

Analysis of Algorithms
CS 375, Fall 2019

Homework 8

Due **AT THE BEGINNING OF CLASS** Monday, October 14

- 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.
- *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. For each of the following assertions, say if they are True or False, and explain your answers by referring to the formal definitions of asymptotic complexity, as we've done in the examples in class.

(a) $\frac{n(n+1)}{2} \in O(n^3)$

(b) $\frac{n(n+1)}{2} \in O(n^2)$

(c) $\frac{n(n+1)}{2} \in \Theta(n^3)$

(d) $\frac{n(n+1)}{2} \in \Omega(n)$

2. 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 last slides of the Oct. 7 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.)

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 the worst-case complexity of 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 using the definition of Θ complexity classes, 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 SELECTION-SORT, 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).