Divide and Conquer.

Divide and Conquer
- Binary Search.
- Powering a number.
- Fibonacci numbers

$T(n) = aT(\frac{n}{b}) + f(n)$

Master Theorem

Case 1:

$f(n) = O(n^{\log_b a - \varepsilon})$, $\varepsilon > 0$.

$\Rightarrow T(n) = \Theta(n^{\log_b a})$

Case 2:

$f(n) = O(n^{\log_b a} \lg^k n)$ $k \geq 0$

$\Rightarrow T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$

Case 3:

$f(n) = \Omega(n^{\log_b a + \varepsilon})$ $\varepsilon > 0$.

$\Rightarrow T(n) = \Theta(f(n))$

1. The Divide-and-Conquer Design Paradigm.

① Divide the Problem (instance) into subproblems.

② Conquer the subproblem by solving them recursively

③ Combine subproblem solutions.

2. Merge Sort.

1. Divided: Trivial
2. Conquer: Recursively sort 2 subarrays
3. Combine: Linear-time merge.

$T(n) = \underbrace{2}_{\text{subproblems.}} T(\underbrace{\tfrac{n}{2}}_{\text{subproblem size}}) + \Theta(n) \longleftarrow$ work dividing and combine.

$\Theta(n) = \Theta(n^{\log_2 2})$ $\quad T(n) = \Theta(n \lg n)$

3. Binary Search

Find an element in a sorted array

1. Divide: Check middle element.
2. Conquer: Recursively search 1 subarray.
3. Combine: Trivial.

$T(n) = T(\frac{n}{2}) + \Theta(1)$ $\quad \Theta(1) = \Theta(n^{\log_2 1})$

$T(n) = \Theta(n^{\log_2 1} \lg n) = \Theta(\lg n)$

Divide and Conquer.

4. Powering a number.

Problem: Compute a^n.

Naive algorithm: $\Theta(n)$

Divide - Conquer algorithm: $a^n \begin{cases} a^{\frac{n}{2}} \cdot a^{\frac{n}{2}} & \text{if } n \text{ is even} \\ a^{\frac{n-1}{2}} \, a^{\frac{n-1}{2}} \, a & n \text{ is odd} \end{cases}$

$$T(n) = T\left(\frac{n}{2}\right) + \Theta(1)$$

$$\Theta(1) = \Theta(n^{\log_2 1}) \Rightarrow T(n) = \Theta(\lg n).$$

5. Fibonacci numbers.

Definition:

$$F_n = \begin{cases} 0 & \text{if } n=0 \\ 1 & \text{if } n=1 \\ F_{n-1} + F_{n-2} & \text{if } n \geq 2 \end{cases}$$

1. Naive algorithm: $\Omega(\varphi^n)$ (exponential time), where $n = \frac{(1+\sqrt{5})}{2}$
2. Bottom-up: $\Theta(n)$.

   compute $F_0, F_1, \ldots F_n$ in order, forming each number by summing the two previous