Bubble Sort

Algorithm 1: Bubble Sort

Data: An array A[0 ... n-1] of size n
Result: Sorted array A in non decreasing order i.e
$A[0] <= A[1] <= ... <= A[n-1]$

```
1  begin
2      for i ← 0 to n − 2 do
3          /* Shift the maximum element of sub-array A[0..n-i-1]
              to A[n-i-1] by adjacent pairwise swapping        */
4          for j ← 0 to n − i − 2 do
5              if A[j] > A[j + 1] then
6                  /* Swap A[j] and A[j+1]                     */
7                  tmp ← A[j];
8                  A[j] ← A[j + 1];
9                  A[j + 1] ← tmp;
10             end
11             j ← j + 1;
12         end
13         i ← i + 1;
14     end
15 end
```

1 Proof of Correctness

1.1 Invariant

At the beginning of each outer for loop, the sub array $A[n-i \,..\, n-1]$ consists of the first $(n-1)-(n-i)+1 = i$ largest elements of the entire array A, in sorted order i.e $A[n-i] <= A[n-i+1] <= \,..\, <= A[n-1]$.

1.2 Initialization

Initially, $i = 0$ and hence the sub array $A[n \,..\, n-1]$ is an empty list and consists of 0 elements. As the list is empty, the invariant trivially holds.

1.3 Maintenance

The inner for loop runs from $j \leftarrow 0$ through $n-i-2$. By performing adjacent comparisons and swaps between $A[j]$ and $A[j+1]$, we ensure that the the maximum element of sub array $A[0 \,..\, n-i-1]$ is shifted and placed at $A[n-i-1]$ giving us the first $(n-1)-(n-i-1)+1 = i+1$ largest elements in $A[n-i-1 \,..\, n-1]$, that too in sorted order. Incrementing i to $i+1$ then makes the invariant hold at the start of the next iteration.

1.4 Termination

The procedure terminates when $i > n - 2$. As i is always incremented by 1, when the outer while loop terminates i will always be equal to $n-1$. According to the invariant, when $i = n - 1$, the sub array $A[1 \,..\, n - 1]$ must contain the first $(n - 1)$ largest elements of A in sorted order. By elimination, $A[0]$ must now contain the n^{th} largest element (the smallest element). In other words, the array A is now sorted. ■