**Selection Sort**

**Definition**

Selection sort is a sorting algorithm based on comparing elements. It divides an array into two parts: the sorted section, which is usually at the beginning, and the unsorted section, which is usually at the end. Selection sort repeatedly finds the smallest element in the unsorted section, swaps with the first element of the unsorted section, extending the sorted section by one element, until all elements are sorted.

**Benefits**

Selection sort is a simple and easy-to-use sorting algorithm, making it an appealing choice for beginner programmers. It is also a stable and in-place sorting algorithm and doesn’t require another data structure to be made while sorting, which preserves memory space. Because it compares every element, selection sort’s performance is heavily based on how many elements are sorted, making it an efficient way to sort smaller arrays.

**Limitations**

However, along with the benefits, there are disadvantages as well. Perhaps the largest one is selection sort’s inefficiency while sorting arrays that contain large amounts of data. To sort an array that contains n number of elements, selection sort needs to scan n elements and take n-1 comparisons to find the lowest element. Finding the second lowest element requires scanning n-1 elements and comparing n-2 elements, and this repeats until all elements are sorted. The total number of comparisons is (n-1)+(n-2)+(n-3)+...+1. If we use arithmetic progression, this equals to 0.5n(n-1)=0.5(n^2-n), which means it has complexity of O(n^2).

Because it requires n square number of steps to arrange n elements, it takes a considerable amount of time to sort a large array.

**Comparisons**

Selection sort is best compared to other algorithms whose approaches are comparison-based, such as bubble sort and insertion sort. They all share a similar approach and time complexity, and they typically average a speed of O(n2), where n is the number of elements. Another thing to note that like several other algorithms, selection sort is in-place, meaning that it doesn’t require auxiliary data structure. However, selection sort is a lot less efficient at sorting arrays than other algorithms, such as quick sort and merge sort.

**Tips:**

Selection sort always finds the smallest/largest element in the unsorted part of the array first, swapping it with the first element in the unsorted section.

Selection sort always uses two nested loops. The outer loop expands the boundary of the sorted section, and the inner loop scans the unsorted section for the next smallest/largest element.

In selection sort, the initial counter value of the inner loop is always greater than the initial counter value of the outer loop by one. This separates the sorted part and unsorted part of the array.