# Selection Sort Algorithm

**Mehmet ÖZER ozermehmet911@gmail.com**

Selection Sort algorithm is a simple and understandable sorting method used to sort elements of an array from smallest to largest. Its basic working principle is based on placing the smallest element of each section sequentially at each position in the array.

The algorithm starts from the first element of an unsorted array, finds the smallest element in the remaining part of the array, and swaps it with the current element. It repeats this process until the end of the array. In each iteration, a part of the array becomes sorted, and in the next step, that part is no longer considered.

The algorithm's performance is independent of the initial state of the array. Even if the array is already sorted, it will still perform the same number of comparisons. This is one of Selection Sort's disadvantages. In contrast, it is quite efficient in terms of memory usage, as it performs an element exchange only once each time.

---

## Algorithm's Working Logic

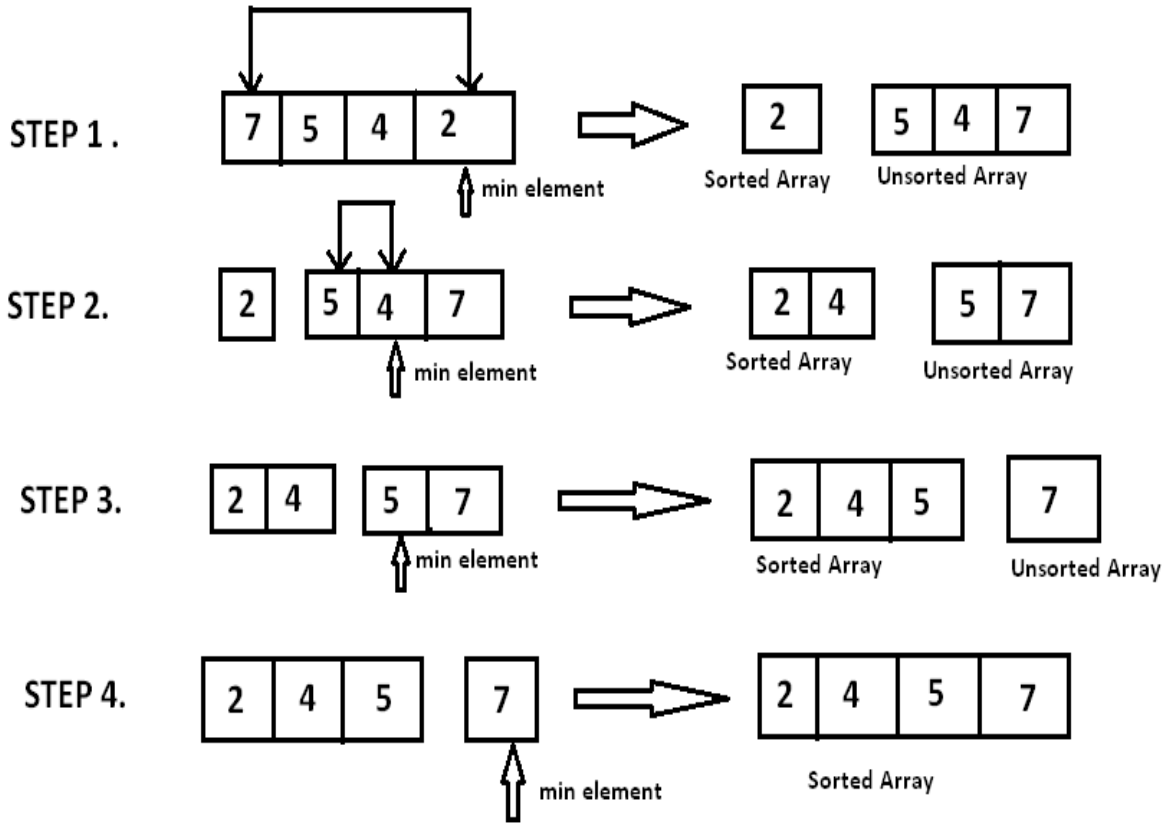
1. The first element of the array is considered the smallest element.
2. A value smaller than this element is searched in the remaining part of the array.
3. When a smaller element is found, this element is swapped with the element at the current position.
4. The process is repeated for each element of the array, and a part of the array is sorted at the end of each iteration.

At the end of each loop, the size of the sorted part increases by one element, and the size of the unsorted part decreases.

---

## Algorithm's Advantages and Disadvantages

Selection Sort is an efficient sorting algorithm for small data sets. Due to its time complexity of $O(n^2)$, its performance is poor for large-sized arrays. The algorithm's most important feature is that it always performs the same number of iterations and comparisons, regardless of the initial state of the array. This is both an advantage and a disadvantage of the algorithm. It is memory-efficient and exhibits an in-place algorithm characteristic because it performs element exchange at a minimum number of times.




## Implementation of Selection Sort Algorithm in Selected Languages

### *Haskell*

```haskell
selectionSort :: (Ord a) => [a] -> [a]
selectionSort [] = []
selectionSort xs = minElement : selectionSort rest
  where
    minElement = findMin xs
    rest = remove minElement xs

findMin :: (Ord a) => [a] -> a
findMin [x] = x
findMin (x:xs) = min x (findMin xs)

remove :: Eq a => a -> [a] -> [a]
remove _ [] = []
remove e (x:xs)
  | e == x    = xs
  | otherwise = x : remove e xs
```

***C++***

```cpp
void selectionSort(int arr[], int n) {
    for (int i = 0; i < n - 1; ++i) {
        int minIdx = i;
        for (int j = i + 1; j < n; ++j) {
            if (arr[j] < arr[minIdx]) {
                minIdx = j;
            }
        }
        swap(arr[i], arr[minIdx]);
    }
}
```

***Python***

```python
def selection_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(i + 1, n):
            if arr[j] < arr[i]:
                arr[i], arr[j] = arr[j], arr[i]
```

**Performance Comparison of Selection Sort Algorithm in Different Programming Languages: Haskell, C++ and Python**

| **Programlama Dili** | **İşletim Sistemi** | **Donanım** | **Süre** | **Veri Seti** |
|---|---|---|---|---|
| Haskell | Linux version 6.8.0-49-generic | Intel i5 12500h | 0.000023s | 10^4 |
| C++ | Linux version 6.8.0-49-generic | Intel i5 12500h | 0.036s | 10^4 |
| Python 3.12.3 | Linux version 6.8.0-49-generic | Intel i5 12500h | 1.46s | 10^4 |