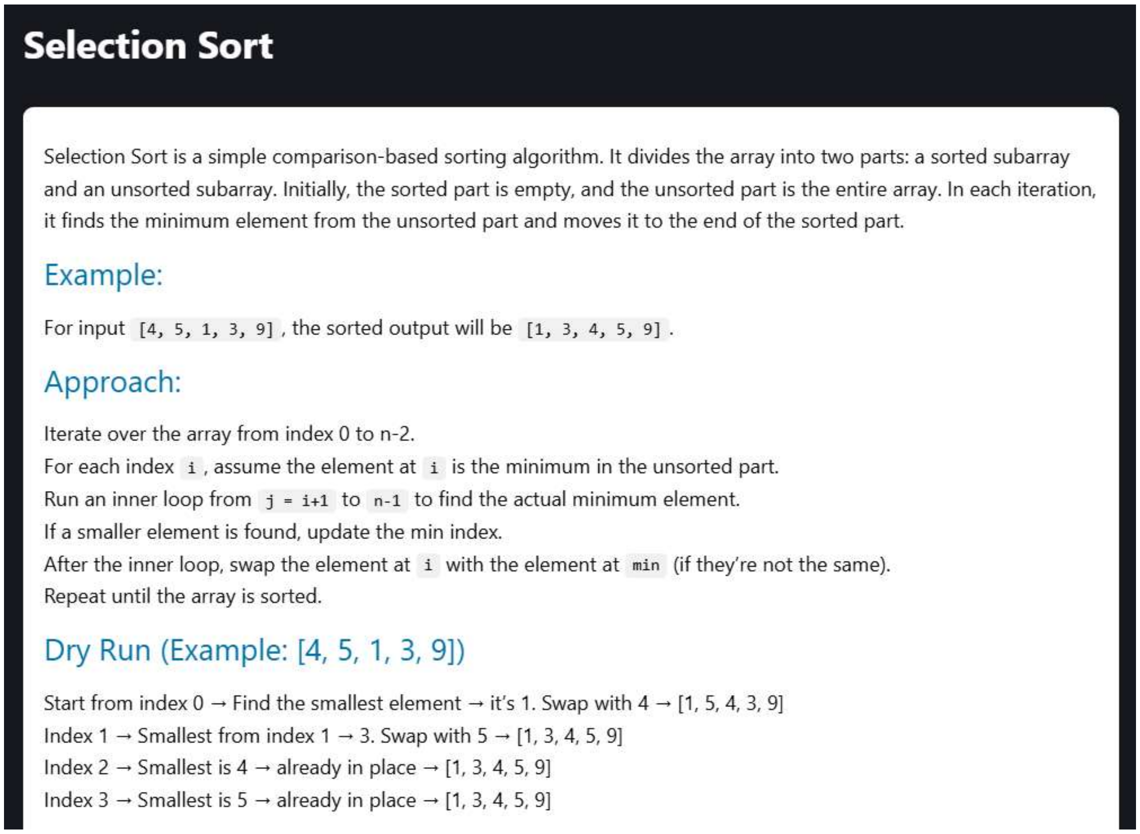


Selection Sort

Selection Sort is a simple comparison-based sorting algorithm. It divides the array into two parts: a sorted subarray and an unsorted subarray. Initially, the sorted part is empty, and the unsorted part is the entire array. In each iteration, it finds the minimum element from the unsorted part and moves it to the end of the sorted part.

Example:

For input `[4, 5, 1, 3, 9]`, the sorted output will be `[1, 3, 4, 5, 9]`.

Approach:

Iterate over the array from index 0 to n-2.
For each index `i`, assume the element at `i` is the minimum in the unsorted part.
Run an inner loop from `j = i+1` to `n-1` to find the actual minimum element.
If a smaller element is found, update the min index.
After the inner loop, swap the element at `i` with the element at `min` (if they're not the same).
Repeat until the array is sorted.

Dry Run (Example: [4, 5, 1, 3, 9])

Start from index 0 → Find the smallest element → it's 1. Swap with 4 → [1, 5, 4, 3, 9]
Index 1 → Smallest from index 1 → 3. Swap with 5 → [1, 3, 4, 5, 9]
Index 2 → Smallest is 4 → already in place → [1, 3, 4, 5, 9]
Index 3 → Smallest is 5 → already in place → [1, 3, 4, 5, 9]

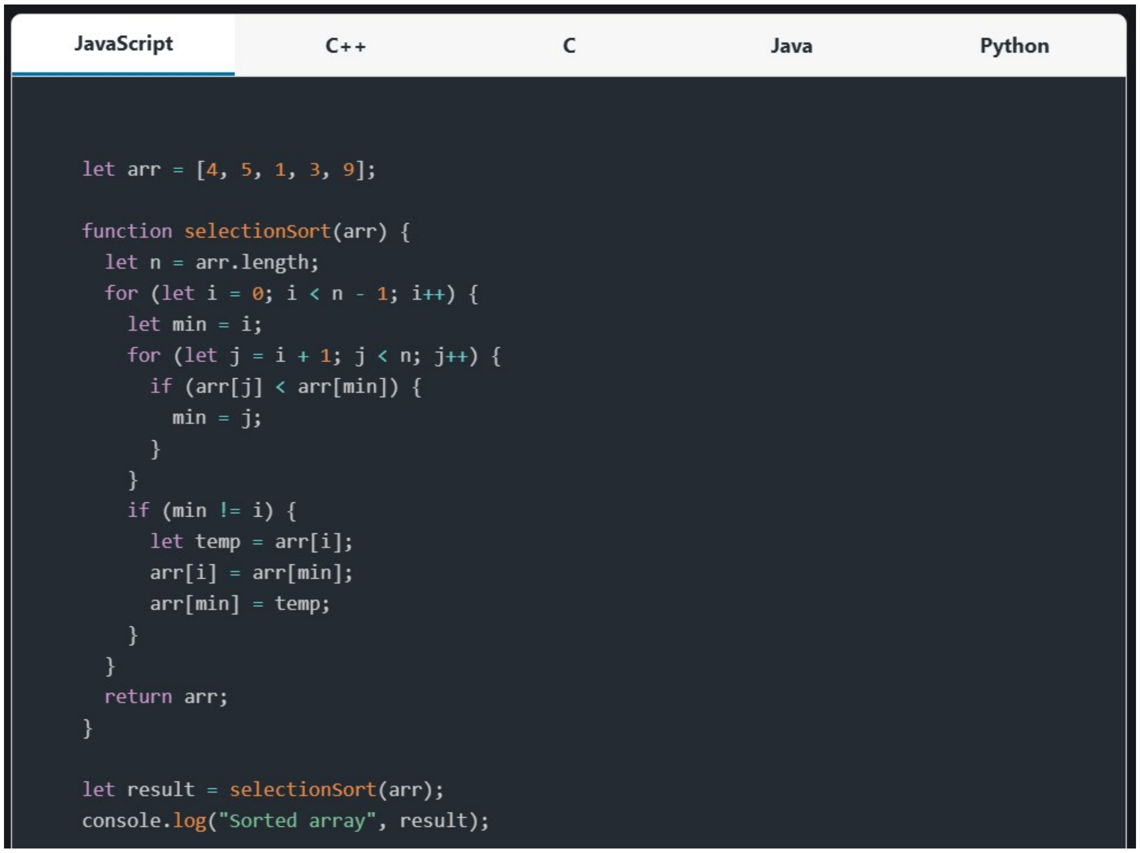
Time Complexity (TC):

O(n^2) in all cases — best, average, and worst.
Roughly n*(n-1)/2 comparisons are always performed.

Time Complexity (TC):

O(1)

Selection Sort is an in-place sorting algorithm, so it doesn't require extra space.



```javascript
let arr = [4, 5, 1, 3, 9];

function selectionSort(arr) {
  let n = arr.length;
  for (let i = 0; i < n - 1; i++) {
    let min = i;
    for (let j = i + 1; j < n; j++) {
      if (arr[j] < arr[min]) {
        min = j;
      }
    }
    if (min != i) {
      let temp = arr[i];
      arr[i] = arr[min];
      arr[min] = temp;
    }
  }
  return arr;
}

let result = selectionSort(arr);
console.log("Sorted array", result);
```