Construct TM for the language L ={0^n1^n} where n>=1.

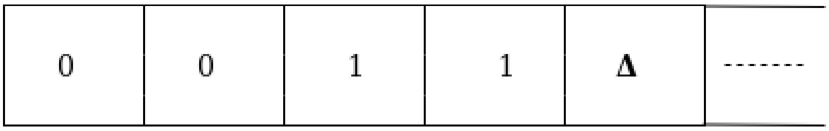
Solution:

We have already solved this problem by PDA. In PDA, we have a stack to remember the previous symbol. The main advantage of the Turing machine is we have a tape head which can be moved forward or backward, and the input tape can be scanned.

The simple logic which we will apply is read out each '0' mark it by A and then move ahead along with the input tape and find out 1 convert it to B. Now, repeat this process for all a's and b's.

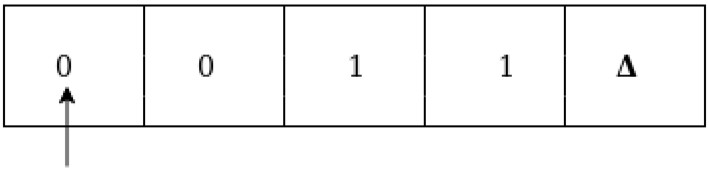
Now we will see how this turing machine work for 0011.

The simulation for 0011 can be shown as below:



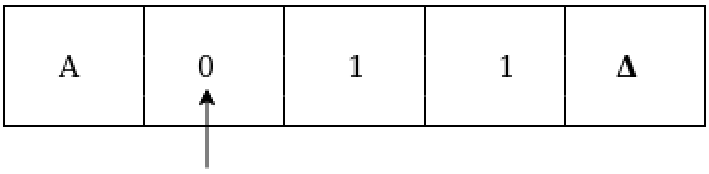
0	0	1	1	Δ	-------

Now, we will see how this turing machine will works for 0011. Initially, state is q0 and head points to 0 as:



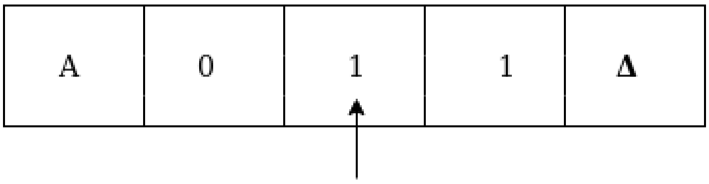
0	0	1	1	Δ

The move will be δ(q0, 0) = δ(q1, A, R) which means it will go to state q1, replaced 0 by A and head will move to the right as:



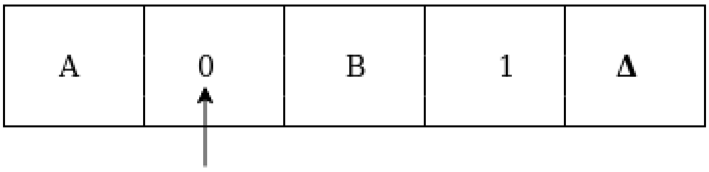
A	0	1	1	Δ

The move will be δ(q1, 0) = δ(q1, 0, R) which means it will not change any symbol, remain in the same state and move to the right as:



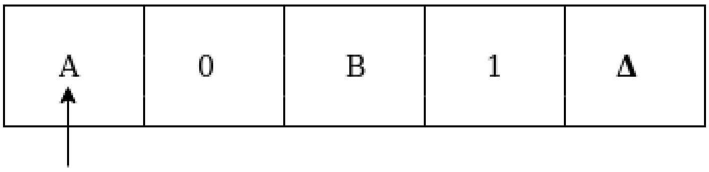
A	0	1	1	Δ

The move will be δ(q1, 1) = δ(q2, B, L) which means it will go to state q2, replaced 1 by B and head will move to left as:



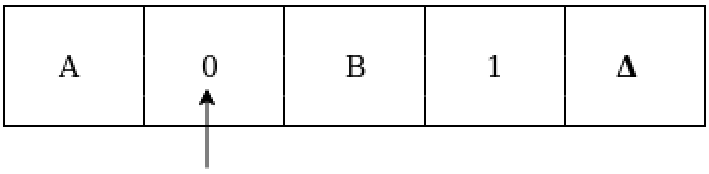
A	0	B	1	Δ

Now move will be δ(q2, 0) = δ(q2, 0, L) which means it will not change any symbol, remain in the same state and move to left as:



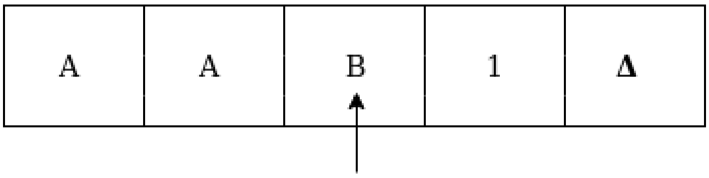
A	0	B	1	Δ

The move will be δ(q2, A) = δ(q0, A, R), it means will go to state q0, replaced A by A and head will move to the right as:



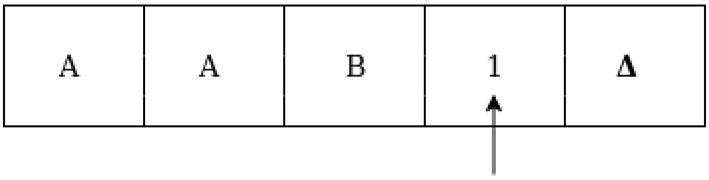
A	0	B	1	Δ

The move will be δ(q0, 0) = δ(q1, A, R) which means it will go to state q1, replaced 0 by A, and head will move to right as:



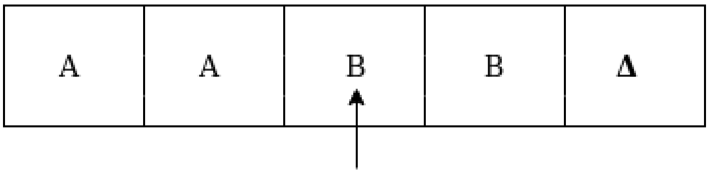
A	A	B	1	Δ

The move will be δ(q1, B) = δ(q1, B, R) which means it will not change any symbol, remain in the same state and move to right as:



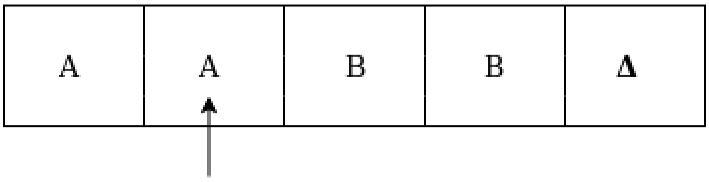
A	A	B	1	Δ

The move will be δ(q1, 1) = δ(q2, B, L) which means it will go to state q2, replaced 1 by B and head will move to left as:

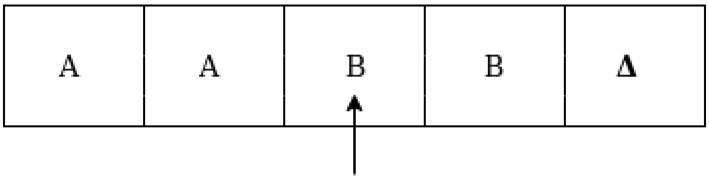


A	A	B	B	Δ

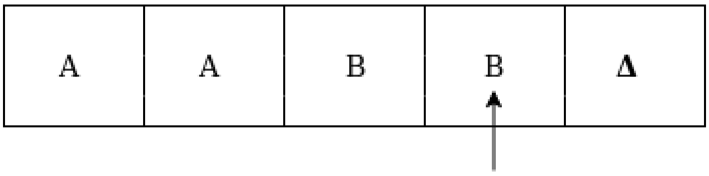
The move δ(q2, B) = (q2, B, L) which means it will not change any symbol, remain in the same state and move to left as:



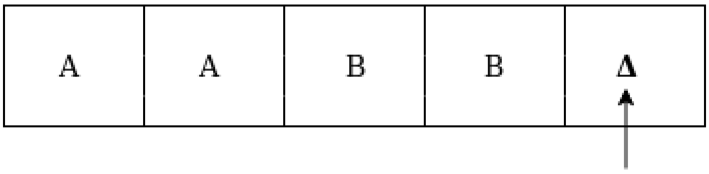
Now immediately before B is A that means all the 0?s are market by A. So we will move right to ensure that no 1 is present. The move will be δ(q2, A) = (q0, A, R) which means it will go to state q0, will not change any symbol, and move to right as:



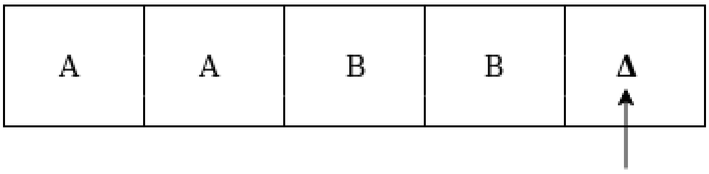
The move δ(q0, B) = (q3, B, R) which means it will go to state q3, will not change any symbol, and move to right as:



The move δ(q3, B) = (q3, B, R) which means it will not change any symbol, remain in the same state and move to right as:



The move δ(q3, Δ) = (q4, Δ, R) which means it will go to state q4 which is the HALT state and HALT state is always an accept state for any TM.



The same TM can be represented by Transition Diagram:

Q. Construct a Turing machine that accepts the language of aba over ∑ = {a, b}.

Solution:

We'll assume the string 'aba' is placed on the input tape as follows:

IMAGE

The tape head will read out the sequence up to the Δ characters. Now, we'll see how this Turing machine works for the string 'aba.'

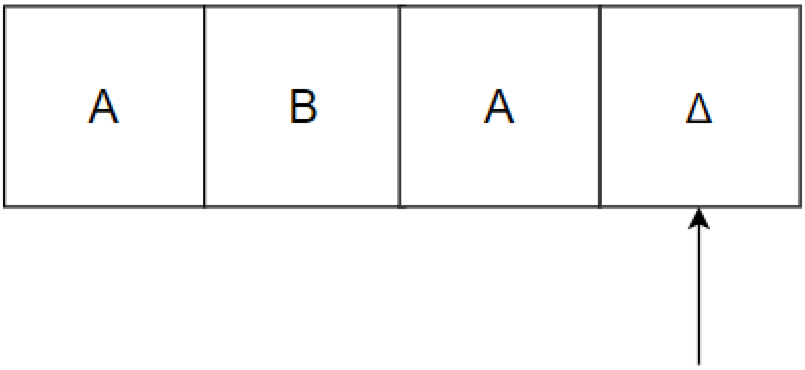
We'll see how this Turing machine works for the string 'aba' now.

1. The initial state is q0, and the head points to an as follows:

2. The move will be δ(q0, a) = δ(q1, A, R), implying that it will proceed to state q1, with a replaced by A and the head moving to the right as follows:

3. The move will be δ(q1, b) = δ(q2, B, R), implying that it will proceed to state q2, with b replaced by B and the head moving to the right as follows:

4. The move will be δ(q2, a) = δ(q3, A, R), meaning it will proceed to state q3, replace a with A, and move the head to the right as follows:



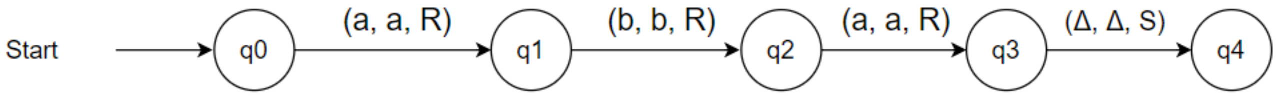
5. The move δ(q3, Δ) = (q4, Δ, S) indicates that it will go to state q4, the HALT state, which is always an accepted state for any TM.

The Transition Table for the above TM is:

States	**a**	**b**	**Δ**
q0	(q1, A, R)	-	-
q1	-	(q2, B, R)	-
q2	(q3, A, R)	-	-

q3	-	-	(q4, Δ, S)
q4	-	-	-

The Transition Diagram for the above TM is:



Construct a Turing Machine for language L = {$0^n1^n2^n$ | n≥1}a

Prerequisite – Turing Machine
The language L = {$0^n1^n2^n$ | n≥1} represents a kind of language where we use only 3 character, i.e., 0, 1 and 2. In the beginning language has some number of 0's followed by equal number of 1's and then followed by equal number of 2's. Any such string which falls in this category will be accepted by this language. The beginning and end of string is marked by $ sign.

Examples –

Input : 0 0 1 1 2 2

Output : Accepted

Input : 0 0 0 1 1 1 2 2 2 2

Output : Not accepted

Assumption: We will replace 0 by X, 1 by Y and 2 by Z

Approach used –
First replace a 0 from front by X, then keep moving right till you find a 1 and replace this 1 by Y. Again, keep moving right till you find a 2, replace it by Z and move left. Now keep moving left till you find a X. When you find it, move a right, then follow the same procedure as above.

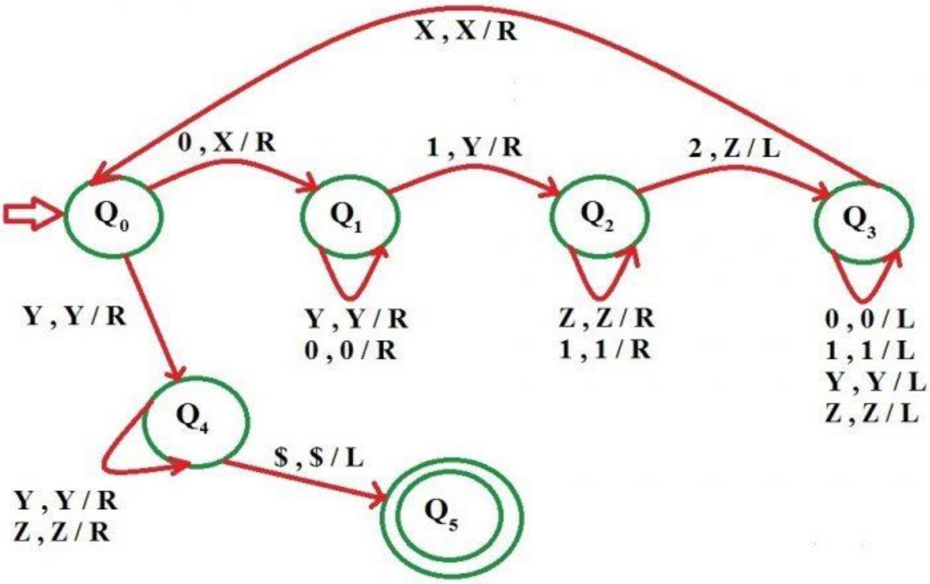
A condition comes when you find a X immediately followed by a Y. At this point we keep moving right and keep on checking that all 1's and 2's have been converted to Y and Z. If not then string is not accepted. If we reach $ then string is accepted.

- **Step-1:**
  Replace 0 by X and move right, Go to state Q1.
- **Step-2:**
  Replace 0 by 0 and move right, Remain on same state
  Replace Y by Y and move right, Remain on same state
  Replace 1 by Y and move right, go to state Q2.
- **Step-3:**
  Replace 1 by 1 and move right, Remain on same state
  Replace Z by Z and move right, Remain on same state
  Replace 2 by Z and move right, go to state Q3.
- **Step-4:**
  Replace 1 by 1 and move left, Remain on same state

Replace 0 by 0 and move left, Remain on same state
Replace Z by Z and move left, Remain on same state
Replace Y by Y and move left, Remain on same state
Replace X by X and move right, go to state Q0.

- **Step-5:**
  If symbol is Y replace it by Y and move right and Go to state Q4
  Else go to step 1

- **Step-6:**
  Replace Z by Z and move right, Remain on same state
  Replace Y by Y and move right, Remain on same state
  If symbol is $ replace it by $ and move left, STRING IS ACCEPTED, GO TO FINAL STATE Q5



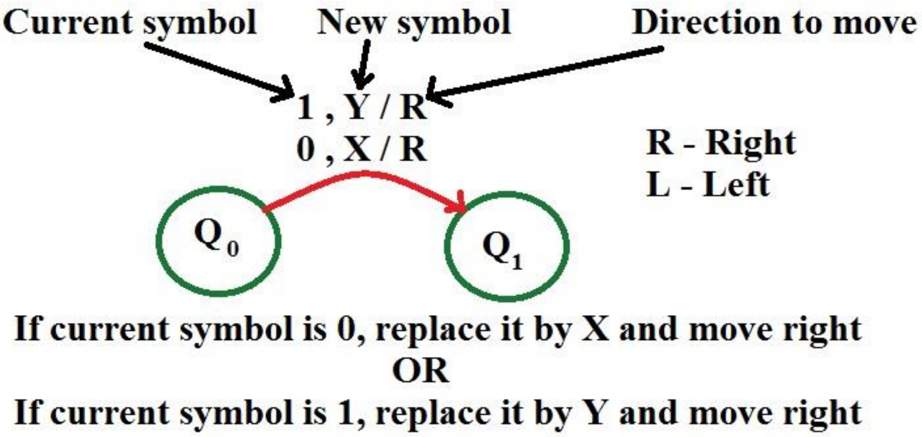
Construct a Turing Machine for language L = {wwr | w ∈ {0, 1}}

-

Prerequisite – Turing Machine
The language L = {wwres | w ∈ {0, 1}} reprents a kind of language where you use only 2 character, i.e., 0 and 1. The first part of language can be any string of 0 and 1. The second part is the reverse of the first part. Combining both these parts out string will be formed. Any such string which falls in this category will be accepted by this language. The beginning and end of string is marked by $ sign. For example, if first part w = 1 1 0 0 1 then second part wr = 1 0 0 1 1. It is clearly visible that wr is the reverse of w, so the string 1 1 0 0 1 1 0 0 1 1 is a part of given language. It can also be said that the strings which are palindrome of even length will be accepted by this language.
Examples –

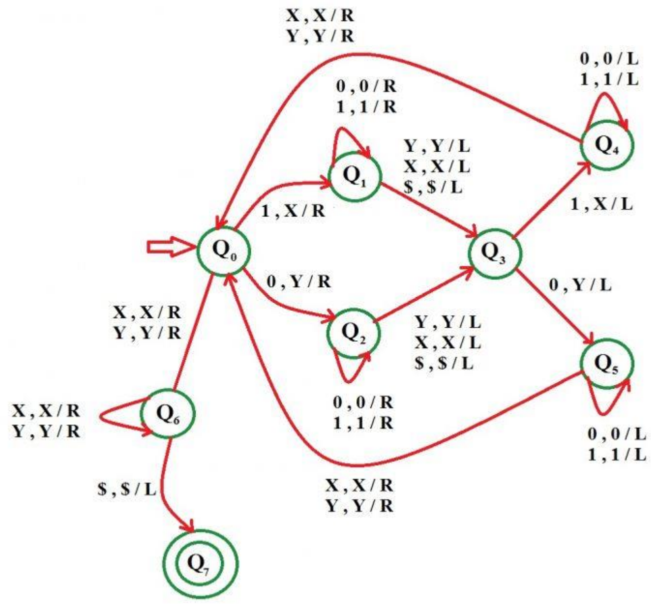
```
Input : 0 0 1 1 1 1 0 0
Output : Accepted
Input : 1 0 1 0 0 1 0 1
Output : Accepted
Input: 0 1 0
Output: Not Accepted
```

Basic Representation –


Current symbol
New symbol
Direction to move
1 , Y / R
0 , X / R
R - Right
L - Left
Q0
Q1
If current symbol is 0, replace it by X and move right
OR
If current symbol is 1, replace it by Y and move right

Approach 1: Two Pointers
Start from the beginning of the input tape. If the symbol is 0, replace it with Y and move right, or if it's 1, replace it with X and move right.
Once at the end of the string, move back to the position next to the symbol replaced at the beginning and repeat the process.
In the new state, check if the symbol at the corresponding position from the end matches the one at the beginning. If they match, continue; otherwise, reject the string.
Move left and repeat the process until the entire string is processed.
If all symbols match as expected, accept the string.
Assumption: We will replace 0 by Y and 1 by X.
Approach Used – First check the first symbol, if it's 0 then replace it by Y and by X if it's 1. Then go to the end of string. So last symbol is same as first. We replace it also by X or Y depending on it. Now again come back to the position next to the symbol replace from the starting and repeat the same process as told above. One important thing to note is that since wr is reverse of w of both of them will have equal number of symbols. Every time replace a nth symbol from beginning of string, replace a corresponding nth symbol from the end.

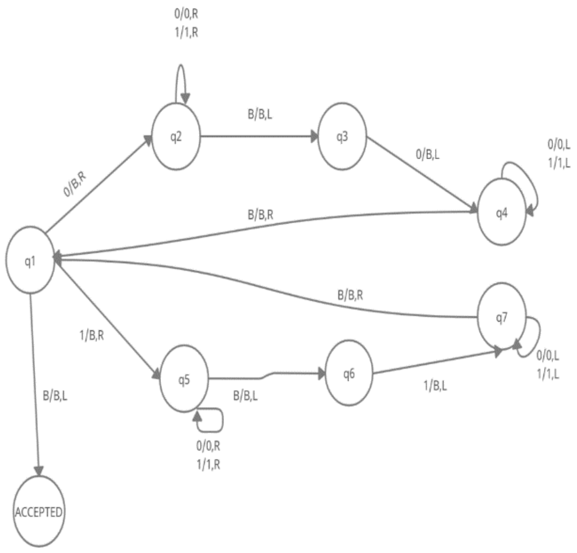
- **Step-1:** If symbol is 0 replace it by Y and move right, Go to state Q2 If symbol is 1 replace it by X and move right, Go to state Q1
- **Step-2:** If symbol is 0 replace it by 0 and move right, remain on same state If symbol is 1 replace it by 1 and move right, remain on same state ——————————————————————————- If symbol is X replace it by X and move right, Go to state Q3 If symbol is Y replace it by Y and move right, Go to state Q3 If symbol is $ replace it by $ and move right, Go to state Q3
- **Step-3:** If symbol is 1 replace it by X and move left, Go to state Q4 If symbol is 0 replace it by Y and move left, Go to state Q5
- **Step-4:** If symbol is 1 replace it by 1 and move left If symbol is 0 replace it by 0 and move left Remain on same state
- **Step-5:** If symbol is X replace it by X and move right If symbol is Y replace it by Y and move right Go to state Q0
- **Step-6:** If symbol is X replace it by X and move right If symbol is Y replace it by Y and move right Go to state Q6 ELSE Go to step 1
- **Step-7:** If symbol is X replace it by X and move right, Remain on same state If symbol is Y replace it by Y and move right, Remain on same state If symbol is $ replace it by $ and move left, STRING IS ACCEPTED, GO TO FINAL STATE Q7



Another solution for the given problem:

If we look at the problem ,it is nothing but an even palindrome so the following is an another solution for the given language. Here I have consider Blank as B. The following turing machine checks that the language L = {wwr | w ∈ {0, 1}} is a palindrome or not.

- **Step 1-** if first symbol is "0" then it replace by the blank symbol goes to the right till it finds the blank and by checking that the symbol is blank or not .
- **Step2-** after getting the last symbol go to the one step left and check that symbol it is "0" or not and
- **step3-** if the last symbol is "0" then replace it with blank symbol and keep going left until it find the blank symbol
- **step4-** after finding the left most blank symbol go one step right and check it "0" or "1"
- **Step 5-** if we get "1" then replace it by the blank symbol and goes to the right until it find the blank symbol.
- **Step 6-** if it find the blank symbol then take one step left and check that symbol is "1" or not .If it is "1" then replace it with the blank symbol.
- **step7-** after replacing it goes to the left and find the blank symbol after finding the blank symbol take one step right and check if the symbol is "1" or "0"
- **Step 8-** repeat the above steps until the whole string become blank .



Construct a Turing Machine for language L = {ww | w ∈ {0,1}}

Last Updated : 29 Oct, 2021

-

Prerequisite – Turing Machine
The language L = {ww | w ∈ {0, 1}} tells that every string of 0's and 1's which is followed by itself falls under this language. The logic for solving this problem can be divided into 2 parts:

1. Finding the mid point of the string

2. After we have found the mid point we match the symbols

Example – Lets understand it with the help of an example. Lets string 1 0 1 1 0 1, so w = 1 0 1 and string is of form (ww).
The first thing that we do is to find the midpoint. For this, we convert 1 in the beginning into Y and move right till the end of the string. Here we convert 1 into y.
Now our string would look like Y 0 1 1 0 Y. Now move left till, find a X or Y. When we do so, convert the 0 or 1 right of it to X or Y respectively and then do the same on the right end. Now our string would look like Y X 1 1 X Y. Thereafter, convert these 1's also and finally it would look like Y X Y **Y** X Y.
At this point, you have achieved the first objective which was to find the midpoint. Now convert all X and Y on the left of midpoint into 0 and 1 respectively, so string becomes 1 0 1 Y X Y. Now, convert the 1 into Y and move right till, find Y in the beginning of the right part of the string and convert this Y into a blank (denoted by B). Now string looks like Y 0 1 B X Y.

Similarly, apply this on 0 and x followed by 1 and Y. After this string looks like Y X Y B B B. Now that you have no 0 and 1 and all X and Y on the right part of the string are converted into blanks so our string will be accepted.

Assumption: We will replace 0 by X and 1 by Y.
Approach Used –
The first thing is to find the midpoint of the string, convert a 0 or 1 from the beginning of the string into X or Y respectively and a corresponding 0 or 1 into X or Y from the end of the string. After continuously doing it a point is reached when all 0's and 1's have been converted into X and Y respectively. At this point, you are on the midpoint of the string. So, our first objective is fulfilled.
Now, convert all X's and Y's on the left of the midpoint into 0's and 1's. At this point the first half the string is in the form of 0 and 1. The second half of the string is in the form of X and Y.

Now, start from the beginning of the string. If you have a 0 then convert it into X and move right till reaching the second half, here if we find X then convert it into a blank(B). Then traverse back till find an X or a Y. We convert the 0 or 1 on the right of it into X or Y respectively and correspondingly, convert its X or Y in the second half of string into a blank(B).

Keep doing this till converted all symbols on the left part of the string into X and Y and all symbols on the right of string into blanks. If any one part is completely converted but still some symbols in the other half are left unchanged then the string will not be accepted. If you did not find an X or Y in the second half for a corresponding 0 or 1 respectively in the first half. Then also string will not be accepted.

Examples:

```
Input  : 1 1 0 0 1 1 0 0

Output : Accepted


Input  : 1 0 1 1 1 0 1

Output : Not accepted
```

- **Step-1:**
  If the symbol is 0 replace it by X and move right

If the symbol is 1 replace it by Y and move right,
Go to state Q1 and step 2

If the symbol is X replace it by X and move left or
If the symbol is Y replace it by Y and move left,
Go to state Q4 and step 5

- **Step-2:**
  If the symbol is 0 replace it by 0 and move right, remain on the same state
  If the symbol is 1 replace it by 1 and move right, remain on the same state

  ---

  If the symbol is X replace it by X and move left or
  If the symbol is Y replace it by Y and move left or
  If the symbol is $ replace it by $ and move left, Go to state Q2 and step 3

- **Step-3:**
  If symbol is 0 replace it by X and move left, or
  If symbol is 1 replace it by Y and move left,
  Go to state Q3 and step 4

- **Step-4:**
  If the symbol is 0 replace it by 0 and move left, remain on the same state
  If the symbol is 1 replace it by 1 and move left, remain on the same state

  ---

  If the symbol is X replace it by X and move right or
  If the symbol is Y replace it by Y and move right,
  Go to state Q0 and step 1

- **Step-5:**
  If symbol is X replace it by X and move left or
  If symbol is Y replace it by Y and move left
  Go to state Q4 and step 6

- **Step-6:**
  If symbol is X replace it by 0 and move left, remain on same state
  If symbol is Y replace it by 1 and move left, remain on same state
  – – – – – – – – – — – – – – – – – – – – —
  If symbol is $ replace it by $ and move right
  Go to state Q4 and step 7

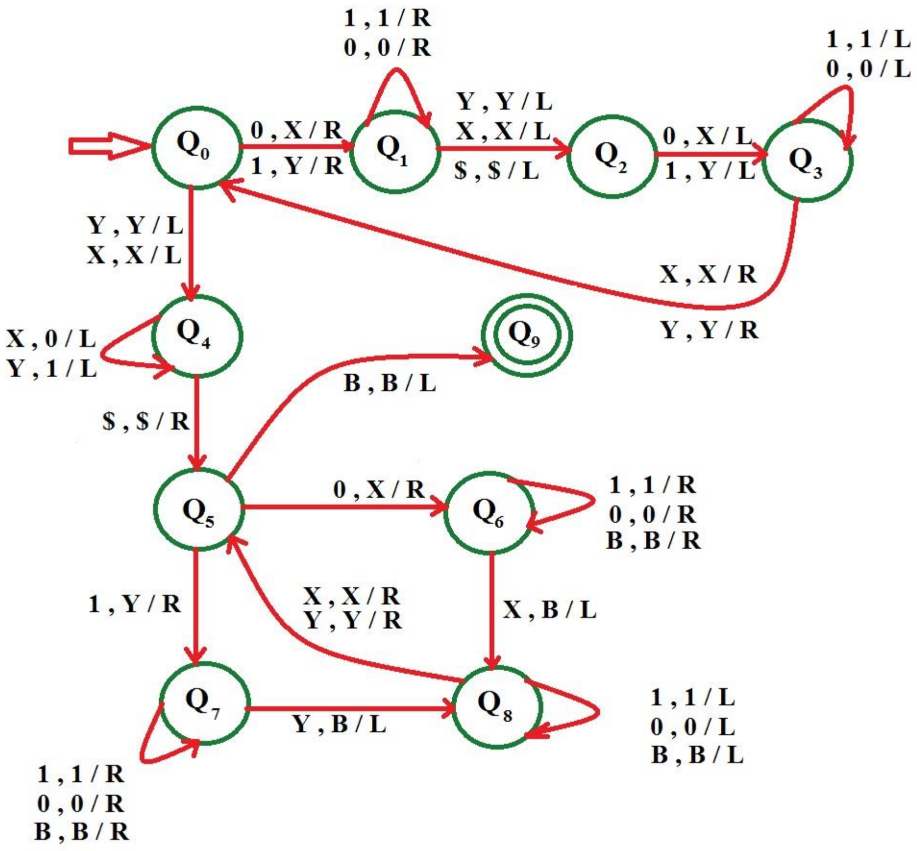
- **Step-7:**
  If symbol is 0 replace it by X and move right, go to state Q6 and step 8
  If symbol is 1 replace it by Y and move right, go to state Q7 and step 9
  – – – – – – – – – — – – – – – – – – – – —
  If symbol is B replace it by B and move left, STRING ACCEPTED, GO TO FINAL STATE Q9

- **Step-8:**
  If symbol is 0 replace it by 0 and move right, remain on same state
  If symbol is 1 replace it by 1 and move right, remain on same state
  If symbol is B replace it by B and move right, remain on same state
  – — – – – – – – – – – – – – – – – – – –
  If symbol is X replace it by B and move left
  Go to state Q8 and step 10

- **Step-9:**

If symbol is 0 replace it by 0 and move right, remain on same state
If symbol is 1 replace it by 1 and move right, remain on same state
If symbol is B replace it by B and move right, remain on same state
– –– – – – – – – – – – – – – – – – – – –
If symbol is Y replace it by B and move left
Go to state Q8 and step 10

- **Step-10:**
  If symbol is 0 replace it by 0 and move left, remain on same state
  If symbol is 1 replace it by 1 and move left, remain on same state
  If symbol is B replace it by B and move left, remain on same state
  – –– – – – – – – – – – – – – – – – – – –
  If symbol is Y replace it by Y and move right or
  If symbol is X replace it by X and move right
  Go to state Q5 and step 7



Construct Turing machine for L = {a^n b^m $a^{(n+m)}$ | n,m≥1}

Last Updated : 31 May, 2018

-

L = {a^n b^m $a^{(n+m)}$ | n,m≥1} represents a kind of language where we use only 2 character, i.e., a and b. The first part of language can be any number of "a" (at least 1). The second part be any number of "b" (at least 1). The third part of language is a number of "a" whose count is sum of count of a's in first part of string and count of b's in second part of string . Any such string which falls in this category will be accepted by this language. The beginning and end of string is marked by $ sign.

Examples:

Input : a a b b b a a a a a // n=2, m=3

Output : Accepted

Input : a a b a a a a // n=2, m=1

Output : Not accepted

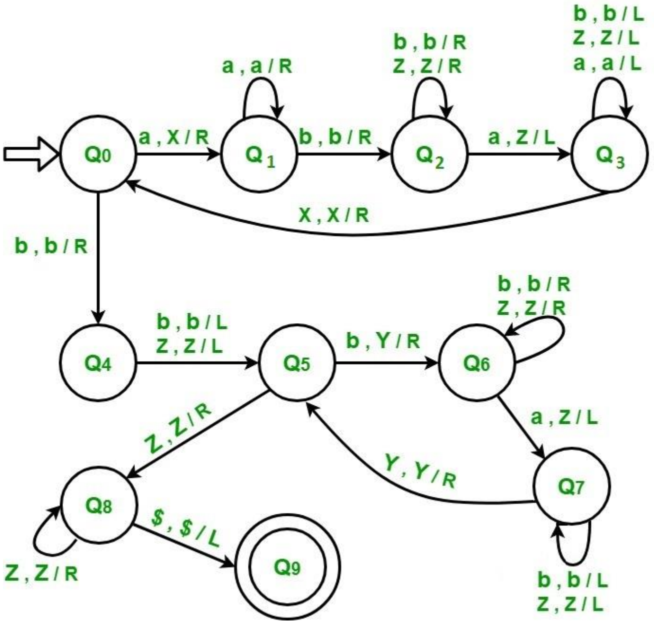
Approach used –

1. Convert “a” in first part into “X” and then move right ignoring all intermediate symbols. When “a” is encountered just after “b” then convert it into “Z” and move left and stop at the position just next to “X”. Repeat the above procedure.
2. When all a’s in first part have been converted then apply the same process on second part. Convert “b” into “Y” and “a” into “Z” in the third part.

When entire first and second part has been converted and if third part is also converted then string will be accepted else not.

Steps –

Step-0: Convert “a” into “X”, move right and go to state 1. If symbol is “b”, ignore it, move right and go to state-4.
Step-1: If symbol is “a”, ignore it and move right, remain on same state. If symbol is “b”, ignore it, move right and go to state-2.
Step-2: If symbol is “Z”, ignore it and move right, remain on same state. If symbol is “b”, ignore it and move right, remain on same state and if symbol is “a”, convert it into “Z”, move left and go to state-3.
Step-3: If symbol is “Z”, ignore it and move left, remain on same state. If symbol is “b”, ignore it and move left, remain on same state. If symbol is “a”, ignore it and move left, remain on same state, and if symbol is “X”, ignore it and move right, go to state-0.
Step-4: If symbol is “b”, ignore it and move left, go to state 5, and if symbol is “Z”, ignore it and move left, go to state-5.
Step-5: Convert “b” into “Y”, move right and go to state 6, and if symbol is “Z”, ignore it and move right, go to state-8.
Step-6: If symbol is “Z”, ignore it and move right, remain on same state. If symbol is “b”, ignore it and move right, remain on same state, and if symbol is “a”, convert it into “Z”, move left and go to state-7.
Step-7: If symbol is “Z”, ignore it and move left, remain on same state. If symbol is “b”, ignore it and move left, remain on same state, and if symbol is “Y”, ignore it and move right, go to state-5.
Step-8: If symbol is “Z”, ignore it and move right, remain on same state, and if symbol is “$”, ignore it and move left, go to state-9.
Step-9: String ACCEPTED



Construct a Turing machine for L = $\{a^i b^j c^k \mid i*j = k;\ i, j, k \geq 1\}$

Last Updated : 13 Jun, 2024

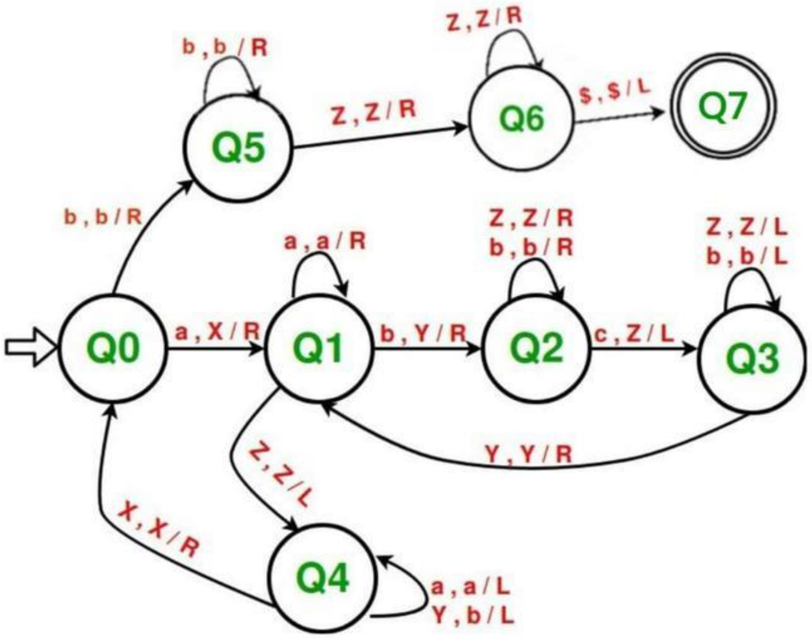
Prerequisite – Turing Machine
In a given language, L = {aibjck | i*j = k; i, j, k ≥ 1}, where every string of ‘a’, ‘b’ and ‘c’ has a certain number of a’s, then a certain number of b’s and then a certain number of c’s. The condition is that each of these 3 symbols should occur at least once. ‘a’ and ‘b’ can occur however many times, but the occurrences of ‘c’ must be equal to the product of occurrences of ‘a’ and occurrences of ‘b’. Assume that string is ending with ‘$’.
Examples –

```
Input: a a b b b c c c c c c
      Here a = 2, b = 3, c = 2 * 3 = 6
Output: NOT ACCEPTED

Input: a a b b c c c
      Here a = 2, b = 2, c = 3 but c should be 4 (c=2*2 must be 4 but here c=3)
Output: NOT ACCEPTED
```

Approach used – Scan the input from the left.

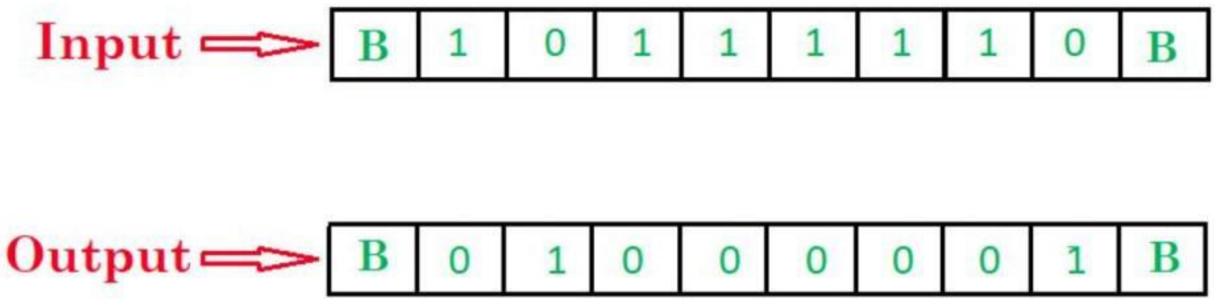
1. First, replace an ‘a’ with ‘X’ and move 1 step right. Then skip all the a’s and move right.
2. When the pointer reaches the first ‘b’ stop. Replace one ‘b’ with ‘Y’, then move right skipping all intermediate b’s and corresponding to the replaced ‘b’ now replace one ‘c’ with ‘Z’ and move left.
3. Now move towards the left skipping all ‘Z’ and ‘b’ in the way.
4. When the pointer reaches the most recent ‘Y’ move right.
5. If the pointer is pointing at ‘b’ then repeat steps 2 to 4, else if the pointer is pointing at ‘Z’ then move towards left while replacing all ‘Y’ to ‘b’ and skipping all a’s.
6. When the pointer comes to the most recent ‘X’ move one step right.
7. If the pointer is still pointing to ‘a’ then repeat all the above steps, else if the pointer is at ‘y’ then move towards right skipping all ‘y’ and ‘Z’.
8. When $ is reached then move left. The string is ACCEPTED.



Turing machine for 1’s and 2’s complement

Last Updated : 02 Jul, 2024

Problem-1:
Draw a Turing machine to find 1’s complement of a binary number.
1’s complement *of a binary number is another binary number obtained by toggling all bits in it, i.e., transforming the 0 bit to 1 and the 1 bit to 0.*
Example:



1’s Complement

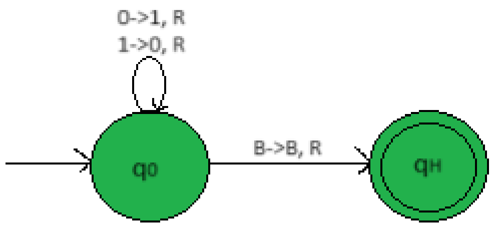
Approach:
1. Scanning input string from left to right
2. Converting 1’s into 0’s
3. Converting 0’s into 1’s
4. Stop when you reach the end of the string (when BLANK is encountered)

States:
- q0: Initial state
- qH: Final state

Symbols:
- 0, 1: Binary digits
- B: Blank symbol
- R, L: Right and left movements

Steps:
- **Step-1.** Convert all 0’s into 1’s and all 1’s into 0’s and if B is found go to right.
- **Step-2.** Stop the machine.

Turing Machine:



Turing Machine for 1’s Complement

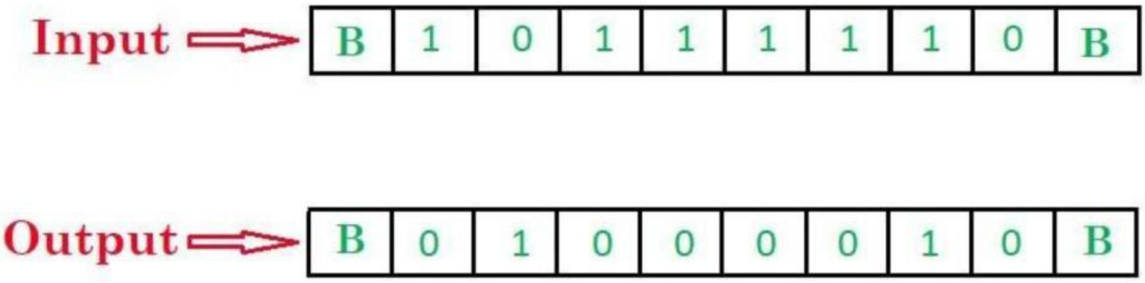
Explanation:
- State q0 replaces ‘1’ with ‘0’ and ‘0’ with ‘1’ and moves to the right.
- When BLANK is reached, move towards the right and reach the final state qH.

Problem-2:

Draw a Turing machine to find 2’s complement of a binary number.

2’s complement *of a binary number is 1 added to the 1’s complement of the binary number.*

Example:



Input	B	1	0	1	1	1	1	1	0	B
Output	B	0	1	0	0	0	0	1	0	B

2’s Complement

Approach:

1. Start from the end of the input string.
2. Pass all consecutive ‘0’s.
3. When you encounter the first ‘1’, do nothing.
4. After that, replace ‘1’ with ‘0’ and ‘0’ with ‘1’.
5. Stop when you reach the beginning of the string.

Intuition:

Whenever a number’s 1’s complement is taken and 1 bit is added to its LSB (Least Significant Bit), all the 1’s (which were originally 0) appearing together from right will add with the carry and change back to 0. Since the first encounter of 0 in 1’s complement string (which was originally 1) , every ‘1’ bit to its right, if any, will add 1 and change to 0. The first encountered 0 in the string will add 1 and change back to 1, as in the original string. This does not impact any further bit to its left, thus final string have

1. All 0’s same to the right of first ‘1’ bit encountered ((complement of 0) + 1) = 0 + carry(=1) to its left element)
2. The first ‘1’ bit encountered will also revert back to 1 the same way.
3. All other bits to its left will be flipped without any restrictions.

States:

- q0: Initial state
- q1, q2: Transition states
- qH: Final state

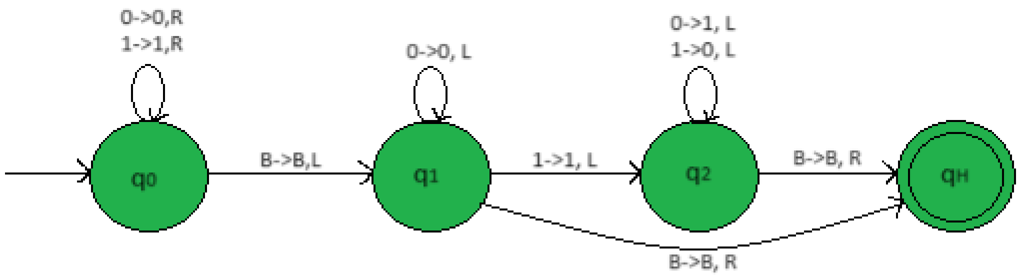
Symbols:

- 0, 1: Binary digits
- B: Blank symbol
- R, L: Right and left movements

Steps:

- In state q0, move right until you encounter a BLANK symbol. Then, transition to state q1 and go to left.
- In state q1, continue moving left until you encounter the first ‘1’. Transition to state q2.
- In state q1, if BLANK symbol is encountered, then transition to state qH.
- In state q2, replace ‘1’ with ‘0’ and ‘0’ with ‘1’.
- If BLANK symbol is encountered, then transition to state qH.
- In state qH, halt the machine.

Turing Machine

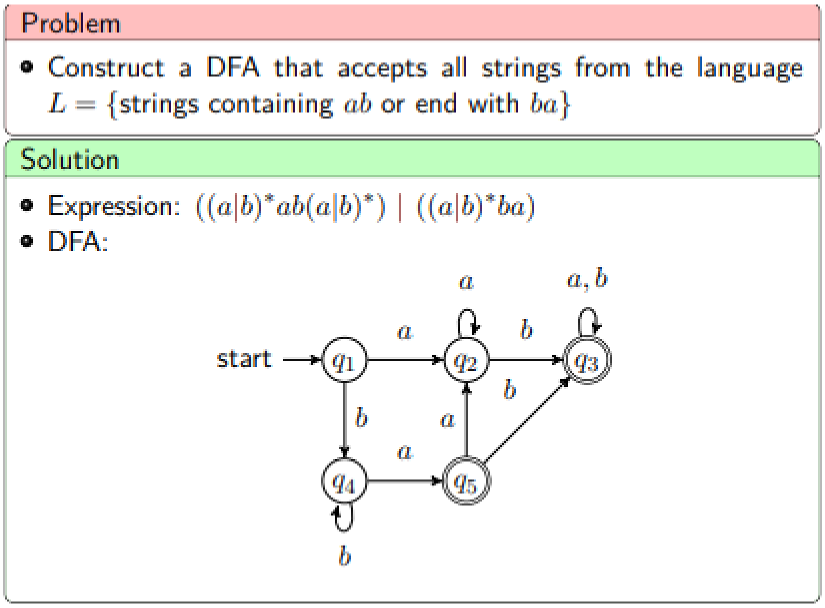


Turing Machine for 2's Complement

Explanation:

- Using state 'q0' we reach the end of the string.
- When BLANK is reached, move towards the left.
- Using state 'q1' we pass all 0's and move left till first '1' is found.
- If before finding '1' we encounter BLANK, it implies the number is 0 i.e. it contains no 1.
- Hence its 2s complement is 0 itself. So no need to change any symbols and just halt the machine.
- If '1' is encountered, skip that '1' and move left.
- Using state 'q2' we complement the each digit and move left.
- When BLANK is reached, move towards the right and reach the final state qH.

A

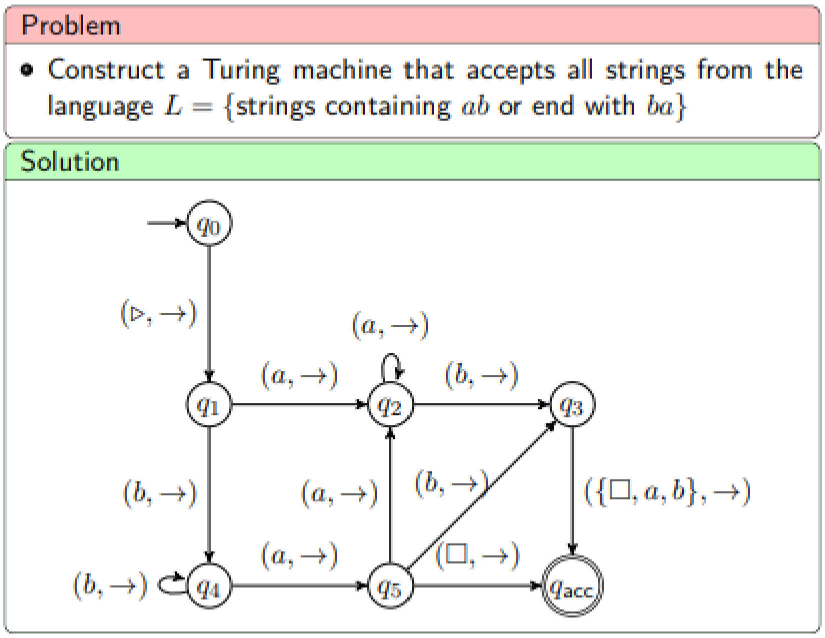


Problem

- Construct a DFA that accepts all strings from the language $L = \{\text{strings containing } ab \text{ or end with } ba\}$

Solution

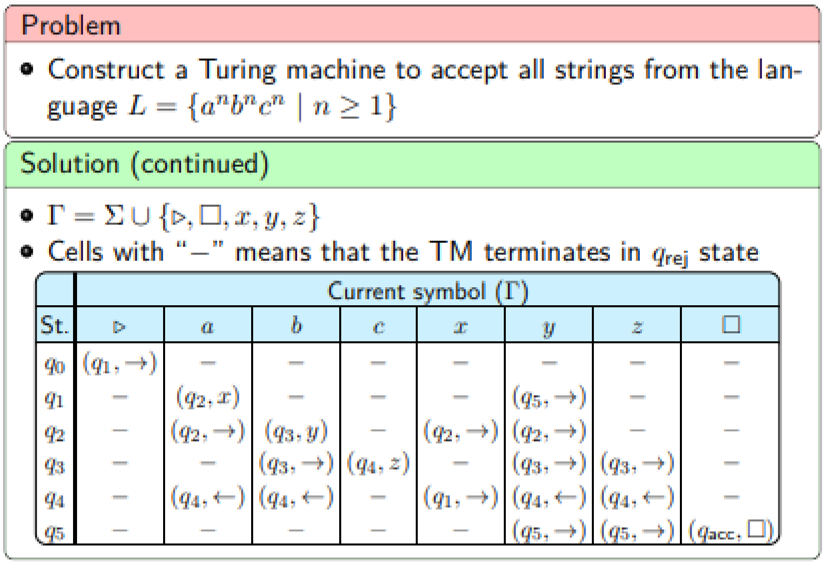
- Expression: $((a|b)^*ab(a|b)^*) \mid ((a|b)^*ba)$
- DFA:



Problem

- Construct a Turing machine that accepts all strings from the language $L = \{\text{strings containing } ab \text{ or end with } ba\}$

Solution



Problem

- Construct a Turing machine to accept all strings from the language $L = \{a^n b^n c^n \mid n \geq 1\}$

Solution (continued)

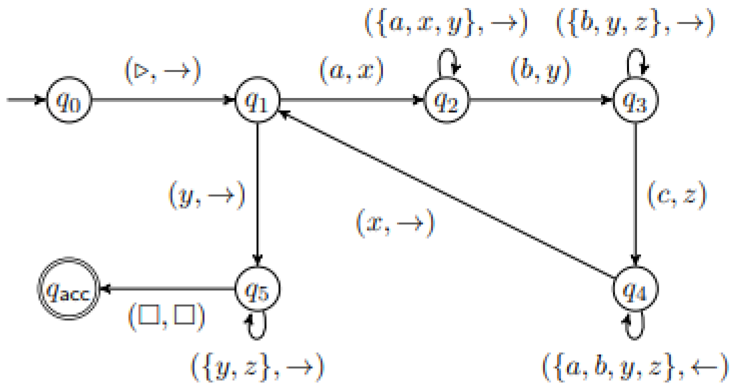
- $\Gamma = \Sigma \cup \{\triangleright, \square, x, y, z\}$
- Cells with "–" means that the TM terminates in q_{rej} state

	Current symbol (Γ)							
St.	$\triangleright$	a	b	c	x	y	z	$\square$
q_0	$(q_1, \rightarrow)$	–	–	–	–	–	–	–
q_1	–	(q_2, x)	–	–	–	$(q_5, \rightarrow)$	–	–
q_2	–	$(q_2, \rightarrow)$	(q_3, y)	–	$(q_2, \rightarrow)$	$(q_2, \rightarrow)$	–	–
q_3	–	–	$(q_3, \rightarrow)$	(q_4, z)	–	$(q_3, \rightarrow)$	$(q_3, \rightarrow)$	–
q_4	–	$(q_4, \leftarrow)$	$(q_4, \leftarrow)$	–	$(q_1, \rightarrow)$	$(q_4, \leftarrow)$	$(q_4, \leftarrow)$	–
q_5	–	–	–	–	–	$(q_5, \rightarrow)$	$(q_5, \rightarrow)$	$(q_{\text{acc}}, \square)$

Problem

- Construct a Turing machine to accept all strings from the language $L = \{a^n b^n c^n \mid n \geq 1\}$

Solution (continued)


q_0
q_1
q_2
q_3
q_4
q_5
q_{acc}
$(\triangleright, \rightarrow)$
(a, x)
(b, y)
$(\{a, x, y\}, \rightarrow)$
$(\{b, y, z\}, \rightarrow)$
$(y, \rightarrow)$
$(x, \rightarrow)$
(c, z)
$(\square, \square)$
$(\{y, z\}, \rightarrow)$
$(\{a, b, y, z\}, \leftarrow)$