: Sequence of numbers $\langle a_1, \ldots, a_n \rangle$
- *Output*: Permutation (reordering) $\langle b_1, \ldots, b_n \rangle$ of the input sequence (perhaps leaving them unchanged) such that $b_1 \le b_2 \le \ldots \le b_n$.

This specification is the same as the one presented in lecture notes, capturing our intuitions of what it means for a list to be sorted. Please see me if there are questions about it, or about any part of the exercise!