Selection Sort

Algorithm 1: Selection 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          min ← i;
4          /* Find minimum element of sub-array A[i..n-1]      */
5          for j ← i + 1 to n − 1 do
6              if A[min] > A[j] then
7                  min ← j;
8              end
9              j ← j + 1;
10         end
11         /* Swap A[i] and A[min]                              */
12         tmp ← A[min];
13         A[min] ← A[i];
14         A[i] ← tmp;
15         i ← i + 1;
16     end
17 end
```

1 Proof of Correctness

1.1 Invariant

At the beginning of each outer for loop, the sub array $A[0 .. i-1]$ consists of the first i smallest elements of the entire array A, in sorted order.

1.2 Initialization

Initially, $i = 0$ and hence the sub array $A[0 .. i-1]$ 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 $j \leftarrow i + 1$ through $n - 1$. It is not too difficult to see that when the inner for loop terminates, the variable min contains the index of the smallest element in the sub array $A[i .. n-1]$. $A[min]$ is then swapped with $A[i]$, thereby giving us the first $i + 1$ smallest elements in $A[0 .. i]$, that too in sorted order. Incrementing i 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 $A[0 \,..\, n - 2]$ must contain the first $(n - 1)$ smallest elements of A in sorted order. By elimination, $A[n - 1]$ must now contain the n^{th} smallest element. In other words, the array A is now sorted. ■