* **Recursion and Backtracking**

Recursion is a process of defining a function or calculating a number of repeated applications of an algorithm.

The main idea behind this problem-solving technique is working your way through a sequence of decisions in which each choice leads you further along some path so if all choices are correct eventually you will end up with a solution.

In above approach we need to understand that all choices were correct which will not be an ideal solution to all problems thus once you have reached a dead-end or you figure out some choice was incorrect, we will have to trace back to previous decision points and algorithms that use this approach are called backtracking algorithms and process is **Backtracking**.

* **Linked Lists**

A linked list is a dynamic data structure. Any application which has unknown number of objects to be dealt with will be utilizing this and is a very common data structure that is used to create other data structures like: graphs, trees etc.

* **Stacks and Queues**

Stack is an abstract data type of container of objects where basically objects are inserted and removed based on LIFO principle.

Queue is also an abstract data structure or a linear data structure in which objects are inserted from end or tail whereas deletion is done at head or front thus using FIFO principle.

* **Tree**

A tree is an abstract data type which is used to organize data in the format of tree and this allows insertion, deletion and search a bit faster. This is an auxiliary data structure for other abstract data types.

* **Priority Queues**

A PQ is an abstract data type in which we are performing deletion or removal of elements based on their priority such as in processing jobs, process management in processor etc.

* **Graph**

Graphs are fundamental data structure in the world of programming. A graph abstract data type is a collection of nodes and vertices and connections between them called edges.

These are non-linear data structures and can be used to model many types of processes in all sorts of physical, biological and digital systems.

* **Disjoint Set**

A Disjoint set is an abstract data type is a collection of sets that are disjoint, which basically means that there are no items found in more than one set.

Application of union/find data structures include maze generation and Kruskal’s algo for computing the minimum spanning tree of a graph

* **Sorting Algorithms**

A Sorting algorithm is one which will arrange elements of list in a certain order [ascending and descending]. Searching elements, Database related algorithms etc.

* **Searching Algorithms**

A Searching algorithm is one which is used for an item with specified properties from a collection of items. The items may be sorted as records in a database, simple data element in arrays, text in files, nodes in trees and vertices /edges in graphs.

* **Selection Algorithms**

A Selection algorithm is one which is used for kth smallest/largest number in a list [also called as kth order statistic]. This includes finding the minimum, maximum and median elements.

* **Symbol tables or Dictionary**

This is basically a key-value pair where each key can also be referred as symbols and value could be definition or statement defining what a symbol is.

* **Hashing Algorithms**

Hashing is a technique used for storing and retrieving information as fast as possible. It is used to perform optimal search and is useful in implementing symbol tables.

Worst case complexity of hashing algorithms is still O(n), but on average it is O (1).

* **String Algorithms**

These algorithms are supposed to process strings for all sort of use cases such as our browser auto filling URL which is also called predictive text algorithms or auto completion algorithms.

* **Greedy Algorithms**

A greedy algorithm is also called a single-minded algorithm, it is easy-to-implement solutions to complex multistep problems by deciding which next step will provide obvious benefits.

* Selection sort
* Prim’s algorithms
* Kruskal’s algorithms
* Dijkstra algorithm
* Huffman coding algorithm
* **Divide and conquer**

These algorithms work on principles of Dividing problems and conquering each sub problems and then later on combine all solutions of sub problems to create solution for the parent or base problem.

**Data Structures**

* **Linear** [elements can be accessed in a sequential order but it is not compulsory to store all elements sequentially]
* **Non-Linear** [elements are accessed /stored in no sequence]

**Abstract data types**