CE100 Algorithms and Programming II

Week-2 (Solving Recurrences)

Spring Semester, 2021-2022

Download DOC, SLIDE, PPTX

<iframe width=700, height=500 frameBorder=0 src="../ce100-week-2-recurrence.md_slide.html"></iframe>

Solving Recurrences

Outline

- Solving Recurrences
  - Recursion Tree
  - Master Method
  - Back-Substitution

- Divide-and-Conquer Analysis
  - Merge Sort
  - Binary Search
  - Merge Sort Analysis
  - Complexity

- Recurrence Solution

Solving Recurrences

- Reminder: Runtime $(T(n))$ of *MergeSort* was expressed as a recurrence

$$T(n) = \begin{cases} \Theta(1) & \text{if n=1} \\ 2T(n/2) + \Theta(n) & otherwise \end{cases}$$

- Solving recurrences is like solving differential equations, integrals, etc.
  - Need to learn a few tricks

Recurrences

<u>Recurrence</u>: An equation or inequality that describes a function in terms of its value on smaller inputs.

Example :

$$T(n) = \begin{cases} 1 & \text{if n=1} \\ T(\lceil n/2 \rceil) + 1 & \text{if n>1} \end{cases}$$

Recurrence Example

$$T(n) = \begin{cases} 1 & \text{if n=1} \\ T(\lceil n/2 \rceil) + 1 & \text{if n>1} \end{cases}$$

- Simplification: Assume $n = 2^k$
- Claimed answer : $T(n) = lgn + 1$
- Substitute claimed answer in the recurrence:

$$lgn + 1 = \begin{cases} 1 & \text{if n=1} \\ lg(\lceil n/2 \rceil) + 2 & \text{if n>1} \end{cases}$$

- True when $n = 2^k$

Technicalities: Floor / Ceiling

Technically, should be careful about the floor and ceiling functions (as in the book).

e.g. For merge sort, the recurrence should in fact be:,

$$T(n) = \begin{cases} \Theta(1) & \text{if n=1} \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) & \text{if n>1} \end{cases}$$

But, it's usually ok to:

- ignore floor/ceiling
- solve for the exact power of 2 (or another number)

Technicalities: Boundary Conditions

- Usually assume: $T(n) = \Theta(1)$ for sufficiently small n
  - Changes the exact solution, but usually the asymptotic solution is not affected (e.g. if polynomially bounded)
- For convenience, the boundary conditions generally implicitly stated in a recurrence
  - $T(n) = 2T(n/2) + \Theta(n)$ assuming that
  - $T(n) = \Theta(1)$ for sufficiently small n

Example: When Boundary Conditions Matter

Exponential function: $T(n) = (T(n/2))2$

Assume

$T(1) = c$ (where c is a positive constant)

$T(2) = (T(1))^2 = c^2$

$T(4) = (T(2))^2 = c^4$

$T(n) = \Theta(c^n)$

e.g.

$$\left.\begin{array}{l} T(1) = 2 \quad \Rightarrow \; T(n) = \Theta(2^n) \\ T(1) = 3 \quad \Rightarrow \; T(n) = \Theta(3^n) \end{array}\right\} \text{However } \Theta(2^n) \neq \Theta(3^n)$$

The difference in solution more dramatic when:

$$T(1) = 1 \Rightarrow T(n) = \Theta(1^n) = \Theta(1)$$

Solving Recurrences

We will focus on 3 techniques

- Substitution method
- Recursion tree approach
- Master method

Substitution Method

The most general method:

- Guess
- Prove by induction
- Solve for constants

Substitution Method: Example

Solve $T(n) = 4T(n/2) + n$ (assume $T(1) = \Theta(1)$)

1. Guess $T(n) = O(n^3)$ (need to prove O and Ω separately)
2. Prove by induction that $T(n) \leq cn^3$ for large n (i.e. $n \geq n_0$)
    - Inductive hypothesis: $T(k) \leq ck^3$ for any $k < n$
    - Assuming ind. hyp. holds, prove $T(n) \leq cn^3$

Substitution Method: Example – cont’d

Original recurrence: $T(n) = 4T(n/2) + n$

From inductive hypothesis: $T(n/2) \leq c(n/2)^3$

Substitute this into the original recurrence:

- $T(n) \leq 4c(n/2)^3 + n$
- $= (c/2)n^3 + n$
- $= cn^3 – ((c/2)n^3 – n) \Longleftarrow$ desired - residual
- $\leq cn^3$
  when $((c/2)n^3 – n) \geq 0$

Substitution Method: Example – cont'd

So far, we have shown:

$$T(n) \leq cn^3 \text{ when } ((c/2)n^3 - n) \geq 0$$

We can choose $c \geq 2$ and $n_0 \geq 1$

But, the proof is not complete yet.

Reminder: Proof by induction:

1.Prove the base cases $\Longleftarrow$ haven't proved the base cases yet

2.Inductive hypothesis for smaller sizes

3.Prove the general case

Substitution Method: Example – cont’d

- We need to prove the base cases
  - Base: $T(n) = \Theta(1)$ for small n (e.g. for $n = n_0$)
- We should show that:
  - $\Theta(1) \leq cn^3$ for $n = n_0$, This holds if we pick c big enough
- So, the proof of $T(n) = O(n^3)$ is complete
- But, is this a tight bound?

Example: A tighter upper bound?

- Original recurrence: $T(n) = 4T(n/2) + n$
- Try to prove that $T(n) = O(n^2)$,
  - i.e. $T(n) \leq cn^2$ for all $n \geq n_0$
- **Ind. hyp:** Assume that $T(k) \leq ck^2$ for $k < n$
- **Prove the general case:** $T(n) \leq cn^2$

Example (cont'd)

Original recurrence: $T(n) = 4T(n/2) + n$

Ind. hyp: Assume that $T(k) \leq ck^2$ for $k < n$

Prove the general case: $T(n) \leq cn^2$

$$\begin{aligned} T(n) &= 4T(n/2) + n \\ &\leq 4c(n/2)^2 + n \\ &= cn^2 + n \\ &= O(n2) \Longleftarrow \text{ Wrong! We must prove exactly} \end{aligned}$$

Example (cont'd)

Original recurrence: $T(n) = 4T(n/2) + n$

Ind. hyp: Assume that $T(k) \leq ck^2$ for $k < n$

Prove the general case: $T(n) \leq cn^2$

- So far, we have:
  $T(n) \leq cn^2 + n$
- No matter which positive c value we choose, this does not show that $T(n) \leq cn^2$
- Proof failed?

Example (cont'd)

- What was the problem?
  - The inductive hypothesis was not strong enough
- **Idea:** Start with a stronger inductive hypothesis
  - Subtract a low-order term
- **Inductive hypothesis:** $T(k) \leq c_1 k^2 - c_2 k$ for $k < n$
- **Prove the general case:** $T(n) \leq c_1 n^2 - c_2 n$

Example (cont'd)

Original recurrence: $T(n) = 4T(n/2) + n$

Ind. hyp: Assume that $T(k) \leq c_1 k^2 – c_2 k$ for $k < n$

Prove the general case: $T(n) \leq c_1 n^2 – c_2 n$

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

$\leq 4(c_1(n/2)^2 – c_2(n/2)) + n$

$= c_1 n^2 – 2c_2 n + n$

$= c_1 n^2 – c_2 n – (c_2 n – n)$

$\leq c_1 n^2 – c_2 n$ for $n(c_2 – 1) \geq 0$

choose $c2 \geq 1$

Example (cont'd)

We now need to prove

$$T(n) \leq c_1 n^2 - c_2 n$$

for the base cases.

$T(n) = \Theta(1) \text{ for } 1 \leq n \leq n_0$ (implicit assumption)

$\Theta(1) \leq c_1 n^2 - c_2 n$ for n small enough (e.g. $n = n_0$)

- We can choose c1 large enough to make this hold

We have proved that $T(n) = O(n^2)$

Substitution Method: Example 2

For the recurrence $T(n) = 4T(n/2) + n$,

prove that $T(n) = \Omega(n^2)$

i.e. $T(n) \geq cn^2$ for any $n \geq n_0$

Ind. hyp: $T(k) \geq ck^2$ for any $k < n$

Prove general case: $T(n) \geq cn^2$

$$T(n) = 4T(n/2) + n$$
$$\geq 4c(n/2)^2 + n$$
$$= cn^2 + n$$
$$\geq cn^2 \text{ since } n > 0$$

Proof succeeded – no need to strengthen the ind. hyp as in the last example

Example 2 (cont'd)

We now need to prove that

$T(n) \geq cn^2$

for the base cases

$T(n) = \Theta(1)$ for $1 \leq n \leq n_0$ (implicit assumption)

$\Theta(1) \geq cn^2$ for $n = n_0$

n_0 is sufficiently small (i.e. constant)

We can choose c small enough for this to hold

We have proved that $T(n) = \Omega(n^2)$

Substitution Method - Summary

- Guess the asymptotic complexity
- Prove your guess using induction
  - Assume inductive hypothesis holds for $k < n$
  - Try to prove the general case for n
    - Note: $MUST$ prove the $EXACT$ inequality $CANNOT$ ignore lower order terms, If the proof fails, strengthen the ind. hyp. and try again
- Prove the base cases (usually straightforward)

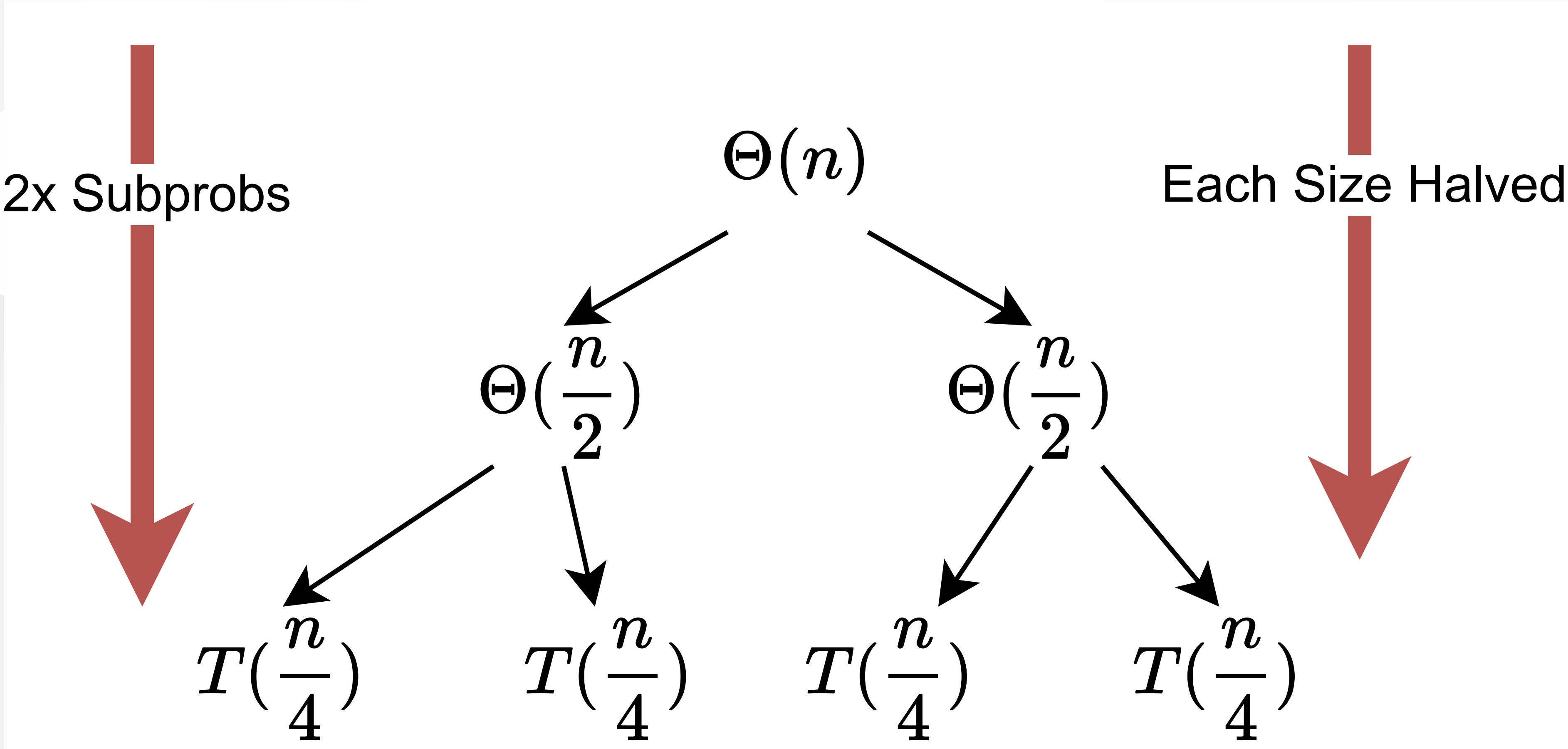
Recursion Tree Method

- A recursion tree models the runtime costs of a recursive execution of an algorithm.
- The recursion tree method is good for generating guesses for the substitution method.
- The recursion-tree method can be unreliable.
    - Not suitable for formal proofs
- The recursion-tree method promotes intuition, however.

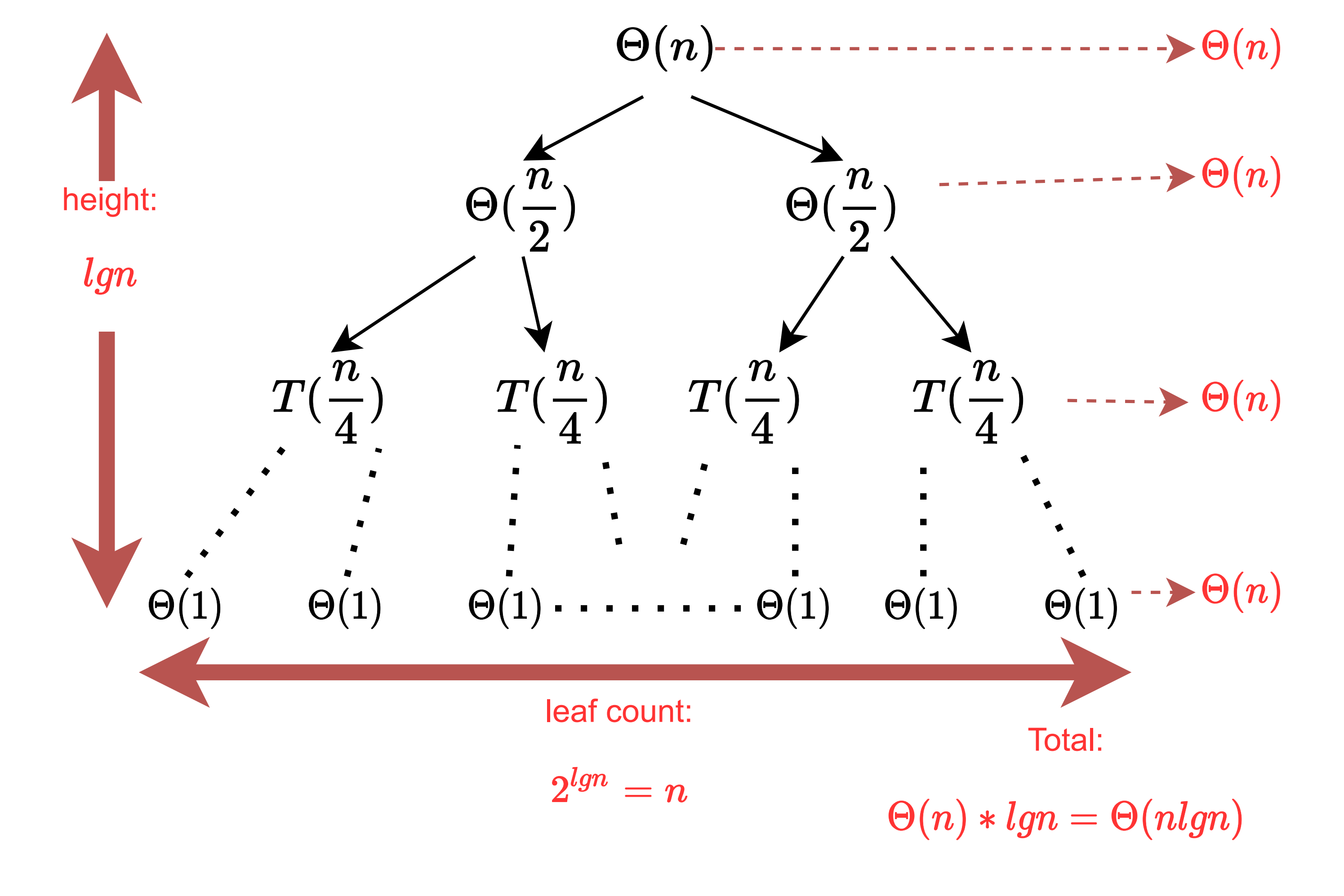
Solve Recurrence : $T(n) = 2T(n/2) + \Theta(n)$

$$
\begin{tikzcd}
 & \Theta(n) \arrow[dl] \arrow[dr] & \\
T(\frac{n}{2}) & & T(\frac{n}{2})
\end{tikzcd}
$$

Solve Recurrence : $T(n) = 2T(n/2) + \Theta(n)$



Solve Recurrence : $T(n) = 2T(n/2) + \Theta(n)$



Example of Recursion Tree

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

TODO

Continue From

Slide : 03

Page : 27

References

TODO