SELECTION SORT CORRECTNESS

Loop Invariant

At the start of each iteration of the outer loop (for each i), the subarray arr[0:i-1] is already sorted and contains the smallest elements in the correct order. The rest of the array, arr[i:n-1], is unsorted. By the end of the iteration, the smallest element in the unsorted portion is placed at index i.

Correctness Proof with Example

Input Example:
arr = [8, 3, 7, 1, 4]

1. Initialization

Before the first iteration, i = 0. The subarray arr[0:-1] (no elements) is trivially sorted. The entire array is unsorted (arr[0:n-1] = [8, 3, 7, 1, 4]).

The algorithm identifies the smallest element (1) in the unsorted portion and swaps it with arr[0].

After the first iteration:
Sorted portion: arr[0:1] = [1]
Unsorted portion: arr[1:n-1] = [3, 7, 8, 4]
Array: [1, 3, 7, 8, 4]

The invariant holds.

2. Progression

After the i-th iteration, the smallest element in the unsorted portion is placed at index i. The subarray arr[0:i] remains sorted, and the unsorted portion becomes arr[i+1:n-1].

Iteration 2 (i = 1)
Sorted portion: arr[0:1] = [1]
Unsorted portion: arr[1:n-1] = [3, 7, 8, 4].

The smallest element (3) is already at index 1, so no swap is needed.

After the second iteration:
Sorted portion: arr[0:2] = [1, 3]
Unsorted portion: arr[2:n-1] = [7, 8, 4]
Array: [1, 3, 7, 8, 4]

The invariant holds.

Iteration 3 (i = 2)
Sorted portion: arr[0:2] = [1, 3]
Unsorted portion: arr[2:n-1] = [7, 8, 4].

The algorithm identifies the smallest element (4) in the unsorted portion and swaps it with arr[2].

After the third iteration:
Sorted portion: arr[0:3] = [1, 3, 4]
Unsorted portion: arr[3:n-1] = [8, 7]
Array: [1, 3, 4, 8, 7]

The invariant holds.

Iteration 4 (i = 3)
Sorted portion: arr[0:3] = [1, 3, 4]
Unsorted portion: arr[3:n-1] = [8, 7].

The algorithm identifies the smallest element (7) in the unsorted portion and swaps it with arr[3].

After the fourth iteration:
Sorted portion: arr[0:4] = [1, 3, 4, 7]
Unsorted portion: arr[4:n-1] = [8]
Array: [1, 3, 4, 7, 8]

The invariant holds

3. Termination

The loop terminates when i = n-1. At this point, the entire array is processed.
The sorted portion arr[0:n-1] contains all elements, and the unsorted portion is empty.

Final iteration (i = 4):
Sorted portion: arr[0:4] = [1, 3, 4, 7]
Unsorted portion: arr[4:4] = [8].

Array after the last iteration: [1, 3, 4, 7, 8].

The entire array is sorted, and the invariant holds.

Conclusion

The correctness of Selection Sort is demonstrated through:

- Initialization: The invariant holds at the beginning.

- Progression: The invariant holds after each iteration, with the smallest element in the unsorted portion moved to the correct position.
- Termination: When the loop ends, the entire array is sorted.