CS 253: Algorithms

Chapter 4

Divide-and-Conquer

Recurrences

Master Theorem

Credit: Dr. George Bebis

Recurrences and Running Time

- Recurrences arise when an algorithm contains recursive calls to itself
- Running time is represented by an equation or inequality that describes a function in terms of its value on smaller inputs.

$$T(n) = T(n-1) + n$$

- What is the actual running time of the algorithm? i.e. $T(n) = ?$
- Need to solve the recurrence
  - Find an explicit formula of the expression
  - Bound the recurrence by an expression that involves n

Example Recurrences

- $T(n) = T(n-1) + n$ $\Theta(n^2)$

  Recursive algorithm that loops through the input to eliminate one item

- $T(n) = T(n/2) + c$ $\Theta(\lg n)$

  Recursive algorithm that halves the input in one step

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

  Recursive algorithm that halves the input but must examine every item in the input

- $T(n) = 2T(n/2) + 1$ $\Theta(n)$

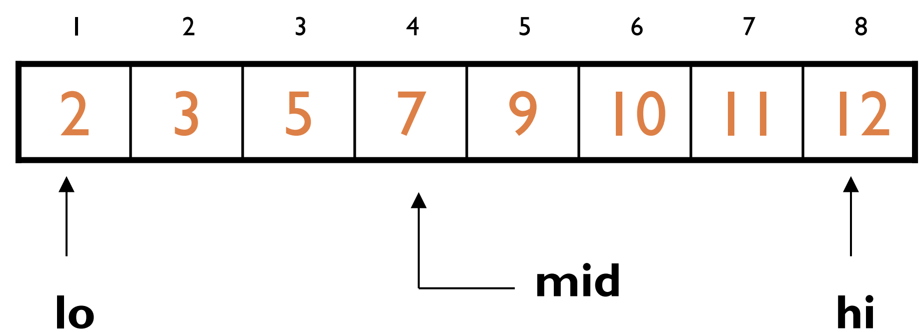
  Recursive algorithm that splits the input into 2 halves and does a constant amount of other work

BINARY-SEARCH

- Finds if x is in the **sorted** array A[lo…hi]

Alg.: BINARY-SEARCH (A, lo, hi, x)

if (lo > hi)
    return FALSE
mid ← $\lfloor$(lo+hi)/2$\rfloor$
if x = A[mid]
    return TRUE
if (x < A[mid])
    BINARY-SEARCH (A, lo, mid-1, x)
if (x > A[mid])
    BINARY-SEARCH (A, mid+1, hi, x)



Example 1

A[8] = {1, 2, 3, 4, 5, 7, 9, 11}

lo = 1 hi = 8 x = 7

1	2	3	4	5	6	7	8
1	2	3	4	5	7	9	11

mid = 4, lo = 5, hi = 8

				5	6	7	8
1	2	3	4	5	7	9	11

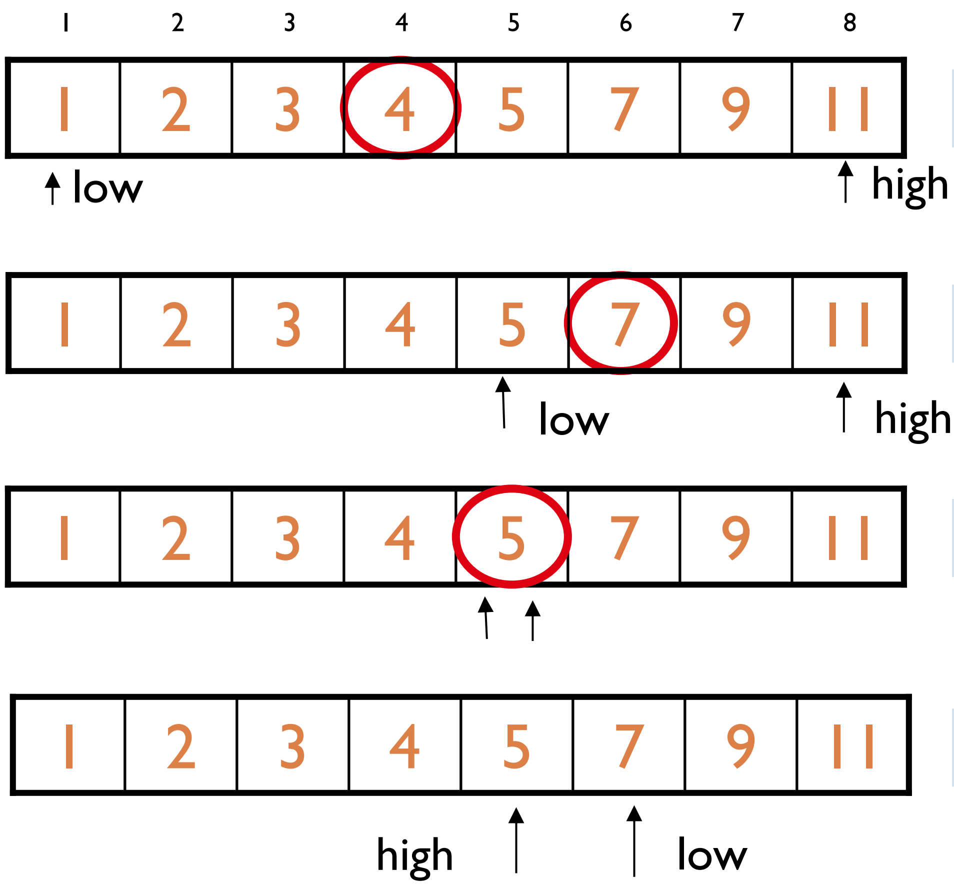
mid = 6, A[mid] = x

Found!

Example II

A[8] = {1, 2, 3, 4, 5, 7, 9, 11}

lo = 1   hi = 8   x = 6



lo = 1, hi = 8, mid = 4. A[4]=4

lo = 5, hi = 8, mid = 6, A[6]=7

lo = 5, hi = 5, mid = 5, A[5]=5

lo = 6, hi = 5 → **NOT FOUND!**

Analysis of BINARY-SEARCH

Alg.: BINARY-SEARCH (A, lo, hi, x)

```
if (lo > hi)
    return FALSE                          ←  constant time: c1
mid ← ⌊(lo+hi)/2⌋                         ←  constant time: c2
if x = A[mid]
    return TRUE                           ←  constant time: c3
if ( x < A[mid] )
    BINARY-SEARCH (A, lo, mid-1, x)       ←  same problem of size n/2
if ( x > A[mid] )
    BINARY-SEARCH (A, mid+1, hi, x)       ←  same problem of size n/2
```

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

Methods for Solving Recurrences

- **Iteration** method
- **Recursion-tree** method
- **Master** method

The Iteration Method

Convert the recurrence into a summation and solve it using a known series

Example: $\mathbf{T(n) = c + T(n/2)}$

$$T(n) = c + T(n/2)$$
$$= c + c + T(n/4)$$
$$= c + c + c + T(n/8)$$
$$= c + c + c + c + T(n/2^4)$$

Assume $n=2^k$ then $k = \lg n$ and

$$T(n) = \underbrace{c + c + c + c + c + \dots}_{(k \text{ times})} + T(n/2^k)$$

$$T(n) = k * c + T(1)$$

$$T(n) = c \lg n$$

Iteration Method – Example 2

$T(n) = n + 2T(n/2)$ **Assume** $n=2^k \Rightarrow k = \lg n$

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

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

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

$$= n + n + 4(n/4 + 2T(n/8))$$

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

$$T(n) = 3n + 2^3T(n/2^3)$$

$$= kn + 2^kT(n/2^k)$$

$$= n\lg n + nT(1)$$

$T(n) = O(n\lg n)$

Methods for Solving Recurrences

- **Iteration** method
- **Recursion-tree** method
- **Master** method

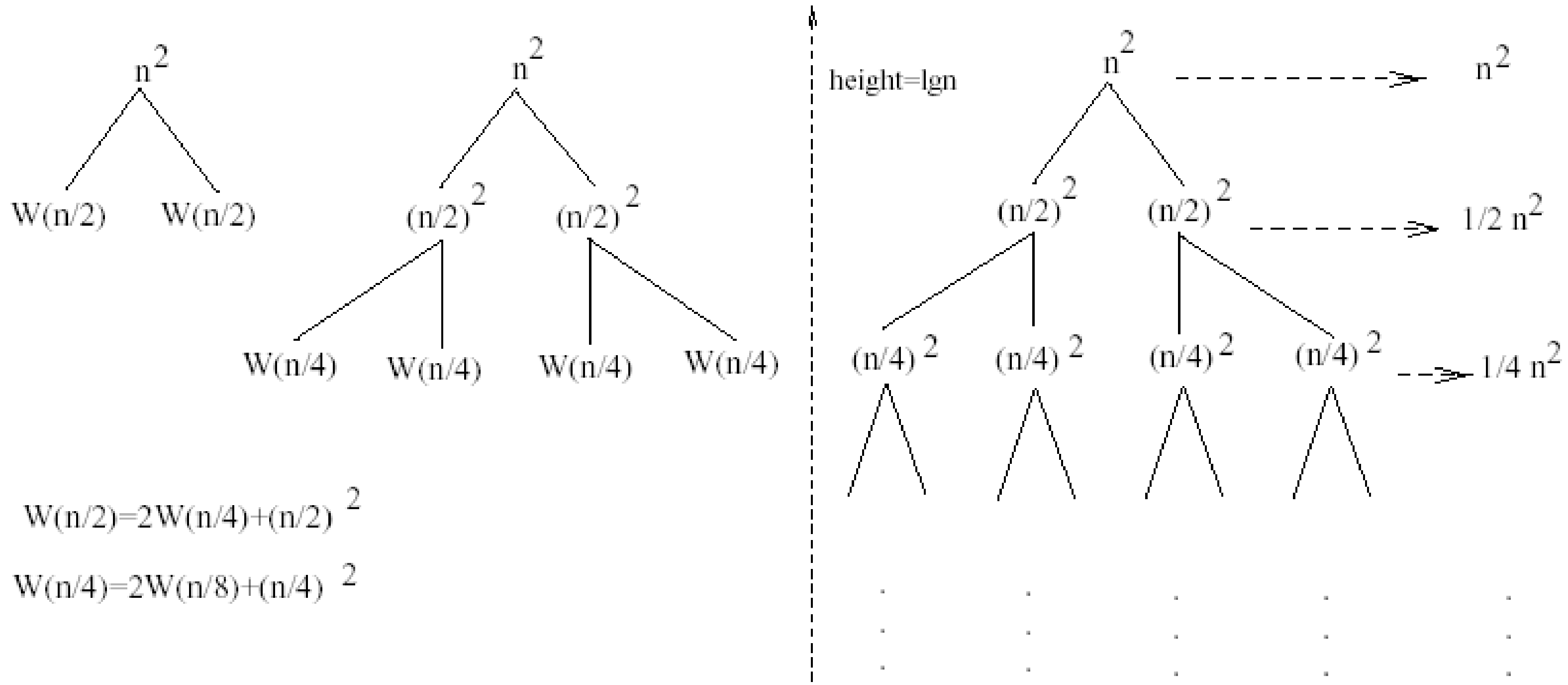
The recursion-tree method

Convert the recurrence into a tree:

- Each node represents the cost incurred at various levels of recursion
- Sum up the costs of all levels

Used to “guess” a solution for the recurrence

Example 1 $W(n) = 2W(n/2) + n^2$

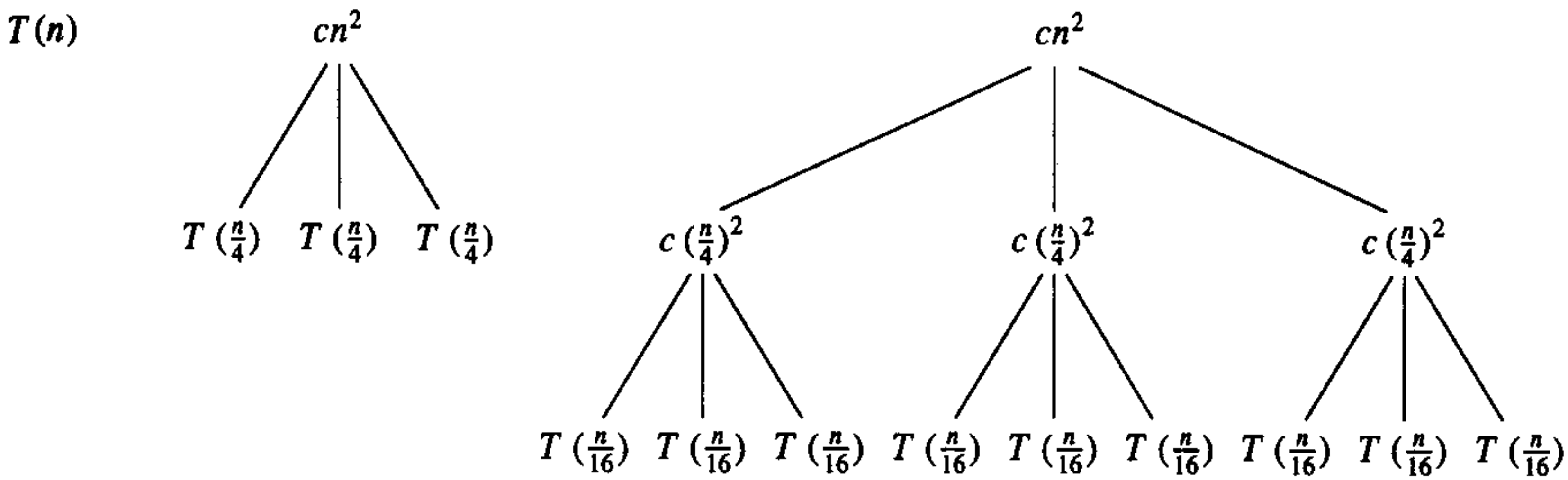


- Subproblem size at level i = $n/2^i$
- **At level i:** Cost of each node = $(n/2^i)^2$ # of nodes = 2^i Total cost = $(n^2/2^i)$
- h = Height of the tree ➔ $n/2^h=1$ ➔ **h = lgn**
- Total cost at all levels:

$$W(n) = \sum_{i=0}^{\lg n} \frac{n^2}{2^i} = n^2 \sum_{i=0}^{\lg n} \left(\frac{1}{2}\right)^i \le n^2 \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i = n^2 \frac{1}{1-\frac{1}{2}} = 2n^2$$

➔ **W(n) = O(n^2)**

Example 2 $T(n) = 3T(n/4) + cn^2$



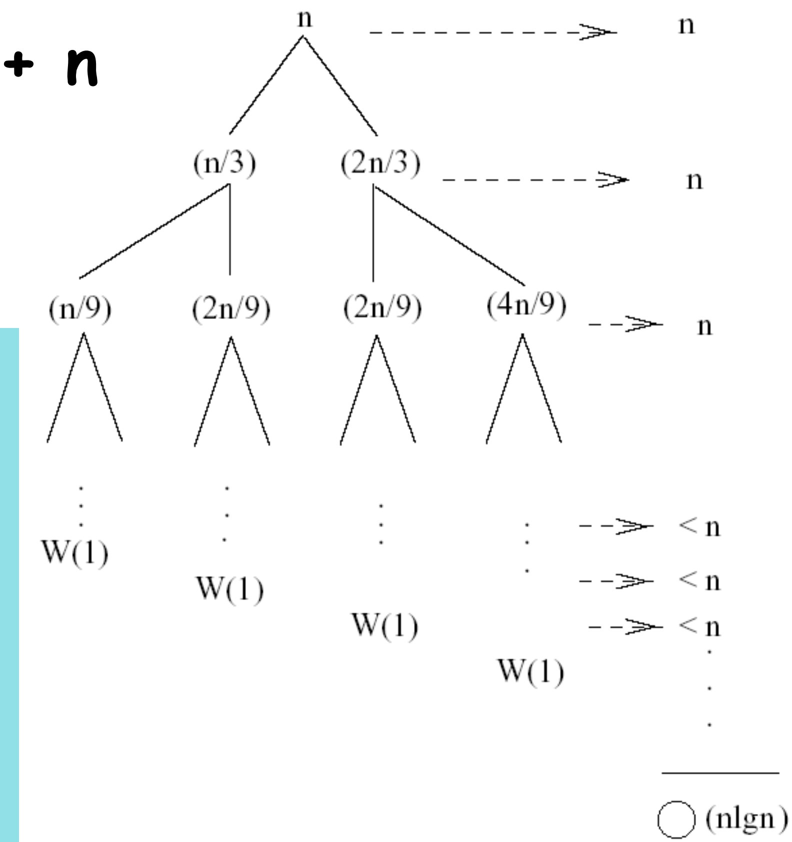
- Subproblem size at level i = $n/4^i$
- **At level i:** Cost of each node= $c(n/4^i)^2$ # of nodes= 3^i Total cost =$cn^2(3/16)^i$
- h = Height of the tree ➔ $n/4^h=1$ ➔ **h = $\log_4 n$**
- Total cost at all levels: (last level has $3^{\log_4 n} = n^{\log_4 3}$ nodes)

$$T(n) = \sum_{i=0}^{\log_4 n-1}\left(\frac{3}{16}\right)^i cn^2 + \Theta\left(n^{\log_4 3}\right) \le \sum_{i=0}^{\infty}\left(\frac{3}{16}\right)^i cn^2 + \Theta\left(n^{\log_4 3}\right) = \frac{1}{1-\frac{3}{16}} cn^2 + \Theta\left(n^{\log_4 3}\right) = O(n^2)$$

➔ **$T(n) = O(n^2)$**

Example 3

W(n) = W(n/3) + W(2n/3) + n



- The longest path from the root to a leaf:

  $n \to (2/3)n \to (2/3)^2\, n \to \ldots \to 1$

  $(2/3)^i n = 1 \quad \Leftrightarrow \quad i = \log_{3/2} n$

- Cost of the problem at level i = n
- Total cost:

$$W(n) = n(\log_{3/2} n) = n\frac{\lg n}{\lg(3/2)} = O(n \lg n)$$

via further analysis ➔**W(n) =Θ(nlgn)**

Methods for Solving Recurrences

- **Iteration** method

- **Recursion-tree** method

- **Master** method

Master Theorem

- "Cookbook" for solving recurrences of the form:

$$T(n) = aT\left(\frac{n}{b}\right) + f(n) \qquad \text{where } a \geq 1,\ b > 1, \text{and } f(n) > 0$$

Idea: compare $f(n)$ with $n^{\log_b a}$

- $f(n)$ is asymptotically **smaller** or **larger** than $n^{\log_b a}$ by a polynomial factor n^{ε}

OR

- $f(n)$ is asymptotically **equal** with $n^{\log_b a}$

Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n) \qquad \text{where } a \geq 1,\ b > 1, \text{and } f(n) > 0$$

Case 1: if $f(n) = O(n^{\log_b a - \varepsilon})$ for some $\varepsilon > 0$, then: $T(n) = \Theta(n^{\log_b a})$

Case 2: if $f(n) = \Theta(n^{\log_b a})$, then: $T(n) = \Theta(n^{\log_b a} \lg n)$

Case 3: if $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some $\varepsilon > 0$, and if

$af(n/b) \leq cf(n)$ for some $c < 1$ and all sufficiently large n, then:

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

regularity condition

Example 1

Case 1: if $f(n) = O(n^{\log_b a - \varepsilon})$ for some $\varepsilon > 0$, then: $T(n) = \Theta(n^{\log_b a})$

Case 2: if $f(n) = \Theta(n^{\log_b a})$, then: $T(n) = \Theta(n^{\log_b a} \lg n)$

Case 3: if $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some $\varepsilon > 0$, and if

$af(n/b) \le cf(n)$ for some $c < 1$ and all sufficiently large n, then: $T(n) = \Theta(f(n))$

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

$a = 2,\ b = 2,\ \log_2 2 = 1$

Compare $n^{\log_b a} = n^1$ with $f(n) = n$

$f(n) = \Theta(n^{\log_b a} = n^1) \Rightarrow$ **Case 2**

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

Example 2

Case 1: if $f(n) = O(n^{\log_b a - \varepsilon})$ for some $\varepsilon > 0$, then: $T(n) = \Theta(n^{\log_b a})$

Case 2: if $f(n) = \Theta(n^{\log_b a})$, then: $T(n) = \Theta(n^{\log_b a} \lg n)$

Case 3: if $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some $\varepsilon > 0$, and if

$af(n/b) \le cf(n)$ for some $c < 1$ and all sufficiently large n, then: $T(n) = \Theta(f(n))$

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

$a = 2, \quad b = 2, \quad \log_2 2 = 1$

Compare $n^{\log_2 2} = n^1$ with $f(n) = n^2$

$f(n) = \Omega(n^{\log_2 2 + \varepsilon})$ ➔ **Case 3** (* need to verify regularity cond.)

$$a\, f(n/b) \le c\, f(n) \Leftrightarrow 2\, n^2/4 \le c\, n^2 \Rightarrow (\tfrac{1}{2} \le c < 1)$$

$$\rightarrow \quad T(n) = \Theta(f(n)) = \Theta(n^2)$$

Example 3

Case 1: if $f(n) = O(n^{\log_b a - \varepsilon})$ for some $\varepsilon > 0$, then: $T(n) = \Theta(n^{\log_b a})$

Case 2: if $f(n) = \Theta(n^{\log_b a})$, then: $T(n) = \Theta(n^{\log_b a} \lg n)$

Case 3: if $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some $\varepsilon > 0$, and if

$af(n/b) \leq cf(n)$ for some $c < 1$ and all sufficiently large n, then: $T(n) = \Theta(f(n))$

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

$a = 2$, $b = 2$, $\log_2 2 = 1$

Compare n with $f(n) = n^{1/2}$

$f(n) = O(n^{1-\varepsilon})$ ➔ **Case 1**

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

Example 4

Case 1: if $f(n) = O(n^{\log_b a - \varepsilon})$ for some $\varepsilon > 0$, then: $T(n) = \Theta(n^{\log_b a})$

Case 2: if $f(n) = \Theta(n^{\log_b a})$, then: $T(n) = \Theta(n^{\log_b a} \lg n)$

Case 3: if $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some $\varepsilon > 0$, and if

$af(n/b) \le cf(n)$ for some $c < 1$ and all sufficiently large n, then: $T(n) = \Theta(f(n))$

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

$a = 3, \; b = 4, \; \log_4 3 = 0.793$

Compare $n^{0.793}$ with $f(n) = n\lg n$

$f(n) = \Omega(n^{\log_4 3 + \varepsilon})$ ➔ **Case 3**

Check **regularity condition**:

$3*(n/4)\lg(n/4) \le (3/4)n\lg n = c * f(n), \quad (3/4 \le c < 1)$

➔ $T(n) = \Theta(n\lg n)$

**here Example 5

Case 1: if $f(n) = O(n^{\log_b a - \varepsilon})$ for some $\varepsilon > 0$, then: $T(n) = \Theta(n^{\log_b a})$

Case 2: if $f(n) = \Theta(n^{\log_b a})$, then: $T(n) = \Theta(n^{\log_b a} \lg n)$

Case 3: if $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some $\varepsilon > 0$, and if

$af(n/b) \leq cf(n)$ for some $c < 1$ and all sufficiently large n, then: $T(n) = \Theta(f(n))$

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

$$a = 2,\ b = 2,\ \log_2 2 = 1$$

- Compare n with $f(n) = n\lg n$

Is this a candidate for **Case 3** ?

- **Case 3:** if $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some $\varepsilon > 0$

In other words, **If** $n\lg n \geq n^1.n^\varepsilon$ for some $\varepsilon > 0$

- $\lg n$ is asymptotically **less than** n^ε for any $\varepsilon > 0$!!

Therefore, this in **not Case 3**! (somewhere between Case 2 & 3)

Exercise:

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

- Use Iteration Method to show that

$$T(n) = n\lg^2 n$$

$$\begin{aligned} T(n) &= n\lg n + 2T(n/2) \\ &= n\lg n + 2(n/2 * \lg(n/2) + 2T(n/4)) \\ &= n\lg n + n\lg(n/2) + 2^2 T(n/2^2) \\ &= \ldots \end{aligned}$$

Changing variables

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

Try to transform the recurrence to one that you have seen before

- Rename: $m = \lg n \Rightarrow n = 2^m$

  $$T(2^m) = 2T(2^{m/2}) + m$$

- Rename: $k = m$ and $S(k) = T(2^k)$

  $$S(k) = 2S(k/2) + k \Rightarrow S(k) = \Theta(k * \lg k)$$

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

**See Page 86 of the Textbook.

Exercise:

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

- Use Iteration Method to show that

$$T(n) = \Theta(\lg n * \lg\lg n)$$

$$
\begin{aligned}
T(n) &= \lg n + 2T(n^{1/2}) \\
&= \lg n + 2(\lg(n^{1/2}) + 2T(n^{1/4})) \\
&= \lg n + \lg n + 2^2T(n^{1/4}) \\
&= \lg n + \lg n + \ldots + 2^kT(2) \qquad k=\lg\lg n
\end{aligned}
$$

Appendix A

Summations

$$\sum_{k=1}^{n} k = 1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$$

$$\sum_{k=1}^{n} k^2 = 1^2 + 2^2 + 3^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\sum_{k=1}^{n} k^3 = 1^3 + 2^3 + 3^3 + \cdots + n^3 = \frac{n^2(n+1)^2}{4}$$

$$\sum_{k=0}^{n} x^k = 1 + x + x^2 + \cdots + x^n = \frac{x^{n+1} - 1}{x - 1} \quad \text{If } |x| < 1 \text{ then } \lim_{n \to \infty} \sum_{k=0}^{n} x^k = \frac{1}{1-x}$$

Harmonic Series $H_n = \sum_{k=1}^{n} \frac{1}{k} = 1 + \frac{1}{2} + \frac{1}{3} + \cdots \frac{1}{n} = \ln n + O(1)$