Assignment 1

Problem 1

$O(8n!) = O(n!)$

$O(\log 9n) = O(\log n)$

$O(2n\log n + 7n^2) = O(n^2)$

$O(4n + 6\log n) = O(n)$

$O(5n^3 + 6n^2) = O(n^3)$

$O(0.00001 \times n^n) = O(n^n)$

$\therefore O(\log 9n) < O(4n + 6\log n) < O(2n\log n + 7n^2)$
$< O(5n^3 + 6n^2) < O(8n!) < O(0.00001 \times n^n)$

Problem 2

(A) $O(4n + 6\log n) = O(n)$

(B) $O(30n + 2n\log(9n) + 800) = O(n\log n)$

(C) $O(1n + 2n + 3n + 4n + 5n + 6n) = O(n)$

(D) $O(\sum_{i=1}^{n} 3n) = O(n^2)$

(E) $O(\sum_{i=1}^{n} (i \times n)) = O(n^2)$

Problem 3

(A) The first 'for' loops N times. the second one loops $[\log_2 N]$ times, '$[x]$' is the largest integer no more than x.

$$T(N) = N + \log_2 N$$

$\because$ $O(N) > O(\log N)$

$\therefore$ (A) Time Complexity is $O(N)$

(B) The outer 'for' loops for $[\log_2 N]$ times. nested cycle loops $N-1$ times. 'sum+=1' loops $(N-1) \times [\log_2 N]$ times.

$$T(N) = (N-1) \cdot \log_2 (N-1)$$

$\therefore$ (B) Time Complexity is $O(N \log_2 N)$

(C) 'return 1+function3(N-1)' calls the function itself.

$$T(N) = N$$

$\therefore$ (C) Time Complexity is $O(N)$

(D) The inner 'for' has a time complexity of $O(N)$.
'return' part loops inner 'for' N times.

$$T(N) = N + N-1 + N-2 + \cdots + 1 = \frac{1}{2}(1+N)\cdot N$$

$\therefore$ (D) Time Complexity is $O(N^2)$

(E) $T(N) = N + T(\frac{N}{2})$

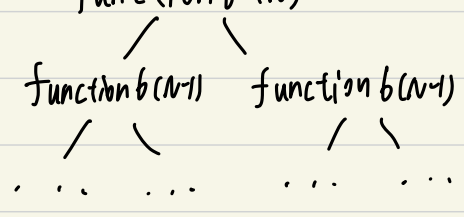
$$T(N) = N + \frac{N}{2} + \frac{N}{4} + \cdots + \frac{N}{2^k} + 1 \quad 0 < \frac{N}{2^k} \leq 1$$

$$= 2N - \frac{N}{2^{k+1}} + 1 \geq 2N$$

$$T(N) < 2N$$

$\therefore$ (E) Time Complexity : $O(N)$

(F) function f (N)



$$T(N) = 2 \cdot T(N-1) = 2^2 \cdot T(N-2) = \cdots = 2^N \cdot T(0)$$

$\therefore$ (F) Time Complexity : $O(2^N)$

(G) 'for' TC: $O(N)$

$$T(N) = N + 2 \cdot T(\tfrac{N}{2}) = N + 2 \cdot (\tfrac{N}{2} + 2 \cdot T(\tfrac{N}{4}))$$

$$= \cdots = k \cdot N + 2^k \cdot T(\tfrac{N}{2^k}), \quad 0 < \tfrac{N}{2^k} \leq 1, \quad k \geq \log_2 N$$

$$\geqslant N \cdot \log_2 N + N$$

$\therefore$ (G) Time Complexity: $O(N \cdot \log_2 N)$

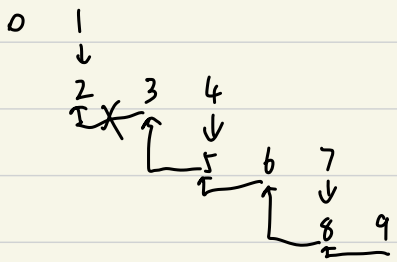
Problem 4

1. When adding or deleting an element into an array, it will move all of the elements after it forward or afterward. So it costs much. However when it comes to 'Linked List', we simply modify the node that the former one linked to.

2. When we want to locate an element in an array or linked list, it takes time complexity of $O(1)$ for an array but $O(N)$ for a linked list. It's the same situation when merging two arrays or two linked lists.

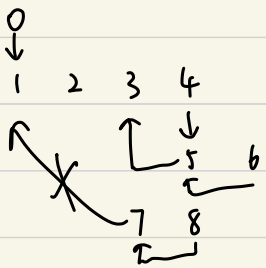
Problem J

(A)



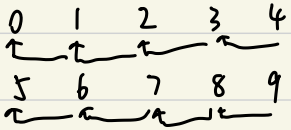
X

(B)



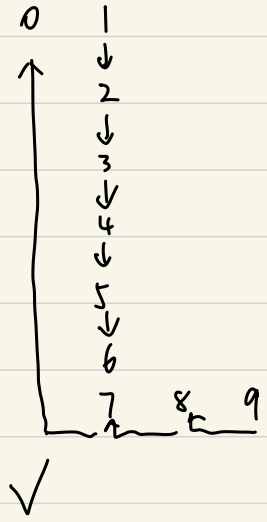
X

(C)

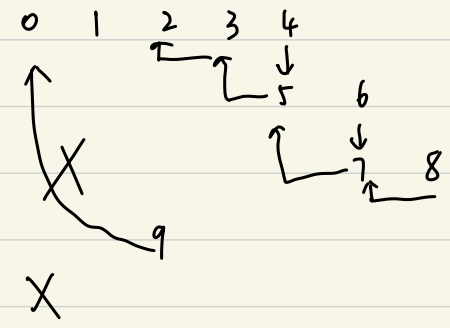


√

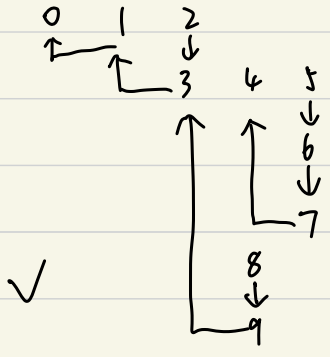
(D)



(E)



(F)



Problem 6

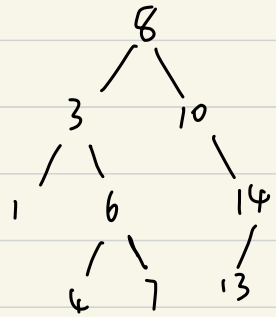
(A) X

(B) ✓

(C) X

(D) X

Problem 7

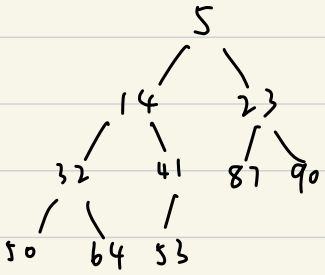


Preorder

Print Order:

8 - 3 - 1 - 6 - 4 - 7 - 10 - 14 - 13

Problem 8



In order

Print Order:

50 - 32 - 64 - 14 - 53 - 41 - 5 - 87 - 23 - 90