Homework 1 solution

1. Please rewrite the **while** loop in the pseudocode of **Insertion sort.**

Insertion-sort(A)

 for $j \leftarrow 2$ **to** length[A]

  do key $\leftarrow A[j]$

   $i \leftarrow j - 1$

   while ___(1)___ and ___(2)___

    do ___(3)___

     $i \leftarrow i - 1$

   $A[i + 1] \leftarrow$ key

(1) $i > 0$ (2) $A[i] >$ key   (3) $A[i + 1] \leftarrow A[i]$ ((1)、(2) 可交換)

2. Please rewrite the **for** loop in the pseudocode of **Merge sort.**

Merge(A, p, q, r)

 $n_1 \leftarrow q - p + 1$

 $n_2 \leftarrow r - q$

 create array $L[1,\ldots, n_1 + 1]$ and $R[1,\ldots, n_2 + 1]$

 for $i \leftarrow 1$ **to** n_1

  do $L[i] \leftarrow A[p + i - 1]$

 for $j \leftarrow 1$ **to** n_2

  do $R[j] \leftarrow A[q + j]$

 $L[n_1 + 1] \leftarrow \infty$

 $R[n_2 + 1] \leftarrow \infty$

 $i \leftarrow 1$

 $j \leftarrow 1$

 for $k \leftarrow p$ **to** r

  do if $L[i] \leq R[j]$

   then ___(1)___

    ___(2)___

   else ___(3)___

    ___(4)___

Merge-sort(A, p, r)

 if $p < r$

  then $q \leftarrow \lfloor (p + r)/2 \rfloor$

MERGE-SORT(A, p, q)

MERGE-SORT(A, $q + 1$, r)

MERGE(A, p, q, r)

(1) $A[k] \leftarrow L[i]$ (2) $i \leftarrow i + 1$ (3) $A[k] \leftarrow R[j]$ (4) $j \leftarrow j + 1$

3. Illustrate the operation of **Merge sort** on the array $\langle 7, 4, 1, 8, 5, 2, 6, 3 \rangle$.

$\langle 7, 4, 1, 8, 5, 2, 6, 3 \rangle$

$\langle 7, 4, 1, 8 \rangle$ $\langle 5, 2, 6, 3 \rangle$

$\langle 7, 4 \rangle$ $\langle 1, 8 \rangle$ $\langle 5, 2 \rangle$ $\langle 6, 3 \rangle$

$\langle 7 \rangle$ $\langle 4 \rangle$ $\langle 1 \rangle$ $\langle 8 \rangle$ $\langle 5 \rangle$ $\langle 2 \rangle$ $\langle 6 \rangle$ $\langle 3 \rangle$

$\langle 4, 7 \rangle$ $\langle 1, 8 \rangle$ $\langle 2, 5 \rangle$ $\langle 3, 6 \rangle$

$\langle 1, 4, 7, 8 \rangle$ $\langle 2, 3, 5, 6 \rangle$

$\langle 1, 2, 3, 4, 5, 6, 7, 8 \rangle$

4. Briefly explain what **in-place sorting algorithm** is.

A sorting algorithm is in-place if the numbers are rearranged within the array A, with at most a constant number of them sorted outside the array at any time.

(沒寫到可使用常數量級之額外記憶體空間則扣 1 分)

5. Determine whether Insertion sort and Merge sort is **in-place** or not.

Insertion sort is in-place.

Merge sort is NOT in-place.

6. Briefly describe the definition of the set $\Theta(g(n))$.

$\Theta(g(n)) =$

$\{ f(n) : \exists$ positive constants c_1, c_2, n_0 s.t. $\forall\, n \geq n_0, 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$

7. Verify that $\frac{1}{1000} n^3 + 10n^2 + 100n = \Theta(n^3)$. (Need to find c_1, c_2, n_0 and verify.)

Let $c_1 = \frac{1}{1000}$, $c_2 = 3$, $n_0 = 10$,

$0 \leq c_1 g(n) = \frac{1}{1000} n^3 \leq \frac{1}{1000} n^3 + 10n^2 + 100n \leq f(n)$ for all $n > 0$ … (A)

As $n \geq n_0 = 10$, $f(n) = \frac{1}{1000}n^3 + 10n^2 + 100n \leq \frac{1}{1000}n^3 + (n)\,n^2 + (n^2)\,n = \frac{2001}{1000}n^3$

$\leq 3n^3 = c_2 g(n)$ … (B)

By (A), (B), $\frac{1}{1000}n^3 + 10n^2 + 100n = \Theta(n^3)$. ■

8. Prove that $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

($\Rightarrow$) :

$f(n) = \Theta(g(n))$ imply $\exists$ positive constants c_1, c_2, n_0 s.t. $\forall\, n \geq n_0$, $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$,

from $0 \leq c_1 g(n) \leq f(n)$, we get $f(n) = \Omega(g(n))$,

from $f(n) \leq c_2 g(n)$, we get $f(n) = O(g(n))$.

($\Leftarrow$):

$f(n) = O(g(n))$ imply $\exists$ positive constants c_1, n_1 s.t. $\forall\, n \geq n_1$, $f(n) \leq c_1 g(n)$,

$f(n) = \Omega(g(n))$ imply $\exists$ positive constants c_2, n_2 s.t. $\forall\, n \geq n_2$, $0 \leq c_2 g(n) \leq f(n)$,

therefore, let $n_0 = \max\{n_1, n_2\}$, $\exists$ positive constants c_1, c_2, n_0 s.t. $\forall\, n \geq n_0$, $0 \leq c_2 g(n) \leq f(n) \leq c_1 g(n)$, we get that $f(n) = \Theta(g(n))$. ■

9. (Multiple choice) Choose the correct statements.

(A) $(n+1)! = \Theta(n!)$

(B) $2^{n+1} = \Theta(2^n)$

(C) $(n+1)^{100} = \Theta(n^{100})$

(D) $\sqrt{n+1} = \Theta(\sqrt{n})$

(E) $\lg(n+1) = \Theta(\lg(n))$

BCDE

10. (Multiple choice) Choose the correct statements.

(A) $(2n)! = \Theta(n!)$

(B) $2^{2n} = \Theta(2^n)$

(C) $(2n)^{100} = \Theta(n^{100})$

(D) $\sqrt{2n} = \Theta(\sqrt{n})$

(E) $\lg(2n) = \Theta(\lg(n))$

CDE

11. (Multiple choice) Let $f(n)$ and $g(n)$ be positive increasing function. Choose the correct statements.

(A) $f(n) = \Theta(g(n))$ imply $(f(n))^2 = \Theta((g(n))^2)$
(B) $f(n) = \Theta(g(n))$ imply $2^{f(n)} = \Theta(2^{g(n)})$
(C) $f(n) = o(g(n))$ imply $2^{f(n)} = o(2^{g(n)})$
(D) $f(n) = \Theta(g(n))$ imply $\lg(f(n)) = \Theta(\lg(g(n)))$
(E) $f(n) = o(g(n))$ imply $\lg(f(n)) = o(\lg(g(n)))$

ACD (C 和 D 的證明在最後面)　(D)可選可不選

沒選(D)不扣分之原因:

有同學反應像 $1-\frac{1}{n}$ 和其他遞增但恆小於 2 的函數取 log 後會變成負數，根據投影片對Θ之定義 $\exists$ positive constants c_1, c_2, n_0 s.t. $\forall\, n \ge n_0$, $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$，因為這些函數恆負，確實無法找到使不等式滿足之正數 c_1, c_2, n_0，因此即使負函數並不合理，仍斟酌不扣分。

12. Derive that $\lg(n!) = \Theta(n \lg n)$.

Method 1: Using Stirling's approximation,

$\lg(n!) = \lg(\sqrt{2\pi n}\,(\frac{n}{e})^n) = \lg(\sqrt{2\pi n}) + \lg((\frac{n}{e})^n) = \Theta(\lg n) + \Theta(n \lg n) = \Theta(n \lg n)$. ■

Method 2: Finding the asymptotic upper and lower bound,

$n! = n \times (n\text{-}1) \times \cdots \times 1 \le n \times n \times \cdots \times n = n^n$ for all $n > 0$

$\Rightarrow \lg(n!) \le \lg(n^n) = n \lg n$ for all $n > 0$

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

$n! = n \times (n\text{-}1) \times \cdots \times \lfloor \frac{n}{2} \rfloor \times \cdots \times 1 \ge n \times (n\text{-}1) \times \cdots \times \lfloor \frac{n}{2} \rfloor \ge (\frac{n}{2})^{\frac{n}{2}}$ for all $n > 0$

$\Rightarrow \lg(n!) \ge \lg((\frac{n}{2})^{\frac{n}{2}}) = \frac{n}{2} \lg \frac{n}{2} = \frac{n}{2} \lg n - \frac{n}{2}$ for all $n > 0$

$\Rightarrow \lg(n!) = \Omega(\frac{n}{2} \lg n - \frac{n}{2}) = \Omega(n \lg n)$.

$\lg(n!) = O(n \lg n)$ and $\lg(n!) = \Omega(n \lg n)$ imply $\lg(n!) = \Theta(n \lg n)$. ■

13. Consider the following pseudocodes

ITERATE(n)

while $n > k$

do $n \leftarrow f(n)$

(a) What is the time complexity when $f(n) = \frac{n}{2}$ and $k = 1$

(b) What is the time complexity when $f(n) = \sqrt{n}$ and $k = 2$

(c) What is the time complexity when $f(n) = \lg(n)$ and $k = 1$

(a) $\Theta(\lg n)$ (b) $\Theta(\lg\lg n)$ (c) $\Theta(\lg^* n)$

14. Rank the following functions by order of growth from higher to lower.

$\sqrt{\lg n}$, $(\lg n)!$, n^n, $\sqrt{n}$, n^{100}, 3^n, $\lg n$, $n^{\lg n}$, n, $n\lg n$, $\sqrt[3]{n}$, 2^{2^n}, $n2^n$, $99n^{99}$, $n!$, $\lg^*(\lg n)$, $\lg(\lg^* n)$, $\lg\lg n$

$2^{2^n} > n^n > n! > 3^n > n2^n > n^{\lg n} > (\lg n)! > n^{100} > 99n^{99} > n\lg n > n > \sqrt{n} > \sqrt[3]{n} > \lg n > \sqrt{\lg n} > \lg\lg n > \lg^*(\lg n) > \lg(\lg^* n)$

(每漏寫一個或寫錯一個即扣一分)

15. Assume that $T(n)$ is constant for sufficiently small n. Show that the tight bound of $\boldsymbol{T(n) = 3T(n/3) + \Theta(n)}$ is $\Theta(n \lg n)$ by substitution method.

Upper bound: $T(n) \leq 3T(\frac{n}{3}) + cn$ for some $c > 0$

Guess: $T(n) \leq dn \lg n$ for some $d > 0$,

$T(n) \leq 3(d\,\frac{n}{3}\lg\frac{n}{3}) + cn$

$= dn \lg\frac{n}{3} + cn$

$= dn \lg n - dn \lg 3 + cn$

$= dn \lg n + n(c - d \lg 3)$

$\leq dn \lg n$ if $c - d \lg 3 \leq 0$

$\Rightarrow \exists\, d = c > 0$ s.t. $T(n) \leq dn \lg n$, therefore, $T(n) = O(n \lg n)$

Lower bound: $T(n) \geq 3T(\frac{n}{3}) + cn$ for some $c > 0$

Guess: $T(n) \geq dn \lg n$ for some $d > 0$,

$T(n) \geq 3(d\,\frac{n}{3}\lg\frac{n}{3}) + cn$

$= dn \lg\frac{n}{3} + cn$

$= dn \lg n - dn \lg 3 + cn$

$= dn \lg n + n(c - d \lg 3)$

$\geq dn \lg n$ if $c - d \lg 3 \geq 0$

$\Rightarrow \exists\, d = \frac{c}{2} > 0$ s.t. $T(n) \geq dn \lg n$, therefore, $T(n) = \Omega(n \lg n)$

$T(n) = O(n \lg n)$ and $T(n) = \Omega(n \lg n)$ imply $T(n) = \Theta(n \lg n)$. ■

16. Assume that $T(n)$ is constant for sufficiently small n. Show that the upper bound of $\boldsymbol{T(n) = T(n\text{-}1) + \Theta(n \lg n)}$ is $O(n^2\lg n)$ by substitution method.

$T(n) \leq T(n\text{-}1) + cn \lg n$ for some $c > 0$

Guess: $T(n) \leq dn^2\lg n$ for some $d > 0$

$T(n) \leq d(n\text{-}1)^2\lg(n\text{-}1) + cn \lg n$

$\leq d(n\text{-}1)^2\lg n + cn \lg n$

$= dn^2\lg n - 2dn \lg n + d \lg n + cn \lg n$

$\leq dn^2\lg n - dn \lg n + cn \lg n$

$\leq dn^2\lg n - (d\text{-c})\, n \lg n$

$\leq dn^2\lg n$ if $d - \mathrm{c} > 0$

$\Rightarrow \exists\, d = 2c > 0$ s.t. $T(n) \leq dn^2\lg n$,

therefore, $T(n) = O(n^2\lg n)$. ■

17. Assume that $T(n)$ is constant for sufficiently small n. Show that the lower bound of $\boldsymbol{T(n) = T(n\text{-}1) + \Theta(n \lg n)}$ is $\Omega(n^2\lg n)$ by substitution method. Conclude that the tight bound is $\Theta(n^2\lg n)$. (Hint: $n \lg\frac{n}{n-1} \leq 2$ for $n \geq 2$)

$T(n) \geq T(n\text{-}1) + cn \lg n$ for some $c > 0$

Guess: $T(n) \geq dn^2\lg n$ for some $d > 0$

$T(n) \geq d(n\text{-}1)^2\lg(n\text{-}1) + cn \lg n$

$= dn^2\lg(n\text{-}1) - 2dn \lg(n\text{-}1) + d \lg(n\text{-}1) + cn \lg n$

$\geq dn^2\lg(n\text{-}1) - 2dn \lg(n\text{-}1) + cn \lg n$

$\geq dn^2\lg(n\text{-}1) - 2dn \lg n + cn \lg n$

$= dn^2\lg(n\text{-}1) + (c\text{-}2d)n \lg n$

$\geq dn^2\lg(n\text{-}1) + (c\text{-}2d)n \quad (\text{if } d \leq \frac{c}{2})$

$\geq dn^2\lg n \qquad \text{if } (c\text{-}2d)n \geq dn^2\lg n - dn^2\lg(n\text{-}1) = dn^2\lg\frac{n}{n-1}$

$c\text{-}2d \geq dn\lg\frac{n}{n-1}$

Since $n\lg\frac{n}{n-1} \leq 2$ for $n \geq 2$, $\exists\, d = \frac{c}{4} > 0$ s.t. $T(n) \geq dn^2\lg n$,

therefore, $T(n) = \Omega(n^2\lg n)$. ■

(with $T(n) = O(n^2\lg n)$, we conclude that $T(n) = \Theta(n^2\lg n)$)

15. ~ 17.扣分標準：

有些許計算錯誤或小細節不夠嚴謹，但整體觀念正確扣 1 分

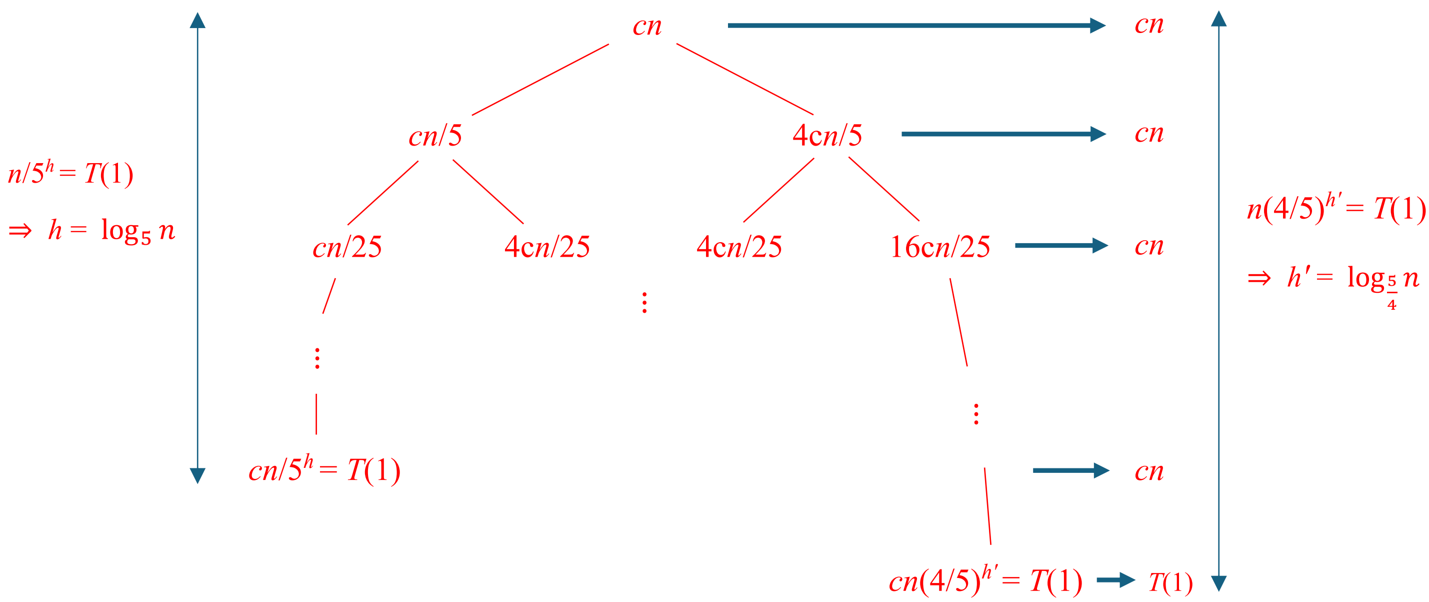
過程有些許問題扣 2 分

過程有較大問題扣 3 分

未使用 substitution method 或連初始條件都寫錯扣 4 分

幾乎完全沒寫或內容毫無相關扣 5 分

18. Assume that $T(n)$ is constant for sufficiently small n. Use a recursion tree to determine the tight bound on the recurrence $\boldsymbol{T(n) = T(n/5) + T(4n/5) + \Theta(n)}$.



$T(n) \geq h(cn) = cn\log_5 n = (c\lg 5)\, n\lg n \;\Rightarrow T(n) = \Omega(n\lg n)$.

$T(n) \leq h'(cn) = cn\log_{\frac{5}{4}} n = (c\lg\frac{5}{4})\, n\lg n \;\Rightarrow T(n) = O(n\lg n)$.

$T(n) = \Omega(n \lg n)$ and $T(n) = O(n \lg n)$ imply $T(n) = \Theta(n \lg n)$. ■

19. Assume that $T(n)$ is constant for sufficiently small n. Use the master method to determine the tight bound on the recurrence $\boldsymbol{T(n) = 4T(n/2) + \Theta(n^2 \lg n)}$.

$a = 4$, $b = 2$, $\log_b a = 2$, $f(n) = \Theta(n^2 \lg n)$

$\Rightarrow$ satisfy the condition of case 2 with $k = 1$

$\Rightarrow$ by master theorem, $T(n) = \Theta(n^2 \lg^{k+1} n) = \Theta(n^2 \lg^2 n)$ ■

若答案是錯的:

有寫出 $k = 1$ 得 4 分，若沒有則

有寫出此題為 Case 2 得 3 分，若仍沒有則

有比較 $n^2 \lg n$ 和 $n^{\log_b a}$ 得 2 分，若還是沒有則 0 分

20. (Multiple choice) Assume that $T(n)$ is constant for sufficiently small n. Choose the correct statements.

(A) If $T(n) = 3T(n/3) + n \lg n$, then the tight bound is $T(n) = \Theta(n \lg n)$

(B) If $T(n) = 9T(n/3) + n \lg n$, then the tight bound is $T(n) = \Theta(n \lg n)$

(C) If $T(n) = 3T(n/9) + n \lg n$, then the tight bound is $T(n) = \Theta(n \lg n)$

(D) If $T(n) = 3T(n/3) + n^2 \lg n$, then the tight bound is $T(n) = \Theta(n^2 \lg^2 n)$

(E) If $T(n) = 9T(n/3) + n^2 \lg n$, then the tight bound is $T(n) = \Theta(n^2 \lg^2 n)$

CE

※Proof of 11 (C) and (D)

(C) :

If $g(n)$ is bounded, then $f(n) = 0$ or $\lim_{n \to \infty} f(n) = 0$, contradict to $f(n)$, $g(n)$ are positive increasing function, therefore $g(n)$ is unbounded.

Given $c > 0$, $f(n) = o(g(n))$ gives that $\exists\, n_1 > 0$ s.t. for all $n > n_1$, $0 < f(n) < \frac{g(n)}{2}$.

Since $g(n)$ is unbounded $\exists\, n_2 > 0$ s.t. for all $n > n_2$, $g(n) > 2|\lg c| \Rightarrow g(n) + \lg c > \frac{g(n)}{2}$.

For all $n > n_0 = \max\{n_1, n_2\}$, $0 < 2^{f(n)} < 2^{\frac{g(n)}{2}} < 2^{g(n) + \lg c} = c2^{g(n)}$.

Therefore, $2^{f(n)} = o(2^{g(n)})$. ■

(D) :

According to (C), ω notation has similar property since

$f(n) = \omega(g(n)) \Rightarrow g(n) = o(f(n)) \Rightarrow 2^{g(n)} = o(2^{f(n)}) \Rightarrow 2^{f(n)} = \omega(2^{g(n)})$.

We can conclude that $f(n) \neq \Theta(g(n))$ imply $2^{f(n)} \neq \Theta(2^{g(n)})$.

Thus, $2^{f(n)} = \Theta(2^{g(n)})$ imply $f(n) = \Theta(g(n))$, which is equivalent to

$$f(n) = \Theta(g(n)) \text{ imply } \lg(f(n)) = \Theta(\lg(g(n)))$$

if $\lg(f(n))$ and $\lg(g(n))$ are positive increasing function. ■