Academic Year of 2024
Admission to the Master's Program
Intelligence Science and Technology Course
Graduate School of Informatics, Kyoto University
(Fundamentals of Informatics)
(International Program)

10:00 - 12:00, February 7, 2024

NOTES

1. This is the Question Booklet in 7 pages including this front cover.
2. Do not open the booklet until you are instructed to start.
3. After start, check the number of pages and notify proctors (professors) immediately if you find missing pages or unclear printings.
4. This booklet has 4 questions written in English. **Answer all questions.**
5. Write your answers in English, unless specified otherwise.
6. Read the notes on the Answer Sheets as well.

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

Q.1 Consider simultaneous linear equations given by

$$\begin{cases} \lambda x_1 + 3x_3 = 0, \\ x_1 + (1-\lambda)x_2 + 2x_3 = 0, \\ 2x_1 + (5-\lambda)x_3 = 0. \end{cases}$$

Find all values of λ such that there is a solution except for $x_1 = x_2 = x_3 = 0$.

Q.2 Consider a quadratic equation given by

$$5x^2 - 4xy + 8y^2 = 1.$$

Draw the ellipse represented by this equation in the xy-plane. The semi-major and semi-minor axes of the ellipse must be specified.

Q.3 Consider an inner product space $\mathbb{R}^3$, where we have two vectors given by

$$\mathrm{x}_1 = \begin{bmatrix} 3 \\ 0 \\ 4 \end{bmatrix} \quad \text{and} \quad \mathrm{x}_2 = \begin{bmatrix} 4 \\ 5 \\ -3 \end{bmatrix}.$$

Let W be a subspace of $\mathbb{R}^3$ spanned by x_1 and x_2.

(1) Compute an orthonormal basis $\{\mathrm{v}_1, \mathrm{v}_2\}$ of W.

(2) Compute $\mathrm{v}_3 \in \mathbb{R}^3$ such that $\{\mathrm{v}_1, \mathrm{v}_2, \mathrm{v}_3\}$ forms an orthonormal basis of $\mathbb{R}^3$.

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

Q.1 Consider the following functions $f_1(x)$ and $f_2(x)$.

$$f_1(x) = \begin{cases} x\sin\left(\frac{1}{x}\right) & (x \neq 0) \\ 0 & (x = 0) \end{cases}$$

$$f_2(x) = \begin{cases} x^2\sin\left(\frac{1}{x}\right) & (x \neq 0) \\ 0 & (x = 0) \end{cases}$$

(1) Evaluate whether $f_1(x)$ and $f_2(x)$ are differentiable, respectively.

(2) Evaluate whether their derivatives $f_1{}'(x)$ and $f_2{}'(x)$ are continuous at $x = 0$, respectively.

Q.2 Evaluate the directional differentiability and continuity of the following function.

$$f(x,y) = \begin{cases} \dfrac{xy^2}{x^2 + y^4} & (\text{Either } x \neq 0 \text{ or } y \neq 0) \\ 0 & (\text{Both } x = 0 \text{ and } y = 0) \end{cases}$$

Q.3 Compute the volume in the xyz-space for each of the following conditions.

(1) Under the surface $z = e^x \cos(y)$ and over the rectangle defined by $D = [0,1] \times \left[0, \frac{\pi}{2}\right]$ on the xy-plane.

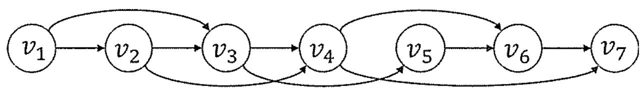
(2) Under the surface of the paraboloid $z = 1 - x^2 - y^2$ and over the xy-plane.

Master's Program	Fundamentals of Informatics	【Algorithms and Data Structures】	Question Number	F2−1

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

Q.1 We have the following weighted directed acyclic graph (DAG) G_1 that has seven nodes named $v_1, v_2, \dots, v_7$. The weight of all edges is equal to 1.

G_1



(1) List all of the shortest paths from v_1 to v_7.

(2) Answer the number of directed paths from v_1 to v_7.

Suppose we have a simple connected weighted DAG $G = (V, E)$ with a set of nodes V and a set of edges E. The weight of all edges is equal to 1. For any node $v \in V$, let $N(v) = \{v' \mid (v', v) \in E\}$. Note that (v', v) indicates an edge from v' to v. We suppose $v_s \in V$ is the only node whose $N(v_s)$ is an empty set.

(3) For any node $v \in V - \{v_s\}$, let $s(v)$ denote the number of paths from v_s to v, and let $s(v_s) = 1$. Express $s(v)$ using elements of $\{s(v')\}_{v' \in V - \{v\}}$.

(4) For any node $v \in V - \{v_s\}$, let $d(v)$ denote the length of the shortest path from v_s to v, and let $d(v_s) = 0$. Express $d(v)$ using elements of $\{d(v')\}_{v' \in V - \{v\}}$.

Q.2 Let $\mathbb{N}$ be the set of all non-negative integers and $\mathbb{N}^3 = \{(i, j, k) \mid i, j, k \in \mathbb{N}\}$. We define a total order $\preccurlyeq$ of two elements in $\mathbb{N}^3$ as $(i, j, k) \preccurlyeq (s, t, u)$ if and only if $(i - s, j - t, k - u) = (0, 0, 0)$ or the rightmost non-zero component of $(i - s, j - t, k - u)$ is negative. For example, $(4, 2, 3) \preccurlyeq (5, 2, 3)$, $(4, 2, 3) \preccurlyeq (5, 4, 3)$, and $(4, 2, 3) \preccurlyeq (2, 3, 3)$.

For a given $N \in \mathbb{N}$, consider the task of listing all elements in $S_N = \{(i, j, i^2 + j^2) \in \mathbb{N}^3 \mid 0 \le i \le N, 0 \le j \le N\}$ following the order $\preccurlyeq$, that is, listing all elements in S_N as

$$(0,0,0), (1,0,1), (0,1,1), \dots, (i_n, j_n, {i_n}^2 + {j_n}^2), (i_{n+1}, j_{n+1}, {i_{n+1}}^2 + {j_{n+1}}^2), \dots, (N, N, 2N^2)$$

so that $(i_n, j_n, {i_n}^2 + {j_n}^2) \preccurlyeq (i_{n+1}, j_{n+1}, {i_{n+1}}^2 + {j_{n+1}}^2)$ holds for all $n = 1, 2, \dots, (N+1)^2$.

Both Algorithm 1 and Algorithm 2 on the next page are for accomplishing the task with a min-heap H for keeping elements in S_N, where "min" means minimum in the order $\preccurlyeq$. Note that "Insert A into H" means to insert A into H as its root and to maintain H so that it keeps the heap property. Also, "Extract A from H" means to remove A from H and to maintain H so that it keeps the heap property.

(continued on the next page)

(1) Draw the heap H of Algorithm 1 as a *tree*, not an array, obtained after executing the part ① for the case $N = 2$. Also illustrate step by step how H is maintained in the first repetition of the while loop.

(2) Fill [ ] in Algorithm 2 so that the size of H is no more than $N + 1$ in the while loop.

(3) For the case $N = 2$, illustrate step by step how H is maintained in the first and second repetitions of the while loop in Algorithm 2.

(4) Let $N \geq 2$. Explain the reason why Algorithm 2 with your answer for (2) works as requested.

Algorithm 1

Let the heap H be empty;
for every j from 0 to N **do**
  for every i from 0 to N **do** ①
    Insert $(i, j, i^2 + j^2)$ into H;
while H is not empty **do**
  Extract the root element from H, and output it;
end while

Algorithm 2

Let the heap H be empty;
for every j from 0 to N **do**
  Insert $(0, j, j^2)$ into H;
while H is not empty **do**
  Extract the root element from H as $(i', j', (i')^2 + (j')^2)$, and output it;
  if $i' < N$ **then**
    Insert [ ] into H;
end while

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

Q. We represent an array P of length m, whose elements are alphabet characters. We call P a pattern. An element of P can be accessed by $P[i]$, where $1 \le i \le m$ is an index. $P[s : t]$ denotes a contiguous subarray of P starting from index s to t inclusively, where $1 \le s \le t \le m$. P_h denotes the h-character prefix $P[1 : h]$ of P, while P_0 is the empty string ε and $P_m = P = P[1 : m]$. The *prefix function* for a pattern P returns an array π of length m, such that each element with an index $1 \le q \le m$ is computed by

$$\pi[q] = \max\{k \mid k < q \text{ and } P_k \sqsupset P_q\},$$

where $P_k \sqsupset P_q$ denotes that P_k is a suffix of P_q. That is, $\pi[q]$ is the length of the longest prefix of P_q that is also a proper suffix of P_q. Note that a proper suffix cannot be the whole string.

(1) Compute the results of the prefix function $\pi[1], \pi[2], ..., \pi[11]$ for the pattern aabaacaabaa.

(2) Algorithm 1 is a pseudo-code of an algorithm for computing the results of the prefix function π for a pattern P. Fill the blanks (a), (b), and (c).

Algorithm 1 COMPUTE-PREFIX-FUNCTION(P)

```
m = P.length
let π be a new array for keeping the results of the prefix function
π[1] = 0
k = 0
for q = 2 to m do
  while k > 0 and P[k + 1] ≠ P[q] do
    [ (a) ]
  end while
  if P[k + 1] == P[q] then
    [ (b) ]
  end if
  [ (c) ]
end for
return π
```

(continued on the next page)

We represent an array T of length n and an array P of length $m \leq n$. The elements of both T and P are alphabet characters. We call T a text and P a pattern. We say that a pattern P occurs with a shift s in a text T if $0 \leq s \leq n-m$ and $T[s+1 : s+m] == P[1 : m]$ (that is, if $T[s+j]$ $== P[j]$, for $1 \leq j \leq m$). The string-matching problem is the problem of finding all shifts with which a given pattern P occurs in a given text T.

(3) Algorithm 2 is a pseudo-code of an algorithm for the string-matching problem utilizing the results of the prefix function computed with Algorithm 1. Fill the blanks (d), (e), and (f).

Algorithm 2 STRING-MATCHING(T, P)

```
n = T.length
m = P.length
π = COMPUTE-PREFIX-FUNCTION(P)
q = 0
for i = 1 to n do
  while q > 0 and P[q + 1] ≠ T[i] do
    [ (d) ]
  end while
  if P[q + 1] == T[i] then
    [ (e) ]
  end if
  if q == m then
    print "Pattern occurs with shift" i − m
    [ (f) ]
  end if
end for
```

(4) Show the time complexity of Algorithm 2 with reasons.