本試卷共 20 題選擇題，作答於答案卡。每題答對 5 分，不答零分，答錯倒扣 2.5 分。

I When CAN the worst case of Quicksort occur? **(1)**

C_1: Quick Sort where leftmost (or rightmost) element is always chosen as pivot
C_2: leftmost element is chosen as pivot and array is already sorted in SAME order
C_3: leftmost element is chosen as pivot and array is already sorted in REVERSE order

(A) C_1	(B) C_2	(C) C_1 and C_2	(D) C_2 and C_3

II A typical dynamic programming example is the snake sequence. Given a grid of numbers, find maximum length snake sequence and print it. A snake sequence is made up of adjacent numbers in the grid such that for each number, the number on the right or the number below it is +1 or -1 its value. For example, if you are at location (x, y) in the grid, you can either move right i.e. (x+1, y) if that number is ± 1 or move down i.e. (x, y+1) if that number is ± 1. The length of a snake sequence is defined as the number of moves. Given the following grid (A matrix with M rows and N columns),

9, 6, 5, 2
8, 7, 6, 5
7, 3, 1, 6
1, 1, 1, 7

(a) The maximum LENGTH of snake sequence is **(2)**

(A) 6	(B) 7	(C) 5	(D) 4

(b) Time complexity of above solution is **(3)**

(A) $O(M \times N)$	(B) $O(M^2)$	(C) $O(N^2)$	(D) $O(M)$

(c) The last number of the maximum-length snake sequence is **(4)**

(A) 9(0,0)	(B) 6(3,2)	(C) 7(3,3)	(D) 1(2,3)	(E) 7(0,2)

(d) The third number of the maximum-length snake sequence is **(5)**

(A) 7(1,1)	(B) 7(0,2)	(C) 3(1,2)	(D) 6(2,1)	(E) 8(0,1)

III For questions with sequences, we constantly split the problem into a number of sub-problems that are smaller instances of the same problem, and we try to solve the smaller problems recursively. We called this type of algorithm **(6)**

(A) Dynamic programming	(B) Divide and conquer	(C) Greedy algorithms	(D) Quicksort

IV There is a large enough array A with indices p and q which are initialized by 0, and functions $f(x)$ and $g()$ are defined as follows. Array A and indices p and q can only be accessed in the functions $f(x)$ and $g()$.

$f(x)$

- $p \leftarrow p + 1$
- $A[p] \leftarrow x$

$g()$

- $x \leftarrow A[p]$
- $p \leftarrow p - 1$
- **return** x

見背面

(a) Which data structure is operated by those functions? **(7)**

(A) Queue	(B) Stack	(C) Hash	(D) Heap

(b) If functions $f(x)$ and $g()$ are changed as following:

$f(x)$

- $A[q] \leftarrow x$
- $q \leftarrow q + 1$

$g()$

- $x \leftarrow A[p]$
- $p \leftarrow p + 1$
- **return** x

Which data structure is operated by those functions? **(8)**

(A) Queue	(B) Stack	(C) Hash	(D) Heap

V There is a binary tree stored in an array as:

$$T[] = \{+, a, *, null, null, -, d, null, null, null, null, b, c\},$$

in which $T[i]$ is the parent of $T[2i+1]$ and $T[2i+2]$, where i is an index and *null* means there is no element in the slot.

(a) What is its postorder traversal result? **(9)**

(A) $+a*-dbc$	(B) $a+b-c*d$	(C) $abc-d*+$	(D) $+a*-bcd$

(b) What is its BFS traversal result? **(10)**

(A) $+a*-dbc$	(B) $a+b-c*d$	(C) $abc-d*+$	(D) $+a*-bcd$

VI There are 10 hexadecimal data, 53, 3B, 66, 43, 60, 5B, 7C, 14, 30, 37, inserted in the given order into an empty hash table. The table is implemented using a circular array of 10 slots, and one slot can only have one item. The hash function for the table is $h(k) = k \mod 9$.

(a) Which data is the first one that occurs a collision? **(11)**

(A) 43	(B) 5B	(C) 14	(D) 37

(b) If the hash table uses linear probing to resolve collisions and has no resizing mechanism, which position in the array is 37 inserted? **(12)**

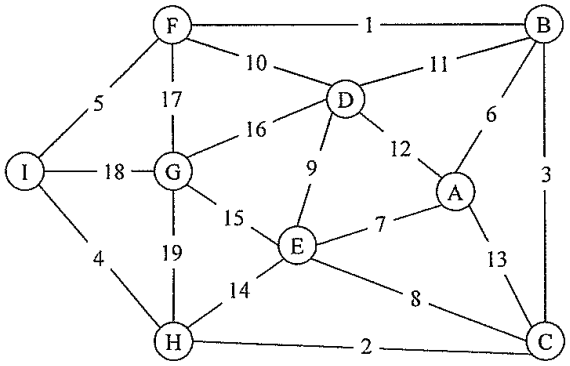
(A) 3	(B) 8	(C) 4	(D) 0

VII There is a linked-list with n elements, which are $x_1, x_2, ..., x_n$. Let $f(n)$ be the time complexity of inserting a new element x_{n+1} to the tail of the list, and $g(n)$ be that of deleting the last element x_n. If there is only one pointer pointed to the head of the list, we know to access the list is time-consuming, and functions $f(n)$ and $g(n)$ are denoted as $f_1(n)$ and $g_1(n)$ in this case. To improve it, we may also add another pointer pointed to the tail of it, so the functions $f(n)$ and $g(n)$ become $f_2(n)$ and $g_2(n)$. Which of the following option is false if n becomes large. **(13)**

(A) $\frac{f_1(n)}{g_1(n)} = 1$	(B) $\frac{f_2(n)}{g_2(n)} = n$	(C) $\frac{f_1(n)}{f_2(n)} = n$	(D) $\frac{g_1(n)}{g_2(n)} = 1$

接次頁

VIII Given the following graph, try to find its minimum spanning tree.



(a) What is the sixth edge that Kruskal's algorithm includes (the numbering of the inclusion order starts from 1, not 0)? **(14)**

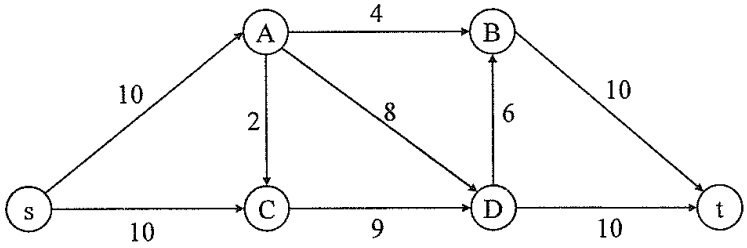
(A) 6	(B) 7	(C) 8	(D) 9

(b) What is the sixth vertex that Prim's algorithm includes if starting from vertex A (the numbering of the inclusion order starts from 1, **not** 0)? **(15)**

(A) C	(B) E	(C) H	(D) I

IX What is the value of the maximum flow for the following *st*-flow network? **(16)**

(A) 16	(B) 17	(C) 18	(D) 19



X The Knuth-Morris-Pratt (KMP) algorithm is an string-matching algorithm. Its core is the prefix function. Given a pattern $P[1..m]$, the prefix function for the pattern P is the function $\pi : \{1, 2, \ldots, m\} \to \{0, 1, \ldots, m-1\}$. What is $\pi[7]$ for the following pattern P? **(17)**

(A) 4	(B) 5	(C) 6	(D) 7

i	1	2	3	4	5	6	7	8	9	10
$P[i]$	a	b	a	b	a	b	a	b	c	a
$\pi[i]$										

見背面

XI Many string-matching algorithms build a finite automata. Below is a partially-completed state transition function for a string s of length 10 over the alphabet $\{A, B, C\}$. It is possible to reconstruct s from the partial table. How many A's does the string s have? **(18)**

(A) 4	(B) 5	(C) 6	(D) 7

	0	1	2	3	4	5	6	7	8	9
A				4	5		2			
B				0	0		7			3
C				0	0		0			10

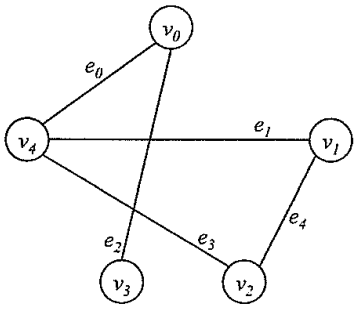
XII What can you infer from the facts that ProblemA is NP-complete and ProblemA linear-time reduces to ProblemB? **(19)**

C_1: If there exists an $O(N^3)$ algorithm for ProblemB, then $P = NP$.
C_2: If there does not exist an $O(N^3)$ algorithm for ProblemB, then $P \neq NP$.
C_3: If there exists an $O(N^3)$ algorithm for ProblemB, then there exists an $O(N^3)$ algorithm for ProblemA.
C_4: If there exists an $O(N^3)$ algorithm for ProblemA, then there exists an $O(N^3)$ algorithm for ProblemB.

(A) C_1 and C_3	(B) C_1 and C_4	(C) C_2 and C_3	(D) C_2 and C_4

XIII The VERTEX-COVER problem is to find a vertex cover of the size k in a given graph G. In the SUBSET-SUM problem, given a finite set $S \subset \mathbf{N}$ and a target $t \in \mathbf{N}$, we ask whether there is a subset $S' \subseteq S$ whose elements sum to t. The VERTEX-COVER problem is polynomial-time reducible to the SUBSET-SUM problem. Given an instance $< G, k >$ of the VERTEX-COVER problem, one can construct a corresponding instance $< S, t >$ of the SUBSET-SUM problem. Given the following graph G and $k = 3$, $\{v_1, v_3, v_4\}$ is the vertex cover of size $k = 3$. The corresponding set $S = \{1, 4, 16, 64, 256, 1040, 1041, 1093, 1284, 1344\}$ is constructed as the following table. What is the target t? **(20)**

(A) 2389	(B) 3417	(C) 3754	(D) 3758



			modified based 4						*decimal*
			e_4	e_3	e_2	e_1	e_0		
x_0	=	1	0	0	1	0	1	=	1041
x_1	=	1	1	0	0	1	0	=	1284
x_2	=	1	1	1	0	0	0	=	1344
x_3	=	1	0	0	1	0	0	=	1040
x_4	=	1	0	1	0	1	1	=	1093
y_0	=	0	0	0	0	0	1	=	1
y_1	=	0	0	0	0	1	0	=	4
y_2	=	0	0	0	1	0	0	=	16
y_3	=	0	0	1	0	0	0	=	64
y_4	=	0	1	0	0	0	0	=	256

試題隨卷繳回