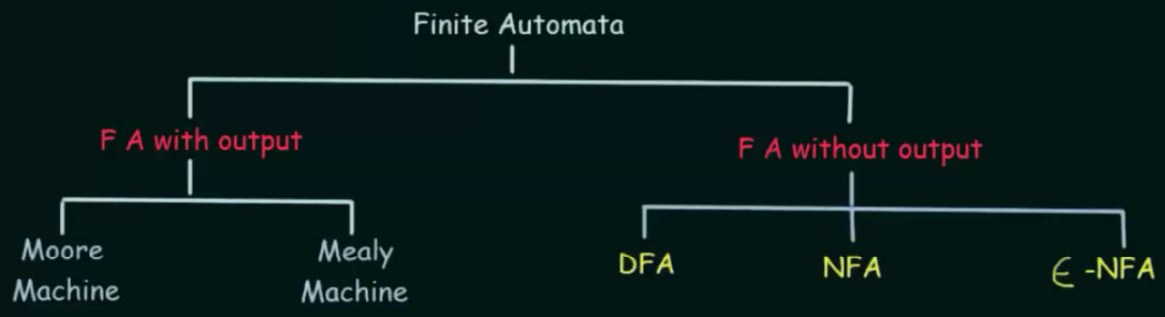
Finite State Machine



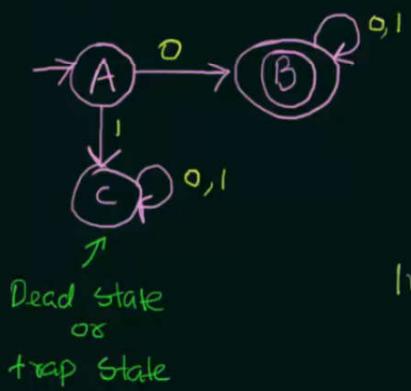
DFA - Deterministic Finite Automata

- It is the simplest model of computation
- It has a very limited memory

Deterministic Finite Automata (Example-1)

L1 = Set of all strings that start with '0'

= { 0, 00, 01, 000, 010, 011, 0000, ... }



Eg. 001 ✓

Initial state A → B → B → B - Final State

Eg. 101 ✗

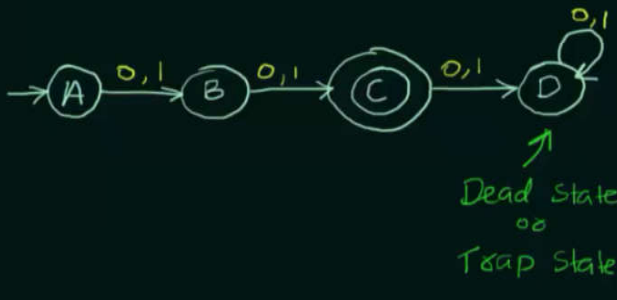
Initial State A → C → C → C - Not final State

Deterministic Finite Automata (Example-2)

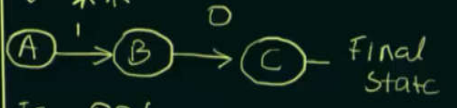
Construct a DFA that accepts sets of all strings over {0,1} of length 2.

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

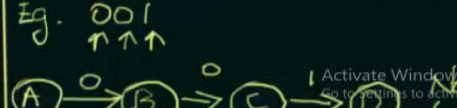
$L = \{00, 01, 10, 11\}$



Eg. 00 ✓



Eg. 10 ✓



Eg. 001

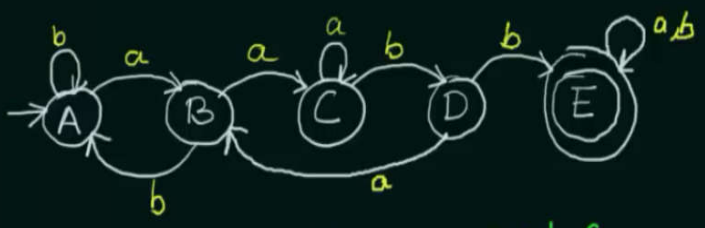
Deterministic Finite Automata (Example-3)

Construct a DFA that accepts any strings over {a,b} that does not contain the string aabb in it.

$\Sigma = \{a, b\}$

Try to design a simpler problem

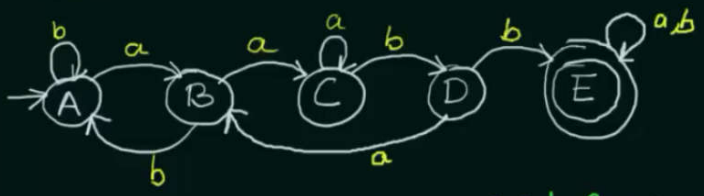
Let us construct a DFA that accepts all strings over {a,b} that contains the string aabb in it



aaabb

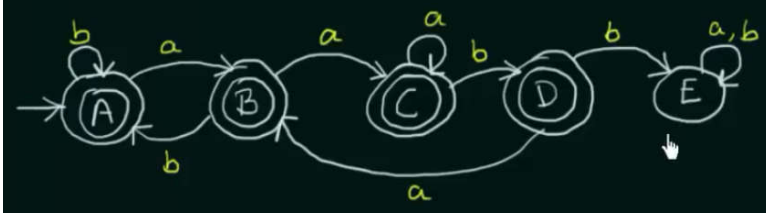
acba

Let us construct a DFA that accepts all strings over {a,b} that contains the string aabb in it



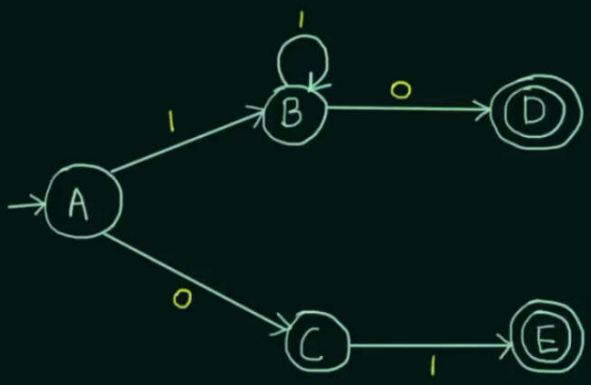
aaabb acaba

- Flip the States
- Make the Final State into non final state and
- Make the non final states into final states



Deterministic Finite Automata (Example-4)

How to figure out what a DFA recognizes?



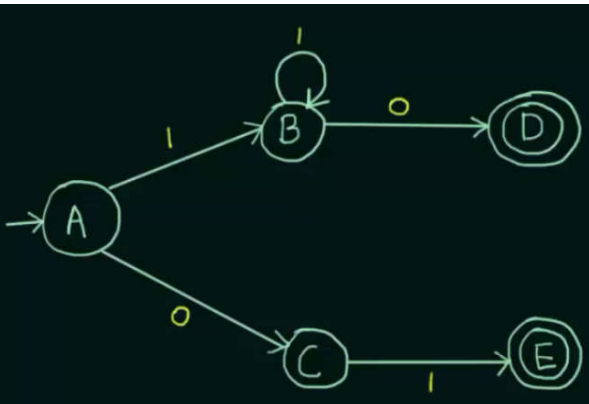
10 ✓

11111 0 ✓

A B D

01 ✓

One binary digit '1'



10 ✓

1 1 1 1 1 0 ✓

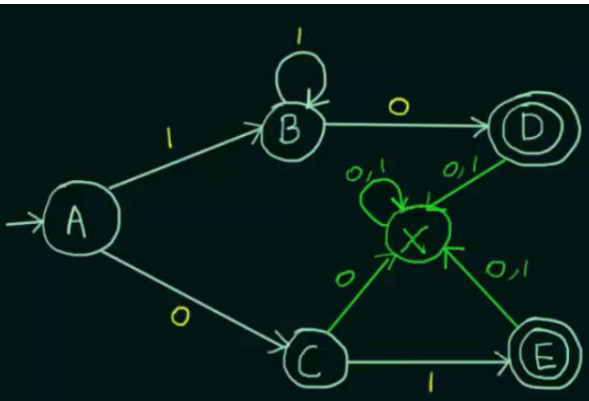
A B D

01 ✓

one binary digit '1'

L = {Accepts the string 01 or a string of atleast one '1' followed by a '0'}

Eg. 001 , 010 , 011, 1101, 1100



10 ✓

1 1 1 1 1 0 ✓

A B D

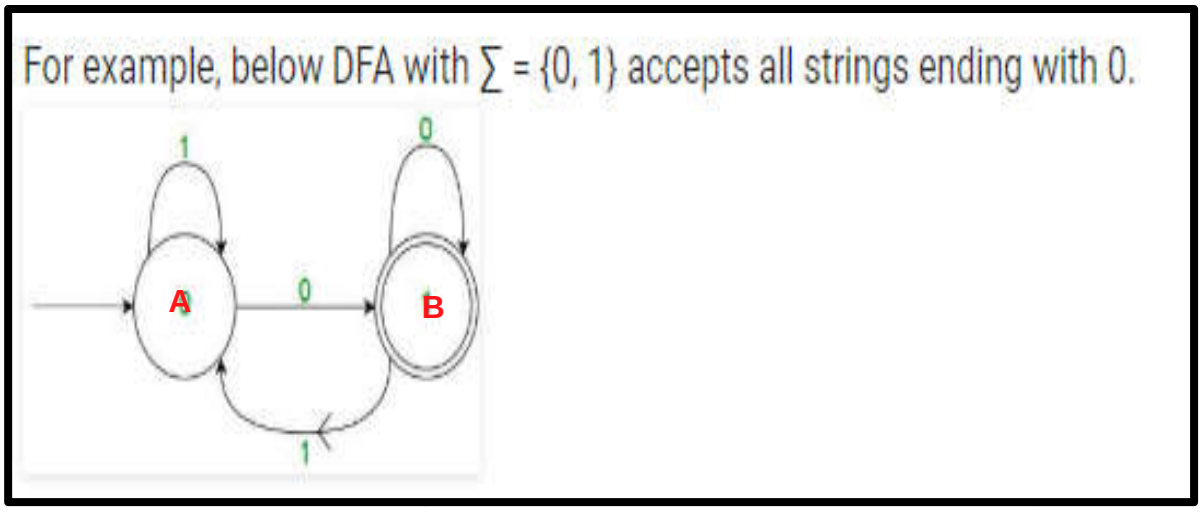
01 ✓

one binary digit '1'

L = {Accepts the string 01 or a string of atleast one '1' followed by a '0'}

Eg. 001 , 010 , 011, 1101, 1100

X - Dead state



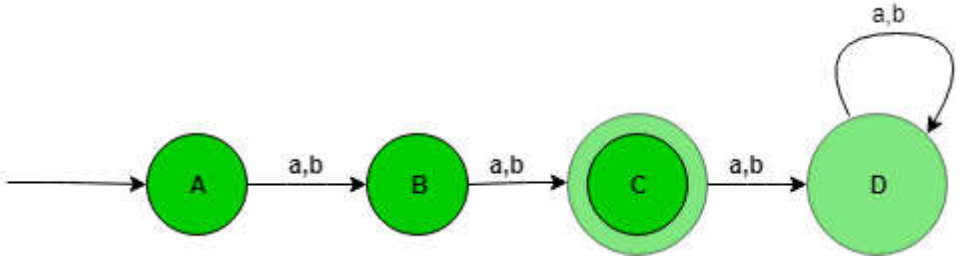
TOC | Designing Deterministic Finite Automata (Set 1)

Problem-1: Construction of a DFA for the set of string over {a, b} such that length of the string |w|=2 i.e, length of the string is exactly 2.

Explanation – The desired language will be like:

```
L = {aa, ab, ba, bb}
```

The state transition diagram of the language will be like:



Here,
State A represent set of all string of length zero (0), state B represent set of all string of length one (1), state C represent set of all string of length two (2). State C is the final state and D is the dead state it is so because after getting any alphabet as input it will not go into final state ever.

```
Number of states: n+2

Where n is |w|=n
```

The above automata will accept all the strings having the length of the string exactly 2. When the length of the string is 1, then it will go from state A to B. When the length of the string is 2, then it will go from state B to C and when the length of the string is

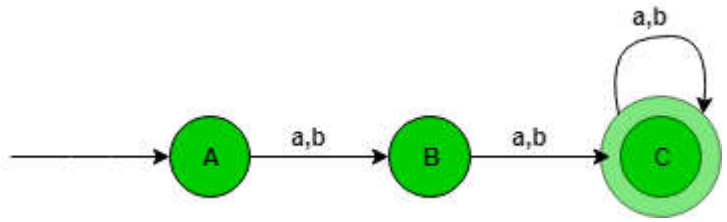
greater than 2, then it will go from state C to D (Dead state) and after it from state D TO D itself.

Problem-2: Construction of a DFA for the set of string over {a, b} such that length of the string |w|>=2 i.e, length of the string should be at least 2.

Explanation – The desired language will be like:

```
L = {aa, ab, ba, bb, aaa, aab, aba, abb........}
```

The state transition diagram of the language will be like:



Here,
State A represent set of all sting of length zero (0), state B represent set of all sting of length one (1), and state C represent set of all sting of length two (2).

```
Number of states: n+1

Where n is |w|>=n
```

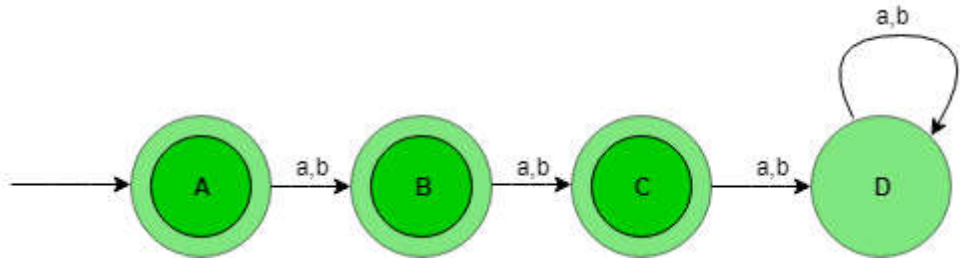
The above automata will accept all the strings having the length of the string at least 2. When the length of the string is 1, then it will go from state A to B. When the length of the string is 2, then it will go from state B to C and lastly when the length of the string is greater than 2, then it will go from state B to B itself.

Problem-3: Construction of a DFA for the set of string over {a, b} such that length of the string |w|<=2 i.e, length of the string is atmost 2.

Explanation – The desired language will be like:

```
L = {ε, aa, ab, ba, bb}
```

The state transition diagram of the language will be like:



Here,
State A represent set of all sting of length zero (0), state B represent set of all sting of length one (1), state C represent set of all sting of length two (2), state A, B, C is the

final state and D is the dead state it is so because after getting any alphabet as input it will not go into final state ever.

```
Number of states: n+2

Where n is |w|<=n
```

The above automata will accept all the strings having the length of the string at most 2. When the length of the string is 1, then it will go from state A to B. When the length of the string is 2, then it will go from state B to C and lastly when the length of the string is greater than 2, then it will go from state C to D (Dead state).

TOC | Designing Deterministic Finite Automata (Set 2)

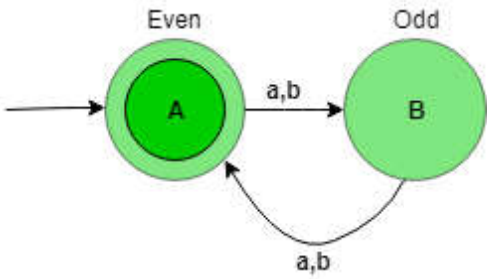
Problem-1: Construction of a DFA for the set of string over {a, b} such that length of the string |w| is divisible by 2 i.e, |w| mod 2 = 0.

Same as Detect Even Number

Explanation – The desired language will be like:

```
L = {ε, aa, ab, ba, bb, aaaa, bbbb, ............}
```

The state transition diagram of the language will be like:



Here, state A represent set of all string of length even (0, 2, 4, …), and state B represent set of all string of length odd (1, 3, 5, …).

```
Number of states: n

If |W| mod n = 0
```

The above automata will accept all the strings having the length of the string divisible by 2. When the length of the string is 1, then it will go from state A to B. When the length of the string is 2, then it will go from state B to A and so on. State A is the final state i.e, it accept all the string having length divisible by 2.

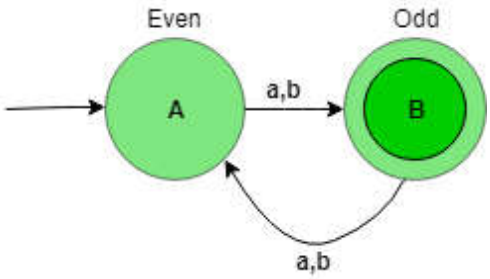
Problem-2: Construction of a DFA for the set of string over {a, b} such that length of the string |w| is not divisible by 2 i.e, |w| mod 2 = 1.

Same as Detect Odd

Explanation – The desired language will be like:

```
L = {a, b, aaa, aab, aba, abb, aaaaa, bbbb, .......}
```

The state transition diagram of the language will be like:



Here, state A represent set of all string of length even (0, 2, 4, …), and state B represent set of all string of length odd (1, 3, 5, …).

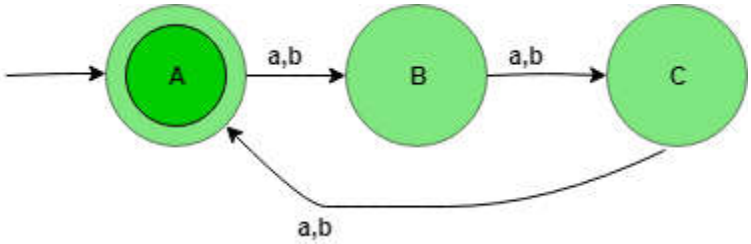
The above automata will accept all the strings having the length of the string not divisible by 2. When the length of the string is 1, then it will go from state A to B. When the length of the string is 2, then it will go from state B to A and so on. State B is the final state i.e, it accept all the string having length not divisible by 2.

Problem-3: Construction of a DFA for the set of string over {a, b} such that length of the string |w| is divisible by 3 i.e, |w| mod 3 = 0.

Explanation – The desired language will be like:

```
L = {ε, aaa, aab, aba, abb, aaaaaa, bbbbbb, .......}
```

The state transition diagram of the language will be like:



Here, state A represents set for which string's length divided by 3 then remainder is zero (0), state B represents set for which string's length divided by 3 then the remainder is one (1), and state C represents set for which string's length divided by 3 then the remainder is two (2).

```
Number of states: n

If |W| mod n = 0
```

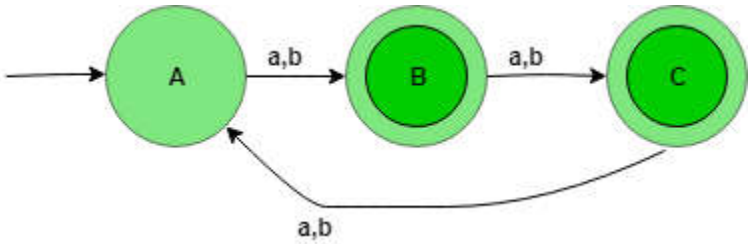
The above automata will accept all the strings having the length of the string divisible by 3. When the length of the string is 1, then it will go from state A to B. When the length of the string is 2, then it will go from state B to C and When the length of the string is 3, then it will go from state C to A (final state). State A is the final state i.e, it accepts all the string having the length divisible by 3.

Problem-4: Construction of a DFA for the set of string over {a, b} such that length of the string |w| is not divisible by 3 i.e, |w| mod 3 = 1.

Explanation – The desired language will be like:

```
L = {a, b, aa, ab, ba, bb, aaaa, bbbb, ........}
```

The state transition diagram of the language will be like:



Here, state A represents set for which string's length divided by 3 then remainder is zero (0), state B represents set for which string's length divided by 3 then the remainder is one (1), and state C represents set for which string's length divided by 3 then the remainder is two (2).

The above automata will accept all the strings having the length of the string not divisible by 3. When the length of the string is 1, then it will go from state A to B. When the length of the string is 2, then it will go from state B to C and When the length of the string is 3, then it will go from state C to A. State B and C are the final state i.e, it accepts all the string having the length not divisible by 3.

TOC | Designing Deterministic Finite Automata (Set 3)

Problem-1: Construction of a minimal DFA accepting set of string over {a, b} where each string containing 'a' as the substring.

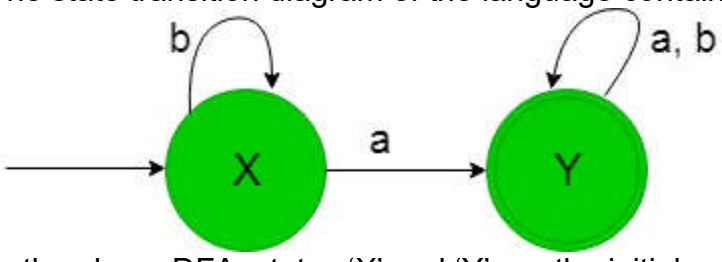
Explanation: The desired language will be like:

```
L1 = {a, aa, ab, ba, ..............}
```

Here as we can see that each string of the language containing 'a' as the substring but the below language is not accepted by this DFA because some of the string of the below language does not contain 'a' as the substring.

```
L2 = {ba, bb, bbb, ..............}
```

The state transition diagram of the language containing 'a' as the substring will be like:



In the above DFA, states 'X' and 'Y' are the initial and final state respectively, it accepts all the strings containing 'a' as the substring. Here as we see that on getting input as 'b' it remains in the state of 'X' itself but on getting 'a' as the input it transit to final state 'Y' and hence such string is accepted by above DFA.

Problem-2: Construction of a minimal DFA accepting set of string over {a, b} where each string containing 'ab' as the substring.

Same as 01 as the substring

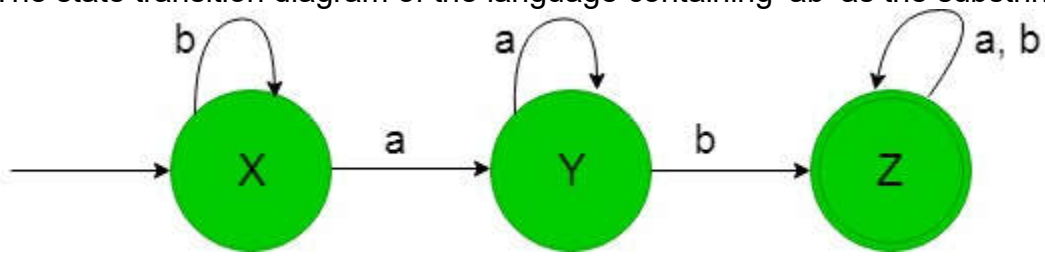
Explanation: The desired language will be like:

```
L1 = {ab, aab, abb, bab, ..............}
```

Here as we can see that each string of the language containing 'ab' as the substring but the below language is not accepted by this DFA because some of the string of the below language does not contain 'ab' as the substring-

```
L2 = {aba, bba, bbbaaa, ..............}
```

The state transition diagram of the language containing 'ab' as the substring will be like:



In the above DFA, states 'X' and 'Z' are the initial and final state respectively, it accepts all the strings containing 'ab' as the substring. Here as we see that on getting 'b' as the input it remains in the state of initial state itself, on getting 'a' as the input it transit to state 'Y' and then on getting 'b' it finally transit to the final state 'Z' and hence this DFA is accepting all the language containing 'ab' as the substring.

TOC | Designing Deterministic Finite Automata (Set 4)

Problem-1: Construction of a minimal DFA accepting set of strings over {a, b} in which every 'a' is followed by a 'b'.

abba, ba, baab, ... is not acceptable

Explanation: The desired language will be like:

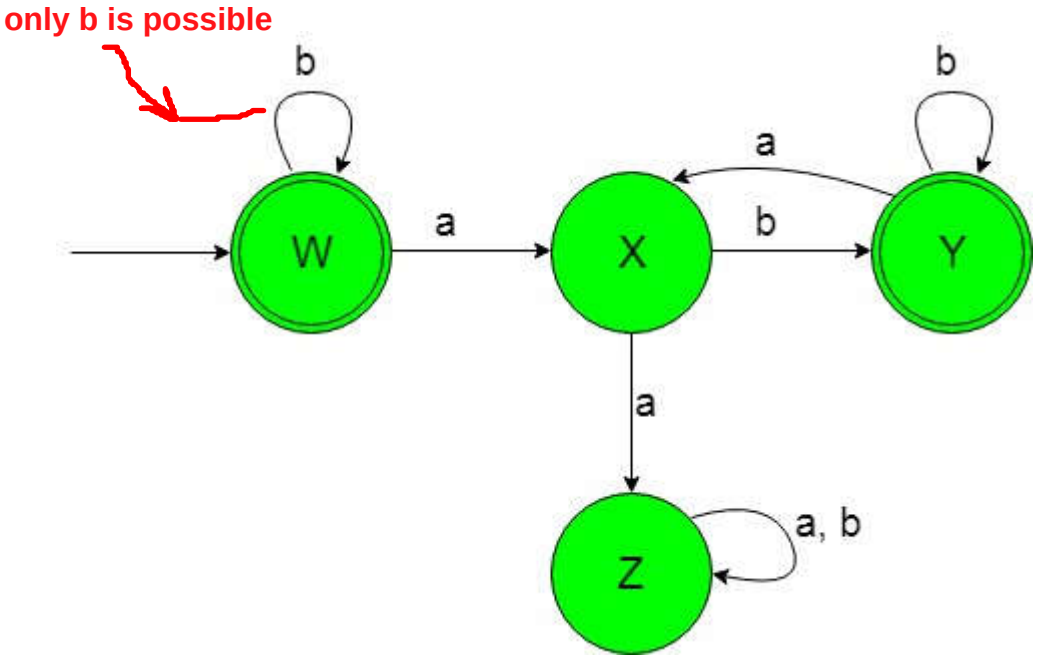
```
L1 = {ε, ab, abab, abbbb, abababababab, ..............}
```

only b, bb, bbbbb, ... also possible

Here as we can see that each string of the language containing 'a' just followed by 'b' but the below language is not accepted by this DFA because some of the string of the below language does not contain 'a' just followed by 'b'.

```
L2 = {ba, baab, bbaba, ..............}
```

The state transition diagram of the language containing 'a' just followed by 'b' will be like:



In the above DFA, state ‘W’ is the initial and final state too which on getting ‘b’ as the input it remains in the state of itself and on getting ‘a’ as the input it transit to a normal state ‘X’ which on getting ‘b’ as the input it transit to the final state ‘Y’ which on getting ‘b’ as input it remains in the state of itself. The state ‘X’ on getting ‘a’ as input it transit to the dead state ‘Z’. The state ‘Z’ is called dead state because on getting any input it can not transit to the final state ever.

Problem-2: Construction of a minimal DFA accepting set of strings over {a, b} in which every ‘a’ is never be followed by ‘b’

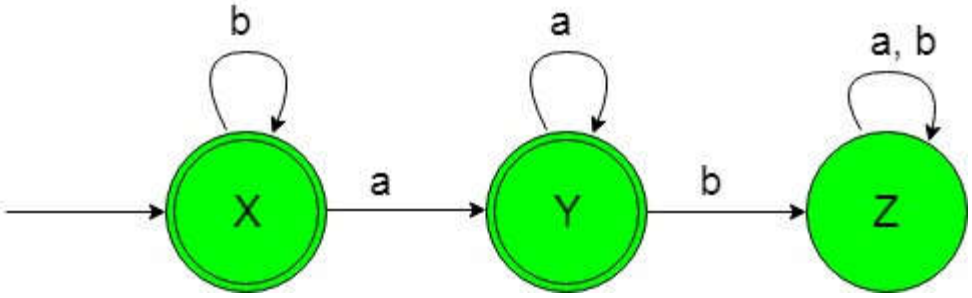
Explanation: The desired language will be like:

```
L1 = {ε, a, aa, aaaa, b, bba, bbbbba.............}
```

Here as we can see that each string of the language containing ‘a’ is never be followed by ‘b’ but the below language is not accepted by this DFA because some of the string of the below language containing ‘a’ is followed by ‘b’.

```
L2 = {ba, baab, bbaba, .............}
```

The state transition diagram of the language containing ‘a’ never be followed by ‘b’ will be like:



In the above DFA, state ‘X’ is the initial and final state which on getting ‘b’ as the input it remains in the state of itself and on getting ‘a’ as input it transit to the final state ‘Y’ which on getting ‘a’ as the input it remains in the state of itself and on getting ‘b’ as input

transit to the dead state ‘Z’. The state ‘Z’ is called dead state this is because it can not ever go to any of the final states.

TOC | Designing Deterministic Finite Automata (Set 5)

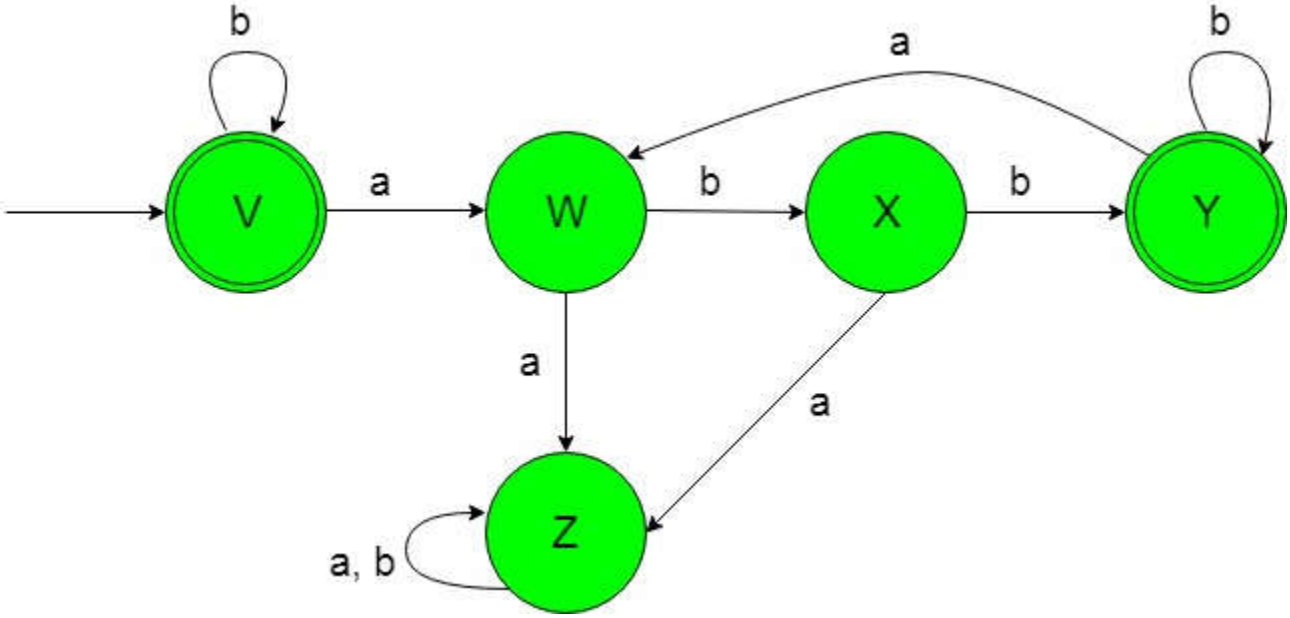
Problem-1: Construction of a minimal DFA accepting set of strings over {a, b} in which every ‘a’ is followed by a ‘bb’.
Explanation: The desired language will be like:

```
L1 = {ε, abb, abbabb, bbbbabb, abbabbabbabbabbbb, ..............}
```

Here as we can see that each string of the language containing ‘a’ is just followed by ‘bb’ but the below language is not accepted by this DFA because some of the string of the below language does not contain ‘a’ just followed by ‘bb’.

```
L2 = {ba, baab, bbaba, ..............}
```

The state transition diagram of the language containing ‘a’ just followed by ‘bb’ will be like:



In the above DFA, state ‘v’ is the initial and final state too which on getting ‘b’ as the input it remains in the state of itself and on getting ‘a’ as the input it transit to a normal state ‘W’ which on getting ‘b’ as the input it transit to the normal state ‘X’ which on getting ‘b’ as input it transit to the final state ‘Y’ which on getting ‘b’ as the input it remains in the state of itself and on getting ‘a’ as the input it transit to the normal state ‘W’ which on getting ‘a’ as the input it transit to the dead state ‘Z’ and the state ‘X’ on getting ‘a’ as the input it transit to the same dead state ‘Z’. The state ‘Z’ is called dead state because on getting any input it can not transit to any of the final states ever.

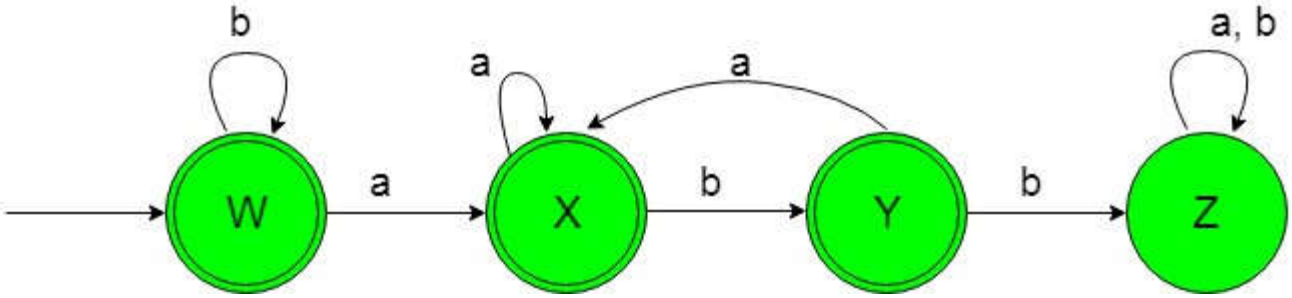
Problem-2: Construction of a minimal DFA accepting set of strings over {a, b} in which every 'a' is never be followed by 'bb'.
Explanation: The desired language will be like:

```
L1 = {ε, a, aa, aabaa, b, bb, bbbbba, ..............}
```

Here as we can see that each string of the language containing 'a' is never be followed by 'bb' but the below language is not accepted by this DFA because some of the string of the below language containing 'a' is followed by 'bb'.

```
L2 = {abb, babbabb, bbaba, ..............}
```

The state transition diagram of the language containing 'a' never be followed by 'bb' will be like:



In the above DFA, the state 'W' is the initial and final state too which on getting 'b' as the input it remains in the state of itself and on getting 'a' as the input it transit to the final state 'X' which on getting 'a' as the input it remains in the state of 'X' and getting 'b' as the input it transit to the final state 'Y' which on getting 'a' as the input it transit to another final state 'X' and the final state 'Y' on getting 'b' as the input it transit to the dead state 'Z'. The state 'Z' is called dead state because on getting any input it can not go to any of the final state ever.

TOC | Designing Deterministic Finite Automata (Set 6)

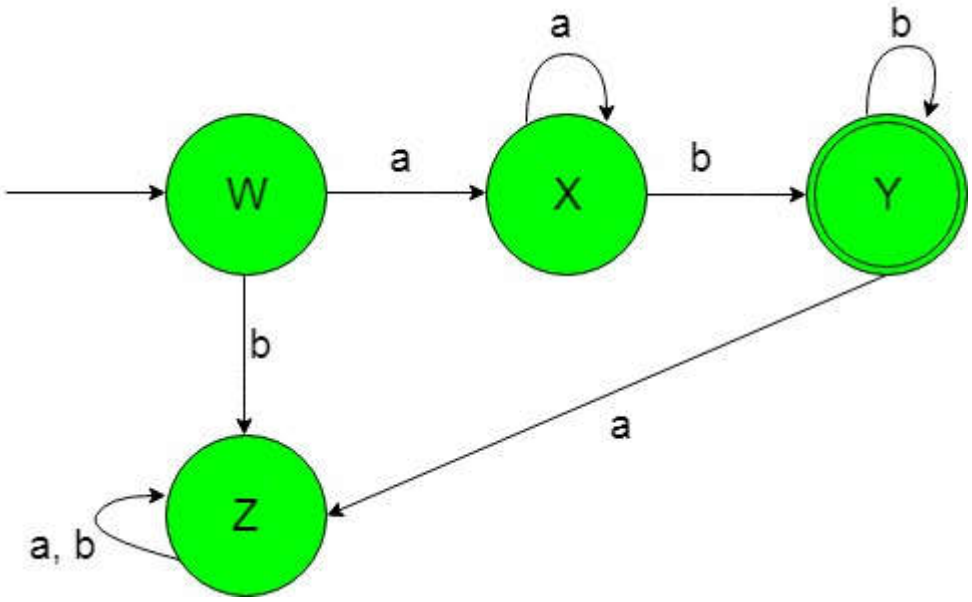
Problem-1: Construction of a minimal DFA accepting set of strings over {a, b} in which $a^n b^m$, where n and m is greater than or equal to 1.
Explanation: The desired language will be like:

```
L1 = {ab, aab, abb, aabb, aaabbb, aaabbbb, ...........}
```

Note: In the above string there should not be any 'a' after 'b'.
Here as we can see that each string of the language containing a and b whose power is greater or equal to 1 but the below language is not accepted by this DFA because some of the string of the below language does not containing a and b whose power is greater or equal to 1.

```
L2 = {ε, a, b, ..............}
```

This language L2 is not accepted by this required DFA.
The state transition diagram of the desired language will be like below:



In the above DFA, the state ‘W’ is the initial state and which on getting ‘a’ as the input it transit to the normal state ‘X’ which on getting ‘a’ as the input it remains in the state of itself and on getting ‘b’ as the input it transit to the final state ‘Y’ which on getting ‘b’ as the input it remains in the state of itself and on getting as the input it transit to the dead state ‘Z’ and when initial state ‘W’ gets ‘b’ as the input then it transit to the dead state ‘Z’.

The state ‘Z’ is called dead state because it can not go to the final state on getting any input.

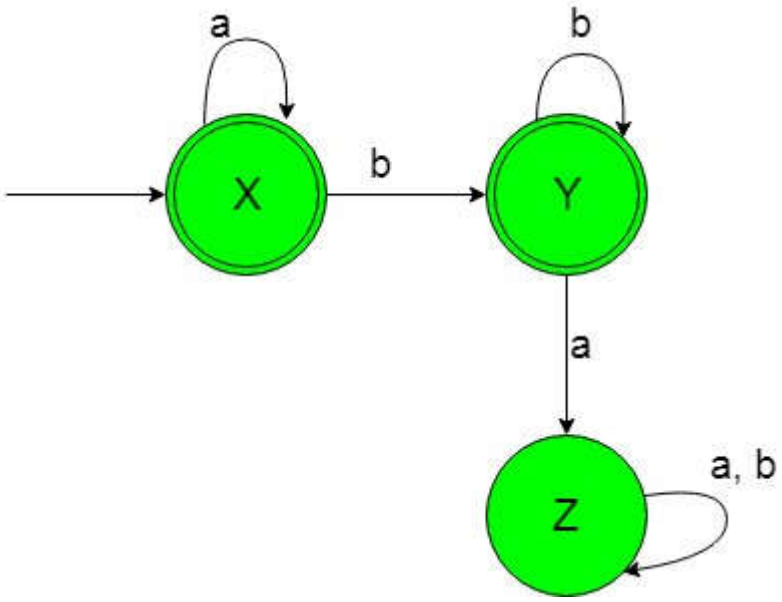
Problem-2: Construction of a minimal DFA accepting set of strings over {a, b} in which $a^n b^m$, where n and m is greater than or equal to 0.
Explanation: The desired language will be like:

```
L1 = {ε, a, b, ab, aab, abb, aabb, aaabbb, aaabbbb, ..........}
```

Note: In the above string there should not be any ‘a’ after ‘b’.
Here as we can see that each string of the language containing a and b whose power is greater or equal to 0 but the below language is not accepted by this DFA because some of the string of the below language does not contain a and b whose power is greater or equal to 0 or they might not follow the format of a and b i.e, there should not be any ‘a’ after ‘b’.

```
L2 = {ba, baa, bbaaa.............}
```

This language L2 is not accepted by this required DFA because it’s string contain ‘a’ after ‘b’.
The state transition diagram of the desired language will be like below:



In the above DFA, the state ‘X’ is the initial and final state which on getting ‘a’ as the input it remains in the state of itself and on getting ‘b’ as the input it transit to the final ‘Y’ which on getting ‘b’ as the input it remains in the state of itself and on getting ‘a’ as the input it transit to dead state ‘Z’.

The state ‘Z’ is called a dead state because it can not go to the final state on getting any input alphabet.

TOC | Designing Deterministic Finite Automata (Set 7)

Problem-1: Construction of a minimal DFA accepting set of strings over {a, b} in which $a^n b^m c^l$, where n, m and l is greater than or equal to 0.

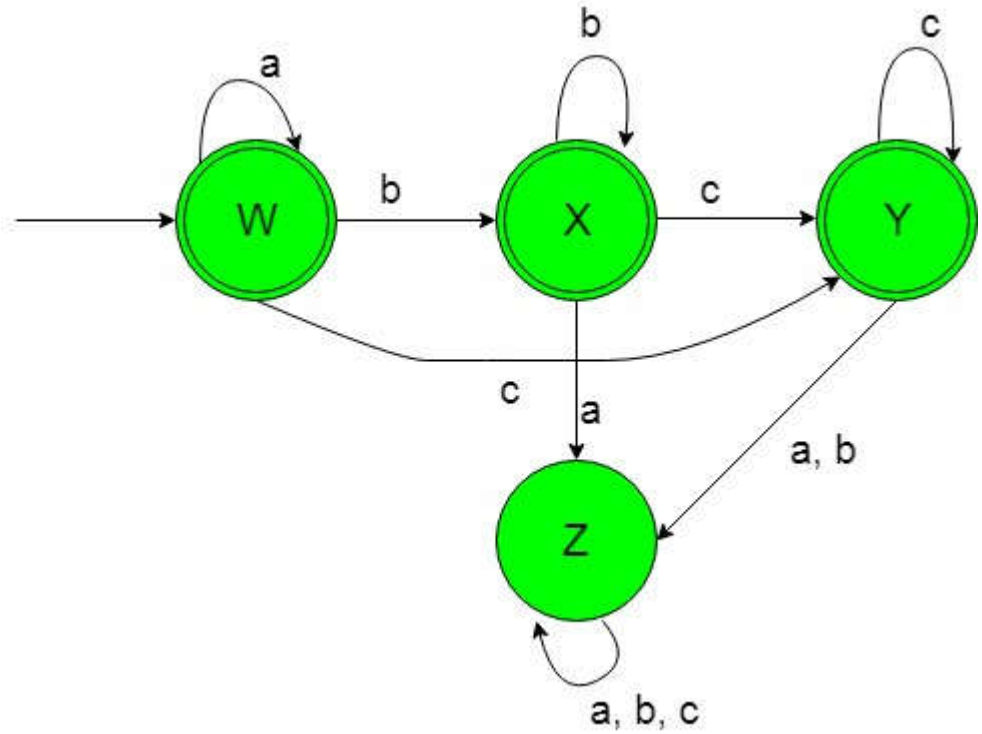
Explanation: The desired language will be like:

```
L1 = {ε, a, aa, aaa, b, bb, bbb, c, cc, ccc, abc, ab, ac, ..........}
```

Note: In the above string there should must be order like abc i.e, there should not be any ‘a’ after ‘b’ or any ‘a’ after ‘c’ etc.
Here as we can see that each string of the language containing a, b and c whose power is greater or equal to 0 but the below language is not accepted by this DFA because some of the string of the below language does not contain a, b and c whose power is greater or equal to 0 or they might not follow the format of a, b and c i.e, there should not be any ‘a’ after ‘b’ or any ‘a’ after ‘c’ etc.

```
L2 = {ba, bac, bbacaa.............}
```

This language L2 is not accepted by this required DFA because it’s string contain ‘a’ after ‘b’ and ‘a’ after ‘c’ etc.
The state transition diagram of the desired language will be like below:



In the above DFA, the state 'W' is the initial and final state which on getting 'a' as the input it remains in the state of itself, on getting 'b' as the input it transit to the final state 'X' and on getting 'c' as the input it transits to another final state 'Y'. The state 'X' is the final state which on getting 'b' as the input it remains in the state of itself, on getting 'c' as the input it transits to another final state 'Y' and on getting 'a' as the input it transits to a dead state 'Z'.

Another final state 'Y' on getting 'c' as the input it remains in the state of itself and on getting input either as 'a' or 'b' it transits to the same dead state 'Z'. The state 'Z' is called a dead state because it can not go to any of the final states on getting any of the input alphabets.

Problem-2: Construction of a minimal DFA accepting set of strings over {a, b} in which $a^n b^m c^l$, where n, m and l is greater than or equal to 1.

Explanation: The desired language will be like:

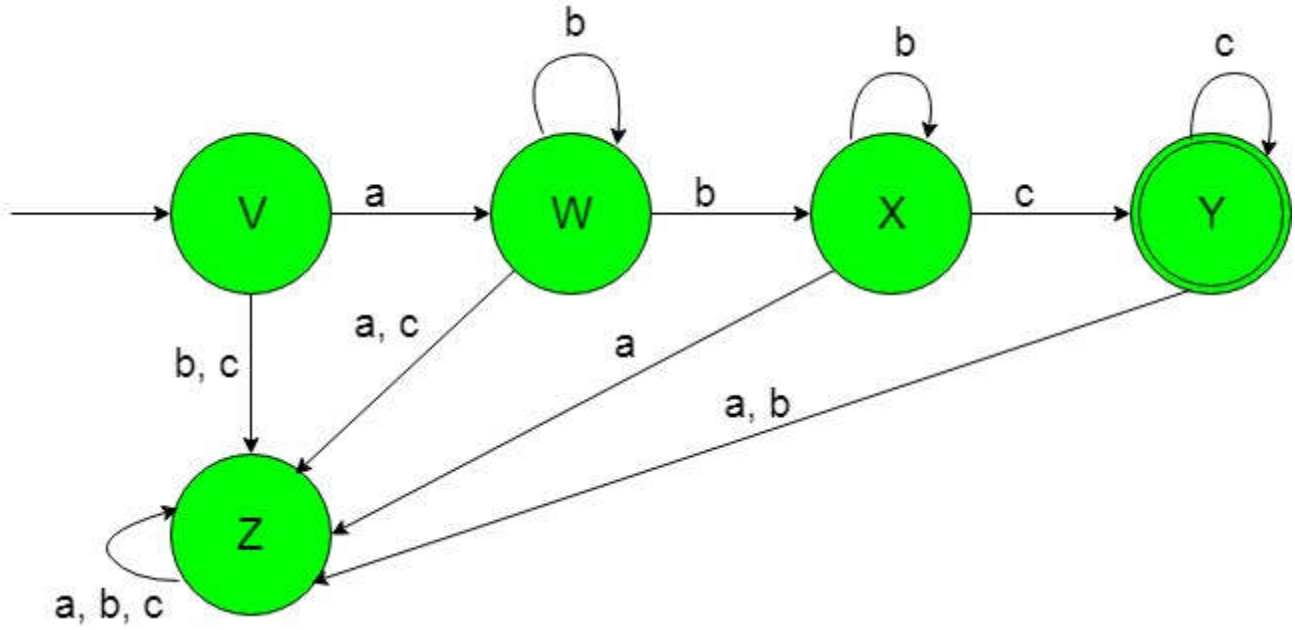
```
L1 = {abc, aabc, aabbc, aabbcc, abbc, ..........}
```

Note: In the above string there should must be order like abc i.e, there should not be any 'a' after 'b' or any 'c' after 'a' etc.

Here as we can see that each string of the language containing a, b and c whose power is greater or equal to 1 but the below language is not accepted by this DFA because some of the string of the below language does not contain a, b and c whose power is greater or equal to 1 or they might not follow the format of a, b and c i.e, there should not be any 'a' after 'b' or any 'c' after 'a' etc.

```
L2 = {ba, bac, bbacaa.............}
```

This language L2 is not accepted by this required DFA because it's string contain 'a' after 'b' and 'c' after 'a' etc.
The state transition diagram of the desired language will be like below:



In the above DFA, the initial state 'V' on getting 'a' as the input it transit to a state 'W' and on getting 'b' or 'c' as input it transit to a dead state 'Z'. The state 'W' on getting 'b' as the input it either remains in the state itself or transit to a state 'X' and on getting 'a' or 'c' it transit to the same dead state 'Z'. The state 'X' on getting 'b' as the input it remains in the state of itself and on getting 'c' as the input it transits to the final state 'Y' and on getting 'a' as the input it transit to the same dead state 'Z'.

The final state 'Y' on getting 'c' as the input it remains in the state of itself and on getting 'a' or 'b' as the input it transits to the same dead state 'Z'. The state 'Z' is called a dead state this is because it can not go to final state ever on getting any of the input alphabets.

TOC | Designing Deterministic Finite Automata (Set 8)

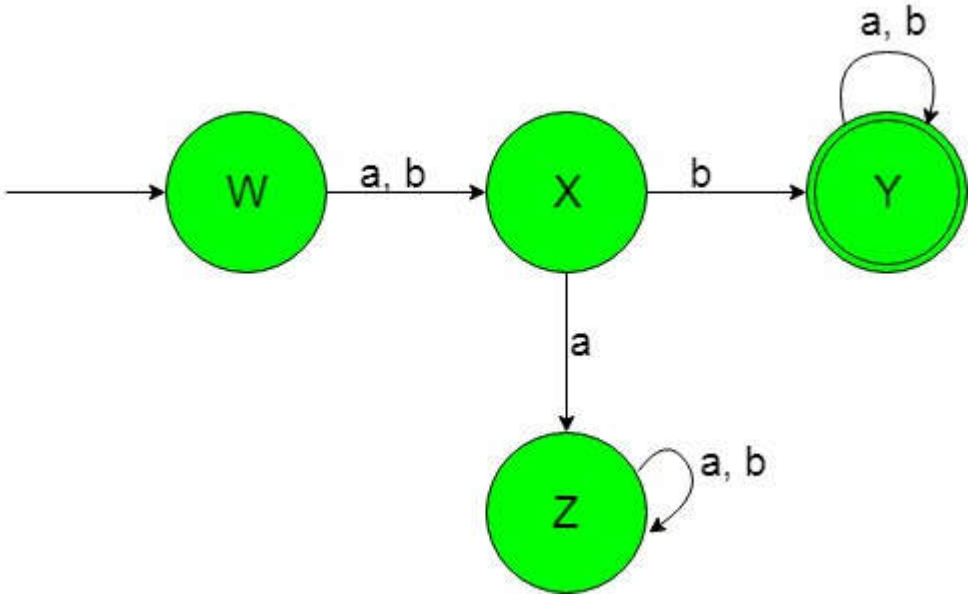
Problem-1: Construction of a minimal DFA accepting a set of strings over {a, b} in which the second symbol from left-hand side is always 'b'.
Explanation: The desired language will be like:

```
L1 = {ab, aba, abaa, bb, bb, bbbb, ..........}
```

Here as we can see that each string of the language containing 'b' as the second symbol from left-hand side but the below language is not accepted by this DFA because some of the string of the below language does not contain 'b' as the second symbol from the left-hand side.

```
L2 = {ba, ba, babaaa.............}
```

This language L2 is not accepted by this required DFA because it does not contain ‘b’ as the second symbol from the left-hand side.
The state transition diagram of the desired language will be like below:



In the above DFA, The state ‘W’ is the initial state which on getting either ‘a’ or ‘b’ as the input it transit to a state ‘X’. The state ‘X’ on getting ‘b’ as the input it transits to the final state ‘Y’ and on getting ‘a’ as the input it transit to a dead state ‘Z’.

The final state ‘Y’ on getting either ‘a’ or ‘b’ as the input it remains in the state of itself. The dead state ‘Z’ is called dead because it can not go to the final state on getting any of the input alphabets.

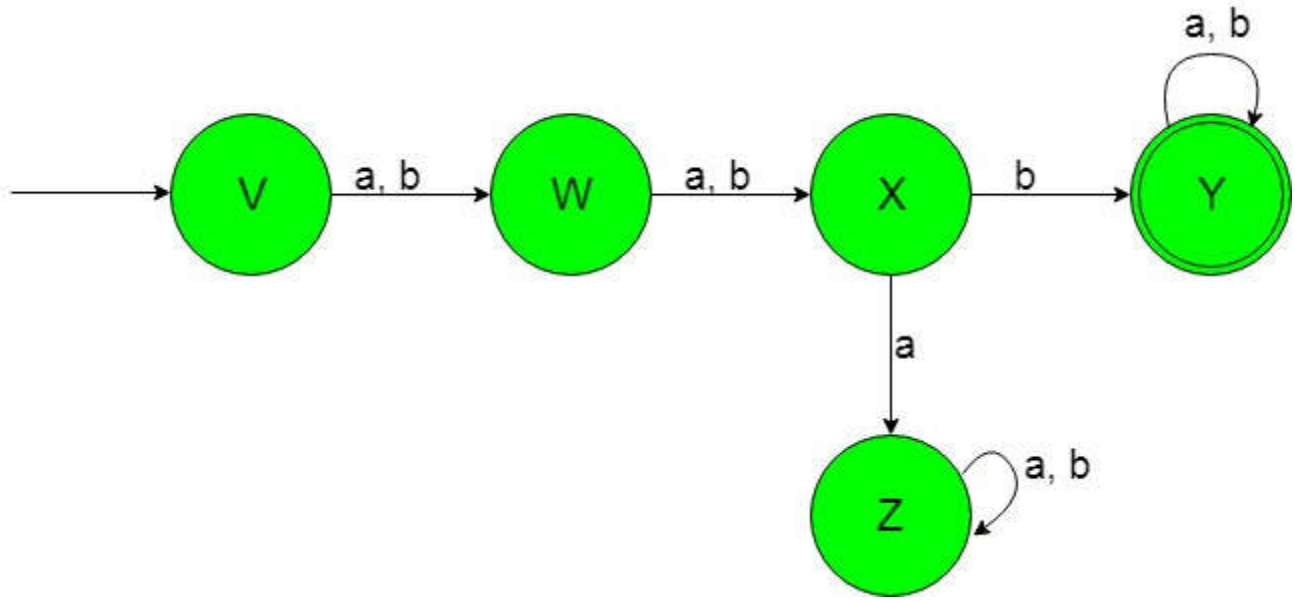
Note: The number of states in the above DFA is (n+2), where ‘n’ is the number from the left-hand side of the string used in the above language.
Problem-2: Construction of a minimal DFA accepting a set of strings over {a, b} in which the third symbol from left-hand side is always ‘b’.
Explanation: The desired language will be like:

```
L1 = {aab, baba, aabaa, bbb, abb, bbbb, ..........}
```

Here as we can see that each string of the language containing ‘b’ as the third symbol from left-hand side but the below language is not accepted by this DFA because some of the string of the below language does not contain ‘b’ as the third symbol from the left-hand side.

```
L2 = {baa, aba, baabaaa.............}
```

This language L2 is not accepted by this required DFA because it does not contain ‘b’ as the third symbol from the left-hand side.
The state transition diagram of the desired language will be like below:



In the above DFA, The state ‘V’ is the initial state which on getting either ‘a’ or ‘b’ as the input it transits to the state ‘W’. The state ‘W’ is a state which on getting either ‘a’ or ‘b’ as the input it transits to a state ‘X’. The state ‘X’ on getting ‘b’ as the input it transits to the final state ‘Y’ and on getting ‘a’ as the input it transit to a dead state ‘Z’.

The final state ‘Y’ on getting either ‘a’ or ‘b’ as the input it remains in the state of itself. The dead state ‘Z’ is called dead because it can not go to the final state on getting any of the input alphabets.

Note: The number of states in the above DFA is (n+2), where ‘n’ is the number from the left-hand side of the string used in the above language.

TOC | Designing Deterministic Finite Automata (Set 9)

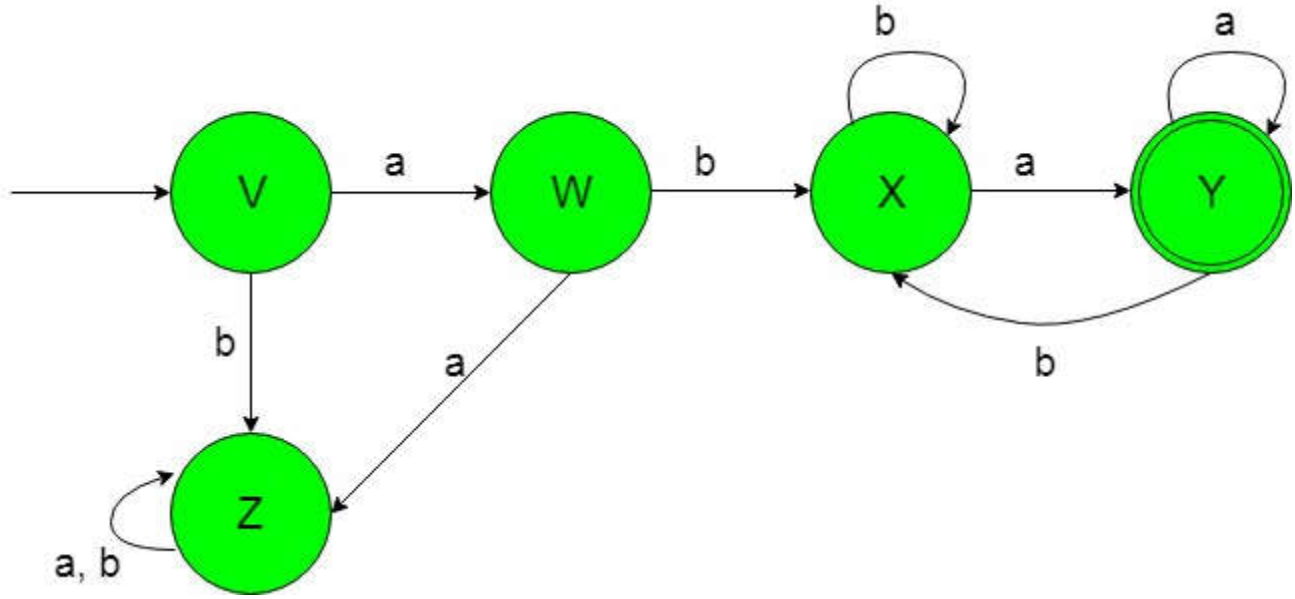
Problem-1: Construction of a minimal DFA accepting a set of strings over {a, b} in which {abwa / wε{a, b}*}.
Here ‘w’ is the substring containing alphabets over any number of ‘a’ and ‘b’ ranging from zero to infinity.
Explanation: The desired language will be like:

```
L1 = {abεa, abaaaa, ababaaa, abaabbaaa}
```

Here as we can see that each string of the above language having constant initial i.e, {ab} and final {a} substring but ‘w’ having a substring over {a, b} whose alphabets ranging from zero to infinity but the below language is not accepted by this DFA because it’s strings are not following the format of string.

```
L2 = {baa, aabaa, baabaaa.............}
```

The state transition diagram of the desired language will be like below:



In the above DFA, the initial state ‘V’ on getting ‘a’ as the input it transits to a state ‘W’ and on getting ‘b’ as the input it transits to a dead state ‘Z’. The state ‘W’ on getting ‘a’ as the input it transits to the dead state ‘Z’ and on getting ‘b’ as the input it transits to a state ‘X’. The state ‘X’ on getting ‘b’ as the input it remains in the state of itself and on getting ‘a’ as the input it transits to a final state ‘Y’.

The final state ‘Y’ on getting ‘a’ as the input it remains in the state of itself and on getting ‘b’ as the input it transits to the state ‘X’. The state ‘Z’ is called as the dead state because it can not go to the final state ever on getting any of the input alphabets.

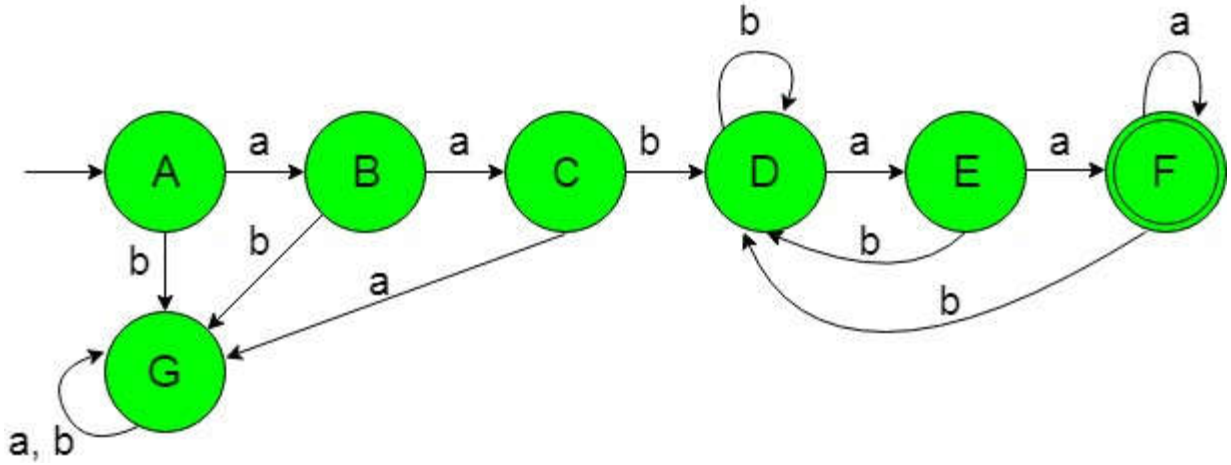
Problem-2: Construction of a minimal DFA accepting a set of strings over {a, b} in which $\{a^2bwa^2 / w\varepsilon\{a, b\}^*\}$.
Here ‘w’ is the substring containing alphabets over any number of ‘a’ and ‘b’ ranging from zero to infinity.
Explanation: The desired language will be like:

```
L1 = {aabεaaa, aabaaaa, aababaaa, aabaabbaaa}
```

Here as we can see that each string of the above language having constant initial i.e, $\{a^2b\}$ and final $\{a^2\}$ substring but ‘w’ having a substring over {a, b} whose alphabets ranging from zero to infinity but the below language is not accepted by this DFA because it’s strings are not following the format of string.

```
L2 = {baa, abaa, baabaaa.............}
```

The state transition diagram of the desired language will be like below:



In the above DFA, The initial state ‘A’ on getting ‘a’ as the input it transits to a state ‘B’ and on getting ‘b’ as the input it transit to a dead state ‘G’. The state ‘B’ on getting ‘a’ as the input it transits to a state ‘C’ and on getting ‘b’ as the input it transit to a dead state ‘G’. The state ‘C’ on getting ‘b’ as the input it transits to a state ‘D’ and on getting ‘a’ as the input it transit to a dead state ‘G’. The state ‘D’ on getting ‘a’ as the input it transits to a state ‘E’ and on getting ‘b’ as the input it remains in the state of itself. The state ‘E’ on getting ‘a’ as the input it transits to a final state ‘F’ and on getting ‘b’ as the input it transits to the state ‘D’.

The final state ‘F’ on getting ‘a’ as the input it remains in the state of itself and on getting ‘b’ as the input it transits to the state ‘D’. The dead state ‘G’ is called dead because it can not go to the final state on getting any of the input alphabets.

TOC | Designing Deterministic Finite Automata (Set 10)

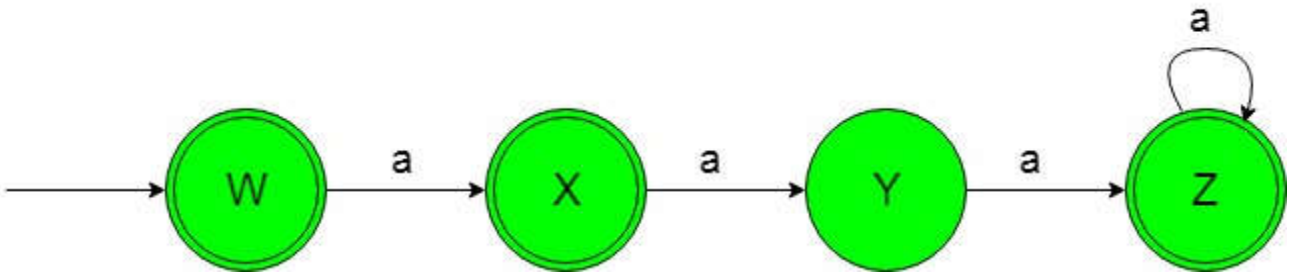
Problem-1: Construction of a minimal DFA accepting set of strings over {a} in which $\{a^n \mid n \geq 0, n \neq 2$ i.e, ‘n’ should be greater than 0 and not equal to 2}.
Explanation: The desired language will be like:

```
L1 = {ε, a, aaa, aaaa, aaaaa, ..................}
```

Here ε is taken as string because value of ‘n’ is greater than or equal to zero and rest of the strings are having ‘a’ to the power of any positive natural number but not 2.
Below language is not accepted by this DFA because some of the string containing ‘a’ to the power 2.

```
L2 = {aa, aaaaa, ..........}
```

This language L2 is not accepted by this required DFA because of its string containing ‘a’ to the power of 2.
The state transition diagram of the desired language will be like below:



In the above DFA, the initial and final state'W' on getting 'a' as the input it transits to a final state 'X'. The final state 'X' on getting 'a' as the input it transits to a state 'Y'. The state 'Y' on getting 'a' as the input it transits to a final state 'Z' which on getting any number of 'a' it remains in the state of itself.

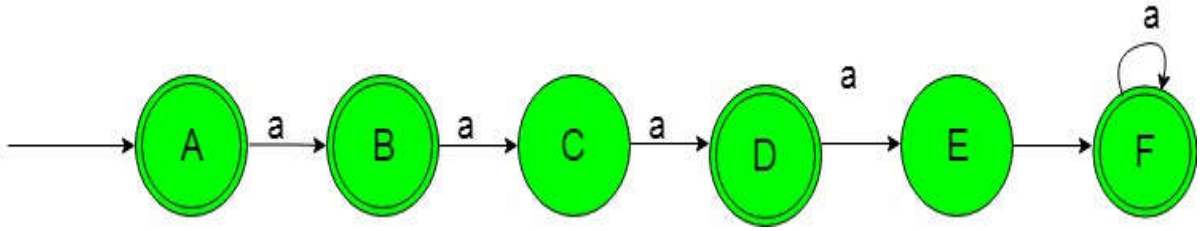
Problem-2: Construction of a minimal DFA accepting set of strings over {a} in which $\{a^n \mid n \geq 0, n \neq 3, n \neq 4$ i.e, 'n' should be greater than 0 and not equal to 2 and 4}.
Explanation: The desired language will be like:

```
L1 = {ε, a, aa, aaaaa, aaaaaa, ..................  }
```

Here ε is taken as string because value of 'n' is greater than or equal to zero and rest of the strings are having 'a' to the power of any positive natural number but not 2 and 4. Below language is not accepted by this DFA because some of the string containing 'a' to the power of 2 and 4.

```
L2 = {aa, aaaaa, aaaaaaaaaa, .............  }
```

The state transition diagram of the desired language will be like below:



In the above DFA, the initial and final state 'A' on getting 'a' as the input it transits to a final state 'B'. The final state 'B' on getting 'a' as the input it transits to a state 'C'. The state 'C' on getting 'a' as the input it transits to a final state 'D'. The final state 'D' on getting 'a' as the input it transits to a state 'E'. The state 'E' on getting 'a' as the input it transits to a final state 'F'. The final state 'F' on getting 'a' as the input it remains in the state of itself.

TOC | Designing Deterministic Finite Automata (Set 11)

Problem: Construction of a minimal DFA accepting set of strings over {a, b} in which Number of a(w) mod 2 = 0 or Number of b(w) mod 2 = 0 i.e, number of 'a' should be divisible by 2 or number of 'b' should be divisible by 2 or both are divisible by 2, where 'w' is the any string over {a, b}.
Explanation: The desired language will be like:

```
L1 = {ε, aa, aabb, aab, bb, bba, ..........}
```

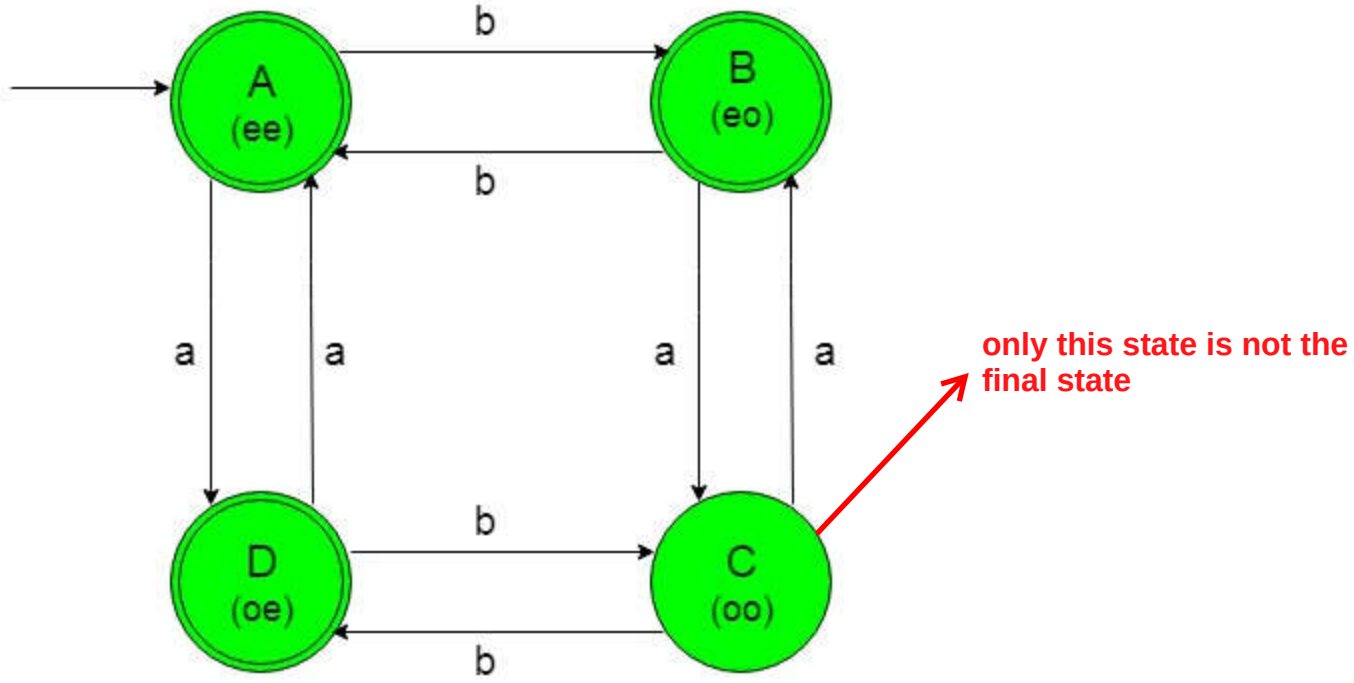
Here as we can see that each string of the above language are satisfying the condition of the given problem i.e, here ε is accepted because the number of 'a' and 'b' both are zero and rest of the strings are having 'a' divisible by 2 or 'b' divisible by 2 or both are divisible by 2.

But the below language is not accepted by this DFA because it's strings are not satisfying the condition of the given problem.

```
L2 = {ba, bbba, baaa, ............}
```

Here as we can see that none of the strings of the above language is satisfying the condition of the given problem i.e, either 'a' or 'b' or both of any of the above strings are not divisible by 2.

The state transition diagram of the desired language will be like below:



In the above DFA, In each state there is a state name as 'A' and just below of it there is (ee) which indicates the number of 'a' is even (e) and number of 'b' is even (e) too. For

state name as ‘B’ and just below of it there is (eo) which indicates the number of ‘a’ is even (e) and number of ‘b’ is odd (o) and so on.

- The initial and final state ‘A’ on getting ‘a’ as the input it transits to a final state ‘D’ and on getting ‘b’ as the input it transits to an another final state ‘B’.
- The final state ‘B’ on getting ‘a’ as the input it transits to a state ‘C’ and on getting ‘b’ as the input returns back to the initial state ‘A’.
- The another final state ‘D’ on getting ‘b’ as the input it transits to a state ‘C’ and on getting ‘a’ as the input it returns back to the initial state ‘A’.
- The state ‘C’ on getting ‘b’ as the input it transits to the final state ‘D’ and on getting ‘a’ as the input it returns back to the state ‘B’.

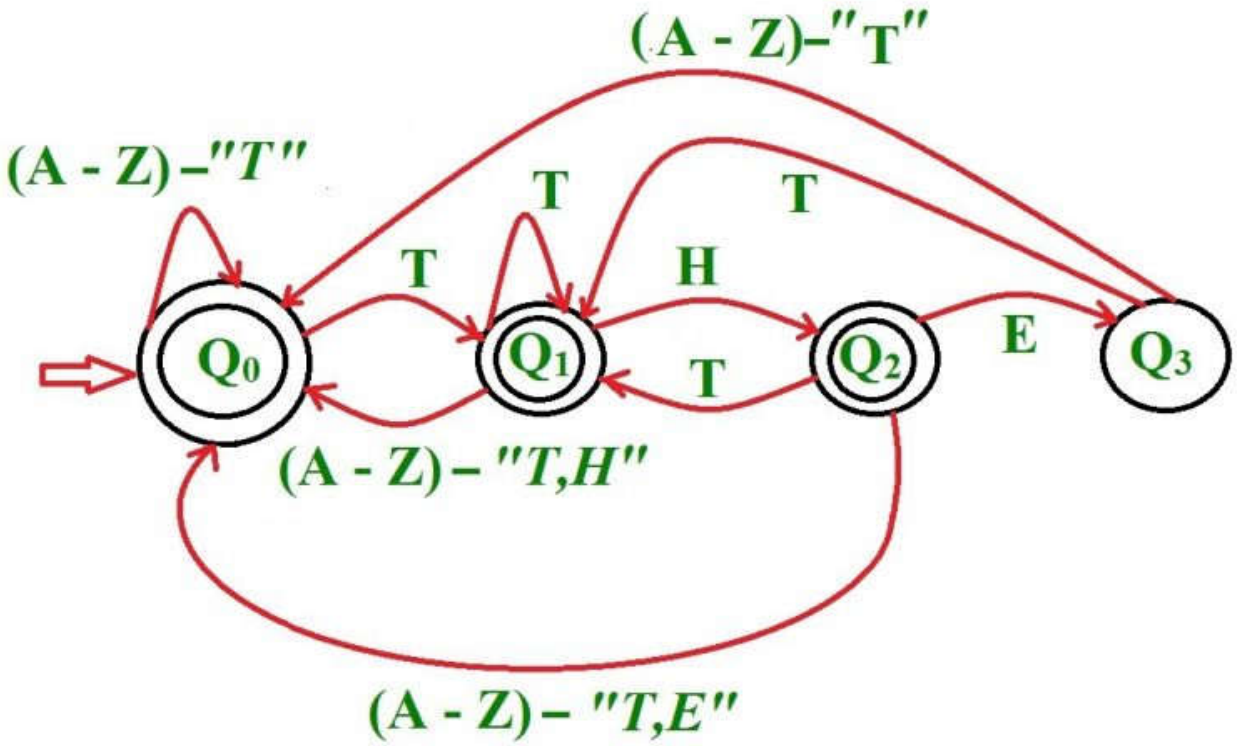
DFA for Strings not ending with “THE”

Problem – Accept Strings that not ending with substring “THE”. Check if a given string is ending with “the” or not. The different forms of “the” which are avoided in the end of the string are:

```
"THE", "ThE", "THe", "tHE", "thE", "The", "tHe" and "the"
```

All those strings that are ending with any of the above mentioned forms of “the” are not accepted.

Determinitis finite automata (DFA) of strings that not ending with “THE” –



The initial and starting state in this dfa is Qo

Approach used in the program –
In this program, consider the 4 states to be 0, 1, 2 and 3. Now let us take a variable named DFA which will be initially 0. Whenever any transition takes place, it will update the value of DFA with the number associated with new state.

Example : If a transition occurs from state 0 to state 1 then the value of DFA will be updated to 1. If a transition occurs from state 2 to state 3 then the value of dfa will be

updated to 3. In this way, apply this algorithm on entire string and if in the end, then reach state 0, 1 or 2 then our string will be accepted otherwise not.

```
Input : XYzabCthe
Output : NOT ACCEPTED
Input :  Themaliktth
Output :  ACCEPTED
```

DFA of a string with at least two 0's and at least two 1's

Problem – Draw detereministic finite automata (DFA) of a string with at least two 0's and at least two 1's.
The first thing that come to mind after reading this question us that we count the number of 1's and 0's. Thereafter if they both are at least 2 the string is accepted else not accepted. But we do not have any concept of memory in a DFA so we cannot do it by this method.

```
Input  : 1 0 1 1 0 0
Output : Accepted


Input  : 1 1 1 0 1
Output : Not accepted
```

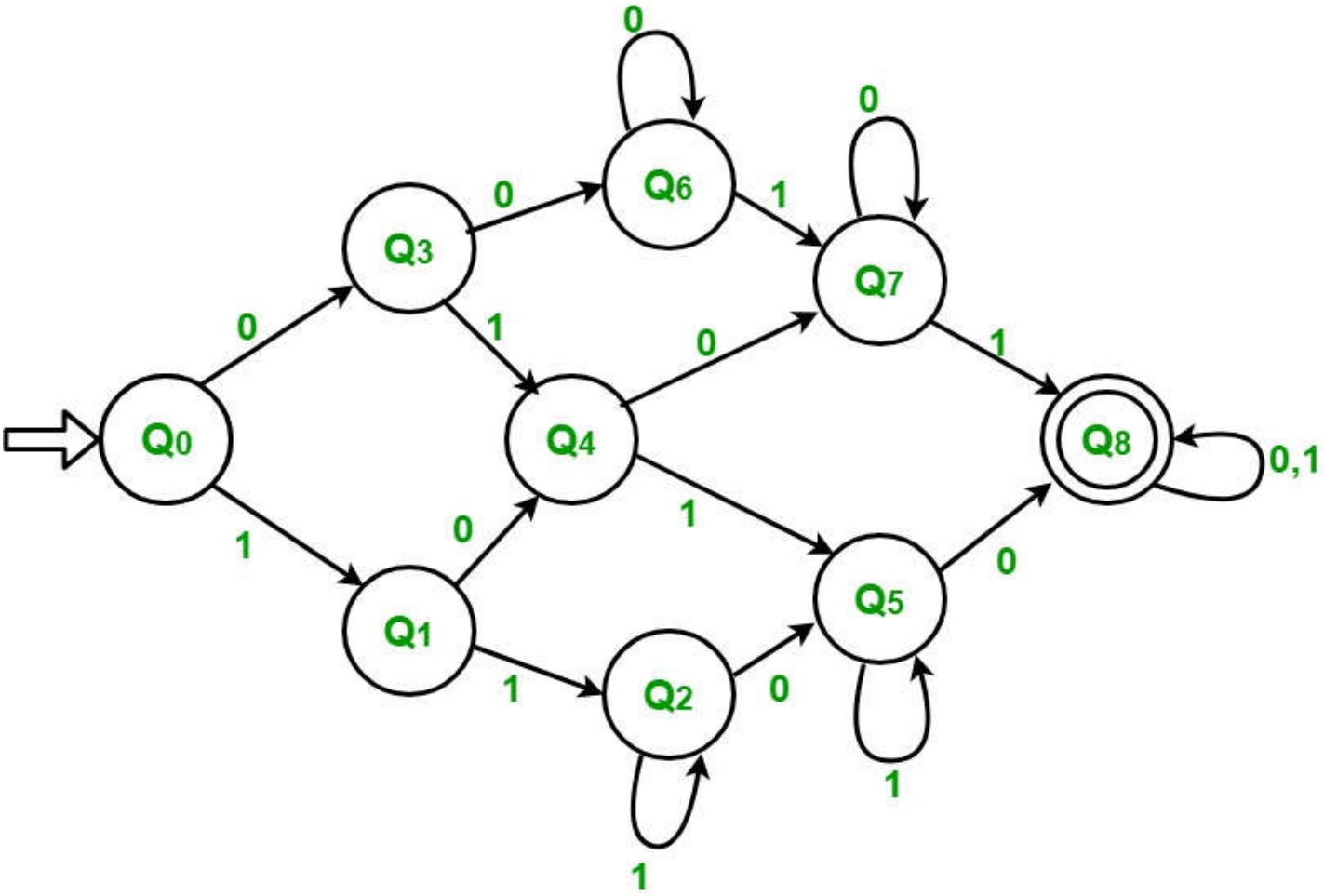
Approach Used –
The first thing we observe is that both 0's and 1's should be at least 2. If any of these is less than 2, then string will not be accepted. In this string will be accepted in only last case where both 0's and 1's will be at least 2.

STATE	COUNT OF 0	COUNT OF 1
Q0	0	0
Q1	0	1
Q2	0	>=2
Q3	1	0
Q4	1	1

STATE	COUNT OF 0	COUNT OF 1
Q5	1	>=2
Q6	>=2	0
Q7	>=2	1
Q8 ACCEPTED	>=2	>=2

Initially count of both 0 and 1 is zero and we are on state Q0.

- **Step-1:** If input is 1 then count of 1 increases to 1. Goto state Q1
  If input is 0 then count of 0 increases to 1. Goto state Q3
- **Step-2:** If input is 1 then count of 1 increases to 2. Goto state Q2
  If input is 0 then count of 0 increases to 1. Goto state Q4
- **Step-3:** If input is 1 then count of 1 keeps increasing by 1. Remain in the same state
  If input is 0 then count of 0 increases to 1. Goto state Q5
- **Step-4:** If input is 1 then count of 1 increases to 1. Goto state Q4
  If input is 0 then count of 0 increases to 2. Goto state Q6
- **Step-5:** If input is 1 then count of 1 increases to 2. Goto state Q5
  If input is 0 then count of 0 increases to 2. Goto state Q7
- **Step-6:** If input is 1 then count of 1 keeps increasing by 1. Remain in the same state.
  If input is 0 then count of 0 increases to 2. Goto state Q8
- **Step-7:** If input is 1 then count of 1 increases to 1. Goto state Q7
  If input is 0 then count of 0 keeps increasing by 1. Remain in the same state.
- **Step-8:** If input is 1 then count of 1 increases to 2. Goto state Q8
  If input is 0 then count of 0 keeps increasing by 1. Remain in the same state.
- **Step-9:** If input is 1 then count of 1 keeps increasing by 1. Remain in the same state.
  If input is 0 then count of 0 keeps increasing by 1. Remain in the same state.
  If string is finished then ACCEPTED



DFA for accepting the language L = { $a^n b^m$ | n+m=even }

Design a deterministic finite automata(DFA) for accepting the language L

=

For creating DFA for language L = { a^n b^m ; n+m=even } use elementary mathematics which says-
even + even = even and odd + odd = even

Examples:

```
Input:  a a b b     // n = 2, m = 2, 2 + 2 = 4 (even)
Output:  ACCEPTED

Input:  a a a b b b b      // n = 3, m = 4, 3 + 4 = 7 (odd)
Output:  NOT ACCEPTED
```

```
Input:  a a a b b b     // n = 3, m = 3, 3 + 3 = 6 (even)
Output:  ACCEPTED
```

Approaches:

There is 2 cases which results in acceptance of string:

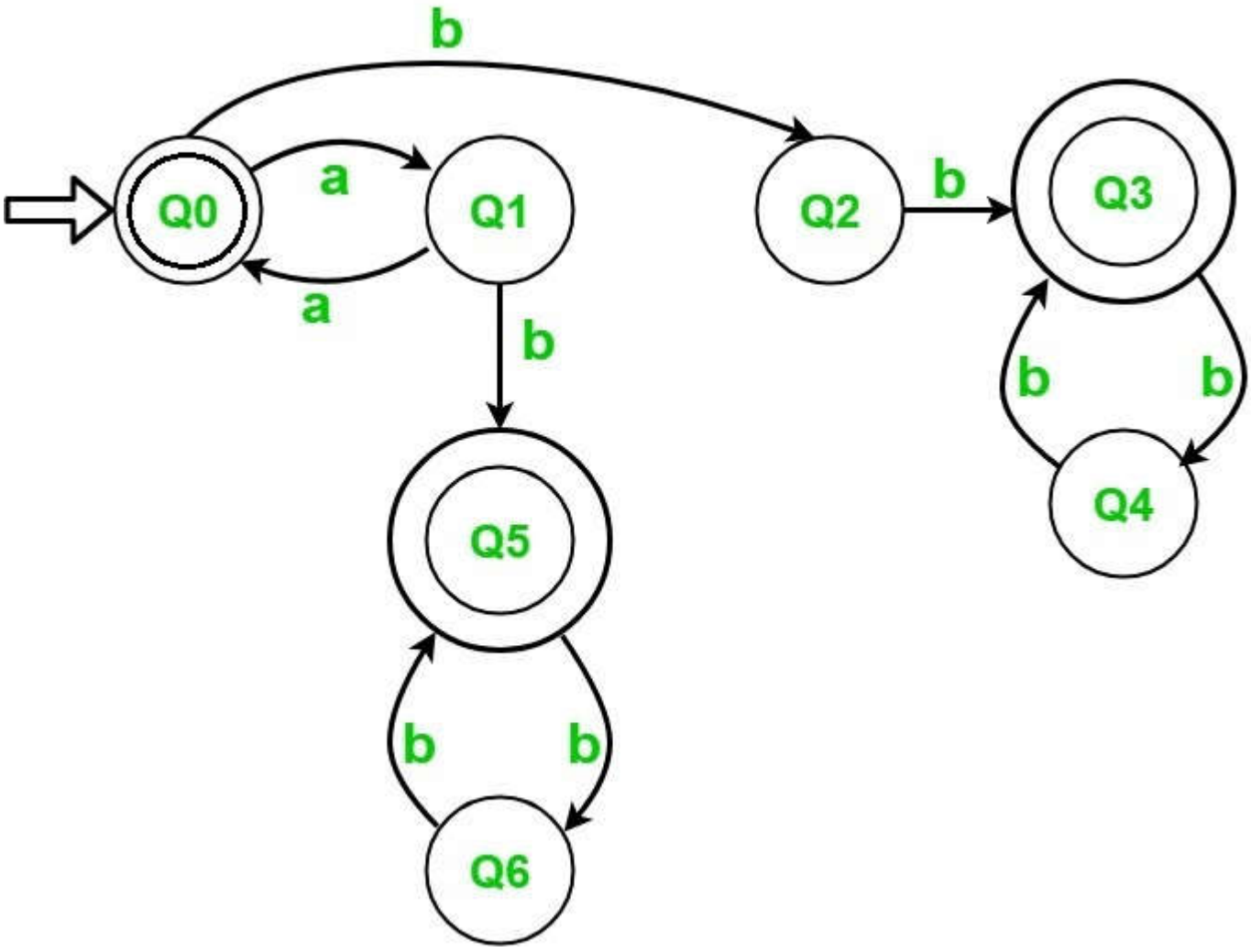
1. If both n and m are even then their sum will be even
2. If both n and m are odd then their sum will be even

Any other combination result is the rejection of the input string.

Description:

Given DFA has 2 parts. First part consisting of states 0, 1, 5 and 6 which is for both n and m being odd. Second part consist of states 2, 3 and 4 is for both n and m is even.

DFA State Transition Diagram:



DFA machines accepting odd number of 0's or/and even number of 1's

Prerequisite – Designing finite automata

Problem – Contruct a DFA machine over input alphabet = {0, 1}, that accepts:

1. Odd number of 0's or even number of 1's
2. Odd number of 0's and even number of 1's
3. Either odd number of 0's or even number of 1's but not the both together

Solution – Let first design two separate machines for the two conditions:

- Accepting only odd number of 0's
- Accepting only even number of 1's

Then, merge these two and find the required final states.

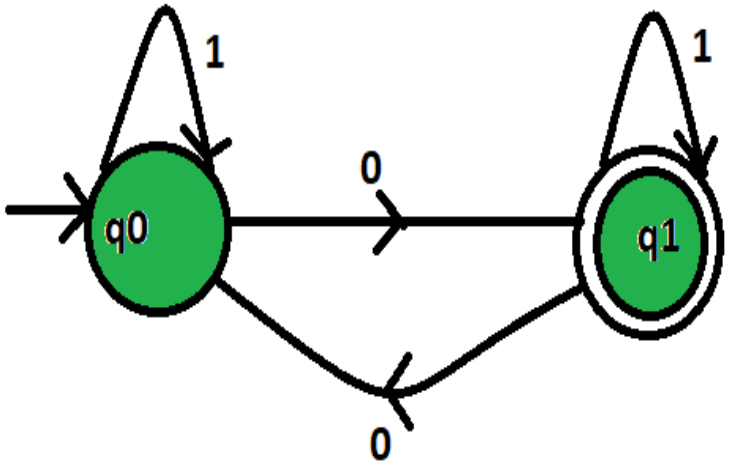


Figure 1

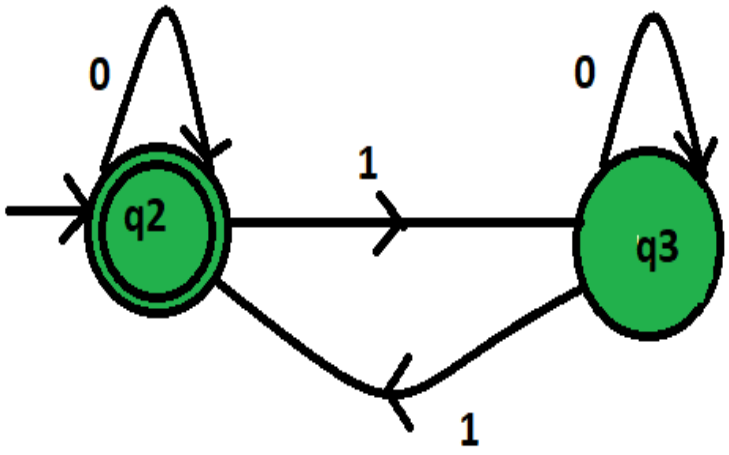
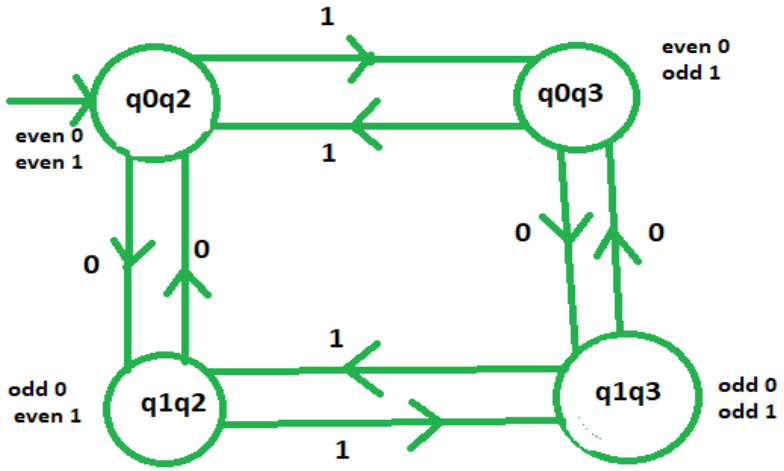


Figure 2

To merge these two machines, we will take the Cartesian product of the states of these two machines:

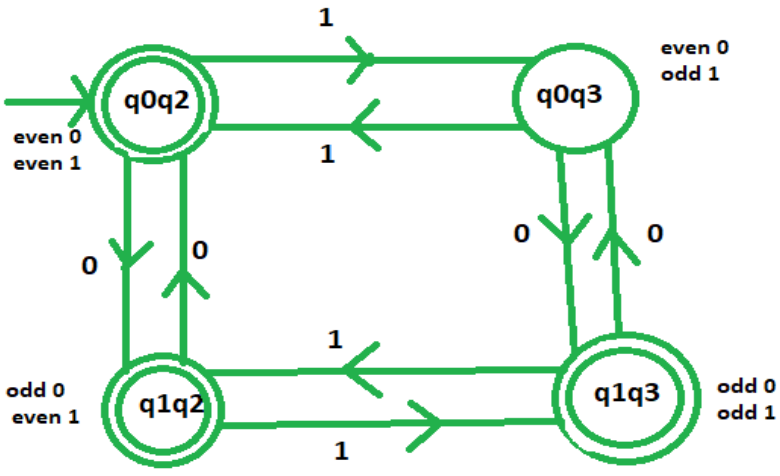


DFA

Initial state of these DFA will be the state which contains the initial states of those two separate machines. As, q0 and q2 are the initial states thus, q0q2 is the initial state of the final DFA.

Now start designing all the DFAs one by one:

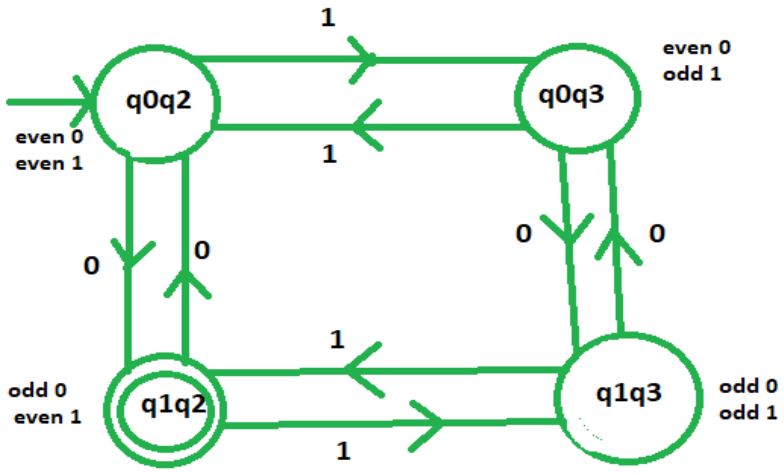
1. **Odd number of 0's or even number of 1's:**
   This machine accept that languages which contains either odd no. of 0's or even no. of 1's. As we know that q1 indicates odd no. of 0's and q2 indicates even no. of 1's. So, the final states of the required DFA will contain either q1 or q2.
   .'. Final states = {(q0q2), (q1q2), (q1q3)}



DFA

This is our required DFA which accept the languages containing odd no. of 0's or even no. of 1's.

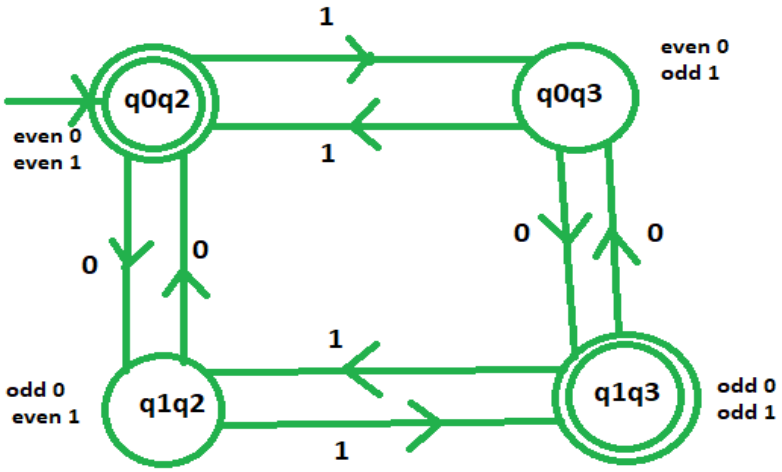
2. **Odd number of 0's and even number of 1's:**
   This machine accept that languages which contains odd no. of 0's and even no. of 1's. As we know that q1 indicates odd no. of 0's and q2 indicates even no. of 1's. So, the final states of the required DFA will contain both q1 and q2.
   .'. Final state = {(q1q2)}



DFA

This is our required DFA which accept the languages containing odd no. and 0's or even no. of 1's.

3. **Either odd number of 0's or even number of 1's but not the both together:**
   This machine accept that languages which contains either odd no. of 0's or even no. of 1's but not that languages which contains both odd no. of 0's and even no. of 1's. As we know that q1 indicates odd no. of 0's and q2 indicates even no. of 1's. So, the final states of the required DFA will contain exactly one among q1 and q2.
   .'. Final states = {(q0q2), (q1q3)}



DFA

This is our required DFA which accept the languages containing odd no. of 0's or even no. of 1's but not the both together.

DFA of a string in which 2nd symbol from RHS is 'a'

Prerequisite – Finite Automata Introduction

Problem – Draw deterministic finite automata (DFA) of the language containing the set of all strings over {a, b} in which 2nd symbol from RHS is 'a'.
The strings in which 2nd last symbol is "a" are:

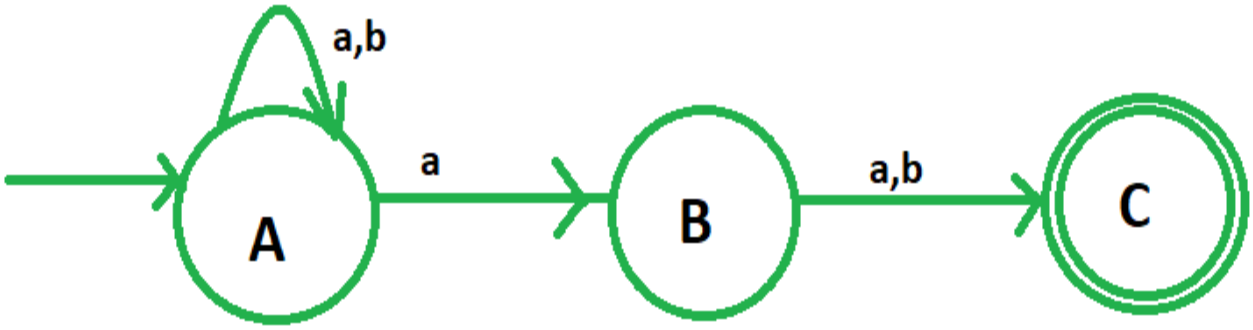
```
aa, ab, aab, aaa, aabbaa, bbbab etc
```

For example:

```
INPUT : baba

OUTPUT: NOT ACCEPTED


INPUT: aaab

OUTPUT: ACCEPTED
```

Constructing the DFA of the given problem directly is very complicated. So, here we are going to design the non-deterministic finite automata (NFA) and then convert it to the deterministic finite automata (DFA).

The NFA of the language containing all the strings in which 2nd symbol from the RHS is "a" is:



NFA

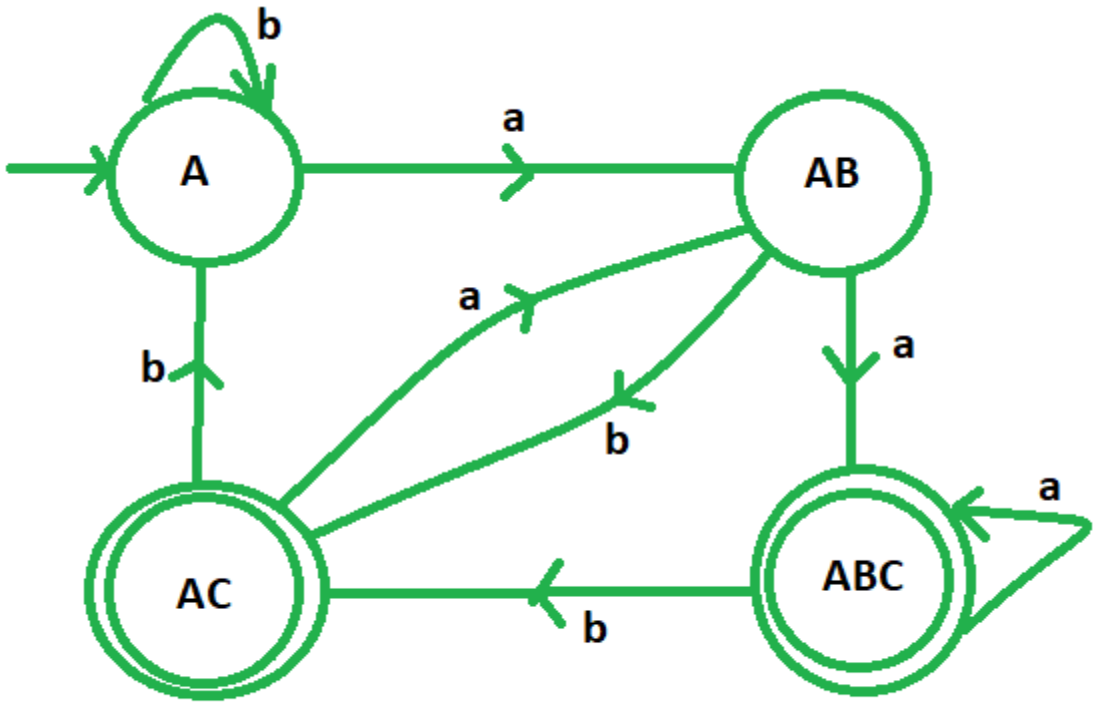
Here, A is the initial state and C is the final state.
Now, we are going to construct the state transition table of the above NFA.

	a	b
→A	{A,B}	{A}
B	{C}	{C}
(C)	0	0

After that we will draw the state transition table of DFA using subset configuration on the state transition table of NFA. We will mention all the possible transition for a and b.

	a	b
→[A]	[AB]	[A]
[AB]	[ABC]	[AC]
(ABC)	[ABC]	[AC]
(AC)	[AB]	[A]

Now it’s become very easy to draw the DFA with the help of its transition table. In this DFA, we have four different states A, AB, ABC and AC, where ABC and AC are the final states and A is the initial state of the DFA.



DFA

This is our required DFA of the language containing the set of all strings over {a, b} in which 2nd symbol from RHS is ‘a’.