《注意》

1. 依照期中考公告，考試範圍為投影片第一章到第五章 Page 16，所以超出範圍的題目即為 Page 16 之後的內容，但老師實際上課進度為 Page 20。
2. 灰色字體為不確定之答案。
3. 感謝 微積分小天才之卷哥之系湖男神 陳識宇 修正第二題答案

1. Explain the following or terms comparisons:

(a) Full binary trees

< 英文版 >

A full binary tree of depth k is a binary tree of depth k having $2^{k}-1$ nodes, $k \geq 0$.

< 中文版 >

深度為 k 的二元樹中有$2^{k}-1$個節點。

(b) Complete binary trees

< 英文原版 >

A binary tree with n nodes and depth k is complete iff its nodes correspond to the nodes numbered from 1 to n in the binary tree of depth k.

< 中文原版 >

深度為 k 且有 n 個節點的二元樹，若且唯若每一個節點都與深度為 k 的 Full binary tree 中，序號 1 到 n 的節點相對應時，稱之為 Complete binary tree。

< 簡易英文版 @ Wikipedia >

In a complete binary tree every level, except possibly the last, is completely filled, and all nodes are as far left as possible.

< 簡易中文版 @ Wikipedia >

各層節點全滿，除了最後一層，最後一層節點全部靠左。

(c) Binary search trees (超出範圍)

< 英文版 >

A binary search tree is a binary tree. If it is not empty it satisfies the following properties:

- Every element has a key, and no two elements have the same key, i.e., the keys are unique.
- The keys in a nonempty left subtree must be smaller than the key in the root of the subtree.
- The keys in a nonempty right subtree must be larger than the key in the root of the subtree.
- The left and right subtrees are also binary search trees.

< 中文版 >

一棵二元樹，若不為空則有以下性質：

- 每個元素都有一個 Key 且不存在兩個元素有相同的 Key
- 左子樹裡所有的 Key 皆小於當前節點內儲存的 Key
- 右子樹裡所有的 Key 皆大於當前節點內儲存的 Key
- 左右子樹都是 Binary search tree

(d) FIFO lists v.s. LIFO lists

FIFO: First In First Out　　EX: Queue

LIFO: Last In First Out　　EX: Stack

(e) Doubly Linked List

< 英文版 >

A node in a doubly linked list has at least three fields.

- A left link field
- A data field
- A right link field

< 中文版 >

雙向鏈結串列至少有以下三種欄位：

- 左鏈結欄位
- 資料欄位
- 右鏈結欄位

(f) Max(Min) trees v.s. Max(Min) heaps (超出範圍)

< 英文版 >

Max(Min) trees:

A max (min) tree is a tree in which the key value in each node is no smaller (larger) than the key values in its children (if any).

Max(Min) heaps:

A max (min) heap is a complete binary tree that is also a max (min) tree.

< 中文版 >

Max(Min) trees:

每個節點的 Key 都不比子節點小(大)的二元樹

Max(Min) heaps:

Max(Min) trees 滿足 Complete binary tree 之條件

(g) AVL Trees (超出範圍，講義未出現)

< 中文版 >

AVL tree 是一種自平衡二元搜尋樹，特性是所有節點的左子樹與右子樹高度差不超過 1，在搜尋、插入以及刪除時可以加快速度。

(h) Performance analysis v.s. Performance measurement

< 英文版 >

Performance analysis:

Obtaining estimates of time and space that are machine-independent.

- Space complexity: Amount of memory that it needs to run to completion.
- Time complexity: The time taken by a program is the sum of its compile time and its run/execution time.

Performance measurement:

Obtaining machine-dependent times.

< 中文版 >

Performance analysis:

在不考慮操作環境下，評估執行時間與使用空間之情況，主要在討論複雜度理論：

- 空間複雜度：完成程式所需的記憶體大小
- 時間複雜度：完成程式所需要的時間

Performance measurement:

在考慮操作環境下，評估執行時間與使用空間之情況。

(i) Tree traversal (超出範圍)

< 英文版 >

Visiting each node in the tree exactly once.

Notations:

- L -- Moving left
- V – Visiting the node
- L -- Moving right

Three possible traversals if we traverse left before right:

- LVR (inorder)
- LRV (postorder)
- VLR (preorder)

< 中文版 >

拜訪樹中每一個 Node 一次而已，且若假設 L、R、V 分別代表「往左走」、「往右走」、「拜訪節點」，加上傳統先左後右的走法，就會有三種走法：

- LVR (中序)
- LRV (後序)
- VLR (前序)

(j) Activation records

< 英文版 >

Each time when a subprogram is invoked, the invoking subprogram creates an AR and places it on top of the system stack.

< 中文版 >

函式被呼叫的時候,程式會建立一個 Activation Record (活動紀錄) 或是 Stack Frame(堆疊框) 的結構，並把它放在系統堆疊的頂端。

(k) Indirect recursion v.s. Direct recursion

< 英文版 >

Indirect recursion: Functions may call other functions that invoke the calling function again.

Direct recursion: Functions call themselves.

< 中文版 >

Indirect recursion：利用其他函式來呼叫自己

Direct recursion：直接呼叫自己本身

(l) Underflow(Overflow)

< 英文版 >

A condition in a computer program where the result of a calculation is a smaller(larger) number than the computer can actually store in memory.

< 中文版 >

計算的結果小於(大於)電腦能儲存在記憶體的範圍

(m) The degree of a tree node

< 英文版 >

The number of subtrees of the node

< 中文版 >

節點所連結的子樹個數

(n) The degree of a tree

< 英文版 >

The maximum degree of the nodes in the tree

< 中文版 >

樹中節點度數的最大值

(o) Row major order

< 英文版 >

Storing multidimensional arrays by rows.

< 中文版 >

先走 row 再換 column (由左至右，由上至下)

(p) Algorithms v.s. Programs

< 英文版 >

An algorithm is a finite set of instructions that, if followed, accomplishes a particular task and must satisfy the following criteria:

- Input
- Output
- Definiteness
- Finiteness
- Effectiveness

A program does not have to satisfy finiteness condition.

< 中文版 >

演算法指利用有限的指令集合來完成指定的工作並且滿足以下條件：

- 輸入
- 輸出
- 明確定義
- 有限次
- 有效率

Program 並不一定滿足「有限次」的條件。

2. Prove or disprove the following statements:

(a) $\sum_{i=0}^{n} i^3 = \Theta(n^4)$

$$\sum_{i=0}^{n} i^3 = \frac{n^2(n+1)^2}{4} = \frac{n^4}{4} + \frac{n^3}{2} + \frac{n^2}{4}$$

若此為正確結果，則存在一正實數c_1、c_2及n_0使得$c_1 n^4 \leq \frac{n^4}{4} + \frac{n^3}{2} + \frac{n^2}{4} \leq c_2 n^4, n \geq n_0$

將兩邊同除以n^4可得$c_1 \leq \frac{1}{4} + \frac{1}{2n} + \frac{1}{4n^2} \leq c_2$

當$c_1 = \frac{1}{4}$時$c_1 \leq \frac{1}{4} + \frac{1}{2n} + \frac{1}{4n^2}$恆成立

當$c_2 = 1$時$\frac{1}{4} + \frac{1}{2n} + \frac{1}{4n^2} \leq c_2$恆成立

故$\sum_{i=0}^{n} i^3 = \Theta(n^4)$為真

(b) $n^{1.001} + nlogn = \Theta(n^{1.001})$

若此為正確結果，則存在一正實數c_1、c_2及n_0使得$c_1 n^{1.001} \leq n^{1.001} + nlogn \leq c_2 n^{1.001}, n \geq n_0$
當$c_1 = 1$時，顯而易見地，$c_1 n^{1.001} \leq n^{1.001} + nlogn$恆成立
當$n \geq 10^{6000}$ 時，$logn \leq n^{0.001}$恆成立
$nlogn \leq n^{1.001} \rightarrow n^{1.001} + nlogn \leq 2n^{1.001} \rightarrow C_2 = 2$
故$n^{1.001} + nlogn = \Theta(n^{1.001})$為真

(c) $\frac{n^2}{logn} = \Theta(n^2)$

若此為正確結果，則存在一正實數c_1、c_2及n_0使得$c_1 n^2 \leq \frac{n^2}{logn} \leq c_2 n^2, n \geq n_0$

將兩邊同除以n^2可得$c_1 \leq \frac{1}{logn} \leq c_2$

又已知$\lim_{n\to\infty} \frac{1}{logn} = 0$，不存在一正實數$c_1$使得$c_1 \leq \frac{1}{logn}$

故$\frac{n^2}{logn} = \Theta(n^2)$是錯誤的

(d) $n! = O(n^n)$

若此為正確結果，則存在一正實數c 及n_0使得$n! \leq cn^n, n \geq n_0$
顯而易見的，$n! = n(n-1)(n-2)(n-3) \times \ldots \times 2 \times 1 < n^n$
因此當$c = 1$時$n! \leq cn^n$恆成立
故 $n! = O(n^n)$ 為真

3. Assume there is an array:

   int A[5][6][10];

   Please calculate the following memory address.

   (a) If A[0][0][0] is stored at address 2000, calculate the address of A[2][3][7].

   2000 + (2 * 6 * 10 + 3 * 10 + 7) * 4 = 2628

   (b) If A[0][0][0] is stored at address 2000, indicate which array element is at the location 2300.

   2300 – 2000 = 300

   300 / 4 = 75

   75 / 60 = 1 … 15

   15 / 10 = 1 … 5

   Ans: A[1][1][5]

   (c) If A[3][0][0] is stored at address 2000, calculate the address of A[1][5][9].

   2000 – (3 * 60) * 4 = 1280

   1280 + (1 * 60 + 5 * 10 + 9) * 4 = 1756

4. For any non-empty binary tree, T, if n_0 is the number of leaf nodes and n_2 is the number of nodes of degree 2, prove that $n_0 = n_2 + 1$.

   令 n 為所有節點數, n_0 為葉子節點數

   n_1 為度數是 1 個節點數, n_2 為度數是 2 個節點數

   B 為所有分枝數

   則 $n = n_0 + n_1 + n_2$

   n = B + 1

   B = 0 * n_0 + 1 * n_1 + 2 * n_2

   $n = n_1 + 2n_2 + 1 = n_0 + n_1 + n_2$

   $n_0 = n_2 + 1$

5. Sparse matrices

(a) How to represent sparse matrices? Answer must include node structure and pseudo code to read matrix and setup its linked list representation.

< Node structure >

header node：

<table>
<tr><td colspan="2">next</td></tr>
<tr><td>down</td><td>right</td></tr>
</table>

element node：

<table>
<tr><td>row</td><td>col</td><td>value</td></tr>
<tr><td>down</td><td colspan="2">right</td></tr>
</table>

< Pseudo code >

Assume that there is an $rows \times cols$ matrix with k non-zero entries.
Implement sparse matrices using the above node structure.

Reference textbook to know detailed implementation.

```
function mread()
{
    MAX = 取 rows 跟 cols 的最大值
    替 hdnode[MAX] 分配記憶體

    for ( 讀取 k 個非零元素 )
    {
        若換到下一個 row，則將 row 的末端節點與 hdnode[row]連結
        替輸入的元素分配記憶體
        將輸入的元素與同一個 row 的末端元素相連
        將輸入的元素與同一個 col 的末端元素相連
    }

    將 row 的末端節點與 hdnode[row]連結
    將所有 cols 的末端節點與 hdnode[cols]連接起來
    將 hdnode[0 ~ MAX-1]連接起來
}
```

(b) Assume that there is an $m \times n$ matrix with k non-zero entries. Determine time complexity of your pseudo code.

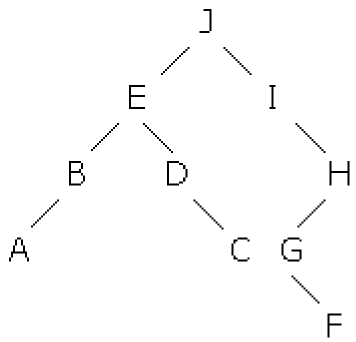
一開始必須建立 hdnode[max{m, n}]，所以這一部分是 O(max{m, n})。

再來讀取 k 個非零元素，在與行的頭/列的頭連接之時只需要單位時間即可完成，故此處為 O(k)。

最後，將所有 n 的末端節點與 *hdnode[n]*連接，和將 *hdnode[0* ~ max{m, n}*-1]*連接起來，這樣複雜度為 O(n + max{m, n})。

最糟時間複雜度為 O(max{m, n} + n + k)

6. Given an inorder sequence ABEDCJIGFH and a postorder sequence ABCDEFGHIJ, can you derive a unique binary tree? If yes, draw a binary tree; or you have to give two distinct binary trees which can generate above sequences. (超出範圍)



7. Write the postfix form of the following expressions:
   (a) A – B * D + E / F + A * D + C
   ABD*-EF/+AD*+C+
   (b) (A – B) * D + E / (F + A * D) + C
   AB-D*EFAD*+/+C+

8. Derive the worst case time complexity of the binary search function.
   搜尋花最久的情況 → 在樹最底下→ 樹的高度
   故時間複雜度為 $O(h) = O(log_2 n)$

9. Tree operations (超出範圍)
   (a) Describe how to delete an element from a binary search tree. Calculate the time complexity of the deletion operation.
   SKIP

   (b) Describe how to insert an element into a min heap. Calculate the time complexity of the insertion operation.
   SKIP

10. Linked list operations

(a) Explain how to implement a circular queue by using an array.

假設共有 n 個空間，為了辨別是否 Full 或 Empty，必須留一個空位

- 初始

  front = n – 1

  rear = n – 1

- 加入

  rear = (rear + 1) % n

- 刪除

  front = (front + 1) % n

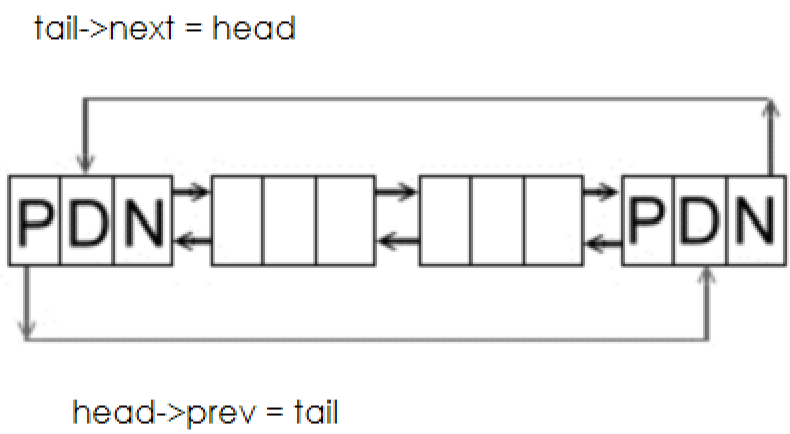
- 是否為 Full

  front == (rear + 1) % n

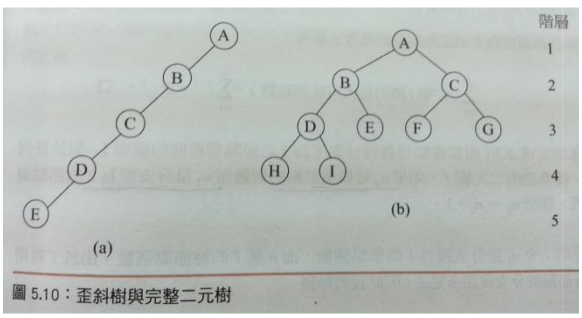
- 是否為 Empty

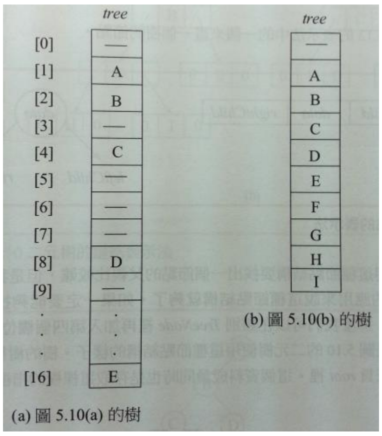
  front == rear

(b) Explain how to implement a doubly linked circular list.



(c) Explain how to implement a binary tree representation of an array. Explain pros and cons.





缺點：若遇到歪斜樹則將浪費記憶體空間。

優點：可快速推算出左兒子、右兒子以及爸爸 ID

(d) Give two applications of stacks.

- 運算式的轉換
- 副程式的呼叫和返回
- 河內塔問題
- 八皇后問題

11. Answer “True” or “False” for the following statements.

(a) An empty binary tree is invalid while a tree may have zero nodes.

否

二元樹若為空也是合法的，但樹則至少要有一個節點。

(b) The order of child is irrelevant in a binary tree.

否

對於二元樹而言，左子樹與右子樹是不相同的，但對於樹而言是一樣的。

(c) The order of operations in infix representation is the same as that in postfix representation.

是

依照 Wikipedia 表示，Order of operations 別名為 Operator precedence，也就是是運算子優先權。而不管是前序或後序均先乘除後加減。

(d) Compared a binary search tree with a heap, the former is more suited for deleting arbitrary elements. (超出範圍)

SKIP

(e) The time complexity of a deletion operation from a n-element max heap is O(n) (超出範圍)

SKIP

12. During the process of transforming a parenthesized infix expression to a postfix one, why do we need two types of precedence, an in-stack precedence and an incoming precedence?

左括號在整個運算時變的比較複雜，只要它在運算式中被發現，它就會立刻被放到堆疊中。但它只有與它匹配的右括號被找到時，才會從堆疊中被移除。因此必須為所有運算子設定兩種優先權，也就是 In-stack precedence 和 Incoming precedence。

13. Solving the equivalence classes problem is an application of binary search trees. Explain how to process an equivalence pair, i≡j (超出範圍)

SKIP

14. System Stack

(a) AR field

Local variables
Parameters
Previous AR pointer
Return address

(b) AR lifetime

A 要呼叫 B 時，系統將 A 的區域變數存在 AR，然後替 B 建立一個新的 AR，再把控制權交給 B。B 執行完後，透過 AR pointer 依照 Return address 返回至 A 要繼續執行的地方，最後再把 B 的 AR 歸還記憶體給系統。

15. Equivalence class

(a) What is an equivalence determination problem?

分類問題，OVER。

(b) Pseudo code

```
void equivalence()
{
    將 seq 指向 NULL 並將 out 設成 TRUE
    while(還有輸入等價序對)
    {
        將序對 <i, j> 讀入
        將 j 放入 seq[j]的串列中
        將 i 放入 seq[i]的串列中
    }
    for (I = 0; I < n; ++i)
    {
        if (out[i])
        {
            out[i] = FALSE;
            將 i 的等價類別印出
        }
    }
}
```

(c) Time complexity

總共有 n 個數字，m 個序對，在第一部分初始化時的複雜度為 O(m + n)

而第二部分開始，每一個節點頂多放入堆疊一次。由於只有 2m 個節點，而 for 迴圈會執行 n 次，所以這個部分的時間複雜度為 O(m + n)。

故整體為 O(m + n)。