Lecture 3
Solving Recurrences
Solving Recurrences

- Reminder: Runtime \((T(n))\) of \textit{MergeSort} was expressed as a recurrence

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

- Solving recurrences is like solving differential equations, integrals, etc.
  
  \textit{Need to learn a few tricks}
Recurrences

- **Recurrence**: 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: \( T(n) = \log n + 1 \)
- Substitute claimed answer in the recurrence:

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

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 exact powers 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. $T(1) = 2 \Rightarrow T(n) = \Theta(2^n)$
  
  $T(1) = 3 \Rightarrow T(n) = \Theta(3^n)$

- 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 in this lecture:
  1. Substitution method
  2. Recursion tree approach
  3. Master method
Substitution Method

- The most general method:
  1. Guess
  2. Prove by induction
  3. 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 $\Omega$ 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:

\[
T(n) \leq 4c (n/2)^3 + n \\
= (c/2) n^3 + n \\
= cn^3 - ((c/2)n^3 - n) \\
\leq cn^3
\]

when \( ((c/2)n^3 - n) \geq 0 \)
Substitution Method: Example – cont’d

- So far, we have shown:
  \[ T(n) \leq cn^3 \quad \text{when } ((c/2)n^3 - n) \geq 0 \]

- 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
  2. Inductive hypothesis for smaller sizes
  3. Prove the general case

  haven’t proved the base cases yet
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:

  \[ \Theta(1) \leq cn^3 \text{ for } n = n_0 \]

  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:** \( T(n) \leq cn^2 \)

\[
T(n) = 4T(n/2) + n \\
\leq 4c(n/2)^2 + n \\
= cn^2 + n \\
= \Theta(n^2) \quad \text{Wrong! We must prove exactly}
\]
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: \( T(n) = 4T(n/2) + n \)
- \textbf{Ind. hyp:} Assume that \( T(k) \leq c_1 k^2 - c_2 k \) for \( k < n \)
- \textbf{Prove the general case:} \( T(n) \leq c_1 n^2 - c_2 n \)

\[
T(n) = 4T(n/2) + n \\
\leq 4 \left( c_1 \left(\frac{n}{2}\right)^2 - c_2 \left(\frac{n}{2}\right) \right) + 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 \\
\text{for } n(c_2 - 1) \geq 0 \\
\text{choose } c_2 \geq 1
Example (cont’d)

- We now need to prove
  \[ T(n) \leq c_1 n^2 - c_2 n \]
  for the base cases.

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

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

We can choose \( c_1 \) large enough to make this hold

- We have proved that \( T(n) = O(n^2) \)
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:** \( T(n) \geq cn^2 \)

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

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

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

  \[ “\Theta(1)” \geq cn^2 \text{ for } n = n_0 \]

  \[ n_0 \text{ is sufficiently small (i.e. constant)} \]

  We can choose \( c \) small enough for this to hold

- We have proved that \( T(n) = \Omega(n^2) \)
Substitution Method - Summary

1. **Guess the asymptotic complexity**

1. **Prove your guess using induction**
   1. Assume inductive hypothesis holds for $k < n$
   2. 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

3. 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: \( T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n) \)
Solve Recurrence: \( T(n) = 2T \left( \frac{n}{2} \right) + \Theta(n) \)
Solve Recurrence: \( T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n) \)

\[ 2^{\log n} = n \]

Total: \( \Theta(n \log n) \)
Example of Recursion Tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$: 
Example of Recursion Tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:
Example of Recursion Tree

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

```
      n^2
     / \  \\
T(n/4) T(n/2)
```
Example of Recursion Tree

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

```
  n^2
   / \
(n/4)^2  (n/2)^2
 /     /     \
T(n/16) T(n/8) T(n/8) T(n/4)
```

Example of Recursion Tree

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

$$\Theta(1)$$

$$n^2$$

$$\frac{n}{4}^2$$  $$\frac{n}{2}^2$$

$$\frac{n}{16}^2$$  $$\frac{n}{8}^2$$  $$\frac{n}{8}^2$$  $$\frac{n}{4}^2$$
Example of Recursion Tree

Solve \( T(n) = T(n/4) + T(n/2) + n^2 \):

\[ n^2 \]

\[ (n/4)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]

\[ (n/16)^2 \]
Example of Recursion Tree

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

$\Theta(1)$

$\frac{5}{16} n^2$

$n^2$

$(n/4)^2$

$(n/2)^2$

$(n/8)^2$

$(n/8)^2$

$(n/16)^2$

$(n/16)^2$
Example of Recursion Tree

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

- $n^2$
- $(n/4)^2$
- $(n/8)^2$
- $(n/16)^2$
- $\Theta(1)$

- $n^2$
- $(n/2)^2$
- $(n/8)^2$
- $(n/4)^2$
- $5/16 \cdot n^2$

- $n^2$
- $(n/8)^2$
- $(n/4)^2$
- $25/256 \cdot n^2$
Example of Recursion Tree

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

$$n^2 \quad \frac{n^2}{4} \quad \frac{n^2}{2}$$

$$\frac{n^2}{16} \quad \frac{n^2}{8} \quad \frac{n^2}{8} \quad \frac{n^2}{4}$$

$\Theta(1)$

Total $= n^2 \left(1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \left(\frac{5}{16}\right)^2 + \ldots\right)$

$= \Theta(n^2)$, geometric series
The Master Method

- A powerful black-box method to solve recurrences.

- The master method applies to recurrences of the form

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

where \( a \geq 1 \), \( b > 1 \), and \( f \) is asymptotically positive.
The Master Method: 3 Cases

- Recurrence: $T(n) = aT(n/b) + f(n)$

- Compare $f(n)$ with $n \log_b a$

- Intuitively:

  **Case 1:** $f(n)$ grows *polynomially slower* than $n \log_b a$

  **Case 2:** $f(n)$ grows *at the same rate* as $n \log_b a$

  **Case 3:** $f(n)$ grows *polynomially faster* than $n \log_b a$
The Master Method: Case 1

- Recurrence: \( T(n) = aT(n/b) + f(n) \)

Case 1: \( \frac{n^{\log_b a}}{f(n)} = \Omega(n^\varepsilon) \) for some constant \( \varepsilon > 0 \)

**i.e.,** \( f(n) \) grows polynomially slower than \( n^{\log_b a} \) (by an \( n^\varepsilon \) factor).

**Solution:** \( T(n) = \Theta(n^{\log_b a}) \)
The Master Method: Case 2 (simple version)

- **Recurrence:** $T(n) = aT(n/b) + f(n)$

**Case 2:**

\[ \frac{f(n)}{n^{\log_b a}} = \Theta(1) \]

- i.e., $f(n)$ and $n^{\log_b a}$ grow at similar rates.

**Solution:** $T(n) = \Theta(n^{\log_b a} \log n)$
The Master Method: Case 3

Case 3: \[ \frac{f(n)}{n^\log_b a} = \Omega(n^\varepsilon) \]
for some constant \( \varepsilon > 0 \)

i.e., \( f(n) \) grows polynomially faster than \( n^\log_b a \) (by an \( n^\varepsilon \) factor).

and the following regularity condition holds:
\[ a f(n/b) \leq c f(n) \]
for some constant \( c < 1 \)

Solution: \( T(n) = \Theta(f(n)) \)
Example: \( T(n) = 4T(n/2) + n \)

- \( a = 4 \)
- \( b = 2 \)
- \( f(n) = n \)
- \( n^{\log_b a} = n^2 \)

\( f(n) \) grows \textit{polynomially} slower than \( n^{\log_b a} \)

\[
\frac{n^{\log_b a}}{f(n)} = \frac{n^2}{n} = n = \Omega(n^\varepsilon)
\]

for \( \varepsilon = 1 \)

\( T(n) = \Theta(n^{\log_b a}) \)

\( T(n) = \Theta(n^2) \)
Example: $T(n) = 4T(n/2) + n^2$

\[
a = 4 \\
b = 2 \\
f(n) = n^2
\]

$f(n)$ grows at similar rate as $n^{\log_b a}$

\[
n^{\log_b a} = n^2
\]

CASE 2

\[
T(n) = \Theta(n^{\log_b a} \log n) = \Theta(n^2 \log n)
\]
Example: $T(n) = 4T(n/2) + n^3$

$a = 4$

$b = 2$

$f(n) = n^3$

$n^{\log_b a} = n^2$

$f(n)$ grows \textit{polynomially} faster than $n^{\log_b a}$

\[
\frac{f(n)}{n^{\log_b a}} = \frac{n^3}{n^2} = n = \Omega(n^\varepsilon)
\]

for $\varepsilon = 1$

seems like CASE 3, but need to check the regularity condition

Regularity condition: $a f(n/b) \leq c f(n)$ for some constant $c < 1$

$4 \left(\frac{n}{2}\right)^3 \leq cn^3$ for $c = 1/2$

CASE 3

$T(n) = \Theta(f(n))$

$T(n) = \Theta(n^3)$
Example: $T(n) = 4T(n/2) + n^2/\lg n$

\[
a = 4 \\
b = 2 \\
f(n) = n^2/\lg n
\]

\[
\frac{n^{\log_b a}}{f(n)} = \frac{n^2}{n^2} = \frac{1}{\lg n} \neq \Omega(n^\varepsilon)
\]

for any $\varepsilon > 0$

\[
\text{is not CASE 1}
\]

\[
\text{Master method does not apply!}
\]
The Master Method: Case 2 (general version)

- **Recurrence:** $T(n) = aT(n/b) + f(n)$

**Case 2:** \[
\frac{f(n)}{n^{\log_b a}} = \Theta(\lg^k n) \quad \text{for some constant } k \geq 0
\]

**Solution:** $T(n) = \Theta \left( n^{\log_b a} \lg^{k+1} n \right)$
General Method (Akra-Bazzi)

\[ T(n) = \sum_{i=1}^{k} a_i T(n/b_i) + f(n) \]

Let \( p \) be the unique solution to

\[ \sum_{i=1}^{k} \left( \frac{a_i}{b^p_i} \right) = 1 \]

Then, the answers are the same as for the master method, but with \( n^p \) instead of \( n^{\log_b a} \)

(Akra and Bazzi also prove an even more general result.)
Idea of Master Theorem

Recursion tree:

\[ T(n) = \begin{cases} 
T(1) & \text{if } a = 1 \\
af(n) & \text{if } a > 1 \text{ and } f(n) = \Omega(n^\log_b a) \\
af(n) + \sum_{i=0}^{h-1} af(n/b^i) & \text{if } a > 1 \text{ and } f(n) = O(n^c) \text{ such that } c < \log_b a \end{cases} \]

\[ h = \log_b n \]

\#leaves = \[ a^h = a^{\log_b n} = n^{\log_b a} \]

\[ T(n) = n^{\log_b a} \]
Idea of Master Theorem

Recursion tree:

CASE 1: The weight increases geometrically from the root to the leaves. The leaves hold a constant fraction of the total weight.

\[ n^{\log_b a} T(1) \]

\[ \Theta \left( n^{\log_b a} \right) \]
Idea of Master Theorem

Recursion tree:

CASE 2: \( (k = 0) \) The weight is approximately the same on each of the \( \log_b n \) levels.

\[
T(n) = \begin{cases} 
T(1) & \text{if } n = 1 \\
T(n/b) + f(n) & \text{otherwise}
\end{cases}
\]

\[
h = \log_b n
\]

\[
a \cdot T(n/b) + f(n) = a \cdot T(n/b) + f(n/b)
\]

\[
a^2 \cdot T(n/b^2) + f(n/b^2) = a^2 \cdot T(n/b^2) + f(n/b^2)
\]

\[
\Theta(n^{\log_b a} \log n)
\]
Recursion tree:

\[ f(n) \]
\[ f(n/b) \]
\[ f(n/b^2) \]

CASE 3: The weight decreases geometrically from the root to the leaves. The root holds a constant fraction of the total weight.

\[ T(1) = n^{\log_b a} \Theta(f(n)) \]
Proof of Master Theorem: Case 1 and Case 2

• Recall from the recursion tree (note $h = \lfloor \log_b n \rfloor =$ tree height)

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

Leaf cost Non-leaf cost = $g(n)$
Proof of Case 1

\[ \frac{n^{\log_b a}}{f(n)} = \Omega(n^\varepsilon) \quad \text{for some } \varepsilon > 0 \]

\[ \frac{n^{\log_b a}}{f(n)} = \Omega(n^\varepsilon) \implies \frac{f(n)}{n^{\log_b a}} = O(n^{-\varepsilon}) \implies f(n) = O(n^{\log_b a - \varepsilon}) \]

\[ g(n) = \sum_{i=0}^{h-1} a^i \bigO\left((n/b^i)^{\log_b a - \varepsilon}\right) = O\left(\sum_{i=0}^{h-1} a^i (n/b^i)^{\log_b a - \varepsilon}\right) \]

\[ = O\left( n^{\log_b a - \varepsilon} \sum_{i=0}^{h-1} a^i b^{-i\varepsilon} / b^{i\log_b a} \right) \]
Case 1 (cont’)

\[ \sum_{i=0}^{h-1} \frac{a^i b^{i \varepsilon}}{b^{i \log_b a}} = \sum_{i=0}^{h-1} a^i \frac{(b^\varepsilon)^i}{(b^{\log_b a})^i} = \sum a^i \frac{b^{\varepsilon i}}{a^i} = \sum_{i=0}^{h-1} (b^\varepsilon)^i \]

= An increasing geometric series since \( b > 1 \)

\[ \frac{b^{\varepsilon h} - 1}{b^\varepsilon - 1} = \frac{(b^h)^\varepsilon - 1}{b^\varepsilon - 1} = \frac{(b^{\log_b n})^\varepsilon - 1}{b^\varepsilon - 1} = \frac{n^\varepsilon - 1}{b^\varepsilon - 1} = O(n^\varepsilon) \]
Case 1 (cont’)

\[-g(n) = O\left(n^{\log_b a - \varepsilon} O(n^{\varepsilon})\right) = O\left(\frac{n^{\log_b a}}{n^{\varepsilon}} O(n^{\varepsilon})\right)\]

\[= O(n^{\log_b a})\]

\[-T(n) = \Theta(n^{\log_b a}) + g(n) = \Theta(n^{\log_b a}) + O(n^{\log_b a})\]

\[= \Theta(n^{\log_b a})\]

Q.E.D.
Proof of Case 2 (limited to $k=0$)

\[
\frac{f(n)}{n^{\log_b a}} = \Theta(\log^n n) = \Theta(1) \Rightarrow f(n) = \Theta(n^{\log_b a}) \Rightarrow f(n/b^i) = \Theta\left(\left(\frac{n}{b^i}\right)^{\log_b a}\right)
\]

\[
\therefore g(n) = \sum_{i=0}^{h-1} a^i \Theta\left((n/b^i)^{\log_b a}\right)
\]

\[
= \Theta\left(\sum_{i=0}^{h-1} a^i \frac{n^{\log_b a}}{b^{i\log_b a}}\right) = \Theta\left(n^{\log_b a} \sum_{i=0}^{h-1} a^i \frac{1}{(b^{\log_b a})^i}\right) = \Theta\left(n^{\log_b a} \sum_{i=0}^{h-1} a^i \frac{1}{a^i}\right)
\]

\[
= \Theta\left(n^{\log_b a} \sum_{i=0}^{\log_n n-1} 1\right) = \Theta\left(n^{\log_b a} \log_b n\right) = \Theta\left(n^{\log_b a} \log n\right)
\]

\[
T(n) = n^{\log_b a} + \Theta(n^{\log_b a} \log n)
\]

\[
= \Theta\left(n^{\log_b a} \log n\right)
\]

Q.E.D.
Conclusion

• Next time: applying the master method.