

CSE3358 Problem Set 1
01/14/05
Due 01/21/05

Please review the Homework Honor Policy on the course webpage
<http://enr.smu.edu/~saad/courses/cse3358/homeworkpolicy.html>

Problem 1: Bubble sort

We have studied Insertion sort in class and argued that it works correctly.

```
INSERTION-SORT( $A$ )
1 for  $j \leftarrow 2$  to  $\text{length}[A]$ 
2   do  $\text{key} \leftarrow A[j]$ 
3      $i \leftarrow j - 1$ 
4     while  $i > 0$  and  $A[i] > \text{key}$ 
5       do  $A[i + 1] \leftarrow A[i]$ 
6          $i \leftarrow i - 1$ 
7      $A[i + 1] \leftarrow \text{key}$ 
```

At the beginning of each iteration of the “outer” **for** loop, which is indexed by j , the subarray consisting of elements $A[1..j - 1]$ constitute the currently sorted elements, and the elements $A[j + 1..n]$, where $n = \text{length}[A]$, correspond to the elements that are still to be considered. In fact, elements $A[1..j - 1]$ are the elements originally in positions 1 through $j - 1$, but now in sorted order. We can state this property of $A[1..j - 1]$ formally as a **loop invariant**:

LOOP INVARIANT: *At the start of each iteration of the **for** loop of lines 1-7, the subarray $A[1..j - 1]$ consists of the elements originally in $A[1..j - 1]$ but in sorted order.*

We use loop invariants to help us understand why an algorithm is correct. We must show three things about a loop invariant:

- **Initialization:** It is true prior to the first iteration of the loop
- **Maintenance:** If it is true before an iteration of the loop, it remains true before the next iteration
- **Termination:** When the loop terminates, the invariant gives us a useful property that helps show that the algorithm is correct

Let us see how these properties hold for insertion sort.

Initialization: We start by showing that the loop invariant holds before the first loop iteration, when $j = 2$. The subarray $A[1..j - 1] = A[1..2 - 1] = A[1..1] = A[1]$, consists of just the single element $A[1]$, which is in fact the original element in $A[1]$. Moreover, this subarray is sorted (trivially, of course), which shows that the loop invariant holds prior to the first iteration of the loop.

Maintenance: Next, we tackle the second property: showing that each iteration maintains the loop invariant. Informally, the body of the outer **for** loop works by moving $A[j - 1]$, $A[j - 2]$, $A[j - 3]$,

and so on by one position to the right until the proper position for $A[j]$ is found (lines 3-6), at which point the value of $A[j]$ is inserted (line 7). A more formal treatment of this property would require us to state and show a loop invariant for the inner **while** loop (but we will not do it at this point).

Termination: Finally, we examine what happens when the loop terminates. For insertion sort, the outer **for** loop ends when j exceeds n , i.e. when $j = n + 1$. Substituting $n + 1$ for j in the wording of the loop invariant, we have that the subarray $A[1..n]$ consists of the elements originally in $A[1..n]$, but in sorted order. But the subarray $A[1..n]$ is the entire array! Hence, the entire array is sorted, which means that the algorithm is correct.

Consider the following algorithm known as Bubble sort. It works by repeatedly swapping adjacent elements that are out of order:

```
BUBBLE-SORT(A)
1 for  $i \leftarrow 1$  to  $\text{length}[A]$ 
2   do for  $j \leftarrow \text{length}[A]$  downto  $i + 1$ 
3     do if  $A[j] < A[j - 1]$ 
4       then exchange  $A[j] \leftrightarrow A[j - 1]$ 
```

(a) (5 points) Explain in english, very clearly, how the algorithm BUBBLE-SORT works (for this you need to understand what the code is doing and construct from it a process in your mind, one thing you could do to help is try it on some small examples, say 6 numbers, and see what it is doing).

(b) (5 points) Let A' denote the output of BUBBLE-SORT(A). To prove that BUBBLE-SORT is correct, we need to prove that **(1)** it terminates and that **(2)** $A'[1] \leq A'[2] \leq \dots \leq A'[n]$, where $n = \text{length}[A]$. What else must we prove to show that BUBBLE-SORT actually sorts?

(c) (10 points) Prove that the following loop invariant holds for the **for** loop in lines 2-4.

LOOP INVARIANT: *At the start of each iteration of the **for** loop of lines 2-4, the smallest element in the subarray $A[i..n]$ is in the subarray $A[i..j]$.*

Your proof should use the structure of the loop invariant proof presented above.

(d) (10 points) Using the termination condition of the loop invariant proved in part (b), state a loop invariant for the **for** loop in lines 1-4 that will allow you to prove $A'[1] \leq A'[2] \leq \dots \leq A'[n]$. Your proof should use the structure of the loop invariant presented above.

(e) (5 points) What is the worst-case running time of BUBBLE-SORT? Use Θ notation as we did in class. How does it compare to the running time of insertion sort?

(f) (5 points) Which one you think is faster in practice, INSERTION-SORT or BUBBLE-SORT, and why?

(g) (10 points) Implement both in any programming language of your choice. Try them on randomly generated inputs of size 10, 20, 50, 100, 500, 1000. Report their real running times as measured by the system clock. Does the numbers agree with your intuition on part (f)?