Recurrences and the Complexity of Divide and Conquer Algorithms

Antonio Carzaniga

Faculty of Informatics
University of Lugano

October 8, 2008
Analysis of recurrence expressions
MergeSort
Complexity Analysis

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

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

\[
T(N) = 2T\left(\frac{N}{2}\right) + \Theta(N)
\]
Complexity Analysis

- 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 $\Theta(N \log N)$
Complexity Analysis

- **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
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 $N > 1$
**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 $P$
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 $N/b$
**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

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

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

- Our goal is to obtain a closed-form formula for $T(N)$
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 $d \geq 0$
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

\[ T(N) = aT(N/b) + f(N) \]
Divide-and-Conquer Tree

\[ T(N) = aT(N/b) + f(N) \]

size

\( N \)

\( N/b \)
Divide-and-Conquer Tree

\[ T(N) = aT\left(\frac{N}{b}\right) + f(N) \]
Divide-and-Conquer Tree

\[ T(N) = aT\left(\frac{N}{b}\right) + f(N) \]
Divide-and-Conquer Tree

\[ T(N) = aT\left(\frac{N}{b}\right) + f(N) \]
Divide-and-Conquer Tree

\[ T(N) = aT\left(\frac{N}{b}\right) + f(N) \]
Divide-and-Conquer Tree

\[ T(N) = aT(N/b) + f(N) \]
Divide-and-Conquer Tree

\[ T(N) = aT\left(\frac{N}{b}\right) + f(N) \]

- \( T(N) \): Total time to complete the task.
- \( a \): Constant factor.
- \( f(N) \): Additional work done at each level.
- \( N \): Size of the input.
- \( \frac{N}{b} \): Subproblem size after division.
- \( \log_b N \): Number of levels in the tree.

The diagram illustrates the tree structure, with each level representing a division of the problem size and the recursive calls at each level.
Divide-and-Conquer Tree

\[ T(N) = aT\left(\frac{N}{b}\right) + f(N) \]

- \( N \)
- \( \log_b N \)
- \( \frac{N}{b^2} \)
- \( \frac{N}{b} \)
- \( \frac{N}{b^2} \)
- Size

© 2006 Antonio Carzaniga
$T(N) = a T(N/b) + f(N)$

Divide-and-Conquer Tree

$size$

$N$

$N/b$

$log_b N$

$N/b^2$

1
Divide-and-Conquer Tree

\[ T(N) = aT\left(\frac{N}{b}\right) + f(N) \]

**Diagram:**
- \(N\) is the root node.
- Size decreases by \(\frac{N}{b}\) at each level.
- The height of the tree is \(\log_b N\).
- At each level, there are \(a\) subproblems.
- The function \(f\) is applied at each level.

© 2006 Antonio Carzaniga
Divide-and-Conquer Tree

\[ T(N) = aT(N/b) + f(N) \]
Divide-and-Conquer Tree

\[ T(N) = aT(N/b) + f(N) \]

\[ \text{size} \]

\[ N \]

\[ N/b \]

\[ \log_b N \]

\[ N/b^2 \]

\[ 1 \]

\[ \Theta(1) \quad \Theta(1) \quad \Theta(1) \quad \Theta(1) \quad \cdots \quad \Theta(1) \quad \Theta(1) \quad \Theta(1) \quad \Theta(1) \quad a^{\log_b N} \Theta(1) \]
Divide-and-Conquer Tree

\[ T(N) = aT(N/b) + f(N) \]

\[ T(N) = \Theta(N^{\log_b a}) + \sum_{i=0}^{\log_b N-1} a^i f(N/b^i) \]
Divide-and-Conquer 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-Conquer 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-Conquer Complexity

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

- Basic assumptions
  \[ a \geq 1, \quad b > 1 \]

- Further assumption
  \[ f(N) = O(N^d) \text{ with } d \geq 0 \]

- So
  \[ T(N) = \Theta(N^{\log_b a}) + \sum_{i=0}^{\log_b N-1} a^i O(N^d / b^{id}) \]

© 2006 Antonio Carzaniga
Divide-and-Conquer 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
  \[ T(N) = \Theta(N^{\log_b a}) + \sum_{i=0}^{\log_b N-1} a^i O(N^d / b^{id}) \]
  \[ T(N) = \Theta(N^{\log_b a}) + \sum_{i=0}^{\log_b N-1} \left( \frac{a}{bd} \right)^i O(N^d) \]
Divide-and-Conquer 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-Conquer 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-Conquer 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-Conquer Complexity (3)

\[ O(N^d) \sum_{i=0}^{\log_b N-1} \left( \frac{a}{b^d} \right)^i \]
Divide-and-Conquer Complexity (3)

\[ O(N^d) \log_b N - 1 \sum_{i=0}^{\log_b N - 1} \left( \frac{a}{bd} \right)^i \]

Generalizing

\[ g(n) = \sum_{i=0}^{n} c^i = c^n + c^{n-1} + \cdots + c + 1 \]
Divide-and-Conquer Complexity (3)

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

- Generalizing

\[ g(n) = \sum_{i=0}^{n} c^i = c^n + c^{n-1} + \cdots + c + 1 \]

- What is the asymptotic behavior of \( g(n) \)?
Divide-and-Conquer Complexity (3)

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

Generalizing

\[ g(n) = \sum_{i=0}^{n} c^i = c^n + c^{n-1} + \cdots + c + 1 \]

What is the asymptotic behavior of \( g(n) \)?

\[ g(n) = \begin{cases} 
\Theta(1) & \text{if } c < 1 \\
\Theta(n) & \text{if } c = 1 \\
\Theta(c^n) & \text{if } c > 1 
\end{cases} \]
Divide-and-Conquer Complexity (4)

- Back to divide and conquer

\[ O(N^d) \leq \log_b N - 1 \sum_{i=0}^{\log_b N - 1} \left( \frac{a}{b^d} \right)^i = \]
Back to divide and conquer

\[ O(N^d) \left( \sum_{i=0}^{\log_b N-1} \left( \frac{a}{b^d} \right)^i \right) = \begin{cases} 
O(N^d) \Theta(1) & \text{if } \frac{a}{b^d} < 1 \\
O(N^d) \Theta(\log_b N) & \text{if } \frac{a}{b^d} = 1 \\
O(N^d) \Theta\left( \left( \frac{a}{b^d} \right)^{\log_b N} \right) & \text{if } \frac{a}{b^d} > 1 
\end{cases} \]
Divide-and-Conquer Complexity (4)

Back to divide and conquer

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

In other words, we proved

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