CSC 320
Foundations of Computer Science

Lecture 12

Instructor: Dr. Ulrike Stege

Territory Acknowledgement

We acknowledge and respect the lək̓ʷəŋən peoples on whose traditional territory the university stands and the Songhees, Esquimalt and W̱SÁNEĆ peoples whose historical relationships with the land continue to this day.

This meeting will be recorded

"Please be aware our sessions are being screen-recorded to allow students who are not able to attend to watch later and will be posted in Brightspace."

Deadlines; Assessment

10%
Quizzes
Quiz 1-8: 1% each
Quiz 9: 2%

25%
Assignments
Assignment 1-5: 5% each

25%
Midterms
Midterm 1: 10%
Midterm 2: 15%

40%
Final Exam

May

S	M	T	W	T	F	S
			3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

June

S	M	T	W	T	F	S
28	29	30	31	1	2	[illegible]
4	5	6	7	8	9	[illegible]
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	1
2	3	4	5	6	7	8

July

S	M	T	W	T	F	S
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	

Timed quizzes (~30 min)
Review before starting quiz

Last time ....

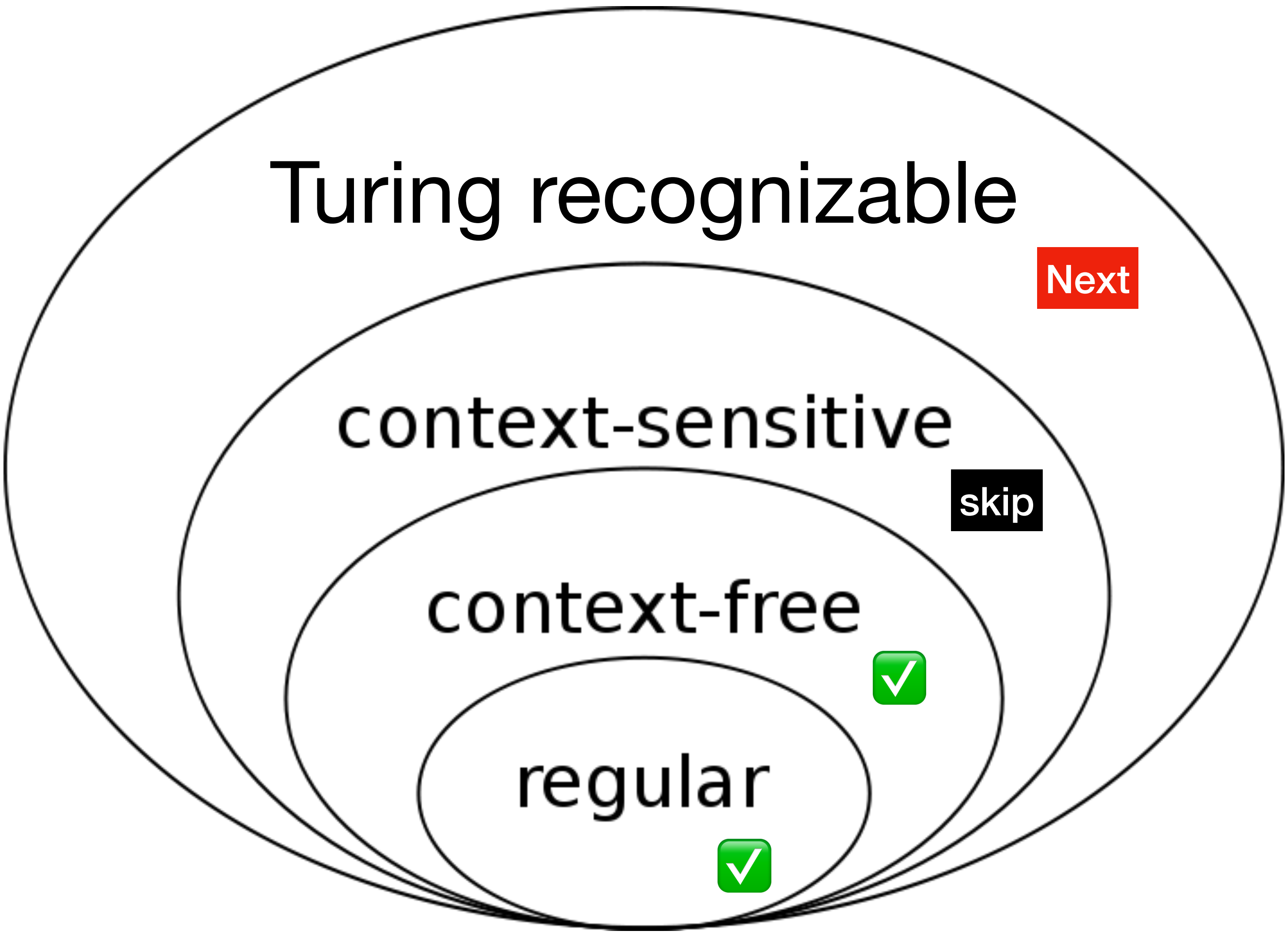
- Pumping lemma for context-free languages
- $B = \{a^n b^n c^n \mid n \geq 0\}$ not context-free
- $L = \{ww \mid w \in \{0,1\}^*\}$ not context-free

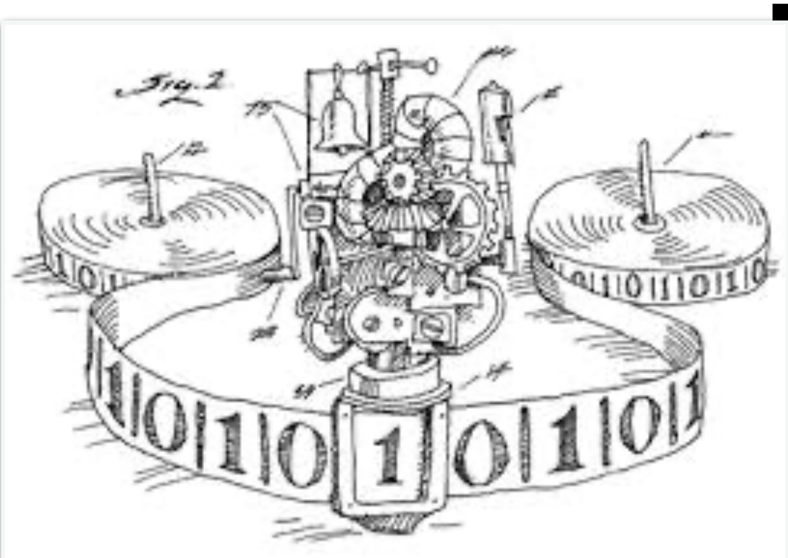
Pumping lemma for context-free languages

If L is a context-free language, then there is a number p (*pumping length*) such that: if s is any string in L of length at least p, then s may be divided into five pieces $s = uvxyz$ satisfying the conditions

1. for each $i \geq 0$, $uv^ixy^iz \in L$
2. $|vy| > 0$, and
3. $|vxy| \leq p$

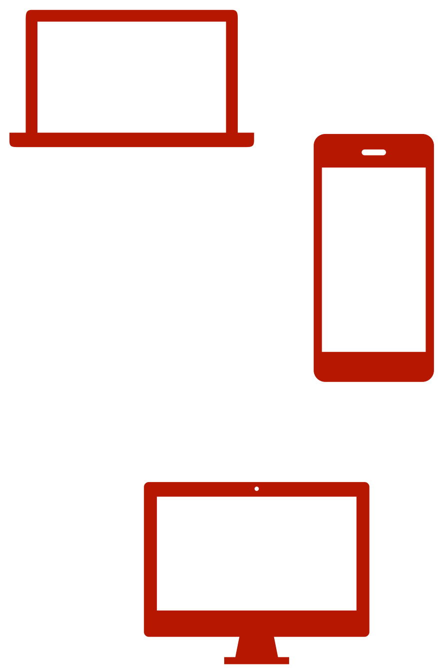
Chomsky Hierarchy

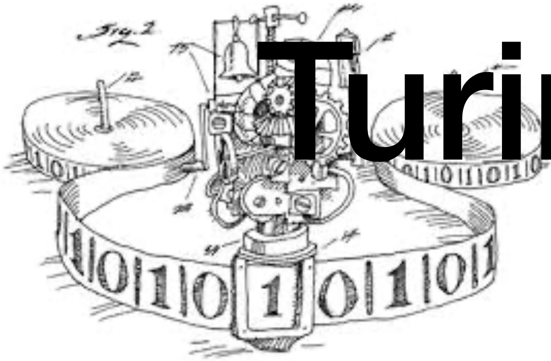




Today

- Turing machines





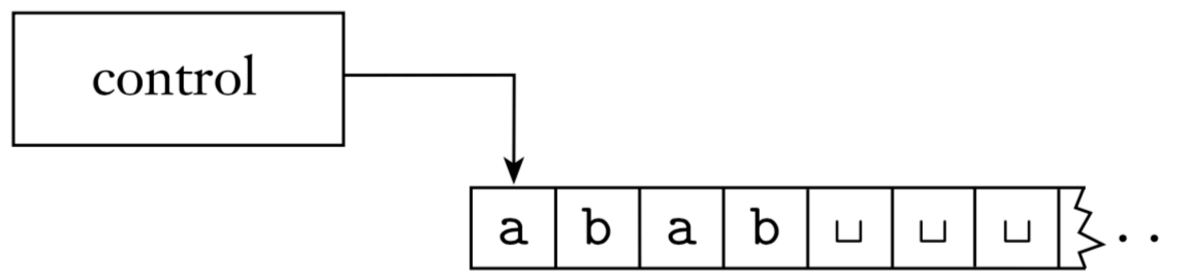
Turing machine (TM)

1912-1954

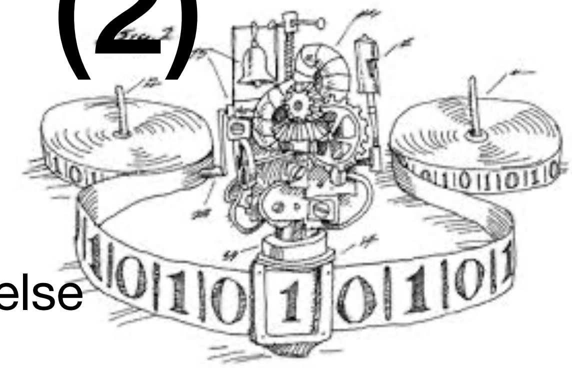
- Much more powerful model than FA/PDA
- First proposed by Alan Turing in 1936
- Similar to finite automaton but:
  unlimited & unrestricted memory
- More accurate model of a general purpose computer
  - a Turing machine can do everything a (classical) computer can do

What does a Turing machine look like?

- **Infinite tape** representing its *unlimited memory*
- **Tape head** can
  - read symbols
  - write symbols
  - move around on the tape



What does a Turing machine look like? (2)



Computation

- Initially: tape contains only input string; blank everywhere else
- If TM needs to store information, it may write this information on tape
- To read information that TM has written, machine can move its head back over it
- TM continues computing until it decides to produce an output
- Outputs ‘accept’ and ‘reject’ are obtained by entering designated accepting and rejecting states
- If TM does not enter accepting or rejecting state, computation will continue (infinite loop)

Differences between finite automata and Turing machines

TM	FA
1. TM can both write on and read from tape	1. FA can only read from tape
2. Read–write head can move both left and right	2. Read head can move right
3. Tape is infinite	3.
4. Special states for rejecting and accepting take effect immediately (no need to finish reading the input)	4. Special state accepting takes effect only after finishing reading of input

How does a TM operate? Example

TM M for testing membership for language $B = \{w\#w \mid w \in \{0,1\}^*\}$

- M can move back and forth over input and make marks on it
- M accepts if its input is a member of B
  - that is: input consists of two identical strings separated by symbol #
- M rejects otherwise
- Strategy: go forth and back to the corresponding places on the two sides of the # and determine whether or not they match
- To keep track of which places correspond, M places marks on tape:
  - M crosses off each pair of symbols as it is examined
  - If M crosses off all the symbols, then everything is matched successfully and M accepts
  - If M discovers a mismatch, M rejects

Example of a TM

$$B = \{w\#w|\ w \in \{0,1\}^*\}$$

- Input: 011000#011000
- Tape

  0 1 1 0 0 0 # 0 1 1 0 0 0

  ↑

Example of a TM

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

- Input: 011000#011000

- Tape

0 1 1 0 0 0 # 0 1 1 0 0 0
↑

Cross
out 0, move to the
right and look for next
symbol right of #.
If 0 then cross out and
move left.

Example of a TM

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

- Input: 011000#011000
- Tape

x 1 1 0 0 0 # 0 1 1 0 0 0

↑

Cross
out 0, move to the
right and look for next
symbol right of #.
If 0 then cross out and
move left.

Example of a TM

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

- Input: 011000#011000

- Tape

x 1 1 0 0 0 # x 1 1 0 0 0

Cross
out 0, move to the
right and look for next
symbol right of #.
If 0 then cross out and
move left.

Example of a TM

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

- Input: 011000#011000
- Tape

x 1 1 0 0 0 # x 1 1 0 0 0

Find first symbol not crossed out

Example of a TM

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

- Input: 011000#011000
- Tape

x 1 1 0 0 0 # x 1 1 0 0 0

Find first symbol not crossed out

Example of a TM

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

- Input: 011000#011000
- Tape

x 1 1 0 0 0 # x 1 1 0 0 0

Cross out 1, move to the right and look for next symbol right of #. If 1 then cross out and move left.

Example of a TM

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

- Input: 011000#011000
- Tape

x x 1 0 0 0 # x 1 1 0 0 0

↑

Cross out 1, move to the right and look for next symbol right of #. If 1 then cross out and move left.

Example of a TM

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

- Input: 011000#011000
- Tape

x x 1 0 0 0 # x 1 1 0 0 0

Cross
out 1, move to the
right and look for next
symbol right of #.
If 1 then cross out and
move left.

Example of a TM

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

- Input: 011000#011000
- Tape

x x 1 0 0 0 # x x 1 0 0 0

Cross out 1, move to the right and look for next symbol right of #. If 1 then cross out and move left.

Example of a TM

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

- Input: 011000#011000

- Tape

x x 1 0 0 0 # x x 1 0 0 0

Find first symbol not crossed out

Example of a TM

$$B = \{w\#w |\ w \in \{0,1\}^*\}$$

- Input: 011000#011000

- Tape

  x x 1 0 0 0 # x x 1 0 0 0

Cross
out 1, move to the
right and look for next
symbol right of #.
If 1 then cross out and
move left.

Example of a TM

$$B = \{w\#w|\ w \in \{0,1\}^*\}$$

- Input: 011000#011000
- Tape

x x x 0 0 0 # x x 1 0 0 0

↑

Cross
out 1, move to the
right and look for next
symbol right of #.
If 1 then cross out and
move left.

Example of a TM

$$B = \{w\#w | \ w \in \{0,1\}^*\}$$

- Input: 011000#011000
- Tape

x x x 0 0 0 # x x 1 0 0 0

Cross
out 1, move to the
right and look for next
symbol right of #.
If 1 then cross out and
move left.

Example of a TM

$$B = \{w\#w |\ w \in \{0,1\}^*\}$$

- Input: 011000#011000
- Tape

x x x 0 0 0 # x x x 0 0 0

Cross
out 1, move to the
right and look for next
symbol right of #.
If 1 then cross out and
move left.

Example of a TM

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

- Input: 011000#011000
- Tape

x x x 0 0 0 # x x x 0 0 0

↑

Find first symbol not crossed out

Example of a TM

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

- Input: 011000#011000
- Tape

x x x 0 0 0 # x x x 0 0 0

↑

Find first symbol not crossed out

Example of a TM

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

- Input: 011000#011000
- Tape

x x x 0 0 0 # x x x 0 0 0

Cross
out 0, move to the
right and look for next
symbol right of #.
If 0 then cross out and
move left.

Example of a TM

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

- Input: 011000#011000
- Tape

x x x 0 0 0 # x x x 0 0 0

Etc.

Example of a TM

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

- Input: 011000#011000
- Tape

x x x x x x # x x x x x x

Etc.

Example of a TM

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

- Input: 011000#011000
- Tape

x x x x x x # x x x x x x

Etc.

Example of a TM

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

- Input: 011000#011000
- Tape

x x x x x x # x x x x x x ␣

Accept

Turing machine Formal Definition

A **Turing machine** (**TM**) is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ where

- Q, Σ, Γ are finite sets
  - Q: set of states
  - Σ: input alphabet **not** containing *blank symbol* $\sqcup$
  - Γ: tape alphabet; $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$
- $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$: transition function
- $q_0 \in Q$: start state
- $q_{accept} \in Q$: accept state
- $q_{reject} \in Q$: reject state with $q_{reject} \neq q_{accept}$

Consider transition function: Is TM deterministic or nondeterministic?

Deterministic vs non-deterministic

- **Deterministic function:** always returns same result for same input values
  - Transition function, for a well-defined input, is uniquely defined
- **Nondeterministic function:** may return different results for different calls for same input values
  - Transition function returns a set of possible outcomes

Turing machine Formal Definition

A **Turing machine** (**TM**) is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ where

- Q, Σ, Γ are finite sets
    - Q: set of states
    - Σ: input alphabet not containing *blank symbol* $\sqcup$
    - Γ: tape alphabet; $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$: transition function
- $q_0 \in Q$: start state
- $q_{accept} \in Q$: accept state
- $q_{reject} \in Q$: reject state with $q_{reject} \neq q_{accept}$

Transition function of TM is deterministic

Configuration of a Turing machine

- Given TM M, a **configuration** of M consists of a description of:
  - Its current state
  - Its current tape content
  - Its current head location
- For a state q and strings $u, v \in \Gamma^*$, we write $\boldsymbol{u\ q\ v}$ for the configuration where
  - Current state is q
  - Current tape content is uv
  - Current head location is the first symbol of v
  - Tape contains only blanks following the last symbol of v

Computation of the Turing machine

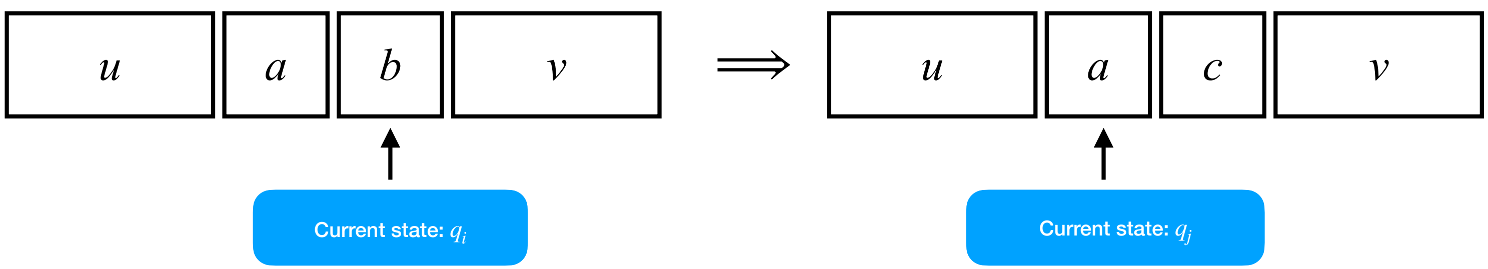
- During computation: changes occur in state, tape contents and head location
- We define
  - Specific configuration(s) of TM
  - Computation: change in configuration

Configuration of a Turing machine (2)

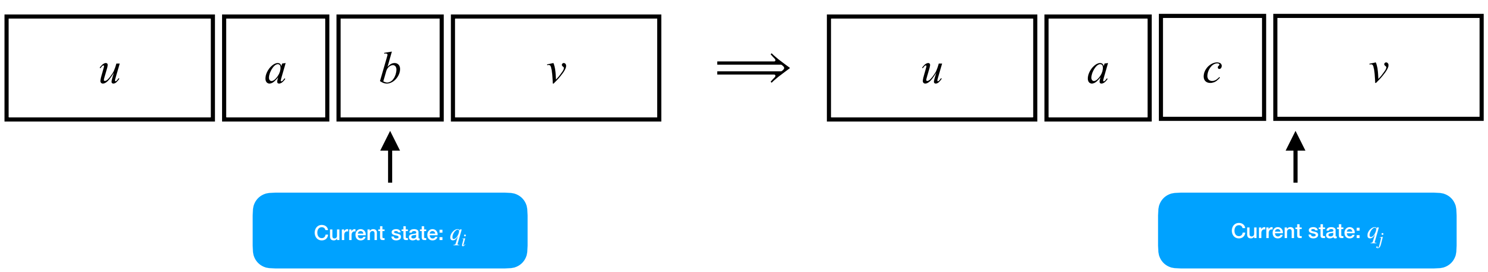
For TM M, let $a, b, c \in \Gamma$, $u,v \in \Gamma^*$ and $q_i, q_j \in Q$, and let $ua\ q_i\ bv$, $u\ q_j\ acv$ and $uac\ q_j\ v$ be **configurations**

We say

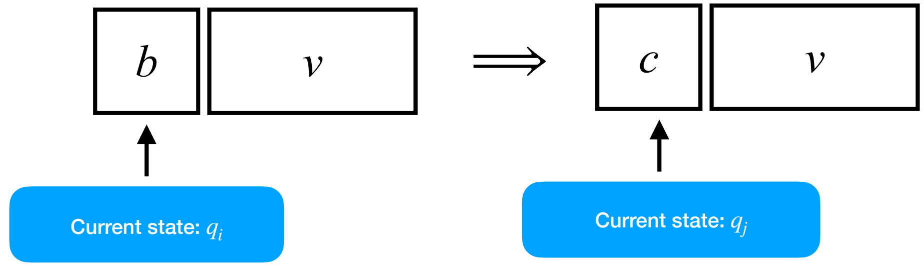
- $ua\ q_i\ bv$ **yields** $u\ q_j\ acv$ if
  - $\delta(q_i, b) = (q_j, c, L)$ (ie M moves leftward)



- $ua\ q_i\ bv$ **yields** $uac\ q_j\ v$ if
  - $\delta(q_i, b) = (q_j, c, R)$ (ie M moves rightward)



Configuration of a Turing machine—special cases



- **Left-hand end** (head is at leftmost cell)

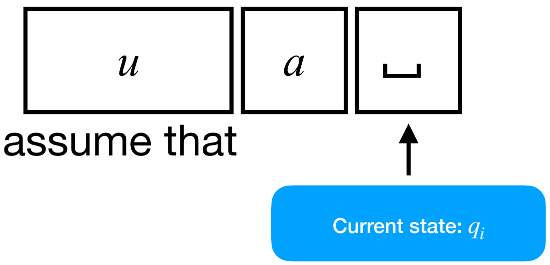
$\delta(q_i, b) = (q_j, c, L)$

  - configuration $q_i\, bv$ yields configuration $q_j\, cv$ if transition is left-moving prevents M from going off left-hand end of tape)

$\delta(q_i, b) = (q_j, c, R)$

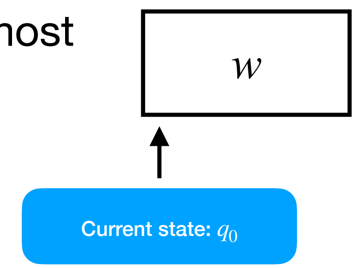
  - $q_i\, bv$ yields $c\ q_j\, v$ if transition is right-moving

As expected



- **Right-hand end**: configuration $ua\ q_i$ is equivalent to $ua\ q_i \sqcup$ because we assume that blanks follow part of tape represented in configuration

- **Start configuration** $q_0 w$: input is w, M is in start state q_0 with its head at leftmost position on the tape



- **Halting configurations**
  - **Accepting configuration**: the state is q_{accept}
  - **Rejecting configuration**: state is q_{reject}

Computation of a Turing machine

- TM M *accepts* input w if a sequence of configurations $C_1, C_2, \ldots, C_k$ exists, where
    - C_1 is start configuration of M on input w
    - each C_i yields C_{i+1}, and
    - C_k is an accepting configuration
- **Language** $L(M)$ of M, or the **language recognized** by M, is the collection of all strings that M accepts

Possible outcomes of a computation

- **Possible outcomes of a TM** on input w
  - *accept*
  - *reject*
  - *loop:* machine also fails to accept w
- A TM that halts on every input is called a **decider**

Turing machines & their languages

- If a language L is recognized by some TM, we call L **Turing-recognizable**

- We call a language L **Turing-decidable** (or **decidable**) if there exists a decider M that recognizes L

  - We also say that M **decides** L

- Note: every Turing-decidable language is also Turing-recognizable

Can a decider enter an infinite loop?

Decidable Languages

- Let $L = \{0^{2^n} | n \geq 0\}$. We describe M that decides L
- On input string $w \in \{0\}^*$

Implementation description

1. Sweep left to right, crossing off every other 0
2. If in step 1 the tape contains exactly one 0: accept
3. If in step 1 the tape contains more than one 0, and the number of zeros is odd: reject
4. Otherwise: return the head to the left-hand end of the tape
5. Go to step 1

Example: Formal description of a TM M

$$L = \{0^{2^n} | n \geq 0\}$$

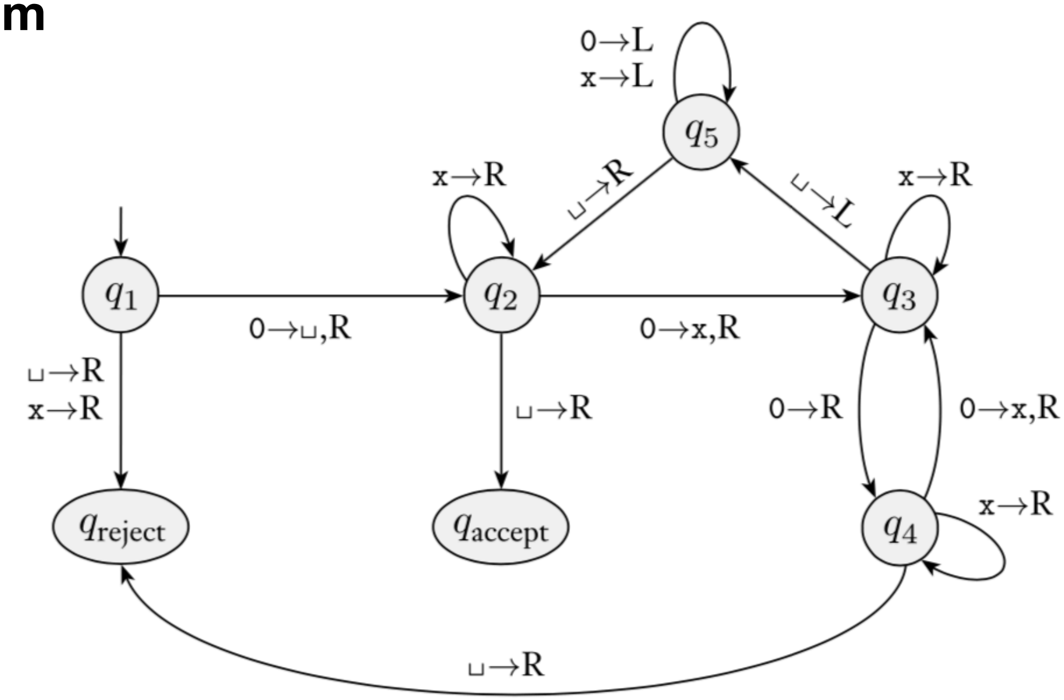
Let $M = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$ with

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{accept}, q_{reject}\}$ where
  - q_1 is start state
  - q_{accept} is accept state and q_{reject} is reject state
- $\Sigma = \{0\}$
- $\Gamma = \{0, \text{x}, \sqcup\}$

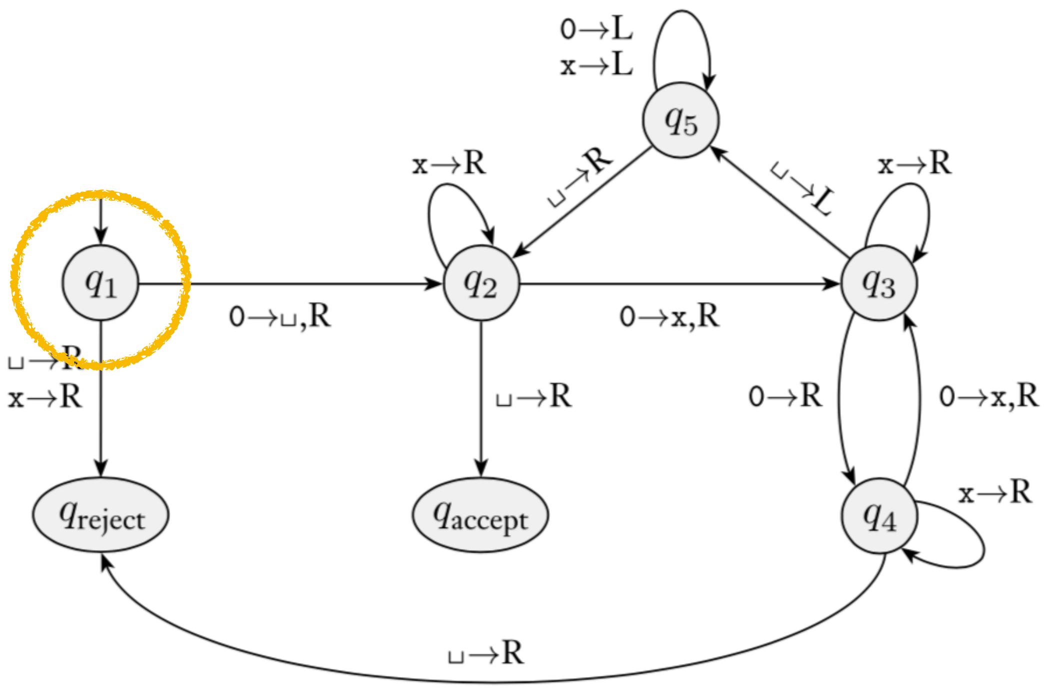
δ next slide

Formal description of TM M: transition function δ

State diagram

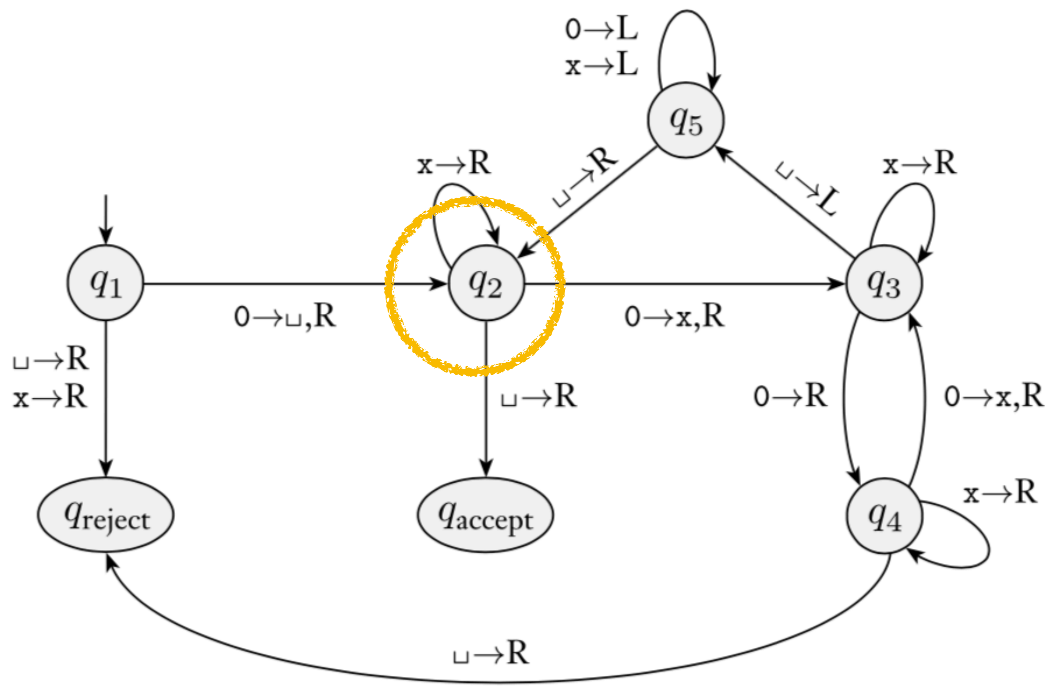


Input: 0000



tape: 0000

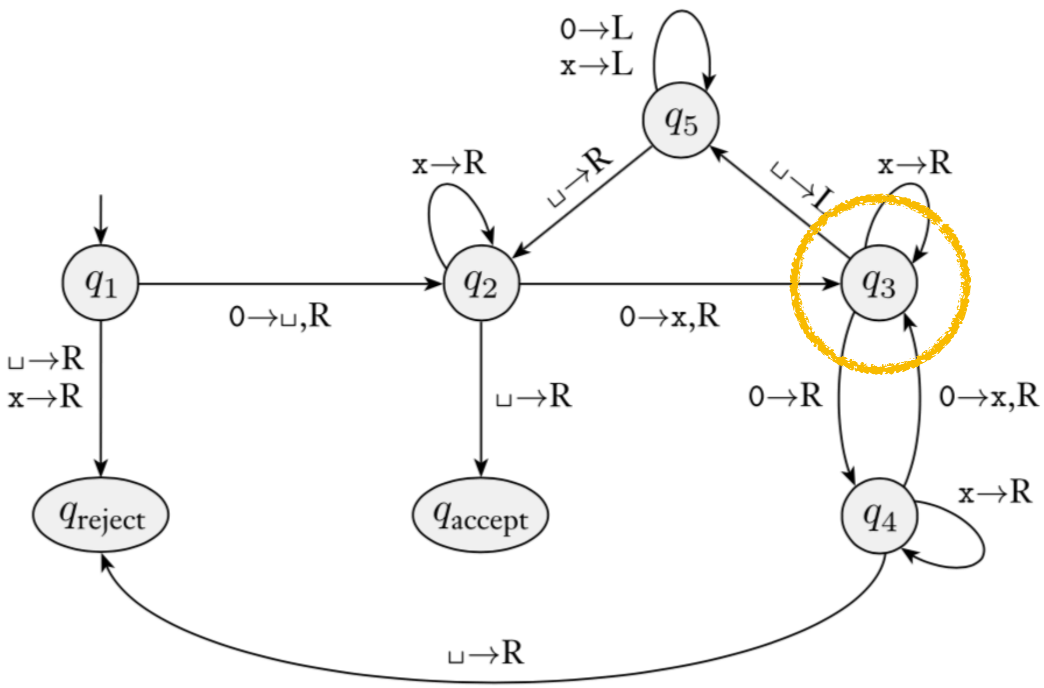
Input: 0000



tape: ␣000

First 0 replaced by ␣ to mark left-hand

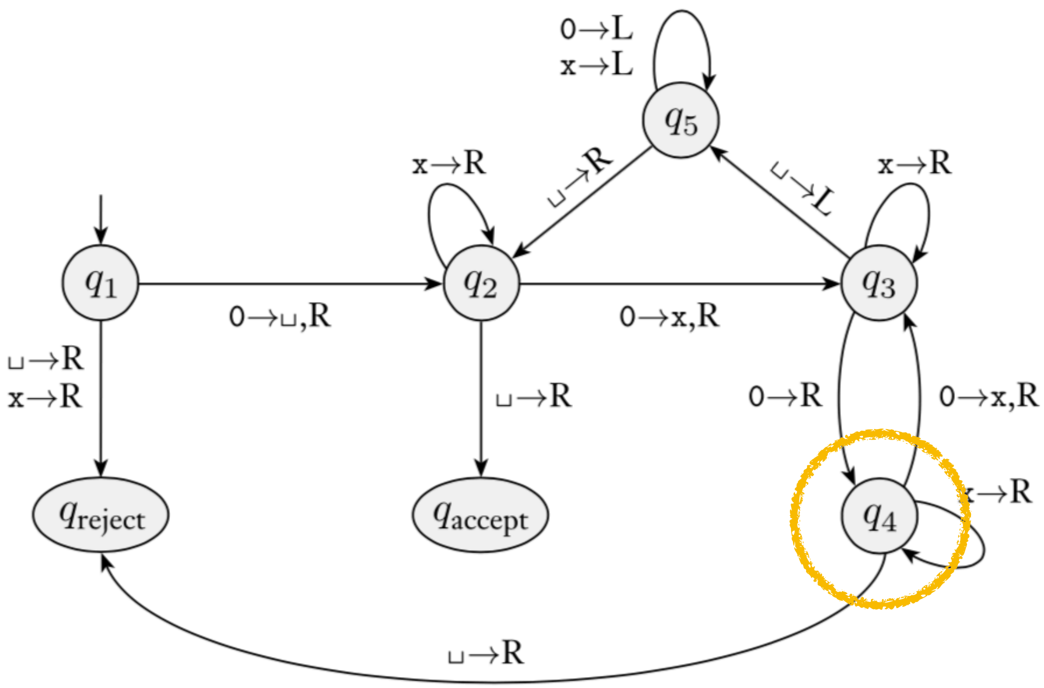
Input: 0000



tape: ⊔x00

Cross out every other 0

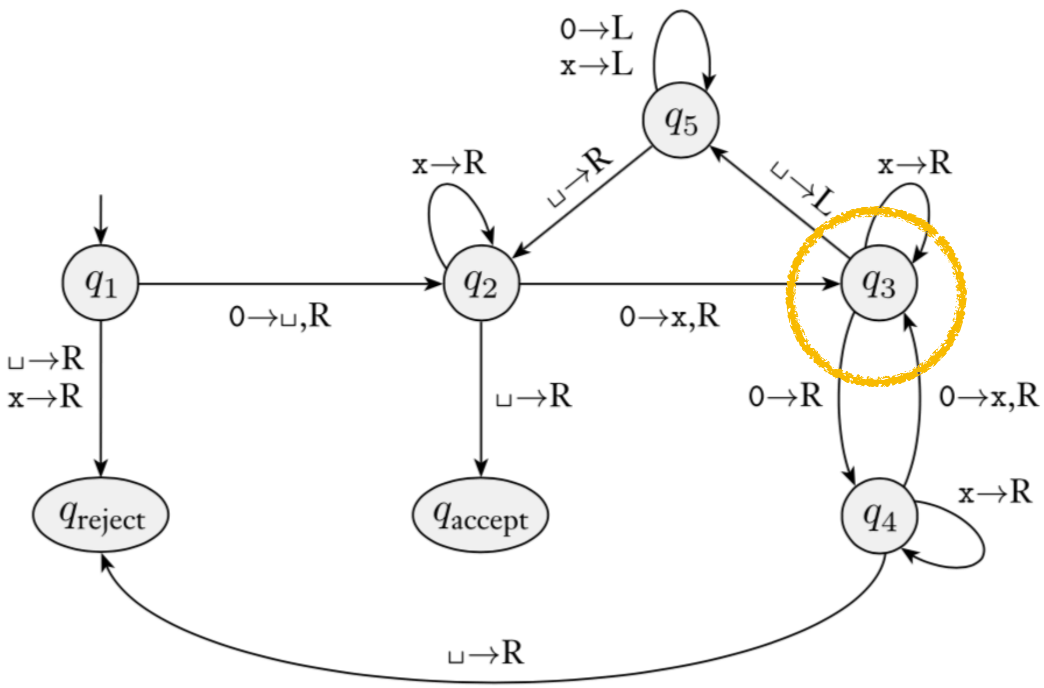
Input: 0000



tape: ⊔x00

Cross out every other 0

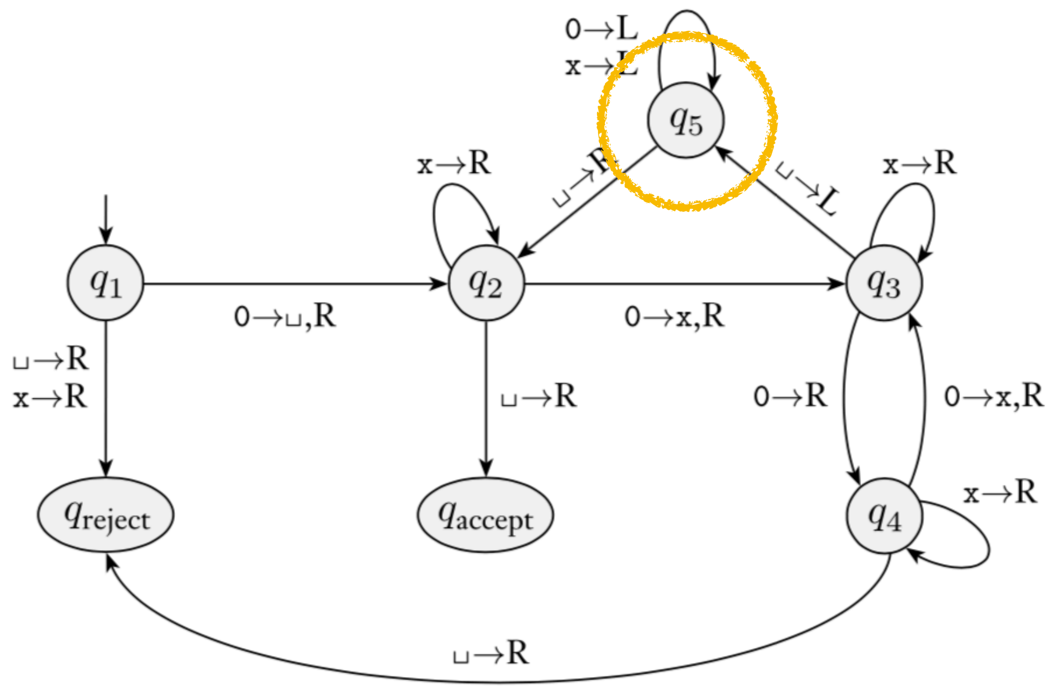
Input: 0000



tape: ␣x0x␣

Cross out every other 0

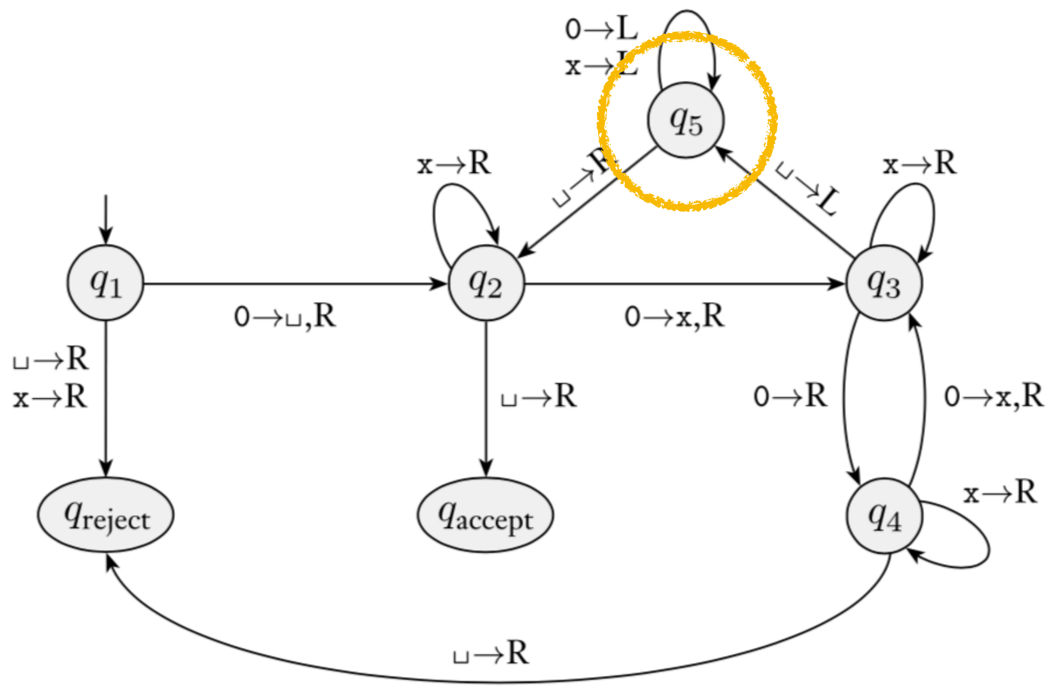
Input: 0000



tape: ⊔x0x⊔

Move to the left

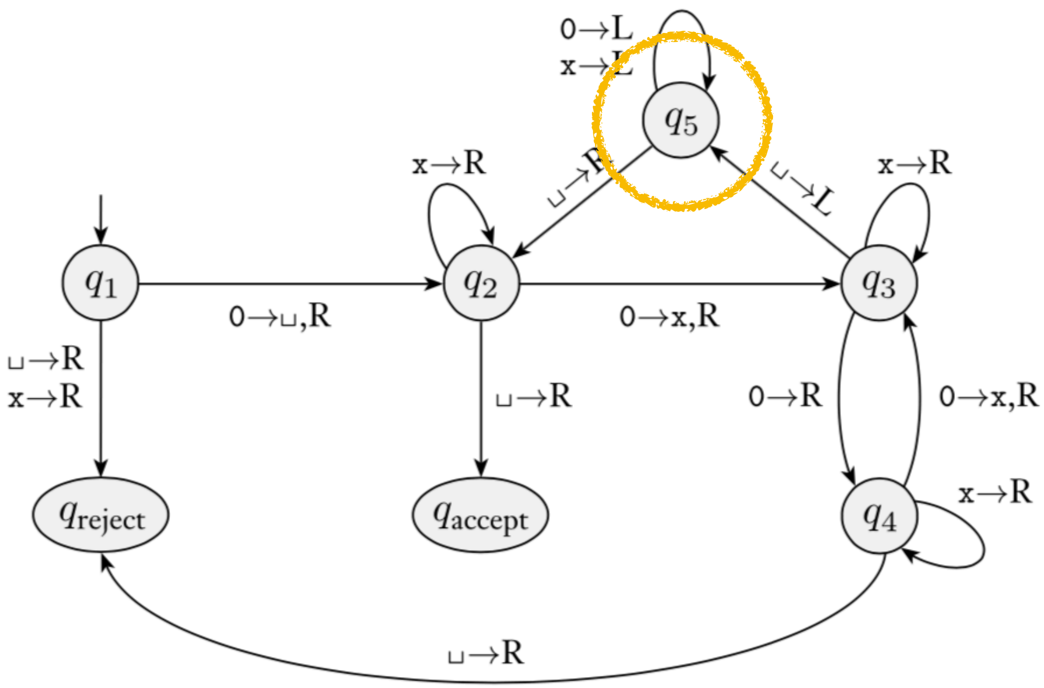
Input: 0000



tape: ␣x0x␣

Move to the left

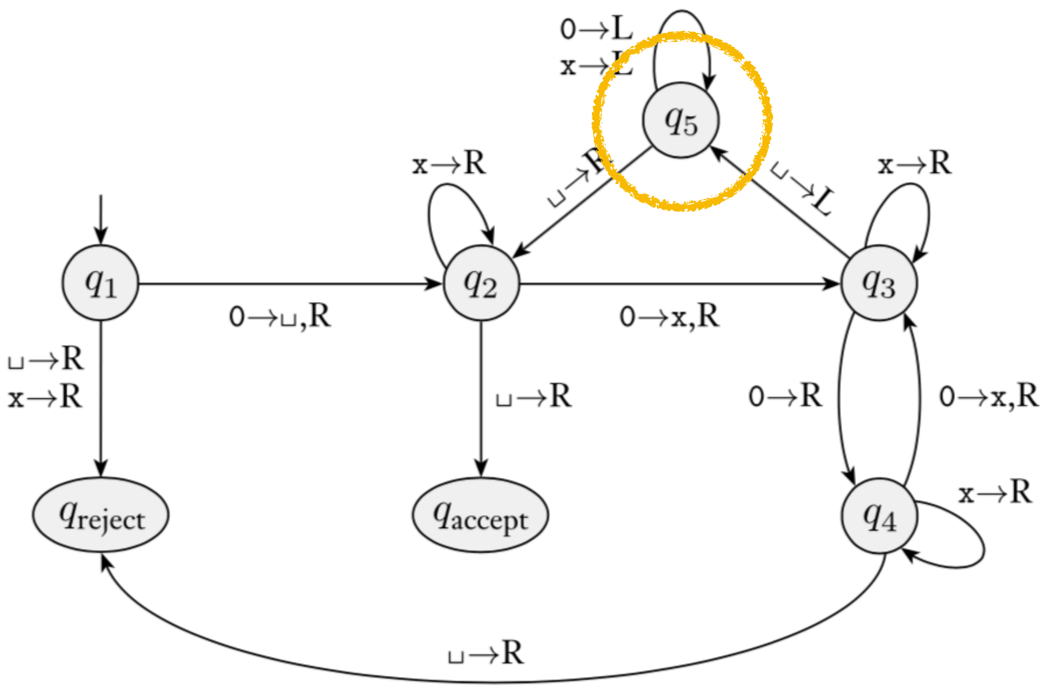
Input: 0000



tape: ␣x0x␣

Move to the left

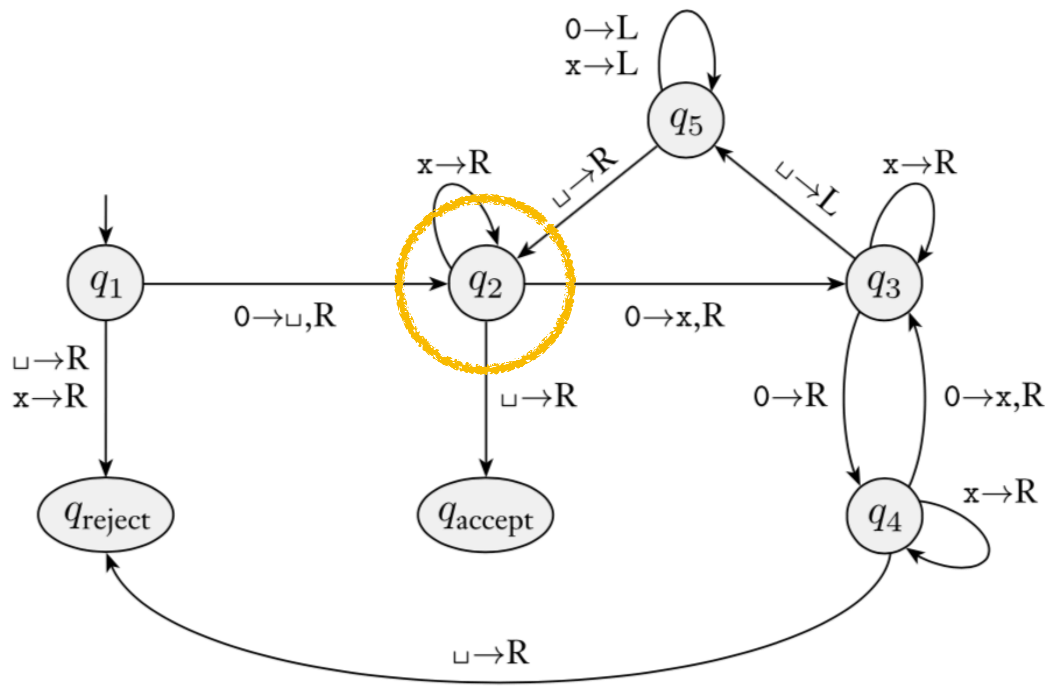
Input: 0000



tape: ⊔x0x⊔

Move to the left

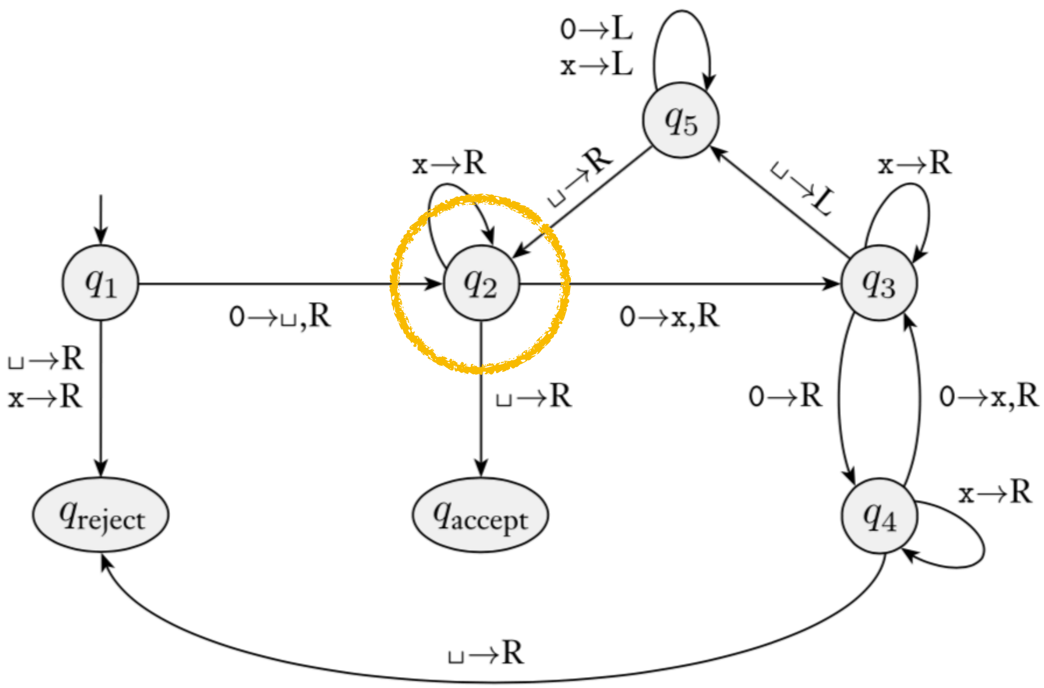
Input: 0000



tape: ⊔x0x⊔

Look for remaining 0s

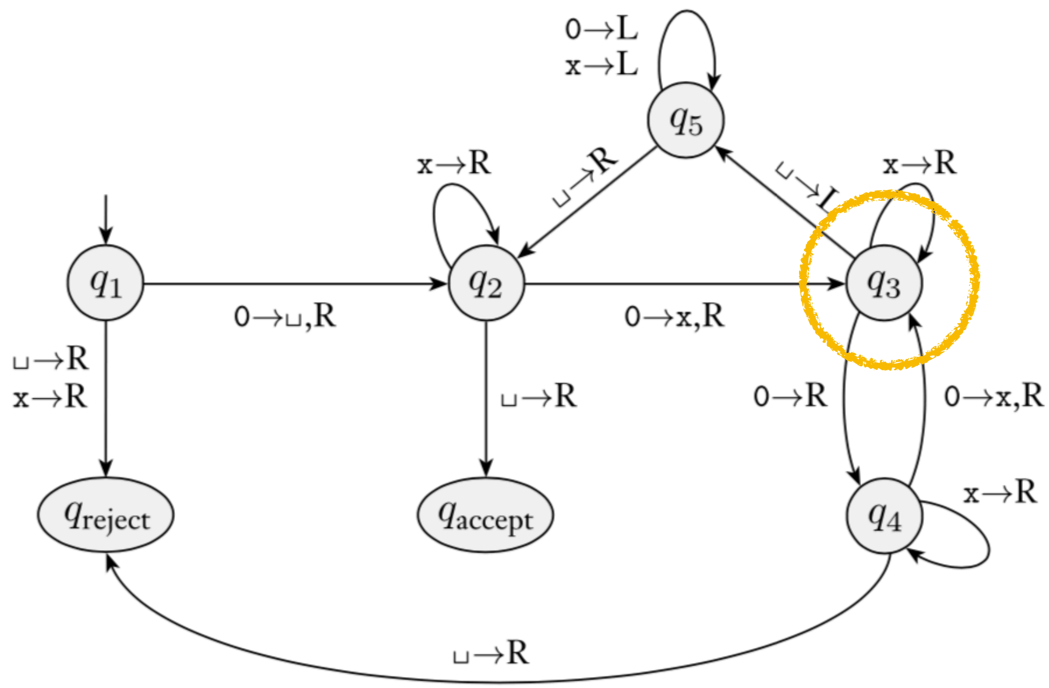
Input: 0000



tape: ␣x0x␣

Cross out every other remaining 0

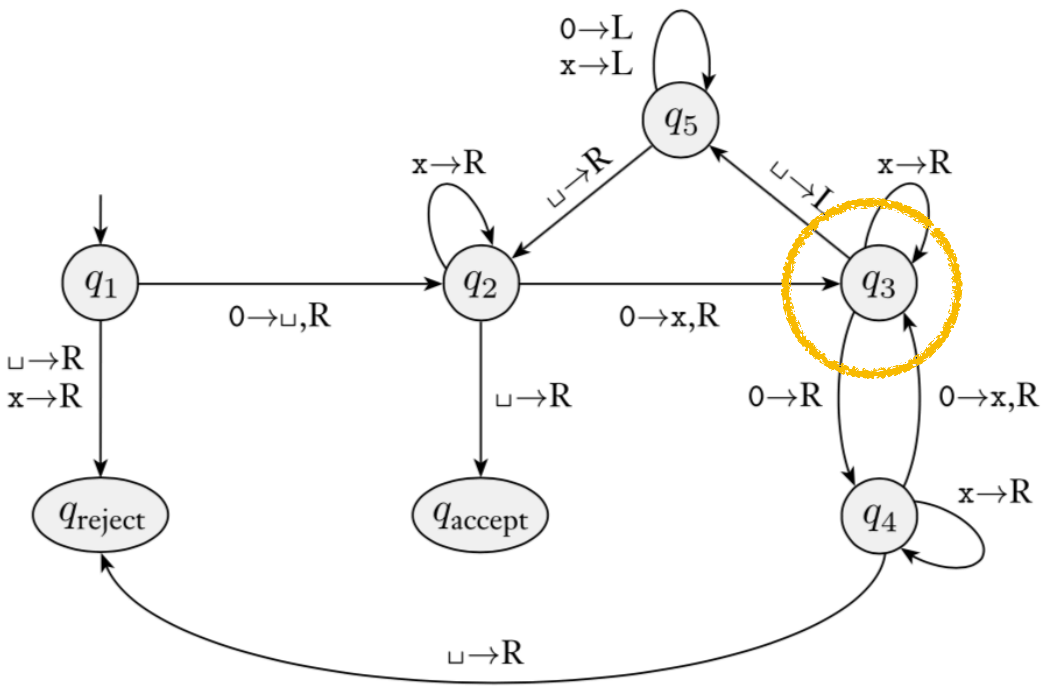
Input: 0000



tape: ⊔xxx⊔

Cross out every other remaining 0

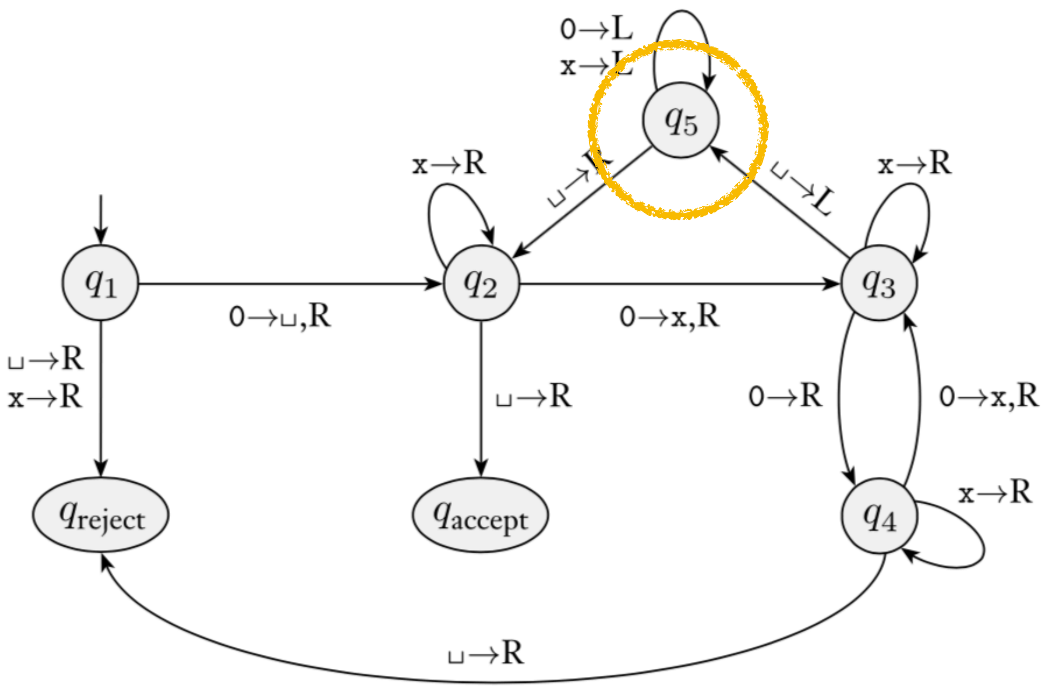
Input: 0000



tape: ␣xxx␣

Move to the left

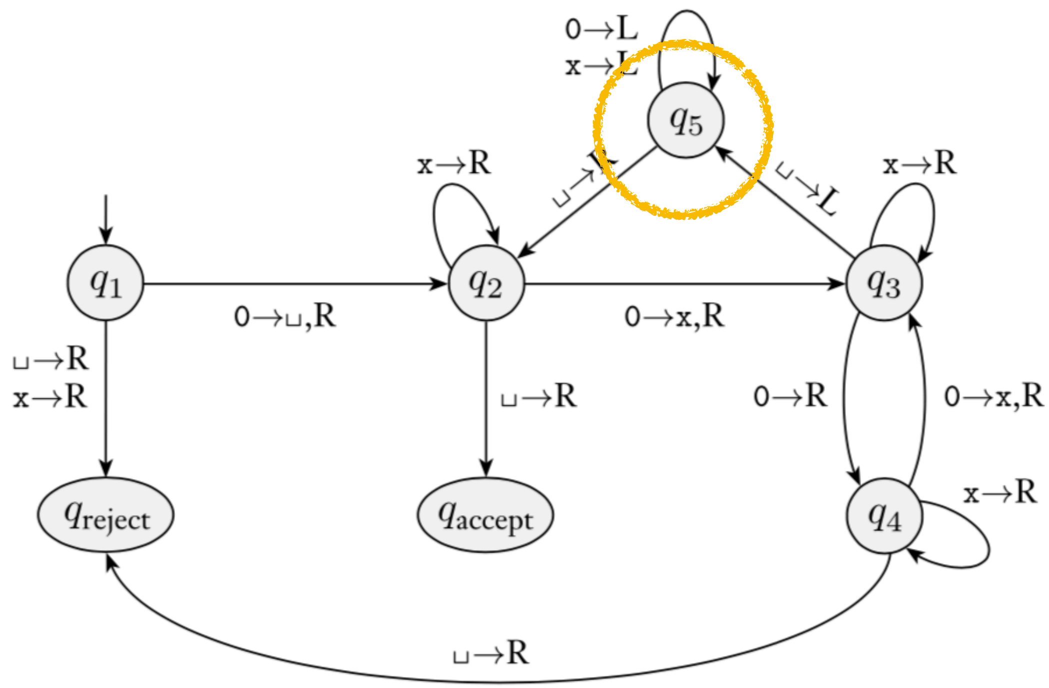
Input: 0000



tape: ⊔xxx⊔

Move to the left

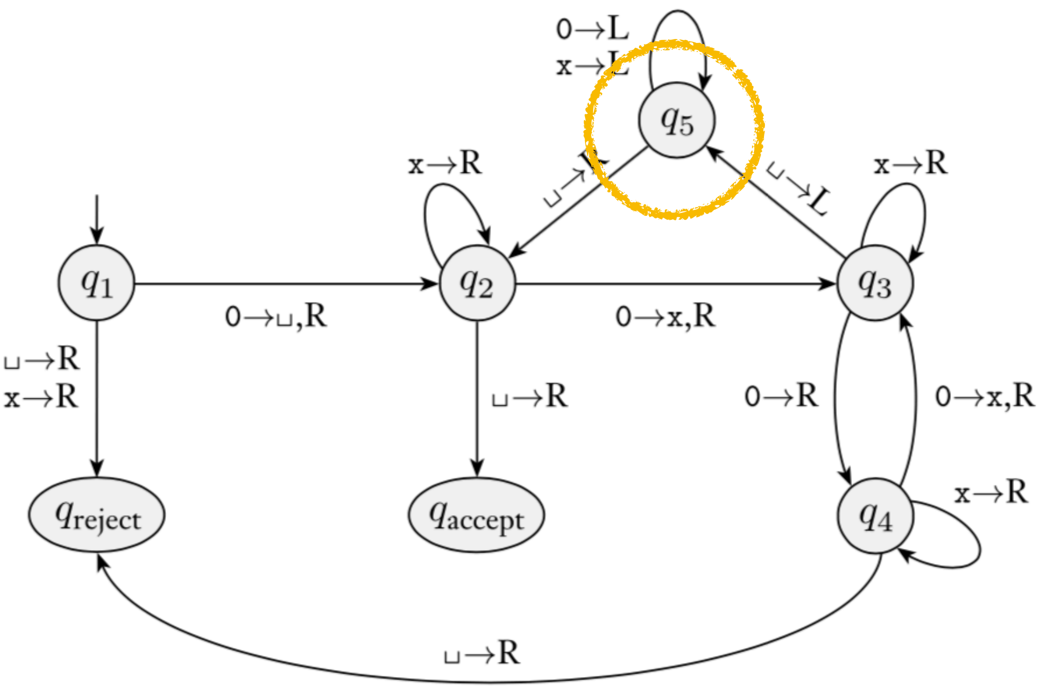
Input: 0000



tape: ␣xxx␣

Move to the left

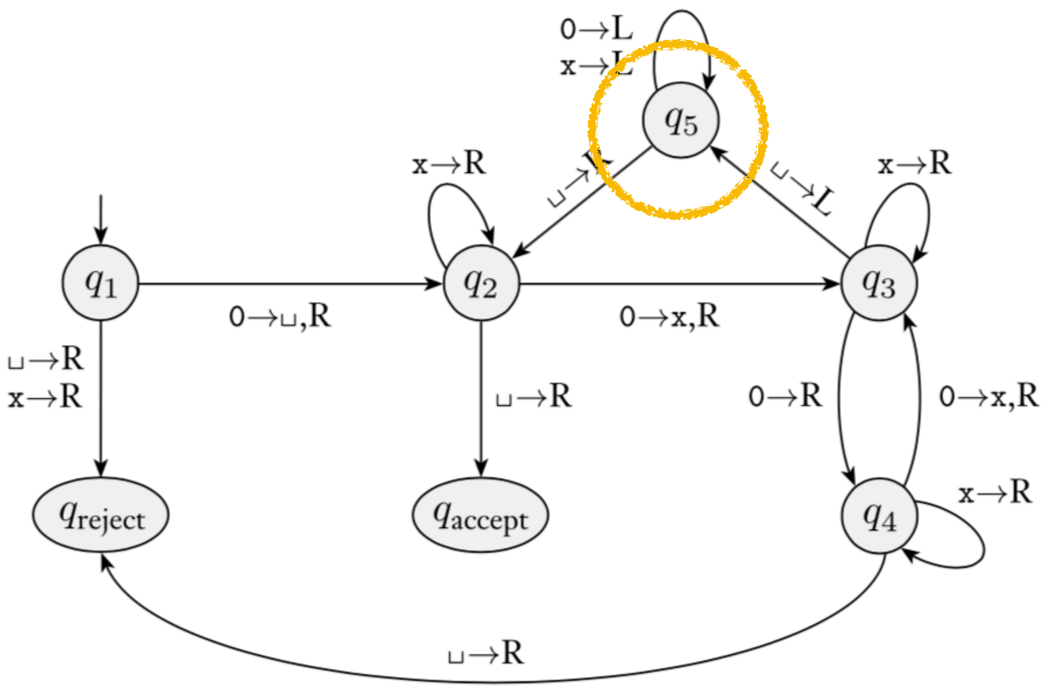
Input: 0000



tape: ␣xxx␣

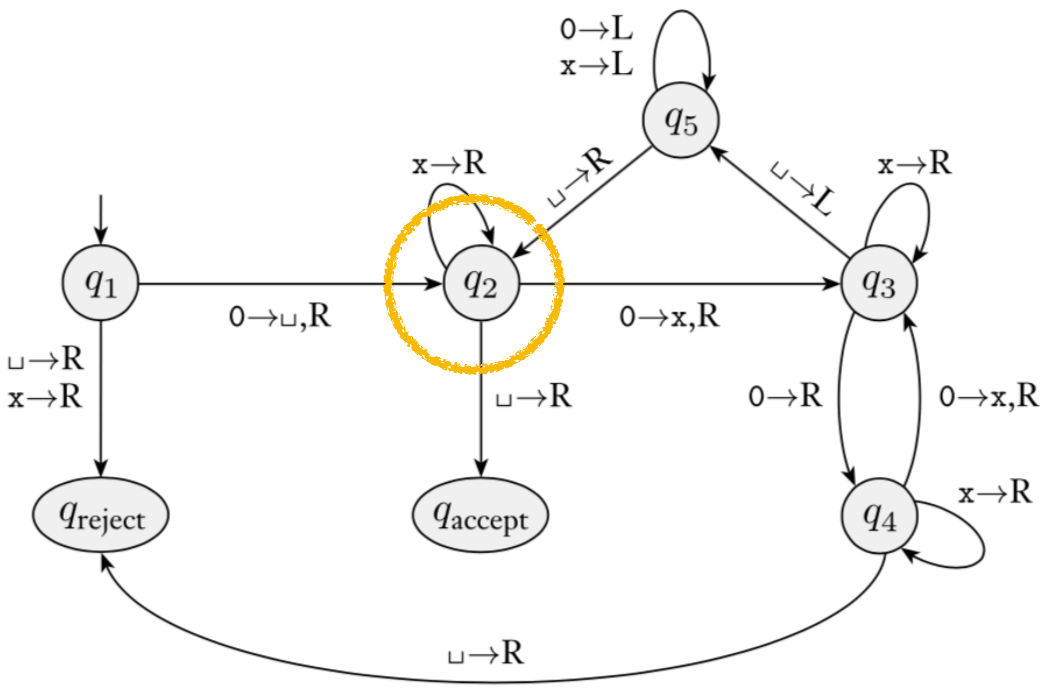
Move to the left

Input: 0000



tape: ␣xxx␣

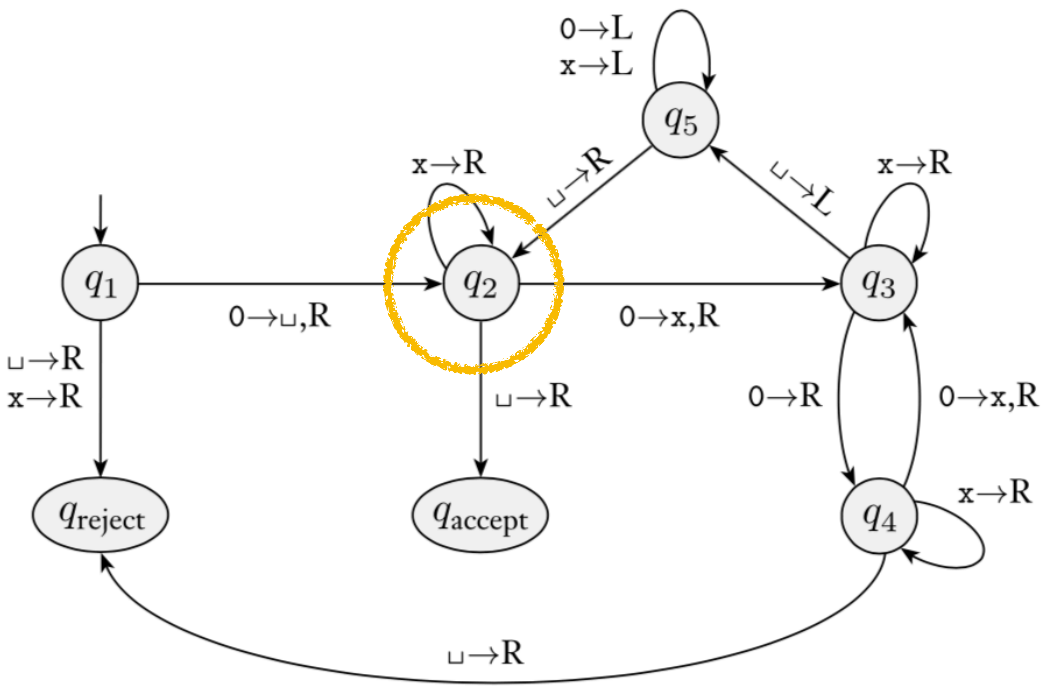
Input: 0000



tape: ⊔xxx⊔

Look for remaining 0s

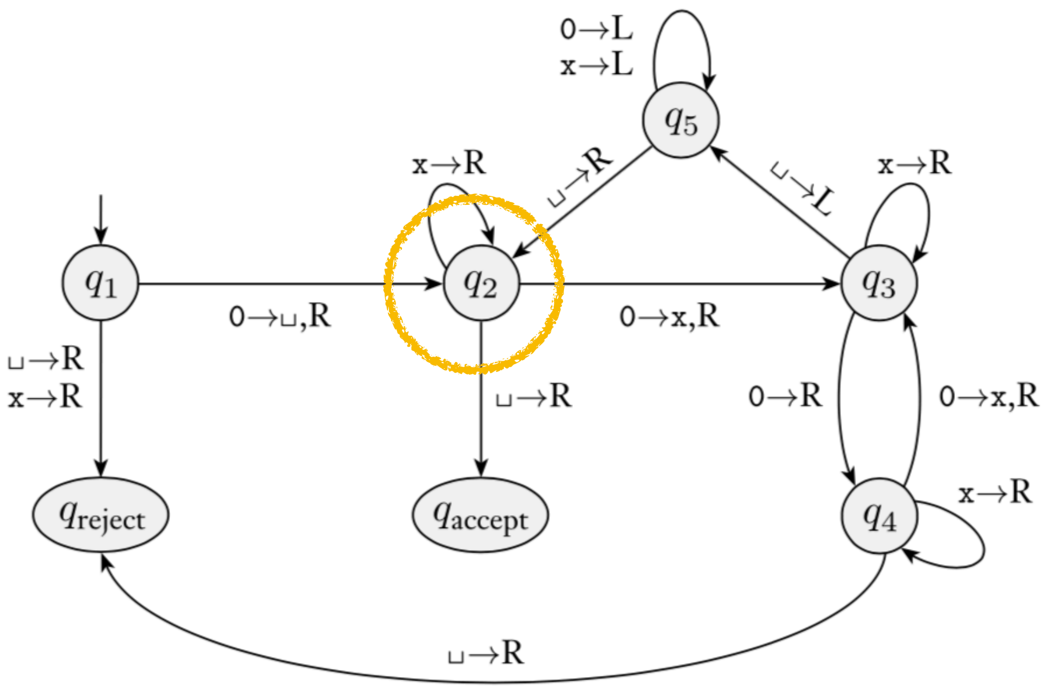
Input: 0000



tape: ␣xxx␣

Look for remaining 0s

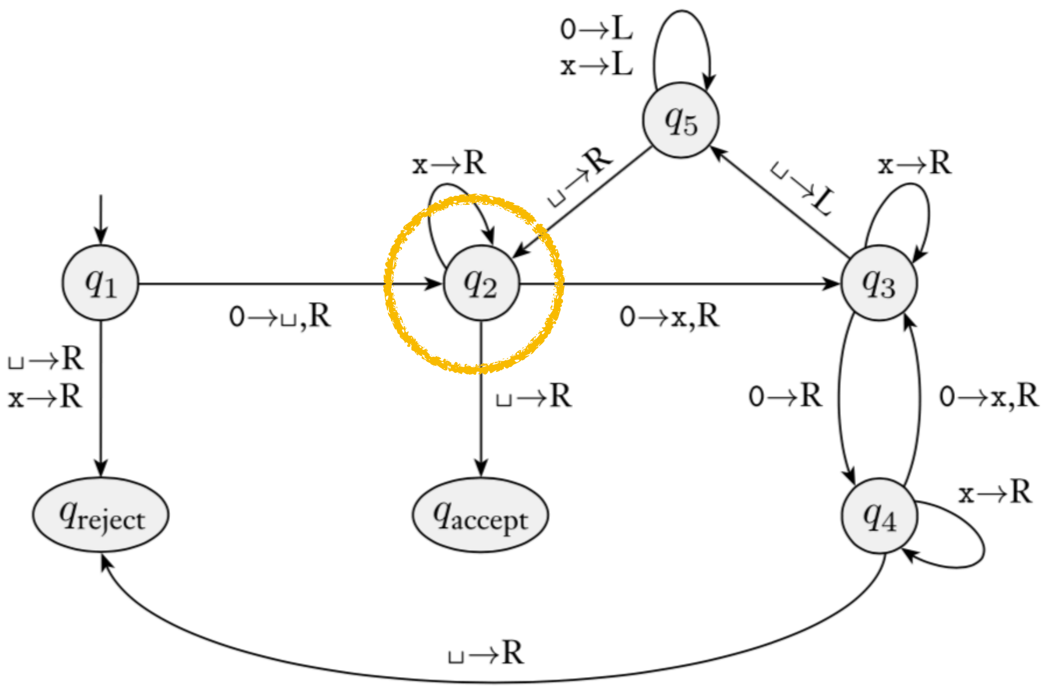
Input: 0000



tape: ⊔xxx⊔

Look for remaining 0s

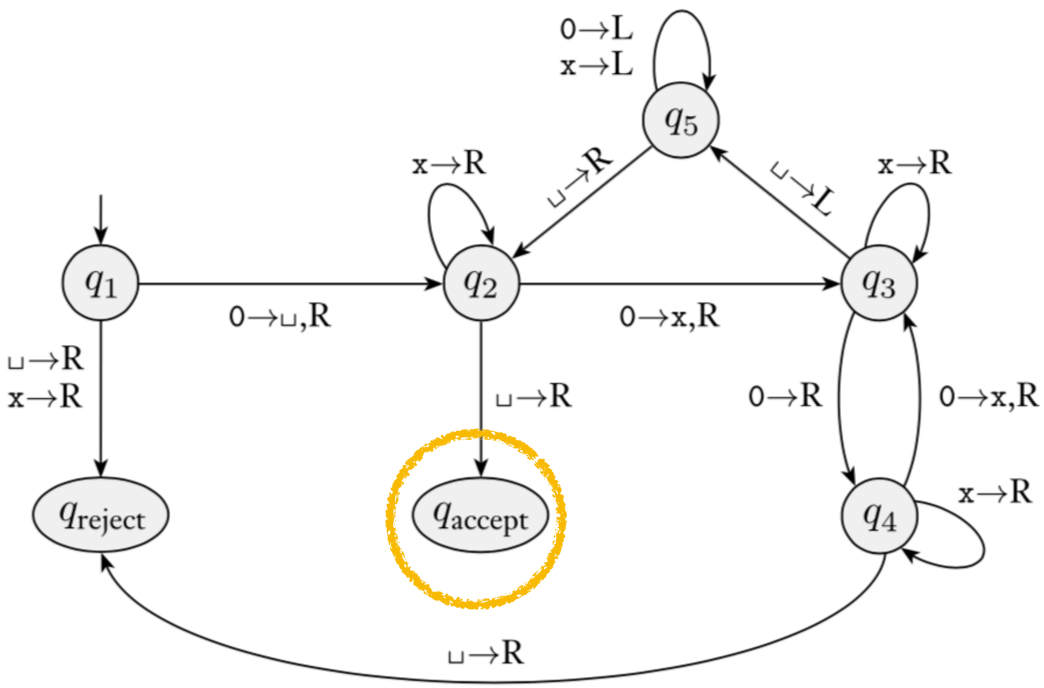
Input: 0000



No 0 left

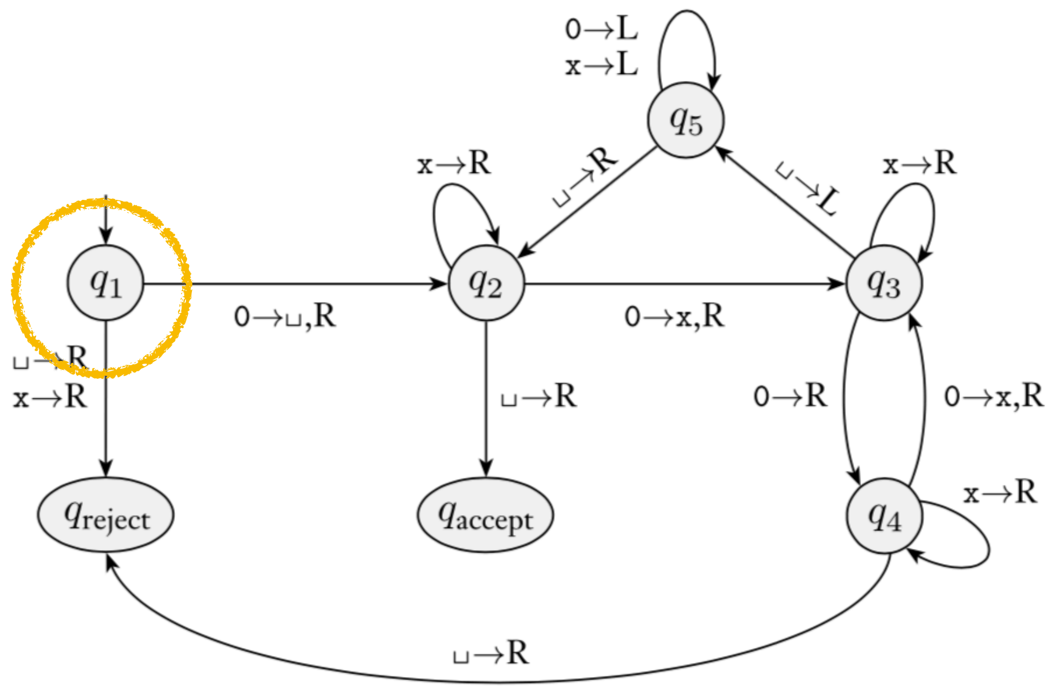
tape: ␣xxx␣

Input: 0000



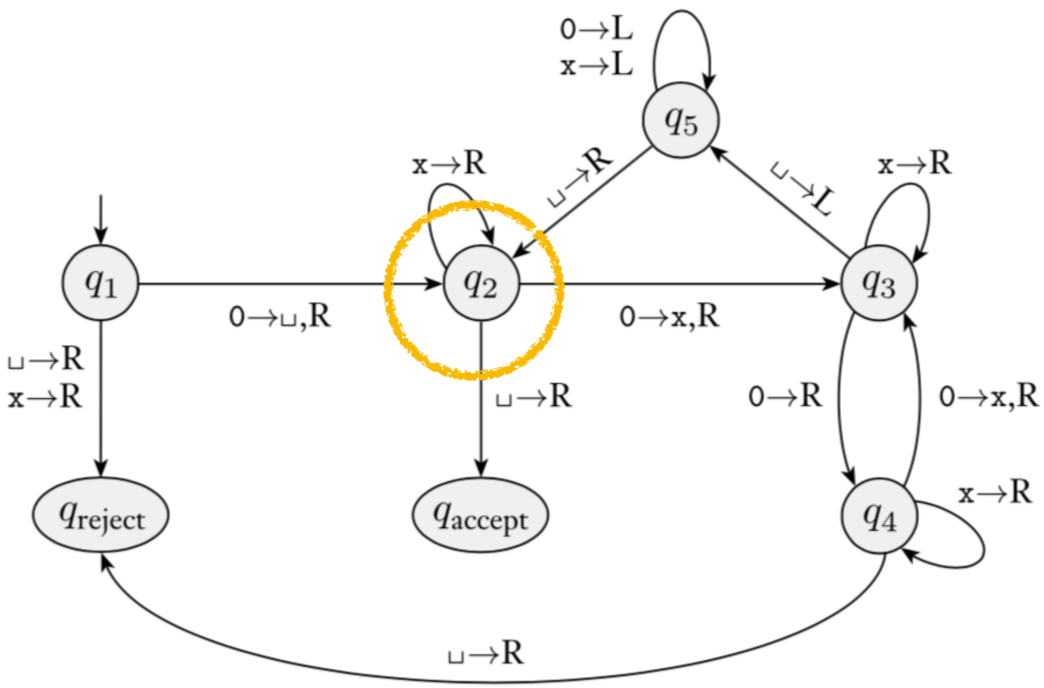
tape: ␣xxx␣␣

Input: 00000



tape: 00000

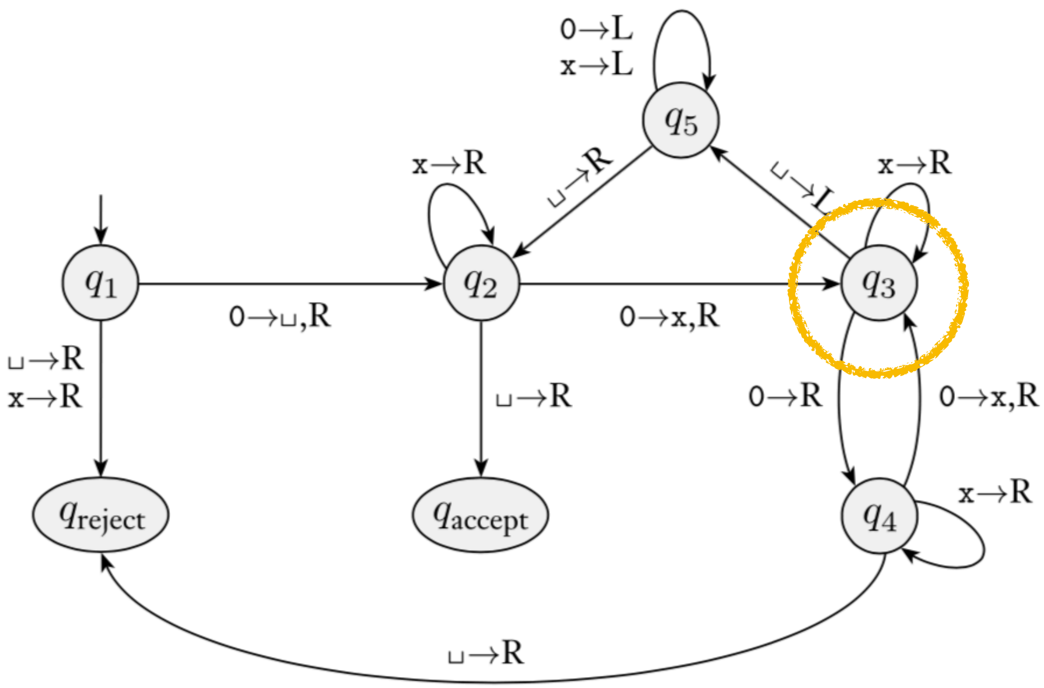
Input: 00000



tape: ⊔0000

First 0 replaced by ⊔ to mark left-hand

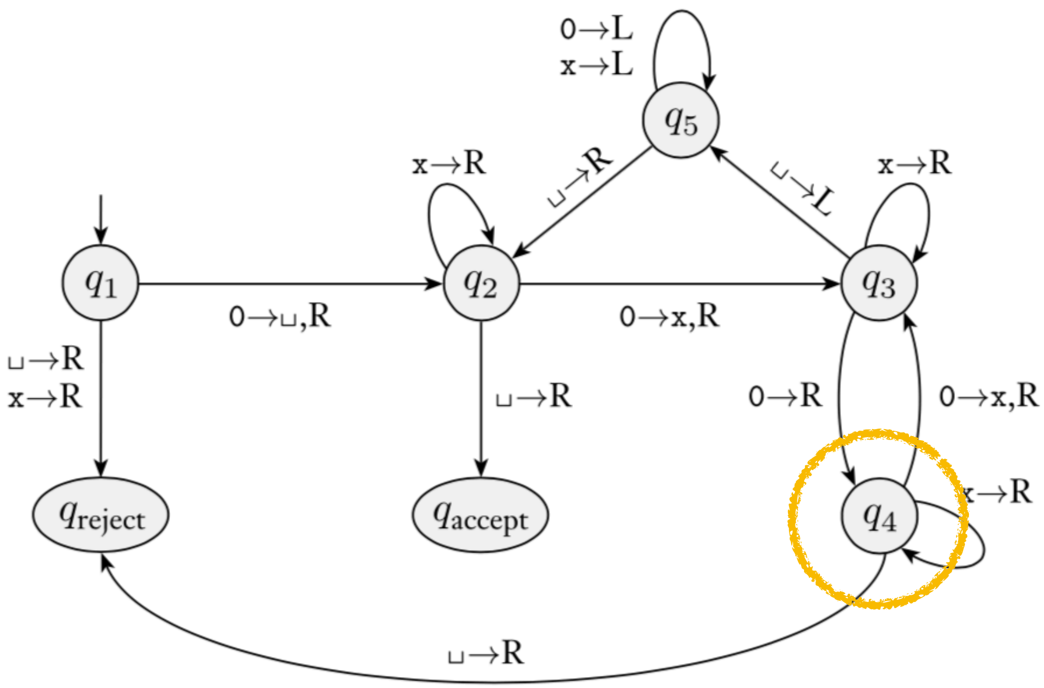
Input: 00000



tape: ⊔x000

Cross out every other 0

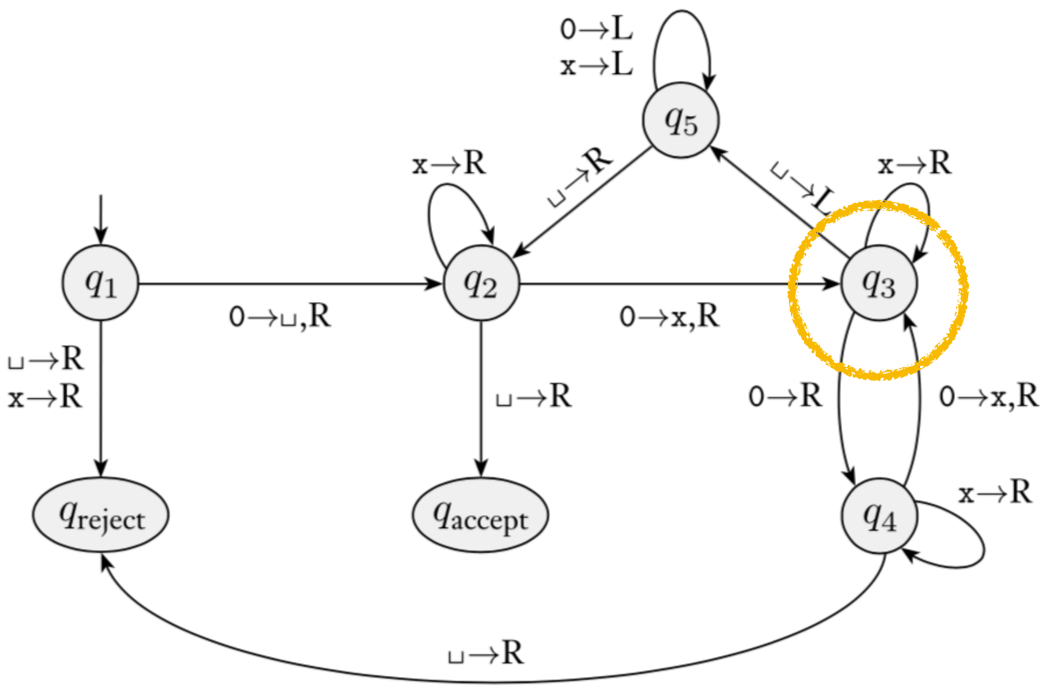
Input: 00000



tape: ␣x000

Cross out every other 0

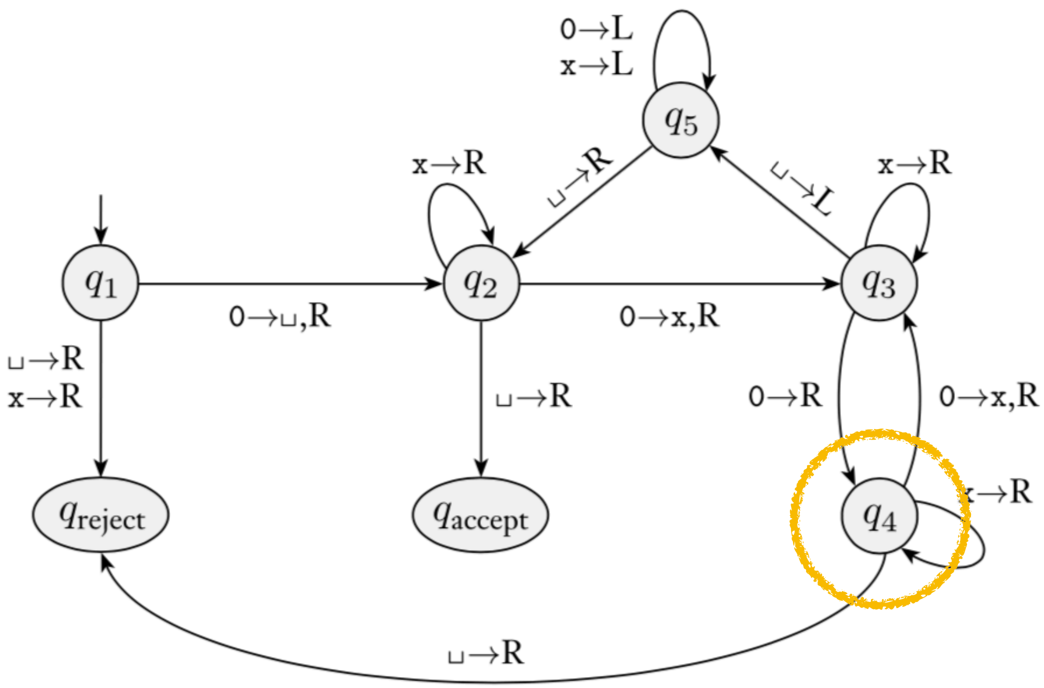
Input: 00000



tape: ␣x0x0

Cross out every other 0

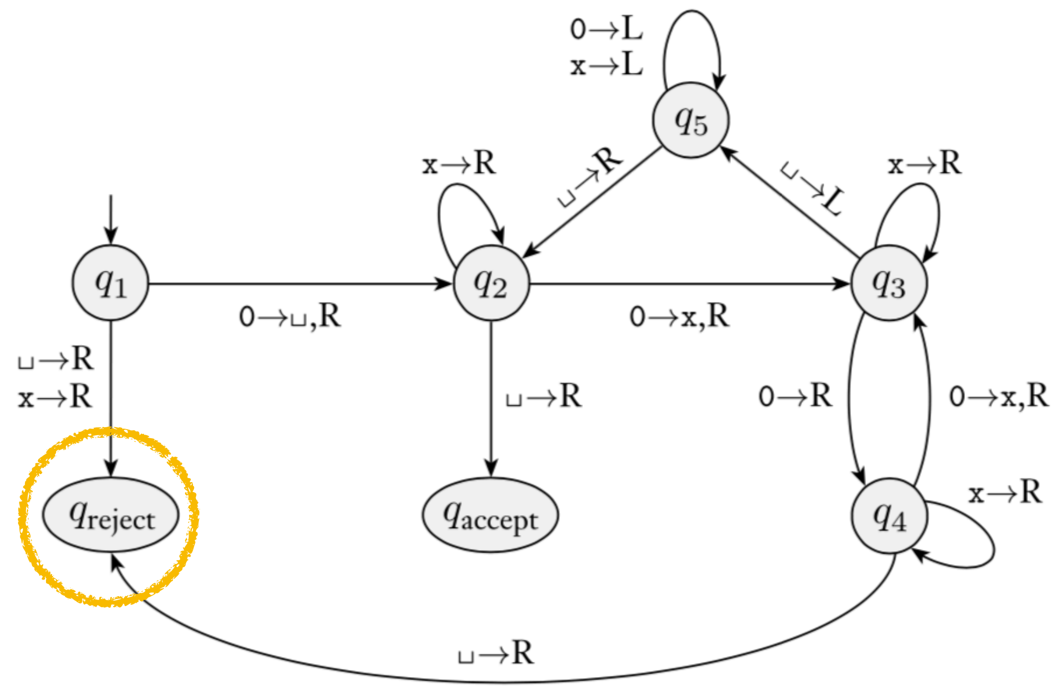
Input: 00000



tape: ␣x0x0␣

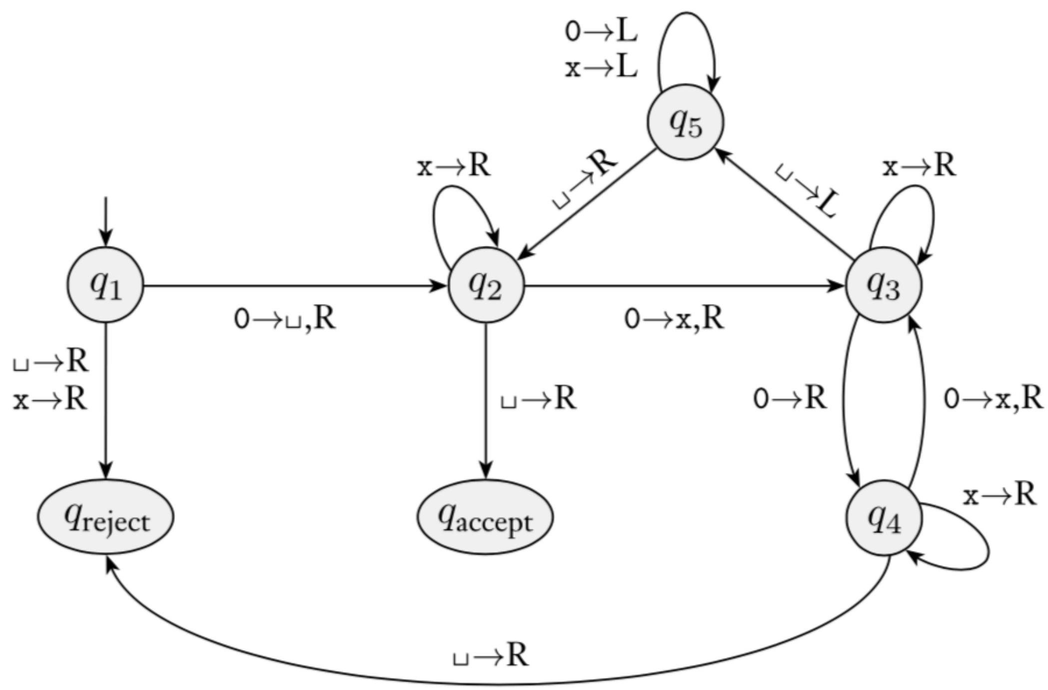
Number if 0s encountered not even

Input: 00000



tape: ␣x0x0␣␣

Your turn!
Input: 000000



tape: 000000

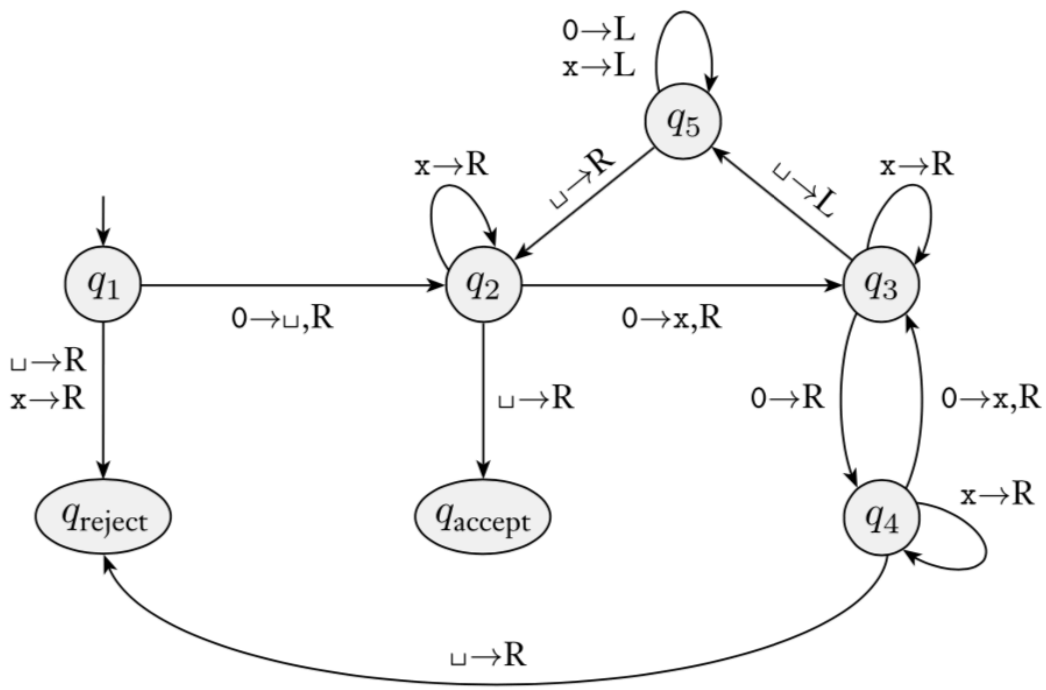
Turing machines & their languages

- If a language L is recognized by some TM, we call L **Turing-recognizable**
- We call a language L **Turing-decidable** (or **decidable**) if there exists a decider M that recognizes L
  - We also say that M **decides** L

Recall—Decider: A TM that halts on every input

- Note: every Turing-decidable language is also Turing-recognizable

Is this TM a decider?



$M = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$

- $\Sigma = \{0\}$
- $\Gamma = \{0, x, \sqcup\}$

Your turn: $L = \{w\#w \mid w \in \{0,1\}^*\}$

- Decider $M = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$ for L
  - $Q = \{q_1, \ldots, q_8, q_{accept}, q_{reject}\}$
  - $\Sigma = \{0,1,\#\}$
  - $\Gamma = \{0,1,\#,\text{x},\sqcup\}$

Describe TM for L as implementation description & as state diagram

$$L = \{\#x_1\#x_2\#\cdots\#x_l \mid \text{each } x_i \in \{0,1\}^* \text{ and } x_i \neq x_j \text{ for each } i \neq j\}$$

- Give an implementation description of a TM that recognizes L

TM we have seen

- Deterministic
- One tape, left-ended, unlimited to the right
- Read/write head

Next

- Variants of TM