- **Please give details of your calculation. A direct answer without explanation is not counted.**
- **Your answers must be in English.**
- **Please carefully read problem statements.**
- **During the exam you are not allowed to borrow others' class notes.**
- **Try to work on easier questions first.**

Problem 1 (15 pts)

Convert the following CFG into CNF with $\Sigma = \{a, b\}$.

$$S \to bS \mid E \mid \epsilon$$
$$E \to aEb \mid a$$

And please follow the formal procedure, i.e. Theorem 2.9 of the textbook.

Answer

Add $S_0 \to S$ to the CFG:

$$S_0 \to S$$
$$S \to bS \mid E \mid \epsilon$$
$$E \to aEb \mid a$$

Remove $S \to \epsilon$:

$$S_0 \to S \mid \epsilon$$
$$S \to bS \mid E \mid b$$
$$E \to aEb \mid a$$

Note that we don't have to remove $S_0 \to \epsilon$. The next step is to remove $S \to E$:

$$S_0 \to S \mid \epsilon$$
$$S \to bS \mid aEb \mid a \mid b$$
$$E \to aEb \mid a$$

And remove $S_0 \to S$:

$$S_0 \to bS \mid aEb \mid a \mid b \mid \epsilon$$
$$S \to bS \mid aEb \mid a \mid b$$
$$E \to aEb \mid a$$

Finally, we add some rules to remove $S_0 \to bS \mid aEb, S \to bS \mid aEb, E \to aEb$:

$$S_0 \to BS \mid AC \mid a \mid b \mid \epsilon$$
$$S \to BS \mid AC \mid a \mid b$$
$$E \to AC \mid a$$
$$A \to a$$
$$B \to b$$
$$C \to EB$$

Common Mistakes

You should indicate the purpose of each step, such as saying "removing $S \to \epsilon$".

- $S_0 \to \epsilon$ should be added when removing $S \to \epsilon$.

Notes: This problem is the easiest, so more points are taken if you make mistakes.

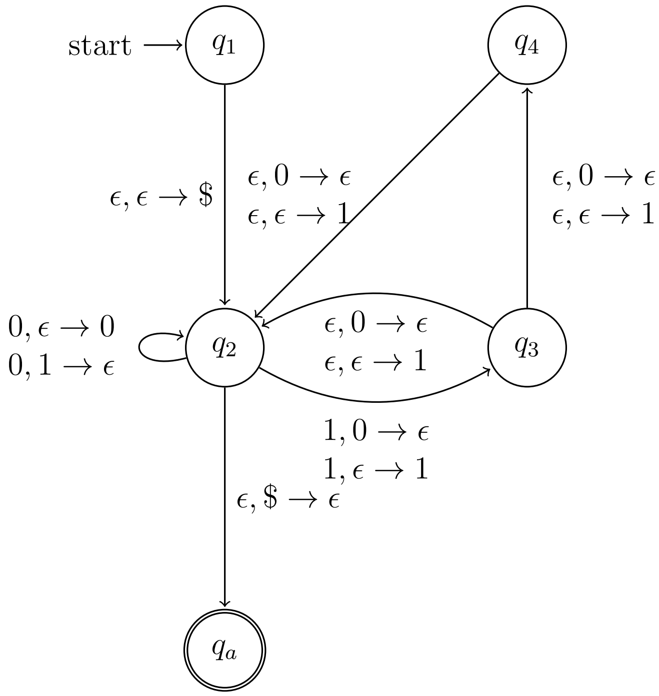
Problem 2 (20 pts)

Consider the following language

$$\{w \mid 2n_1(w) \leq n_0(w) \leq 3n_1(w)\},$$

where $\Sigma = \{0, 1\}$ and $n_{0/1}(w)$ means the number of 0's (or 1's) in w. Construct a PDA with ≤ 5 states to recognize this language. Give the formal definition of your PDA.

Answer



Explanation: for each '1', we need two '0's or three '0's to match it; therefore, the smallest number of '0' is $2n_1(w)$ and the largest is $3n_1(w)$.

- (q_2, q_3): exchange two '0's for one '1'. If there are not enough '0's, push '1'; otherwise, pop '0'.

- (q_2, q_3, q_4): exchange three '0's for one '1'. If there are not enough '0's, push '1'; otherwise, pop '0'.

Formal Definition:

$$M = (Q, \Sigma, \Gamma, \delta, q_1, \{q_a\}), \text{ where}$$

$Q = \{q_1, q_2, q_3, q_4, q_a\}$,

$\Sigma = \{0, 1\}$,

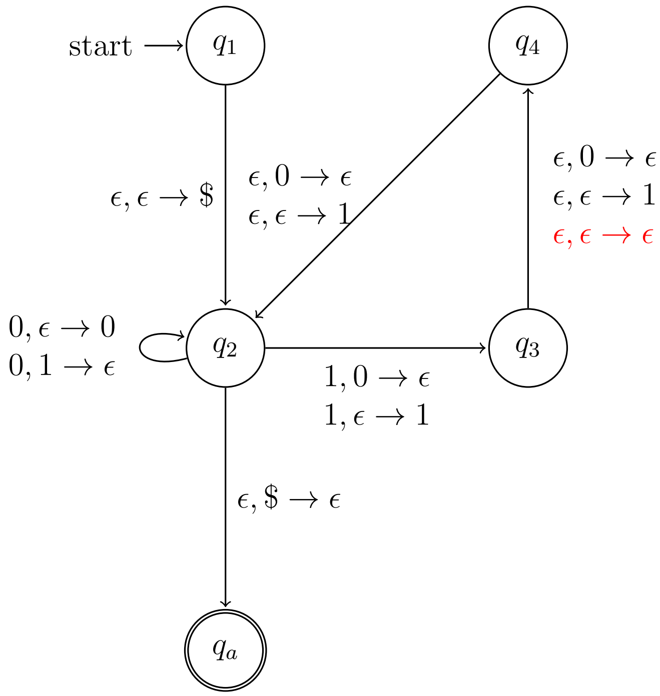
$\Gamma = \{0, 1, \$\}$, and

$\delta =$

Input	0				1				ϵ			
Stack	0	1	\$	ϵ	0	1	\$	ϵ	0	1	\$	ϵ
q_1												$\{(q_2, \$)\}$
q_2		$\{(q_2, \epsilon)\}$		$\{(q_2, 0)\}$	$\{(q_3, \epsilon)\}$			$\{(q_3, 1)\}$			$\{(q_a, \epsilon)\}$	
q_3									$\{(q_2, \epsilon), (q_4, \epsilon)\}$			$\{(q_2, 1), (q_4, 1)\}$
q_4									$\{(q_2, \epsilon)\}$			$\{(q_2, 1)\}$
q_a												$\{(q_2, \$)\}$

Other Solutions

1.



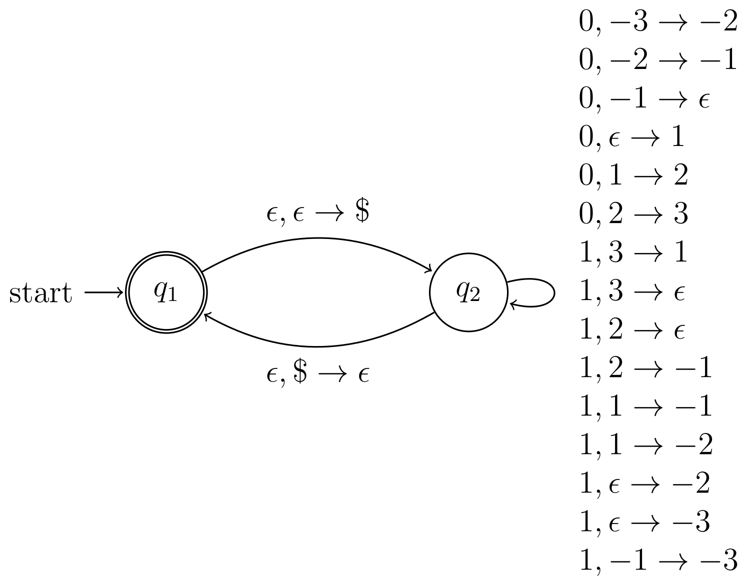
2.

The stack alphabet set is {**-3, -2, -1, 1, 2, 3** }. In the stack alphabets set, **1** represents one '0', **2** represents two '0's, **-1** represents lacking one '0' and so on.

- When PDA reads '0', it can non-deterministically push a **1** to stack or promote the symbol at the top of stack.

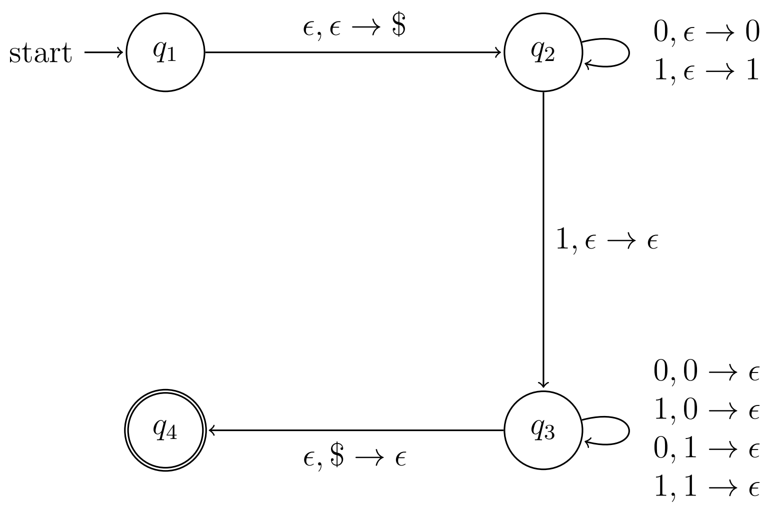
$$\mathbf{-3} \to \mathbf{-2} \to \mathbf{-1} \to \epsilon \to \mathbf{1} \to \mathbf{2} \to \mathbf{3}$$

- When PDA reads '1', it can non-deterministically push a **-3** to stack, push a **-2** to stack, demote a symbol's rank by 2 or demote a symbol's rank by 3. For example, **3** can be demoted to **1** or ϵ, **2** can be demoted to ϵ or **-1**.



Problem 3 (20 pts)

Consider the following PDA with $\Sigma = \{0, 1\}$



(a) What is the language ? (4 pts)

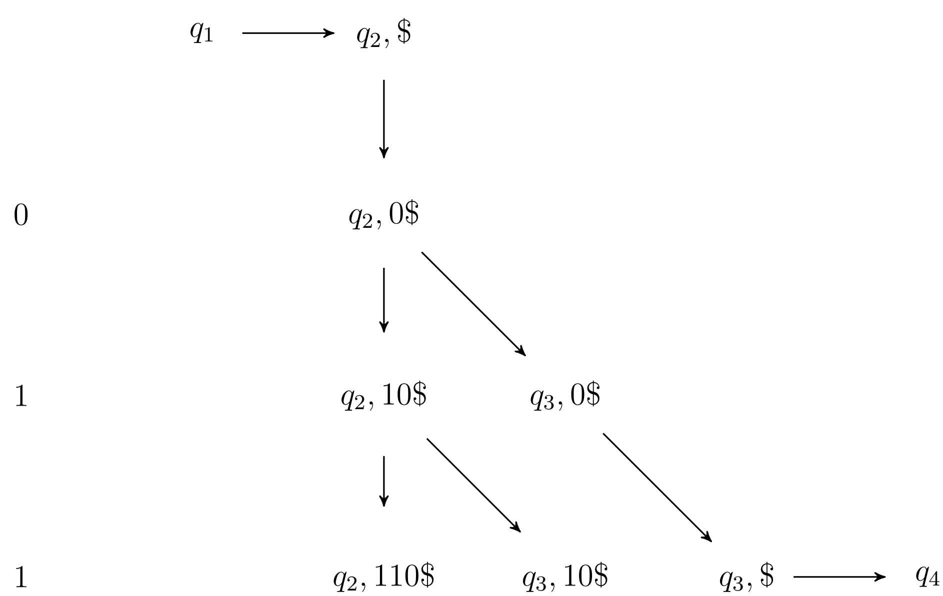
(b) Draw the tree to process the string 011. Your tree must be complete. Note that we mean a tree to process an input string. We do not mean a parse tree of CFG. (4 pts)

(c) Find CFG of this PDA's language. You are required to follow the same procedure in lemma 2.27 to generate rules. You should **not** remove any redundant rules generated by the lemma. (12 pts)

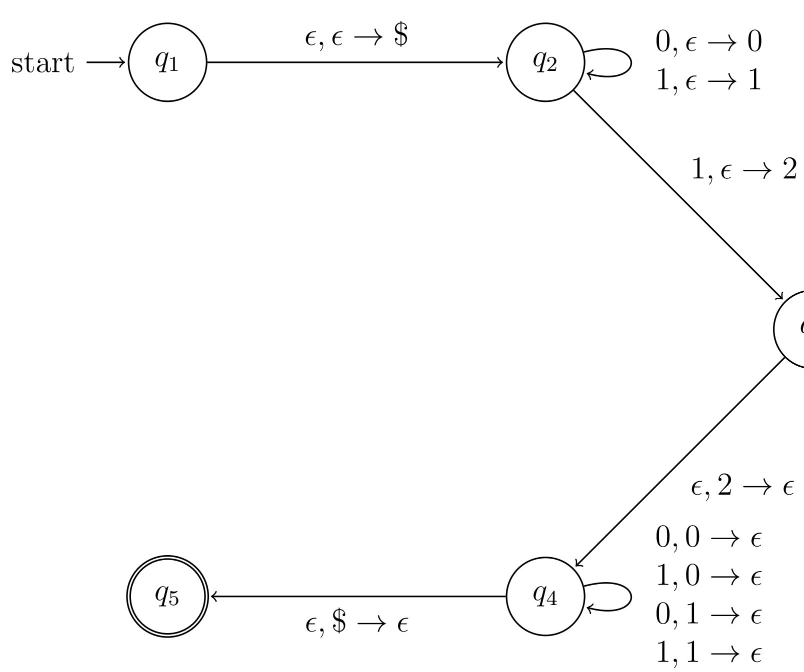
Answer

(a) This language is the set of words which have an odd number of alphabets and '1' at the middle position.

(b)



(c) First, each link in PDA should be either pop or push, so we have to change the PDA to be:



Now we construct a CFG from this PDA. The CFG has 25 variables, $V = \{A_{ij} \mid 1 \le i, j \le 5\}$, start variable is A_{15}, Σ is the same as PDA, and the rules are:

$$A_{ik} \to A_{ij}A_{jk} \ (1 \le i, j, k \le 5,\ \text{total 125 rules})$$

$$A_{ii} \to \epsilon \ (1 \le i \le 5,\ \text{total 5 rules})$$

$$A_{15} \to A_{24}$$

$$A_{24} \to 0A_{24}0$$

$$A_{24} \to 0A_{24}1$$

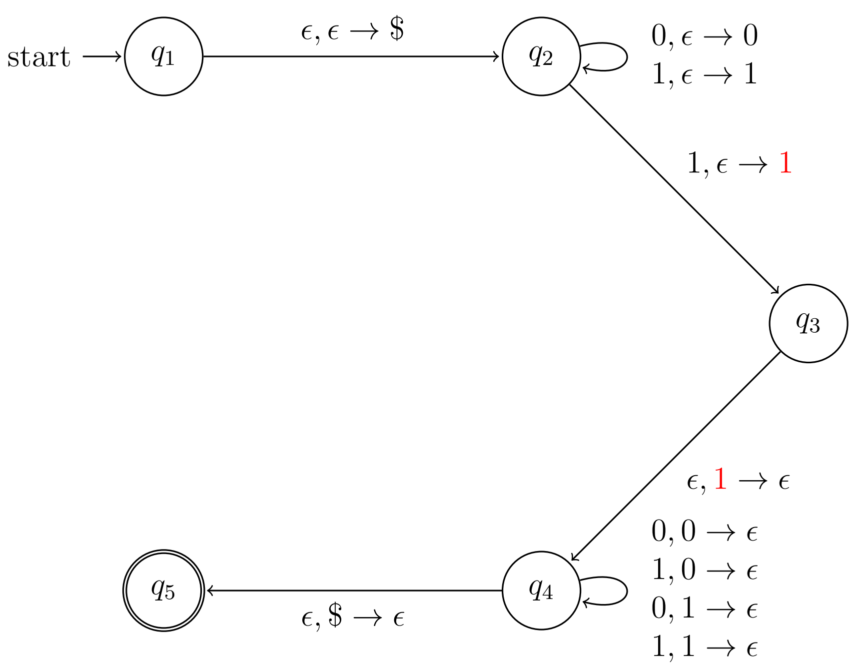
$$A_{24} \to 1A_{24}0$$

$$A_{24} \to 1A_{24}1$$

$$A_{24} \to 1A_{33}$$

The first two rules are always needed if you follow the formal procedure. The other rules come from δ of the PDA. You can see lemma 2.27 for more information.

Common mistakes



$A_{ik} \to A_{ij}A_{jk}$ $(1 \le i, j, k \le 5,$ total 125 rules)

$A_{ii} \to \epsilon$ $(1 \le i \le 5,$ total 5 rules)

$A_{15} \to A_{24}$

$A_{24} \to 0A_{24}0$

$A_{24} \to 0A_{24}1$

$A_{24} \to 1A_{24}0$

$A_{24} \to 1A_{24}1$

$A_{24} \to 1A_{33}$

$A_{24} \to 1A_{23}$

$A_{24} \to 1A_{34}0$

$A_{24} \to 1A_{34}1$

Problem 4 (15 pts)

(a) Construct a Turing machine (i.e., showing the state diagram) for the language

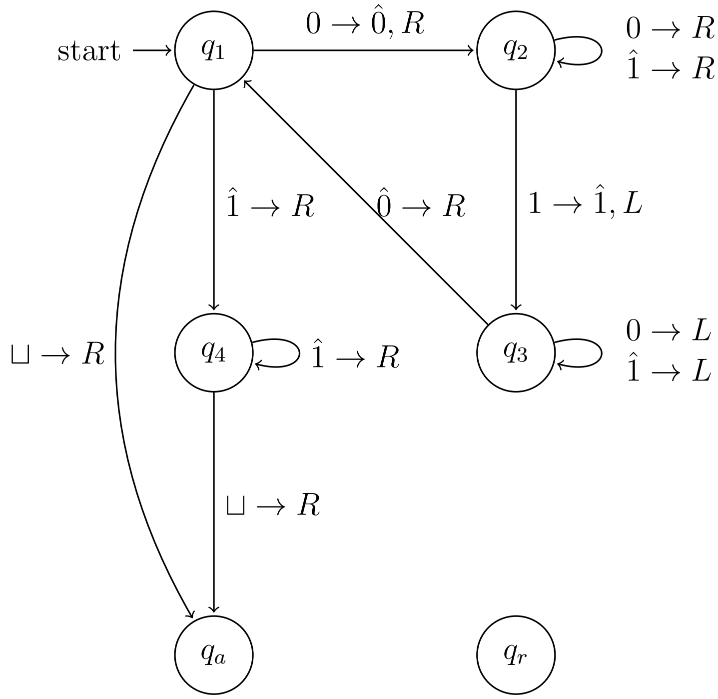
$$\{0^n1^n \mid n \ge 0\}.$$

Note that we use the standard Turing machine rather than extensions such as nondeterministic Turing machine. The number of states is ≤ 6, including q_a and q_r. You can assume $\Sigma = \{0, 1\}$.

(b) Give the formal definition.

Answer

(a)



- (q_1, q_2): Mark 0 in 0^n.
- (q_2, q_2): Move to the first 1 which is not marked.
- (q_2, q_3): Mark 1 in 1^n.
- (q_3, q_3): Move back to the last marked 0.
- (q_3, q_1): Move to the first 0 which is not marked.
- (q_1, q_4): All 0's are marked.
- (q_4, q_4): Check all 1's are marked.
- (q_4, q_a): Accept the string.
- (q_1, q_a): Accept empty string.

(b) $M = (Q, \Sigma, \Gamma, \delta, q_1, q_a, q_r)$, where

$$Q = \{q_1, q_2, q_3, q_4, q_r, q_a\},$$

$$\Sigma = \{0, 1\},$$

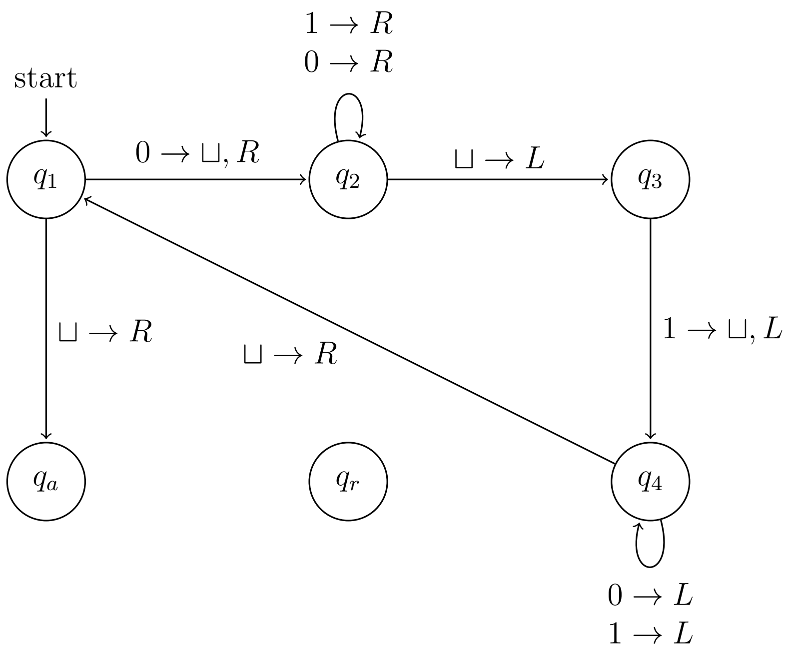
$$\Gamma = \{0, 1, \hat{0}, \hat{1}, \sqcup\}, \text{ and}$$

$\delta =$

	0	1	$\hat{0}$	$\hat{1}$	$\sqcup$
q_1	$(q_2, \hat{0}, R)$			$(q_4, \hat{1}, R)$	$(q_a, \sqcup, R)$
q_2	$(q_2, 0, R)$	$(q_3, \hat{1}, L)$		$(q_2, \hat{1}, R)$	
q_3	$(q_3, 0, L)$		$(q_1, \hat{0}, R)$	$(q_3, \hat{1}, L)$	
q_4				$(q_4, \hat{1}, R)$	$(q_a, \sqcup, R)$

Empty entries in δ means "reject" (q_r).

Other Solutions



This setting sequentially cancels 0 in the begining and 1 in the end.

Common Mistakes

- Reject ϵ.
- Assume $\sqcup$ at the left end of the tape.
- Accept $(0^n 1^n)^*$.
- Accept some strings with wrong 01 order, e.g. 001011.

Problem 5 (15 pts)

Consider the language

$$\{w\#w \mid w \in \{0,1\}^*\},$$

where $\Sigma = \{0, 1\}$.

(a) Construct a 2-tape Turing machine to recognize this language. We assume that

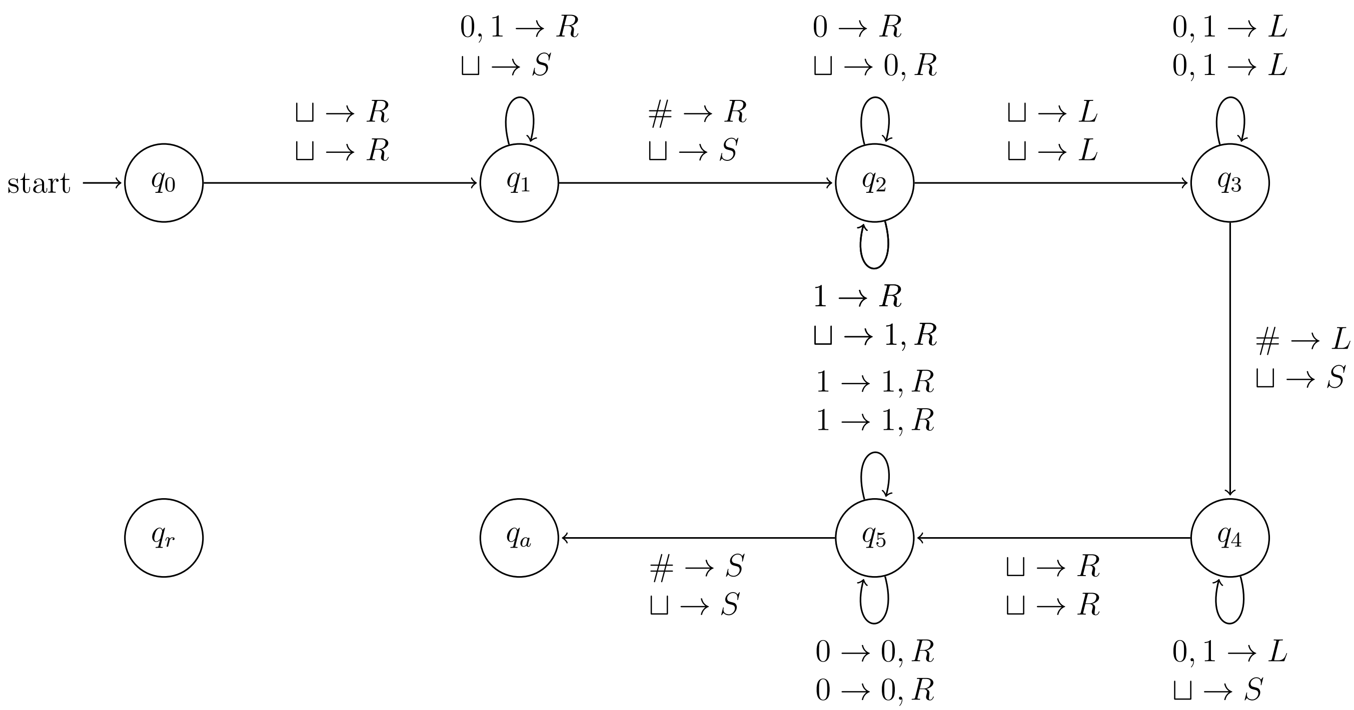
1. in the beginning, $\sqcup$(input) in the 1st tape.
2. we copy the second part to the 2nd tape and then compare strings in both tapes.
3. the number of states (including q_a and q_r) should be no more than 8.

No need to give the formal definition.

(b) Simulate one string 01#01.

Answer

1.



- (q_0, q_1): Skip $\sqcup$.
- (q_1, q_1): Move to #.
- (q_1, q_2): Skip #.
- (q_2, q_2): Copy the second part to the 2nd tape.
- (q_2, q_3): Finish copy.
- (q_3, q_3): Go back to #.
- (q_3, q_4): Skip #.
- (q_4, q_4): Go back to the head of first part.
- (q_4, q_5): Skip $\sqcup$.
- (q_5, q_5): Compare string.

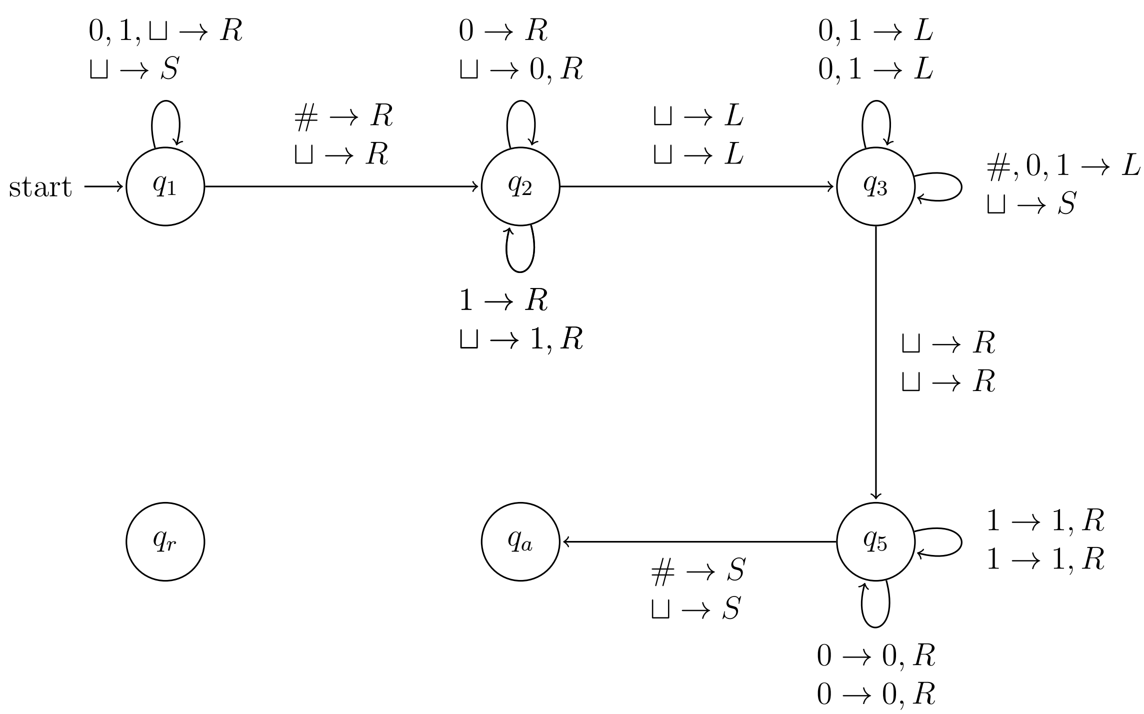
- (q_5, q_a): Accept.

2. - 01#01

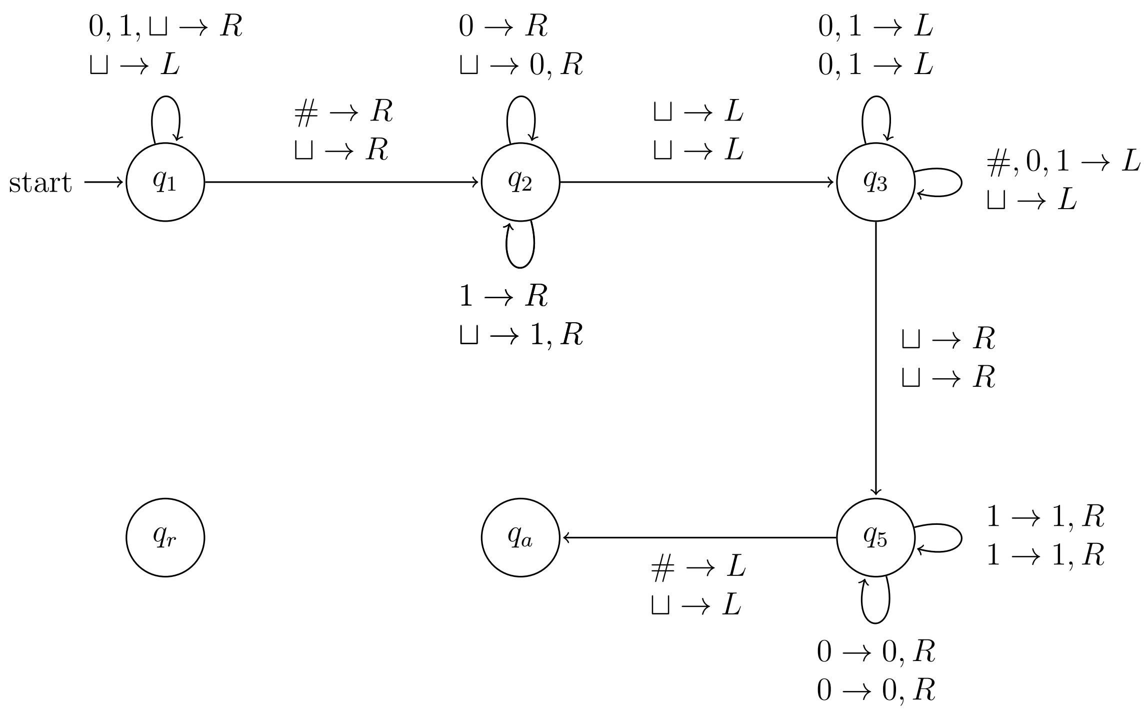
$$
\begin{array}{llllllllll}
\begin{matrix} q_0 \sqcup 01\#01\sqcup \\ q_0 \sqcup \sqcup \end{matrix} & \rightarrow & \begin{matrix} \sqcup q_1 01\#01\sqcup \\ \sqcup q_1 \sqcup \end{matrix} & \rightarrow & \begin{matrix} \sqcup 0 q_1 1\#01\sqcup \\ \sqcup q_1 \sqcup \end{matrix} & \rightarrow & \begin{matrix} \sqcup 01 q_1 \#01\sqcup \\ \sqcup q_1 \sqcup \end{matrix} & \rightarrow & \begin{matrix} \sqcup 01\# q_2 01\sqcup \\ \sqcup q_1 \sqcup \end{matrix} & \rightarrow \\
\begin{matrix} \sqcup 01\#0 q_2 1\sqcup \\ \sqcup 0 q_2 \sqcup \end{matrix} & \rightarrow & \begin{matrix} \sqcup 01\#01 q_2 \sqcup \\ \sqcup 01 q_2 \sqcup \end{matrix} & \rightarrow & \begin{matrix} \sqcup 01\#0 q_3 1\sqcup \\ \sqcup 0 q_3 1\sqcup \end{matrix} & \rightarrow & \begin{matrix} \sqcup 01\# q_3 01\sqcup \\ \sqcup q_3 01\sqcup \end{matrix} & \rightarrow & \begin{matrix} \sqcup 01 q_3 \#01\sqcup \\ q_3 \sqcup 01\sqcup \end{matrix} & \rightarrow \\
\begin{matrix} \sqcup 0 q_4 1\#01\sqcup \\ q_4 \sqcup 01\sqcup \end{matrix} & \rightarrow & \begin{matrix} \sqcup q_4 01\#01\sqcup \\ q_4 \sqcup 01\sqcup \end{matrix} & \rightarrow & \begin{matrix} q_4 \sqcup 01\#01\sqcup \\ q_4 \sqcup 01\sqcup \end{matrix} & \rightarrow & \begin{matrix} \sqcup q_5 01\#01\sqcup \\ \sqcup q_5 01\sqcup \end{matrix} & \rightarrow & \begin{matrix} \sqcup 0 q_5 1\#01\sqcup \\ \sqcup 0 q_5 1\sqcup \end{matrix} & \rightarrow \\
\begin{matrix} \sqcup 01 q_5 \#01\sqcup \\ \sqcup 01 q_5 \sqcup \end{matrix} & \rightarrow & \begin{matrix} \sqcup 01 q_a \#01\sqcup \\ \sqcup 01 q_a \sqcup \end{matrix} & & & & & & &
\end{array}
$$

Other Solutions

1. Merge (q_0, q_1) and (q_3, q_4).



2. Because of the property that the head of a Turning machine stays in the same place for an attempt to move its head to the left of the beginning of the tape, S (Stay) can be removed.



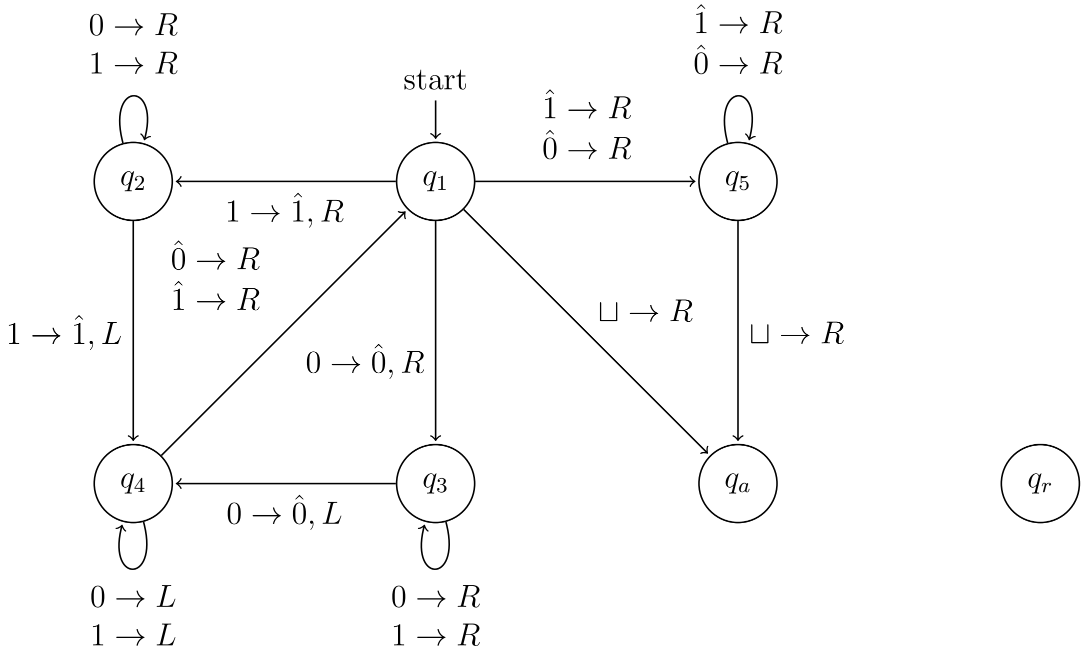
Problem 6 (15 pts)

Construct a nondeterministic Turing Machine with no more than 7 states (including q_a and q_r) to recognize the following language:

$$\{ww^R \mid w \in \{0,1\}^*\},$$

where w^R is the reverse of a string. No need to give the formal definition.

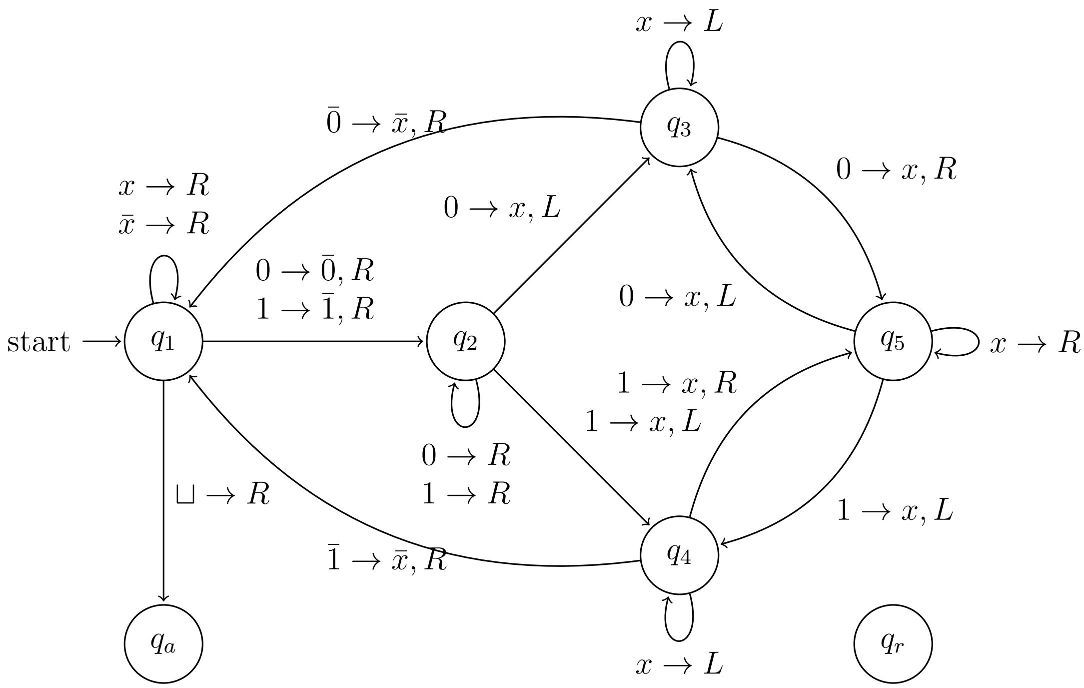
Answer



- (q_1, q_2): Mark 1 in w.
- (q_1, q_3): Mark 0 in w.
- (q_2, q_2): Ignore some symbols between marked symbols.
- (q_3, q_3): Ignore some symbols between marked symbols.
- (q_2, q_4): Mark 1 in w^R.
- (q_3, q_4): Mark 0 in w^R.
- (q_4, q_4): Move back to the last marked symbol in w.
- (q_4, q_1): Move to the first symbol which is not marked.
- (q_1, q_5): Find the end of w.
- (q_5, q_5): Check all symbols are marked.
- (q_5, q_a): All symbols are marked, so accept the input string.
- (q_1, q_a): Accept empty string.

Common Mistakes

1.
   - Reject ϵ.
   - Accept $\Sigma^* w w^R$
   - Accept $(ww^R)^*$. The following example nondeterministically finds the middle point of the input string (at q_2) then compares the two parts (w and w^R).

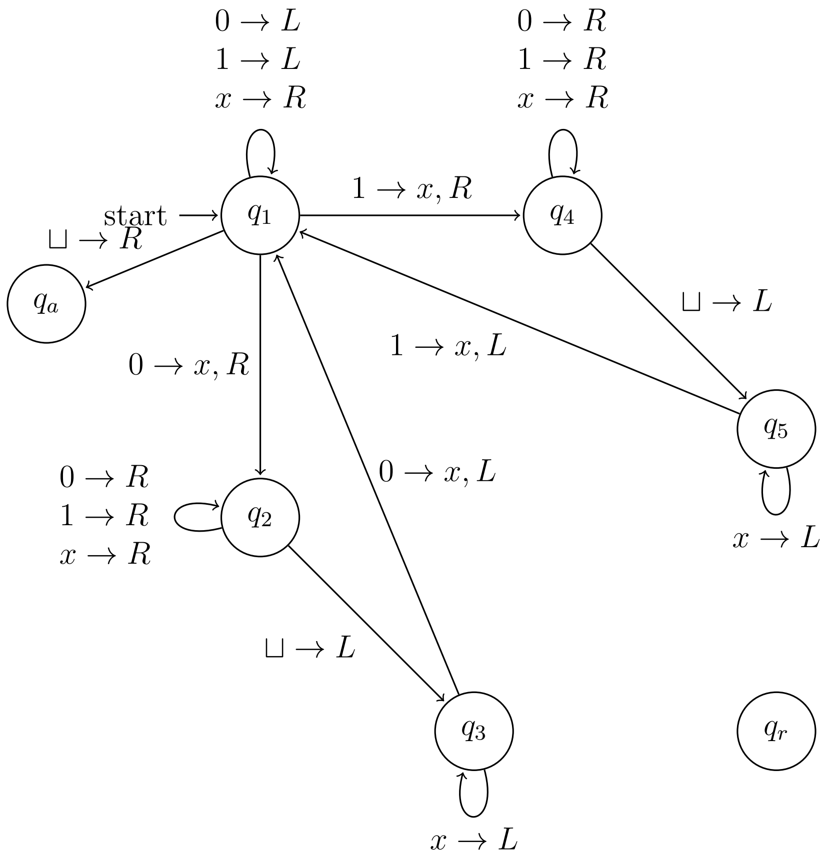


   Problem: q_1 can be the end of ww^R and the start of another ww^R. The following procedure shows how it accepts 0011: $q_1 0011 \to \bar{0} q_2 011 \to q_3 \bar{0} x 11 \to \bar{x} q_1 x 11 \to \bar{x} x q_1 11 \to \bar{x} x \bar{1} q_2 1 \to \bar{x} x q_4 \bar{1} x \to \bar{x} x \bar{x} q_1 x \to \bar{x} x \bar{x} x q_1 \sqcup \to \bar{x} x \bar{x} x \sqcup q_a$.

   Similarly, in the diagram of the answer, q_1 and q_5 cannot be combined. Otherwise, $(ww^R)^*$ will be accepted.

   - Assume $\sqcup$ at the left end of the tape.
   - # states > 7.
   - The first part may contain some redundant symbols, for example,

2. In the following diagram, it does not check whether all symbols are marked before accept; therefore, it may accept more strings.


0 → L
1 → L
x → R
0 → R
1 → R
x → R
start
1 → x, R
q1
q4
⊔ → R
qa
⊔ → L
1 → x, L
0 → x, R
q5
0 → R
1 → R
x → R
q2
0 → x, L
x → L
⊔ → L
q3
qr
x → L