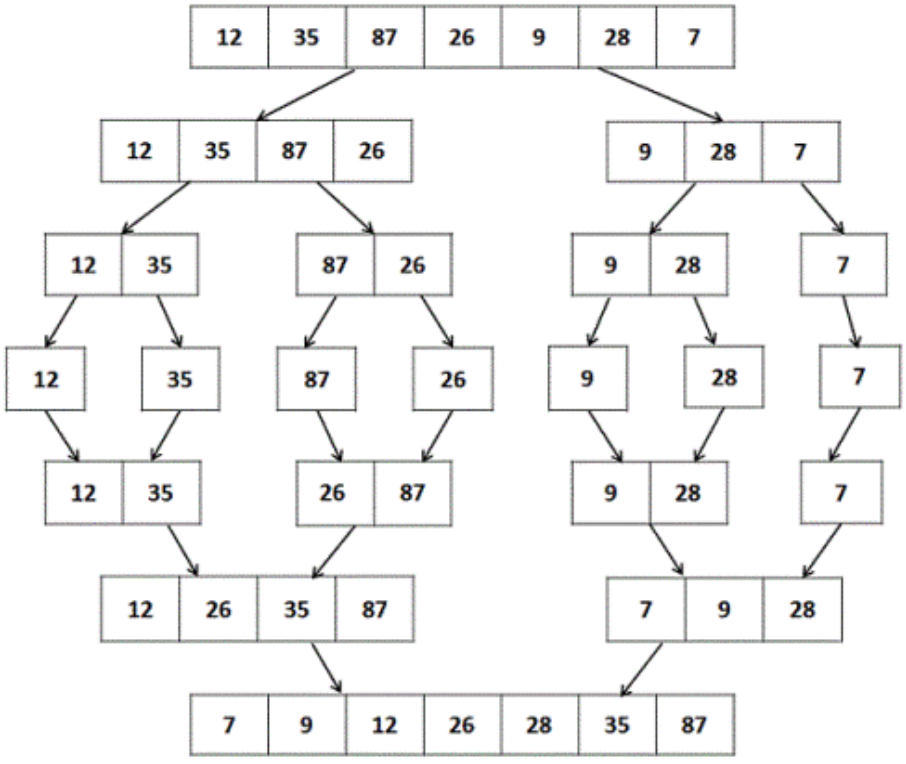
1. Use examples to explain the sorting algorithms?

Arranging the given items in a particular order is called sorting whether it could be ascending order or descending order

- **Merge sort**

Merge sort Is the most efficient algorithm when compared to all algorithms. This algorithm follows Divide and Conquer method. In every iteration it divides the given list into two halves' up to it reaches the single element and sorts the elements and return the sorted arra

Example:



Merge Sort

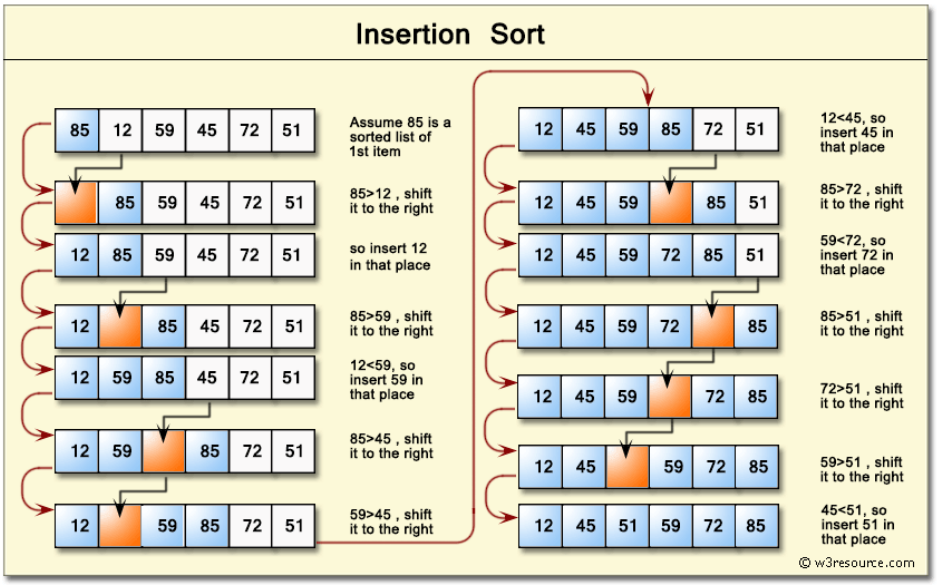
➢ Insertion sort

Insertion sort is like sorting the cards in our hands. Taking a particular card and keep that card in a correct position repeat until u get all the elements in an array are sorted

Step1: Take 2 elements in an array and check if they are not in correct positions then swap

Step2: Repeat these steps until u get the sorted array

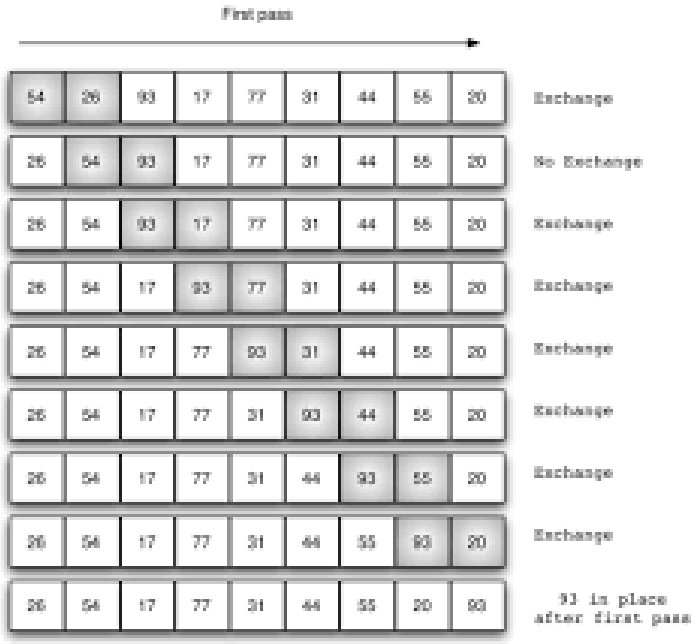
Example:



➢ Bubble sort:

Swapping every adjacent elements in an array until we get the sorted array

Example:



2. What are the benefits of stacks?

- Easy to manage data
- More securable
- Size must be fixed cant resize
- Stacks are more useful those who follows LIFO operations

3. What is the difference between stack and queue?

Stack	Queue
• Follows Last in First out operation • Insertion and deletion both happened in a single end • Inserting called Push • Deleting called Pop	• Follow First in First out operations • Insertion and deletion happened in a different end • Inserting called enqueue • Deleting called Dequeue

4. What are the different forms of Queue?

- Simple queue
- Circular queue
- Priority queue
- Double ended queue

5. Why should I use Stack or Queue data structures instead of Arrays or Lists, and when should I use them?

- ✓ Stacks and Queues are more accurate than arrays Stacks and queues are dynamic data structures.
- ✓ Time taken to operate the stacks and queues are easy than arrays.
- ✓ These are more useful if we want to solve the problems in a recursive manner.
- ✓ Stack and queues are recursive data structures

6. What is a significance of stack being a recursive data structure?

- ✓ Divide the task into smaller until reaching to solvable pieces
- ✓ Stack is necessary to call functions and return from those function calls
- ✓ Stack is in stable condition whenever u pushed or pop the values
- ✓ Easy to implement
- ✓ Reduce the complexity