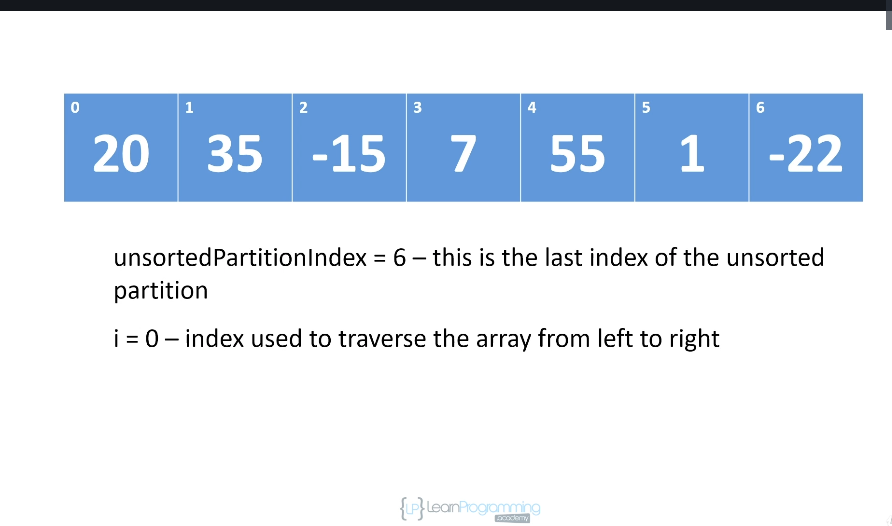
# Searching alghoritms

# Bubble sort

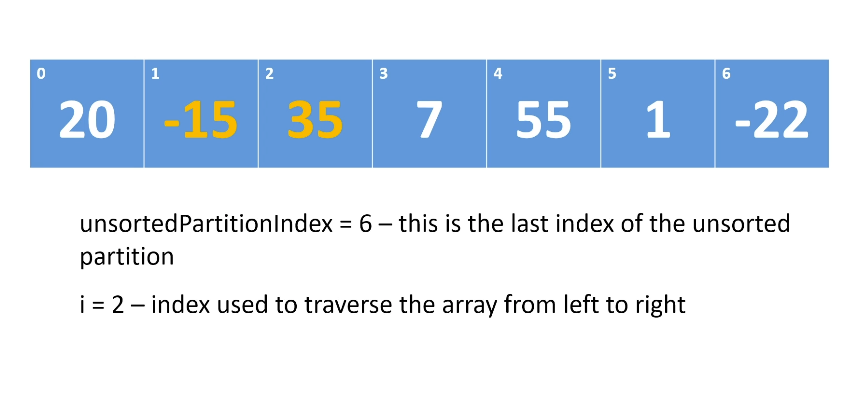


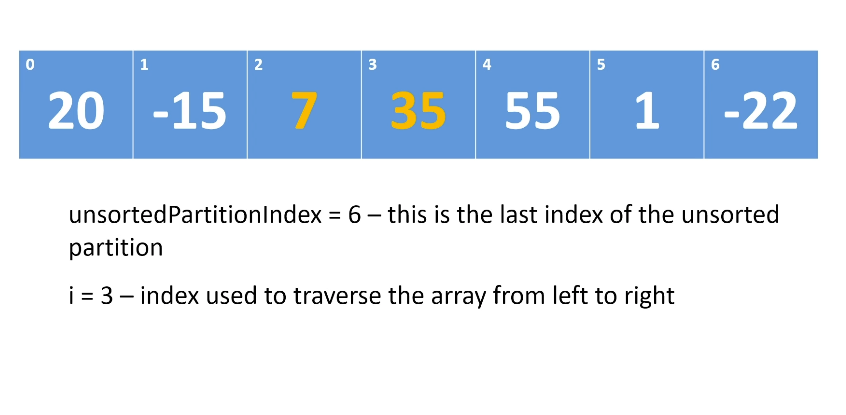
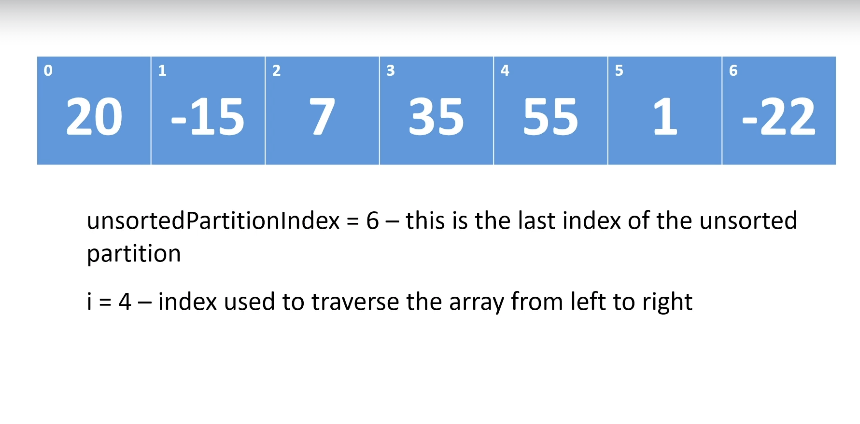
As the alghoritms continues in alters the data structure into ordered and unordered partitions. Its logical ordering, it means it we dont create separate array instances. Everything is done using the array that we are sorting. We particitate the array we are sorting.

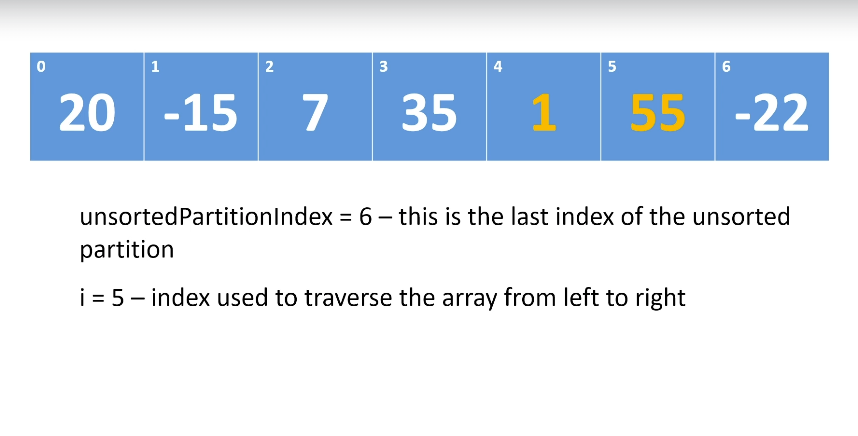
UsortedPartitionIndex – at the beginning 6, we havent sorted anything yet. Always starts with the last index because at the beginning we havent sorted anything.

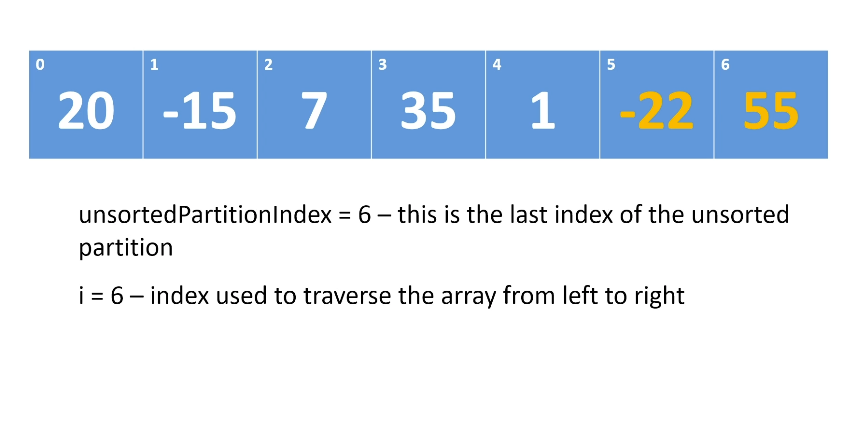
i = 0 , sorting start from left at 0

We go from zero, if element at zero is smaller than element at 1 we do no swaping. We increase i to 1. Then we continue

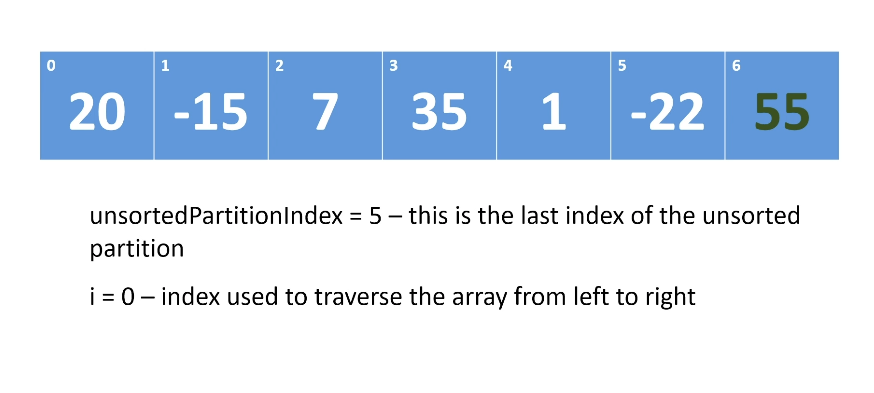






I is equal to unsortedPartitionIndex, we completed the first traversal. We now consider position 6 to be sorted, we change unsorted index to 5. Everything to index 6 and more is an sorted partition, less than 5 is unsorted partition. We set i back to 0.



Bubble sort – the large elements will buble up to the top of the array.

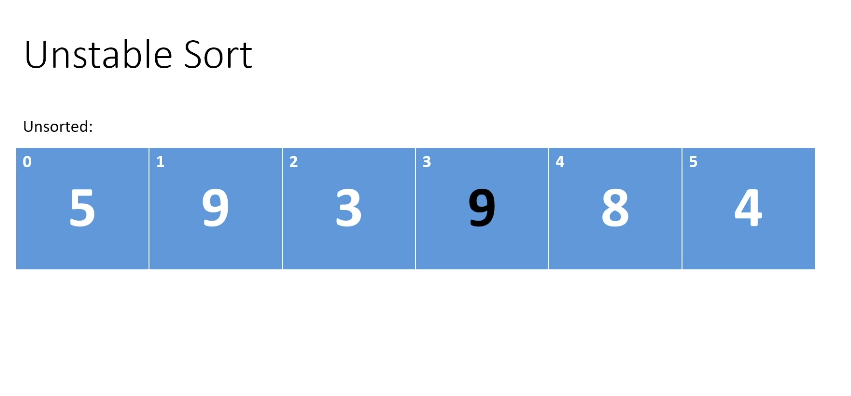
In place alghoritm – logically partitioned, in memory its in placed we dont have to create new arrays.

Creating local variables is okay. If with increasing n you dont have huge increase in memory consumption, its okay and its called in place alghoritm.

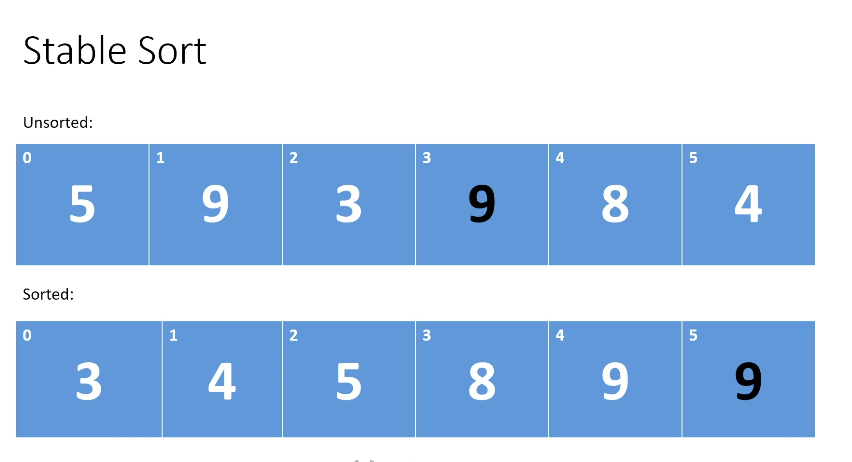
O(n\*\*2) complexity, bad. It degrades quickly.

# Stable vs unstable sort alghoritms

Do we have duplicate numbers when sorting ?



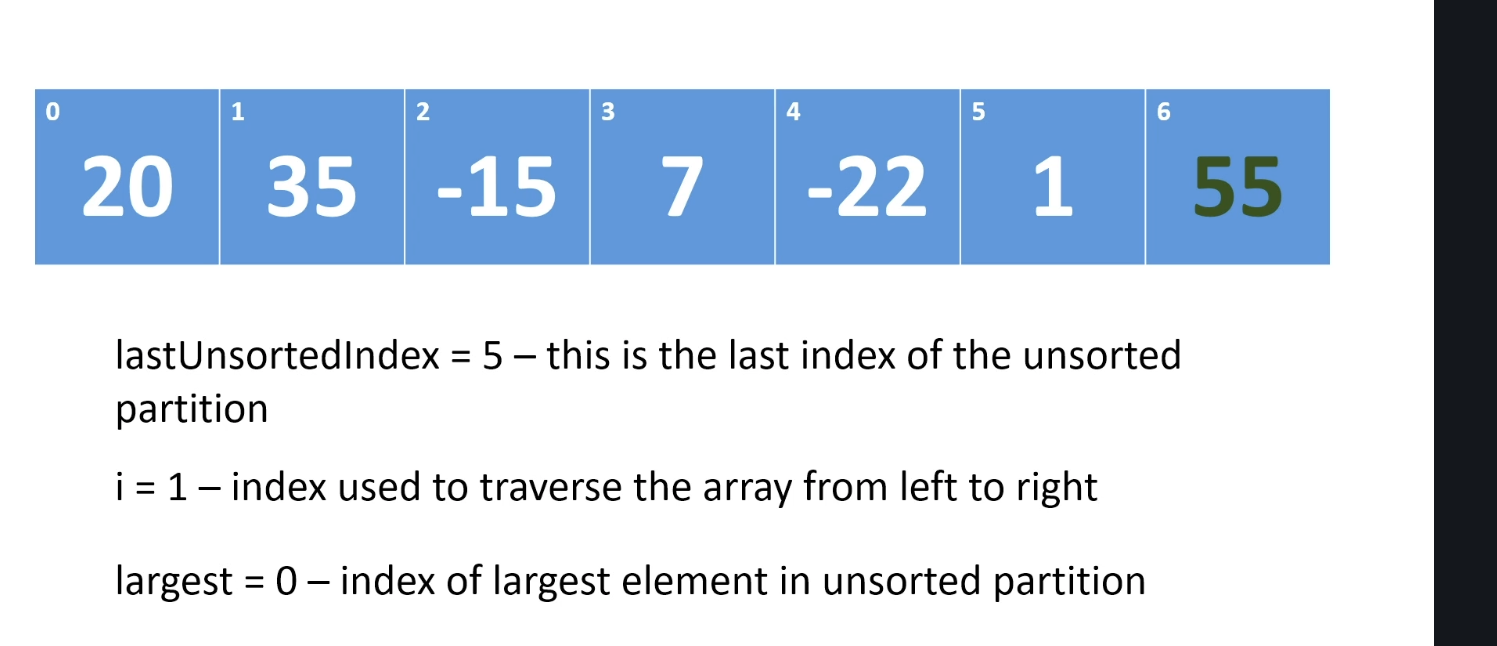
Will the relative order stay the same with the same elements ? Will the black nine after sorting be after white nine or not ? In unstable sorting, relative ordering of duplicate items will not be preserved. In stable sorting it will be preserved.

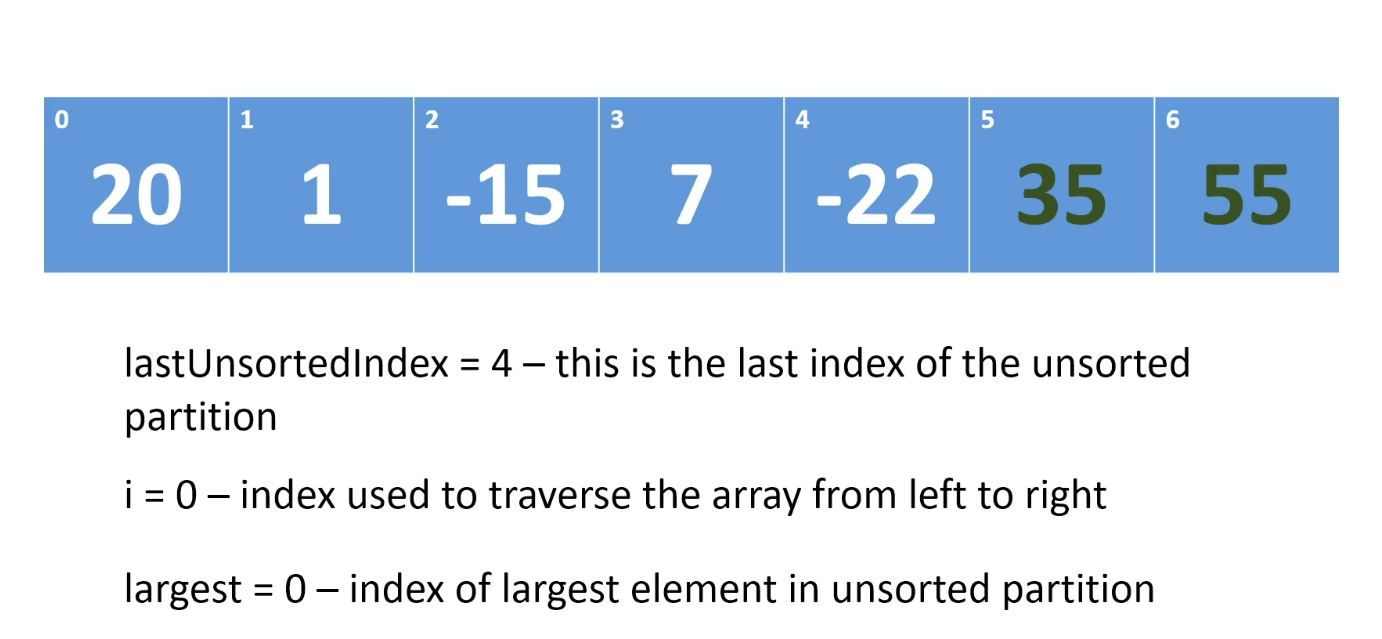


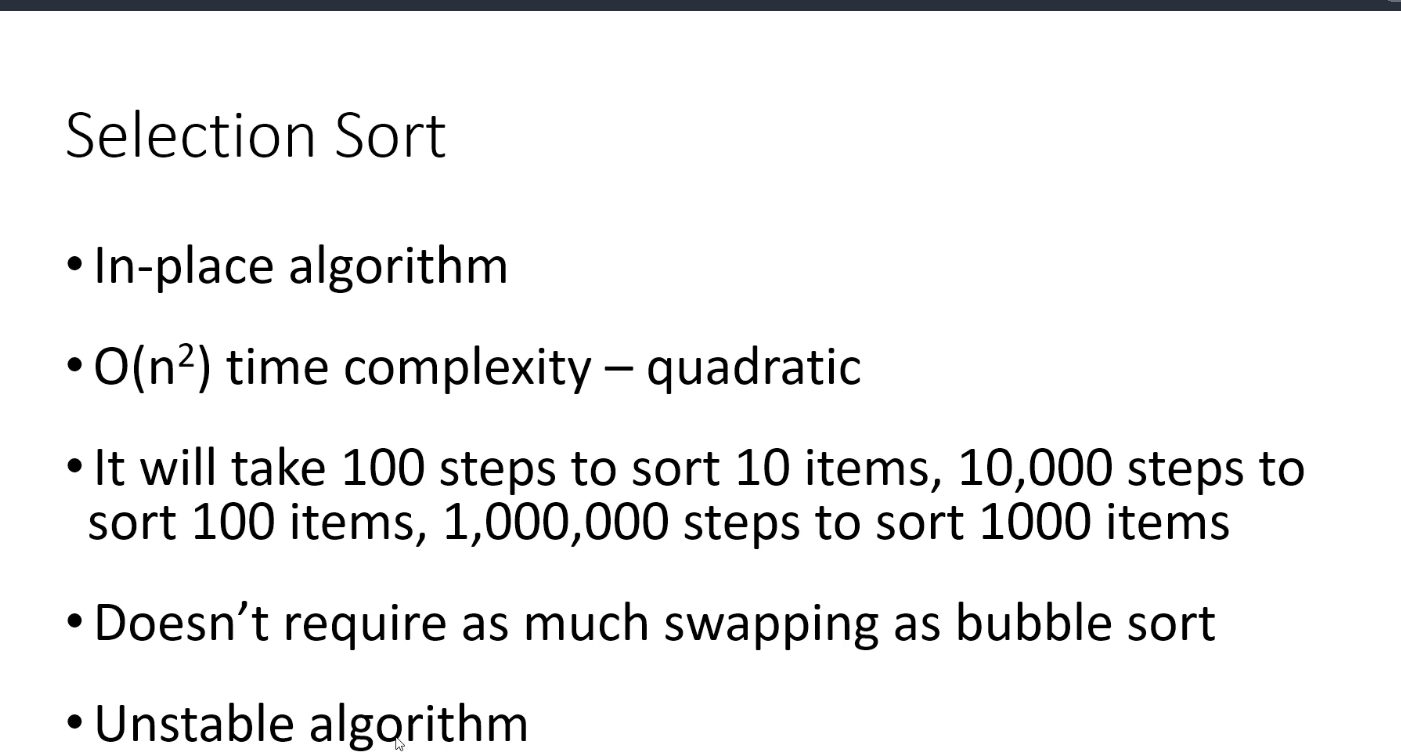
Stable sort is preferable to unstable sort. For integers it doesnt matter, if you are hovewer sorting objects it can make difference depending on what you are going to do with this sorted array afterwards.

Bubble sort is a stable sort alghoritm.

# Selection Sort Alghortim

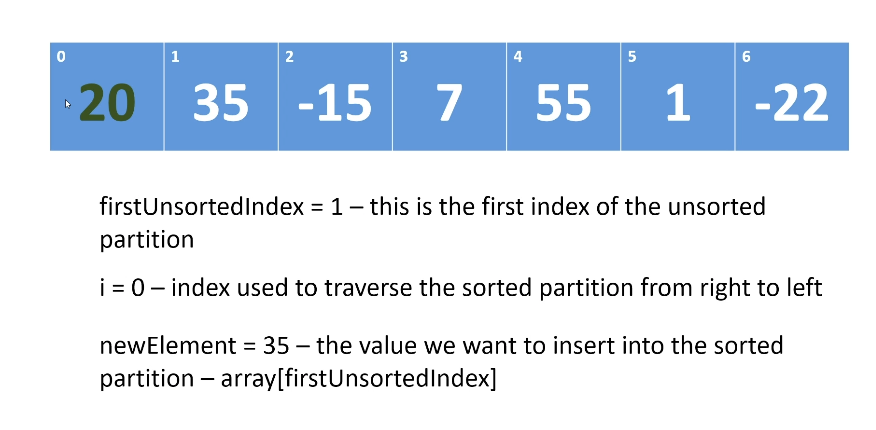




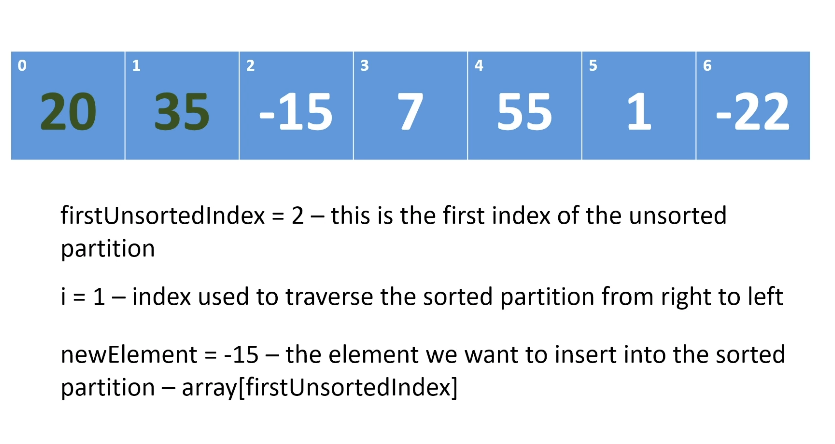


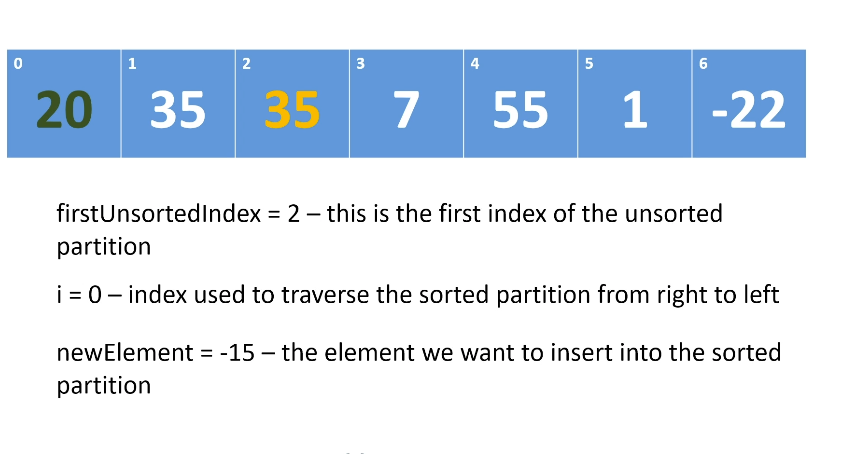
In average case selection sort is better than bubble sort, but not much.

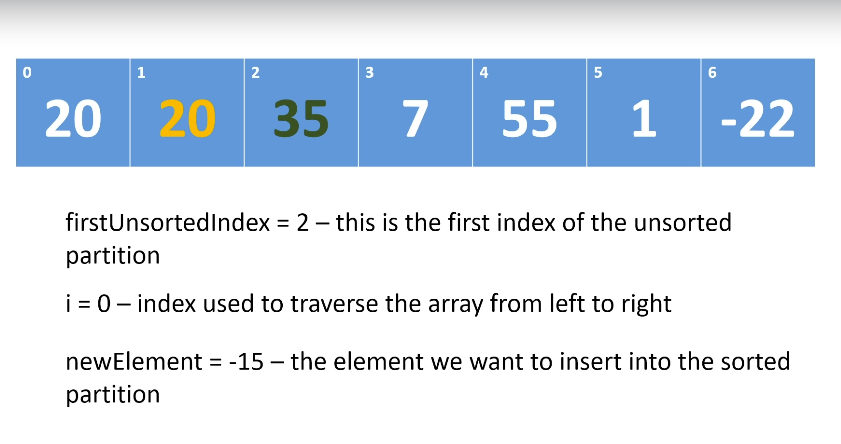
# Insertion Sort

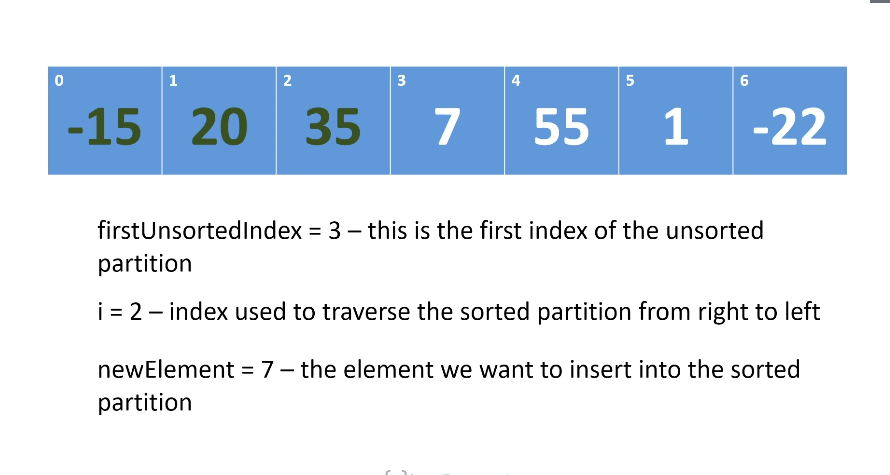


We také 35 and insert it into the sorted position where 20 is at the beginning(index 0). After that we compare the inserted element (35) with elements in the sorted partition, if we find element that is smaller than the element we are trying to sort, we insert it **after** this element. We are going from the right side.

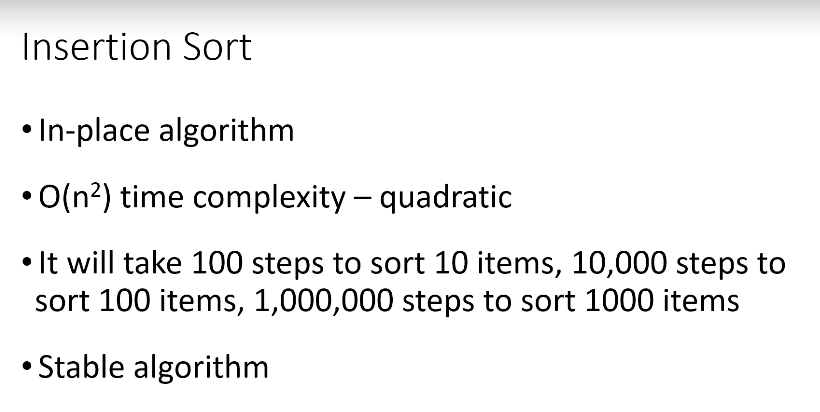






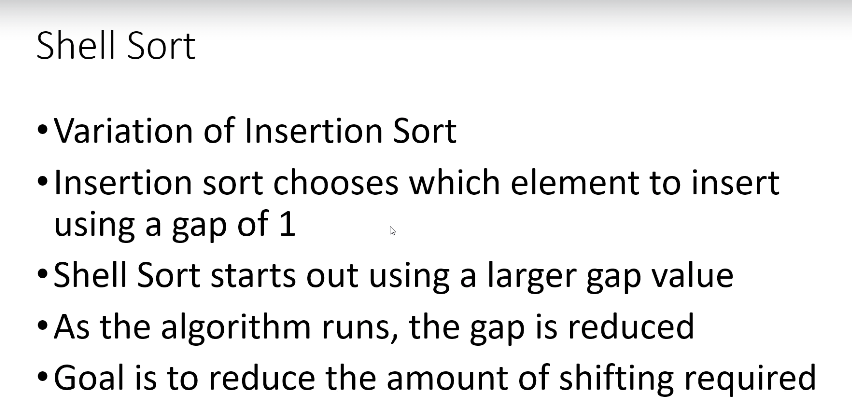


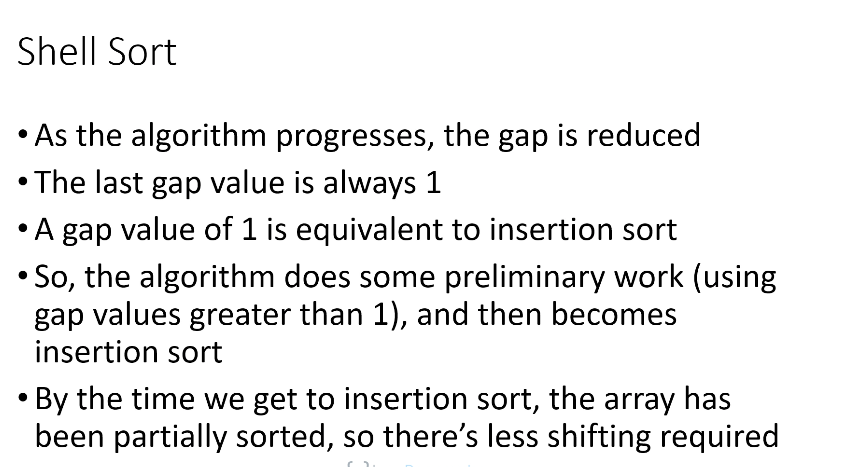
<https://www.hackerearth.com/practice/algorithms/sorting/insertion-sort/visualize/>



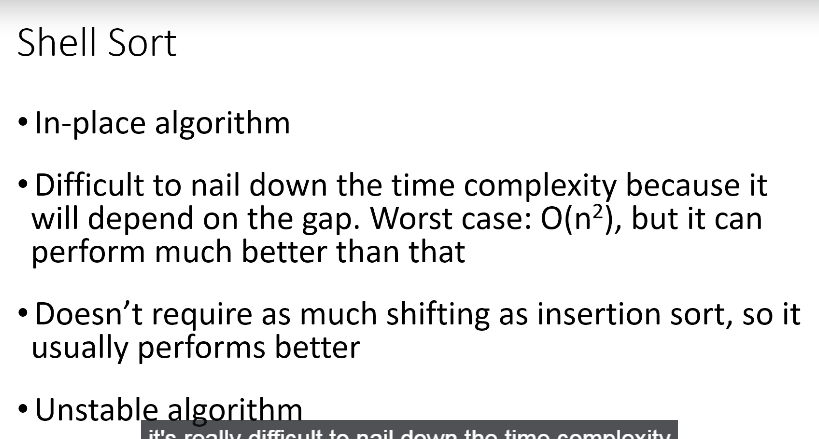
Insertion sort if we have a sequence of data that are almost sorted has basically linear complexity.

# Shell Sort



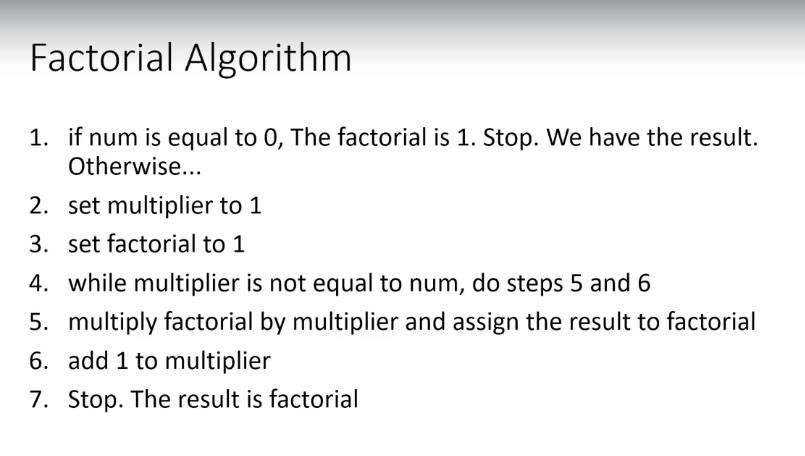


Last gap value is always one, last one is insertion sort.

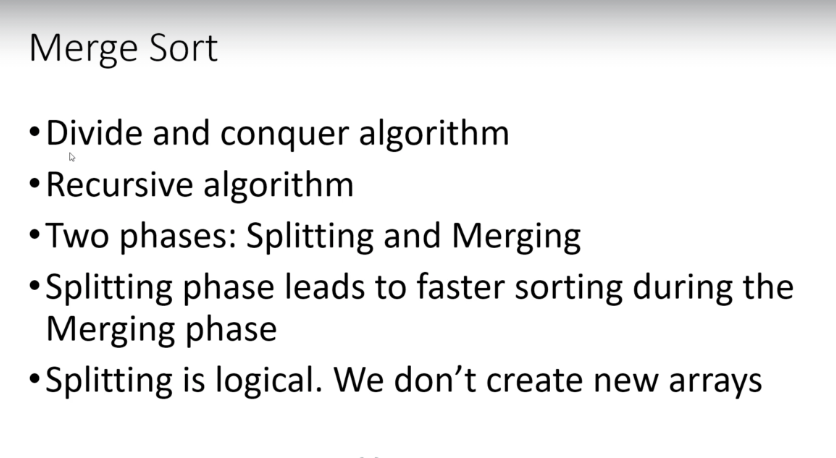


We can also improve bubble sort with shell sort

# Recursion

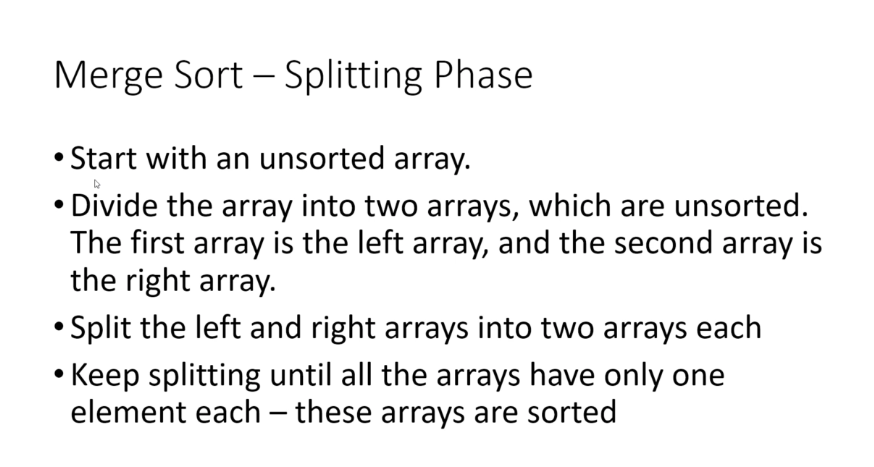


# Merge Sort

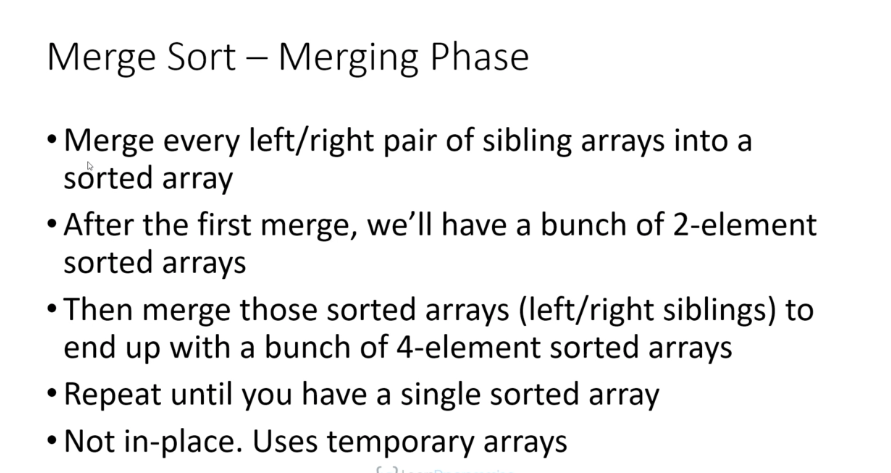


Divide and conquer – we are splitting the initial array to a smaller arrays.

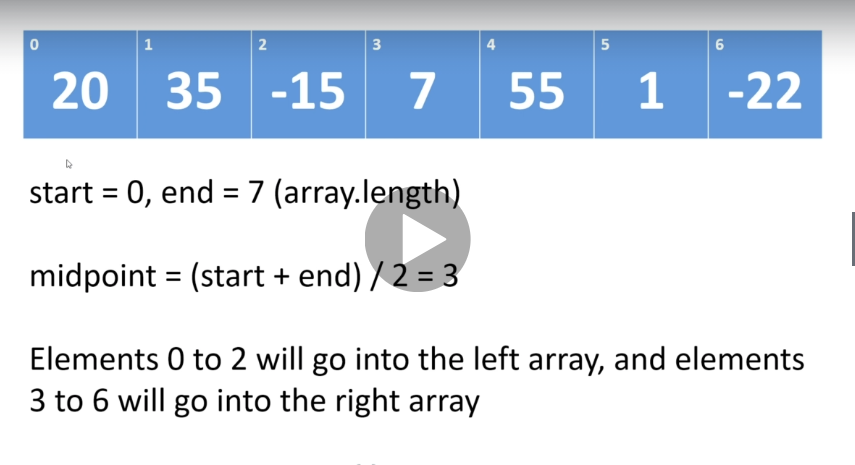
Spliting is logical, we dont create new arrays.

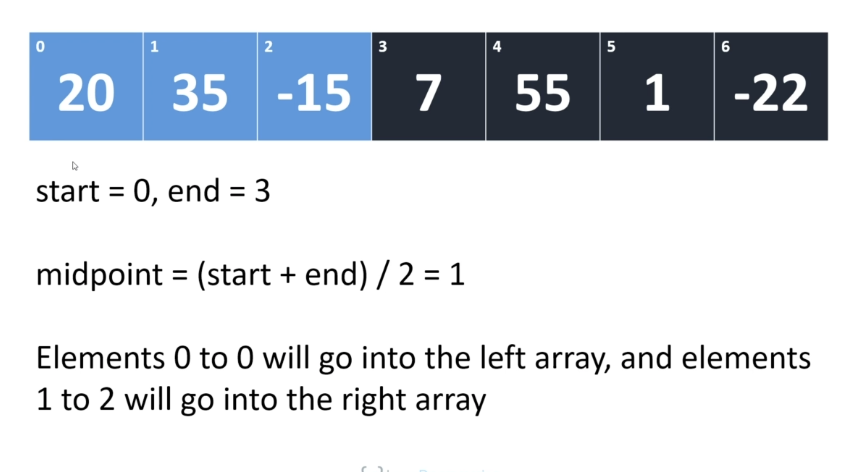


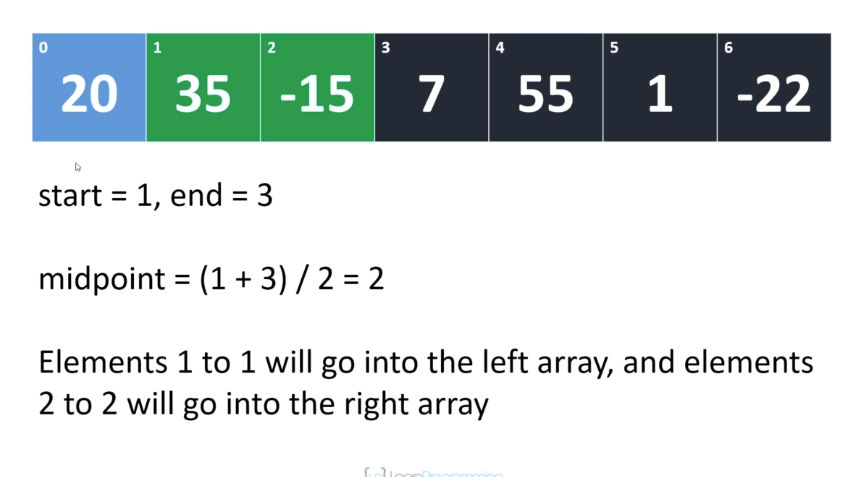
One element array is sorted by default because it has only one element.



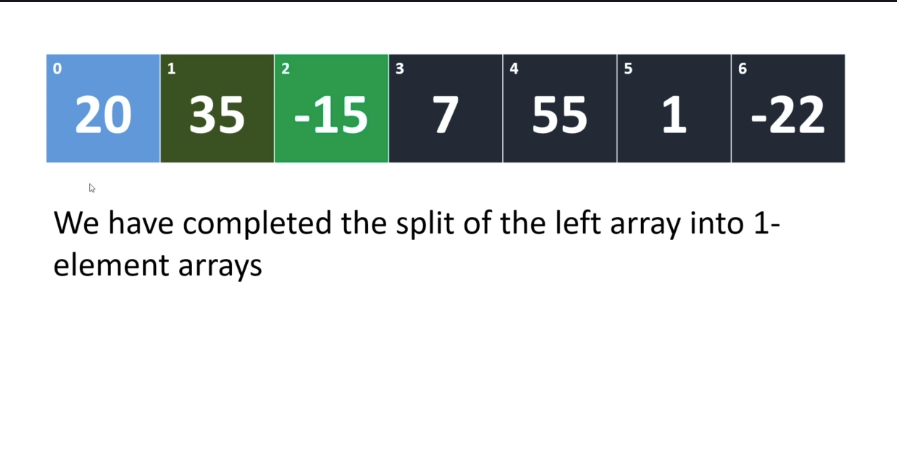
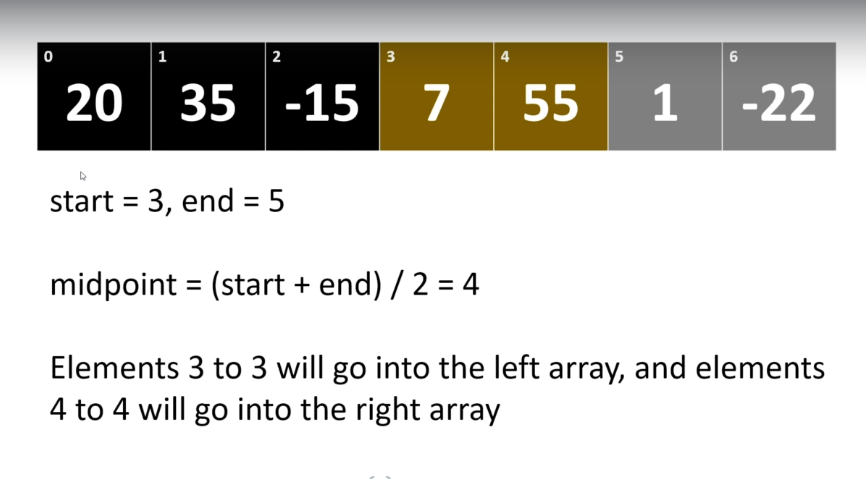
Merging phace is not in place, its not logical partitioning.

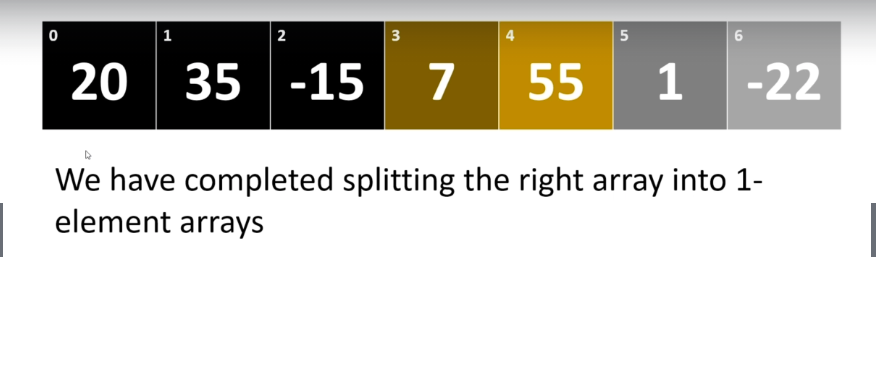




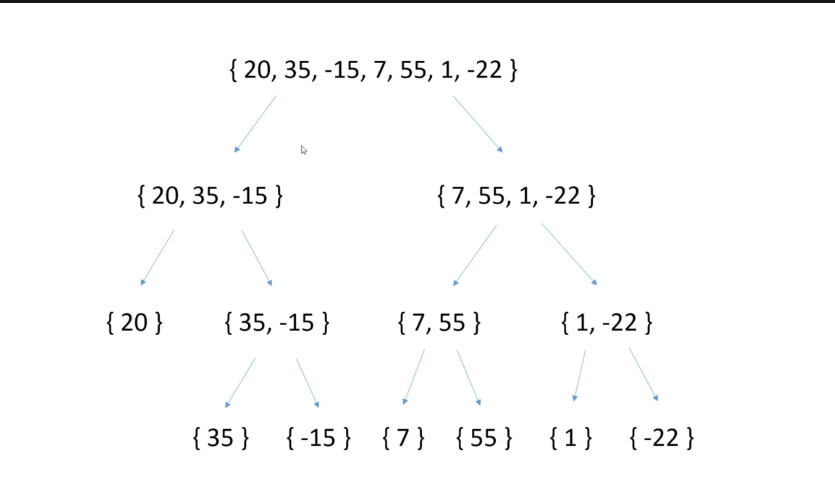


Sibling arrays are the ones that were divided.





Now is the merging phase. We are sorting during merging.



We first handle the entire left side and then right side. We partition top – down. We merge bottom up.

