**Argument of the selection sort:**

The selection sort method is an easy to understand way of sorting data as it follows a systematic approach, in arranging the elements of an array.

**Correctness Argument**:

\*\*1. **Invariant of Sorted Portion**:

**Definition**: An invariant is a property that remains true throughout the execution of an algorithm.

**Selection Sort**: At the beginning of each iteration of the outer loop, the part of the array before the position we’re currently on is sorted. At the beginning of the first iteration, that just means nothing is sorted yet. But each time we pass an item past the current position of most Size, it means that part of the array is sorted.

\*\*2. **Selection of Minimum Element**:

During each iteration, the algorithm selects the smallest element from the unsorted portion.

That makes sure the selected element is the smallest of all those unselected elements so far, and maintains this sortedness in the already sorted part of the list.

\*\*3. **Correctness of Swapping**:

After selecting the smallest element, swapping it with the element at the current position ensures that the element at the current position is the smallest in the unsorted portion.

This placement maintains the invariant that the left portion of the array (up to the current position) is sorted.

\*\*4. **Termination**:

The algorithm terminates when all positions have been processed. By this point, the entire array is sorted, since the invariant holds at every step, and the unsorted part eventually becomes empty.

\*\*5. **Formal Proof**:

**Initialization**: At the start, the sorted portion is empty, which is trivially sorted.

**Maintenance**: Each iteration places the smallest element of the part not yet sorted in the correct position, and after each iteration maintains the invariant that the part before the current position is sorted.

**Termination**: If all of the array elements are in the correct positions, then the algorithm is finished.