

Use one answer sheet for each of F1-1, F1-2, F2-1, and F2-2.

**Q.1** We consider a matrix

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 \\ 2 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix}.$$

- (1) Derive the inverse matrix of  $\mathbf{A}$ .
- (2) Derive all the eigenvalues of  $\mathbf{A}$ .
- (3) Derive  $\mathbf{A}^{10}$ .

**Q.2** We consider a column vector

$$\mathbf{x} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{pmatrix}$$

and a matrix  $\mathbf{B}$  that has ten rows and ten columns.  $\mathbf{B}(i, j)$  represents the  $i$ -th row and the  $j$ -th column element of  $\mathbf{B}$ . All the elements of  $\mathbf{B}$  are zero except

- $$\begin{aligned} \mathbf{B}(1, 6) &= 8, \\ \mathbf{B}(3, 7) &= 2, \\ \mathbf{B}(4, 8) &= 1/8, \\ \mathbf{B}(6, 4) &= 5, \\ \mathbf{B}(7, 1) &= 1/4, \\ \mathbf{B}(8, 10) &= 4, \\ \mathbf{B}(10, 3) &= 1/10. \end{aligned}$$

Derive  $\mathbf{B}^{50}\mathbf{x}$ .

(1) Prove  $\log(xy) = \log x + \log y$  ( $x, y > 1$ ) using the following definition of the natural logarithm

$$\log x = \int_1^x \frac{1}{t} dt.$$

(2) Let  $f(x) = a^x$  ( $a > 0$ ). Derive

$$\frac{df(x)}{dx},$$

using the chain rule.

**Q.2** Let

$$f(p_1, p_2, \dots, p_n) = - \sum_{i=1}^n p_i \log p_i,$$

where

$$\sum_{i=1}^n p_i = 1, \quad 0 < p_i < 1, \quad n \geq 2,$$

and  $\log x$  denotes the natural logarithm of  $x$ .

(1) Prove that  $f(p_1, p_2, \dots, p_n)$  is strictly concave and non-negative.

(2) Derive  $p_i$  ( $i = 1, \dots, n$ ) that maximize  $f(p_1, p_2, \dots, p_n)$  using the method of Lagrange multipliers, and give the maximum value of  $f(p_1, p_2, \dots, p_n)$ .

Use one answer sheet for each of F1-1, F1-2, F2-1, and F2-2.

**Q.1** The following function `func` returns the largest value of  $A[j] - A[i]$  ( $i \leq j$ ) given an integer array  $A$  whose size is  $n$ .

```
int func(int A[], int n){
    int d=0;

    for(int i=0; i<n; i++)
        for(int j=i; j<n; j++)
            if(A[j] - A[i] > d)
                d = A[j] - A[i];
    return d;
}
```

Answer the following questions.

- (1) Describe the time complexity of the algorithm with reasons.
- (2) Is there any algorithm that has better time complexity than `func`? If so, describe its code and its time complexity.

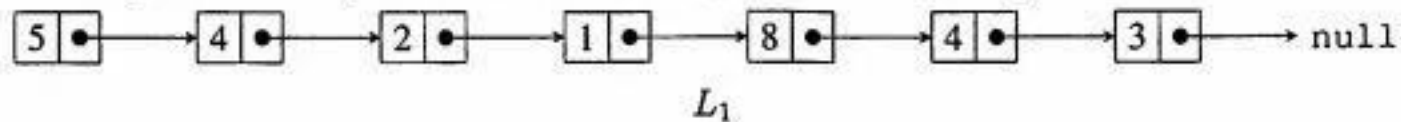
**Q.2** A maximum value contiguous subsequence (MVCS) is a contiguous subsequence of a sequence of integers for which the sum of the elements is the maximum for all possible contiguous subsequences. For example, the MVCS of a sequence  $[-5, 3, 7, -4]$  is  $[3, 7]$  and the sum is 10, and that of  $[-5, 3, 7, -4, 5, 3, -20]$  is  $[3, 7, -4, 5, 3]$  and the sum is 14. Show an efficient algorithm to compute the sum of the elements of an MVCS of a sequence  $A$  whose length is  $n$ , and answer its time complexity.

Use one answer sheet for each of F1-1, F1-2, F2-1, and F2-2.

We consider a data structure called a linked list, which is a chain of nodes. Each node contains two items: a value and a reference to the next node. Answer the following questions about linked lists.

**Q.1** Discuss two advantages of linked lists over arrays regarding computational efficiency.

**Q.2** Given a linked list  $L$  and a threshold value  $t$ , we want to reorder  $L$  such that all nodes with values greater than  $t$  come before nodes with values less than or equal to  $t$ , while preserving the relative order of the nodes in  $L$  as much as possible. We call this operation *partition*. Given the following linked list  $L_1$  and a threshold value  $t_1 = 4$ , show  $L_1$  after *partition*.



**Q.3** Now we consider an algorithm for the linked list *partition* operation introduced in Q.2. Let us define a data structure for a node with two items:

value: an int value, and

next: a reference to the next node or null for the last node.

We define a new variable type Node as a reference to this data structure. newNode() is a predefined function that returns a variable of type Node, which is a reference to a node with its value and next initialized with 0 and null, respectively. := denotes reference assignment. Algorithm 1 is a pseudo-code of this algorithm. Fill the blanks (a) – (g).

---

**Algorithm 1**

**Input:** The head node head (type Node) of a linked list, and a threshold value  $t$  (type int);

**Output:** The head node of the input linked list after partition;

```
1: before_head := newNode(); after_head := newNode();
2: before := before_head; after := after_head;
3: while head is not null do
4:   if head.value > t then
5:     before.next := (a); before := (b);
6:   else
7:     after.next := (c); after := (d);
8:   end if
9:   head := head.next;
10: end while
11: after.next := (e); before.next := (f);
12: Output (g);
```

---

**Q.4** Let  $n$  be the number of nodes in a linked list with the input head node of Algorithm 1. Show the time complexity of Algorithm 1 with reasons.