115EE07

# 國立臺北科技大學115學年度碩士班招生考試

## 系所組別：2152 電機工程系碩士班戊組

## 第一節　資料結構　試題（選考）



**注意事項：**

1. 本試題分為四大部分，共24題，每題配分標註於題後，共100分。
2. 不必抄題，作答時請將試題題號及答案依照順序寫在答案卷上。
3. 全部答案均須在答案卷之答案欄內作答，否則不予計分。

I. True or False: (30%)

1. A stack follows the Last-In-First-Out (LIFO) principle. **(3%)**

2. In a binary search tree, the left child of any node must be greater than its parent node. **(3%)**

3. A queue can be efficiently implemented using two stacks. **(3%)**

4. The time complexity of accessing an element in a hash table is always $O(1)$ in the worst case. **(3%)**

5. A complete binary tree with n nodes has a height of $O(\log n)$. **(3%)**

6. In a doubly linked list, each node contains pointers to both the next and previous nodes. **(3%)**

7. A circular queue eliminates the problem of wasted space that occurs in a linear queue. **(3%)**

8. Depth-first search (DFS) always finds the shortest path between two nodes in a graph. **(3%)**

9. The maximum number of nodes in a binary tree of height $h$ is $2^{h+1}-1$. **(3%)**

10. An AVL tree is a self-balancing binary search tree where the height difference between left and right subtrees of any node is at most 1. **(3%)**

II. Multiple Choice: (15%)

1. What is the time complexity of inserting an element at the beginning of a singly linked list? **(3%)**
   a) $O(1)$　b) $O(n)$　c) $O(\log n)$　d) $O(n^2)$

2. Which data structure is most suitable for implementing a browser's back button functionality? **(3%)**
   a) Queue　b) Stack　c) Binary Tree　d) Hash Table

3. In a max heap, where is the largest element located? **(3%)**
   a) At a leaf node　b) At the root node
   c) At the rightmost node　d) At the deepest level

4. In a B-tree of order m, what is the maximum number of children a non-root internal node can have? **(3%)**
   a) $m-1$　b) $m$　c) $m+1$　d) $2m$

5. What is the worst-case time complexity of Dijkstra's algorithm for finding the shortest path in a graph with V vertices and E edges when implemented with a binary min-heap? **(3%)**
   a) $O(V)$　b) $O(V+E)$　c) $O\big((V+E)\log V\big)$　d) $O(V^2)$

III. Short answers: (20%)

1. The evaluation of the prefix expression: / + * - * 5 3 7 2 10 + 4 2. **(4%)**

2. Given a binary search tree with root 45, and the following nodes: 25, 65, 15, 35, 55, 75, 10, 20. Insert 18, 30, 50, 70 (in this order) into the binary search tree. What is the pre-order traversal? **(4%)**

3. Let $T(n)=3T\left(\frac{n}{3}\right)+n, T(1)=2, then\ T(n)=O(b)$. What is b? **(4%)**

4. In a hash table with 20 slots using linear probing, what is the worst-case number of comparisons needed to search for an element that is NOT in the table when the load factor is 0.75? **(4%)**

5. Order the following functions by their growth rate from slowest to fastest: $2^{\sqrt{n}}, n^{1.5}, \log(n!), \frac{n^2}{\log n}, n\log^2 n, 3^n, n^{\log n}$. **(4%)**

注意：背面尚有試題

IV. **Answer the following questions: (35%)**

1. Consider a CPU task scheduler where processes are prioritized based on urgency:
   Priority 1: Critical system tasks (highest priority)
   Priority 2: Real-time applications
   Priority 3: User applications
   Priority 4: Background tasks (lowest priority)
   Given the following task arrivals:

| Task A | Task B | Task C | Task D | Task E | Task F | Task G |
|---|---|---|---|---|---|---|
| 3 | 1 | 4 | 2 | 1 | 3 | 2 |

   (a) Show the min-heap after all tasks have been added (lower priority number = higher urgency). **(5%)**
   (b) If 3 tasks are executed (removed from queue), show the order they are executed and the heap after each removal. **(5%)**

2. Consider the following undirected graph:
   Vertices: A, B, C, D, E, F
   Edges: (A,B), (A,C), (B,D), (C,D), (C,E), (D,E), (D,F), (E,F)
   (a) Draw the adjacency matrix representation of this graph. **(5%)**
   (b) Starting from vertex A, show the order of vertices visited using Breadth-First Search (BFS). Assume that when multiple vertices are available, they are visited in alphabetical order. **(5%)**

3. Consider a hash table of size 11 using the hash function $h(k) = k\%11$. Use linear probing to resolve collisions.
   Insert the following keys in order: 23, 45, 12, 67, 34, 78, 56, 89, 31.
   (a) Show the final state of the hash table after all insertions. **(5%)**
   (b) How many collisions occurred during the insertion process? **(5%)**

4. Given the following C++ function, what will this function return for **abc##d**. Explain your reasoning. **(5%)**

```
string processString(string input) {
    stack<char> S;
    string result = "";
    for(char c : input) {
        if(c == '#') { if(!S.empty()) S.pop(); }
        else { S.push(c); }
    }
    while(!S.empty()) { result = S.top() + result; S.pop(); }
    return result;
}
```