Recurrences and the Complexity of Divide and Conquer Algorithms

Antonio Carzaniga

Faculty of Informatics
Università della Svizzera italiana

October 12, 2010

Outline

- Analysis of recurrence expressions

Complexity Analysis

- **MergeSort**

- **MergeSort**
  - splits the n-long sequence
  - in 2 sub-sequences of size $n/2$
  - then combines the partial results in $\Theta(n)$

- **MergeSort**
  - splits the n-long sequence
  - in 2 sub-sequences of size $n/2$
  - then combines the partial results in $\Theta(n)$

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

- **MergeSort**
  - splits the n-long sequence
  - in 2 sub-sequences of size $n/2$
  - then combines the partial results in $\Theta(n)$

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

- We figured the complexity of **MergeSort** is $\Theta(n \log n)$

- **MergeSort**
  - splits the n-long sequence
  - in 2 sub-sequences of size $n/2$
  - then combines the partial results in $\Theta(n)$

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

- We figured the complexity of **MergeSort** is $\Theta(n \log n)$
  - is this right?
  - can we generalize?

Recurrence

- *Generic divide-and-conquer algorithm*

- *Generic divide-and-conquer algorithm*

  Base case:

  - solve a problem P of size ***1*** immediately, in $\Theta(1)$ steps

- *Generic divide-and-conquer algorithm*

  Base case:

  - solve a problem P of size ***1*** immediately, in $\Theta(1)$ steps

  Recursive case:

  - divide a problem P of size $n > 1$

- *Generic divide-and-conquer algorithm*

  Base case:

  - solve a problem P of size ***1*** immediately, in $\Theta(1)$ steps

  Recursive case:

  - divide a problem P of size $n > 1$
  - into a sub-problems of the same kind P

- *Generic divide-and-conquer algorithm*

  Base case:

  - solve a problem P of size ***1*** immediately, in $\Theta(1)$ steps

  Recursive case:

  - divide a problem P of size $n > 1$
  - into a sub-problems of the same kind P
  - each sub-problem is of size n/b

Recurrence

- *Generic divide-and-conquer algorithm*

  Base case:

  - solve a problem P of size ***1*** immediately, in $\Theta(1)$ steps

  Recursive case:

  - divide a problem P of size $n > 1$
  - into a sub-problems of the same kind P
  - each sub-problem is of size n/b
  - combine the solutions to the subproblems, in $f(n)$ steps

- *Generic divide-and-conquer algorithm*

  Base case:

  - solve a problem P of size ***1*** immediately, in $\Theta(1)$ steps

  Recursive case:

  - divide a problem P of size $n > 1$
  - into a sub-problems of the same kind P
  - each sub-problem is of size n/b
  - combine the solutions to the subproblems, in $f(n)$ steps

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ aT(n/b) + f(n) & \text{if } n > 1 \end{cases}$$

- *Generic divide-and-conquer algorithm*

  Base case:

  - solve a problem P of size ***1*** immediately, in $\Theta(1)$ steps

  Recursive case:

  - divide a problem P of size $n > 1$
  - into a sub-problems of the same kind P
  - each sub-problem is of size n/b
  - combine the solutions to the subproblems, in $f(n)$ steps

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ aT(n/b) + f(n) & \text{if } n > 1 \end{cases}$$

- Our goal is to obain a closed-form formula for $T(n)$

Master Theorem

- We then assume for simplicity that $f(n) = O(n^d)$ with $d \geq 0$

- We then assume for simplicity that $f(n) = O(n^d)$ with $d \geq 0$

- **Theorem:**

- We then assume for simplicity that $f(n) = O(n^d)$ with $d \geq 0$

- **Theorem:**

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ aT(n/b) + O(n^d) & \text{if } n > 1 \end{cases}$$

- We then assume for simplicity that $f(n) = O(n^d)$ with $d \geq 0$

- **Theorem:**

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ aT(n/b) + O(n^d) & \text{if } n > 1 \end{cases}$$

with $a > 0$, $b > 1$, $d \geq 0$ the asymptotic complexity is

$$T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a \end{cases}$$

Divide-and-Conquer Tree

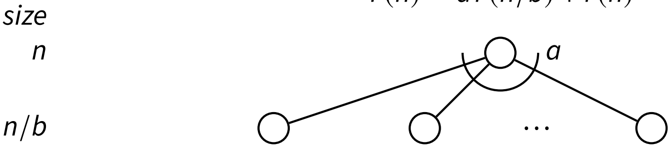
$$T(n) = aT(n/b) + f(n)$$

size

n

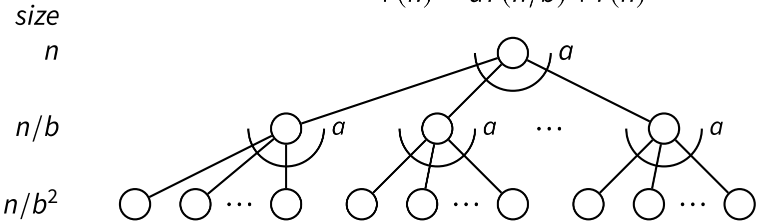
Divide-and-Conquer Tree

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



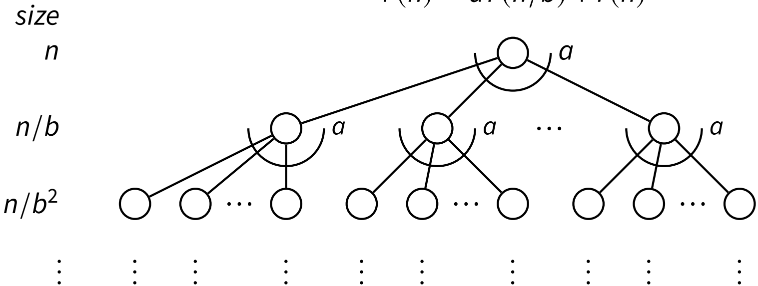
Divide-and-Conquer Tree

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



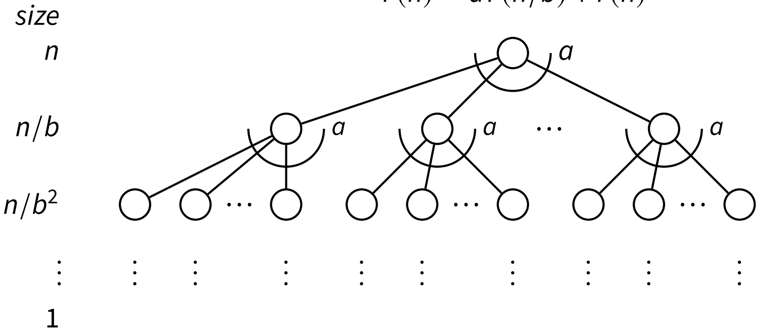
Divide-and-Conquer Tree

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

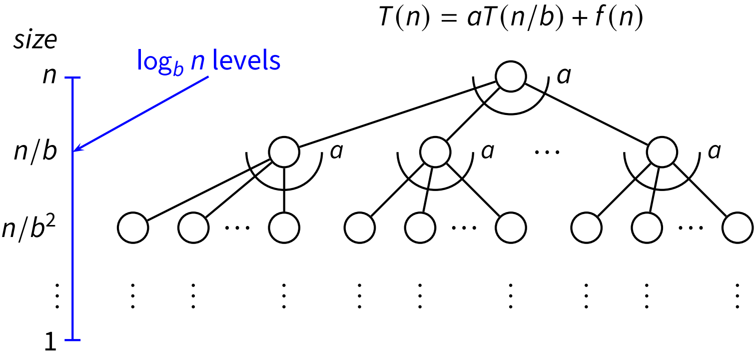


Divide-and-Conquer Tree

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

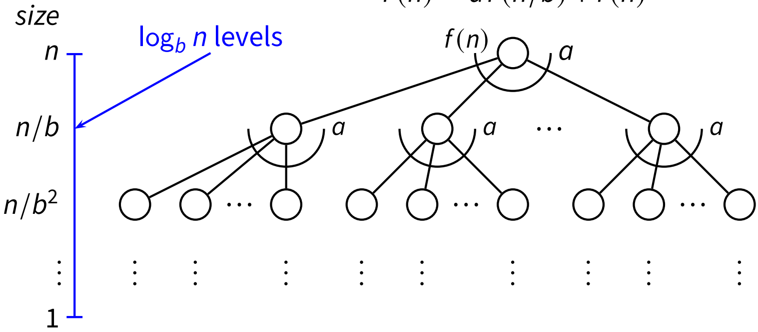


Divide-and-Conquer Tree

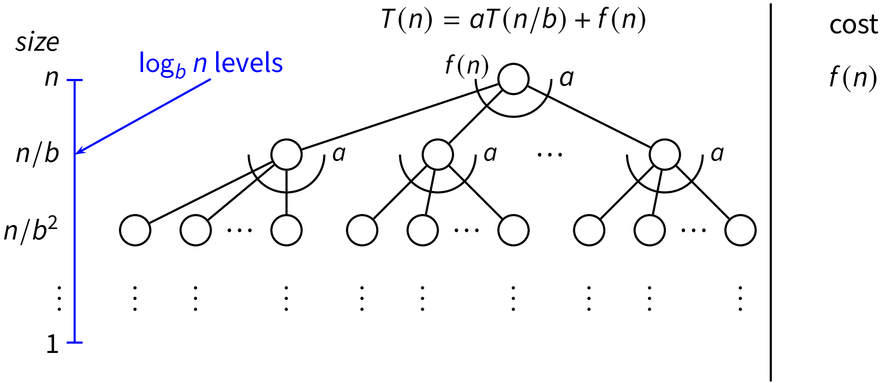


Divide-and-Conquer Tree

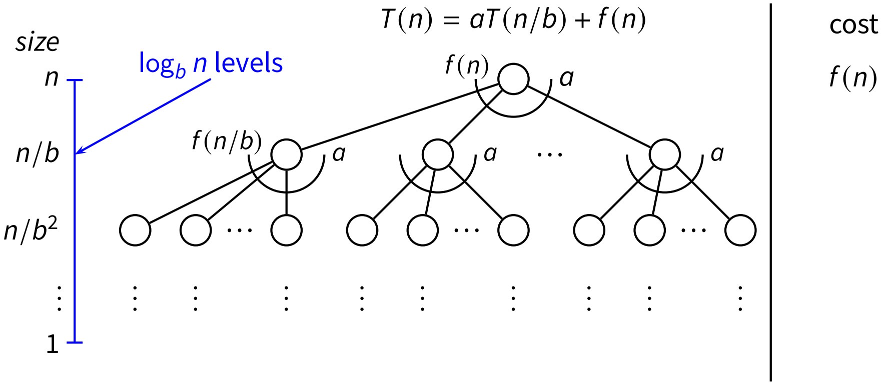
$$T(n) = aT(n/b) + f(n)$$



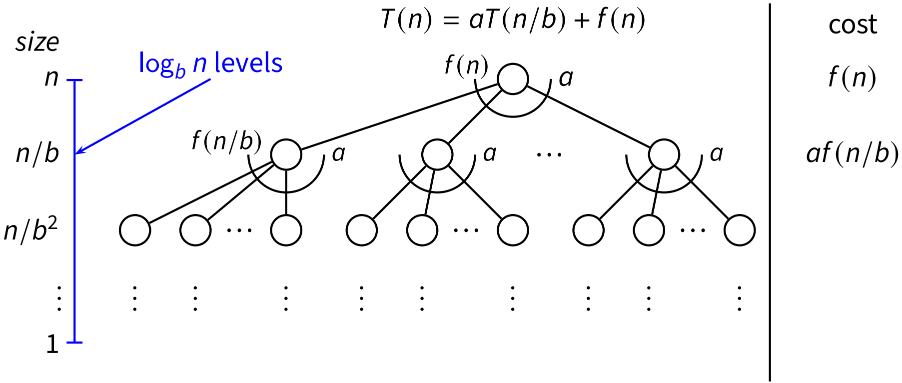
Divide-and-Conquer Tree



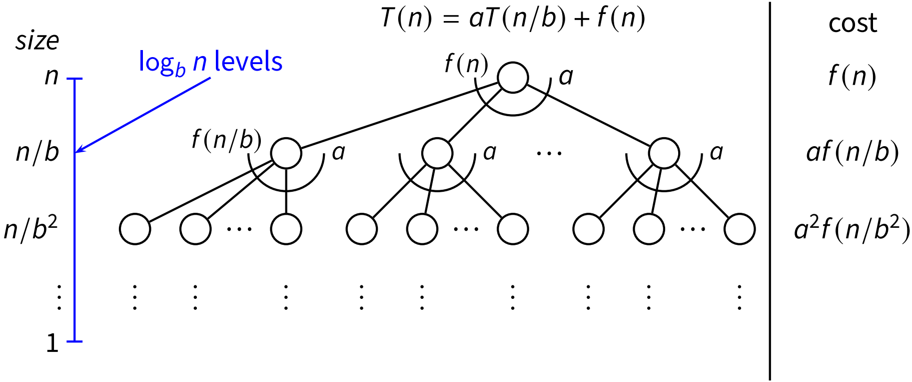
Divide-and-Conquer Tree



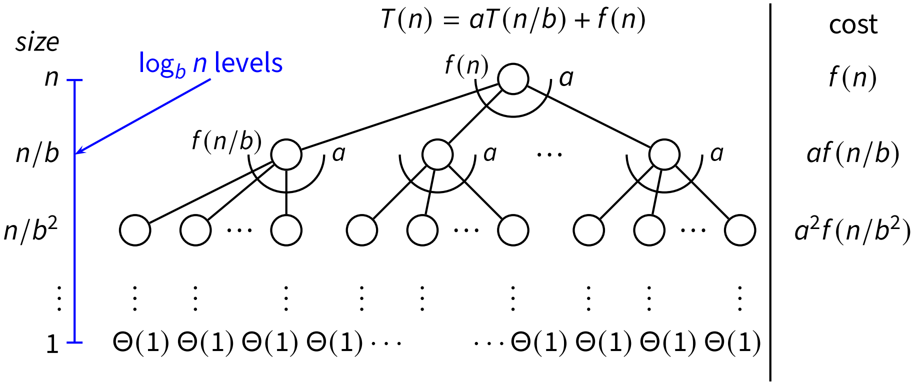
Divide-and-Conquer Tree



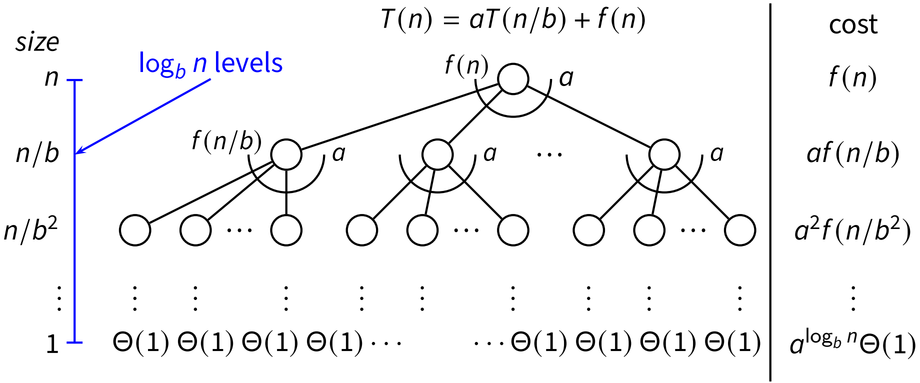
Divide-and-Conquer Tree



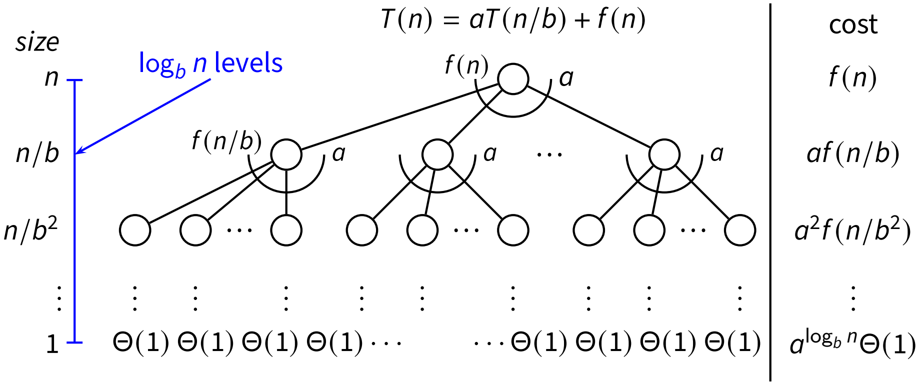
Divide-and-Conquer Tree



Divide-and-Conquer Tree



Divide-and-Conquer Tree



$$T(n) = \Theta(n^{\log_b a}) + \sum_{i=0}^{\log_b n - 1} a^i f(n/b^i)$$

Divide-and-Conqer Complexity

$$T(n) = \Theta(n^{\log_b a}) + \sum_{i=0}^{\log_b n - 1} a^i f(n/b^i)$$

- Basic assumptions

$$a \geq 1, b > 1$$

Divide-and-Conqer Complexity

$$T(n) = \Theta(n^{\log_b a}) + \sum_{i=0}^{\log_b n - 1} a^i f(n/b^i)$$

- Basic assumptions

$$a \geq 1, b > 1$$

- Further assumption

$$f(n) = O(n^d) \text{ with } d \geq 0$$

Divide-and-Conqer Complexity

$$T(n) = \Theta(n^{\log_b a}) + \sum_{i=0}^{\log_b n - 1} a^i f(n/b^i)$$

- Basic assumptions

$$a \geq 1, b > 1$$

- Further assumption

$$f(n) = O(n^d) \text{ with } d \geq 0$$

- So

$$T(n) = \Theta(n^{\log_b a}) + \sum_{i=0}^{\log_b n - 1} a^i O(n^d / b^{id})$$

Divide-and-Conqer Complexity

$$T(n) = \Theta(n^{\log_b a}) + \sum_{i=0}^{\log_b n - 1} a^i f(n/b^i)$$

- Basic assumptions

$$a \geq 1, b > 1$$

- Further assumption

$$f(n) = O(n^d) \text{ with } d \geq 0$$

- So

$$T(n) = \Theta(n^{\log_b a}) + \sum_{i=0}^{\log_b n - 1} a^i O(n^d / b^{id})$$

$$T(n) = \Theta(n^{\log_b a}) + \sum_{i=0}^{\log_b n - 1} \left(\frac{a}{b^d}\right)^i O(n^d)$$

Divide-and-Conqer Complexity (2)

$$T(n) = \Theta(n^{\log_b a}) + \sum_{i=0}^{\log_b n - 1} \left(\frac{a}{b^d}\right)^i O(n^d)$$

Divide-and-Conqer Complexity (2)

$$T(n) = \Theta(n^{\log_b a}) + \sum_{i=0}^{\log_b n - 1} \left(\frac{a}{b^d}\right)^i O(n^d)$$

$$T(n) = \Theta(n^{\log_b a}) + O(n^d) \sum_{i=0}^{\log_b n - 1} \left(\frac{a}{b^d}\right)^i$$

Divide-and-Conqer Complexity (2)

$$T(n) = \Theta(n^{\log_b a}) + \sum_{i=0}^{\log_b n - 1} \left(\frac{a}{b^d}\right)^i O(n^d)$$

$$T(n) = \Theta(n^{\log_b a}) + O(n^d) \sum_{i=0}^{\log_b n - 1} \left(\frac{a}{b^d}\right)^i$$

- Let's look at the geometric-series component; we should figure out the asymptotic behavior of this term

$$O(n^d) \sum_{i=0}^{\log_b n - 1} \left(\frac{a}{b^d}\right)^i$$

Divide-and-Conqer Complexity (3)

$$O(n^d) \sum_{i=0}^{\log_b n - 1} \left(\frac{a}{b^d} \right)^i$$

Divide-and-Conqer Complexity (3)

$$O(n^d) \sum_{i=0}^{\log_b n-1} \left(\frac{a}{b^d}\right)^i$$

- Generalizing

$$g(\ell) = \sum_{i=0}^{\ell} c^i = c^\ell + c^{\ell-1} + \cdots + c + 1$$

Divide-and-Conqer Complexity (3)

$$O(n^d) \sum_{i=0}^{\log_b n-1} \left(\frac{a}{b^d}\right)^i$$

- Generalizing

$$g(\ell) = \sum_{i=0}^{\ell} c^i = c^\ell + c^{\ell-1} + \cdots + c + 1$$

- What is the asymptotic behavior of $g(\ell)$?

Divide-and-Conqer Complexity (3)

$$O(n^d) \sum_{i=0}^{\log_b n - 1} \left(\frac{a}{b^d}\right)^i$$

- Generalizing

$$g(\ell) = \sum_{i=0}^{\ell} c^i = c^\ell + c^{\ell-1} + \cdots + c + 1$$

- What is the asymptotic behavior of $g(\ell)$?

$$g(\ell) = \begin{cases} \Theta(1) & \text{if } c < 1 \\ \Theta(\ell) & \text{if } c = 1 \\ \Theta(c^\ell) & \text{if } c > 1 \end{cases}$$

Divide-and-Conqer Complexity (4)

- Back to divide and conquer

$$O(n^d) \sum_{i=0}^{\log_b n - 1} \left(\frac{a}{b^d}\right)^i =$$

Divide-and-Conqer Complexity (4)

- Back to divide and conquer

$$O(n^d) \sum_{i=0}^{\log_b n - 1} \left(\frac{a}{b^d}\right)^i = \begin{cases} O(n^d)\Theta(1) & \text{if } \frac{a}{b^d} < 1 \\ O(n^d)\Theta(\log_b n) & \text{if } \frac{a}{b^d} = 1 \\ O(n^d)\Theta\left(\left(\frac{a}{b^d}\right)^{\log_b n}\right) & \text{if } \frac{a}{b^d} > 1 \end{cases}$$

Divide-and-Conqer Complexity (4)

- Back to divide and conquer

$$O(n^d) \sum_{i=0}^{\log_b n - 1} \left(\frac{a}{b^d}\right)^i = \begin{cases} O(n^d)\Theta(1) & \text{if } \frac{a}{b^d} < 1 \\ O(n^d)\Theta(\log_b n) & \text{if } \frac{a}{b^d} = 1 \\ O(n^d)\Theta\left(\left(\frac{a}{b^d}\right)^{\log_b n}\right) & \text{if } \frac{a}{b^d} > 1 \end{cases}$$

- In other words, we proved

$$T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a \end{cases}$$