

Data Structures and Algorithms

(CS210A)

Lecture 16:

- Solving recurrences
that occur frequently in the analysis of algorithms.

Commonly occurring recurrences

$$T(n) = \alpha \log(n) + \beta T(n/5)$$

Methods for solving **Recurrences**

commonly occurring in
algorithm analysis

Methods for solving common Recurrences

- **Unfolding** the recurrence.
- **Guessing** the solution and then proving by induction.
- **A General solution** for a large class of recurrences (**Master theorem**)

Solving a recurrence by **unfolding**

Let $T(1) = 1$,

$T(n) = cn + 4 T(n/2)$ for $n > 1$, where c is some positive constant

Solving the recurrence for $T(n)$ by **unfolding** (expanding)

$$\begin{aligned} T(n) &= cn + 4 T(n/2) \\ &= cn + 2cn + 4^2 T(n/2^2) \\ &= cn + 2cn + 4cn + 4^3 T(n/2^3) \\ &= cn + 2cn + 4cn + 8cn + \dots + 4^{\log_2 n} \\ &= \underbrace{cn + 2cn + 4cn + 8cn + \dots}_{\text{A geometric increasing series with } \log n \text{ terms and common ratio } 2} + n^2 \end{aligned}$$

A geometric increasing series with $\log n$ terms and common ratio 2

$$= O(n^2)$$

Solving a recurrence by guessing and then proving by induction

$$T(1) = c_1$$

$$T(n) = 2T(n/2) + c_2 n$$

Guess: $T(n) \leq a n \log n + b$ for some con.

Proof by induction:

Base case: holds true if $b \geq c_1$

Induction hypothesis: $T(k) \leq a k \log k + b$ for all $k < n$

To prove: $T(n) \leq a n \log n + b$

Proof: $T(n) = 2T(n/2) + c_2 n$

$$\leq 2\left(a \frac{n}{2} \log \frac{n}{2} + b\right) + c_2 n \quad // \text{ by induction hypothesis}$$

$$= a n \log n - a n + 2b + c_2 n$$

$$= a n \log n + b + (b + c_2 n - a n)$$

$$\leq a n \log n + b \quad \text{if } a \geq b + c_2$$

It looks similar/identical to the recurrence of merge sort.

So we guess

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

These inequalities can be satisfied simultaneously by selecting $b = c_1$ and $a = c_1 + c_2$

Hence $T(n) \leq (c_1 + c_2) n \log n + c_1$ for all value of n .

So $T(n) = O(n \log n)$

Solving a recurrence by **guessing** and then proving by **induction**

Key points:

- You have to make a right guess (past experience may help)
- What if your guess is too loose ?
- Be careful in the **induction step**.

Solving a recurrence by guessing and then proving by induction

Exercise: Find error in the following reasoning.

For the recurrence $T(1) = c_1$, and $T(n) = 2T(n/2) + c_2 n$,

one guesses $T(n) = O(n)$

Proposed (wrong) proof by induction:

Induction hypothesis: $T(k) \leq ak$ for all $k < n$

$$\begin{aligned} T(n) &= 2T(n/2) + c_2 n \\ &\leq 2\left(a \frac{n}{2}\right) + c_2 n \quad // \text{ by induction hypothesis} \\ &= an + c_2 n \\ &= O(n) \end{aligned}$$

A General Method for solving a large class of Recurrences

Solving a large class of recurrences

$$T(1) = 1,$$

$$T(n) = f(n) + a T(n/b)$$

Where

- a and b are constants and $b > 1$
- $f(n)$ is a *multiplicative* function:

$$f(xy) = f(x)f(y)$$

AIM : To solve $T(n)$ for $n = b^k$

Warm-up

$f(n)$ is a *multiplicative* function:

$$f(xy) = f(x)f(y)$$

$$f(1) = 1$$

$$f(a^i) = f(a)^i$$

$$f(n^{-1}) = 1/f(n)$$

Example of a *multiplicative* function : $f(n) = n^\alpha$

Question: Can you express $a^{\log_b c}$ as power of c ?

Answer: $c^{\log_b a}$

Solving a slightly general class of recurrences

$$\begin{aligned}T(n) &= f(n) + a T(n/b) \\&= f(n) + a f(n/b) + a^2 T(n/b^2) \\&= f(n) + a f(n/b) + a^2 f(n/b^2) + a^3 T(n/b^3) \\&= \dots \\&= \underbrace{f(n) + a f(n/b) + \dots + a^i f(n/b^i) + \dots + a^{k-1} f(n/b^{k-1})}_{\text{... after rearranging ...}} + a^k T(1) \\&= \left(\sum_{i=0}^{k-1} a^i f(n/b^i) \right) + a^k \\&= a^k + \sum_{i=0}^{k-1} a^i f(n/b^i) \\&\quad \dots \text{ continued to the next page } \dots\end{aligned}$$

$$\begin{aligned}
T(n) &= a^k + \sum_{i=0}^{k-1} a^i f(n/b^i) \\
&\quad (\text{since } f \text{ is multiplicative}) \\
&= a^k + \sum_{i=0}^{k-1} a^i f(n)/f(b^i) \\
&= a^k + \sum_{i=0}^{k-1} a^i f(n)/(f(b))^i \\
&= a^k + f(n) \sum_{i=0}^{k-1} a^i/(f(b))^i \\
&= a^k + f(n) \underbrace{\sum_{i=0}^{k-1} (a/f(b))^i}_{\text{A geometric series}} \\
&= a^k + (f(b))^k \sum_{i=0}^{k-1} (a/f(b))^i
\end{aligned}$$

Case 1: $a = f(b)$, $T(n) = a^k(k+1) = O(a^{\log_b n} \log_b n) = O(n^{\log_b a} \log_b n)$

$$\begin{aligned}
T(n) &= a^k + \sum_{i=0}^{k-1} a^i f(n/b^i) \\
&\text{(since } f \text{ is multiplicative)} \\
&= a^k + \sum_{i=0}^{k-1} a^i f(n)/f(b^i) \\
&= a^k + \sum_{i=0}^{k-1} a^i f(n)/(f(b))^i \\
&= a^k + f(n) \sum_{i=0}^{k-1} a^i/(f(b))^i \\
&= a^k + f(n) \sum_{i=0}^{k-1} (a/f(b))^i \\
&= a^k + (f(b))^k \sum_{i=0}^{k-1} (a/f(b))^i
\end{aligned}$$

For $a < f(b)$, the sum of this series is bounded by

$$\frac{1}{1 - \frac{a}{f(b)}} = O(1)$$

Case 2: $a < f(b)$, $T(n) = a^k + (f(b))^k \cdot O(1) = O((f(b))^k) = O(f(n))$

$$\begin{aligned}
T(n) &= a^k + \sum_{i=0}^{k-1} a^i f(n/b^i) \\
&\text{(since } f \text{ is multiplicative)} \\
&= a^k + \sum_{i=0}^{k-1} a^i f(n)/f(b^i) \\
&= a^k + \sum_{i=0}^{k-1} a^i f(n)/(f(b))^i \\
&= a^k + f(n) \sum_{i=0}^{k-1} a^i/(f(b))^i \\
&= a^k + f(n) \sum_{i=0}^{k-1} (a/f(b))^i \\
&= a^k + (f(b))^k \sum_{i=0}^{k-1} (a/f(b))^i
\end{aligned}$$

For $a > f(b)$, the sum of this series is equal to

$$\frac{\left(\frac{a}{f(b)}\right)^k - 1}{\frac{a}{f(b)} - 1}$$

Case 3: $a > f(b)$, $T(n) = a^k + O(a^k) = O(n^{\log_b a})$

Three cases

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

Case 1: $a = f(b)$,

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

Case 2: $a < f(b)$,

$$T(n) = O(f(n))$$

Case 3: $a > f(b)$,

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

Master theorem

$$T(1) = 1,$$

$$T(n) = f(n) + a T(n/b) \text{ where } f \text{ is } \textit{multiplicative}.$$

There are the following solutions

$$\text{Case 1: } a = f(b), T(n) = n^{\log_b a} \log_b n$$

$$\text{Case 2: } a < f(b), T(n) = O(f(n))$$

$$\text{Case 3: } a > f(b), T(n) = O(n^{\log_b a})$$

Examples

Master theorem

If $T(1) = 1$, and $T(n) = f(n) + a T(n/b)$ where f is *multiplicative*, then there are the following solutions

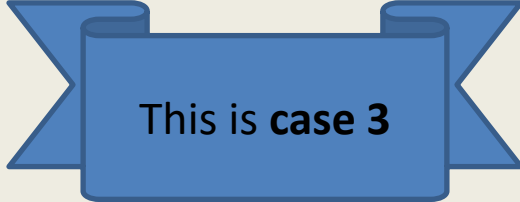
Case 1: $a = f(b)$, $T(n) = n^{\log_b a} \log_b n$

Case 2: $a < f(b)$, $T(n) = O(f(n))$

Case 3: $a > f(b)$, $T(n) = O(n^{\log_b a})$

Example 1: $T(n) = n + 4 T(n/2)$

Solution: $T(n) = O(n^2)$



This is case 3

Master theorem

If $T(1) = 1$, and $T(n) = f(n) + a T(n/b)$ where f is *multiplicative*, then there are the following solutions

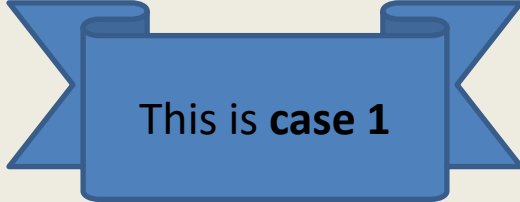
Case 1: $a = f(b)$, $T(n) = n^{\log_b a} \log_b n$

Case 2: $a < f(b)$, $T(n) = O(f(n))$

Case 3: $a > f(b)$, $T(n) = O(n^{\log_b a})$

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

Solution: $T(n) = O(n^2 \log_2 n)$



This is case 1

Master theorem

If $T(1) = 1$, and $T(n) = f(n) + a T(n/b)$ where f is *multiplicative*, then there are the following solutions

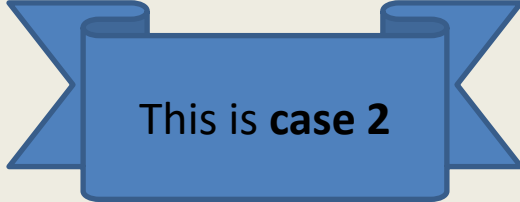
Case 1: $a = f(b)$, $T(n) = n^{\log_b a} \log_b n$

Case 2: $a < f(b)$, $T(n) = O(f(n))$

Case 3: $a > f(b)$, $T(n) = O(n^{\log_b a})$

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

Solution: $T(n) = O(n^3)$



This is case 2

Master theorem

If $T(1) = 1$, and $T(n) = f(n) + a T(n/b)$ where f is *multiplicative*, then there are the following solutions

Case 1: $a = f(b)$, $T(n) = n^{\log_b a} \log_b n$

Case 2: $a < f(b)$, $T(n) = O(f(n))$

Case 3: $a > f(b)$, $T(n) = O(n^{\log_b a})$

Example 4: $T(n) = 2 n^{1.5} + 3 T(n/2)$

Solution: $T(n) =$

We **can not** apply master theorem **directly** since $f(n) = 2 n^{1.5}$ is not multiplicative.

Master theorem

If $T(1) = 1$, and $T(n) = f(n) + a T(n/b)$ where f is *multiplicative*, then there are the following solutions

Case 1: $a = f(b)$, $T(n) = n^{\log_b a} \log_b n$

Case 2: $a < f(b)$, $T(n) = O(f(n))$

Case 3: $a > f(b)$, $T(n) = O(n^{\log_b a})$

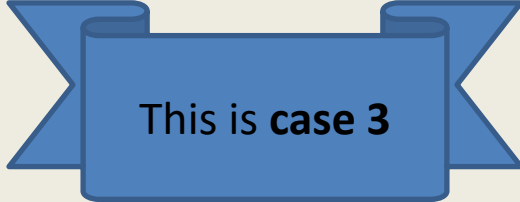
Example 4: $T(n) = 2 n^{1.5} + 3 T(n/2)$

Solution: $G(n) = T(n)/2$

→ $G(n) = n^{1.5} + 3 G(n/2)$

→ $G(n) = O(n^{\log_2 3}) = O(n^{1.58})$

→ $T(n) = O(n^{1.58})$.



This is case 3

Master theorem

If $T(1) = 1$, and $T(n) = f(n) + a T(n/b)$ where f is *multiplicative*, then there are the following solutions

Case 1: $a = f(b)$, $T(n) = n^{\log_b a} \log_b n$

Case 2: $a < f(b)$, $T(n) = O(f(n))$

Case 3: $a > f(b)$, $T(n) = O(n^{\log_b a})$

Example 6: $T(n) = T(\sqrt{n}) + c n$

Solution: $T(n) =$

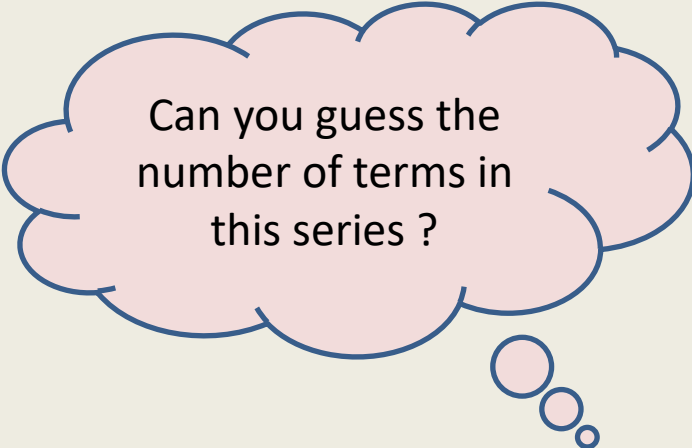
We can **not** apply master theorem **directly** since $T(\sqrt{n}) \neq T(n/b)$ for any constant b .

Solving $T(n) = T(\sqrt{n}) + c n$ using the method of unfolding

$$\begin{aligned} T(n) &= c n + T(\sqrt{n}) \\ &= c n + c \sqrt{n} + T(\sqrt[4]{n}) \\ &= c n + c \sqrt{n} + c \sqrt[4]{n} + T(\sqrt[8]{n}) \\ &= \underbrace{c n + c \sqrt{n} + \dots c \sqrt[i]{n} + \dots + T(1)} \end{aligned}$$

A series which is decreasing at a rate faster than any geometric series

$$= O(n)$$



Can you guess the number of terms in this series ?



$\lceil \log \log n \rceil$

Master theorem

If $T(1) = 1$, and $T(n) = f(n) + a T(n/b)$ where f is *multiplicative*, then there are the following solutions

Case 1: $a = f(b)$, $T(n) = n^{\log_b a} \log_b n$

Case 2: $a < f(b)$, $T(n) = O(f(n))$

Case 3: $a > f(b)$, $T(n) = O(n^{\log_b a})$

Example 5: $T(n) = n (\log n)^2 + 2 T(n/2)$

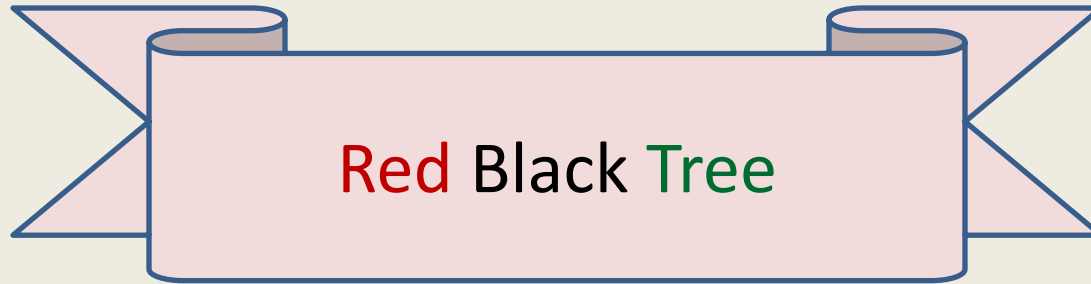
Solution: $T(n) =$ Using the method of “**unfolding**”, it can be shown that $T(n) = O(n (\log n)^3)$.

Homework

Solve the following recurrences systematically (if possible by various methods). Assume that $T(1) = 1$ for all these recurrences.

- $T(n) = 1 + 2 T(n/2)$
- $T(n) = n^3 + 2 T(n/2)$
- $T(n) = n^2 + 7 T(n/3)$
- $T(n) = n / \log n + 2T(n/2)$
- $T(n) = 1 + T(n/5)$
- $T(n) = \sqrt{n} + 2 T(n/4)$
- $T(n) = 1 + T(\sqrt{n})$
- $T(n) = n + T(9n/10)$
- $T(n) = \log n + T(n/4)$

Next 2 classes



Either miss both or attend both the classes 😊