

Chapter 4 *Recurrences*

1

Outlines: Recurrences

- ❖ Solving Recurrences
 - **Substitution Method**
 - **Iteration Method**
 - **The Master Theorem**

2

Solving Recurrences

- ❖ A **recurrence** is an equation or inequality that describes a function in terms of itself by using smaller inputs
- ❖ The expression:

$$T(n) = \begin{cases} c & n = 1 \\ 2T\left(\frac{n}{2}\right) + cn & n > 1 \end{cases}$$

- ❖ is a *recurrence*.

3

Solving Recurrences

- ❖ Examples:
 - $T(n) = 2T(n/2) + \Theta(n) \rightarrow T(n) = \Theta(n \lg n)$
 - $T(n) = 2T(n/2) + n \rightarrow T(n) = \Theta(n \lg n)$
 - $T(n) = 2T(n/2) + 17 + n \rightarrow T(n) = \Theta(n \lg n)$
- ❖ Three methods for solving recurrences
 - **Substitution method**
 - **Iteration method**
 - **Master method**

4

Recurrence Examples

$$T(n) = \begin{cases} 0 & n = 0 \\ c + T(n-1) & n > 0 \end{cases}$$

$$T(n) = \begin{cases} 0 & n = 0 \\ n + T(n-1) & n > 0 \end{cases}$$

$$T(n) = \begin{cases} c & n = 1 \\ 2T\left(\frac{n}{2}\right) + c & n > 1 \end{cases}$$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

5

Substitution Method

- ❖ The substitution method
 - “making a good guess method”
 - Guess the form of the answer, then
 - use **induction** to find the constants and show that solution works

- ❖ Our goal: show that

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

6

Substitution Method

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

❖ Thus, we need to show that $T(n) \leq c n \lg n$ with an appropriate choice of c

➤ **Inductive hypothesis: assume**

$$T(n/2) \leq c (n/2) \lg (n/2)$$

➤ **Substitute back into recurrence to show that $T(n) \leq c n \lg n$ follows, when $c \geq 1$**

$$\begin{aligned} \square T(n) &= 2T(n/2) + n \\ &\leq 2(c(n/2) \lg(n/2)) + n \\ &= cn \lg(n/2) + n \\ &= cn \lg n - cn \lg 2 + n \\ &= cn \lg n - cn + n \\ &\leq cn \lg n && \text{for } c \geq 1 \\ &= O(n \lg n) && \text{for } c \geq 1 \end{aligned}$$

7

Substitution Method

• Consider

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

– **Simplify** it by letting $m = \lg n \rightarrow n = 2^m$

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

– **Rename** $S(m) = T(2^m)$

$$- S(m) = 2S(m/2) + m = O(m \lg m)$$

– Changing back from $S(m)$ to $T(n)$, we obtain

$$- T(n) = T(2^m)$$

$$= S(m)$$

$$= O(m \lg m)$$

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

Iteration Method

- ❖ Iteration method:
 - Expand the recurrence k times
 - Stop the recurrence and work some algebra to express as a summation
 - Evaluate the summation

9

$$T(n) = \begin{cases} 0 & n = 0 \\ c + T(n-1) & n > 0 \end{cases}$$

$$\begin{aligned} \text{❖ } T(n) &= c + T(n-1) &= c + c + T(n-2) \\ &= 2c + T(n-2) &= 2c + c + T(n-3) \\ &= 3c + T(n-3) &\dots kc + T(n-k) \\ &= ck + T(n-k) \end{aligned}$$

❖ So far for $n \geq k$ we have

➤ $T(n) = ck + T(n-k)$

❖ To stop the recursion, we should have

➤ $n - k = 0 \rightarrow k = n$

➤ $T(n) = cn + T(0) = cn$

❖ Thus in general $T(n) = O(n)$

10

$$T(n) = \begin{cases} 0 & n = 0 \\ n + T(n-1) & n > 0 \end{cases}$$

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

$$= n + n-1 + T(n-2)$$

$$= n + n-1 + n-2 + T(n-3)$$

$$= n + n-1 + n-2 + n-3 + T(n-4)$$

$$= \dots \sum_{i=n-k+1}^n i + T(n-k)$$

$$= n + n-1 + n-2 + n-3 + \dots + (n-k+1) + T(n-k)$$

$$= \sum_{i=1}^n i + T(0) = \sum_{i=1}^n i + 0 = n \frac{n+1}{2} \text{ for } n \geq k$$

$$\diamond \text{ To stop the recursion, we should have } n - k = 0 \rightarrow k = n$$

$$T(n) = n \frac{n+1}{2} = O(n^2)$$

11

$$T(n) = \begin{cases} c & n = 1 \\ 2T\left(\frac{n}{2}\right) + c & n > 1 \end{cases}$$

$$\begin{aligned} \diamond T(n) &= 2T(n/2) + c & 1 \\ &= 2(2T(n/2/2) + c) + c & 2 \\ &= 2^2 T(n/2^2) + 2c + c & \\ &= 2^2(2T(n/2^2/2) + c) + (2^2-1)c & 3 \\ &= 2^3 T(n/2^3) + 4c + 3c & \\ &= 2^3 T(n/2^3) + (2^3-1)c & \\ &= 2^3(2T(n/2^3/2) + c) + 7c & 4 \\ &= 2^4 T(n/2^4) + (2^4-1)c & \\ &= \dots & \\ &= 2^k T(n/2^k) + (2^k - 1)c & k \end{aligned}$$

12

$$T(n) = \begin{cases} c & n = 1 \\ 2T\left(\frac{n}{2}\right) + c & n > 1 \end{cases}$$

❖ So far for $n \geq k$ we have

$$\triangleright T(n) = 2^k T(n/2^k) + (2^k - 1)c$$

❖ To stop the recursion, we should have

$$\triangleright n/2^k = 1 \rightarrow k = \lg n$$

$$\begin{aligned} \triangleright T(n) &= 2^{\lg n} T(n/2^{\lg n}) + (2^{\lg n} - 1)c \\ &= n T(n/n) + (n - 1)c \\ &= n T(1) + (n-1)c \\ &= nc + (n-1)c \\ &= nc + nc - c = 2cn - c = cn - c/2 \\ &\leq cn = O(n) \text{ for all } n \geq \frac{1}{2} \end{aligned}$$

13

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

$$\begin{aligned} \triangleright T(n) &= aT(n/b) + cn && 1 \\ &= a(aT(n/b/b) + cn/b) + cn && 2 \\ &= a^2T(n/b^2) + cna/b + cn \\ &= a^2T(n/b^2) + cn(a/b + 1) \\ &= a^2(aT(n/b^2/b) + cn/b^2) + cn(a/b + 1) && 3 \\ &= a^3T(n/b^3) + cn(a^2/b^2) + cn(a/b + 1) \\ &= a^3T(n/b^3) + cn(a^2/b^2 + a/b + 1) \\ &= \dots \\ &= a^kT(n/b^k) + cn(a^{k-1}/b^{k-1} + a^{k-2}/b^{k-2} + \dots + \\ &\quad a^2/b^2 + a/b + 1) && k \end{aligned}$$

14

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

❖ So we have

$$\triangleright T(n) = a^k T(n/b^k) + cn(a^{k-1}/b^{k-1} + \dots + a^2/b^2 + a/b + 1)$$

❖ To stop the recursion, we should have

$$\triangleright n/b^k = 1 \quad \rightarrow n = b^k \quad \rightarrow k = \log_b n$$

$$\begin{aligned} \triangleright T(n) &= a^k T(1) + cn(a^{k-1}/b^{k-1} + \dots + a^2/b^2 + a/b + 1) \\ &= a^k c + cn(a^{k-1}/b^{k-1} + \dots + a^2/b^2 + a/b + 1) \\ &= ca^k + cn(a^{k-1}/b^{k-1} + \dots + a^2/b^2 + a/b + 1) \\ &= cna^k/b^k + cn(a^{k-1}/b^{k-1} + \dots + a^2/b^2 + a/b + 1) \\ &= cn(a^k/b^k + \dots + a^2/b^2 + a/b + 1) \end{aligned}$$

15

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

❖ So with $k = \log_b n$

$$\triangleright T(n) = cn(a^k/b^k + \dots + a^2/b^2 + a/b + 1)$$

❖ What if $a = b$?

$$\begin{aligned} \triangleright T(n) &= cn(1 + \dots + 1 + 1 + 1) \quad //k+1 \text{ times} \\ &= cn(k + 1) \\ &= cn(\log_b n + 1) \\ &= \Theta(n \log_b n) \end{aligned}$$

16

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

❖ So with $k = \log_b n$

➤ $T(n) = cn(a^k/b^k + \dots + a^2/b^2 + a/b + 1)$

❖ What if $a < b$?

➤ Recall that

$$\square \sum(x^k + x^{k-1} + \dots + x + 1) = (x^{k+1} - 1)/(x - 1)$$

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \dots + \frac{a}{b} + 1 = \frac{(a/b)^{k+1} - 1}{(a/b) - 1} = \frac{1 - (a/b)^{k+1}}{1 - (a/b)} < \frac{1}{1 - a/b}$$

➤ $T(n) = cn \cdot \Theta(1) = \Theta(n)$

17

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

❖ So with $k = \log_b n$

➤ $T(n) = cn(a^k/b^k + \dots + a^2/b^2 + a/b + 1)$

❖ What if $a > b$?

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \dots + \frac{a}{b} + 1 = \frac{(a/b)^{k+1} - 1}{(a/b) - 1} = \Theta\left(\frac{(a/b)^{k+1}}{(a/b) - 1}\right)$$

➤ $T(n) = cn \cdot \Theta(a^k / b^k)$

$$= cn \cdot \Theta(a^{\log_b n} / b^{\log_b n}) = cn \cdot \Theta(a^{\log_b n} / n)$$

recall logarithm fact: $a^{\log_b n} = n^{\log_b a}$

$$= cn \cdot \Theta(n^{\log_b a} / n) = \Theta(cn \cdot n^{\log_b a} / n)$$

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

18

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

❖ So...

$$T(n) = \begin{cases} \Theta(n) & a < b \\ \Theta(n \log_b n) & a = b \\ \Theta(n^{\log_b a}) & a > b \end{cases}$$

19

The Master Theorem

- ❖ Given: a *divide and conquer* algorithm
 - An algorithm that divides the problem of size n into a subproblems, each of size n/b
 - Let the cost of each stage (i.e., the work to divide the problem + combine solved subproblems) be described by the function $f(n)$
- ❖ Then, the Master Theorem gives us a cookbook for the algorithm's running time:

20

The Master Theorem

❖ if $T(n) = aT(n/b) + f(n)$ where $a \geq 1$ & $b > 1$

❖ then

$$T(n) = \left\{ \begin{array}{ll} \Theta(n^{\log_b a}) & f(n) = O(n^{\log_b a - \epsilon}) \\ \Theta(n^{\log_b a} \lg n) & f(n) = \Theta(n^{\log_b a}) \\ \Theta(f(n)) & f(n) = \Omega(n^{\log_b a + \epsilon}) \& \\ & af(n/b) < cf(n) \text{ for large } n \end{array} \right\} \begin{array}{l} \epsilon > 0 \\ c < 1 \end{array}$$

21

Understanding Master Theorem

- ❖ In each of the three cases, we are comparing $f(n)$ with $n^{\log_b a}$, the solution to the recurrence is determined by the larger of the two functions.
- ❖ In case 1, if the function $n^{\log_b a}$ is the larger, then the solution $T(n) = \Theta(n^{\log_b a})$.
- ❖ In case 3, if the function $f(n)$ is the larger, then the solution is $T(n) = \Theta(f(n))$.
- ❖ In case 2, if the two functions are the same size, then the solution is $T(n) = \Theta(n^{\log_b a} \lg n) = \Theta(f(n) \lg n)$.

22

Understanding Master Theorem

- ❖ In case 1, not only must $f(n)$ be smaller than $n^{\log_b a}$, it must be polynomially smaller. That is $f(n)$ must be asymptotically smaller than $n^{\log_b a}$ by a factor of n^ϵ for some constant $\epsilon > 0$.
- ❖ In case 3, not only must $f(n)$ be larger than $n^{\log_b a}$, it must be polynomially larger and in addition satisfy the “regularity” condition that:

$$a f(n/b) \leq c f(n).$$

23

Understanding Master Theorem

- ❖ It is important to realize that the three cases do not cover all the possibilities for $f(n)$.
- ❖ There is a **gap** between **cases 1 and 2** when $f(n)$ is smaller than $n^{\log_b a}$ but not polynomially smaller.
- ❖ There is a **gap** between **cases 2 and 3** when $f(n)$ is larger than $n^{\log_b a}$ but not polynomially larger.
- ❖ If $f(n)$ falls into one of these gaps, or if the regularity condition in case 3 fails to hold, the master method cannot be used to solve the recurrence.

24

Using The Master Method Case 1

- ❖ $T(n) = 9T(n/3) + n$
 - $a=9, b=3, f(n) = n$
 - $n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$
 - Since $f(n) = O(n^{\log_3 9 - \epsilon}) = O(n^{2-0.5}) = O(n^{1.5})$
 - where $\epsilon=0.5$
 - case 1 applies:

$$T(n) = \Theta(n^{\log_b a}) \text{ when } f(n) = O(n^{\log_b a - \epsilon})$$

- Thus the solution is
 - $T(n) = \Theta(n^2)$

25

Using The Master Method Case 2

- ❖ $T(n) = T(2n/3) + 1$
 - $a=1, b=3/2, f(n) = 1$
 - $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$
 - Since $f(n) = \Theta(n^{\log_b a}) = \Theta(1)$
 - case 2 applies:

$$T(n) = \Theta(n^{\log_b a} \lg n) \text{ when } f(n) = \Theta(n^{\log_b a})$$

- Thus the solution is
 - $T(n) = \Theta(\lg n)$

26

Using The Master Method Case 3

❖ $T(n) = 3T(n/4) + n \lg n$

➤ $a=3, b=4, f(n) = n \lg n$

➤ $n^{\log_b a} = n^{\log_4 3} = n^{0.793} = n^{0.8}$

➤ Since $f(n) = \Omega(n^{\log_4 3 + \epsilon}) = \Omega(n^{0.8+0.2}) = \Omega(n)$

➤ where $\epsilon \approx 0.2$, and for sufficiently large n ,

□ $a \cdot f(n/b) = 3(n/4) \lg(n/4) < (3/4) n \lg n$ for $c = 3/4$

➤ case 3 applies:

$$T(n) = \Theta(f(n)) \text{ when } f(n) = \Omega\left(n^{\log_b a + \epsilon}\right)$$

➤ Thus the solution is

□ $T(n) = \Theta(n \lg n)$

27

When the Master Method does not apply to recurrence

❖ $T(n) = 2T(n/2) + n \lg n$

➤ $a = 2, b = 2, f(n) = n \lg n$, and $n^{\log_b a} = n$

➤ Note that $f(n) = n \lg n$ is asymptotically larger than $n^{\log_b a} = n$.

➤ The problem is that it is not polynomially larger.

➤ The ratio $f(n) / n^{\log_b a} = (n \lg n) / n = \lg n$ is asymptotically less than n^ϵ for any positive constant ϵ .

➤ The recurrence falls into the gap between case 2 and case 3.

28

When the Master Method does not apply to recurrence

❖ $T(n) = 2^n T(n/2) + n^n$

➤ Does not apply (a is not constant)

❖ $T(n) = 2T(n/2) + n/\log n$

➤ Does not apply (non-polynomial difference between $f(n)$ and $n^{\log_b a}$)

❖ $T(n) = 0.5T(n/2) + 1/n$

➤ Does not apply ($a < 1$)

❖ $T(n) = 64T(n/8) - n^2 \log^n$

➤ Does not apply ($f(n)$ is not positive)

❖ $T(n) = T(n/2) + n(2 - \cos n)$

➤ Does not apply. We are in Case 3, but the regularity condition is violated.