Chris Hoorn
Sean Cavanaugh
Baibhab Dutta
Capsule #3: Intro to Sorting Algorithms

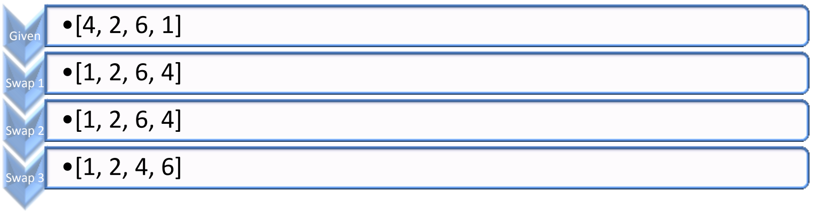
A sorting algorithm is an algorithm which arranges the elements of a collection so that they are in sorted order. There are many different sorting algorithms to choose from. They range from simple bubble sort algorithms, which are inefficient yet easy to implement, to complex, more efficient algorithms such as quicksort.

Sorting algorithms can be used to sort arrays, ArrayLists, or any other kind of collection. Certain sorting algorithms already exist and are programmed into the java library. Static methods such as Arrays.sort() sort given arrays based on either their primitive value, or, if the elements are objects, by the result of their compareTo() methods. When implementing our own sorting algorithms, it must be decided how to compare objects of the same type, whether to use the compareTo() method, or simply compare some aspect of the class. Our implementation changes when we want collections sorted for different aspects, such as size, color, or simply at random.

Two common, relatively simple sorting algorithms implementable by programmers are the selection and insertion sorts. Both algorithms sort a collection into sorted order, but accomplish the job by different methods and with slightly different efficiencies.

Selection sort involves taking an array, then starting from the beginning, searching the elements one by one for the one which should be placed at the beginning. Once it has traversed the entire list, it swaps the current first element with the element in the array that the algorithm determines should be first based on some comparator. The process then repeats starting from the second element, searching the list for the desired element, and then swapping. This continues until the starting position becomes the end of the collection and the entire collection has been sorted. The entire algorithm takes the same amount of time regardless of the state of the array before being sorted because each element is visited, regardless of whether it's already in the correct position. The Big-Oh efficiency of selection sort is O(n^2).

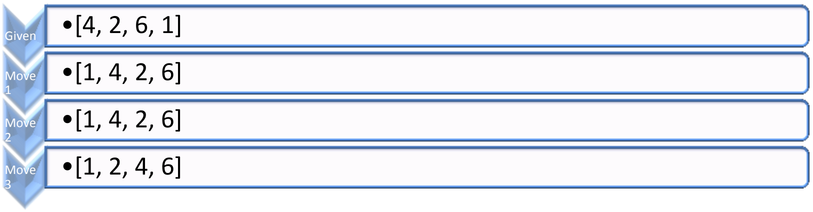
Selection Sort of an array of 4 ints.



- [4, 2, 6, 1]
- [1, 2, 6, 4]
- [1, 2, 6, 4]
- [1, 2, 4, 6]

Insertion sort involves a different approach to sorting collections. Insertion sort, as its name suggests, involves inserting elements into a collection so that they are always in sorted order. It functions by adding each element, searching to the left until it finds an element that should come before it, and moving it to that position by pushing all other elements to the right. This process repeats until all elements have been moved to their correct position. Insertion sort is also of efficiency O(n^2), but its efficiency reduces to O(n) when the given array is already sorted. This variable efficiency is due to the fact that the algorithm only accesses elements that should be placed after the inserted element. This means that if a collection is partially sorted an insertion sort may be more efficient than selection sort.

Insertion Sort of an array of 4 ints.



- [4, 2, 6, 1]
- [1, 4, 2, 6]
- [1, 4, 2, 6]
- [1, 2, 4, 6]