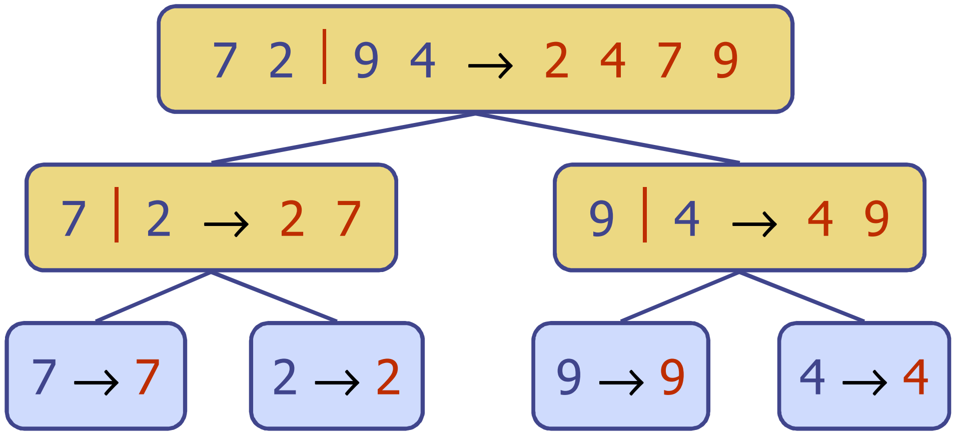
More About Analysis of Algorithms

Topics :

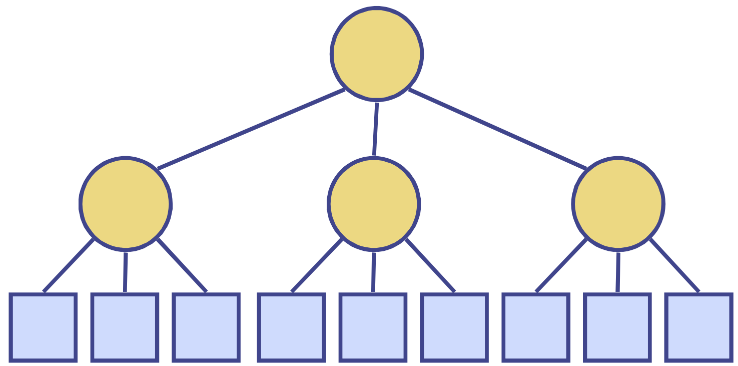
- Divide-and Conquer
- Recurrence relations
- Master Theorem

Divide-and-Conquer



Divide-and-Conquer

- Divide-and conquer is a general algorithm design paradigm:
  - Divide: divide the input data S in two or more disjoint subsets S_1, S_2, …
  - Recur: solve the subproblems recursively
  - Conquer: combine the solutions for S_1, S_2, …, into a solution for S
- The base case for the recursion are subproblems of constant size
- Analysis can be done using **recurrence equations**



Merge-Sort

- Merge-sort on an input sequence S with n elements consists of three steps:
  - Divide: partition S into two sequences S_1 and S_2 of about $n/2$ elements each
  - Recur: recursively sort S_1 and S_2
  - Conquer: merge S_1 and S_2 into a unique sorted sequence

Algorithm ***mergeSort***(S, C)
    Input sequence S with n elements, comparator C
    Output sequence S sorted according to C
    if $S.size() > 1$
        $(S_1, S_2) \leftarrow partition(S, n/2)$
        $mergeSort(S_1, C)$
        $mergeSort(S_2, C)$
        $S \leftarrow merge(S_1, S_2)$

Recurrence Equation Analysis

- The conquer step of merge-sort consists of merging two sorted sequences, each with $n/2$ elements and implemented by means of a doubly linked list, takes at most bn steps, for some constant b.
- Likewise, the basis case ($n < 2$) will take at b most steps.
- Therefore, if we let $T(n)$ denote the running time of merge-sort:

$$T(n) = \begin{cases} b & \text{if } n < 2 \\ 2T(n/2) + bn & \text{if } n \geq 2 \end{cases}$$

- We can therefore analyze the running time of merge-sort by finding a **closed form solution** to the above equation.
  - That is, a solution that has $T(n)$ only on the left-hand side.

Iterative Substitution

- In the iterative substitution, or "plug-and-chug," technique, we iteratively apply the recurrence equation to itself and see if we can find a pattern:

$$\begin{aligned} T(n) &= 2T(n/2) + bn \\ &= 2(2T(n/2^2)) + b(n/2)) + bn \\ &= 2^2 T(n/2^2) + 2bn \\ &= 2^3 T(n/2^3) + 3bn \\ &= 2^4 T(n/2^4) + 4bn \\ &= \ldots \\ &= 2^i T(n/2^i) + ibn \end{aligned}$$

- Note that base, T(n)=b, case occurs when 2^i=n. That is, i = log n.
- So,

$$T(n) = bn + bn\log n$$

- Thus, T(n) is O(n log n).

Changing variables

$$T(n) = 2T(\sqrt{n}) + 1$$

- Let $n = 2^m$

=> $sqrt(n) = 2^{m/2}$

- We then have $T(2^m) = T(2^{m/2}) + 1$
- Let $T(n) = T(2^m) = S(m)$

=> $S(m) = S(m/2) + 1$

$\Rightarrow S(m) = \Theta\ (\log m) = \Theta\ (\log \log n)$

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

Changing variables

$$\boldsymbol{T(n) = 2T(n\text{-}1) + 1}$$

- Let $n = log\ m$, i.e., $m = 2^n$

=> $T(log\ m) = 2\ T(log\ m/2) + 1$

- Let $S(m) = T(log\ m)$

=> $S(m) = 2S(m/2) + 1$

=> $S(m) = \Theta(m)$

=> $T(n) = S(m) = \Theta(m) = \Theta(2^n)$

Changing variables

- $T(n) = 2T(n-2) + 1$
- Let $n = \log m$, i.e., $m = 2^n$

$\Rightarrow T(\log m) = 2\, T(\log m/4) + 1$

- Let $S(m) = T(\log m)$

$\Rightarrow S(m) = 2S(m/4) + 1$

$\Rightarrow S(m) = m^{1/2}$

$\Rightarrow T(n) = S(m) = 2^{n/2} = (sqrt(2))^{n} = 1.4^{n}$

Obtaining bounds

- $T(n) = T(n/2) + \log n$
- $T(n) \in \Omega(\log n)$
- $T(n) \in O(T(n/2) + n^{\varepsilon})$
- Solving $T(n) = T(n/2) + n^{\varepsilon}$, we obtain $T(n) = n^{\varepsilon}$, for any $\varepsilon > 0$
- So: $T(n) \in O(n^{\varepsilon})$ therefore $T(n)$ is unlikely polynomial.
- In fact $T(n) = \Theta(\log^2 n)$

Obtaining bounds

- $T(n) = T(n-1) + T(n-2) + 1$
- $T(n) >= 2(T-2) + 1$ [1]
- $T(n) <= 2T(n-1) + 1$ [2]

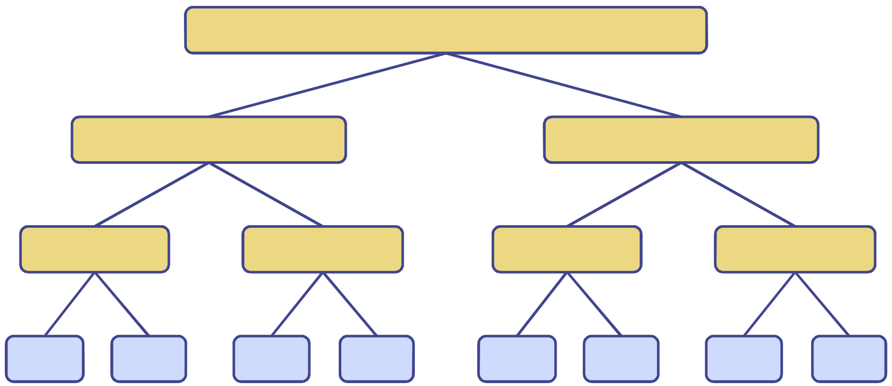
- Solving [1], we obtain $T(n) >= 1.4^n$
- Solving [2], we obtain $T(n) <= 2^n$
- In fact, $T(n) = 1.62^n$

The Recursion Tree

- Draw the recursion tree for the recurrence relation and look for a pattern:

$$T(n) = \begin{cases} b & \text{if } n < 2 \\ 2T(n/2) + bn & \text{if } n \geq 2 \end{cases}$$

depth	T's	size	time
0	1	$\boldsymbol{n}$	$\boldsymbol{bn}$
1	2	$\boldsymbol{n}/2$	$\boldsymbol{bn}$
$\boldsymbol{i}$	2^i	$\boldsymbol{n}/2^i$	$\boldsymbol{bn}$
…	…	…	…



Total time = $\boldsymbol{bn} + \boldsymbol{bn} \log \boldsymbol{n}$

(last level plus all previous levels)

Guess-and-Test Method

- In the guess-and-test method, we guess a closed form solution and then try to prove it is true by induction:

$$T(n) = \begin{cases} b & \text{if } n < 2 \\ 2T(n/2) + bn\log n & \text{if } n \geq 2 \end{cases}$$

- Guess: T(n) < cn log n.

$$\begin{aligned} T(n) &= 2T(n/2) + bn\log n \\ &= 2(c(n/2)\log(n/2)) + bn\log n \\ &= cn(\log n - \log 2) + bn\log n \\ &= cn\log n - cn + bn\log n \end{aligned}$$

- Wrong: we cannot make this last line be less than cn log n

Guess-and-Test Method, Part 2

- Recall the recurrence equation:

$$T(n) = \begin{cases} b & \text{if } n < 2 \\ 2T(n/2) + bn\log n & \text{if } n \geq 2 \end{cases}$$

- Guess #2: T(n) < cn $\log^2$ n.

$$\begin{aligned} T(n) &= 2T(n/2) + bn\log n \\ &= 2(c(n/2)\log^2(n/2)) + bn\log n \\ &= cn(\log n - \log 2)^2 + bn\log n \\ &= cn\log^2 n - 2cn\log n + cn + bn\log n \\ &\leq cn\log^2 n \end{aligned}$$

  - if c > b.
- So, T(n) is O(n $\log^2$ n).
- In general, to use this method, you need to have a good guess and you need to be good at induction proofs.

Master Method

- Many divide-and-conquer recurrence equations have 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:

$$1.\ \text{if } f(n) \text{ is } O(n^{\log_b a - \varepsilon}), \text{ then } T(n) \text{ is } \Theta(n^{\log_b a})$$

$$2.\ \text{if } f(n) \text{ is } \Theta(n^{\log_b a} \log^k n), \text{ then } T(n) \text{ is } \Theta(n^{\log_b a} \log^{k+1} n)$$

$$3.\ \text{if } f(n) \text{ is } \Omega(n^{\log_b a + \varepsilon}), \text{ then } T(n) \text{ is } \Theta(f(n)),$$

$$\text{provided } af(n/b) \leq \delta f(n) \text{ for some } \delta < 1.$$ for all n≥d

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 $\delta < 1$. for all n≥d
- Example: $$T(n) = 4T(n/2) + n$$

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

Master Method, Example 2

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

$$1.\ \text{if } f(n) \text{ is } O(n^{\log_b a - \varepsilon}), \text{ then } T(n) \text{ is } \Theta(n^{\log_b a})$$

$$2.\ \text{if } f(n) \text{ is } \Theta(n^{\log_b a} \log^k n), \text{ then } T(n) \text{ is } \Theta(n^{\log_b a} \log^{k+1} n)$$

$$3.\ \text{if } f(n) \text{ is } \Omega(n^{\log_b a + \varepsilon}), \text{ then } T(n) \text{ is } \Theta(f(n)),$$

$$\text{provided } af(n/b) \le \delta f(n) \text{ for some } \delta < 1.$$ for all n≥d

- Example:

$$T(n) = 2T(n/2) + n \log n$$

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

Master Method, Example 3

- The form: $$T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \geq d \end{cases}$$
- 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 $\delta < 1$. for all n≥d
- Example:

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

Solution: $\log_b a=0$, so 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:

$$1.\ \text{if } f(n) \text{ is } O(n^{\log_b a - \varepsilon}), \text{ then } T(n) \text{ is } \Theta(n^{\log_b a})$$

$$2.\ \text{if } f(n) \text{ is } \Theta(n^{\log_b a} \log^k n), \text{ then } T(n) \text{ is } \Theta(n^{\log_b a} \log^{k+1} n)$$

$$3.\ \text{if } f(n) \text{ is } \Omega(n^{\log_b a + \varepsilon}), \text{ then } T(n) \text{ is } \Theta(f(n)),$$

$$\text{provided } af(n/b) \leq \delta f(n) \text{ for some } \delta < 1.$$ for all n≥d

- Example:

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

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

Master Method, Example 5

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

$$1.\ \text{if } f(n) \text{ is } O(n^{\log_b a - \varepsilon}), \text{ then } T(n) \text{ is } \Theta(n^{\log_b a})$$

$$2.\ \text{if } f(n) \text{ is } \Theta(n^{\log_b a} \log^k n), \text{ then } T(n) \text{ is } \Theta(n^{\log_b a} \log^{k+1} n)$$

$$3.\ \text{if } f(n) \text{ is } \Omega(n^{\log_b a + \varepsilon}), \text{ then } T(n) \text{ is } \Theta(f(n)),$$

$$\text{provided } af(n/b) \leq \delta f(n) \text{ for some } \delta < 1.$$ for all n≥d

- Example:

$$T(n) = 9T(n/3) + n^3$$

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

Master Method, Example 6

- The form: $$T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \ge d \end{cases}$$
- 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 $\delta < 1$. for all n≥d
- Example:

$$T(n) = T(n/2) + 1$$ (binary search)

Solution: $\log_b a=0$, so case 2 says T(n) is Θ(log n).

Master Method, Example 7

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

$$1.\ \text{if } f(n) \text{ is } O(n^{\log_b a - \varepsilon}), \text{ then } T(n) \text{ is } \Theta(n^{\log_b a})$$

$$2.\ \text{if } f(n) \text{ is } \Theta(n^{\log_b a} \log^k n), \text{ then } T(n) \text{ is } \Theta(n^{\log_b a} \log^{k+1} n)$$

$$3.\ \text{if } f(n) \text{ is } \Omega(n^{\log_b a + \varepsilon}), \text{ then } T(n) \text{ is } \Theta(f(n)),$$

$$\text{provided } af(n/b) \leq \delta f(n) \text{ for some } \delta < 1.$$ for all n≥d

- Example:

$$T(n) = 2T(n/2) + \log n$$ (heap construction)

Solution: $\log_b a=1$, so case 1 says T(n) is $\Theta(n)$.

Master Method, Example 8

- 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 $\delta < 1$. for all n≥d
- Example:

$$T(n) = 3T(n/4) + n \lg n$$

Solution: $a = 3$, $b=4$, thus $n^{\log_b a} = n^{\log_4 3} = O(n^{0.793})$

$f(n) = n \lg n = \Omega(n^{\log_4 3} + \varepsilon)$ where $\varepsilon \approx 0.2 \Rightarrow$ **Case 3**.

Therefore, $T(n) = \Theta(f(n)) = \Theta(n \lg n)$.

Iterative "Proof" of the Master Theorem

- Using iterative substitution, let us see if we can find a pattern:

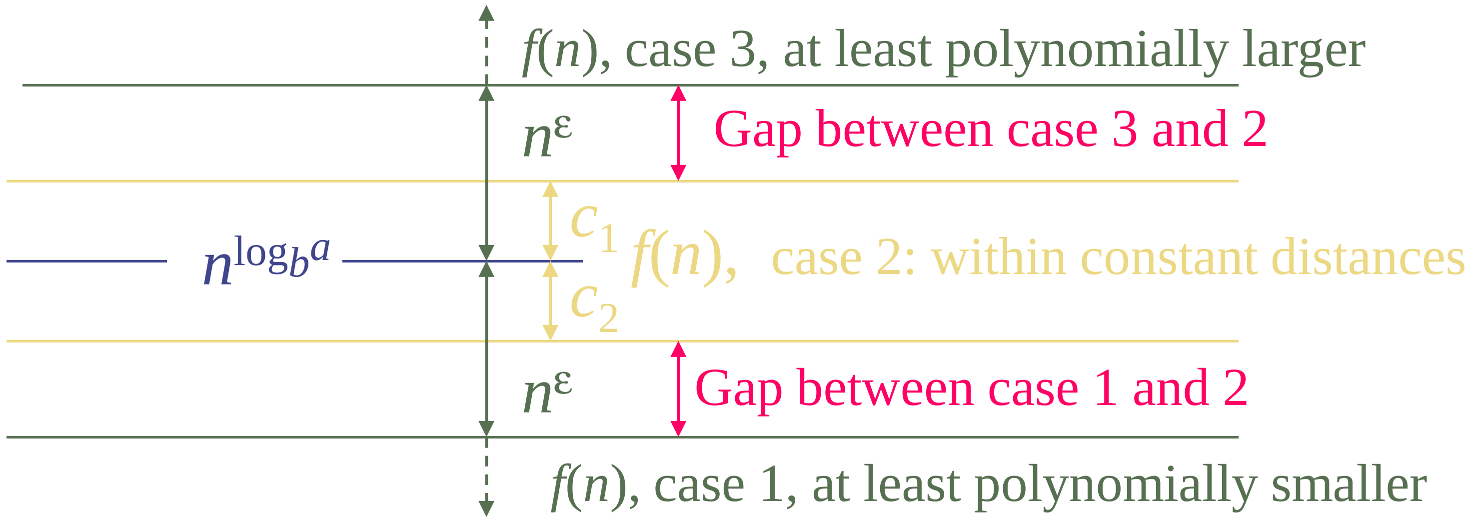
$$
\begin{aligned}
T(n) &= aT(n/b) + f(n) \\
&= a(aT(n/b^2)) + f(n/b)) + bn \\
&= a^2T(n/b^2) + af(n/b) + f(n) \\
&= a^3T(n/b^3) + a^2 f(n/b^2) + af(n/b) + f(n) \\
&= \ldots \\
&= a^{\log_b n}T(1) + \sum_{i=0}^{(\log_b n)-1} a^i f(n/b^i) \\
&= n^{\log_b a}T(1) + \sum_{i=0}^{(\log_b n)-1} a^i f(n/b^i)
\end{aligned}
$$

- We then distinguish the three cases as
  - The first term is dominant
  - Each part of the summation is equally dominant
  - The summation is a geometric series

Exception to Master Theorem

- $T(n) = 2T(n/2) + n\lg n$;
  - $a=2, b=2, f(n) = n\lg n$
  - $n^{\log_b a} = n^{\log_2 2} = \Theta(n)$
  - $f(n)$ is asymptotically larger than $n^{\log_b a}$, but not polynomially larger because
  - $f(n)/n^{\log_b a} = \lg n$, which is asymptotically less than n^{ε} for any $\varepsilon>0$.
  - Therefore,this is a gap between 2 and 3.

Where Are the Gaps



Note: 1. for case 3, the regularity also must hold.

2. if $f(n)$ is $\lg n$ smaller, then fall in gap in 1 and 2

3. if $f(n)$ is $\lg n$ larger, then fall in gap in 3 and 2

4. if $f(n)=\Theta(n^{\log_b a}\lg^k n)$, then $T(n)=\Theta(n^{\log_b a}\lg^{k+1} n)$. (as exercise)

The simple format of master theorem

- $T(n)=aT(n/b)+cn^k$, with a, b, c, k are positive constants, and $a\geq 1$ and $b\geq 2$,
- $$T(n) = \begin{cases} O(n^{\log_b a}), & \text{if } a>b^k. \\ O(n^k \log n), & \text{if } a=b^k. \\ O(n^k), & \text{if } a<b^k. \end{cases}$$