PROGRAMING FUNDERMENTALS

ASSIGNMENT 08

Sorting Algorithm Project
H.M. Yehani Harshika Pamunuwa

SORTING ALGORITHMS

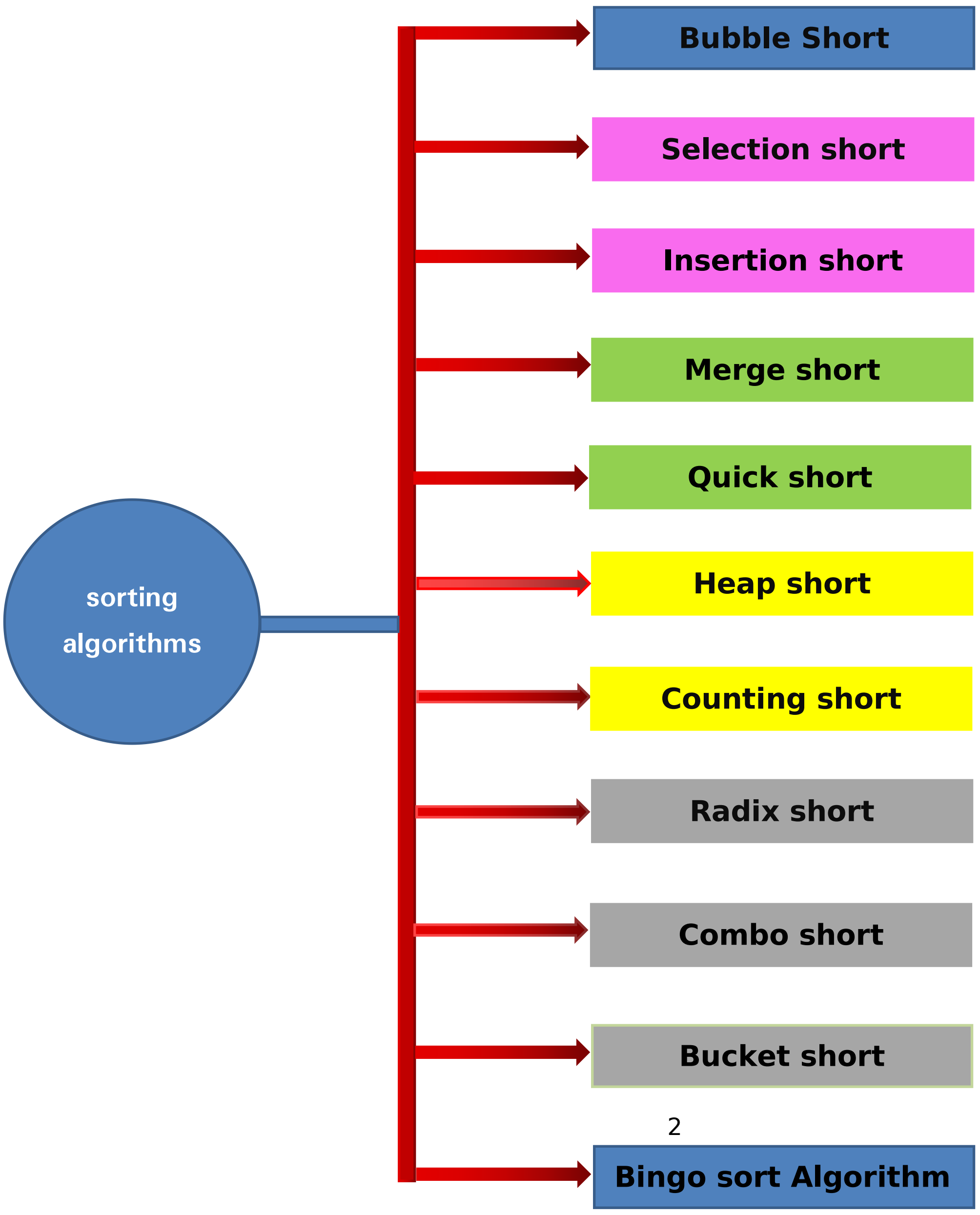
What is sorting?

A Sorting Algorithm is used to rearrange a given array or list of elements according to a comparison operator on the elements. The comparison operator is used to decide the new order of elements in the respective data structure.

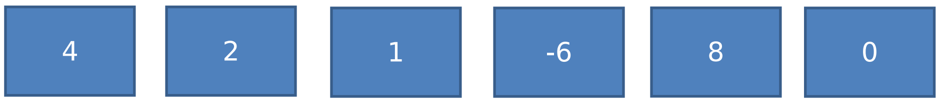
My Definition:-

Sorting means arranging data in particular order.

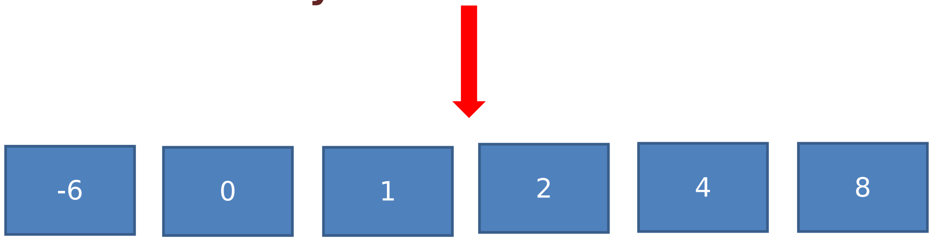
Types of Sorting algorithms,,,,,,,,,,,,,,,,,



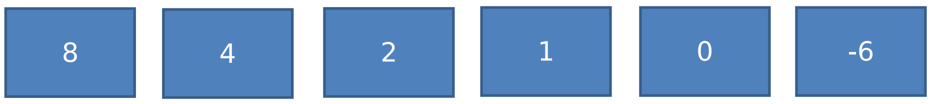
- **Example for Shorting Algorithm,**



Unsorted Array



Array shorted in ascending order



Array shorted in descending order

01)Bubble Short Algorithm :

What is Bubble Sort?

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order. This algorithm is not suitable for large data sets as its average and worst-case time complexity is quite high.

In this algorithm,,,,

- **traverse from left and compare adjacent elements and the higher one is placed at right side.**

- **In this way, the largest element is moved to the rightmost end at first.**

- **This process is then continued to find the second largest and place it and so on until the data is sorted.**

How does Bubble Sort Work?

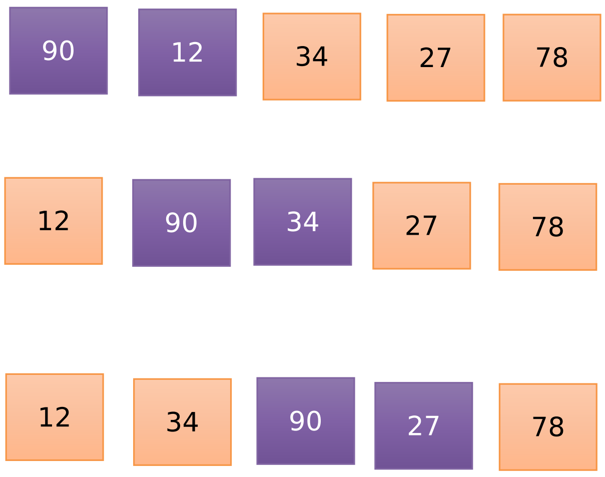
ar [ ] = {6,3,0,5};

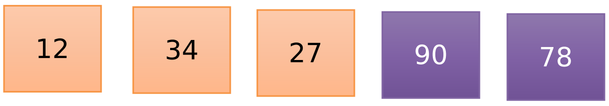
Frist Pass:

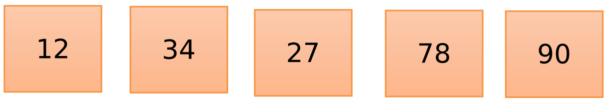
➢ **The largest element is placed in its correct position, i.e., the end of the array.**

STEP 01

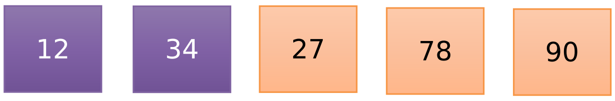
EXPLANATION OF HOW IT WORKS

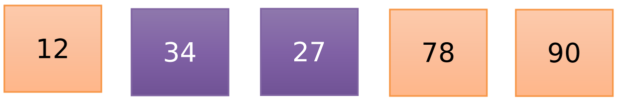


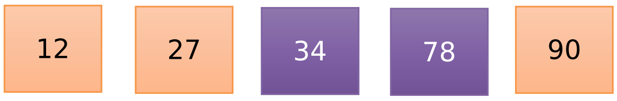


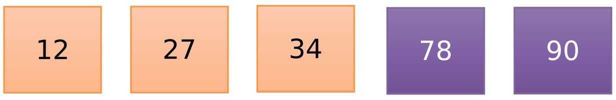


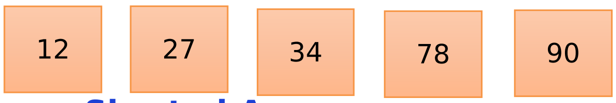
STEP 02 EXPLANATION OF HOW IT WORKS











Shorted Array

Bubble Sort Java Code:

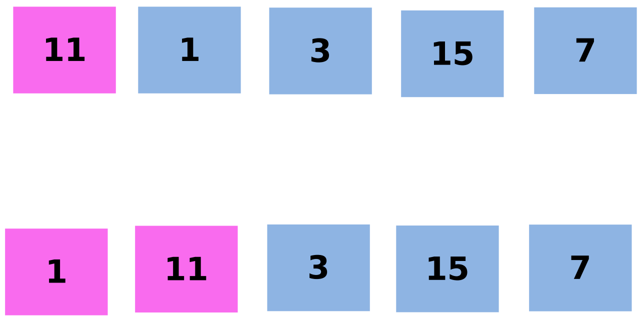
```
class Sort {
  public static void bubbleSort(int arr[], int n)
  {
    if (n == 1)                  //passes are done
    {
      return;
    }

    for (int i=0; i<n-1; i++)      //iteration through
unsorted elements
    {
      if (arr[i] > arr[i+1])     //check if the
elements are in order
      {                          //if not, swap them
        int temp = arr[i];
        arr[i] = arr[i+1];
        arr[i+1] = temp;
      }
    }

    bubbleSort(arr, n-1);         //one pass done,
proceed to the next
  }

  void display(int arr[])                //display the
array
  {
    for (int i=0; i<arr.length; ++i)
    {
      System.out.print(arr[i]+" ");
    }
}

public static void main(String[] args)
{
    Sort ob = new Sort();
    int arr[] = {6, 4, 5, 12, 2, 11, 9};
    bubbleSort(arr, arr.length);
    ob.display(arr);
     }
}
```

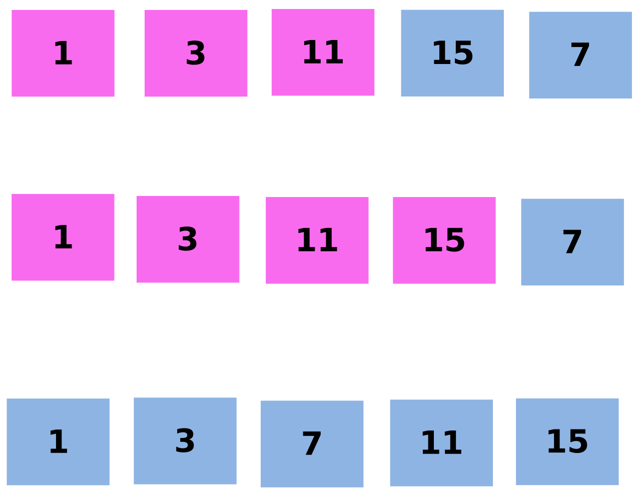
02) Selection Short Algorithm :

what is Selection Short?

Quadratic sorting algorithms are some of the more popular sorting algorithms that are easy to understand and implement. These don’t offer a unique or optimized approach for sorting the array - rather they should offer building blocks for the concept of sorting itself for someone new to it. In selection sort, two loops are used. The inner loop one picks the minimum element from the array and shifts it to its correct index indicated by the outer loop. In every run of the outer loop, one element is shifted to its correct location in the array. It is a very popular sorting algorithm in python as well.

Explanation Of How It Works:





Algorithm:---

1. **START**
2. **Run two loops: an inner loop and an outer loop.**
3. **Repeat steps till the minimum element are found.**
4. **Mark the element marked by the outer loop variable as a minimum.**
5. **If the current element in the inner loop is smaller than the marked minimum element, change the value of the minimum element to the current element.**
6. **Swap the value of the minimum element with the element marked by the outer loop variable.**
7. **END**

Selection Sort Java Code :

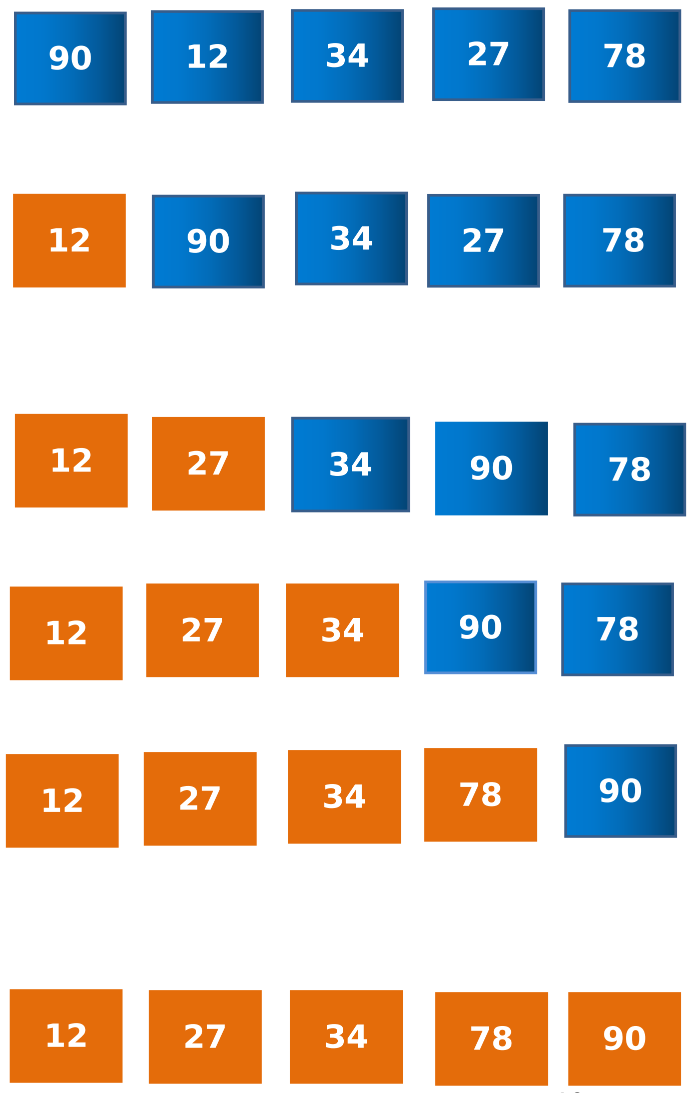
```
class Sort {
    void selectionSort(int arr[])
    {
        int pos;
        int temp;
        for (int i = 0; i < arr.length; i++)
        {
            pos = i;
            for (int j = i+1; j < arr.length; j++)
           {
                if (arr[j] < arr[pos])                    //find the
index of the minimum element
                {
                    pos = j;
                }
            }

            temp = arr[pos];             //swap the current
element with the minimum element
            arr[pos] = arr[i];
            arr[i] = temp;
        }
    }

    void display(int arr[])                     //display the
array
    {
        for (int i=0; i<arr.length; i++)
        {
            System.out.print(arr[i]+" ");
        }
    }

    public static void main(String args[])
    {
        Sort ob = new Sort();
        int arr[] = {64,25,12,22,11};
        ob.selectionSort(arr);
        ob.display(arr);
         }
    }
```

03)Insertion Sort Algorithm :

what is Insertion Sort?

It you are quite done with more complex sorting algorithms and want to move on to something simpler: insertion sort is the way to go. While it isn't a much-optimized algorithm for sorting an array,it is one of the more easily understood ones.Implementation is pretty easy too.In insertion sort, one picks up an element and considers it ti be the key.If the key is smaller than its predecessor,it is Shifted to its correct location in tha array.

Explanation Of How It Works:



Algorithm:

1. **START**
2. **Repeat steps 2 to 4 till the array end is reached.**
3. **Compare the element at current index i with its predecessor. If it is smaller, repeat step 3.**
4. **Keep shifting elements from the “sorted” section of the array till the correct location of the key is found.**
5. **Increment loop variable.**
6. **END**

Insertion Sort Java Code :

```
class Sort  {
   static void insertionSort(int arr[], int n)
   {
      if (n <= 1)                          //passes are done
      {
         return;
      }

      insertionSort( arr, n-1 );       //one element sorted, sort
the remaining array

      int last = arr[n-1];                     //last element of the
array
      int j = n-2;                              //correct index of last
element of the array

      while (j >= 0 && arr[j] > last)               //find the
correct index of the last element
      {
         arr[j+1] = arr[j];                        //shift section of
sorted elements upwards by one element if correct index
isn't found
         j--;
      }
```

```
        arr[j+1] = last;                    //set the last element at
its correct index
    }

    void display(int arr[])             //display the array
    {
        for (int i=0; i<arr.length; ++i)
        {
            System.out.print(arr[i]+" ");
        }
    }


    public static void main(String[] args)
    {
        int arr[] = {22, 21, 11, 15, 16};

        insertionSort(arr, arr.length);
        Sort ob = new Sort();
        ob.display(arr);
           }
   }
```

04)Heap Sort Algorithm :

what is Heap Sort?

Heap sort is one of the most important sorting methods in java that one needs to learn to get into sorting. It combines the concepts of a tree as well as sorting, properly reinforcing the use of concepts from both. A heap is a complete binary search tree where items are stored in a special order depending on the requirement. A min-heap contains the minimum element at the root, and every child of the root must be greater than the root itself. The children at the level after that must be greater than these children, and so on. Similarly, a max-heap contains the maximum element at the root. For the sorting process, the heap is stored as an array where for every parent node at the index i, the left child is at index 2 * i + 1, and the right child is at index 2 * i + 2.

A max heap is built with the elements of the unsorted array, and then the maximum element is extracted from the root of the array and then exchanged with the last element of the array. Once done, the max heap is rebuilt for getting the next maximum element. This process continues till there is only one node present in the heap.

This algorithm has two main parts:-

- **heap sort() -** This function helps construct the max heap initially for use. Once done, every root element is extracted and sent to the end of the array. Once done, the max heap is reconstructed from the root. The root is again extracted and sent to the end of the array, and hence the process continues.

- **heapify() -** This function is the building block of the heap sort algorithm. This function determines the maximum from the element being examined as the root and its two children. If the maximum is among the children of the root, the root and its child are swapped. This process is then repeated for the new root. When the maximum element in the array is found (such that its children are smaller than it) the function stops. For the node at index i, the left child is at index 2 * i + 1, and the right child is at index 2 * i + 1. (indexing in an array starts from 0, so the root is at 0).

05)Merge Short Algorithm :

Merge sort is the sorting technique that follows the divide and conquer approach. This article will be very helpful and interesting to students as they might face merge sort as a question in their examinations. In coding or technical interviews for software engineers, sorting algorithms are widely asked. So, it is important to discuss the topic.

Merge sort is similar to the quick sort algorithm as it uses the divide and conquer approach to sort the elements. It is one of the most popular and efficient sorting algorithm. It divides the given list into two equal halves, calls itself for the two halves and then merges the two sorted halves. We have to define the merge() function to perform the merging.

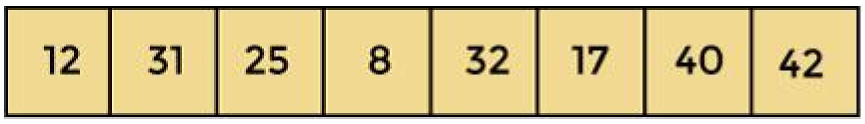
The sub-lists are divided again and again into halves until the list cannot be divided further. Then we combine the pair of one element lists into two-element lists, sorting them in the process. The sorted two-element pairs is merged into the four-element lists, and so on until we get the sorted list.

Working of Merge sort Algorithm

Now, let's see the working of merge sort Algorithm.

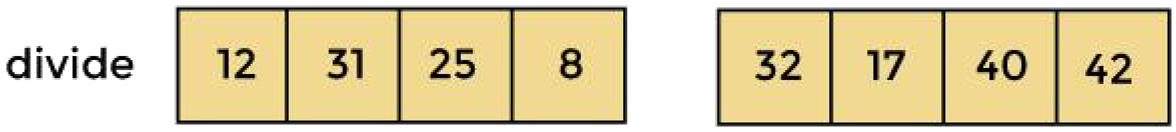
To understand the working of the merge sort algorithm, let's take an unsorted array. It will be easier to understand the merge sort via an example.

Let the elements of array are -

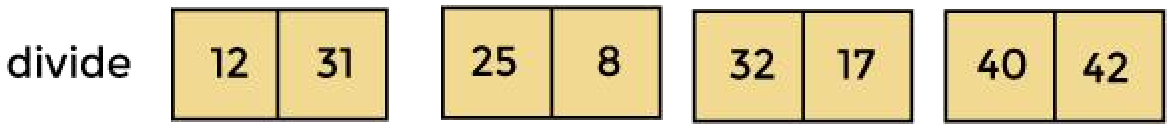


According to the merge sort, first divide the given array into two equal halves. Merge sort keeps dividing the list into equal parts until it cannot be further divided.

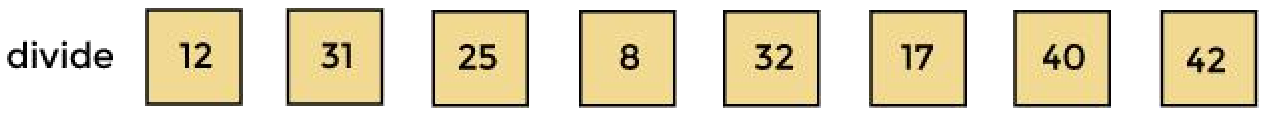
As there are eight elements in the given array, so it is divided into two arrays of size 4.



Now, again divide these two arrays into halves. As they are of size 4, so divide them into new arrays of size 2.



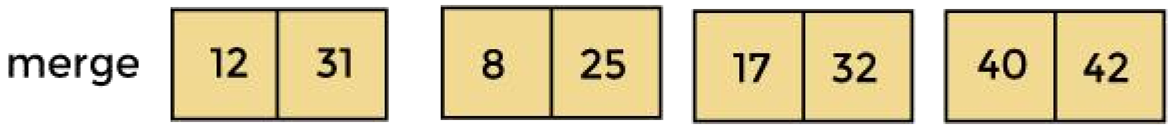
Now, again divide these arrays to get the atomic value that cannot be further divided.



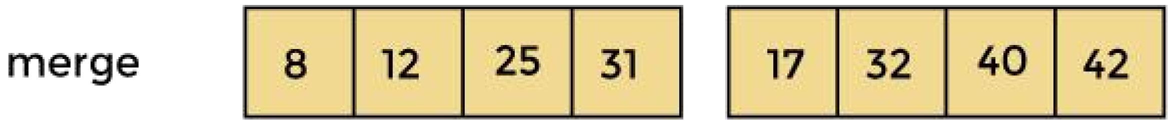
Now, combine them in the same manner they were broken.

In combining, first compare the element of each array and then combine them into another array in sorted order.

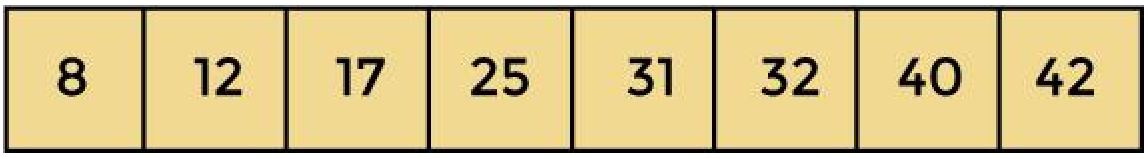
So, first compare 12 and 31, both are in sorted positions. Then compare 25 and 8, and in the list of two values, put 8 first followed by 25. Then compare 32 and 17, sort them and put 17 first followed by 32. After that, compare 40 and 42, and place them sequentially.



In the next iteration of combining, now compare the arrays with two data values and merge them into an array of found values in sorted order.



Now, there is a final merging of the arrays. After the final merging of above arrays, the array will look like -



8	12	17	25	31	32	40	42

Now, the array is completely sorted.