**DIVIDE AND CONQUER ALGORITHM**

**Introduction**

WHAT IS DIVIDE AND CONQUER ALGORITHM?

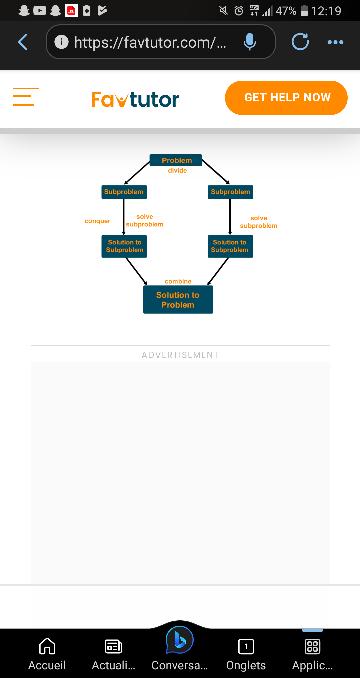
Divide and conquer algorithm is a problem-solving strategy that recursively breaks down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly. The solutions to the sub-problems are then combined to give a solution to the original problem.

**How do we solve a problem using divide and conquer algorithm?**

Solving a problem using divide and conquer algorithm involves three main steps which include;

* Divide: the problem is divided into two or more smaller sub-problems
* Conquer: the sub-problems are conquered by solving them recursively
* Combine: the solutions to the sub-problems are combine into the solutions for the original problem

The diagram below will explain to us more detailly how the divide and conquer function



**Type of algorithms which applies the principle of divide and conquer algorithm**

* Binary search algorithm: This algorithm is used to sort an array of data by dividing the array into half until the value is found or determined to not be present.
* Merge sort algorithm: This algorithm is used to sort an array of data by dividing the array into smaller sub-arrays, sorting each array independently and then combining the sorted sub-arrays.
* Quick sort algorithm: This algorithm is used to sort an array of data by selecting a pivot element, dividing the array into two sub-arrays based on the pivot element and then recursively sorting each sub-array.
* Strassen’s algorithm: This algorithm is used to multiply two matrices of size n\*n by dividing each matrix into 4 smaller sub-matrices, multiplying the sub-matrices recursively, and then combining the results.
* Closest pair of points algorithms: This algorithm is used to find the closest pair of points in a set of two-dimensional points by dividing the smaller sub-sets, recursively and then combining the results.

**Sample problems and their divide and conquer solution**

Problem 1: we need to look for something that we have lost in the house.

**How can we solve this problem?**

* Instead of trying to search the entire house, we can sub-divide the problem into smaller part by looking in each room separately
* We can further sub-divide the problem by looking at each piece of furniture individually
* At the end of the day we may have search all the house

Problem 2: sort the array A in ascending order

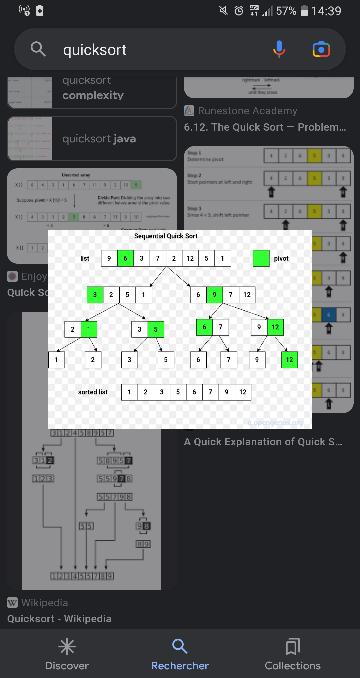
Here, to solve this problem using divide and conquer we can use two algorithm which are;

* Merge sort algorithm
* Quick sort algorithm

For this particular problem we are going to use quick sort

Pseudocode for quicksort is

* Select a pivot element from the array (this can be any element, but for simplicity, you can choose the first or the last element)
* Partition the array such that all elements smaller than the pivot are placed before it, and all elements larger than the pivot are placed after it.
* Recursively apply step 1 and 2 to the subarrays formed on the left and right sides of the pivot.
* Continue this process until the subarrays have only one element, which means the array is now sorted.



Worst case: The worst-case scenario occurs when the array is already sorted in ascending order so we have to waste time going through the algorithm again and setting pivot. This can lead to a runtime complexity of O(n^2)

Best case: The best-case scenario occurs when the pivot element divides the array into two sub-arrays of equal or nearly equal size. In this case the runtime complexity is O (n log n). Which is the same as the average case complexity.

**Advantages and limitations of divide and conquer algorithm**

**Advantages of divide and conquer algorithm**

* Efficiency: Divide and conquer algorithms often lead to efficient solutions by reducing the problem size. The subproblems can be solved concurrently or in a recursive manner, reducing the overall time complexity of the algorithm.
* Scalability: Divide and conquer algorithm are easily scalable. As the problem size increases, the algorithm can be divided into more subproblems, which can be solved independently. This makes it suitable for solving large-scale problems efficiently.
* Algorithmic design: Divide and conquer algorithm promotes modular design and can be easily implemented using recursion or iteration. It allows for clear separation of concerns, making the algorithm more maintainable and easier to understand.
* Parallelization: The independent nature of subproblems in divide and conquer algorithms allows for parallel execution. This makes it possible to harness the power of multiple processors or distributed computing systems, improving overall performance.
* Optimized solutions: Divide and conquer often lead to optimized solutions because they allow for the application of specific techniques applicable to subproblems. For example, sorting subarrays independently.

**Limitations of divide and conquer algorithm**

* Overhead: The use of divide and conquer algorithms may introduce additional overhead due to the recursive calls or the need to merge subproblem’s solutions. This overhead can impact the overall performance, especially for small problem sizes.
* Memory requirement: Divide and conquer may require additional memory to store intermediate results or subproblems solutions. This can be a concern when dealing with large problem sizes, as it may increase the memory consumption of the algorithm
* Complexity: The complexity of dividing the problem and merging the solutions can add complexity to the implementation and debugging process. The algorithm usually requires careful analysis and understanding of the problem to accurately split it into subproblems and combine the result.