Sorting Algorithms

Algorithm	**Strategy**
Bubble Sort	Starting at index 0, compare the first two elements and sort them correctly. After, check the second and third elements and so on until the largest element is at the end. After the largest element is at the end, that index gets blocked and the program resorts through everything before the last element.
Selection Sort	In selection sort, you start by assuming the first element is the smallest and then compare it with every other element in the array. If you find a smaller element, you swap it with the first element. You keep doing this until you reach the end of the array. Then, the first element is in its correct sorted position. You repeat this process for the second element, and so on, until the entire array is sorted.
Insertion Sort	The second item is originally set as the key. If it's smaller than the item before it, the key is placed before that item. When the key has no bigger items before it, the key moves to the next item and the algorithm starts over.
QuickSelect	Quick select finds the k-th smallest number in a list. It works by splitting the list in half over and over until there's only one number left. That's the k-th smallest number.
QuickSort	Quick Sort is a sorting algorithm that works by dividing an array into smaller and smaller subarrays until each subarray is sorted. Then, the algorithm puts the subarrays back together in the correct order to sort the entire array.
Merge Sort	Merge sort works by splitting an array into smaller and smaller pieces until each piece has only one element. Then, it combines the pieces back together in sorted order.

Pros/Cons and Time Complexity

Algorithm	Advantages	Disadvantages	Time Complexity		
			Best Case	Avg. Case	Worst Case
Bubble Sort	easy to understand.	Very slow for large unsorted arrays/high time complexity.	O(N)	O(N^2)	O(N^2)
Selection Sort	Performs well on small arrays. Easy to understand.	Slow for large arrays due to high time complexity	O(N^2)	O(N^2)	O(N^2)
Insertion Sort	Low time complexity. Easy to understand.	Slow for large unsorted arrays due to high time complexities	O(N)	O(N^2)	O(N^2)
QuickSelect	Unique sorting method. Can be used for other methods.	High Time complexity Does not fully sort the array (only does part u “request”	O(n)	O(n^2)	O(n^2)
QuickSort	Low time complexity.	More complex	O(nlogn)	O(nlogn)	O(N^2)
Merge Sort	Low time complexity. Much faster.	More complex and uses helper methods.	O(nlogn)	O(nlogn)	O(nlogn)
