Divide and Conquer Algorithms

Divide and conquer algorithms recursively breaks down a problem into two or more sub-problems until the sub problem simple enough to solve directly. The solution of the sub problem is then combines and gives a solution to the original problem.

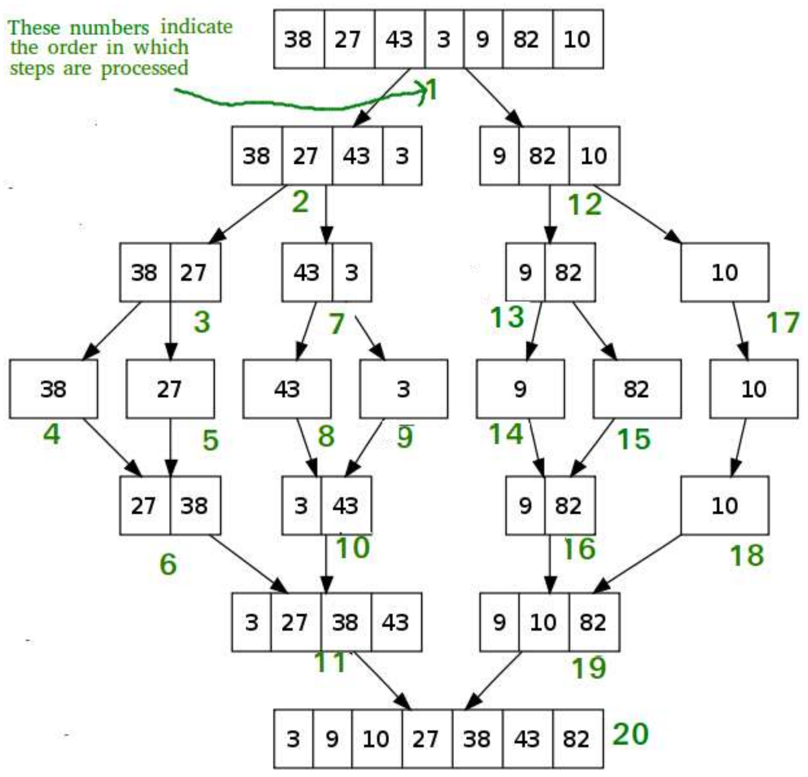
This algorithm says divide a big problem into small problem and try to solve them

The steps involve in this algorithm are:--

1. Divide: divide the problem into small sub problems
2. Conquer:-- solve those sub-problems recursively
3. Combine:-- combine the solution of the sub problem we will get the solution of the original problem

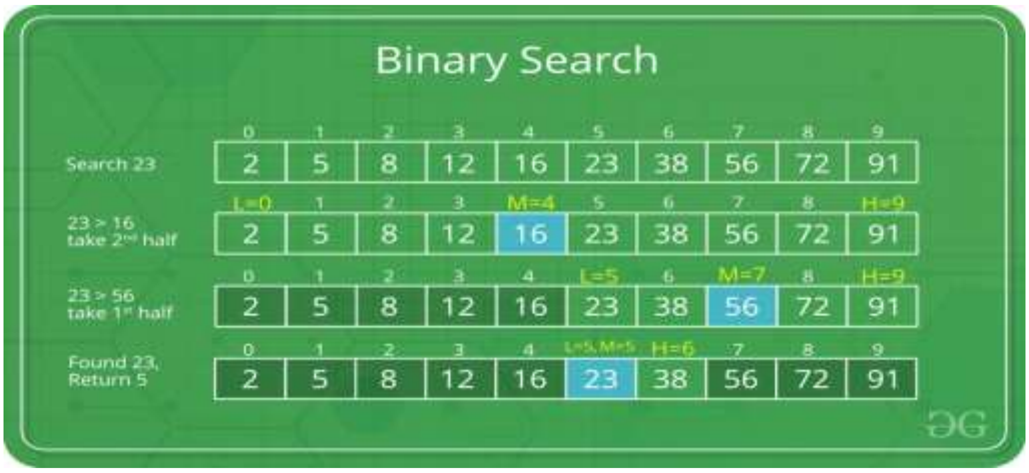
Example of Divide and conquer Algorithm are:--

1. Merge sort: --**Merge Sort is a sorting algorithm**. Find the middle element of the array base on that divide the array into two half and again in each half find the middle element and base on that divide it into two parts this is recursively done into both sib array until no comparison is left. After that and recursively sort them and combine we will get a sorted array.
   Complexity is O (n log n)



2. Quick Sort: -- **Quick sort is a sorting algorithm** It works by selecting a 'pivot' element from the array and partitioning the other elements into two sub-arrays, according to whether they are less than or greater than the pivot. For this reason, it is sometimes called **partition-exchange sort**. The sub-arrays are then sorted recursively.
   it select an element from the array considers as a pivot and then rearrange the elements of the array in such a way that all the smaller elements then the pivot move to the left side of the pivot and all elements greater than pivot move to the Right of the pivot element. And finally the algorithm recursively sort the subarray.
   Time Complexity is O (n log n)

3. Binary search: -- **binary search is searching algorithm** which works on sorted array. The targeted value is compare with the middle most elements of the collection if it matches then return the index of the middle element. If targeted value is greater than the middle element the search is shifted to the right of the middle element(new interval) and if it less then search moves to left of the middle value and. This repeatedly execute until a match is hit or sub array reduce to zero. Time complexity is O (log n).. Very good



Greedy Algorithm

Greedy algorithm is an algorithm and also a technique which is design to obtain optimum solution for a given problem. The algorithm says that pick the closest solution that seems to be provide an optimum solution. It says find a localized optimum solution which may sometimes not provide global optimum solution.

Explanation: --- Counting coins.

Coins provide Rs. 1, 2, 5, 10 and ask to count Rs. 18

1. Select one Rs 10 remaining 8
2. Select one Rs 5 remaining 3
3. Select one Rs2 remaining 1
4. Select one Rs1 solved

Every step we go for the optimum solution (picking up the coins)

Coins provide Rs 1, 7,10 and ask to count Rs. 15

1. Select one Rs 10 remain 5
2. Select one Rs 1 remain 4
3. Select one Rs 1 remain 3
4. Select one Rs 1 remain 2
5. Select one Rs 1 remain 1
6. Select one Rs 1 solved

But which can be done by (7+7+1) but greedy approach says always go for locally optimum solution

Example:--

1. Travelling salesman problem
2. Prim’s minimal spanning tree
3. Kruskal’s minimal spanning tree
4. Dijkstra’s minimal spanning tree