1

問 1（日本語版）

次の各問いに答えなさい.

(1) $\{a, b\}$ を文字集合とする，長さ 3 の文字列全体の集合を S_3 とする．S_3 の要素をすべて列挙しなさい.

(2) 長さ n の文字列 $c_1c_2\cdots c_n$ に対して，末尾の文字 c_n を文字列の先頭に移動させて$c_nc_1\cdots c_{n-1}$ に変換することを，巡回シフトという．巡回シフトを用いて，文字列からなる集合に，次の関係“~”を定義する.

文字列 $\boldsymbol{s_1}$ と $\boldsymbol{s_2}$ に対して，$\boldsymbol{s_1 \sim s_2}$とは，一方を他方の 0 回以上の巡回シフトによって得られることである.

例えば，(1)で求めた集合 S_3 において，$aba \sim aab, aba \sim baa$である．また，一方の文字列の巡回シフトで他方の文字列が得られないときは，$\boldsymbol{s_1 \nsim s_2}$と書く．例えば，$aba \nsim abb$である.

上で定義された関係“~”が，同値関係の条件を満たしていることを証明しなさい．長さ n の文字列の集合 S_n 上の同値関係とは，任意の $\boldsymbol{s_1, s_2, s_3} \in S_n$ に対して，反射律：$\boldsymbol{s_1 \sim s_1}$，対称律：$\boldsymbol{s_1 \sim s_2 \Longrightarrow s_2 \sim s_1}$，推移律：$\boldsymbol{s_1 \sim s_2}$かつ$\boldsymbol{s_2 \sim s_3 \Longrightarrow s_1 \sim s_3}$の 3 つの条件が全て成り立つことである.

(3) (1)で求めた S_3 を，(2)の同値関係“~”による同値類に分けて，各同値類の要素をすべて列挙しなさい.

(4) $\{a, b\}$ を文字集合とする，長さ $n\,(\geq 1)$ の文字列全体の集合を S_n とする．S_n に対して，(2) で与えた同値関係を定義する．長さ n の文字列の巡回シフトによる同値類の個数を$N(n)$ と表す．$N(1), N(2), N(3), N(4)$ をそれぞれ求めなさい.

(5) (4)の問題文で定義した$N(n)$について，次の不等式を証明しなさい.

$$N(n) \geq \frac{2^n}{n}$$

問 1（English version）

Solve the following problems.

(1) Let S_3 be the set of strings of length 3 over the alphabet $\{a, b\}$. List all element of S_3.

(2) A *circular shift* on a string $\boldsymbol{c} = c_1c_2 \ldots c_n$ of length $n \geq 1$ is an operation that turns $\boldsymbol{c}$ into $c_nc_1 \ldots c_{n-1}$ by moving the last character c_n of $\boldsymbol{c}$ to its first position while shifting all other characters to the right by one position. Circular shifts naturally induce a binary relation $\sim$ between strings, where for any pair of strings $\boldsymbol{s_1}, \boldsymbol{s_2}$, the relation $\boldsymbol{s_1} \sim \boldsymbol{s_2}$ holds if and only if one can be obtained by applying a circular shift, possibly multiple times, to the other. We write $\boldsymbol{s_1} \nsim \boldsymbol{s_2}$ to mean that $\boldsymbol{s_1}$ is never reachable by circular shifting $\boldsymbol{s_2}$ and vice versa. For instance, we have $aba \sim aab$ and $aba \sim baa$ while $aba \nsim abb$.

Prove that the binary relation $\sim$ is an equivalence relation on a set of strings of length n by showing that it satisfies all the required conditions for it to be one, namely for any strings $\boldsymbol{s_1}, \boldsymbol{s_2}, \boldsymbol{s_3}$ of length n, it holds that $\boldsymbol{s_1} \sim \boldsymbol{s_1}$ (reflexivity), that $\boldsymbol{s_1} \sim \boldsymbol{s_2} \implies \boldsymbol{s_2} \sim \boldsymbol{s_1}$ (symmetry), and that $\boldsymbol{s_1} \sim \boldsymbol{s_2}$ and $\boldsymbol{s_2} \sim \boldsymbol{s_3} \implies \boldsymbol{s_1} \sim \boldsymbol{s_3}$ (transitivity).

(3) Explicitly write down the equivalence classes which are induced by $\sim$ and partition S_3 given in (1).

(4) Let S_n be the set of strings of length $n \geq 1$ over $\{a, b\}$ and denote by $N(n)$ the number of equivalence classes induced by $\sim$ on S_n. Find $N(1)$, $N(2)$, $N(3)$, and $N(4)$.

(5) Prove the following inequality

$$N(n) \geq \frac{2^n}{n},$$

where $N(n)$ is the number of equivalence classes as defined in (4).

問 2（日本語版）

要素数が 4 の集合だけを要素とする集合を 4−一様集合族といい，集合族 $\mathcal{F} = \{S_0, \dots, S_{n-1}\}$ の要素の和集合

$$S = \bigcup_{i=0}^{n-1} S_i$$

を $\mathcal{F}$ の台集合と呼ぶ．例えば，集合族

$$\mathcal{A} = \{\{1,2,3,4\},\{3,4,5,6\},\{1,3,5,7\}\}$$

は $\{1,2,3,4\}$，$\{3,4,5,6\}$，$\{1,3,5,7\}$ の 3 つの集合を要素とするが，それぞれ 4 つの要素からなるため $\mathcal{A}$ は 4−一様集合族であり，$\mathcal{A}$ の台集合は $\{1,2,3,4,5,6,7\}$ である．

$\mathcal{F} = \{S_0, \dots, S_{n-1}\}$ を n 個の集合からなる 4−一様集合族とし，その台集合 S の要素を赤と青の 2 色で塗り分けることを考える．ここで $\mathcal{F}$ に含まれる集合 S_i の要素が全て同一の色で塗られるとき，S_i は単色であるという．例えば前述の $\mathcal{A}$ の台集合 $\{1,2,3,4,5,6,7\}$ の要素を，それぞれ，$1,2,7$を赤，$3,4,5,6$を青に彩色すると，$\mathcal{A}$ の要素 $\{1,2,3,4\}$ と $\{1,3,5,7\}$ はどちらも単色ではないが，$\{3,4,5,6\}$ は単色となる．

今，$\mathcal{F}$ に含まれるどの集合も赤の要素と青の要素を両方少なくとも 1 つは含むよう，つまりどの S_i も単色ではないように台集合の要素を彩色したい．以下の問いに答えなさい．

(1) $\mathcal{F}$ の台集合 S の要素をそれぞれ独立かつ均等確率で無作為に赤と青で塗り分ける．つまり S のどの要素も独立に，赤となる確率と青となる確率が等しく $\frac{1}{2}$ となる．このとき，$\mathcal{F}$ のとある要素 S_i に含まれる 4 つの要素が全て赤となる確率を求めなさい．また S_i が単色となる確率を求めなさい．

(2) 前問と同様，$\mathcal{F}$ の台集合 S の要素をそれぞれ独立かつ均等確率で無作為に赤と青で塗り分ける．$\mathcal{F}$の要素 $S_0, \dots, S_{n-1}$ のうち，単色である集合の個数について，その期待値を求めなさい．

(3) $\mathcal{F}$ の要素数 n が 7 以下であるならば，どの S_i も単色でないように，台集合 S の要素を 2 色で塗り分けられることを証明しなさい．

(4) k を 2 以上の自然数とする．要素数が k の集合だけを要素とする集合を k−一様集合族という．4−一様集合族と同様，その台集合の各要素を 2 色で塗り分けたとき，集合族の要素のうち，その要素が全て同一の色で塗られたものを単色であるという．任意の k−一様集合族は，その要素数が 2^{k-1} 未満であるならば，どの要素も単色でないように，その台集合の要素を 2 色で塗り分けられることを証明しなさい．

問 2（English version）

A family $\mathcal{F} = \{S_0, \ldots, S_{n-1}\}$ of finite sets with the *ground set*

$$S = \bigcup_{i=0}^{n-1} S_i$$

is 4-*uniform* if for any $S_i \in \mathcal{F}$, it holds that $|S_i| = 4$. For instance, the following family

$$\mathcal{A} = \{\{1, 2, 3, 4\}, \{3, 4, 5, 6\}, \{1, 3, 5, 7\}\}$$

is 4-uniform because each of its elements $\{1, 2, 3, 4\}$, $\{3, 4, 5, 6\}$, $\{1, 3, 5, 7\}$ contains exactly 4 elements, while its ground set is their union $\{1, 2, 3, 4, 5, 6, 7\}$.

Let $\mathcal{F} = \{S_0, \ldots, S_{n-1}\}$ be a 4-uniform family that consists of n sets. Consider coloring each element of its ground set S blue or red. We say that a set $S_i \in \mathcal{F}$ is *monochromatic* if every element in S_i is of the same color. As an example, coloring the ground set $\{1, 2, 3, 4, 5, 6, 7\}$ of the family $\mathcal{A}$ given above by coloring 1, 2, and 7 red and 3, 4, 5, and 6 blue makes the element $\{3, 4, 5, 6\} \in \mathcal{A}$ monochromatic, while the other two in $\mathcal{A}$ are mixed colored under this coloring. We investigate whether it is possible to color the ground set S of $\mathcal{F}$ in such a way that none of $S_0, \ldots, S_{n-1}$ is monochromatic.

(1) Color each element of the ground set S of a 4-uniform family $\mathcal{F} = \{S_0, \ldots, S_{n-1}\}$ red or blue independently and uniformly at random, so that for any $s \in S$, its color is chosen independently with $Pr(s \text{ is red}) = Pr(s \text{ is blue}) = \frac{1}{2}$. What is the probability that all four elements of a given set $S_i \in \mathcal{F}$ are colored red? Under the same random coloring, what is the probability that a given set $S_i \in \mathcal{F}$ is monochromatic?

(2) Randomly color S as in the previous question. What is the expected number of monochromatic sets in $\mathcal{F}$?

(3) Show that if the number n of sets in $\mathcal{F}$ is at most 7, then it is possible to color S with two colors such that none of $S_0, \ldots, S_{n-1}$ is monochromatic.

(4) Let $k \geq 2$ be a natural number. A family is k-*uniform* if each of its elements consists of exactly k elements. Consider coloring the ground set of a k-uniform family with two colors. As before, a set is *monochromatic* if its elements are all of the same color. Show that if the number of sets in a k-uniform family is less than 2^{k-1}, it is possible to color the ground set with two colors in such a way that no elements in the family are monochromatic.

2

問 1（日本語版）

出力が現在状態と入力によって決定される有限オートマトンであるミーリー・マシンは，タプル $M = (S, I, O, \delta, \lambda, \sigma_0)$で記述できる．ここで，$S$は状態の集合，$I$は入力アルファベット，$O$は出力アルファベットである．また，$\delta$は遷移関数であり，$\delta: S \times I \to S$である．$\lambda$は出力関数であり，$\lambda: S \times I \to O$ である．$\sigma_0 \in S$は開始状態である．以下の問に答えなさい．

(1) 入力アルファベットI，および，出力アルファベットOは，いずれも 2 値信号$\{0, 1\}$ とする．入力系列 101 を検出し，検出した際に1を，それ以外は0を出力する検出器を実現する状態数が 4 のミーリー・マシンを考える．状態集合$S = \{S_0, S_1, S_2, S_3\}$とする．開始状態$\sigma_0$は任意とする．この検出器の入力系列が0101010 の場合の出力系列は0X01010となる．開始状態が任意であるため一意に定まらないビットの値はXとしている．また，次の(a)の解答でもそのようにしなさい．

(a) 入力系列が001011，1010101の場合の出力系列をそれぞれ答えなさい．
(b) 検出器を実現する遷移関数δに対応する，解答欄に示された状態遷移表を完成させなさい．
(c) 解答欄に示された，検出器を実現するミーリー型状態遷移図を完成させなさい．

(2) 以下では，検出器を 2 つのマスタースレーブ型 JK-フリップフロップF_0，F_1 と組合せ回路を利用して実現することを考える．F_0の入力J，Kおよび出力Qをそれぞれ J_0，K_0，Q_0とし，F_1の入力J，Kおよび出力Qをそれぞれ J_1，K_1，Q_1とする．

検出器の入力をxとし，出力をyとする．各 1 ビットのm_0，m_1を利用して$i = 2 \times m_1 + m_0$とし，現在状態をS_iによって表す．同様に各 1 ビットのm_0'，m_1'を利用して$j = 2 \times m_1' + m_0'$とし，遷移後状態をS_j'によって表す．F_0でm_0から m_0' への遷移を，F_1でm_1からm_1' への遷移を表すとする．

JK-フリップフロップの特性方程式を次に示す．Q_t，Q_{t+1}はそれぞれ，Qの現在状態，遷移後状態を表す．

$$Q_{t+1} = J\overline{Q_t} + \bar{K}Q_t$$

(a) 解答欄に示された JK-フリップフロップの励起表を完成させなさい．値が0と1のどちらでもよい場合には Xを記入しなさい．
(b) JK-フリップフロップ F_0，F_1を利用した検出器の論理回路に対応する，解答欄に示された状態遷移表を完成させなさい．値が0と1のどちらでもよい場合には Xを記入しなさい．

次に，現在状態と入力からJ_1を構成する論理回路を簡単化する．

(c) 解答欄に示されたJ_1 のカルノー図を完成させなさい．値が0と1のどちらでもよい場合には Xを記入しなさい．
(d) J_1を最小限の項とリテラルで構成される論理式で表しなさい．

問 1 (English version)

A *Mealy Machine*, which is a finite automaton whose output is determined by the current state and the input, can be described as a tuple $M = (S, I, O, \delta, \lambda, \sigma_0)$. Here, S is the *set of states*, I is the *input alphabet*, and O is the *output alphabet*. $\delta: S \times I \to S$ is the *transition function*. $\lambda: S \times I \to O$ is the *output function*. $\sigma_0 \in S$ is the *initial state*. Answer the following questions.

(1) Consider a 4-state Mealy Machine that detects the sequence 101 from a binary input stream. This machine, henceforth called the detector, outputs 1 when the sequence is detected, and 0 otherwise. Let $I = O = \{0, 1\}$ be the input and output alphabets, and let $S = \{S_0, S_1, S_2, S_3\}$ be the set of states. The initial state σ_0 is arbitrary. When the input sequence is 0101010, the corresponding output sequence is $0X01010$, where X represents a bit that cannot be uniquely determined unless the initial state is fixed. Use the same notation in your answer to (a).

(a) Give the output sequences for the input sequences 001011 and 1010101.
(b) Complete the state transition table, which is provided in the answer sheet, for the transition function δ of the detector.
(c) Complete the Mealy-type state transition diagram implementing the detector. The diagram is provided in the answer sheet.

(2) In the following, consider implementing the detector using two master-slave JK flip-flops, F_0 and F_1, along with combinational logic circuits. Let J_0, K_0 and Q_0 be the inputs J, K, and output Q of F_0, respectively. Similarly, let J_1, K_1 and Q_1 be the inputs J, K, and output Q of F_1, respectively.

Let x be the input to the detector, and y be the output. Let S_i be the current state with $i = 2 \times m_1 + m_0$ where m_0 and m_1 are 1-bit variables. Likewise, let S'_j be the next state with $j = 2 \times m'_1 + m'_0$ where m'_0 and m'_1 are 1-bit variables. The transition from m_0 to m'_0 is represented by F_0, and the transition from m_1 to m'_1 is represented by F_1.

The characteristic equation of a JK flip-flop is given below,

$$Q_{t+1} = J\overline{Q_t} + \overline{K}Q_t ,$$

where Q_t and Q_{t+1} are the current and next states of Q, respectively.

(a) Complete the excitation table of the JK flip-flop provided in the answer sheet. Use X for a bit that can be either 0 or 1.
(b) Complete the state transition table, provided in the answer sheet, for the detector implemented using JK flip-flops F_0 and F_1. Use X for a bit that can be either 0 or 1.

Next, we would like to simplify the logic circuit that generates J_1 from the current state and the input.

(c) Complete the Karnaugh map for J_1 provided in the answer sheet. Use X for a bit that can be either 0 or 1.
(d) Express J_1 as a logic expression using the minimal number of terms and literals.

問 2（日本語版）

次の各問に答えなさい.

(1) (A)〜(D)は OSI (Open Systems Interconnection) 参照モデルの一部である．以下の各層について，最も適切な処理を選択肢①の中からそれぞれ一つ選びなさい.

(A) ネットワーク層

(B) データリンク層

(C) 物理層

(D) トランスポート層

選択肢①：

(ア) ビット列を電気信号に変換して送信する

(イ) ネットワークの端から端までの通信管理を行う

(ウ) IP (Internet Protocol) アドレスを利用し 1 つ以上のネットワークを経由して宛先までパケットを転送する

(エ) MAC (Media Access Control) アドレスを利用し物理層を介して隣接ノードにフレームを転送する

(2) CSMA/CD (Carrier Sense Multiple Access with Collision Detection)は， イーサネットで利用されている通信方式の一つであり，通信を監視し回線に空きがあれば通信を開始，仮に衝突が発生した場合にランダムな時間待機した後に再送する方式である．衝突検出には一定時間（往復伝播時間以上）送信が続いている必要がある．電気信号の伝播速度が 2.0×10^8 m/s，ネットワークの長さが 500 m，通信速度が 10 Mbps のとき，CSMA/CD において衝突を検出可能とするために必要な最小フレーム長を bit 単位で求めなさい.

(3) イーサネットスイッチのポート 1 に MAC アドレス A が学習されている．MAC アドレス A を持つ端末から未学習の MAC アドレス B を持つ端末宛にフレームを送信する．MAC アドレス B を持つ端末はスイッチのポート 2 に接続されている．このとき，スイッチはどのように転送するかを，処理の対象を選択肢②から，処理の方法を選択肢③からそれぞれ一つ選びなさい.

選択肢②：

(ア) 全ポート

(イ) ポート 1 を除くすべてのポート

(ウ) ポート 1

(エ) ポート 2

(オ) MAC アドレス A

(カ) MAC アドレス B

選択肢③：

(ア) フレーム送信

(イ) フレーム破棄

(ウ) フレーム生成

(エ) ルーティングテーブルを参照

(4) あるローカルエリアネットワーク (Local Area Network, LAN) について，Internet Protocol Version 4 の IP アドレスおよびサブネットマスクがドット付き 10 進表記でそれぞれ 192.168.123.123, 255.255.255.0 であった．このネットワークを，以下の 2 つの条件を共に満たすようにサブネットに分割する．

条件 1：1 サブネットあたりに利用可能なホストの台数が少なくとも 20 台

条件 2：使用可能なサブネットの数が最大

IP アドレス 192.168.123.123 が属するサブネットのサブネットマスク，ネットワークアドレス，このサブネット内のホストの始点 IP アドレス，終点 IP アドレス，ブロードキャストアドレス，およびサブネット数を求めなさい．

問 2 (English version)

Solve the following questions.

(1) Items (A) through (D) are layers of the OSI (Open Systems Interconnection) reference model. For each item, choose the most appropriate function from the list of options① below, and write on the answer sheet the letter of your answer for each item.

(A) Network Layer

(B) Data link Layer

(C) Physical Layer

(D) Transport Layer

Options①:

(ア) Converts a stream of raw bits into electrical signals and transmits them

(イ) Provides end-to-end communication services

(ウ) Transfers packets to a destination across one or more networks based on IP (Internet Protocol) address

(エ) Transfers frames to a next hop node based on the MAC (Media Access Control) address

(2) CSMA/CD (Carrier Sense Multiple Access with Collision Detection) is a communication method used in Ethernet. A node senses the communication channel and begins transmission when the channel is idle. If a collision occurs, the node waits a random period of time and retransmits the data. In order to detect a collision, the transmission must continue until one complete round-trip propagation time has passed. Given an electrical signal propagation speed of 2.0×10^8 m/s, a network length of 500 m, and a transmission rate of 10 Mbps, calculate the minimum frame length in units of bits required to ensure that terminals can detect collisions in a CSMA/CD network.

(3) Consider the following situation: MAC address A has been learned on port 1 of an Ethernet switch. A node with MAC address A sends a frame to a node with MAC address B, which has not yet been learned by the switch. A node with MAC address B is actually connected to port 2. Then, how does the switch forward the frame? Choose the destination (i.e., target(s)) from Option ②, and the way to forward from Option ③.

Option② :

(ア) All ports

(イ) All ports except for Port 1

(ウ) Port 1

(エ) Port 2

(オ) MAC Address A

(カ) MAC Address B

Option③:

(ア) Transmits the frame

(イ) Drops the frame

(ウ) Generates the frame

(エ) Refers the routing table

(4) Consider a local area network (LAN) whose IPv4 address and subnet mask are 192.168.123.123 and 255.255.255.0, respectively (in dotted decimal notation). We would like to divide the network into subnets that satisfy both of the following conditions:

Condition 1：Each subnet must support at least 20 usable host addresses

Condition 2：The number of available subnets must be as large as possible

Under these conditions, for the subnet to which the IP address 192.168.123.123 belongs, what are the subnet mask, network address, starting IP address, ending IP address, broadcast address, and the number of subnets?

3

問 1（日本語版）

図 1 はある 2 種類の内部ソートアルゴリズムを sortA 関数と sortB 関数として実装した C 言語のプログラムの一部である．ここで，ソートの対象は要素数 n の整数型配列 in で表され，swap は in[i]と in[j]の要素を交換する関数である．要素は昇順にソートされる．以下の問いに答えなさい．

(1) 以下の記述はそれぞれの関数の動作を説明したものである．空欄 A ～ J に当てはまるものを (a) ～ (m) の選択肢の中からひとつ選んで解答欄に記号を記入しなさい．同じ選択肢を複数回選択しても構わない．なお，sortB 関数では配列の先頭からある部分までの要素が昇順に並んでいれば，例えそれらの要素が後で他の要素と交換されるとしても，その部分はソート済であると呼ぶ．

sortA 関数は A の要素の中から B の要素をひとつ取り出す．それを C の要素の D にある要素と配列の位置を交換した上でソート済とみなす．それを全ての要素がソートされるまで繰り返す．

sortB 関数は E の要素の中から F の要素を取り出す．それを G の要素の H にある要素と比較し，自身より I 場合相互に交換する．さらに自身よりひとつ前の要素に対して同様の比較と交換を繰り返し行い，配列の先頭に達するか自身より J 要素が見つかったらはじめのステップに戻り，それを全ての要素がソートされるまで繰り返す．

(a)未ソート (b)ソート済 (c)全部 (d)最小 (e)最大 (f)任意 (g)先頭 (h)末尾
(i)小さい (j)大きい (k)小さいもしくは等しい (l)大きいもしくは等しい (m)等しい

(2) $n = 8$ として配列{4,8,7,2,1,5,6,9}を in に入力し，sortA 関数と sortB 関数を用いてソートさせる．ループ変数 i が i=3 から 4 に増加する時点の in に格納されている数値を，要素番号が小さい順にカンマ区切りで解答欄に記入しなさい．例えば in={0,1,2,3,4,5,6,7}であれば解答欄には 0,1,2,3,4,5,6,7 と記入しなさい．

(3) sortA 関数について，in に格納されている数値同士が比較される回数と，swap 関数が呼び出される回数をそれぞれ n を用いて表しなさい．

(4) 与えられた入力列 in があらかじめ降順に並んでいる，かつ要素に重複がない場合，sortB 関数では 20 行目の while 文は常に in[j]<in[j-1]の条件を満たす．このとき in に格納されている数値同士が比較される回数と，swap 関数が呼び出される回数をそれぞれ n を用いて表しなさい．

(5) swap 関数の K で表される空欄を埋めなさい．複数行になっても構わない．

```
1. void swap( int *in, int i, int j){
2.     K
3. }
4.
5. void sortA( int n, int *in){
6.   int i, j;
7.   for( i=0; i<n; i++){
8.     int k = i;
9.     for( j=i+1; j<n; j++){
10.      if( in[j]<in[k])  k = j;
11.    }
12.    swap( in, i, k);
13.  }
14.}
15.
16.void sortB( int n, int *in){
17.  int i, j;
18.  for( i=0; i<n-1; i++){
19.    int j = i+1;
20.    while( in[j]<in[j-1] && j>0){
21.      swap( in, j, j-1);
22.      j--;
23.    }
24.  }
25.}
```

図１　2 種類の内部ソートアルゴリズムを実装した C 言語のプログラムの一部

問1 (English version)

Figure 1 shows part of a C language program that implements two different in-place sorting algorithms as functions `sortA` and `sortB`. Both functions sort the elements of an integer type array `in` of size n, in ascending order using function `swap`, which exchanges elements of `in[i]` and `in[j]`. Solve the following problems.

(1) The following two paragraphs describe the procedures taken by the pair of sorting functions. Choose the correct answer from items (a) - (m) and fill in the blanks [A]-[J]. Write on the answer sheet the letter of your answer for each blank. You may choose any item more than once if necessary. Note that when describing `sortB`, at any step of sorting, the xth element of `in` is said to be *sorted* if the elements of `in[i]` for $i = 0, \ldots, x$ are in ascending order regardless of whether its position can be swapped later in the sorting procedure.

`sortA` first scans [A] elements to locate [B] element, and then exchanges the located one with the one at [D] element of [C] elements, making it sorted. Repeating this procedure until all the elements are sorted.

`sortB` function takes [F] element of [E] elements and compares it against [H] element of [G] elements. If [H] element is [I], their positions are exchanged and unless it reaches the head of the array, it is then compared against the next one forward. Repeating this compare-and-exchange process until it reaches the head of the array or a [J] element is found. Repeating the whole procedure described above until all the elements are sorted.

(a) the unsorted (b) the sorted (c) all the (d) a minimum (e) a maximum (f) any
(g) the head (h) the tail (i) smaller (j) larger (k) smaller or equal (l) larger or equal
(m) equal

(2) Set $n = 8$ and suppose sorting `in={4,8,7,2,1,5,6,9}` by `sortA` and `sortB`. What is `in` at the point when the loop variable `i` increments from 3 to 4? Write the elements in ascending order of array index from `in[0]` to `in[7]` for each case. Use a comma between elements. For example, if `in={0,1,2,3,4,5,6,7}`, write `0,1,2,3,4,5,6,7` on the answer sheet.

(3) Give the total number of comparisons `sortA` makes between elements of `in` and that of `swap` calls both as a function of n.

(4) If the input array `in` is sorted in descending order and has no identical elements, the first condition `in[j]<in[j-1]` to continue the `while` loop at line 20 of Figure 1 is always met. Give the total number of comparisons `sortB` makes between elements of `in` and that of `swap` calls both as a function of n.

(5) Complete the code given in Figure 1 by filling in the blank [K] in the `swap` function. Multiple lines are allowed if desired.

```
1. void swap( int *in, int i, int j){
2.     K
3. }
4.
5. void sortA( int n, int *in){
6.   int i, j;
7.   for( i=0; i<n; i++){
8.     int k = i;
9.     for( j=i+1; j<n; j++){
10.      if( in[j]<in[k])  k = j;
11.    }
12.    swap( in, i, k);
13.  }
14.}
15.
16.void sortB( int n, int *in){
17.  int i, j;
18.  for( i=0; i<n-1; i++){
19.    int j = i+1;
20.    while( in[j]<in[j-1] && j>0){
21.      swap( in, j, j-1);
22.      j--;
23.    }
24.  }
25.}
```

Figure 1 Part of a C language program that implements two different in-place sorting algorithms

3

問２（日本語版）

長さNのテキストと呼ばれる文字列$T = T_0T_1T_2 \ldots T_{N-1}$の中から長さ$M$のパターンと呼ばれる文字列$P = P_0P_1P_2 \ldots P_{M-1}$ を照合して探し出し，テキスト中のパターンと一致する部分文字列の先頭位置の添え字番号を出力する文字列照合のアルゴリズムについて考える．ただし，テキストの中にパターンと同じ文字列が複数存在する場合は，添え字番号の一番小さいものだけを返し，存在しなければ-1が返されて終了する．ここで，$N \geq M \geq 1$を仮定し，対象とする文字は簡単のためアルファベットの小文字 a から j までの 10 文字に限定する．つまり，T_i，$P_j \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}, \mathtt{h}, \mathtt{i}, \mathtt{j}\}$ $(i = 0, 1, 2, \cdots, N-1,\ j = 0, 1, 2, \cdots, M-1)$とする．例えば，$T$=abcdefghiabcdef，$P$=def とすると，$P_0P_1P_2$と一致するテキスト中の部分文字列は，$T_3T_4T_5$と$T_{12}T_{13}T_{14}$である．パターンとの最初の完全一致は$T_3$から始まっているため，アルゴリズムは 3 を返す．

さて，このような文字列照合のアルゴリズムを一般的なプログラミング言語で実装する際には，文字列を配列で保存することが一般的に考えられる．そこで，長さNの文字列Xを配列$X[0:N-1]$で表し，$X[m]$ $(0 \leq m \leq N-1)$は文字列X中のm番目の文字を表すものとする．また，$X[m:n]$ $(0 \leq m < n \leq N-1)$は文字列Xのm番目の文字からn番目の文字までの部分文字列を表すとする．

図 2 はT=aceafadcadcfcaf，P=cfcaf のとき，テキストの最初の文字（つまり，添え字番号 0 の文字）から始めて，長さMの連続した部分文字列（そのテキストでの位置を比較対象箇所と呼ぶ）とパターンとの照合を順次前から 1 文字ずつ一致しているかを調べ，不一致が起こったら，テキストの比較対象箇所を 1 文字ずつずらしていくアルゴリズム(StringMatch1 とする)の様子をステップ 3 まで示している．ステップ 1 では①で不一致，ステップ 2 では③で不一致が起こり，テキストにおける比較対象箇所を順次 1 文字ずつずらしている．図 3 は StringMatch1 の疑似コードである．このとき次の(1)～(3)の問いに答えなさい．

(1) 空欄 A ～ C に当てはまるものを(a)～(i)の選択肢の中からひとつ選び，疑似コードが上記の StringMatch1 の文字列照合アルゴリズムを表すように完成させなさい．

(a) j>n　(b) j<n　(c) j>m　(d) j<m　(e) n　(f) m　(g) i　(h) j　(i) i+j

(2) StringMatch1 において，T=aciacadabgag，P=abga としたとき，アルゴリズムが終了した時点での変数 sum の値を求めなさい．

(3) StringMatch1 では長さNのテキストと長さMのパターンが与えられたとき，文字の比較の回数は最大何回になるか，NとMを用いて表しなさい．

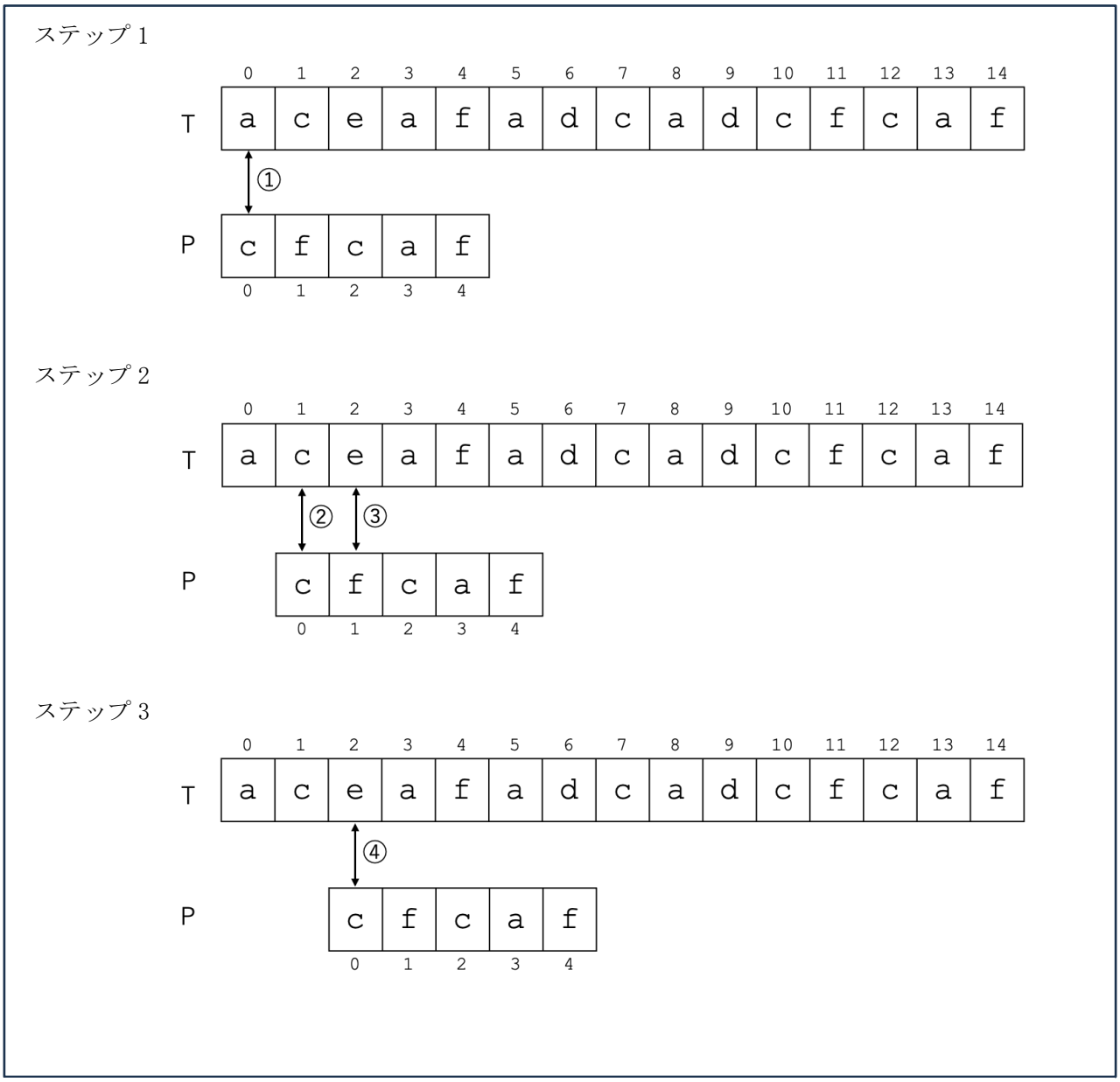


図 2　StringMatch1 の探索の例

```
FUNCTION StringMatch1(T, P):
    // T: テキスト（0 〜 N−1 のインデックス）
    // P: パターン（0 〜 M−1 のインデックス）
    n ← length(T) // length(X)は文字列 X の長さを返す
    m ← length(P)

    // テキストの開始位置 i の照合対象となる部分文字列についてチェックする
    sum ← 0     // 総比較回数
    for i from 0 to n − m do
        j ← 0
        // 前から順に 1 文字ずつパターンと比較
        while [   A   ] do
            sum ← sum + 1
            if T[   B   ] == P[   C   ] then
                j ← j + 1
            else
                break while
            end if
        end while

        // 完全一致したら，その添え字 i を返す
        if j = m then
            return i
        end if
    end for

    // 見つからなければ-1 を返す
    return -1
end FUNCTION
```

図 3　StringMatch1 の疑似コード

StringMatch1 の探索方法では，不一致が起こるたびにテキスト中の比較対象箇所をたった 1 文字しかずらさないため効率が悪い．そこで，より効率的に探索するアルゴリズムを考える(StringMatch2 とする)．StringMatch2 では，StringMatch1 と同様，テキストの先頭からパターンを探索するが，このとき，パターンの後ろから順に 1 文字ずつ比較する．

StringMatch2 では，文字の不一致が起こった場合，テキスト中の現在の比較対象箇所における部分文字列の末尾文字に注目し，その文字がパターンのどこにあるか（または，ないか）によって，テキストの次の比較対象箇所を効率的にずらす．現在の比較対象箇所の末尾を $T[k]$ としたとき，$P[0:M-2]$の中に$T[k]$と同一の文字が含まれているか否かに関して，次の 2 つが考えられる：

Ⅰ： $P[0:M-2]$の中に$T[k]$と同一の文字は存在しない，

Ⅱ： $P[0:M-2]$の中に$T[k]$と同一の文字が存在する．

Ⅰのとき，テキストの次の比較対象箇所を$T[k+1]$からになるようにずらし，新しい部分文字列$T[k+1:k+M]$と$P[0:M-1]$の照合を後ろから行う．Ⅱの場合，$T[k]$が表す文字と$P[j]$（ただし，$j=0,1,\ldots,M-2$）が表す文字が等しくなるjのうち一番大きいものをj'としたとき，テキストの次の比較対象箇所を，$T[k]$と$P[j']$が揃うようにずらし，パターンとの照合を後ろから行う．

具体的に，StringMatch2 で，テキストT=aceafadcadcfcaf からパターンP=cfcaf を探し出す際のステップごとの照合の様子を図 4 に示す．ステップ 1 では，①②と比較を行い③で文字の不一致が起こっている．そこで，テキストの比較対象箇所の末尾 (ここでは$T[4]$) に注目し，$P[0:3]$に$T[4]$と同一の文字が存在すれば，その位置と$T[4]$が揃うようにテキストの比較対象箇所をずらす．この例では，$P[1]$がそれに対応するため，$T[4]$と$P[1]$が揃うように，テキストの比較対象箇所を 3 文字ずらし$T[3:7]$とする．同様に，ステップ 2 では④で不一致が起こり，$T[7]$に注目する．$P[2]$と$T[7]$が揃うようにテキストの比較対象箇所を 2 文字ずらす. ステップ 3 では⑤で不一致が起こり, $T[9]$に注目する. $P[0:3]$には$T[9]$と同一の文字はないので，パターンの文字数分(5 文字)テキストの比較対象箇所をずらす． ステップ 4 では完全一致する様子を表している．

このアルゴリズム StringMatch2 では，探索に入る前に，パターンの各文字が最後に現れる位置に基づいて，不一致が起こった際の各文字のずらす量を事前に計算する．このようなアルゴリズムに対して次の(4)～(7)の問いに答えなさい．

(4) 図 5 は各文字のずらす量を保存するシフト表を求める関数の疑似コードである．空欄 D に当てはまるコードを m と i を用いて答えなさい．

(5) パターンがP=abcdeiabfe のとき，シフト表はどのようになるか求めなさい．

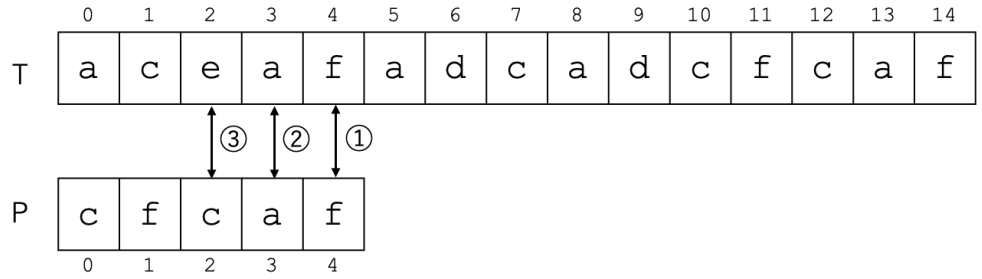
(6) StringMatch2 による文字列の照合を行う図 6 の疑似コード中の空欄 E ～ H に当てはまるものを次の(a)～(o)の選択肢の中からひとつ選び，疑似コードが上記の StringMatch2 の文字列照合を表すように完成させなさい．

(a) i　(b) j　(c) m　(d) n　(e) i+1　(f) j+1　(g) i-1　(h) j-1

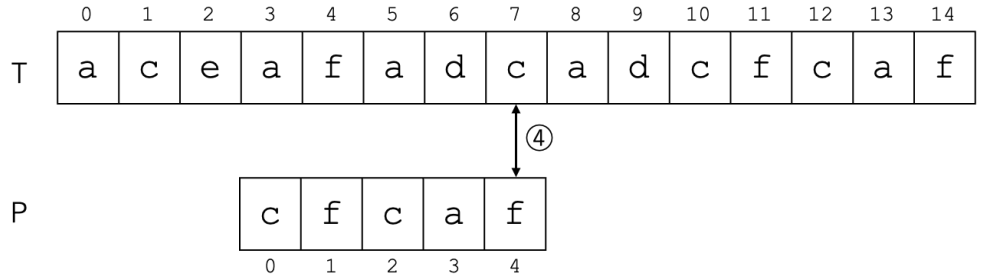
(i) i+j　(j) i+m　(k) i+m-1　(l) i+m+1　(m) i+n　(n) i+n-1　(o) i+n+1

(7) StringMatch2 において，T=aciacadabgag，P=abga としたとき，アルゴリズムが終了した時点での変数 sum の値を求めなさい．

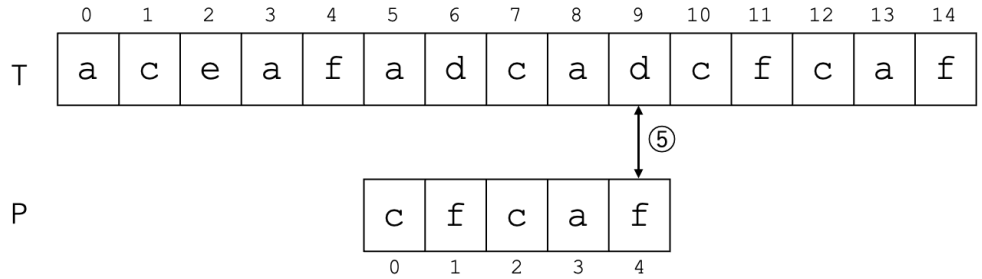
ステップ 1



ステップ 2



ステップ 3



ステップ 4

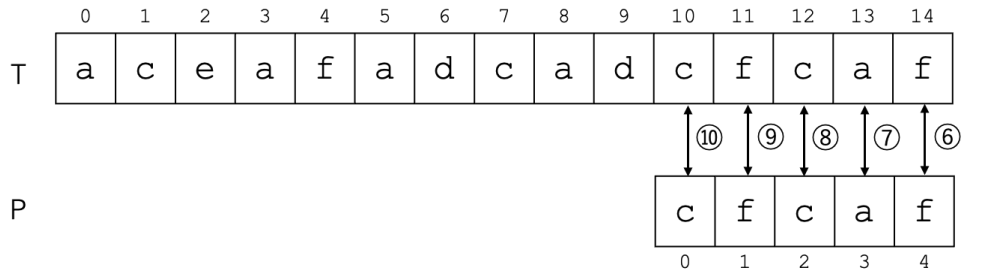


図 4　StringMatch2 における照合の例

```
FUNCTION BuildShiftTable(P):
    // P: パターン
    // P に基づいて ShiftTable(シフト表)を作成する
    // 例えば，文字 a のずらす量が 5 だった場合，
    //            ShiftTable['a'] ← 5
    // と保存する

    m ← length(P)   // length(X)は文字列 X の長さを返す

    // すべての文字のずらす量を m で初期化
    for c from 'a' to 'j' do
        ShiftTable[c] ← m
    end for

    for i from 0 to m − 2 do
        c ← P[i]
        ShiftTable[c] ← [      D      ]
    end for

    return ShiftTable
end FUNCTION
```

図 5　シフト表作成関数

```
FUNCTION StringMatch2(T, P):
    // T: テキスト
    // P: パターン

    shift ← BuildShiftTable(P)   // シフト表の作成
    n ← length(T)   // length(X)は文字列 X の長さを返す
    m ← length(P)
    i ← 0
    sum ← 0     // 総比較回数
    while i ≤ n - m do
        j ← m - 1
        // 後ろから 1 文字ずつ比較
        while j ≥ 0 do
            sum ← sum + 1
            if T[   E   ] == P[   F   ] do
                j ←   G
            else
                break while
        end while

        if j < 0 then
            // 完全一致したら, その添え字 i を返す
            return i
        end if

        // シフト表に従ってテキストの比較対象部分をずらす
        c ← T[   H   ]
        i ← i + shift[c]
    end while

    // 見つからなければ-1 を返す
    return -1
end FUNCTION
```

図 6　StringMatch2 の疑似コード

問 2 (English version)

Consider a string-matching algorithm that searches a text string $T = T_0T_1T_2 \ldots T_{N-1}$ of length N for a pattern string $P = P_0P_1P_2 \ldots P_{M-1}$ of length M, and outputs the index of the starting position where the pattern first occurs in the text. If the pattern appears multiple times in the text, the algorithm should return the smallest index where a match begins. If the pattern does not occur in the text, the algorithm should return -1 and terminate. Assume $N \geq M \geq 1$, and for simplicity, all characters are restricted to the 10 lowercase letters from 'a' to 'j', that is,

$T[i],\ P[j] \in \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}, \texttt{e}, \texttt{f}, \texttt{g}, \texttt{h}, \texttt{i}, \texttt{j}\}$ for $i = 0, 1, 2, \cdots,\ N-1,\ j = 0, 1, 2, \cdots,\ M-1.$

For example, if T=abcdefghiabcdef and P=def, then the substrings in the text that match the pattern $P_0P_1P_2$ are $T_3T_4T_5$ and $T_{12}T_{13}T_{14}$. Since the match starting at index 3 occurs first, the algorithm should return 3.

Now, when implementing such a string-matching algorithm in a popular programming language, it is common to represent strings as arrays. Accordingly, a string X of length N is represented as an array $X[0:N-1]$, where $X[m]$ (for $0 \leq m \leq N-1$) denotes the character at index m in the string X. Furthermore, $X[m:n]$ (for $0 \leq m < n \leq N-1$) represents the substring of X from the m-th character to the n-th character, inclusive.

Figure 2 illustrates the behavior of an algorithm, referred to as StringMatch1, in which a pattern P=cfcaf is compared against a text T=aceafadcadcfcaf. Starting from the first character of the text (i.e., index 0), the algorithm compares the pattern with each candidate substring of length M in the text, with the current position in the text referred to as the comparison window, from left to right, one character at a time. If a mismatch occurs during comparison, the algorithm shifts the comparison window in the text by one character to the right and repeats the comparison with the pattern. Figure 2 shows the comparison process up to Step3. In Step 1, a mismatch occurs at position ①. In Step 2, a mismatch occurs at position ③. In each case, the comparison window in T is shifted by one character to the right, as a result of a mismatch. Figure 3 shows the pseudocode for StringMatch1. Answer the following questions (1) through (3) based on this information.

(1) Choose the appropriate options from choices (a) through (i) to fill in the blanks [A] to [C], So that the pseudocode correctly represents the string matching algorithm StringMatch1 described above.
(a) j>n (b) j<n (c) j>m (d) j<m (e) n (f) m (g) i (h) j (i) i+j

(2) Suppose T=aciacadabgag and P=abga. What is the value of the variable sum at the point when StringMatch1 terminates?

(3) In StringMatch1, suppose a text of length N and a pattern of length M. What is the maximum number of character comparisons in terms of N and M?

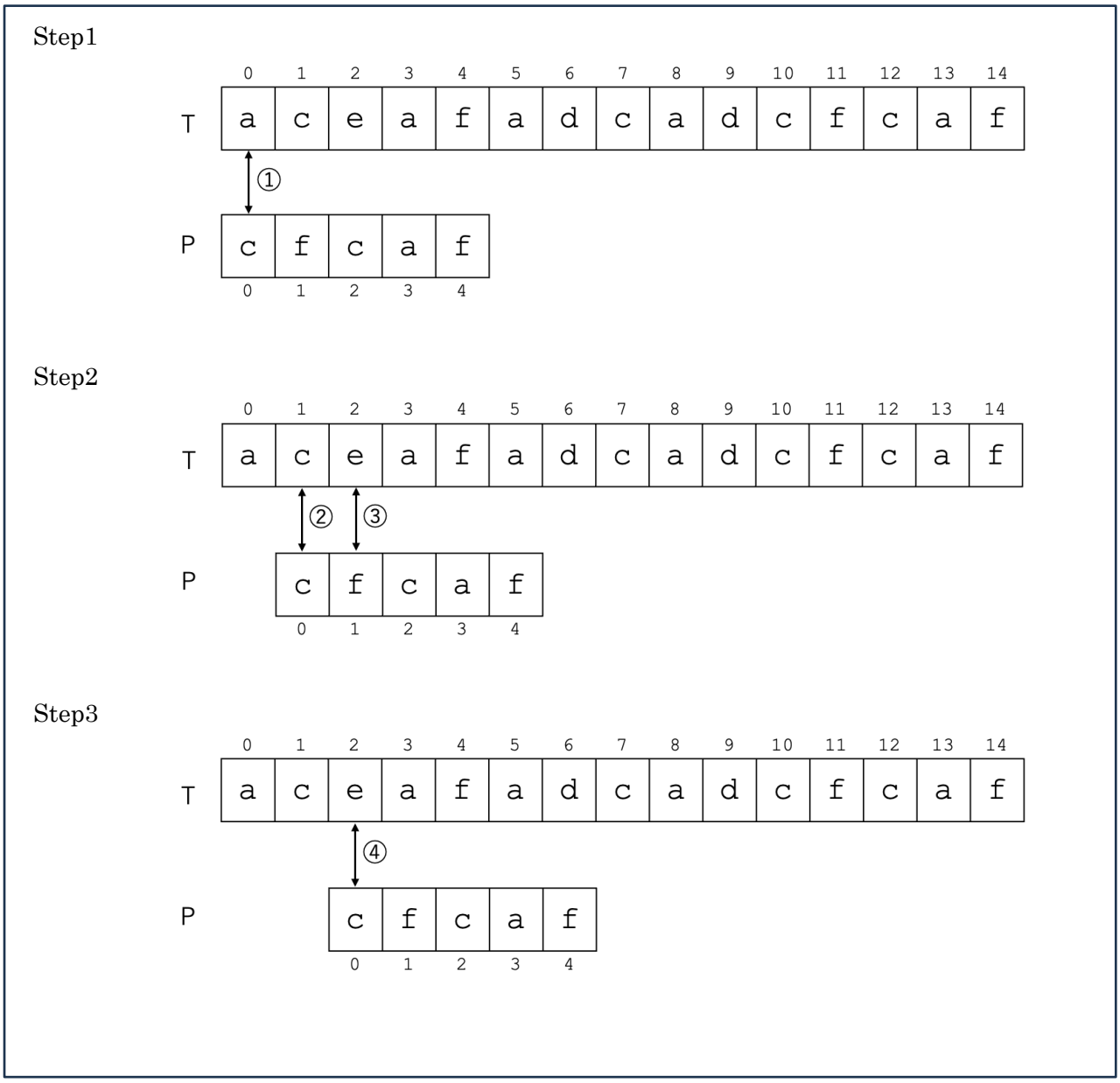


Figure 2　An example of `StringMatch1`.

```
FUNCTION StringMatch1(T, P):
     // T: text indexed from 0 to N-1
     // P: pattern indexed from 0 to M-1
     n ← length(T) // length(X) return the length of X
     m ← length(P)

     // Examine the candidate substring of the text beginning at index i
     sum  ← 0     // Total number of comparisons
     for i from 0 to n − m do
        j ← 0
        // Compare each character of the pattern one at a time
        //                                          from beginning
        while [    A    ] do
           sum ← sum + 1
           if T[ [    B    ] ] == P[ [    C    ] ] then
              j ← j + 1
           else
              break while
           end if
        end while

        // Return the position i if an exact match occurs
        if j = m then
           return i
        end if
     end for

     // Return -1 if no match is found
     return -1
end FUNCTION
```

Figure 3 The pseudocode of StringMatch1.

In the search method used by `StringMatch1`, the comparison window in the text is shifted by one character every time a mismatch occurs, resulting in low efficiency. To improve efficiency, we consider a more efficient algorithm, referred to as `StringMatch2`. As is the case with `StringMatch1`, `StringMatch2` searches for the pattern starting from the beginning of the text; however, it compares characters from the end of the pattern toward the beginning, one character at a time. When a mismatch occurs, `StringMatch2` examines the last character of the current comparison window in the text. Based on whether and where this character appears in the pattern, the algorithm determines how far to shift the comparison window for the next attempt. Let this last character of the comparison window be denoted as $T[k]$. To decide the shift amount, we consider whether or not $T[k]$ appears in the prefix $P[0:M-2]$ of the pattern. There are two possible cases:

Ⅰ: $T[k]$ does not appear in $P[0:M-2]$;

Ⅱ: $T[k]$ does appear in $P[0:M-2]$.

In case Ⅰ, the comparison window in the text is shifted so that the next comparison starts at $T[k+1]$. The new substring $T[k+1:k+M]$ is then compared with $P[0:M-1]$, starting from the last character of the pattern. In case Ⅱ, let j' be the largest index $j \in \{0, 1, \ldots, M-2\}$ such that $P[j] = T[k]$. The comparison window in the text is shifted so that $T[k]$ aligns with $P[j']$, and the pattern is compared with the new text substring starting from the last character.

Figure 4 shows the first four steps of `StringMatch2` for when T=`aceafadcadcfcaf` and P=`cfcaf`. At Step 1, Comparisons ① and ② find matches but Comparison ③ fails. At this point, the algorithm examines the last character of the current comparison window in the text, which is $T[4]$. If this character appears in $P[0:3]$, the comparison window is shifted so that this character is aligned with its rightmost occurrence in $P[0:3]$. In this example, since $P[1]$ matches $T[4]$, so the comparison window is shifted 3 characters to the right, aligning $T[4]$ with $P[1]$, resulting in a new comparison substring $T[3:7]$. Similarly, the mismatch found by Comparison ④ at Step 2 leads to checking if $T[7]$ appears in P, which results in a shift of 2 due to $P[2] = T[7]$. At Step 3, the first comparison in the reverse comparison process at this position finds a mismatch. Because $T[9]$ does not appear in $P[0:3]$, the comparison window is shifted by the full length of the pattern, which is 5, to move on to the next reverse comparison process. Step 4 finds an instance of P.

In the `StringMatch2` algorithm, before starting the search, the shift amount for each character is precomputed based on its last occurrence in the pattern string. Answer the following questions (4) through (7) regarding this algorithm.

(4) Figure 5 shows the pseudocode for a function that precomputes the shift table, which stores the shift amount for each character. Fill in the blank [D] using variables `m` and `i`.

(5) Give the table of appropriate shifts for P=`abcdeiabfe`.

(6) Figure 6 is a pseudocode for the main part of `StringMatch2`, where the precomputed table is used for string matching. Choose the appropriate options from (a) to (o) to fill in the blanks [E] through [H] so that the pseudocode correctly represents the `StringMatch2` algorithm.

(a) `i` (b) `j` (c) `m` (d) `n` (e) `i+1` (f) `j+1` (g) `i-1` (h) `j-1`

(i) `i+j` (j) `i+m` (k) `i+m-1` (l) `i+m+1` (m) `i+n` (n) `i+n-1` (o) `i+n+1`

(7) Suppose T=`aciacadabgag` and P=`abga`. What is the value of the variable `sum` at the point when `StringMatch2` terminates?

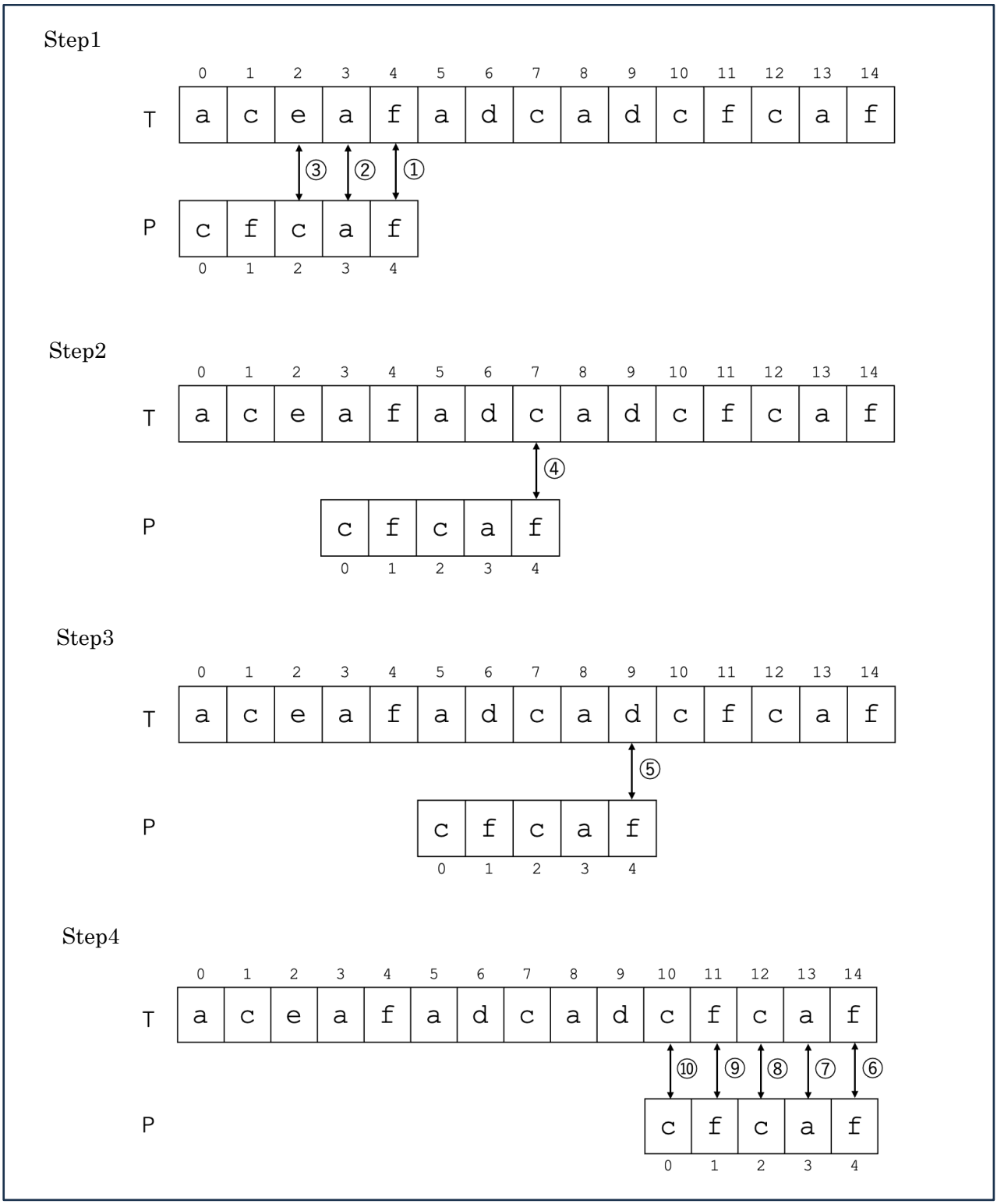


Figure 4　An example of `StringMatch2`.

```
FUNCTION BuildShiftTable(P):
     // P: Pattern
     // Build the shift table from the pattern P
     // For example, if the shift amount for the character 'a' is 5:
     //           ShiftTable['a'] ← 5

     m ← length(P)   // length(X) return the length of X

     // Initialize the shift amount for all characters to m
     for c from 'a' to 'j' do
        ShiftTable[c] ← m
     end for

     for i from 0 to m − 2 do
        c ← P[i]
        ShiftTable[c] ← [      D      ]
     end for

     return ShiftTable
end FUNCTION
```

Figure 5　The function to build the shift table.

```
FUNCTION StringMatch2(T, P):
     // T: text
     // P: pattern

     shift ← BuildShiftTable(P)   //  Shift table
     n ← length(T)   // length(X) return the length of X
     m ← length(P)
     i ← 0
     sum ← 0     // Total number of comparisons
     while i ≤ n - m do
        j ← m - 1
        // Compare each character from the end, one at a time
        while j ≥ 0 do
           sum ← sum + 1
           if T[      E      ] == P[      F      ] do
              j ←     G
           else
              break while
        end while

        if j < 0 then
           // Return the position i if an exact match occurs
           return i
        end if

        // Shift the comparison window
        //           in the text according to the ShiftTable
        c ← T[     H     ]
        i ← i + shift[c]
     end while

     // Return -1 if no match is found
     return -1
end FUNCTION
```

Figure 6 The pseudocode of StringMatch2.