# طراحی و تحلیل الگوریتم‌ها

## الگوریتم‌های تقسیم و غلبه

## Divide and Conquer

CLRS, Ch. 4

# Divide and Conquer

- **Divide** the problem into a number of subproblems that are smaller instances of the same problem.
- **Conquer** the subproblems by solving them recursively. If the subproblem sizes are small enough, however, just solve the subproblems in a straightforward manner. ( Ad-hoc )
- **Combine** the solutions to the subproblems into the solution for the original problem.

1

# Merge sort

چند مثال معروف از بکارگیری روش تقسیم و غلبه در طراحی الگوریتم

```
MERGE-SORT(A, p, r)
if p < r
    q = ⌊(p + r)/2⌋
    MERGE-SORT(A, p, q)
    MERGE-SORT(A, q + 1, r)
    MERGE(A, p, q, r)
```

$A[p, p+1, \ldots, r]$

# Merge sort

**Divide:** Divide the $n$-element sequence to be sorted into two subsequences of $n/2$ elements each.

**Conquer:** Sort the two subsequences recursively using "merge sort."

**Combine:** Merge the two sorted subsequences to produce the sorted answer.

②

# Quick sort

QUICKSORT$(A, p, r)$

```
1  if p < r
2      q = PARTITION(A, p, r)
3      QUICKSORT(A, p, q − 1)
4      QUICKSORT(A, q + 1, r)
```

$A[p, p+1, \ldots, r]$ → $A[p \ldots (q-1)]$

→ $A[(q+1) \cdots r]$

$A[q]$

# Quick sort

**Divide:** Partition (rearrange) the array $A[p \mathinner{..} r]$ into two (possibly empty) subarrays $A[p \mathinner{..} q-1]$ and $A[q+1 \mathinner{..} r]$ such that each element of $A[p \mathinner{..} q-1]$ is less than or equal to $A[q]$, which is, in turn, less than or equal to each element of $A[q+1 \mathinner{..} r]$. Compute the index $q$ as part of this partitioning procedure.

**Conquer:** Sort the two subarrays $A[p \mathinner{..} q-1]$ and $A[q+1 \mathinner{..} r]$ by recursive calls to "quicksort."

**Combine:** Because the subarrays are already sorted, no work is needed to combine them: the entire array $A[p \mathinner{..} r]$ is now sorted.

3

# Binary search

فرض کنید آرایه A از قبل به ترتیب صعودی مرتب شده است. حال به دنبال یک مقدار x در A می گردیم.

```
Bsearch( A, p, r, x)
  if p<r then
    q = [(p+r)/2]
    if x < A[q] then
      return Bsearch(A,p,q  ,x)
    elseif x > A[q] then
      return Bsearch(A,q+1,r,x)
    else
      return q
  return "not found"
```

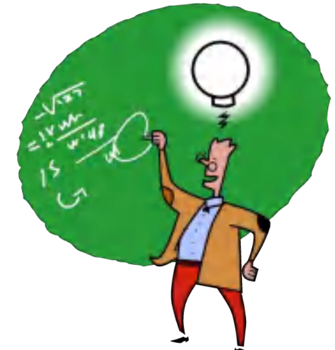
$x \in A[p, p+1, \dots, r-1]$ ?

# Master Theorem

in Recursive Functions

قضیه اساسی در توابع بازگشتی

# Master Method (Appendix)



- Many divide-and-conquer recurrence equations have the form:

هزینه زمان اجرای الگوریتم

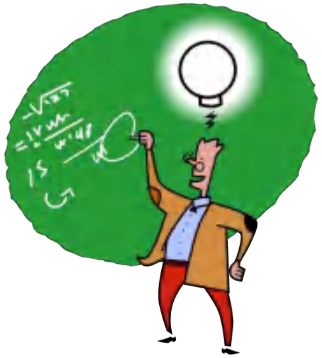
$$T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \ge d \end{cases}$$

$\beta := \log_b a$

- The Master Theorem:

1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$
3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$,
   provided $af(n/b) \le \delta f(n)$ for some $\delta < 1$.

# Master Method, Example 1



- The form: $$T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \geq d \end{cases}$$
- The Master Theorem:

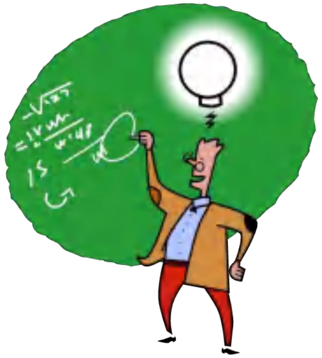
  1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
  2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$
  3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$, provided $af(n/b) \leq \delta f(n)$ for some $\delta < 1$.

- Example:

$$T(n) = 4T(n/2) + n$$

Solution: $\log_b a = 2$, so $\boxed{\text{case 1}}$ says $T(n)$ is $\Theta(n^2)$.

# Master Method, Example 2



- The form: $T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \geq d \end{cases}$
- The Master Theorem:

  1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
  2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$
  3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$,

     provided $af(n/b) \leq \delta f(n)$ for some $\delta < 1$.

- Example:

$$T(n) = \underbrace{2}T(\underbrace{n/2}) + \underbrace{n \log n}$$

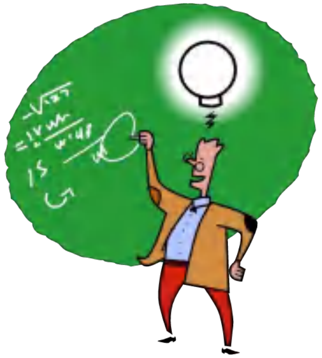
$\beta = 1$

$n \log n = \Theta(n^{\beta} \log^{k} n)$

$k = 1$

Solution: $\log_b a = 1$, so $\boxed{\text{case 2}}$ says $\underline{T(n) \text{ is } \Theta(n \log^2 n)}$. ✓

# Master Method, Example 3



- The form: $T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \ge d \end{cases}$
- The Master Theorem:
  1. if $f(n)$ is $O(n^{\log_b a-\varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
  2. if $f(n)$ is $\Theta(n^{\log_b a}\log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a}\log^{k+1} n)$
  3. if $f(n)$ is $\Omega(n^{\log_b a+\varepsilon})$, then $T(n)$ is $\Theta(f(n))$,
     provided $af(n/b) \le \delta f(n)$ for some $\delta < 1$.

$\beta = \log_3 1 = 0$

$n^0 = \underline{1}$

✓

$n \log n = \Omega(n^{\varepsilon})$

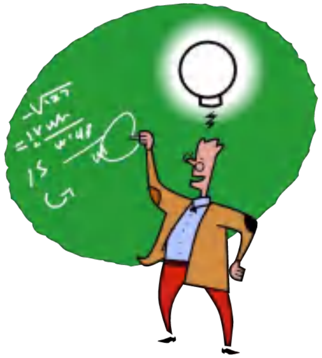
- Example:

$$T(n) = T(n/3) + n\log n$$

$f(n) = \underline{\underline{n \log n}}$

Solution: $\log_b a=0$, so $\boxed{\text{case 3}}$ says T(n) is Θ(n log n).

# Master Method, Example 4



- The form: $T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \geq d \end{cases}$
- The Master Theorem:

  1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
  2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$
  3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$, provided $af(n/b) \leq \delta f(n)$ for some $\delta < 1$.

- Example:

$$T(n) = 8T(n/2) + n^2$$

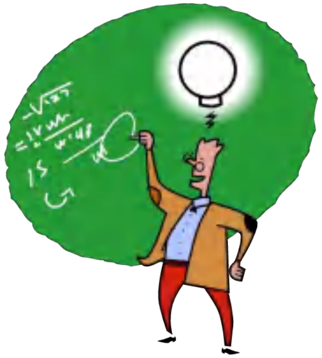
$a=8$ $\beta=3$

$b=2$

$f(n)=n^2$

Solution: $\log_b a=3$, so $\boxed{\text{case 1}}$ says $\underline{T(n) \text{ is } \Theta(n^3)}$.

# Master Method, Example 5



- The form: $T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \ge d \end{cases}$
- The Master Theorem:
  1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
  2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$
  3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$, provided $af(n/b) \le \delta f(n)$ for some $\delta < 1$.
- Example:

$a=9$ $\beta=2$

$b=3$

$$T(n) = \underline{9}T(\underline{n/3}) + n^3 \qquad f(n)=\underline{n^3}$$

Solution: $\log_b a=2$, so $\boxed{\text{case 3}}$ says $\underline{T(n) \text{ is } \Theta(n^3)}$.

## فرم ساده شده قضیه اساسی

$$T(n)=\begin{cases} \text{Const} & \text{if } n<d \\ aT(n/b)+n^k & \text{if } n\geqslant d \end{cases}$$

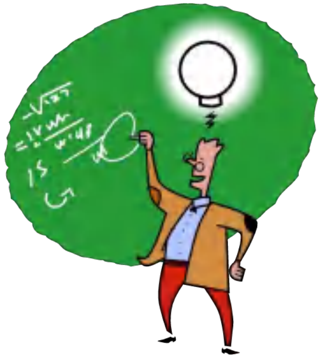
$\underline{f(n)=n^k}$ ، قرار دهید $\beta := \log_b a$

case 1. if $\beta > k \Rightarrow T(n)=\Theta(n^{\beta})$

case 2. if $\beta = k \Rightarrow T(n)=\Theta(n^{\beta}\log n)$

case 3. if $\beta < k \Rightarrow T(n)=\Theta(n^{k})$

# Master Method, Example 6



- The form: $T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \ge d \end{cases}$

$a = 1$

$b = 2$

- The Master Theorem:

1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$
3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$,

   provided $af(n/b) \le \delta f(n)$ for some $\delta < 1$.

$\beta = \log_b a = 0$

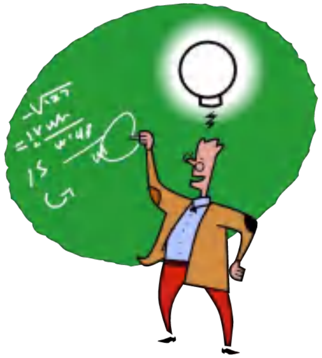
- Example:

$n^0 = 1 = f(n) \qquad f(n) = \underline{\underline{1}}$

$$T(n) = T(\underline{n/2}) + \underline{\underline{1}} \quad \text{(binary search)}$$

Solution: $\log_b a = 0$, so $\boxed{\text{case 2}}$ says $\underline{T(n) \text{ is } \Theta(\log n)}$.

# Master Method, Example 7



- The form: $$T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \geq d \end{cases}$$

$a = 2$

$b = 2$

- The Master Theorem:
  1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
  2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$
  3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$, provided $af(n/b) \leq \delta f(n)$ for some $\delta < 1$.

$\beta = \log_b a = 1$

$\exists \varepsilon > 0$

✓

$\log n = O(n^{1-\varepsilon})$

- Example:

$$T(n) = 2T(n/2) + \log n \quad \text{(heap construction)}$$

Solution: $\log_b a = 1$, so $\boxed{\text{case 1}}$ says $\boxed{T(n) \text{ is } O(n)}$.