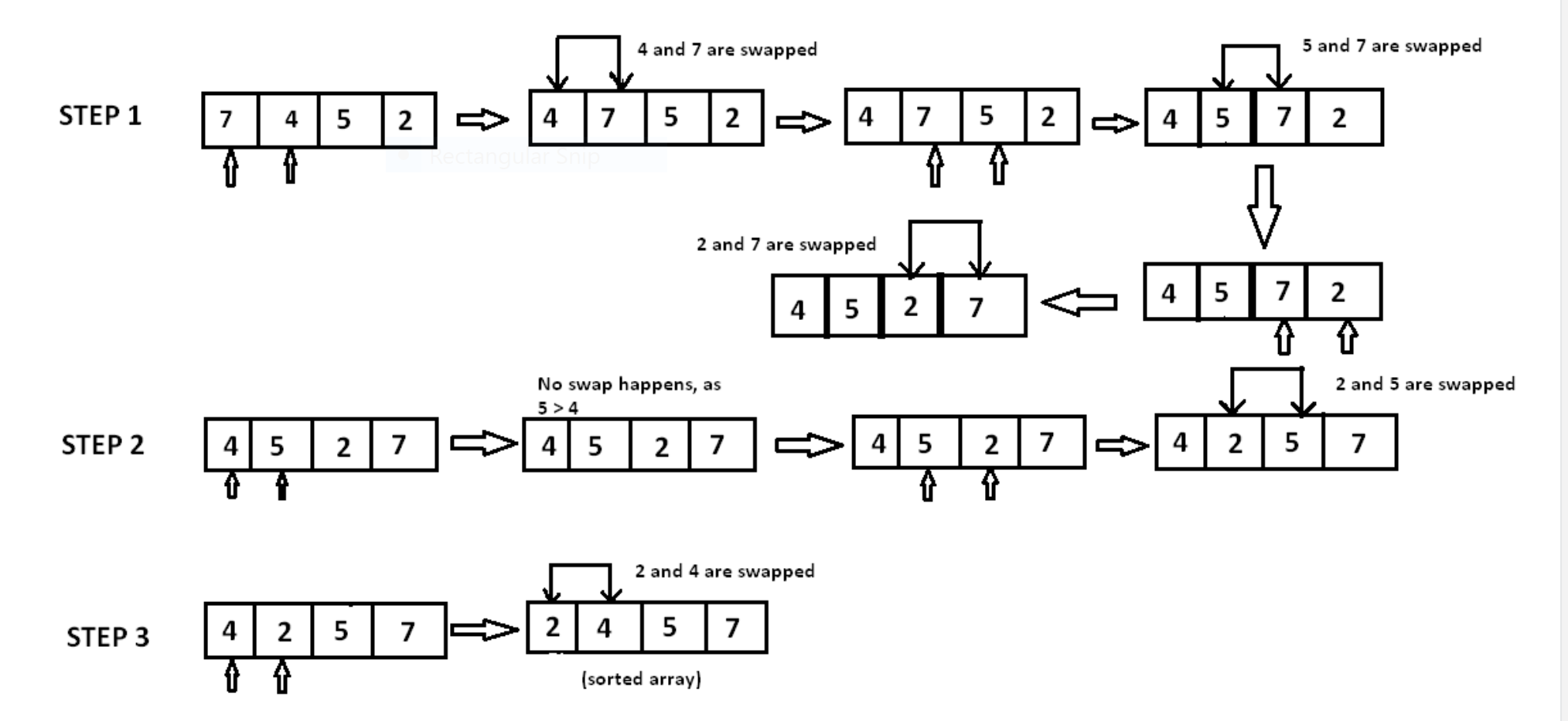
**Nabin Neupane**

**Assignment 3**

**Bubble sort:**

Bubble sort is a simple sorting algorithm. It is a comparison-based algorithm that in which each pairs of adjacent elements are compared, and the elements are swapped if they are not in order. This algorithm is not suitable for large datasets as its average and worst-case complexity are of O(n2) where n is the number of items. The first step is to find the smallest piece of the data (assuming that we are sorting it from smallest to largest) and swaps it with the first element of the array. The process is repeated; it finds the smallest piece of data in what’s left over, and swaps that in to the second element of the array. This process is repeated until the second to last data item has been selected swapped (the reason why it stop here because at this time there is only one data item left over; since it was not selected previously, it must therefore be the largest piece of the data).

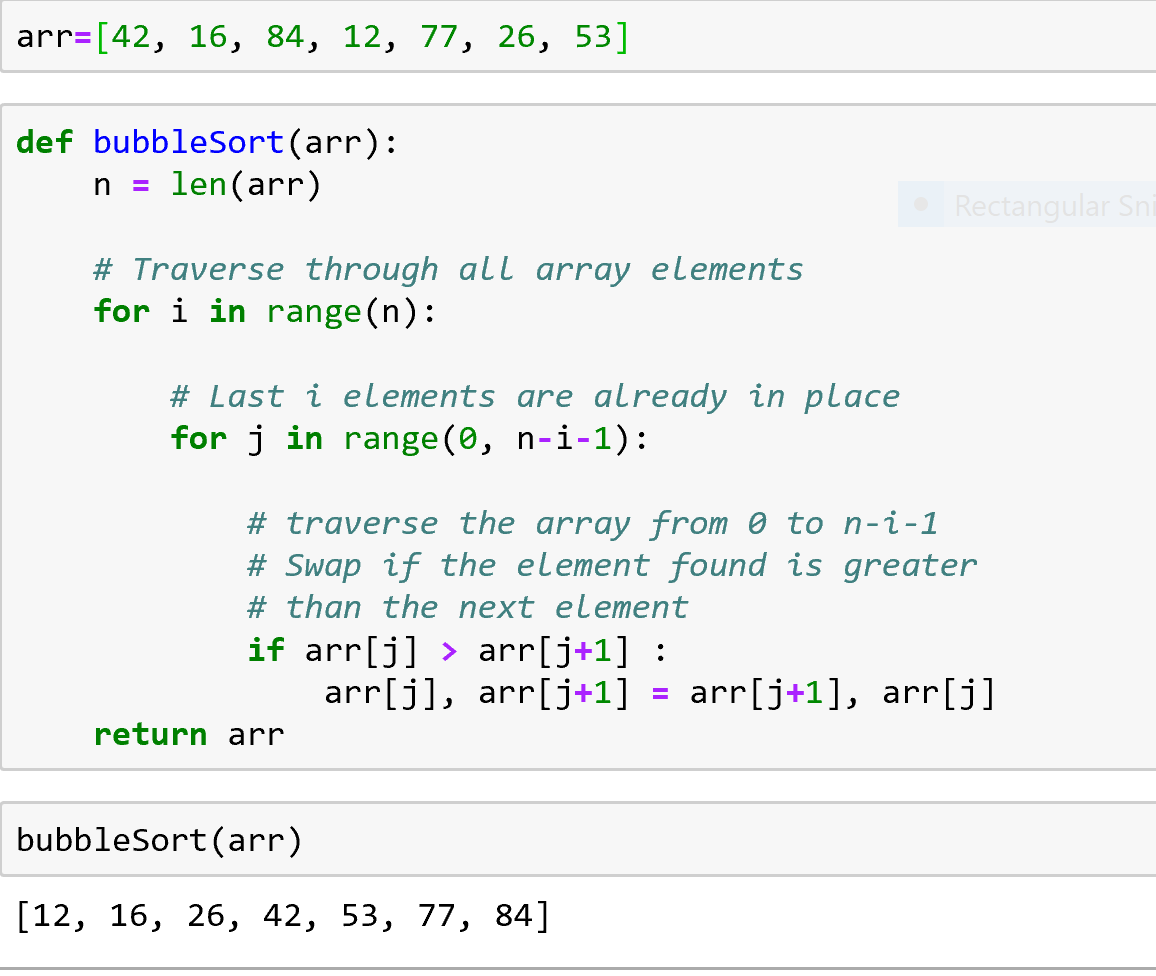


In step 1, 7 is compared with 4. Since 7>4, 7 is moved ahead of 4. Since all the other elements are of a lesser value than 7, 7 is moved to the end of the array. Now the array is A is {4,5,2,7} In step 2, 4 is compared with 5. Since 5>4 and both 4 and 5 are in ascending order, these elements are not swapped. However, when 5 is compared with 2, 5>2 and these elements are in descending order. Therefore 5 and 2 are swapped. Now, the array is {4,2,5,7}

In step 3, the element 4 is compared with 2. Since 4>2 and the elements are in descending order,4 and 2 are swapped. The sorted array is {2,4,5,7}.

**Complexity:**   
The complexity of bubble sort is O(n2) is both worst and average cases, because the entire array needs to be iterated for every element.

Python program to implement the bubble sort



There are different advantage of the bubble sort algorithm, it is simple and easy to understand and it takes only few lines of code. The data is sorted in place so there is little memory overhead and, once sorted the data is in memory, ready for processing.

The major disadvantage is the amount of time that it takes to sort. The average time increases almost exponentially as the number of the table elements increase. Ten times the number of items takes one hundred times as long to sort.

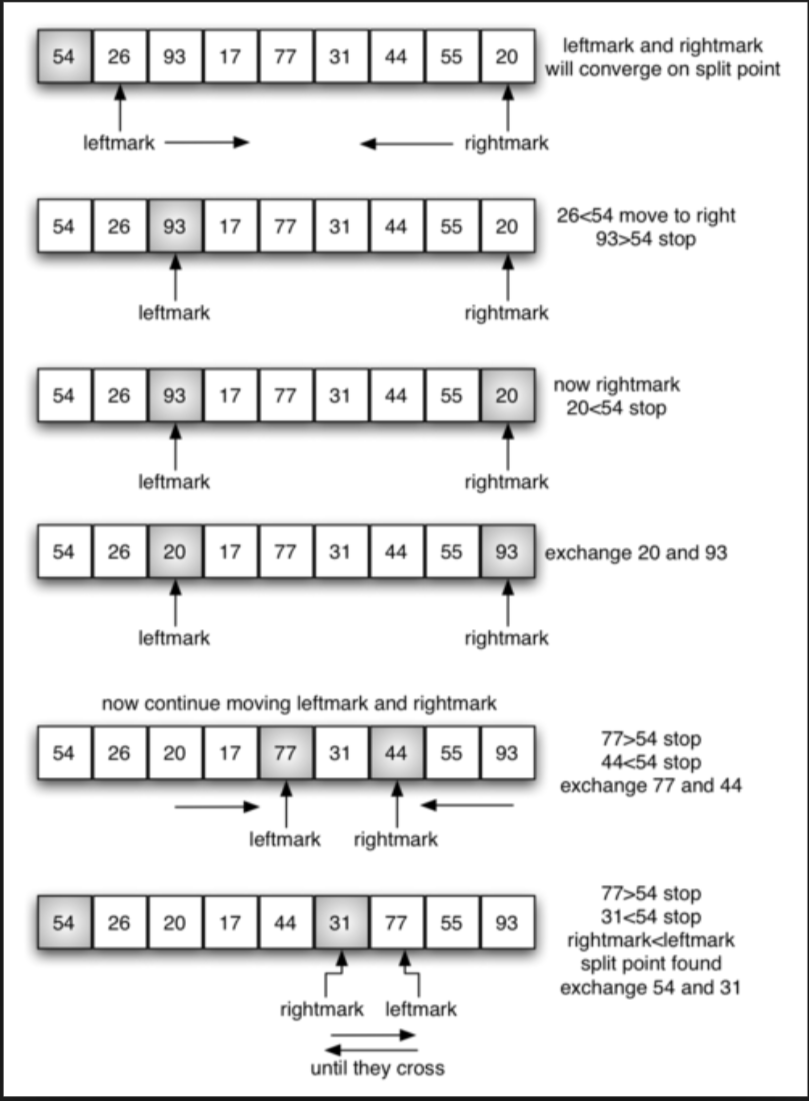
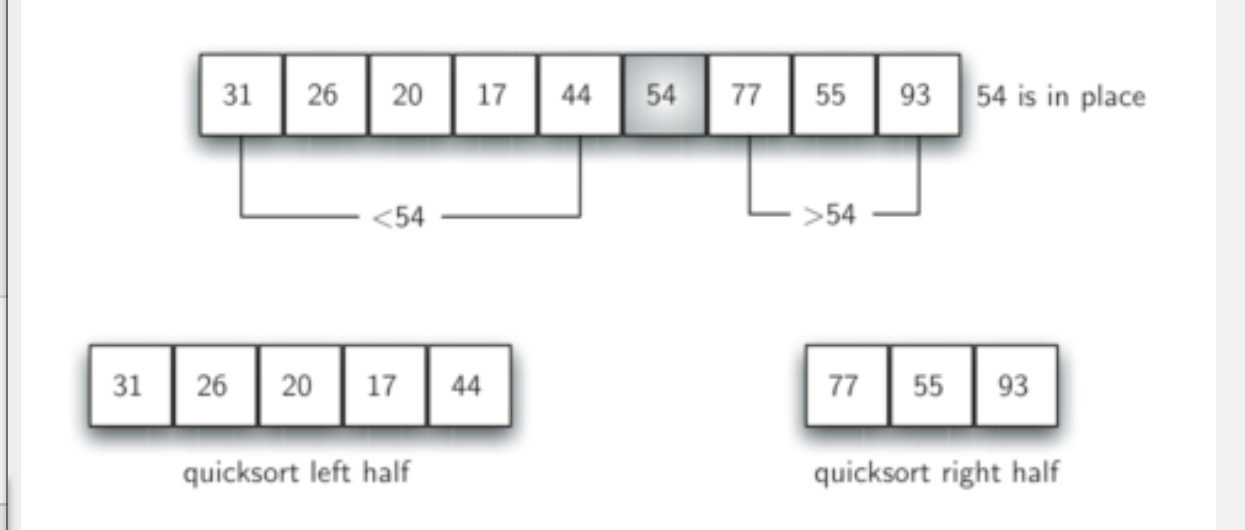
**Quick Sort:**

**Qui**cksort is one of the most efficient and most commonly used algorithms to sort a list of numbers. Quicksort can sort a list in place, saving the need to create a copy of the list, and therefore saving on memory requirements. The main intuition behind Quicksort is that if we can efficiently partition a list, then we can efficiently sort it. Partitioning a list means that we pick a pivot to right and all smaller items to the left. Once the pivot is done, we can do the same operation to the left and right sections of the list recursively until the list is sorted.

A quick sort first selects the value, which is called pivot value. There are different ways to choose the pivot value, we will use first item in the list. The role of the pivot value is to assist with splitting list, commonly called split point, will be used to divide the list for subsequent calls to sort.

From figure below 54 serves as the pivot value. Partition begin by locating two position markers, we can call them as left mark and right mark at the beginning and end of the remaining item in the list. The goal of the partition is to move items that are on the wrong side with respect to the pivot value while converging on the split point. Figure below shoes this process as we locate the position 54.

We start by incrementing left mark until we locate a value that is greater than the pivot value. Also we then decrement the right mark until we find a value that is less than the pivot value. At this point we find two items that are out of place with respect to split point. Here in our example this occurs at 93 and 20. We can exchange these two items and then repeat the process again. At the point right mark becomes less than left mark we stop. The position of right mark is now splits point. The pivot value can be exchanged with the contents of the split point and pivot value is now in place figure2. All items to the left of split point are less than the pivot value and all item to the right of the split point are greater than the pivot value.

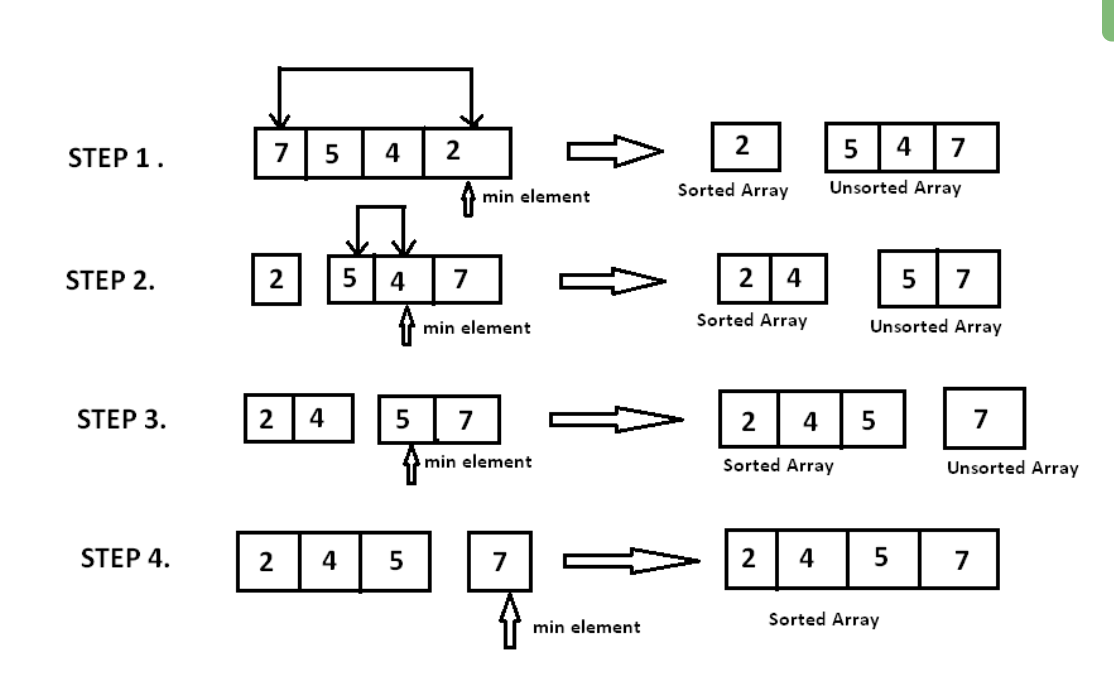
Complexity The worst case time complexity of this algorithm is O(N2) , but as this is randomized algorithm, its time complexity fluctuates between O(N2) and O(NlogN) and mostly it comes out to be O(NlogN).

**Selection sort:**

Selection sort algorithm is based on the idea of finding the minimum or maximum element in unsorted array and then putting it in its correct position in sorted array. Selection sort is a simple sorting algorithm. This sorting algorithm is an in-place comparison-based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the entire list.

Assume that array A=[7,5,3,2] needs to be sorted in ascending order.

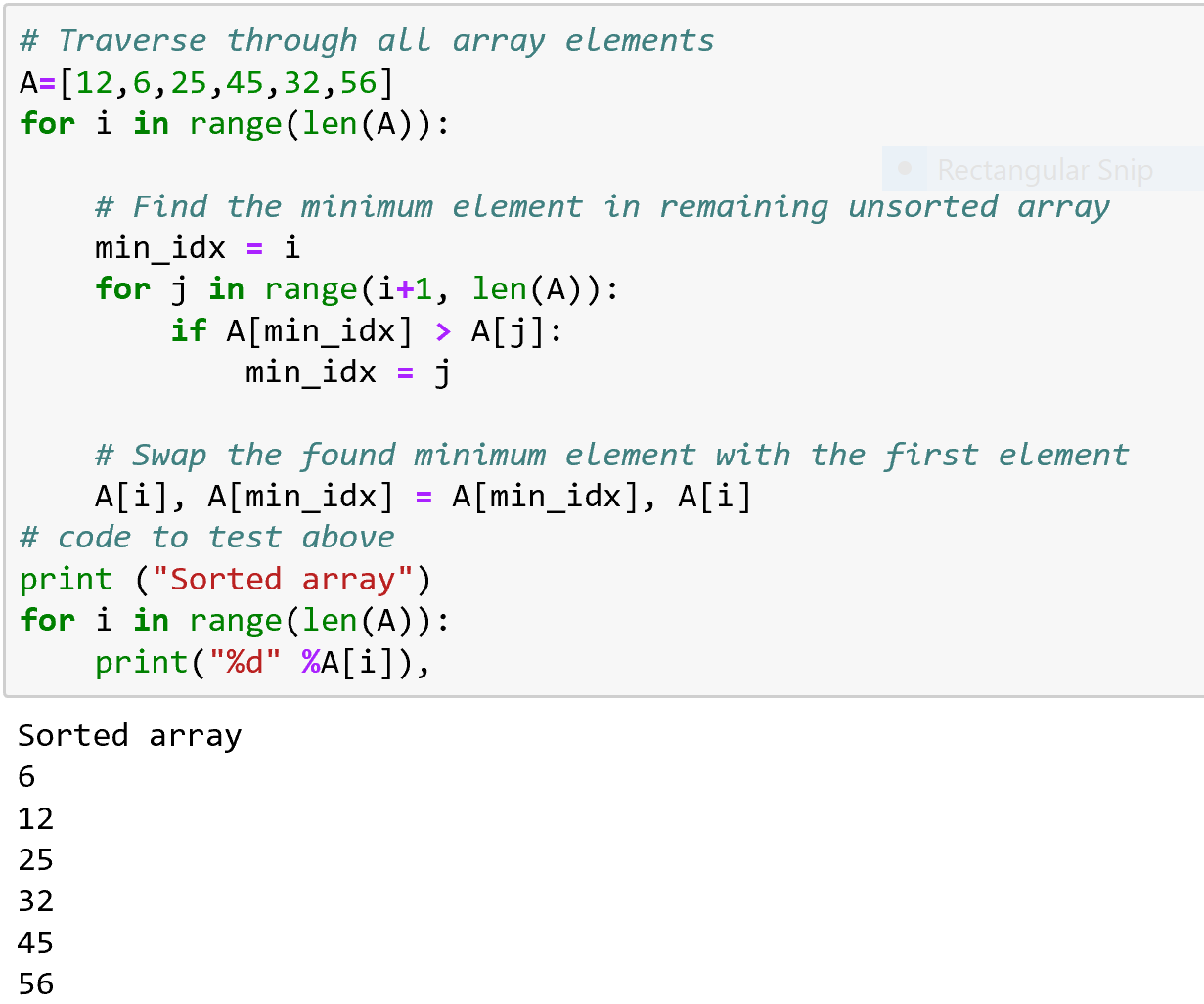
The minimum element in the array i.e. 2 is searched for and then swapped with the element that is currently located at the first position i.e . is 7. Now the minimum elements in the remaining unsorted array is searched for and put in the second position and so on.



The main advantage of the selection sort is that it performs well on a small list. Furthermore, because it is an in-place sorting algorithm, no additional temporary storage is required beyond what is needed to hold the original list. The primary disadvantage of the selection sort is its poor efficiency when dealing with a huge list of items. Similar to the bubble sort, the selection sort requires n-squared number of steps for sorting n elements. Additionally, its performance is easily influenced by the initial ordering of the items before the sorting process. Because of this, the selection sort is only suitable for a list of few elements that are in random order.

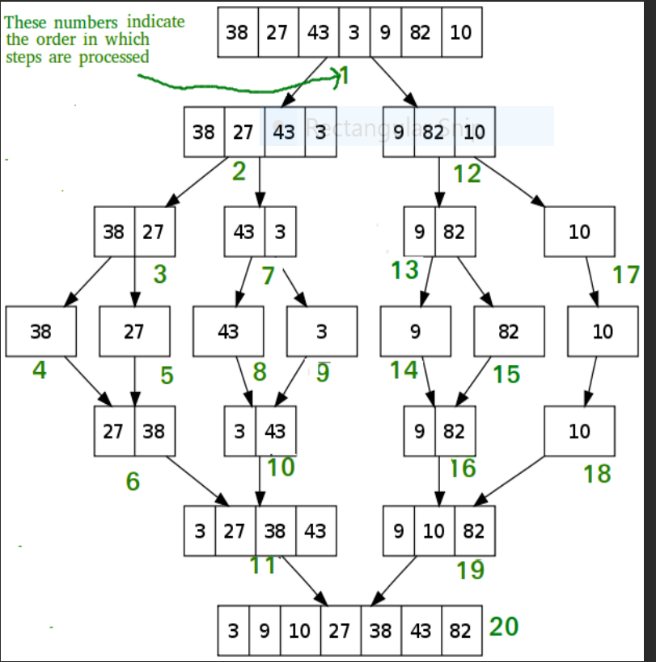
**Time Complexity:** O(n2) as there are two nested loops.

Python program for the selection sort



**Merge sort:**

merge sort is a sorting technique based on divide and conquer technique. With worst-case time complexity being Ο (n log n), it is one of the most respected algorithms. It divides input array in two halves, calls itself for the two halves and then merges the two sorted halves. **The merge () function** is used for merging two halves.

The diagram shows that complete merge sort process for an example array {38,27,43,3,9,82,10}. If we take closer look at the diagram, it can be seen that the array is recursively divided in to two halves till the size become 1. Once the size becomes to 1, the merge processes come s into action and starts merging array back till the complete array is merged.

The advantage of the merge sort are

1. O(nlogn) worst case asymptotic complexity
2. Can be used for external sorting
3. Highly parallelizable
4. Can be used to implement a stable sort

The disadvantage of merge sort are

1.marginally slower than quicksort in practice

2. Not as space efficient as block sort