DATA STRUCTURES AND ALGORITHMS

**DATA STRUCTURES:**

* Organise and store data.
* Each has advantages as well as disadvantages.
* The best data structure depends on the data fed.

**ALGORITHM:**

* It is the step needed to perform a particular task.
* There can be more than 1 algorithm for a particular task.
  + Many methods of implementation for an algorithm is possible as well.

**Big-O-Notation:**

* The time complexity is the no. of steps taken by an algorithm to accomplish a task.
* The memory complexity is the amount of memory used by the algorithm for execution.
* Time complexity is the main constraint.
* Time complexity based on the scale of data fed to it.
* O(1), O(logn), O(n), O(nlogn), O(n2) – Best to the worst algorithm.
* Big-O-Notation is hardware independent part.

**ARRAYS:**

* Indices start from 0(zero) and go upto n-1, where n is the size of the array.
* The highest valid index is n-1 and the index values can’t go negative.
* Arrays stored as contiguous block in memory.
* The array can’t be resized as the memory gets allocated in a contiguous block.
* Every element occupies the same amount of space in the memory based on the datatype of the array.
* An array objects are stored as the object references in the array, as a result maintaining the uniform size for each element in the array.
* The accessing is thus easier if we know the index.

**Bubble Sort:**

* As the algorithm progresses, the array gets partition into a sorted partition and an unsorted partition.
* Unsorted index starts from the end of the array. The other iteration variable ‘I’ goes to is increased as  
  I = size of array – 1 – index used to traverse from left to right.
* The first iteration of outer loop fixes max. value
* Bubble sort is an in-place algorithm, and it is O(n2) time complexity{quadratic}. Algorithm degrades quickly.

**Stable vs Unstable Sorts:**

* In case of duplicate values, the unstable sort basically yields the position/ relative ordering of the same data is not preserved, thus considered unstable.
* Stable is vice versa case of unstable sort.
* Stable sort helps when dealing with large repository kind of data.
* Relative ordering of duplicate items should be preserved.

**Selection Sort:**

* Just like bubble sort.
* The largest elements are listed from the end in each iterations finally obtaining an ascending ordered array.
* Repetitive iteration to compare the values in the array. We get the value to be the largest value
* In place algorithm, And O(n2) algorithm.
* Doesn’t require much swapping as bubble sort and is an unstable algorithm.

**Insertion Sort:**

* This also partitions the array.
* The sorted partition is from front of the array.
* It is actually assumed that 1st element is a sorted one.
* So the sorted partition builds up in forward manner in each iteration.
* Inserted value is compared to that of the already present values in the sorted partition.
* Sorted partition as a whole get the original sorted position based on the criteria satisfied.
* It is an in-place algorithm, is quadratic algorithm but is a stable algorithm.

**Shell Sort:**

* Variation of insertion sort to reduce runtime.
* Shell starts out using a larger gap value.
* As the algorithm progresses, the gap is reduced.
* By the time it obtains insertion sort, the array would have been partially sorted.
* Gap value chosen can influence the amount of steps taken to sort the algorithm
* Gap is calculated using **Knuth Sequence**:
* ‘k’ is based on the length of the array.
* ‘k’ is taken as array.length/2; which gets divided by 2 again and again to obtain a ‘k’ value of 1 after which we perform the insertion sort.
* The pre-sorting state takes place with some gap until gap reduces to 1.
* It is an in-place algorithm, Worst case is quadratic time.
* It is an unstable algorithm.
* Reduction in the shifting compared to the insertion sort.
* Also, a bubble sort can also implemented using a shell sort.

**Recursive Algorithm:**

* A function which calls itself again and again.
* Factorial computation is an example.

**Merge Sort:**

* Divide and conquer algorithm.
* Splitting is logical, without creating a new array. Indices take care of splitting.
* Recursive algorithm.
* Two phase – Split and merge.
* Splitting phase leads to faster sorting during merging phase.
* Splitting phase:
  + Start with unsorted array
  + Divide array into 2 arrays – the left and right array
  + The splitting goes on for the sub arrays until sorting reaches a 1 element array.
* Merge Phase:
  + Every left/right pair of sibling arrays into a sorted array.
  + From one element to a whole single array.
  + Temporary arrays used for merging.
* It is not an in-place algorithm
* Time complexity O(nlogn)-base 2.
* Stable algorithm

**Quick Sort:**

* Chooses a pivot element.
* The left half has elements value less than the pivot element value and right is greater than pivot value.
* Recursive operation.
* Pivot is chosen again and again to sort the array.
* In-place algorithm
* Time complexity O(nlogn)-base 2.
* It is an unstable algorithm.
* The worst possible case leads complexity to O(n2).

**Counting Sort:**

* Make assumptions about the data.
* No comparisons used.
* Count, no. of occurrences of each value.
* Non-negative values can only be used(integers).
* Values within a specific range.
* Not an in-place algorithm.
* O(n)
* Stability with increase of steps.

**Radix Sort:**

* The data has same radix(no of unique digits) and width.
* Sort based on individual digit or letter.
* Must use a stable sort algorithm at each stage.
* Start at rightmost point.
* Based on least significant digit.
* O(n) is the time complexity.
* Use of stable counting sort algorithm for radix sort.