Divide And Conquer

Main Focus: Introduction

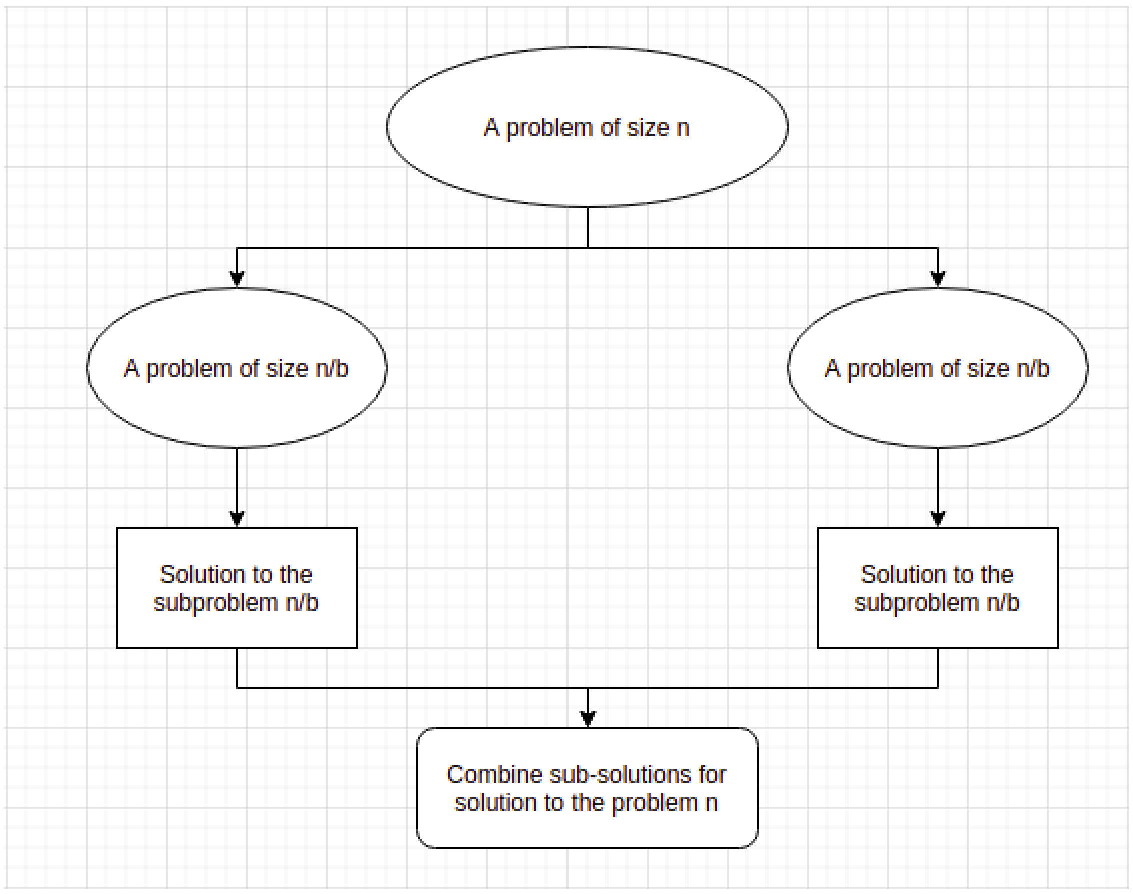
The Strategy:

The Divide And Conquer solves a problem by:

1. **Divide:** Breaking the problem into sub problems that are smaller instances of the same type of problem.
2. **Recursion:** Recursively solving these sub problems.
3. **Conquer:** Combining answers with the best case solution.

NOTE: It is must be understood that it is not possible to solve all problems using this method. D & C aims to solve questions by using recursion to solve the subproblems of the *same type*.

For better understanding I can use the following visualization:



A problem of size n
A problem of size n/b
A problem of size n/b
Solution to the subproblem n/b
Solution to the subproblem n/b
Combine sub-solutions for solution to the problem n

Master Theorem:

In Divide and Conquer, we solve the sub problems recursively. We use this theorem to determine the running time of the divide and conquer based algorithm we are using and implementing. For a given algorithm, we first try and find the recurrence relation of the problem. If the recurrence is of the form:

$T(n) = a \times T(n/b) + \Theta(n^k log^p n)$
where, $a \geq 1, b \geq 1, k \geq 0$ and $p \in R$. Then:

1. If $a > b^k$, then $T(n) = \Theta(n^{log_b^a})$
2. If $a = b^k$, then

   a. If $p > -1$, then $T(n) = \Theta(n^{log_b^a} log^{p+1} n)$.

   b. If $p = -1$, then $T(n) = \Theta(n^{log_b^a} loglogn)$.

   c. If $p < -1$, then $T(n) = \Theta(n^{log_b^a})$.
3. If $a < b^k$, then

   a. If $p \geq 0$, then $T(n) = \Theta(n^k log^p n)$

   b. If $p < 0$, then $T(n) = O(n^k)$

Example based: Merge sort algorithm, which operates on 2 sub-problems, each which is half the size of the original, and then performs(n) additional work for merging.

> $T(n) = 2 * T(n/2) + O(n)$

This gives us a more general equation using more variables to account for minute computational changes.

(The above formulas are just a set of rules and inferences which we can use in order to calculate the time complexity effeciently and quickly).