本試卷共 17 題選擇題及 2 題非選擇題。選擇題每題有 5 個選項，第 1-7 題為單選題，第 8-17 題為複選題。單選題每題答對 5 分，不答或答錯為 0 分。複選題每題 5 分，每個選項單獨計分，選項答對者得 1 分，選項答錯者得 0 分。另若複選題整題未作答者，整題得 0 分。選擇題作答於答案卡，非選擇題 (第 18 及 19 題) 請作答於試卷本。

1. (5 points) Consider the following randomized algorithm, which prints out some number of exclamation marks (!) and some number of asterisks (*). Assume that the size of A is n initially. What are the expected numbers of !'s and *'s that the algorithm prints out? Choose the tightest possible answers.

```
RandomizedPrint(A):
    n = length(A)
    if n <= 1:
        print "!"
        return
    for i in {0,...,n-1}:
        print "*"
    Choose a uniformly random integer p in {1,...,n-1}
    RandomizedPrint(A[:p])
    RandomizedPrint(A[p:])
```

(A) !: $O(\log n)$, *: $O(n \log n)$
(B) !: $O(\log n)$, *: $O(n^2)$
(C) !: $O(n)$, *: $O(n \log n)$
(D) !: $O(n)$, *: $O(n^2)$
(E) !: $O(n \log n)$, *: $O(n^2)$

2. (5 points) Which of the following statements about the Akra-Bazzi theorem is correct?

(A) It can only be applied to recurrences where all subproblems have equal size.
(B) It requires that the function $g(n)$ in the recurrence satisfies a polynomial growth condition.
(C) It always produces a closed-form solution for the recurrence.
(D) It is less general than the Master Theorem for solving recurrences.
(E) The value of p in the theorem must always be an integer.

3. (5 points) Consider the following five DNA sequences:

a) ATCGTA
b) ATCGTT
c) GCATAG
d) GCATAC
e) TTAGCG

Which of the following statements is correct regarding the application of a c-means clustering algorithm to these sequences ($c = 3$)?

(A) The algorithm will always halt with exactly three clusters regardless of the chosen distance metric.
(B) Using Hamming distance as the similarity measure, sequences a and b will always be in the same cluster.
(C) The algorithm guarantees finding the globally optimal clustering solution for these sequences.
(D) Using k-mer frequency vectors ($k = 2$) as features, sequences c and d will never be in the same cluster.
(E) The time complexity of the k-means algorithm for clustering these sequences is $O(n)$, where n is the number of sequences.

見背面

4. (5 points) Consider two protein sequences A and B of lengths m and n respectively. You are tasked with implementing a dynamic programming algorithm to find the optimal global alignment between these sequences. Given the following definitions:

   - A match score of $+2$ for identical amino acids.
   - A mismatch penalty of -1 for different amino acids.
   - A gap penalty of -2 for each gap introduced.

   Which of the following statements is correct regarding the dynamic programming solution for this alignment problem?

   (A) The time complexity of the algorithm is $O(m+n)$.

   (B) The space complexity of the algorithm is $O(mn)$.

   (C) The recurrence relation for the optimal alignment score $S(i,j)$ at position (i,j) in the dynamic programming matrix is:

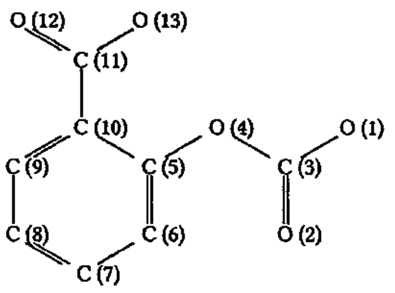
   $$S(i,j) = \max\{S(i-1,j-1)+\text{match}(A[i],B[j]), S(i-1,j)-2, S(i,j-1)-2\}.$$

   (D) The traceback procedure to reconstruct the optimal alignment always starts from the top-left corner of the dynamic programming matrix.

   (E) The algorithm guarantees finding all possible optimal alignments between the two sequences.

5. (5 points) Many graph theory can be used to find the number of a chemical structure; a possible one can be used according to the following steps:
   a) Represent the molecule as an undirected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges (bonds).
   b) Initialize each vertex $v \in V$ with a value equal to its degree (number of incident edges).
   c) For a fixed number of iterations or until convergence:
   a. For each vertex v, compute a new value as the sum of the current values of its neighbors (not including itself).
   b. Update all vertex values simultaneously.
   d) After the iterations, sort the vertices based on their final values in non-increasing order, tie breaking with smaller labels.

   Consider the following 2-Carboxyoxybenzoic acid structure represented as a graph (Numbers in parentheses are temporary labels for reference, not the final numbering.):



   Apply the above algorithm for two iterations to this graph. What is the correct ordering of the vertices after these iterations?

   (A) 5, 6, 10, 11, 9, 4, 3, 7, 8, 12, 13, 1, 2

   (B) 5, 10, 11, 9, 3, 6, 4, 7, 8, 12, 13, 1, 2

   (C) 5, 6, 10, 11, 9, 3, 4, 7, 8, 12, 13, 1, 2

   (D) 5, 10, 11, 9, 6, 3, 4, 7, 8, 12, 13, 1, 2

   (E) None of the above choices.

接 次 頁

6. (5 points) Consider a knapsack with capacity 17 and five types of items, each with its size and value listed below. All items are available in unlimited quantities and each item can be selected more than once. The goal is to maximize the total value of the items placed in the knapsack. If the maximum value of the knapsack is $10a+b$ where a and b are integers in $\{0,1,2,\cdots,9\}$, what is the value of $a+b$?

item	A	B	C	D	E
size	3	4	7	8	9
value	4	5	10	11	12

(A) 3
(B) 4
(C) 5
(D) 6
(E) 7

7. (5 points) Max-flow algorithms can be applied to determine the elimination of sports teams in a group. Consider the following record for a group of five teams. Note that the number of remaining games for a team does not necessarily equal the number of remaining games against the group's rivals, as teams may play opponents outside their own group. Only the team with the highest number of wins will advance to the next round. Our goal is to determine whether team E has already been eliminated.

The intuition behind constructing the graph is to assume that team E wins all of its remaining 28 (r_E) games, and then attempt to ensure that the number of wins for each competing team does not exceed the total possible wins of team E ($w_E + r_E = 76$). The constructed graph consists of 12 nodes: in addition to the source (s) and sink (t) nodes, there are six nodes representing the head-to-head games between teams A-D and four nodes representing teams A-D. What is the maximum flow value of the constructed graph?

				Games against each other				
Team	Wins (w_i)	Losses (l_i)	To play (r_i)	team A	team B	team C	team D	team E
team A	75	59	28	-	5	7	4	3
team B	71	63	28	5	-	2	4	4
team C	69	65	28	7	2	-	4	0
team D	63	71	28	4	4	4	-	0
team E	48	86	28	3	4	0	0	-

(A) 22
(B) 23
(C) 24
(D) 25
(E) 26

8. (5 points) Select all pairs of functions in the increasing order of growth. That is, select g_1, g_2 if g_2 grows faster than g_1.

$$f_1(n) = \binom{n}{5}, \quad f_2(n) = 2^{\log^4 n}, \quad f_3(n) = \binom{n}{n-4}, \quad f_4(n) = \sqrt{2^{\sqrt{n}}}, \quad f_5(n) = n^{5(\log n)^2}.$$

(A) $f_1(n), f_3(n)$
(B) $f_2(n), f_5(n)$
(C) $f_2(n), f_4(n)$
(D) $f_4(n), f_5(n)$
(E) $f_1(n), f_5(n)$

見背面

9. (5 points) Select all the correct statement(s) from the following statements regarding recurrence.

   (A) The solution to the recurrence $T(n) = 2T(n/2) + \Theta(n)$ is $\Theta(n^2)$.
   (B) The solution to the recurrence $T(n) = 7T(n/2) + \Theta(n^2)$ is $\Theta(n^{\lg 7})$.
   (C) The solution to the recurrence $T(n) = \log n + T(\sqrt{n})$ is $\Theta(\log n)$.
   (D) The solution to the recurrence $T(n) = 2^n T(n-1)$ is $\Theta((\sqrt{2})^{n^2+n})$. (Assume $T(n) = 1$ for n smaller than some constant c).
   (E) The solution to the recurrence $T(n) = T(n/6) + T(7n/9) + O(n)$ is $O(n)$. (Assume $T(n) = 1$ for n smaller than some constant c).

10. (5 points) Select all the correct statement(s) from the following statements regarding binary trees and binary search trees.

   (A) Every binary tree is a binary search tree.
   (B) Every binary search tree on n nodes has height $O(\log n)$.
   (C) In a binary search tree, we can find the next smallest element to a given element in $O(1)$ time.
   (D) Let A_1, A_2, and A_3 be three sorted arrays of n distinct real numbers. Constructing a balanced binary search tree of the set $A_1 \cup A_2 \cup A_3$ requires $O(n)$ time.
   (E) Let T be a complete binary tree with n nodes. Finding a path from the root of T to a given vertex $v \in T$ using breadth-first search takes $\Omega(n \lg n)$ time.

11. (5 points) Let S be a collection of n objects, each with a unique integer key $a_i \in \{0, 1, 2, \ldots, n^2 - 1\}$. Consider hashing all the objects in S into an array $A[0..m-1]$ of $m = \lceil n^{1.5} \rceil$ buckets. Each bucket stores a (singly) linked list of all objects hashed into the same bucket. The hashing algorithm works as follows. The algorithm first picks a hash function h uniformly at random from a hash family $\mathcal{H}$. Then, for each object $i \in S$, the algorithm computes $h(a_i)$ and inserts the object into the bucket $A[h(a_i)]$. The *number of collisions* τ is defined to be the total number of (unordered) object pairs hashed into the same bucket. We focus on the choice of the hash family $\mathcal{H}$ and the worst-case expected number of collisions $\mathbf{E}[\tau]$. Note that the term worst-case refers to the choice of the input set S. Select all statement(s) that are correct for every sufficiently large n.

   (A) If $\mathcal{H}$ is a uniform hash family, then $\mathbf{E}[\tau] = O(\sqrt{n})$.
   (B) If $\mathcal{H}$ is a universal hash family, then $\mathbf{E}[\tau] = O(\sqrt{n})$.
   (C) If $\mathcal{H} = \{h(x) = (ax + b) \bmod m \mid a, b \in \mathbb{Z} \text{ and } 0 \le a, b \le m-1\}$, then $\mathbf{E}[\tau] = O(\sqrt{n})$.
   (D) If $\mathcal{H} = \{h(x) = (x^2 + x + c) \bmod m \mid c \in \mathbb{Z} \text{ and } 0 \le c \le m-1\}$, then $\mathbf{E}[\tau] = O(\sqrt{n})$.
   (E) The hash family described in (C) is a universal hash family.

12. (5 points) Let $G = (V, E, w)$ be an undirected, weighted, simple, and connected graph. Let T_1 and T_2 be two minimum spanning trees of G. Define the set of edges $F \subseteq E$ as all edges that are in either T_1 or T_2 but not in both. Suppose that $F \neq \emptyset$. Select all correct statement(s) below.

   (A) The subgraph $H = (V, F)$ is a disjoint union of cycles and paths.
   (B) All edges in F have the same weight.
   (C) There is no edge in F that has a unique weight.
   (D) There exists a graph G such that $w(F) > w(T_1)$.
   (E) Consider the problem MOST-DISTANT-MSTs, which seeks two minimum spanning trees T_1 and T_2 in order to maximize $w(F)$ on G. Assume P $\neq$ NP. Then MOST-DISTANT-MSTs is NP-hard.

接次頁

13. (5 points) Consider the (incomplete) pseudocode snippet SOLVE().

```
function SOLVE($G = (V, E, w)$, $s$, $t$)
    $P = V$                                              ▷ Initializes a set with all vertices.
    Set $A[s] = 0$ and $A[x] = \infty$ for all $x \neq s$.   ▷ Initializes an array A.
    while $P \neq \emptyset$ do
        $x = \text{argmin}_{x \in P}\{A[x]\}$                 ▷ Chooses a vertex.
        $P = P - \{x\}$                                  ▷ Removes the vertex.
        for each $y \in P$ such that $(x, y) \in E$, do
            $\boxed{(\star)}$
    return $\boxed{(\star\star)}$
```

You have realized that this code snippet can be used to solve different problems by filling the blanks in different ways. Here are three problems we consider:

- In the ST-DISTANCE problem, you are given a graph G with non-negative weights, and two vertices $s, t \in V$. The goal is to compute the shortest distance between s and t.
- In the ST-BOTTLENECK problem, you are given a graph G with non-negative weights, and two vertices $s, t \in V$. The goal is to compute the smallest possible weight *wgt* such that there exists at least one path from s to t using only edges whose weights are less than or equal to *wgt*.
- In the MST problem, you are given an undirected, weighted, and connected graph G. The goal is to compute the total weight of a minimum spanning tree.

Select all correct statement(s) below.

(A) To solve ST-DISTANCE, we fill $(\star)$ with $A[y] = A[x] + w(x, y)$, and we fill $(\star\star)$ with $A[t]$.

(B) To solve ST-BOTTLENECK, we fill $(\star)$ with $A[y] = \max\{A[y], \min\{A[x], w(x, y)\}\}$, and we fill $(\star\star)$ with $A[t]$.

(C) To solve MST, we fill $(\star)$ with $A[y] = \min\{A[y], w(x, y)\}$, and we fill $(\star\star)$ with $A[t]$.

(D) The code snippet can be implemented with Fibonacci heaps, so there exist algorithms with running time $O(|E| + |V| \log |V|)$ for all three problems stated above.

(E) The value $A[t]$ will always be non-negative in any correct implementation for solving any of the problems stated above.

14. (5 points) Let G be an $n \times n$ grid graph, with each undirected edge having a certain weight. Specifically, the vertices of G can be labeled as (i, j) where $1 \leq i, j \leq n$. There is an edge between (i, j) and (k, ℓ) if and only if $|i - k| + |j - \ell| = 1$. Suppose we want to compute the *value* of a minimum cut that separates $(1, 1)$ and (n, n). The value of a cut is defined as the sum of weights to all edges crossing the cut.

Four students proposed different ideas for solving this task. Let us examine their approaches and select all correct statement(s) below.

(A) Student A: Some minimum cut must cut through at most $2n$ edges. Since there are $2n(n-1) \leq 2n^2$ edges in G, we can design a brute force algorithm that enumerates at most $\sum_{k=1}^{2n} \binom{2n^2}{k}$ subsets of edges and check: if the removal of those edges separates $(1, 1)$ and (n, n). Any such subset leads to a candidate for the minimum cut.

(B) Student B: For some minimum cut, the crossing edges must form a monotonic path from $(n, 1)$ to $(1, n)$. In other words, those edges form a path of exactly $2n - 2$ edges from $(n, 1)$ to $(1, n)$. Thus, we can solve this problem via dynamic programming.

(C) Student C: Why don't we simply computes a maximum flow and then turning it to a minimum cut? First, construct a flow network by letting $(1, 1)$ be the source and (n, n) be the sink. For each undirected edge connecting (i, j) and (k, ℓ), create two directed edges in opposite directions, both

見背面

with capacities equal to that edge's weight. By using Dinitz's (Dinic's) algorithm, the maximum flow from $(1, 1)$ to (n, n) can be computed in $O(n^6)$. Using the Max-Flow Min-Cut Theorem, the returned maximum flow value equals the value of the minimum cut.

(D) Student D: To separate $(1, 1)$ and (n, n), for some minimum cut, the crossing edges must form a shortest path from the south-west border $WS := \{(i, 1) \mid 1 < i \leq n\} \cup \{(n, j) \mid 1 \leq j < n\}$ to the north-east border $NE := \{(i, n) \mid 1 \leq i < n\} \cup \{(1, j) \mid 1 < j \leq n\}$. Thus, we can run the Bellman-Ford algorithm that seeks for any shortest path from a vertex in WS to any vertex in NE. Finally, the length of the shortest path is the value of the minimum cut.

(E) None of the above choices.

15. (5 points) In this problem we use the following notations. For any binary tree T, we define $T.x$ to be its root node, $T.l$ (resp. $T.r$) to be the left (resp. right) child of $T.r$, and $T.L$ (resp. $T.R$) to be the left (resp. right) subtree of $T.x$. For any vertex x on a binary tree, we define $x.l, x.L, x.r$, and $x.R$ similarly. These values are defined as $\perp$ (empty) if they do not exist. The set of leaves of T are denoted as Leaf(T). For any statement P we define $[\![P]\!]$ to be 1 if P is true and 0 if P is false.

Let T be a binary tree of n nodes. Each node v on T has a weight $w(v)$ and a color $c(v)$. We want to make the tree colorful in the sense that every non-root tree node has a color different from its parent. To do so, some re-coloring to the tree nodes are necessary, and we would like to minimize the total weights of the tree nodes that are re-colored. Notice that there are infinitely many colors to choose so once a node is re-colored it gets a unique color. Let $f(T)$ be the answer. Suppose that both $T.L$ and $T.R$ are non-empty. What is the correct recursive formulations for $f(T)$? Select all choice(s) that apply.

(A) Let $T' \subseteq T$ be the (partial) binary tree consisting of all tree node v who has the same color as the tree root $T.x$ and all v's ancestors have the same color $c(T.x)$ too. Then, $f(T) = g(T') + \sum_{a \in \text{Leaf}(T')}(f(a.L) + f(a.R))$, here $g(T')$ is the weight of the maximum weighted independent set on T'.

(B) With the same notion T' defined above, $f(T) = h(T') + \sum_{a \in \text{Leaf}(T')}(f(a.L) + f(a.R))$, here $h(T')$ is the weight of the minimum weighted vertex cover on T'.

(C) With the same notion T' defined above, $f(T) = h(T') + \sum_{a \in \text{Leaf}(T')}(f(a.L) + f(a.R))$, here $h(T')$ is the weight of the minimum weighted dominating set on T'.

(D) $f(T) = \min\{f(T.L) + f(T.R) + w(T.x), [\![c(T.l){=}{=}c(T.x)]\!] \cdot (w(T.l) + f(T.L)) + [\![c(T.l) \neq c(T.x)]\!] \cdot f(T.L) + [\![c(T.r){=}{=}c(T.x)]\!] \cdot (w(T.r) + f(T.R)) + [\![c(T.r) \neq c(T.x)]\!] \cdot f(T.R)\}$

(E) None of the above choices apply.

16. (5 points) Consider the following 0-1 integer linear program P:

$$\begin{aligned}
\text{maximize} \quad & 2 \cdot x_1 + 0 \cdot x_2 + 2 \cdot x_3 + 5 \cdot x_4 \\
\text{subject to} \quad & 2x_1 + x_2 + 3x_3 + 4x_4 \leq 8 \\
& 4x_1 + x_2 + 2x_3 + 3x_4 \leq 8 \\
& x_1 + 8x_2 + x_3 + 7x_4 \leq 8 \\
& x_1, x_2, x_3, x_4 \in \{0, 1\}
\end{aligned}$$

Let α be the number of feasible solutions to P. Let β be the number of optimal solutions to P. Let γ be the integrality gap. Let P' be the relaxed linear program of P. Let ν_4 be the four-dimensional volume of the convex polytope that consists of all feasible solutions to P'. Let ν_3 be the three-dimensional volume of the convex polyhedron that consists of all optimal solutions to P'. Select all correct statement(s) below.

(A) $\alpha = 8$ (B) $\beta \neq 1$ (C) $\gamma = 1$ (D) $\nu_4 < 1$ (E) $\nu_3 = 0$

17. (5 points) Let A, B, C, D, E be decision problems. We denote a polynomial time reduction with the symbol "$\leq_p$". Suppose that $A \leq_p B$, $B \leq_p C$, $C \leq_p D$, and $C \leq_p E$. We also assume $A \in$ P and $B \in$ NP. Select all correct statement(s) below.

接次頁

(A) If A is NP-hard, then $B \in$ P.

(B) If B is NP-hard and $E \leq_p B$, then C is NP-complete and D is NP-hard.

(C) If C is NP-complete, then both D and E are NP-complete.

(D) If $D \in$ NP, then $C \in$ NP.

(E) If E is NP-hard, then C is NP-hard.

18. (5 points) Consider the task of incrementing a k-bit binary counter from all zeroes to all ones. Let A be an array of length k that simulates the counter, where $A[0]$ represents the least significant bit and $A[k-1]$ represents the most significant bit. We implement the INCREMENT() operation as follows:

```
function INCREMENT()
    $i = 0$
    while $A[i] == 1$ do
        $A[i] = 0$
        $i = i + 1$
    $A[i] = 1$
```

Unfortunately, this counter is implemented on a deficient random access machine (RAM). On this machine, each read-access to a single array entry takes 1 unit of time. However, a write to an entry may take different amount of time. In particular, for any index i $(0 \leq i < k)$, if i is a multiple of 3 but not a multiple of 5, the memory stick will fizz and it will take exactly $3^{i/3}$ units of time to write the data into $A[i]$. If i is a multiple of 5 but not a multiple of 3, the memory stick will buzz and it will take exactly $5^{i/5}$ units of time to write the data into $A[i]$. Writing into all other entries not mentioned above takes 1 unit of time as usual.

Please design a potential ~~function $\Phi(A)$~~ such that the amortized time (using $\Phi(A)$) per INCREMENT() operation is still $O(1)$ when incrementing from zero. Make sure your potential function is rigurously defined, can be evaluated into an actual number by human beings, and as clean and clear without ambiguity. For example, if you left any undetermined variable (such as "for some constant c") in your expression, you **must** state an actual number for that variable (e.g., "$c = 1$"). You do **not** need to verify your answer in the answer sheet. This problem has no partial score.

19. (10 points) Consider the following decision problem ODDNUMBERSUM:

> ODDNUMBERSUM
> **Input:** A set of positive odd numbers S. A target odd number k.
> **Question:** Is there an odd number of elements in S whose sum is exactly k?

(a) (5 points) Prove that ODDNUMBERSUM is NP-complete.

(b) (5 points) For any set of numbers T we define sum(T) to be the sum of all numbers in T. Let S and k be the input from ODDNUMBERSUM. Define $k^* = \max\{\text{sum}(T) \mid T \subseteq S \text{ and } \text{sum}(T) \leq k\}$ be the largest possible subset sum among all subsets whose sum does not exceed k.

Let $\varepsilon > 0$ be any sufficiently small constant. Give an algorithm that computes an $(1-\varepsilon)$-approximate solution to k^*. Specifically, your algorithm should output a value k_{ALG} such that $(1-\varepsilon)k^* \leq k_{\text{ALG}} \leq k^*$. The running time of your algorithm should be a polynomial of $|S|$ and $1/\varepsilon$. State the algorithm as concisely and percisely as possible. You should **not** write anything besides the algorithm.