編號： 181

國立成功大學 104 學年度碩士班招生考試試題

系所組別：電機工程學系丁組

考試科目：資料結構

考試日期：0211，節次：2

※ **考生請注意：**

(1) 本試題不可使用計算機。 請於答案卷作答，於本試題紙上作答者，不予計分。

(2) 本試題共計 4 頁。作答時可不必抄題，但請務必於答案卷將各作答題之題號標註清楚。

(3) Note that, throughout this paper, the height of a tree with only one node is defined as 1 while an empty tree has height of zero.

1. 是非題 [A] (10 分；下列 (1-1) ~ (1-5) 各題敘述若正確請以 **T** 或 O 表示，若錯誤請以 **F** 或 × 表示；每題 2 分，答對得 2 分，答錯或未作答得 0 分)
For each statement in (1-1) ~ (1-5), please indicate **T** or O if it is correct and indicate **F** or × otherwise. (2 points each)

(1-1) 使用二分搜尋(Binary search)法搜尋資料結果失敗之時間複雜度(Time complexity)為 $O(n)$。

(1-2) 若|V|表示樹狀(Tree)結構 T 中節點(Node)數量且|E|表示 T 之樹枝(Branch)數量，則 |V| > |E|。

(1-3) 任一圖形(Graph)資料結構 G 中必存在一頂點 x，其 ***DFS*** 與 ***BFS*** 產生相同之輸出結果，即 $DFS(x) = BFS(x)$。([註] DFS: Depth First Search; BFS: Breadth First Search)

(1-4) AVL 樹(AVL tree) T 之內部節點(Internal node)其平衡因子(Balance factor)之值皆為 1。

(1-5) 陣列(Array)儲存於記憶體中之配置方式是將各組成元素依其值大小順序連續存放於記憶體。

2. 是非題 [B] (10 分；下列 (2-1) ~ (2-5) 各題敘述若正確請以 **T** 或 O 表示，若錯誤請以 **F** 或 × 表示；每題 2 分，答對得 2 分，答錯得 0 分並倒扣 1 分，未作答得 0 分)
For each statement in (2-1) ~ (2-5), please indicate T or O if it is correct and indicate F or × otherwise. (You earn 2 points for each correct answer and lose 1 point for each incorrect answer.)

(2-1) 欲實現樹狀(Tree)結構之資料結構可使用單鏈結串列(Singly linked list)。

(2-2) 若將二元樹(Binary tree) T_1 改變為引線二元樹(Threaded binary tree)之組成結構 T_2，比較儲存所需使用之記憶體空間則 T_1 少於 T_2；但比較進行走訪(Traversal)所需使用之記憶體空間則 T_1 多於 T_2。

(2-3) 一圖形(Graph)資料結構 $G = (V, E)$之頂點(Vertex)數並以$|V|$表示，其邊線(Edge)數以$|E|$表示；若 $|E| > |V|^2$，則 G 必為一多重圖形(Multigraph)。

(2-4) 若 T 為一 2-3 樹(2-3 tree)，則亦 T 為一 B 樹(B tree)。

(2-5) 若 T 為一 B 樹(B tree)則對 T 插入(Insert)一新節點，可能減少 T 之高度；對 T 刪除(Delete)一節點，可能增加 T 之高度。

3. 名詞或術語解釋 (15 分；請解釋下列 (3-1) ~ (3-5)各題之名詞或術語；每題 3 分)
Term explanation (3 points each)

(3-1) Priority Queue
(3-2) Uniform hash function
(3-3) Optimal binary search tree
(3-4) Minimum cost spanning tree
(3-5) Weight-biased leftist tree

國立成功大學 104 學年度碩士班招生考試試題

系所組別：電機工程學系丁組

考試科目：資料結構　　　　考試日期：0211，節次：2

4. 選擇題 (30 分，每小題 3 分；若你認爲該題無適合之選項可作爲答案，請以符號 Ø 作答)
For each question from (4-1) to (4-10), please choose the most suitable item from the ones given. In case that you think none of the given items can be selected as answer, mark Ø as your answer. (3 points each)

(4-1) 下列何者資料搜尋方法於使用前，不必先將資料依據鍵值(Key value)以特定方式排列並儲存？
(A) 循序搜尋(Sequential search)法　(B) B^+-Tree 搜尋法
(C) 費氏搜尋(Fibonacci search)法　(D) 內插搜尋(Interpolation search)法
(E) 索引搜尋(Index search)法

(4-2) 使用雜湊(Hashing)法時，以 ***ht*** 表示雜湊表(Hash table)，且 ***ht*** 分爲 ***ht***[0]、***ht***[1]、…、***ht***[***b***-1] 等共計 ***b*** 組戽斗(Bucket)，每一戽斗(Bucket)可存入 ***s*** 筆資料；雜湊函式(Hash function)以 ***f*** 表示。若 ***n*** 爲實際存入 ***ht*** 之資料筆數，且發生碰撞(Collision)之情形；下列何者爲發生碰撞(Collision)可能之原因：
(A) $n < b$　(B) $n > b$　(C) $s = 1$　(D) $s > 1$　(E)可能原因與 ***s*** 及 ***b*** 之值無關

(4-3) 下列何者已知之資訊可決定一"二元樹(Binary tree)" ***T*** 之組成結構？
① 已知將 ***T*** 視爲圖形(Graph)結構與表示 ***T*** 之接鄰矩陣(Adjacency matrix)。
② 已知對 ***T*** 分別進行階序走訪(Level-Order traversal)與前序走訪(Preorder traversal)之輸出。
③ 已知對 ***T*** 分別進行前序走訪(Preorder traversal)與後序走訪(Postorder traversal)之輸出。
④ 已知對 ***T*** 進行前序走訪(Preorder traversal)之輸出且 ***T*** 之組成中無度數(Degree)爲 1 之節點(Node)。
⑤ 已知對 ***T*** 進行中序走訪(Inorder traversal)之輸出且 ***T*** 爲完整二元樹(Complete binary tree)。
請由下列選項中選出最適合者：
(A) ①③足以決定 ***T*** 之組成結構，④⑤不足以決定 ***T*** 之組成結構。
(B) ②⑤足以決定 ***T*** 之組成結構，①④不足以決定 ***T*** 之組成結構。
(C) ②③足以決定 ***T*** 之組成結構。
(D) ②④不足以決定 ***T*** 之組成結構。
(E) 僅③可足以決定 ***T*** 之組成結構。

(4-4) If a set of ***n*** data records is processed through the three stages as listed below one by one, what is the total time complexity of the three stages?
Stage 1: Sort the data records into non-decreasing order according to key K1 by merge sort;
Stage 2: Sort the data records into non-increasing order according to key K1 by bubble sort;
Stage 3: Sort the data records into non-decreasing order according to key K1 by quick sort;
(A) $O(1)$　(B) $O(n)$　(C) $O(n \cdot \log n)$　(D) $O(n^2)$　(E) $O(n^3)$

(4-5) 下列何者爲使用雜湊(Hashing)法進行資料搜尋之優點：
(A) 進行資料搜尋時不需要比較鍵值(Key value)　(B) 最糟情況之搜尋時間複雜度爲 $O(1)$
(C) 節省記憶體儲存空間　(D) 鍵值密度(Key density)可大於 100%
(E) 資料不需要依據鍵值(Key value)排序後儲存。

Questions (4-6) to (4-10) are independent unless otherwise indicated, but they are based on the same scenario as described below:
Suppose that we need to sort a data set of n data records using a comparison-based sorting algorithm. If the length of data record R_i is L_i where $(1 \leq i \leq n)$ and $(m_1 \leq L_i \leq m_2)$) hold, and there are two keys K1 and K2 associated with each data record for the purpose of identification.

(4-6) If $n = 250000$ and $1500 \leq L_i \leq 2250$, then which sorting algorithm is most suitable to sort the data records into non-decreasing order based on K1?

(A) Bubble sort (B) Selection sort
(C) Insertion sort (D) Quick sort
(E) Merge sort (F) Heap sort

(4-7) If $n > 200000$ and the data records have been sorted into non-increasing order based on K1 and stored as a doubly linked list, and we want to sort the data records into non-decreasing order based on K2, what is the average case time complexity that we can expected?

(A) $O(1)$ (B) $O(n)$ (C) $O(n \cdot \log n)$ (D) $O(n^2)$ (E) $O(n^3)$

(4-8) If K1 is not unique and we want to sort the data records into the non-decreasing order based on both K1 and K2 such that the data records with the same value of K1 are ordered by non-decreasing order of K2, what is the worst case time complexity that we can expected?

(A) $O(n \cdot \log n)$ (B) $O(n^2)$ (C) $O(n^2 \cdot \log n)$ (D) $O(n^2 \cdot (\log n)^2)$ (E) $O(n^3)$ (F) $O(n^4)$

(4-9) If K1 is unique and the data records have been sorted into non-increasing order based on K1, now we want to sort the data records into the non-decreasing order based on K2, what is the best case time complexity that we can expected when $n > 200000$ and the data records are stored into an array?

(A) $O(1)$ (B) $O(n)$ (C) $O(n \cdot \log n)$ (D) $O(n^2)$ (E) $O(n^2 \cdot \log n)$

(4-10) If $n < 1000$ and $150000 \leq L_i \leq 250000$, then which sorting approach is most suitable to sort the data records into non-decreasing order based on K1?

(A) Store the data records into a singly linked list and use selection sort algorithm
(B) Store the data records into an array and use selection sort algorithm
(C) Store the data records into an array and use insertion sort algorithm
(D) Store the data records into an array and use quick sort algorithm
(E) Store the data records into a singly linked list and use quick sort algorithm
(F) Store the data records into a singly linked list and use merge sort algorithm

5. Please do the following: (15 points)

(5-1) Please develop an algorithm based on the conditions listed below.

(a) Name: **Max2nd**

(b) Function/Purpose: Find the element with the second maximum value in an array of ***n*** integers.

(c) Input: an array ***A*** of ***n*** integers.

(d) Output: the index of the element with the second maximum value in ***A***. For example, if the elements in array ***A*** of 5 integers are:

A[1] = 26, A[2] = 49, A[3] = 14, A[4] = 82, A[5] = 31,

then the output of algorithm **Max2nd** is 2.

(5-2) Analyze the time complexity of your algorithm developed in (5-1).

6. Assume that the current population of a country is 10 millions. If the birth rate per year and death rate per year are 1.25% and 0.75% respectively, and the country admits 1000 immigrants each month. Please find the number of years from now that the population of this country will exceed 50 millions. (5 points)

7. Please give an implementation for each program units listed below. Note that your implementation must use programming language C or C++ and meet the criteria to get the points. (5 points each)

(7-1) A function ***Swap*** that, when given 2 integer variables as its arguments, swap (exchange) their values. Note that

(a) the ***Swap*** function can not have local data declaration in side it; and

(b) Neither accessing global data nor calling a function is allowed inside the ***Swap*** function.

(7-2) A function ***r4bits*** that returns the number of 1's in the rightmost 4 bits of the 32-bit integer argument. For example, the values returned by ***r4bits(98)*** and ***r4bits(11)*** are respectively 1 and 3.

(7-3) A program that reads an integer ***m*** (100 < ***m*** < 365) and print out the following message:

"The ***m***th day of 2014 is in *XXX*"

where XXX is the name of a month in a year. For example, if 168 is read, the message printed will be

The 168th day of 2014 is in June.