國立中央大學112學年度碩士班考試入學試題

所別： 資工類

科目： 資料結構與演算法

一、 複選題(50分；每題5分，答錯1題倒扣1分，倒扣至複選題0分為止)

1. Which of the following isn't open addressing overflow handling?
(A) Linear probing (B) dynamic hashing (C) rehashing (D) quadratic probing

2. Apply Quick sort on a given list [9, 7, 24, 6, 11, 4, 2, 19]. Which of the following two are the resulting sequences of the first phase, when the first element or second element is chosen as pivot?
(A) 4, 7, 2, 6, 9, 11, 24, 19
(B) 7, 6, 24, 11, 9, 4, 2, 19
(C) 2, 6, 4, 7, 11, 24, 9, 19
(D) 7, 6, 4, 2, 9, 24, 11, 19

3. A leftist tree is a min tree satisfying that dist(RChild(i)) <= dist(LChild(i)), where dist(j) denotes the number of edges on the shortest path from node j to a leaf node, RChild(i) and LChild(i) denote the right child and left child of node i, respectively. Which of the following statements about leftist trees are correct? (A) The length of path to rightmost leaf is O(log n) for a leftist tree with n nodes. (B) Merging two leftist trees is to merge the right subtree of one tree with the other. (C) Delete min takes O(log n) time. (D) If the path to leftmost leaf has x nodes, then the leftist tree has at least $2^x - 1$ nodes.

4. Given the input list L = [28, 203, 16, 30, 123, 521, 63, 528, 210, 216, 941, 45]. Which of the following statements is (are) correct? (A) LSD radix sort is a non-comparative sorting algorithm. (B) At the end of the second pass of LSD Radix sort, the sixth element of the resulting chain is 123. (C) At the end of the third pass of LSD Radix sort, the sixth element of the resulting chain is 123. (D) At the end of the first pass of LSD Radix sort, the sixth element of the resulting chain is 63.

5. Which of the following statement(s) is (are) correct? (A) The time complexity of inserting an element to an n-node min-Heap is O(n). (B) The time complexity of searching an element in an n-node max-Heap is O(n). (C) Building the min-Heap by inserting these nodes one by one {42, 9, 23, 37, 4, 34, 2}. The level order traversal sequence of the min-Heap is {2, 9, 4, 42, 37, 23, 34}. (D) The time complexity of deleting min in an n-node min-Heap is O(log n)

6. In a stack structure, let X denote a push operation and let Y denote a pop operation. After 8 stack operations, consisting of 4 pushes and 4 pops, an input sequence 1234 may change its order. For example, after XYXXYYXY is performed, 1234 will become 1324. Which of the following is (are) possible output sequences? (A) 1243 (B) 4123 (C) 2134 (D) 4321

注意:背面有試題

7. Given an undirected graph *G*, where n denotes the number of vertices and m denotes the number of edges (m >> n). Which of the following statement(s) is (are) true about the graph *G*?
   (A) If G is represented by an adjacency matrix, the space complexity is O(n^2).
   (B) If G is represented by an adjacency list, the space complexity is Θ(m).
   (C) If G is represented by an adjacency multilist, the space complexity is Θ(m).
   (D) If G is represented by an adjacency matrix, the time complexity of determining whether G is connected is O(m).

8. Given an n-node undirected graph. Which of the following statements are correct? (A) Single source single destination shortest path algorithm takes O(n^2) time complexity. (B) Single source multiple destination shortest path algorithm takes O(n^2) time complexity. (C) All pair shortest path algorithm takes O(n^3) time complexity. (D) All pair shortest path algorithm takes O(n^2) time complexity.

9. Which of the following statements are correct? (A) Inserting the first element of an n-node one-dimension array has time complexity higher than a singly linked list. (B) Removing the third element of an n-node one-dimension array takes O(n) time complexity. (C) Inserting the first element of a circular singly linked list has time complexity much higher than that of a singly linked list. (D) Inserting the last element of an n-node singly linked list takes O(n) time complexity.

10. Which of the following statements are correct? (A) The postfix of infix expression **b+a*c/d** is **bac+d/***. (B) The postfix of **(a+b)/(c-d)*e** is **ab+cd-*/**. (C) The infix of the postfix expression **ab+c*d/ef-/** is **(a+b)*c/(d/(e-f))**. (D) The prefix of the postfix expression **abc+d/*** is ***a/+bcd**.

二、問答題(50分) 請用深色筆書寫

1. Given a directed graph *G*=(*V*, *E*), and two vertices *u* and *v* in *V*, we call vertex *v* is reachable from *u*, if there exists a directed path from *u* to *v*. A vertex *s* in *V* is called a source vertex if every vertex in *V* is reachable from *s*.

   (a) (8%) Given a directed graph *G*=(*V*, *E*), and a specified vertex *v* in *V*, design a linear time algorithm (i.e. your algorithm should run in $O(|V|+|E|)$ time) to determine if *v* is a source vertex. You need to describe the data structure used in your algorithm.

   (b) (8%) Given a directed acyclic graph (DAG; a directed graph is acyclic if it contains no directed cycles) *G*=(*V*, *E*), you are asked to determine if *G* contains a source vertex. If you apply the algorithm of subproblem (a) on every vertex of *G*, you will get an algorithm runs in $O(|V|^2+|V||E|)$ time. It is not desirable. Design a more efficient algorithm for this problem. Analyze the time complexity of your algorithm.

注意:背面有試題

(c) (9%) Given a directed graph $G=(V, E)$, you are asked to determine if G contains a source vertex. Note that the given graph may contain directed cycles. As in subproblem (b) an $O(|V|^2+|V||E|)$ time algorithm is not acceptable. Design a more efficient algorithm for this problem and analyze the time complexity of your algorithm.

2. A non-deterministic (ND) algorithm has two phases, the choosing phase and the checking phase, for solving a given problem. The former is for selecting one from a specific set of choices iteration by iteration. The latter is for checking if all selected choices constitute a solution to the problem. If so, the algorithm returns SUCCESS; otherwise, FAILURE. It is assumed that an ND algorithm always selects choices that lead to the return of SUCCESS unless there are no such choices. A problem is called an NP problem if there exists a polynomial time-complexity ND algorithm solving the problem. For example, the famous satisfiability (SAT) problem is an NP problem. The SAT problem is to determine if a given Boolean formula $f(x_1,\ldots,x_n)$ of n Boolean variables $x_1,\ldots,x_n$ is satisfiable or unsatisfiable. A formula $f(x_1,\ldots,x_n)$ is satisfiable (resp., unsatisfiable) if there exists an (resp., no) TRUE-FALSE assignment of the n variables to make the formula TRUE. The following polynomial time-complexity ND algorithm, called ND-SAT, can solve the SAT problem, which is the evidence that the SAT problem is an NP problem.

```
Algorithm: ND-SAT
Input: a Boolean formula $f(x_1,\ldots,x_n)$ of $n$ variables $x_1,\ldots,x_n$
Output: SUCCESS if $f$ is satisfiable; FAILURE, otherwise.
for $i \leftarrow 1$ to $n$ do
  $x_i \leftarrow$ choice({TRUE, FALSE}) //Choose TRUE or FALSE to assign to $x_i$
if $f(x_1,\ldots,x_n)$ == TRUE then //Check if $f(x_1,\ldots,x_n)$ is satisfiable or unsatisfiable
   return SUCCESS
else
   return FAILURE
```

In practice, we can prove a problem to be an NP problem by showing a polynomial time-complexity ND algorithm solving the problem. (a) By this concept, please prove that the exact cover decision problem (ECDP) is an NP problem by showing a polynomial time-complexity ND algorithm solving the ECDP (18%). Note that you should follow the above-mentioned ND algorithm definition and the format of the ND-SAT algorithm. That is, the ND algorithm should contain the input description, the output description, the choosing phase, the checking phase, and return statements; otherwise, you will lose some points. (b) Furthermore, please analyze the time complexity of your ND algorithm in terms of the big O notation to show that it indeed has a polynomial time complexity (7%). The ECDP is defined as follows. Given a universal set $U=\{u_1,\ldots,u_m\}$ of m elements, and a collection $S=\{S_1,\ldots,S_n\}$ of n sets, where S_i is a non-empty subset of U, $1\leq i\leq n$, the ECDP is to determine if there exists a collection S^* of sets that is an exact cover of U, where $S^*\subseteq S$. A collection S^* of sets is an exact cover of U if every element u in U appears exactly once in only one set of S^*. For example, suppose $U=\{1,2,3,4,5,6,7\}$ is a universal set of seven elements, and $S=\{A,B,C,D,E\}$ is a collection of five sets, where $A=\{1,2,7\}$, $B=\{1,4\}$, $C=\{4,5\}$, $D=\{3,5,6\}$, and $E=\{4\}$. Then, $S^*=\{A,D,E\}\subseteq S$ is an exact cover of U. In summary, the ECDP with the input of $U=\{1,2,3,4,5,6,7\}$ and $S=\{A=\{1,2,7\}, B=\{1,4\}, C=\{4,5\}, D=\{3,5,6\}, E=\{4\}\}$ will return SUCCESS, since $S^*=\{A,D,E\}\subseteq S$ is an exact cover of U.