# 中原大學 95 學年度碩士班入學考試

**3 月 18 日 14:00~15:30 資訊工程系**

**誠實是我們珍視的美德，**
**我們喜愛「拒絕作弊，堅守正直」的你！**

**科目： 資料結構與演算法** **（共 6 頁第 1 頁）**

**■可使用計算機，惟僅限不具可程式及多重記憶者 □不可使用計算機**

**注意：請按題號作答，並標明答案。**

1. For each of the following statements, determine **True** or **False**. (5%)
   (a) $f(n) = \lg(n!),\ g(n) = n^2,\ f(n) = \Omega(g(n))$.
   (b) $f(n) = 2^{\lg n} + \lg n,\ g(n) = n,\ f(n) = \Theta(g(n))$.
   (c) $f(n) = n^3 + n!,\ g(n) = 2^n + \lg n,\ f(n) = O(g(n))$.
   (d) $f(n) = 2f(n/2) + n,\ g(n) = n^2,\ f(n) = O(g(n))$.
   (e) $f(n) = f(n-1) + f(n-2),\ g(n) = n!,\ f(n) = \Omega(g(n))$.
   (Note: lg is logarithm of base 2).

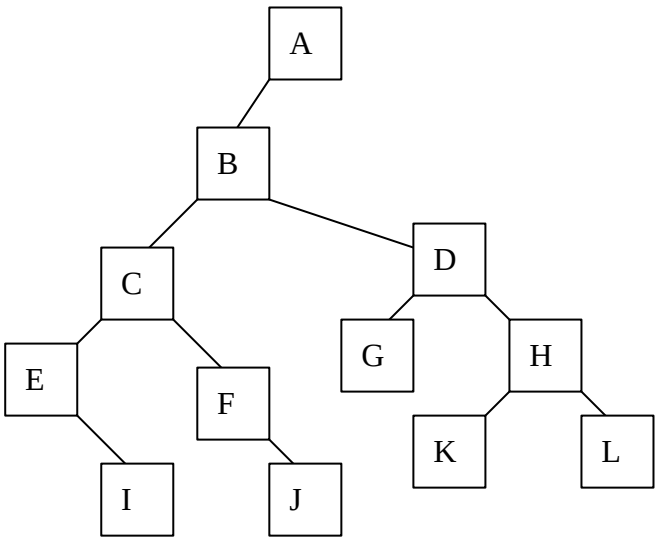
2. Tower of Hanoi problem is one of the most important problems in computer science. Suppose there are $n$ disks and three poles, namely (1) source, (2) destination, and (3) spare, respectively, the objective is to transfer the disks originally stacked on the source pole to the destination pole.
   (a) If we develop the C++ version of the Towers solution as follows, please complete the codes (3%): **(不可直接作答於試題)**

```
void Towers (int Count, char Source, char Dest, char Spare)
{
    if (Count == 1) {
      Cout << "Move disk from pole " << Source <<
              " to pole " << Dest << endl;
    }
    else
    {
        Towers (Count-1, _____, _____, _____);
        Towers (1, _____, _____, _____);
        Towers(Count-1, _____, _____, _____);
    } // end if
} // end Towers
```

   (b) Model the complexity of the Hanoi tower problem, i.e., write the recurrence that determines the number of moves needed for $n$ disks (2%).
   (c) Solve the recurrence to determine the number of moves needed for $n$ disks (2%).

3. (a) Convert the expression A-B/(C*D/E) from infix to prefix and postfix. (3%)
(b) Convert the expression /-*+ABC-DE+FG from prefix to infix and postfix. (3%)

4. Determine the sequence of characters to traverse the following binary tree in (a) preorder; (b) inorder; (c) postorder. (6%)




5. (a) Complete the routine **DELETE** to delete a node *p* from a doubly linked list.

```
struct node {
        int info;
        struct node *left, *right;
};
typedef struct node *NODEPTR;

DELETE (p, px)
        NODEPTR p;
        int *px;
{
        NODEPTR q, r;
        *px = p -> info;
        q = p -> left;
        _____          (不可直接作答於試題)
        freenode (p);
        return;
} (7%)
```

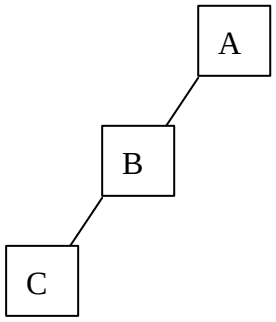
(b) Complete the routine **InsertRight** to insert a node with information field *x* to the right of node *p* in a doubly linked list. (9%)

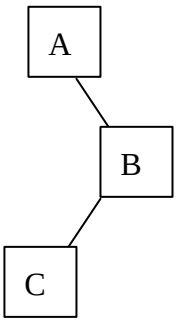
```
struct node {
        int info;
        struct node *left, *right;
};
typedef struct node *NODEPTR;

InsertRight (p, x)
        NODEPTR p;
        int x;
{
        NODEPTR q, r;
        q = getnode();
        q-> info = x;
        _____________   (不可直接作答於試題)
        return;
}
```

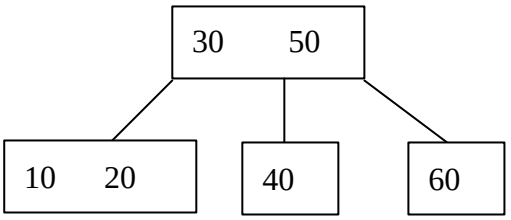
6. Determine the time complexity of the following sorting algorithms to unsorted *n* numbers $X_1, X_2 \ldots, X_n$ in sequence:
(a) Quick Sort; (b) Heap Sort; (c) Selection Sort; (d) Bubble Sort. (4%)

7. Balance the following two trees to satisfy the AVL tree property. (8%)
(a) (b)



8. Insert (a) 70; (b) 70, 80, 90; (c) 70, 80, 90, 100 into the following 2-3-4 tree, and show the 2-3-4 tree results? (10%)

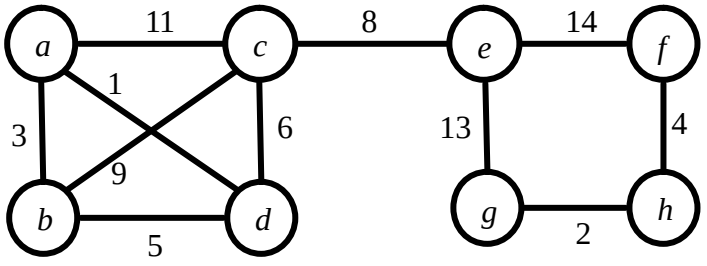



9. Huffman codes are widely used for data compression. Suppose we have a 1000-character data file that we wish to store compactly. The text file contains only 6 distinct characters, i.e., {*A, B, C, D, E, F*}, and the corresponding probabilities are given in the following table.

| Character | Probability |
|---|---|
| *A* | 0.2 |
| *B* | 0.05 |
| *C* | 0.15 |
| *D* | 0.23 |
| *E* | 0.3 |
| *F* | 0.07 |

(a) Illustrate the Huffman's tree corresponding to the Huffman codes. (4%)
(b) Using the Huffman's codes, encode the word "*ACE*". (3%)
(c) Compute the total number of bits required to store the data file if the Huffman codes are used. (3%)

10. Given the undirected graph below, please answer the following questions:




(a) Use the Breath-first search (BFS) to traverse the graph with node *a* as the starting node. Determine the BFS sequence (Note: At any moment, if more than one node can be visited next, always select the one in alphabetic order) (2%)

(b) Instead, use the Depth-First search (DFS) to traverse the graph with node *a* as the starting node. Determine the DFS sequence (Similarly, when more than one node can be visited next, always select the one in alphabetic order) (2%).

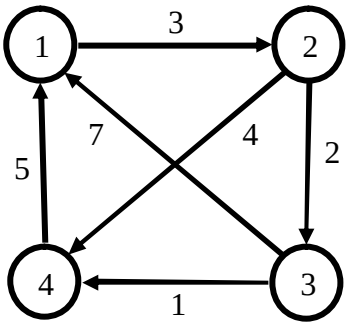
(c) Calculate the total number of all possible spanning trees? (2%)

(d) Use the *Prim*'s algorithm to find the minimum-cost spanning tree (starting from node *a*), illustrate your results step by step. (3%)

(e) Does an Euler's circuit exist for the graph (Yes / No)? (2%)

(f) Does a Hamiltonian cycle exist for the graph (Yes / No)? (2%)

11. Suppose the ***Floyd-Warshall*** algorithm is used to solve the all-pairs shortest-paths problem for the following directed graph. Please answer the following questions:




(a) If $d_{ij}^{(k)}$ is defined as the weight of a shortest path from vertex *i* to vertex *j* for which all intermediate vertices are in the set {1, 2, …, *k*}. Complete the following algorithm: (3%)

**FLOYD-WARSHALL**(*W*)
1. $n \leftarrow rows[W]$
2. $D^{(0)} \leftarrow W$
3. **for** $k \leftarrow 1$ to $n$
4. **for** $i \leftarrow 1$ to $n$
5. **for** $j \leftarrow 1$ to $n$
6. do $d_{i,j}^{(k)} =$ _____
7. Return $D^{(n)}$

**(不可直接作答於試題)**

(b) Compute the following intermediate results: (4%)

$$D^{(0)} = \begin{pmatrix} 0 & - & - & - \\ - & 0 & & - \\ - & - & 0 & - \\ - & - & - & 0 \end{pmatrix} \quad D^{(1)} = \begin{pmatrix} 0 & - & - & - \\ - & 0 & & - \\ - & - & 0 & - \\ - & - & - & 0 \end{pmatrix}$$

**(不可直接作答於試題)**

(c) Express the running time in asymptotic Θ-notation. (2%)

12. For each of the following problems, determine if it's in class **P** or **NP**: (6%)
    (a) Tower of Hanoi problem;
    (b) Longest Common Subsequence (LCS);
    (c) Traveling-Salesman problem (TSP);
    (d) 2-Coloring of a graph;
    (e) Euler's Tour problem;
    (f) Hamitonian-Cycle problem;