Divide and conquer

The divide-and-conquer strategy solves a problem by:

1. Breaking it into ***sub-problems*** that are themselves smaller instances of the same type of Problem
2. Recursively solving these sub-problems.
3. Appropriately combining their answers.

The real work is done in three different phase: the partitioning of problems(divide phase) into sub-problems; at the very tail end of the recursion, when the sub-problems are so small that they are solved outright(recursively solved small problem); and in the combining together of partial answers(conquer phase). These are held together and coordinated by the algorithm's core recursive structure.

Basics of divide and conquer

- Recursion
- Recurrence relation
- Recurrence relation solving

Recursion: The process in which a function calls itself directly or indirectly is called recursion and corresponding function is called as recursive function.

Recurrence relation: A Recurrence is an equation or inequality that describes a function in terms of its values on smaller inputs.

Example: T(n) = T(n-1) + 1

T(n) is defined in terms of T(n-1)

Recurrence relation solving

i. Substitution method
ii. Iteration method
    - Recursion-tree method
    - (Master method)

Backward substitution method

- Recurrence: T(n) = T(n-1) + 1, with initial condition T(1) = 2

Recurrence solving

  - T(n) = T(n-1) + 1
  - T(n) = [T(n-2)+1] + 1
  - T(n) = [[T(n-3)+1]+1] + 1
  - T(n) = [[[T(n-4)+1]+1]+1] + 1
  - ...
  - T(n) = [...[[T(n-k)+1]+1] ... +1] + 1 [has k ones]
  - ... [Let k = n-1]
  - T(n) = [...[[T(n-(n-1))+1]+1] ... +1] + 1 [has k=n-1 ones]
  - T(n) = T(n-(n-1)) + (n - 1)
  - T(n) = T(1) + (n-1) T(1) =2
  - T(n) = 2 + (n-1) = n-1

- This process leads to this Guess:
  - T(n) = n + 1

The master theorem

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

$T(n) = aT(n/b) + f(n)$,

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ can be bounded asymptotically as follows.

Case1: If $f(n)=O(n^{\log_b a - €})$ for some constant > 0, then $T(n)= \Theta(n^{\log_b a})$.

Case 2: If $f(n)= \Theta(n^{\log_b a})$, then $T(n)= \Theta(n^{\log_b a} \log n)$.

Case 3: If $f(n)=E(n^{\log_b a + €})$ for some constant > 0, and if $a f(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$.

In each of the three cases, we are comparing the function $f(n)$ with the function $n^{\log_b a}$. Intuitively, the solution to the recurrence is determined by the larger of the

two functions. If, as in case 1, the function $n^{\log_b a}$ is the larger, then the solution is $T(n)= \Theta(n^{\log_b a})$.If, as in case 3, the function $f(n)$ is the larger, then the solution is $T(n) = \Theta(f(n))$. If, as in case 2, the two functions are the same size, we multiply by a logarithmic factor, and the solution is $T(n)= \Theta(n^{\log_b a} \log n) = \Theta(f(n) \log n)$.

In the first case, not only must $f(n)$ be smaller than $n^{\log_b a}$, it must be *polynomially* smaller. That is, $f(n)$ must be asymptotically smaller than $n^{\log_b a}$ by a factor of $n\_$ for some constant > 0.
In the third case, not only must $f(n)$ be larger than $n^{\log_b a}$, it must be polynomially larger and in addition satisfy the "regularity" condition that $af(n/b) \leq cf(n)$. This condition is satisfied by most of the polynomially bounded functions that we shall encounter.
It is important to realize that the three cases do not cover all the possibilities for $f(n)$. There is a gap between cases 1 and 2 when $f(n)$ is smaller than $n^{\log_b a}$ but not polynomially smaller. Similarly, there is a gap between cases 2 and 3 when $f(n)$ is larger than $n^{\log_b a}$ but not polynomially larger. If the function $f(n)$ falls into one of these gaps, or if the regularity condition in case 3 fails to hold, **the master method cannot be used to solve the recurrence.**

Example: 1
Given $T(n) = 9T(n/3) + n$.
$a=9, b=3$
$f(n)=n$
calculate: $n^{\log_b a} = n^{\log_3 9} = n^2$

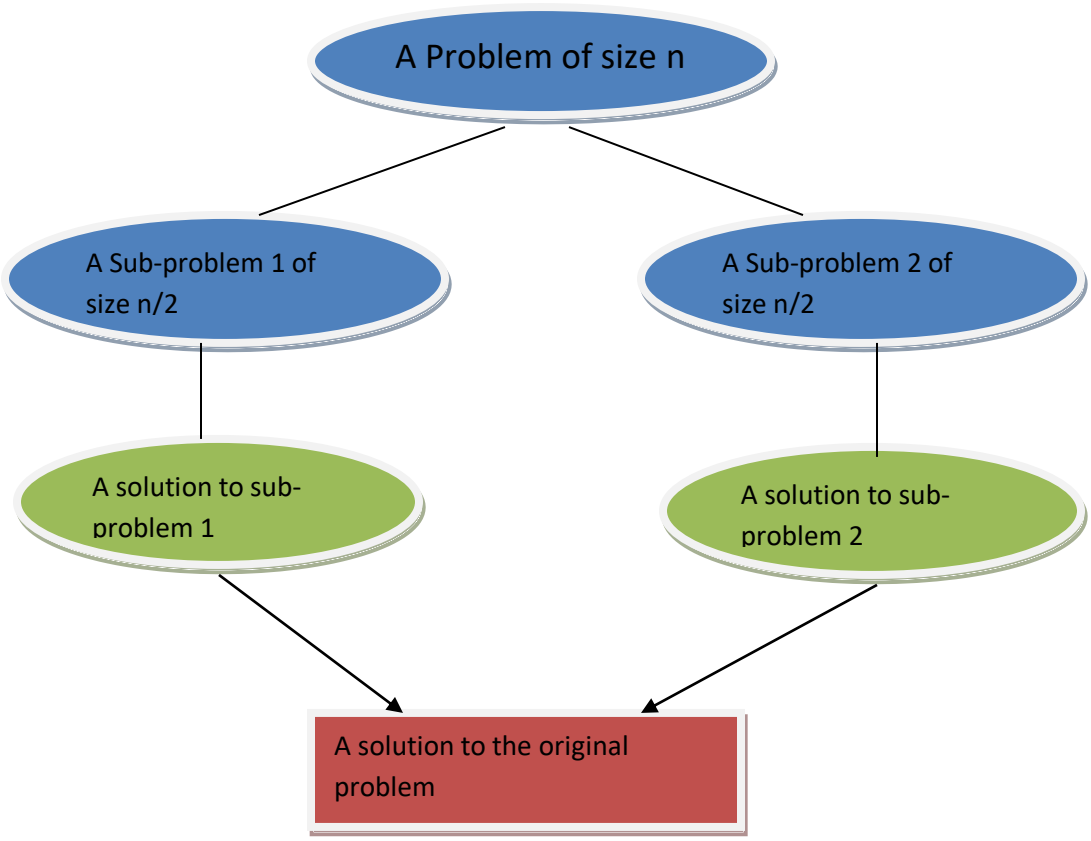
$f(n) < n^2$ which stratify Case 1: so $T(n) = \Theta(n^{\log_3 9}) = \Theta(n^2)$

Example: 2
Given $T(n) = T(2n/3) + 1$,
$a = 1, b = 3/2, f(n) = 1$
calculate: $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$ so $f(n) = n^{\log_{3/2} 1}$ are equal so case 2 arise and thus the solution to the recurrence is $T(n) = \Theta(\lg n)$.

Example: 3
Given $T(n) = 3T(n/4) + n \lg n$,
$a = 3, b = 4, f(n) = n \lg n$
calculate: $n^{\log_b a} = n^{\log_4 3} = n^{0.793}$ so $f(n) > n^{\log_4 3}$

so case 3 applies if we can show regularity condition holds for f(n.)
For sufficiently large n,
af (n/b) = 3(n/4)lg(n/4) ≤ (3/4)n lg n = cf (n) for c =s3/4. Consequently, by case 3, the solution to the recurrence is T(n) = Θ(nlg n).

Divide-and-conquer technique example



Divide and Conquer Examples

- Sorting: merge sort and quick sort
- Binary search
- Matrix multiplication: Strassen's algorithm
- Convex-hull algorithms