Data Structures and Algorithms

(ESO207)

Lecture 16:

- Solving recurrences

  that occur frequently <u>in the analysis of algorithms</u>.

Commonly occurring recurrences

$$T(n) = \ldots T(n/5)$$

Methods for solving Recurrences

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

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

$$T(n) = cn + 4\,T(n/2)$$

$$= cn + 2cn + 4^2\,T(n/2^2)$$

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

$$= cn + 2\,cn + 4cn + 8cn + \ldots + 4^{\log_2 n}$$

$$= \underbrace{cn + 2\,cn + 4cn + 8cn + \ldots + n^2}_{\text{A geometric increasing series with } \log n \text{ terms and common ratio } 2}$$

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: $T(n) \le a\, n \log n + b$ for some con

It looks similar/identical to the recurrence of merge sort.
So we guess
$T(n) = O(n \log n)$

Proof by induction:

Base case: holds true if $\boxed{b \ge c_1}$

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

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

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

$\le 2(a \frac{n}{2} \log \frac{n}{2} + b) + c_2\, n$ // by induction hypothesis

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

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

$\le a\, n \log n + b$ **if** $\boxed{a \ge b + c_2}$

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

Hence $T(n) \le (c_1 + c_2)\, n \log n + c_1$ **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) = 2**T**(n/2) + $c_2\ n$**,**

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

Proposed (wrong)**proof by induction**:

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

$$
\begin{aligned}
\mathbf{T}(n) &= 2\mathbf{T}(n/2) + c_2\ n \\
&\le 2(a\tfrac{n}{2}) + c_2\ n \quad \text{// by induction hypothesis} \\
&= an + c_2 n \\
&= \mathbf{O}(n)
\end{aligned}
$$

A General Method for solving a large class of Recurrences

Solving a large class of recurrences

$\mathbf{T}(1) = 1$,

$\mathbf{T}(n) = \mathbf{f}(n) + a\,\mathbf{T}(n/b)$

Where

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

$$\mathbf{f}(xy) = \mathbf{f}(x)\mathbf{f}(y)$$

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)$^{\boldsymbol{i}}$

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

Example of a *multiplicative* function : f(n) = n^α

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

Answer: $c^{\log_b a}$

Solving a slightly general class of recurrences

$\mathbf{T}(n) = \mathbf{f}(n) + a\ \mathbf{T}(n/b)$

$= \mathbf{f}(n) + a\ \mathbf{f}(n/b) + a^2\mathbf{T}(n/b^2)$

$= \mathbf{f}(n) + a\ \mathbf{f}(n/b) + a^2\mathbf{f}(n/b^2) + a^3\mathbf{T}(n/b^3)$

$= \quad \ldots$

$= \underbrace{\mathbf{f}(n) + a\ \mathbf{f}(n/b) + \ldots + a^i\mathbf{f}(n/b^i) + \ldots + a^{k-1}\mathbf{f}(n/b^{k-1})} + a^k\mathbf{T}(1)$

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

… after rearranging …

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

… continued to the next page …

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

(since **f** is multiplicative)

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

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

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

$= a^k + \mathbf{f}(n) \underbrace{\sum_{i=0}^{k-1} (a/\mathbf{f}(b))^i}_{\text{A geometric series}}$

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

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

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

(since $\mathbf{f}$ is multiplicative)

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

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

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

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

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

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

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

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

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

(since $\mathbf{f}$ is multiplicative)

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

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

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

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

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

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

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

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

Three cases

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

Case 1: $a = \mathbf{f}(b)$**,**

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

Case 2: $a < \mathbf{f}(b)$**,**

$\mathbf{T}(n) = \mathbf{O}(\mathbf{f}(n))$

Case 3: $a > \mathbf{f}(b)$**,**

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

Master theorem

T(1) = 1,

T(n) = **f**(n) + a **T**(n/b) where **f** is *multiplicative*.

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**($n^{\log_b a}$)

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)

This is **case 3**

Solution: **T**(n)= $\boxed{\mathbf{O}(n^2)}$

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 2: **T**(n)= n^{2}+ 4 **T**(n/2)

This is **case 1**

Solution: **T**(n)= $\boxed{O(n^2 \log_2 n)}$

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)

This is **case 2**

Solution: **T**(n)= $\boxed{\mathbf{O}(\ n^3)}$

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 4: **T**(n)= 2 $n^{\mathbf{1.5}}$ + 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**($n^{\log_b a}$)

This is **case 3**

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**($n^{1.58}$).

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**($\sqrt{n}$) <> **T**(n/b) for any constant b.

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

$$\mathbf{T}(n) = c\,n + \mathbf{T}(\sqrt{n})$$

$$= c\,n + c\sqrt{n} + \mathbf{T}(\sqrt[4]{n})$$

$$= c\,n + c\sqrt{n} + c\,\sqrt[4]{n} + \mathbf{T}(\sqrt[8]{n})$$

$$= \underbrace{c\,n + c\sqrt{n} + \ldots\ c\,\sqrt[i]{n} + \ldots + \mathbf{T}(1)}_{\text{A series which is decreasing at a rate faster than any geometric series}}$$

$$= \mathbf{O}(n)$$

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