1 Bubble Sort: Proof Of Correctness

The pseudocode for the algorithm is as follows:

\[\text{Bubble-Sort}(A[1..n]):\]
\[\text{for } i = 1 \text{ to } n - 1 \text{ do}\]
\[\quad \text{for } j = 1 \text{ to } n - i \text{ do}\]

For the sake of simplicity, we assume that all the elements of \(A[]\) are distinct. Below we prove that the Bubble Sort algorithm correctly sorts all input arrays.

To do so, we establish a loop invariant:

For any \(i (0 \leq i \leq n - 1)\), after \(i\) iterations of the outer for-loop, the last \(i\) elements of \(A[1..n]\) are the largest in \(A[1..n]\) and are already in the increasing order (see Lemma 2). Notice that if after \(n - 1\) iterations the last \(n - 1\) elements are the largest in \(A[1..n]\) and already sorted, then the first element is the smallest and thus all \(n\) elements of the array are sorted.

In order to prove the above invariant we first prove the following loop invariant for the inner loop:

after \(j\) iterations of the inner for-loop the largest element of \(A[1..n - i]\) is not among the first \(j\) elements (see Lemma 1).

Let \(n' = n - i + 1\); in the lemma below we focus on array \(A[1..n']\) and the inner for-loop:

\[\text{for } j = 1 \text{ to } n' - 1 \text{ do if } A[j] > A[j + 1] \text{ then swap}( A[j], A[j + 1] );\]  

\[\text{Lemma 1. } \forall j (0 \leq j \leq n' - 1) \text{ any time after } j \text{ iterations of the for-loop (1) max } A[1..n'] \text{ is not in } A[1..j].\]

Notice: there are only \(n' - 1\) iterations of the for-loop (1), so, since after all of them are completed the max \(A[1..n']\) cannot be in \(A[1..n' - 1]\), it must be in the position \(A[n']\).

\[\text{Proof. (by induction on the value of the loop variable } j)\]
\[\text{Base case: We can use a trivial \(1\) base case } j = 0.\]
\[\text{Induction Hypothesis: Assume that after } j = k - 1 \geq 0 \text{ iterations of the for-loop (1), max } A[1..n'] \text{ is not in } A[1..k].\]
\[\text{Induction Step: Prove that after } j = k \text{ iterations of the for-loop (1) max } A[1..n'] \text{ is not in } A[1..k].\]

Indeed, by Induction Hypothesis, after \(k - 1\) (and thus after \(k\)) iterations of the for-loop, max \(A[1..n']\) is not in \(A[1..k - 1]\). The \(k\)-th iteration assures that \(A[k + 1] > A[k]\), and so max \(A[1..n']\) is not in \(A[k]\) either. Furthermore, the for-loop (1) can move the max \(A[1..n']\) only to the right (and the subsequent for-loops will not even access it). And thus max \(A[1..n']\) is not in \(A[1..k]\) any time after \(k\) iterations of the inner for-loop.

\[\square\]

1 “Trivial” here is not used as a synonym for “simple” but rather in a stronger formal sense indicating that there is literally nothing to prove, i.e., in the sense of “self-evident”.

2 Just to be on the safe side you can check for yourself the case of \(j = 1\): after one iteration of the for-loop (1), \(A[1] \leq A[2]\) and can only be reduced further by the subsequent iterations can only make it smaller, thus at no time after the 1st iteration can it contain max \(A[1..n']\). Notice that this is essentially the same argument as in the Induction Step.
**Lemma 2.** \( \forall i (0 \leq i \leq n-1) \) any time after \( i \) iterations of the outer for-loop of Bubble-Sort algorithm above, \( A[n-i+1] \) is the \( i \)th largest element of \( A[1..n] \).

**Proof.** (by strong induction on the value of the loop variable \( i \))

*Base case:* \( i = 0 \) is trivially true.\(^3\)

*Induction Hypothesis:* Assume that any time after \( i = k \geq 0 \) iterations of the outer for-loop, the last \( k \) elements of \( A[1..n] \) are the largest and are in the increasing order.

*Induction Step:* Prove that any time after \( i = k+1 \) iterations of the outer for-loop, \( A[n-k] \) is the \((k+1)\)st largest element of \( A[1..n] \).\(^4\)

Indeed, (by Induction Hypothesis) after \( k+1 \) iterations the last \( k \) elements of \( A[1..n] \) are the largest. So \( \max A[1..n-k] \) is the \((k+1)\)st largest in \( A[1..n] \). The \((k+1)\)st iteration of the outer loop executes the inner loop (1) above with \( n' = n-k-1 \) and Lemma 1 guarantees that when this inner for-loop (1) completes, \( \max A[1..n-k] \) is not in \( A[1..n'] \), and thus it must be in \( A[n-k] \). \( \square \)

---

\(^3\) Just as in the Base Case of Lemma 1, you can check \( i = 1 \), and here it also turns out to be essentially the same as our Induction Step.

\(^4\) Notice that the Induction Hypothesis already assumes that the last \( k \) elements of \( A[1..n] \) are the largest and are in the increasing order. Thus, the Inductive step would prove it for the last \( k+1 \) elements.