Introduction to the Theory of Computation 2025 — Final

Solutions

Problem 1 (10 pts). Please prove that the language

$$A_{\text{DFA}} = \{\langle B, w\rangle \mid B = \langle Q, \Sigma, \delta, q_0, F\rangle \text{ is a DFA that accepts } w\} \text{ is decidable.}$$

Note that you can give a high-level description of a Turing machine instead of explicitly constructing it.

Solution.

Step 1: Put the encoding $\langle B, w\rangle$ on the input tape, where $B = \langle Q, \Sigma, \delta, q_0, F\rangle$.

Step 2: Check whether $w \in \Sigma^*$ and whether B is a valid DFA. If not, reject.

Step 3: Simulate the DFA B on input the string w according to the transition function δ.

Step 4: After processing the last symbol of w, check whether the current state is a final state. If it is, accept; otherwise, reject.

Problem 2 (30 pts). Assume f and g are functions $f, g : \mathbb{N} \to \mathbb{R}^+$. Prove or disprove the sub-problems by using the following definitions.

Definition 1. We say

$$f(n) = O(g(n))$$

if **there exists** $c > 0$ and $n_0 \in \mathbb{N}$ such that for every integer $n \geq n_0$, $f(n) \leq cg(n)$.

Definition 2. We say

$$f(n) = o(g(n))$$

if **for each** $c > 0$, there exists $n_0 \in \mathbb{N}$ such that for every integer $n \geq n_0$, $f(n) \leq cg(n)$.

To prove the statements, you must give the specific n_0 for one c or all c's, depending on the definition of big-O or small-o. To disprove the statements, you must prove the opposite of the definition in detail. Note that we use the natural logorithm in this problem, i.e.,

$$\log n \coloneqq \log_e n.$$

(a) (5 pts) Let

$$f(n) = n^{\frac{1}{\log n}} \text{ and } g(n) = 1.$$

Prove or disprove that $f(n) = O(g(n))$.

(b) (5 pts) Please show that for every integer $n \geq 2$, we have

$$\frac{n}{4}\log\left(\frac{3n}{4}\right) > \frac{1}{8}n\log n. \tag{1}$$

(c) (5 pts) Let

$$f(n) = \log(n!) \text{ and } g(n) = n\log n.$$

Prove or disprove that $f(n) = o(g(n))$. *Hint:* You may need to use (1) in your proof, and think about the relation between

$$\sum_{k=1}^{n}\log k \text{ and } \sum_{k=\lfloor 3n/4\rfloor}^{n}\log k.$$

(d) (5 pts) Let

$$f(n) = O(g(n)).$$

Prove or disprove that $2^{f(n)} = O(2^{g(n)})$.

(e) (5 pts) We define

$$f(n) = f_1(n) + f_2(n),$$

where

$$f_1(n) = o(g(n)), \text{ and } f_2(n) = o(g(n)).$$

Please prove that

$$f(n) = o(g(n))$$

with **Definition 2**. Note that proving with the limit would not receive any points.

(f) (5 pts) Let

$$f(n) = e^{\sqrt{n}} + 7n \text{ and } g(n) = 5^{\sqrt{n}}.$$

Prove or disprove that $f(n) = o(g(n))$. *Hint:* The inequality $\log n < \sqrt{n}$ might be useful.

Solution.

(a) To show that there exist $c \geq 0$ and $n_0 \in \mathbb{N}$ such that for every integer $n \geq n_0$,

$$f(n) \leq cg(n).$$

We can take logarithms on both sides to derive

$$\log f(n) \leq \log(cg(n)) \equiv \frac{1}{\log n}\log n \leq \log c \equiv 1 \leq \log c.$$

By taking

$$c = 3 \text{ and } n_0 = 1,$$

we obtain

$$\log f(n) = 1 \leq \log(3) = \log(3 \cdot g(n)),\ \forall n \geq n_0,$$

which shows that $f(n) = O(g(n))$.

(b) To show that for every $n \geq 2$,

$$\frac{n}{4}\log\left(\frac{3n}{4}\right) \;>\; \frac{1}{8}n\log n.$$

This inequality is equivalent to

$$2\log\left(\frac{3n}{4}\right) \;>\; \log n \equiv \left(\frac{3n}{4}\right)^2 > n \equiv \frac{9n}{16} > 1.$$

(c) The opposite of **Definition 2** is

For some $c > 0$ such that for any $n_0 \in \mathbb{N}$, there exists $n \geq n_0$ so that we have $f(n) > c\,g(n)$.

By combining (1) with the definition of $f(n)$, we can derive that

$$\log(n!) = \sum_{k=1}^{n}\log k \geq \sum_{k=\lfloor 3n/4\rfloor}^{n}\log k \geq \frac{n}{4}\log\left(\frac{3n}{4}\right) > \frac{1}{8}n\log n, \text{ for every integer } n \geq 2.$$

Let us take

$$c = \frac{1}{8}.$$

Then, we can prove that for every $n \geq 2$,

$$f(n) = \log(n!) \;>\; \frac{1}{8}g(n) = cg(n),$$

and which implies that

for any $n_0 \in \mathbb{N}$, there exists $n \geq n_0$ such that $f(n) > cg(n)$.

Thus the statement is disproved.

(d) The opposite of

$$2^{f(n)} = O(2^{g(n)})$$

is

for all $c > 0$ and $n_0 \in \mathbb{N}$, there exists an $n \geq n_0$ such that $2^{f(n)} > c2^{g(n)}$.

Let us take $f(n) = 2n$ and $g(n) = n$, such that $f(n)$ and $g(n)$ satisfy $f(n) = O(g(n))$. Thus, for any $c > 0$ and $n_0 \in \mathbb{N}$, we can pick $n \geq \max\{c, n_0\}$ such that

$$\frac{2^{f(n)}}{2^{g(n)}} = \frac{2^{2n}}{2^n} = \frac{4^n}{2^n} = 2^n > c.$$

Therefore, the statement is disproved.

(e) Consider any real number $c > 0$. Let

$$c_1 = c_2 = \frac{c}{2}.$$

By **Definition 2**, we can find $n_1, n_2 \in \mathbb{N}$ such that

$$f_1(n) \leq c_1 g(n) \text{ for every integer } n > n_1$$

and

$$f_2(n) \leq c_2 g(n) \text{ for every integer } n > n_2.$$

If we take

$$n_0 = \max(n_1, n_2),$$

then for every integer $n > n_0$, we can see that

$$f(n) = f_1(n) + f_2(n) \leq \frac{c}{2}g(n) + \frac{c}{2}g(n) = cg(n).$$

Therefore, $f(n) = o(g(n))$.

(f) We utilize the property established in Problem 2 (e) to show that $f(n) = o(g(n))$. First, let

$$f_1(n) = e^{\sqrt{n}} \text{ and } f_2(n) = 7n.$$

For any real number $c > 0$, we determine n_1 and n_2 by the following two inequalities

$$e^{\sqrt{n}} \leq c \cdot 5^{\sqrt{n}} \tag{2}$$

and

$$7n \leq c \cdot 5^{\sqrt{n}}, \tag{3}$$

respectively. From inequality (2), we can derive

$$\begin{aligned} &e^{\sqrt{n}} \leq c \cdot 5^{\sqrt{n}} \\ \Rightarrow &\sqrt{n} \leq \log c + \sqrt{n} \log 5 \\ \Rightarrow &\sqrt{n}(1 - \log 5) \leq \log c. \end{aligned}$$

Since we use the natural log in this problem, it is clear that $1 - \log 5 < 0$. Consequently, we obtain

$$\begin{aligned} &\sqrt{n}(1 - \log 5) \leq \log c \\ \Rightarrow &\sqrt{n} \geq \frac{\log c}{1 - \log 5} \\ \Rightarrow &n \geq \left(\frac{\log c}{1 - \log 5}\right)^2. \end{aligned} \tag{4}$$

The inequality (4) implies that we can choose

$$n_1 = \max\left(\left\lceil \left(\frac{\log c}{1 - \log 5}\right)^2 \right\rceil, 1\right)$$

such that

$$f_1(n) \leq cg(n)$$

holds for all integers $n \geq n_1$.

From the inequality (3), we obtain

$$\begin{aligned} &7n \leq c \cdot 5^{\sqrt{n}} \\ \Rightarrow &\log 7 + \log n \leq \log c + \sqrt{n} \log 5 \\ \Rightarrow &\log 7 + \log n - \sqrt{n} \log 5 \leq \log c. \end{aligned} \tag{5}$$

Furthermore, by utilizing the hint, we have

$$\log 7 + \log n - \sqrt{n}\log 5 < \log 7 + \sqrt{n} - \sqrt{n}\log 5. \tag{6}$$

The combination of inequalities (5) and (6) implies that we can use the following inequality

$$\log 7 + \sqrt{n} - \sqrt{n}\log 5 \leq \log c \tag{7}$$

to determine a suitable n_2. From the inequality (7), we can further derive

$$\begin{aligned} &\log 7 + \sqrt{n} - \sqrt{n}\log 5 \leq \log c \\ \Rightarrow &\sqrt{n}(1-\log 5) \leq \log c - \log 7 \\ \Rightarrow &\sqrt{n} \geq \frac{\log c - \log 7}{1 - \log 5} \\ \Rightarrow &n \geq \left(\frac{\log c - \log 7}{1-\log 5}\right)^2. \end{aligned} \tag{8}$$

The inequality (8) implies that we can choose

$$n_2 = \max\left(\left\lceil \left(\frac{\log c - \log 7}{1-\log 5}\right)^2 \right\rceil, 1\right)$$

such that

$$f_2(n) \leq cg(n)$$

holds for all integers $n \geq n_2$. Since we have already shown that

$$f_1(n) = o(g(n)) \text{ and } f_2 = o(g(n)),$$

the property proven in Problem 2 (e) directly yields $f(n) = o(g(n))$.

Problem 3 (40 pts). In Midterm 2, you learned how to design a TM for the task "string masking." For example, at the beginning of the TM, we have the tape

$$a \quad b \quad a \quad a \quad b \quad \# \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad \# \quad \sqcup \quad \cdots,$$

which represents that we have the input string

$$abaab$$

and the mask

$$01010.$$

After the masking, i.e.,

$$\begin{matrix} a & b & a & a & b \\ 0 & 1 & 0 & 1 & 0 \end{matrix} \Rightarrow ba,$$

we should add the result at the end of the tape as

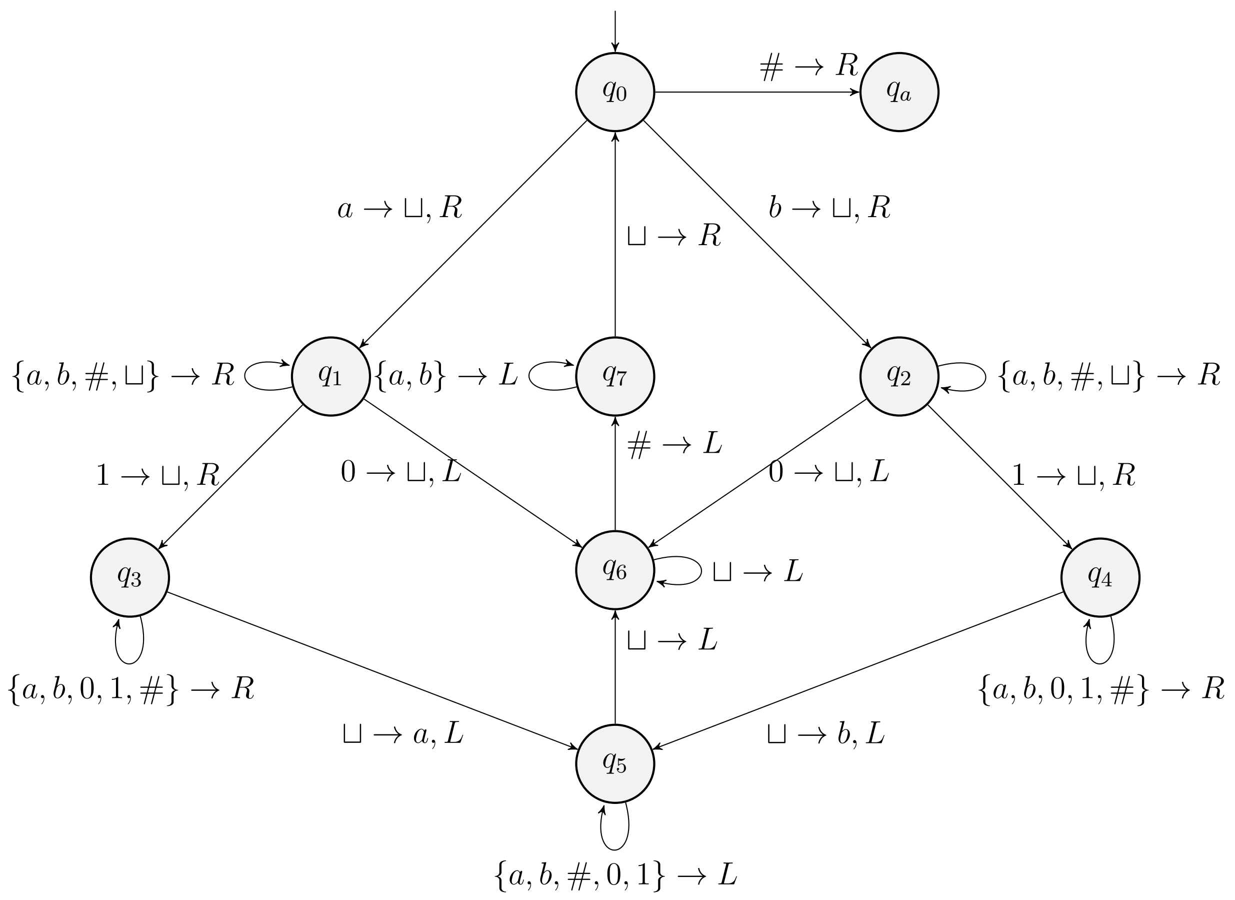
$$\cdots \quad \# \quad \cdots \quad \# \quad b \quad a \quad \sqcup \quad \cdots.$$

Note that we assume the input tape must be

$$\{a,b\}^*\#\{0,1\}^*\#.$$

Moreover, the input and the mask strings already have the **same** length (i.e., no need to check this).

(a) (10 pts) As the solution, we have the TM



Please derive the steps that the TM uses on the input

$$aba\#101\#\sqcup$$

and complete the parts (I) $\cdots$ (VIII) of the following table. Note that the 1st round and the 2nd round handle the underlined bits of

$$\underline{a}ba\#101\#\sqcup \text{ and } \sqcup\underline{b}a\#\sqcup 01\#a\sqcup, \text{ respectively.}$$

terms	1st round	**# step(s)**	2nd round	**# step(s)**
$q_0 \to q_1$ or $q_0 \to q_2$	$\sqcup q_1 ba\#101\#\sqcup$	1	$\sqcup\sqcup q_2 a\#\sqcup 01\#a\sqcup$	1
$q_1 \to q_1$ or $q_2 \to q_2$	$\sqcup ba\#q_1 101\#\sqcup$	$\underline{\text{(I)}}$	$\sqcup\sqcup a\#\sqcup q_2 01\#a\sqcup$	$\underline{\text{(VI)}}$
$q_1 \to q_3$ or $q_2 \to q_4$	$\sqcup ba\#\sqcup q_3 01\#\sqcup$	1		0
$q_3 \to q_3$ or $q_4 \to q_4$	$\sqcup ba\#\sqcup 01\#q_3\sqcup$	$\underline{\text{(II)}}$		0
$q_3 \to q_5$ or $q_4 \to q_5$	$\sqcup ba\#\sqcup 01 q_5\#a\sqcup$	1		0
$q_5 \to q_5$	$\sqcup ba\#q_5\sqcup 01\#a\sqcup$	$\underline{\text{(III)}}$		0
$q_5 \to q_6$	$\sqcup ba q_6\#\sqcup 01\#a\sqcup$	1		0
$q_1 \to q_6$ or $q_2 \to q_6$			$\sqcup\sqcup a\#q_6\sqcup\sqcup 1\#a\sqcup$	1
$q_6 \to q_6$	$\sqcup ba q_6\#\sqcup 01\#a\sqcup$	$\underline{\text{(IV)}}$	$\sqcup\sqcup a q_6\#\sqcup\sqcup 1\#a\sqcup$	$\underline{\text{(VII)}}$
$q_6 \to q_7$	$\sqcup b q_7 a\#\sqcup 01\#a\sqcup$	1	$\sqcup\sqcup q_7 a\#\sqcup\sqcup 1\#a\sqcup$	$\underline{\text{(VIII)}}$
$q_7 \to q_7$	$q_7\sqcup ba\#\sqcup 01\#a\sqcup$	$\underline{\text{(V)}}$	$\sqcup q_7\sqcup a\#\sqcup\sqcup 1\#a\sqcup$	1
$q_7 \to q_0$	$\sqcup q_0 ba\#\sqcup 01\#a\sqcup$	1	$\sqcup\sqcup q_0 a\#\sqcup\sqcup 1\#a\sqcup$	1
$q_0 \to q_a$		0		0

You can directly write your answers of (I) $\cdots$ (VIII) without any explanation.

(b) (5 pts) Please show the time complexity of this single-tape Turing machine in big-O notation with respect to the input string length n.

(c) (15 pts) Please follow the steps to design a **two-tape** TM:

Step 1: Copy the first part of the string (i.e., the part before the first #) into the 2nd tape.

Step 2: Move the head of the 1st tape to the position of the second #.

Step 3: Scan the mask string from right to left. During the scan, we apply the mask to the corresponding character in the 2nd tape. If the mask bit is 1, we keep the character. Otherwise, we modify the character to $\sqcup$.

Step 4: Move the head of the 1st tape to the answer part (i.e., the part after the second #).

Step 5: Write unmasked characters from the 2nd tape to the 1st tape. The head of the two tapes should move from left to right.

Note that your **two-tape** TM must satisfy the following conditions.

- $\Sigma = \{\#, a, b, 0, 1\}$,
- $\Gamma = \Sigma \cup \{\sqcup\}$,
- no more than 6 states are used (the reject state is excluded),
- the head can move left, move right, or **stay** in a two-tape TM, and
- **do not modify any input string before the second # in the 1st tape.**

Please give the explanation for your diagram. To simplify the diagram, you can utilize

$$\begin{tikzcd}
\bigcirc \arrow[rrrr, "\begin{cases}\{0,1\} \to L \\ \{a,b,\sqcup\} \to S\end{cases}"] & & & & \bigcirc
\end{tikzcd}$$

to represent

$$\begin{tikzcd}
\bigcirc \arrow[rrrr, "\begin{cases}x \to L \\ y \to S\end{cases} \quad ,\forall x \in \{0,1\}, \forall y \in \{a,b,\sqcup\}"] & & & & \bigcirc
\end{tikzcd}$$

(d) (5 pts) Please simulate your two-tape TM in (c) with the input

$$ab\#01\#\sqcup\cdots.$$

(e) (5 pts) Please show the time complexity of your two-tape TM in big-O notation with respect to the input string length n.

Solution.

(a) After directly simulating the input string, we know that

$$(\text{I}) = 3,\ (\text{II}) = 3,\ (\text{III}) = 3,\ (\text{IV}) = 0,\ (\text{V}) = 2,\ (\text{VI}) = 3,\ (\text{VII}) = 1,\ (\text{VIII}) = 1.$$

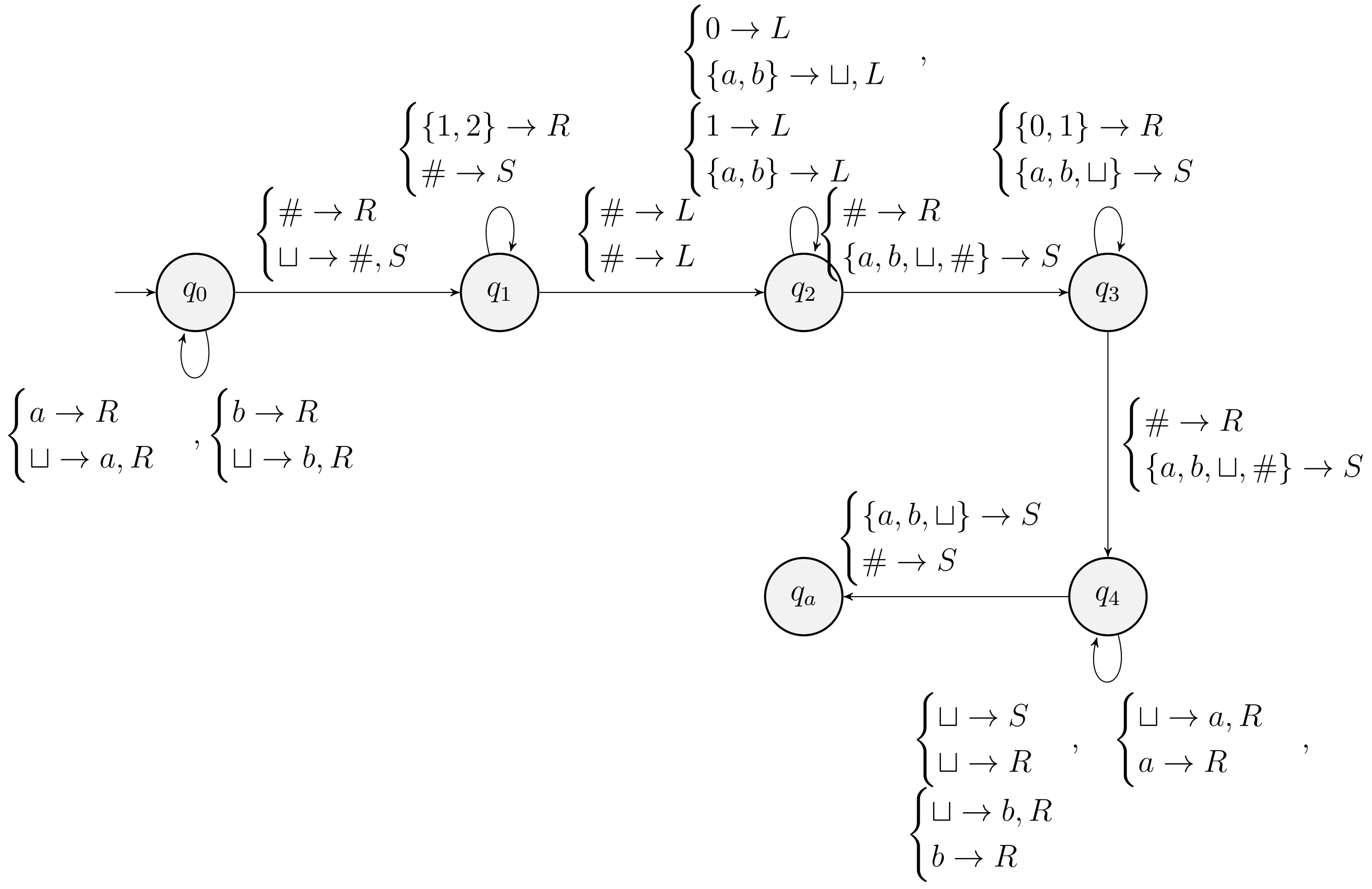
(b) For each character before the first #, we consider the following two cases.

- If the corresponding mask bit is 1, then the TM takes approximately $2n$ steps to process the character:
  - $n/2$ steps for checking the mask bit,
  - $n/2$ steps for writing the character to the answer part, and
  - n steps for moving the head to the next processed character.
- If the corresponding mask bit is 0, then the TM takes approximately n steps to process the character:
  - $n/2$ steps for checking the mask bit, and
  - $n/2$ steps for moving the head to the next processed character.

By combining the above cases, since there are $n/2$ characters that need to be processed, the time complexity is

$$\frac{n}{2} \cdot O(n) = O(n^2).$$

(c) Please see the following diagram.



q_0
q_1
q_2
q_3
q_4
q_a
$\begin{cases} a \to R \\ \sqcup \to a, R \end{cases}, \begin{cases} b \to R \\ \sqcup \to b, R \end{cases}$
$\begin{cases} \# \to R \\ \sqcup \to \#, S \end{cases}$
$\begin{cases} \{1,2\} \to R \\ \# \to S \end{cases}$
$\begin{cases} \# \to L \\ \# \to L \end{cases}$
$\begin{cases} 0 \to L \\ \{a,b\} \to \sqcup, L \end{cases}, \begin{cases} 1 \to L \\ \{a,b\} \to L \end{cases}$
$\begin{cases} \# \to R \\ \{a,b,\sqcup,\#\} \to S \end{cases}$
$\begin{cases} \{0,1\} \to R \\ \{a,b,\sqcup\} \to S \end{cases}$
$\begin{cases} \# \to R \\ \{a,b,\sqcup,\#\} \to S \end{cases}$
$\begin{cases} \{a,b,\sqcup\} \to S \\ \# \to S \end{cases}$
$\begin{cases} \sqcup \to S \\ \sqcup \to R \end{cases}, \begin{cases} \sqcup \to a, R \\ a \to R \end{cases}, \begin{cases} \sqcup \to b, R \\ b \to R \end{cases}$

Explanation of states:

- q_0: Copy the first part of the string (i.e., the part before the first #) from tape 1 to tape 2.
- q_1: Move the head of the first tape to the position of the second #.
- q_2: Scan the mask string from right to left. During the scan, apply the mask to the corresponding characters on the second tape. If the mask bit is 1, keep the character; otherwise, replace the character with $\sqcup$.
- q_3: Move the head of the first tape to the answer part (i.e., the part after the second #).
- q_4: Write the unmasked characters from the second tape to the first tape. The heads of the two tapes move from left to right.

(d) Here is the simulation:

$$\to \begin{matrix} q_0 & a & b & \# & 0 & 1 & \# & \sqcup \\ q_0 & \sqcup & \sqcup & \sqcup & \sqcup & \sqcup & \sqcup & \sqcup \end{matrix} \to \begin{matrix} a & q_0 & b & \# & 0 & 1 & \# & \sqcup \\ a & q_0 & b & \sqcup & \sqcup & \sqcup & \sqcup & \sqcup \end{matrix} \to \begin{matrix} a & b & q_0 & \# & 0 & 1 & \# & \sqcup \\ a & b & q_0 & \sqcup & \sqcup & \sqcup & \sqcup & \sqcup \end{matrix}$$

$$\to \begin{matrix} a & b & \# & q_1 & 0 & 1 & \# & \sqcup \\ a & b & q_1 & \# & \sqcup & \sqcup & \sqcup & \sqcup \end{matrix} \to \begin{matrix} a & b & \# & 0 & q_1 & 1 & \# & \sqcup \\ a & b & q_1 & \# & \sqcup & \sqcup & \sqcup & \sqcup \end{matrix} \to \begin{matrix} a & b & \# & 0 & 1 & q_1 & \# & \sqcup \\ a & b & q_1 & \# & \sqcup & \sqcup & \sqcup & \sqcup \end{matrix}$$

$$\to \begin{matrix} a & b & \# & 0 & q_2 & 1 & \# & \sqcup \\ a & q_2 & b & \# & \sqcup & \sqcup & \sqcup & \sqcup \end{matrix} \to \begin{matrix} a & b & \# & q_2 & 0 & 1 & \# & \sqcup \\ q_2 & a & b & \# & \sqcup & \sqcup & \sqcup & \sqcup \end{matrix} \to \begin{matrix} a & b & q_2 & \# & 0 & 1 & \# & \sqcup \\ q_2 & \sqcup & b & \# & \sqcup & \sqcup & \sqcup & \sqcup \end{matrix}$$

$$\to \begin{matrix} a & b & \# & q_3 & 0 & 1 & \# & \sqcup \\ q_3 & \sqcup & b & \# & \sqcup & \sqcup & \sqcup & \sqcup \end{matrix} \to \begin{matrix} a & b & \# & 0 & q_3 & 1 & \# & \sqcup \\ q_3 & \sqcup & b & \# & \sqcup & \sqcup & \sqcup & \sqcup \end{matrix} \to \begin{matrix} a & b & \# & 0 & 1 & q_3 & \# & \sqcup \\ q_3 & \sqcup & b & \# & \sqcup & \sqcup & \sqcup & \sqcup \end{matrix}$$

$$\to \begin{matrix} a & b & \# & 0 & 1 & \# & q_4 & \sqcup \\ q_4 & \sqcup & b & \# & \sqcup & \sqcup & \sqcup & \sqcup \end{matrix} \to \begin{matrix} a & b & \# & 0 & 1 & \# & q_4 & \sqcup \\ \sqcup & q_4 & b & \# & \sqcup & \sqcup & \sqcup & \sqcup \end{matrix} \to \begin{matrix} a & b & \# & 0 & 1 & \# & b & q_5 \\ \sqcup & b & q_5 & \# & \sqcup & \sqcup & \sqcup & \sqcup \end{matrix}$$

$$\to \begin{matrix} a & b & \# & 0 & 1 & \# & b & q_5 \\ \sqcup & b & q_5 & \# & \sqcup & \sqcup & \sqcup & \sqcup \end{matrix}$$

(e) Let the input string have length n. The following shows the time complexity of each step.

Step 1: This step requires $n/2$ steps.
Step 2: This step requires $n/2$ steps.
Step 3: This step requires $n/2$ steps.
Step 4: This step requires $n/2$ steps.
Step 5: This step requires $n/2$ steps.

Therefore, the time complexity of this two-tape Turing machine is

$$O\left(\frac{5n}{2}\right) = O(n).$$

Problem 4 (20 pts). In this problem, we want to find out whether the substring "CSIE" is in the tape string. For example,

"I am a student in CSIE. The courses are great."

includes the substring "CSIE." For an opposite example,

"I am a CS student. I sometimes use the IE browser."

does **not** include the substring "CSIE."

(a) (10 pts) Please design a nondeterministic Turing machine (NTM), where

- Θ is the vocabulary pool we use in the tape string,
- $\Sigma = \Theta$,
- $\Gamma = \Theta \cup \{\sqcup\}$,
- ≤ 5 nodes (reject state is excluded).

Please give the explanation for your diagram.

(b) (5 pts) Now consider

$$L = \{D \mid D \text{ contains the substring “CSIE”}\}.$$

To show that $L \in \text{NP}$, please design a polynomial-time verifier V for L with the certificate

$$c = \text{the string index } i.$$

Analyze your algorithm at each step and further show the time complexity on a TM. Note that your verifier algorithm must include clear, step-by-step, and high-level descriptions.

(c) (5 pts) Show that $L \in \text{NP}$ again by designing an NTM that uses the verifier V in (b) as a subroutine.

Solution.

(a) Please see the following diagram.

$$\begin{tikzcd}
{} \arrow[r] & q_0 \arrow[loop above, "\Gamma \to R"] \arrow[r, "C \to R"] & q_1 \arrow[r, "S \to R"] & q_2 \arrow[r, "I \to R"] & q_3 \arrow[r, "E \to R"] & q_a
\end{tikzcd}$$

Explanation of states:

- q_0: Nondeterministically guess whether it is the start of the substring “CSIE.”
- $q_1 \sim q_3$: Check whether the following characters corresponding “S,” “I,” and “E.”

(b) Here is the algorithm:

Step1: Check whether $i \leq (|D| - 3)$, where $|D|$ is the length of D. Obtaining $|D|$ requires a single pass over the input string, which takes $O(n)$.

Step2: Move the head to the i-th position of D, and check whether $D_{[i]}$ = “C,” $D_{[i+1]}$ = “S,” $D_{[i+2]}$ = “I,” and $D_{[i+3]}$ = “E,” which requires $O(n)$.

Therefore, this algorithm requires $O(n + n)$ time to verify the certificate c, which implies that $L \in \text{NP}$.

(c) We can nondeterministically pick the index i

$$i \in \{1, 2, \ldots, (|D| - 3)\},$$

with the polynomial-time verifier V we defined in (b). Thus $L \in \text{NP}$.