Homework 2 林業誠 E24105038 資訊114甲

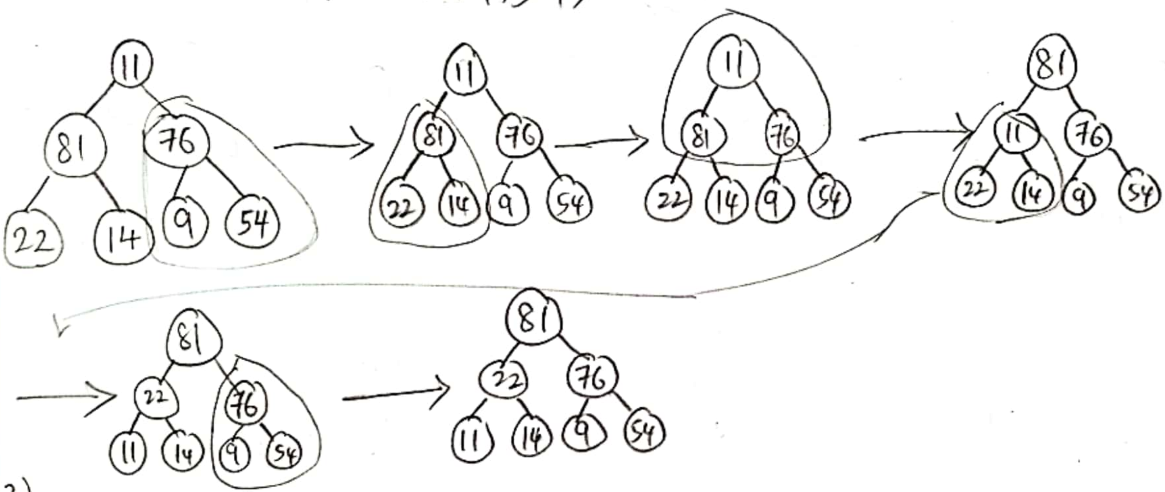
1) Assuming $\boxed{\text{leaf node / root node}}$ is at $\boxed{\text{height } 0}$, the minimum number of elements in a heap is $\boxed{2^h}$ ①, and the maximum number of elements in the heap is $\boxed{2^{(h+1)}-1}$ ②.

$h=2$, $\# = 4$ $\therefore 2^h$

↳ For the reason of ①, there is only one leaf node and since heap is a binary tree, so the answer is 2^h.

↳ For ②, since it is a full binary tree, so the number of elements is $2^{(h+1)}-1$.

2) $\langle 11, 81, 76, 22, 14, 9, 54 \rangle$



3)

(1) $T(n) = T(\frac{9n}{10}) + n$

$\Rightarrow n^{\log_{\frac{10}{9}} 1}$ vs n

↳ Since $\log_{\frac{10}{9}} 1 = 0$, $0 + \varepsilon = 1$

$\Rightarrow a f(\frac{n}{5}) = \frac{9}{10} n \leq cn$, for $c = \frac{9}{10} < 1$

$\therefore$ Using Case 3, $\boxed{T(n) = \Theta(n)}$ #

(2) $T(n) = 4T(\frac{n}{2}) + n^2$

$\Rightarrow n^{\log_2 4}$ vs $n^2 = n^2$

↳ Since $n^{\log_2 4} = n^2$

$\therefore$ Using Case 2 with $k=0$,

$\boxed{T(n) = \Theta(n^2 \lg n)}$

#

4) Guess $F(n) \leq d \cdot 2^n$

$\hookrightarrow F(n) = F(n-1) + F(n-2)$

$\leq 2F(n-1)$

$= 2 \cdot (d \cdot 2^{n-1}) = d \cdot 2^n$.

$\therefore F(n) \leq d \cdot 2^n$, if $d > 0$

Therefore, $\boxed{T(n) = O(2^n)}$ #

(upper)

Guess $F(n) \geq 2^{\frac{n}{2}}$

$\hookrightarrow F(n) = F(n-1) + F(n-2)$

$\geq 2F(n-2)$

$= 2(d \cdot 2^{\frac{n-2}{2}}) = d \cdot 2^{\frac{n}{2}}$

$\therefore F(n) \geq d \cdot 2^{\frac{n}{2}}$, if $d > 0$

Therefore, $\boxed{T(n) = \Omega(2^{\frac{n}{2}})}$ #

(lower)

5) - The worst case will occur when the subarrays are completely unbalanced, having 0 elements in one subarray and n-1 elements in the other subarray. The time complexity is $O(n^2)$.

- The best case will occur when the subarrays are completely balanced every time that each subarray has $\leq \frac{n}{2}$ elements. The time complexity is $O(n\log_2 n)$.

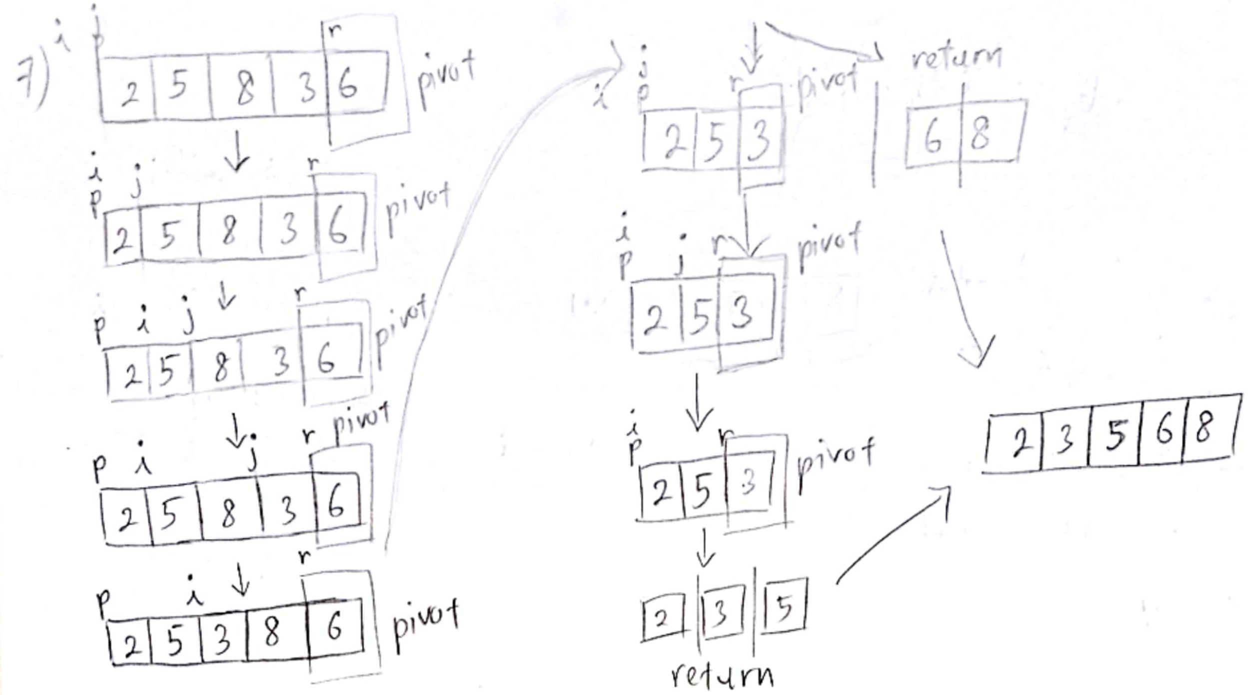
6) A) $O(n)$
B) $O(n^2)$
C) $O(n\lg n)$
D) $O(n\lg n)$
E) $O(n\lg n)$
F) $O(n^2)$

$\Rightarrow$ The Heap Sort Best and Worst Case are the same time complexity as the Quick Sort Best Case.

$\Rightarrow$ The Insertion Sort Worst Case is same as the Quick Sort Worst Case

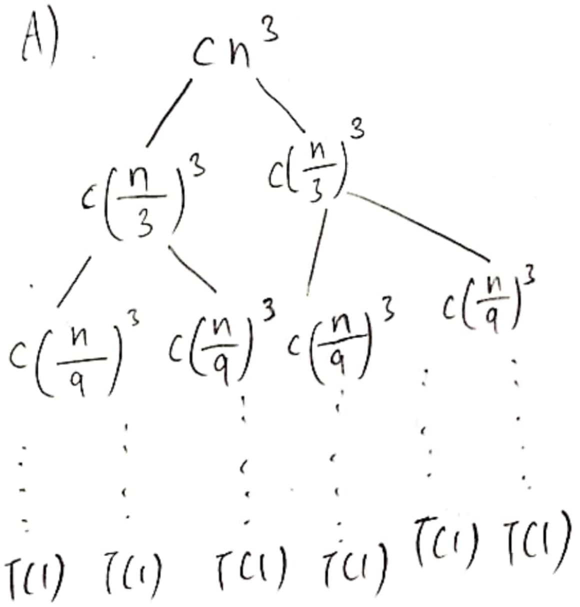
$\Rightarrow$ Overall, the lowest time complexity is the Insertion Sort Best Case and then followed by the Heap Sort Best / Worse Case and Quick Sort Best Case. Lastly, the highest time complexity is the Insertion Sort Worst Case and Quick Sort Worst Case.

7)



8) $T(n)=2T(\frac{n}{3})+\Theta(n^3)$

A)



B) level-1 : cn^3

level-2 : $c(\frac{2}{27})n^3$

level-3 : $c(\frac{2}{27})^2 n^3$

last level : $\Theta(n^{\log_3 2})$

$\Rightarrow \frac{n}{3^k}=1 \Leftrightarrow n=3^k \rightarrow k=\log_3 n$

$\Rightarrow n^{\log_3 2}$

C) $T(n)=cn^3+\frac{2}{27}cn^3+(\frac{2}{27})^2 cn^3+\ldots+(\frac{2}{27})^{\log_3(n-1)}cn^3+\Theta(n^{\log_3 2})$

$$=\sum_{i=0}^{\log_3 n-1}(\frac{2}{27})^i cn^3+\Theta(n^{\log_3 2})$$

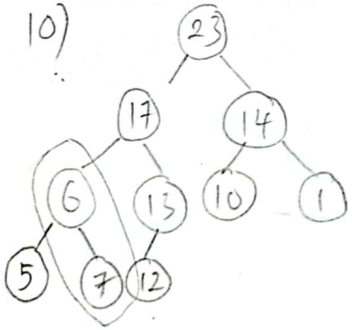
$$<\sum_{i=0}^{\infty}(\frac{2}{27})^i cn^3+\Theta(n^{\log_3 2})$$

$$=\frac{1}{1-\frac{2}{27}}cn^3+\Theta(n^{\log_3 2})=\frac{27}{25}cn^3+\Theta(n^{\log_3 2})$$

$$\boxed{\therefore T(n)=O(n^3)\ \#}$$

9) a, b

10)



↳ It is not a max-heap, since the right-child of 6 is 7, and the rules of max-heap is the number of the parent must be larger than the number of both left-child and right child.