

Recurrences

4.3 The substitution method

The substitution method:

(i) 猜一個答案

(ii) 用歸納法證明

(for both upper and lower bounds)

範例：找到 $T(n) = 2T(\lfloor n/2 \rfloor) + n$ 的 upper bound

$$(T(1) = 1)$$

猜測 $T(n) = O(n \lg n)$

試著證明存在著常數 c 和 n_0 使得 $T(n) \leq cn \lg n$ 對於所有的 $n \geq n_0$ 都成立。

Basis: ($n=n_0$)

在 $n=1$ 時，沒有常數 c 能夠滿足 $T(1) \leq cn \lg n = 0$ 。

在 $n=2$ 時，任何大於等於 $T(n)/(n \lg n)$ 的常數 c 都會滿足 $T(n) \leq cn \lg n$ 。所以我們可以選擇

$$n_0 \geq 2 \text{ 且 } c \geq T(n_0)/(n_0 \lg n_0) \dots\dots\dots (1)$$

Induction: ($n > n_0$)

假設 $T(n) \leq cn \lg n$ 這個式子對於所有介在 n_0 和 $n-1$ 之間的 n 都成立。我們可以得到

$$\begin{aligned} T(n) &\leq 2(c\lfloor n/2 \rfloor \lg \lfloor n/2 \rfloor) + n \quad (\textit{Substitution}) \\ &\leq cn \lg (n/2) + n \\ &= cn \lg n - cn \lg 2 + n \\ &= cn \lg n - cn + n \\ &\leq cn \lg n , \end{aligned}$$

其中最後一步在 $c \geq 1$ 的時候成立.....(2)

根據(1)(2)，我們可以選擇 $n_0 = 2$ 以及 $c = \max\{1, T(2)/(2 \lg 2)\} = 2$ 讓 basis 及 induction 都成立。

Substitution Method

步驟 1. 猜測 $T(n) = O(g(n))$

步驟 2. 透過歸納法證明 $T(n) = O(g(n))$

⇒ 證明存在 c 和 n_0 使得

$T(n) \leq cg(n)$ 對於全部的 $n \geq n_0$ 都成立.....(1)

- ⇒ 如果我們已知 c 和 n_0 ，那麼我們就可以用歸納法證明(1)
- (a) Basis step: (1) 在 $n = n_0$ 時成立
 - (b) Induction step: (1) 在 $n > n_0$ 時成立
- ⇒ 如何找到 c 和 n_0 符合歸納法中的證明？
- (i) 找到符合 basis step 的 c 和 n_0 的條件
 - (ii) 找到符合 induction step 的 c 和 n_0 的條件
 - (iii) 綜合 (i) 和 (ii) 的條件

Subtleties:

(修改假設：減一個低次方項)

範例: $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$ ($T(1) = 1$)

猜測 $T(n) = O(n)$

試著證明 $T(n) \leq cn$

Basis: ok!

$$\begin{aligned}\text{Induction: } T(n) &= T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1 \\ &\leq c(\lfloor n/2 \rfloor + \lceil n/2 \rceil) + 1 \\ &= cn + 1\end{aligned}$$

無法證明 $T(n) \leq cn$!!!

試著證明 $T(n) \leq cn - b$

$$\begin{aligned}\text{Induction: } T(n) &\leq (c\lfloor n/2 \rfloor - b) + (c\lceil n/2 \rceil - b) + 1 \\ &= cn - 2b + 1 \\ &\leq cn - b,\end{aligned}$$

最後一步對於任意大於等於 1 的常數 b 都成立

Avoiding pitfalls:

$$T(n) = 2T(\lfloor n/2 \rfloor) + n$$

猜測 $T(n) = O(n)$ 。 試著證明 $T(n) \leq cn$ 。

$$\begin{aligned} \text{Induction: } T(n) &\leq 2c\lfloor n/2 \rfloor + n \\ &\leq cn + n \\ &= O(n) \Leftarrow \text{錯!!} \end{aligned}$$

Changing variable:

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

- 為了簡單起見，假設 $n=2^m$ 。變成

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

- 令 $S(m) = T(2^m)$ ，我們可以得到

$$S(m) = 2S(m/2) + m \text{ (把 } m \text{ 換成 } \lg n)$$

- 因為我們知道 $S(m) = O(m \lg m)$ ，所以可以推得 $T(n) = O(\lg n \lg \lg n)$

4.4 The iteration (recursion-tree) method

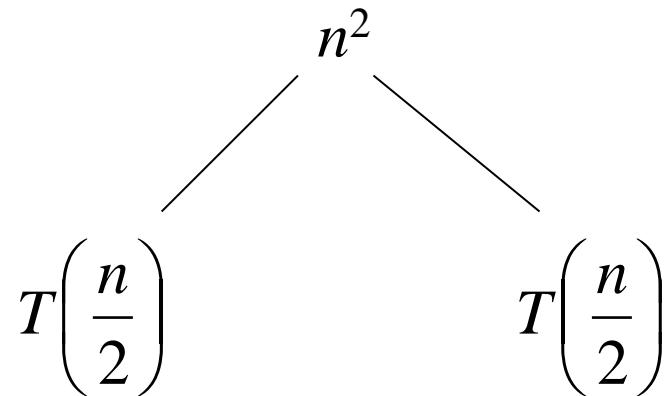
範例: $T(n)=3T(\lfloor n/4 \rfloor) + n$

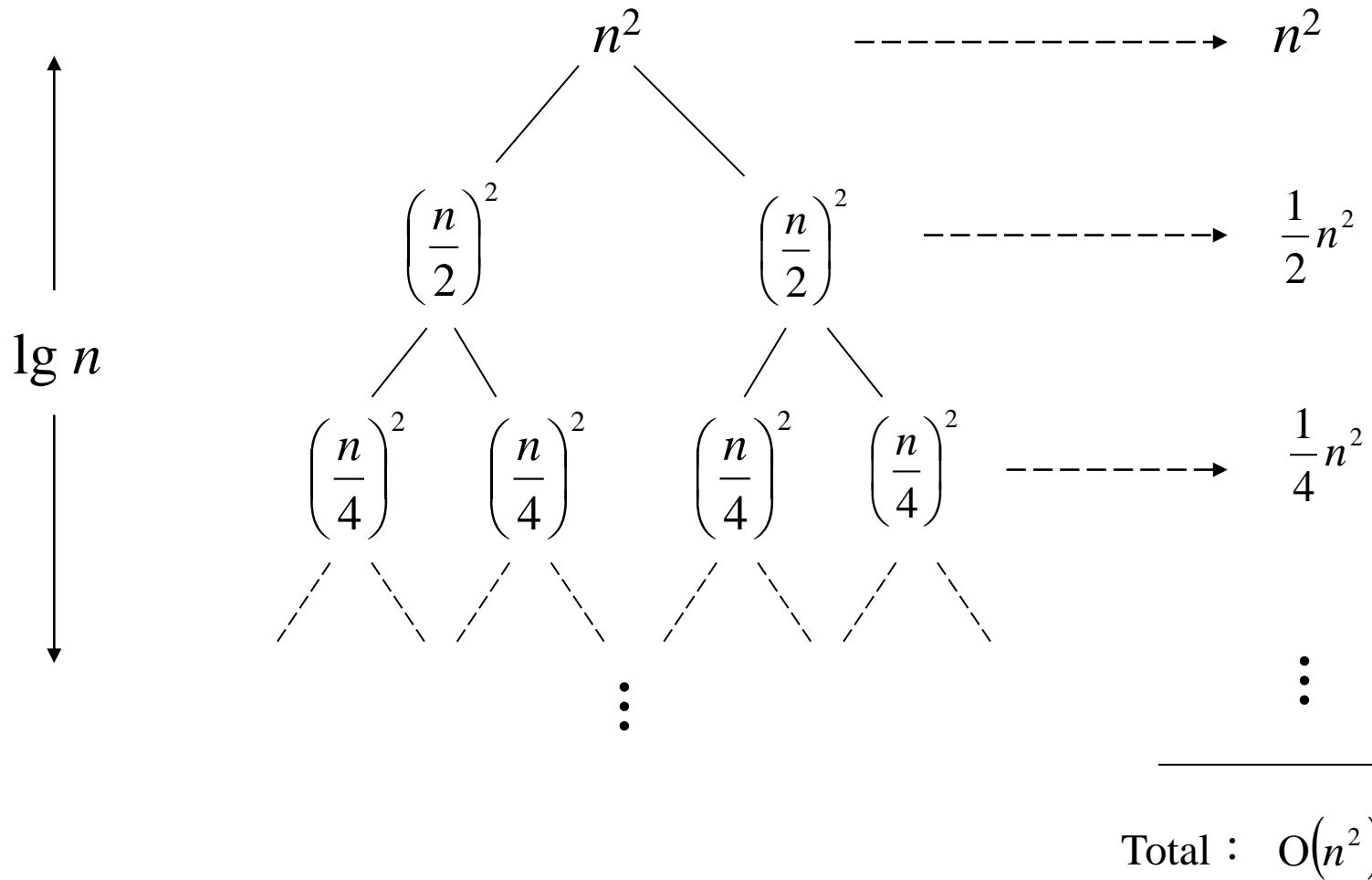
$$\begin{aligned} T(n) &= n + 3(\lfloor n/4 \rfloor + 3T(\lfloor n/16 \rfloor)) \\ &= n + 3\lfloor n/4 \rfloor + 9(\lfloor n/16 \rfloor + 3T(\lfloor n/64 \rfloor)) \\ &\quad \cdot \\ &\quad \cdot \qquad \text{(note that } n/(4^{\log_4 n}) \leq 1) \\ &\leq n + 3n/4 + 9n/16 + 27n/64 + \dots + \Theta(1) \\ &\leq n \sum_{i=0}^{\infty} \left(\frac{3}{4}\right)^i + \Theta(n^{\log_4 3}) = 4n + o(n) = O(n) \end{aligned}$$

Recursion trees: (for visualizing the iteration)

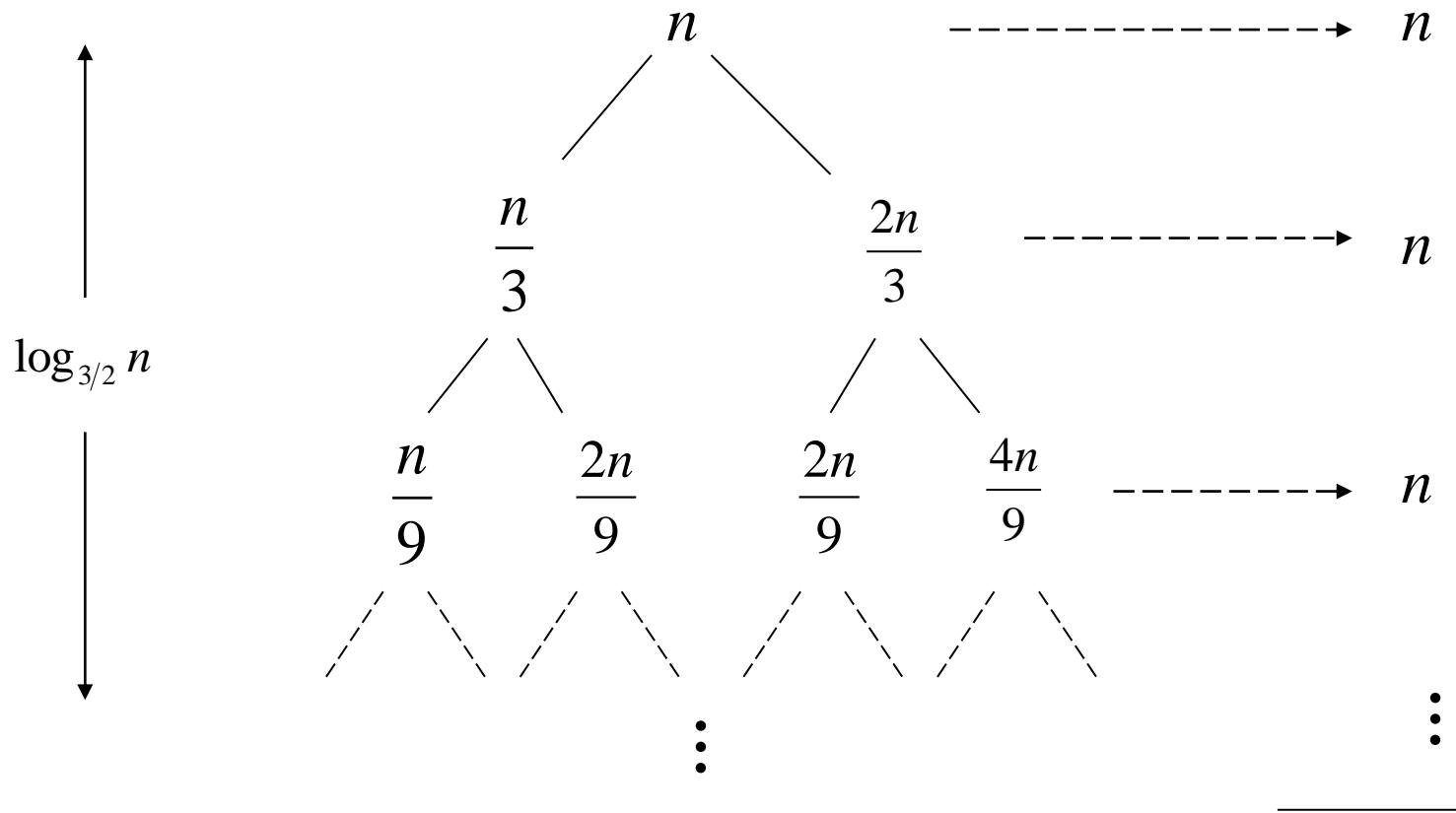
- $T(n) = 2T(n/2) + n^2$

$T(n)$





範例: $T(n) = T(n/3) + T(2n/3) + n$



Total : $O(n \lg n)$

4.5 The master method

Theorem 4.1 (Master theorem)

令 $a \geq 1$ 及 $b > 1$ 為常數，令 $f(n)$ 為一函數，且定義 $T(n)$ 為非負的整數，且其遞迴式為 $T(n) = aT(n/b) + f(n)$ ，在此 n/b 表示 $\lfloor n/b \rfloor$ 或 $\lceil n/b \rceil$ 。則依不同的情況， $T(n)$ 的大小範圍如下：

1. 若存在常數 $\varepsilon > 0$ 使得 $f(n) = O(n^{\log_b a - \varepsilon})$ ，則 $T(n) = \Theta(n^{\log_b a})$
2. 若 $f(n) = \Theta(n^{\log_b a})$ ，則 $T(n) = \Theta(n^{\log_b a} \lg n)$
3. 若存在常數 $\varepsilon > 0$ 使得 $f(n) = \Omega(n^{\log_b a + \varepsilon})$ ，且存在常數 $c < 1$ 使得對所有夠大的 n ， $af(n/b) \leq cf(n)$ ，則 $T(n) = \Theta(f(n))$ 。

範例: $T(n) = 9T(n/3) + n$, $a=9$, $b=3$, $n^{\log_b a} = n^2$

套用 case 1 , 可得 $T(n)=\Theta(n^2)$

範例: $T(n) = T(2n/3) + 1$, $a=1$, $b=3/2$, $n^{\log_b a} = n^0$

套用 case 2 , 可得 $T(n)=\Theta(\lg n)$

範例: $T(n) = 3T(n/4) + n$, $a=3$, $b=4$, $n^{\log_b a} = n^{\log_4 3}$

套用 case 3 , 可得 $T(n)=\Theta(n)$

註：這三個 case 並沒有包含所有 $f(n)$ 的可能性。
Case 1 與 2 之間有 gap, case 2 與 3 之間也有。

範例： $T(n) = 2T(n/2) + n \lg n$

在這個例子中，第二個 case 以及第三個 case 都不能被套用。

4.7 Akra-Bazzi recurrences

- $T(n) = f(n) + \sum_{i=1}^k a_i T\left(\frac{n}{b_i}\right), a_1, \dots, a_k \in R^+ , b_i > 1, \text{for } i = 1..k.$
- A function $f(n)$ satisfies the *polynomial-growth condition* if $\exists \hat{n} > 0$, such that: \forall constant $\phi \geq 1$, there exists a constant $d > 1$ such that $\frac{f(\psi n)}{d} \leq f(n) \leq df(n)$ for all $1 \leq \psi \leq \phi$ and $n \geq \hat{n}$.

Akra-Bazzi method

- Solve $T(n) = f(n) + \sum_{i=1}^k a_i T\left(\frac{n}{b_i}\right)$
- First determine the unique real number p such that $\sum_{i=1}^k a_i/b_i^p = 1$.
- The Akra-Bazzi method gives the solution:

$$T(n) = \Theta(n^p \left(1 + \int_1^n \frac{f(x)}{x^{p+1}} dx \right))$$

- Eg. $T(n) = T(n/5) + T(7n/10) + n$

Solve $T(n) = T(n/5) + T(7n/10) + n$

- $a_1 = a_2 = 1, b_1 = 5, b_2 = \frac{10}{7}, f(n) = n$
- Find p such that $\sum_{i=1}^k a_i/b_i^p = 1, p \approx 0.83978$
- By Akra-Bazzi method :

$$T(n) = \Theta(n^p \left(1 + \int_1^n \frac{f(x)}{x^{p+1}} dx \right))$$

- $\int_1^n \frac{f(x)}{x^{p+1}} dx = \int_1^n \frac{x}{x^{p+1}} dx = \frac{1}{-p+1} x^{-p+1} |_1^n = \Theta(n^{-p+1})$
- $T(n) = \Theta(n^p \cdot c n^{-p+1}) = \Theta(n)$

Exercises

Problem 1:

在德國的樂透中你必須從 1 到 49 號之中選出六個數字。玩樂透有一個很普遍的策略（雖然不會增加你贏的機會），就是選擇一個子集合 S 包含 49 個數字裡面的 k ($k > 6$) 個數字，然後只從 S 裡面選擇 6 個數字，重複玩很多次。

例如， $k=8$ 且 $S=\{1, 2, 3, 5, 8, 13, 21, 34\}$ ，總共有 28 種組合：
 $[1, 2, 3, 5, 8, 13]$, $[1, 2, 3, 5, 8, 21]$, $[1, 2, 3, 5, 8, 34]$,
 $[1, 2, 3, 5, 13, 21]$, ..., $[3, 5, 8, 13, 21, 34]$ 。

你的工作就是寫一個程式，讀入 k 以及 S ，然後印出所有可能的組合。

Exercises

輸入：有好幾組測資。每一組測資會先給一個整數 k ($6 < k < 13$)，接著之後的 k 個整數就是集合 S (數字由小排到大)。 $k=0$ 代表輸入結束。

輸出：對於每組測資，印出所有可能的組合（一行印一種組合）。每一種組合的數字要從小到大印出，每個數字之間要用一個空白隔開。所有的組合必須要根據字典順序印出，也就是先從最小的數字當作排序的依據，相等的時候再根據第二小的數字，依此類推。每一組測資的輸出要用一個空行隔開。不要在最後一組測資的輸出之後多印一個空行。

以下是一個輸出入的實例：

Sample Input	Sample Output
7 1 2 3 4 5 6 7 0	1 2 3 4 5 6 1 2 3 4 5 7 1 2 3 4 6 7 1 2 3 5 6 7 1 2 4 5 6 7 1 3 4 5 6 7 2 3 4 5 6 7

Exercises

Problem 2:

產生排列組合在資訊科學方面一直是個重要的問題。這一題要求你把一段字串的所有排列組合依照字典順序由前到後排好。請記住你的演算法必須要有效率。

輸入：第一行包含一個整數 n ，接下來的 n 行包含了 n 個字串。字串只會由英文字母或數字所構成，而且不會有空白字元在裡頭。字串最大的長度為 10。

輸出：對於每組測資，依照先後順序把所有的排列組合印出來。字母的大小視為不同，而且要注意印出來的排列組合不能重複。在每組輸出後面要多印一個空行。

以下是一個輸出入的實例：

Sample Input	Sample Output
3 ab abc bca	ab ba abc acb bac bca cab cba abc acb bac bca cab cba