Pushdown Automata (PDA)

↳ To accept a context-free language.

A pushdown automata is nothing but a $F.A + \underbrace{\text{stack}}_{\text{Memory element}}$.

→ Mathematically, PDA is defined by a seven-tuple

$$(Q, \Sigma, \delta, q_0, Z_0, F, \Gamma)$$

where,

Q : Finite set of states

Σ : input symbol

δ : Transition function

q_0 : initial state

Z_0 : It is a special symbol used to mark the bottom of the stack.

F : Set of final states $(F \subseteq Q)$

Γ : Stack Alphabet

PDA

↙ Deterministic PDA (DPDA)

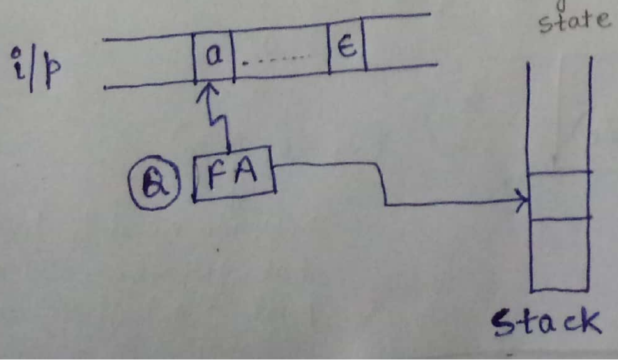
↘ Non-deterministic PDA (NPDA)

Deterministic PDA (DPDA) ↓

$$\delta : Q \times \{\Sigma \cup \epsilon\} \times \Gamma \rightarrow \underbrace{Q}_{\text{Only 1 state}} \times \underbrace{\Gamma^*}_{\text{pushing something particular}}$$

Non-deterministic PDA (NPDA) ↓

$$\delta : Q \times \{\Sigma \cup \epsilon\} \times \Gamma \rightarrow \underbrace{2^{(Q \times \Gamma^*)}}_{\text{going to more than 1 state and pushing more than 1 symbol on to the stack.}}$$



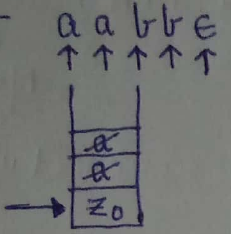
* Note :- More than 1 symbol can be pushed by making copies of itself.

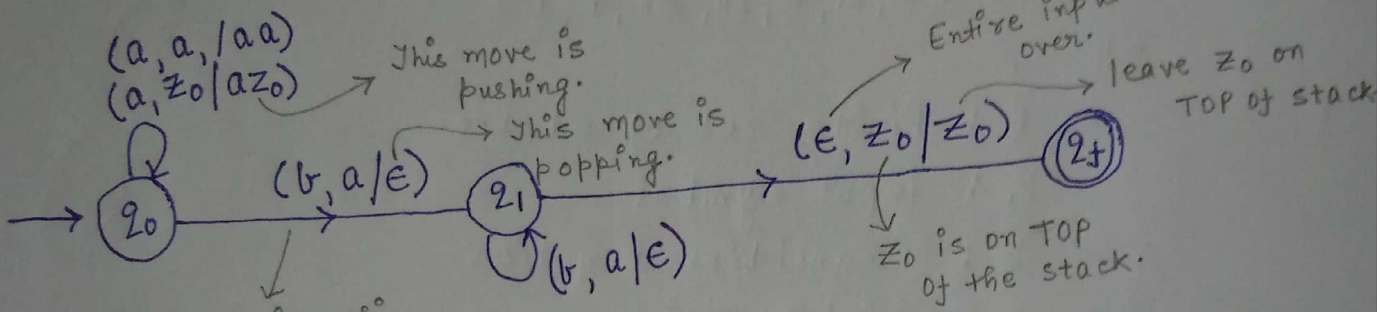
Eg:-

$L = \{ a^n b^n / n \geq 1 \}$

We have to see all a's first and count the a's and then, whenever we see a 'b', we need to match it against a's.

i.e Eg:-





This move is pushing.

This move is popping.

Entire input is over.

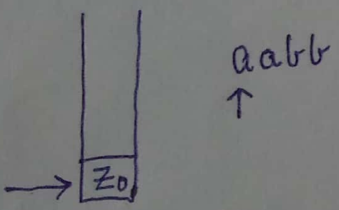
leave Z₀ on TOP of stack

Z₀ is on TOP of the stack.

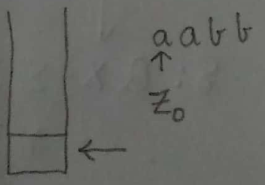
by doing this, we are remembering we have seen a 'b' & then, no 'a' is allowed.

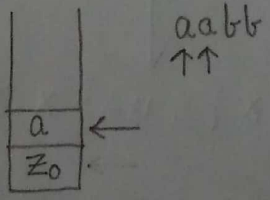
* Note :- If we stay in the same state and keep on popping 'a' for every 'b', then, after 'b' if an 'a' comes, we will still accept it due to (a, a / aa) transition which is on q₀.

*Explanation :-



* Whenever you are saying that top of the stack is Z₀, actually you are taking that symbol out of the stack and holding it in your hands.





(a, Z₀

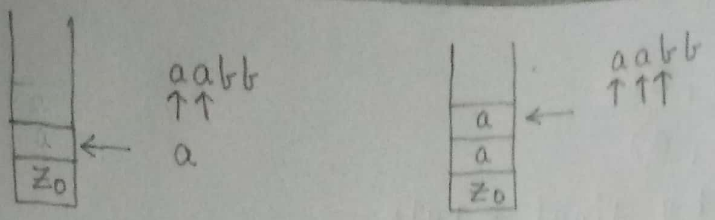
input TOP of the stack

FOR
(a, Z₀ | a Z₀)

(a, Z₀, a Z₀)

Now, we need to push 'a' but for that we need to push 'Z₀' first since we are holding it in our hands.

Now,



FOR
(a, a/aa)

Now, Above was the state transition diagram for a PDA. We can also show a PDA using transition function δ.

i.e

state, input, TOP → New State, pushed symbol

$$\delta(q_0, a, Z_0) = (q_0, aZ_0)$$
$$\delta(q_0, a, a) = (q_0, aa)$$
$$\delta(q_0, b, a) = (q_1, \epsilon)$$

POP symbol

$$\delta(q_1, b, a) = (q_1, \epsilon)$$
$$\delta(q_1, \epsilon, Z_0) = (q_f, Z_0)$$

Keep Z_0 on TOP of stack

input is over

Z_0 on TOP of stack

NEW STATE is the final state

→ One thing to observe is that we will accept the string once we reach the final state. This is known as acceptance by final state.

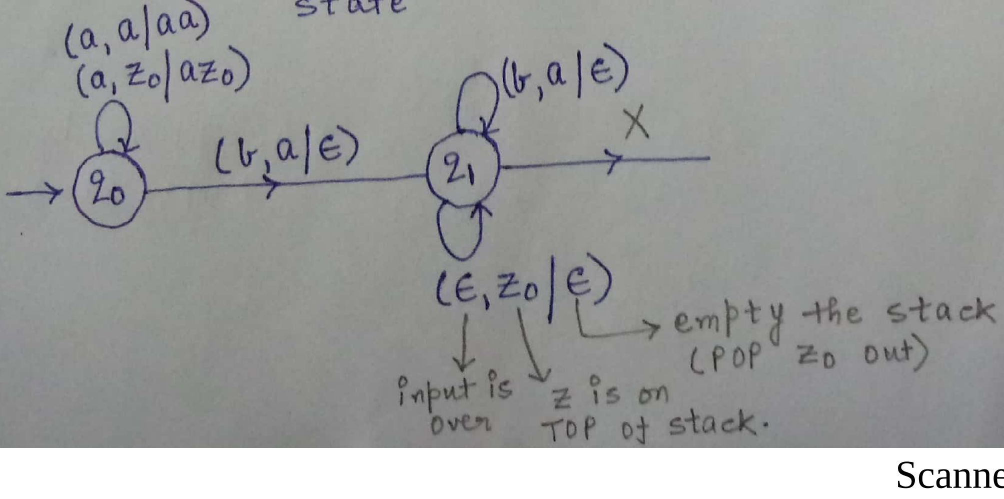
Acceptance of a string by PDA

Acceptance through final state

Acceptance by Empty Stack

* for PDA, there are 2 ways in whic a string can be accepted.

i.e



* Note :-

$\delta(q, \epsilon, Z_0) = (q_f, Z_0)$ → Acceptance by final state

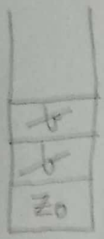
$\delta(q, \epsilon, Z_0) = (q, \epsilon)$ → Acceptance by empty stack.

Eg:- $L = \{ w : n_a(w) = n_b(w) \}$

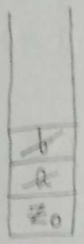
Eg:- ab bbaa
aabb baba
abab

*Approach to construct the PDA:- We will push the symbols if we do not get to POP and we will POP if we get a chance.

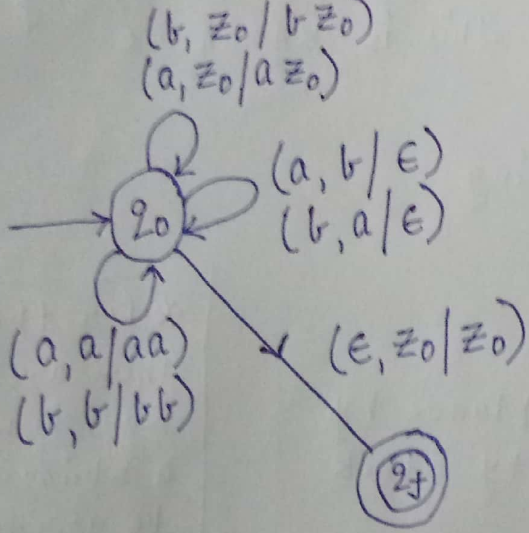
Eg:-



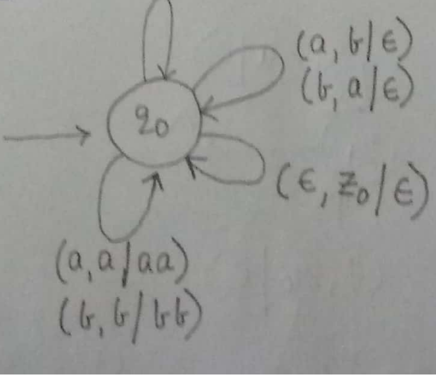
for baba



for abba



Also,

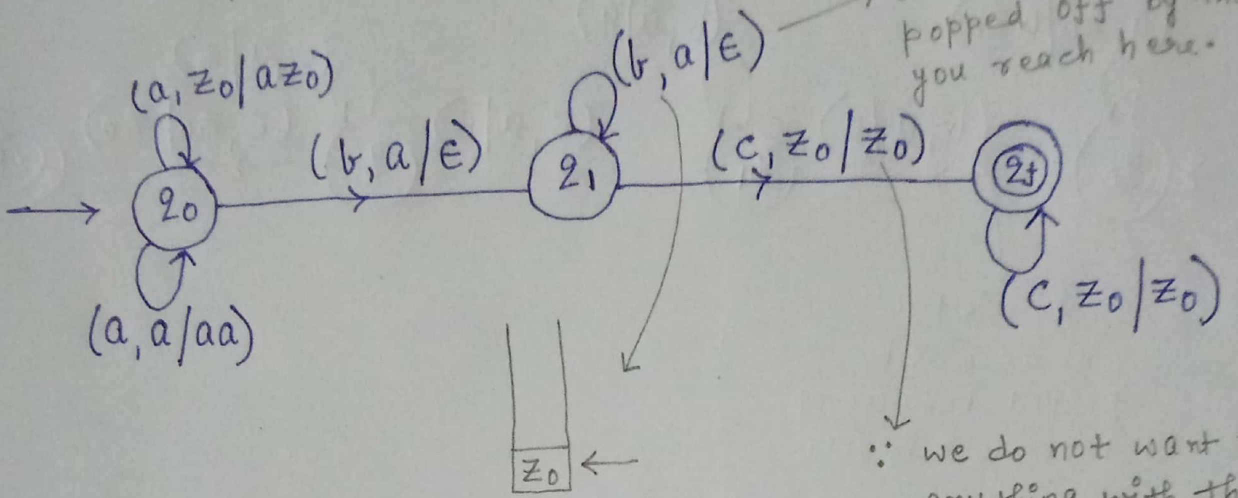


* Acceptance by Empty Stack.

Eg:- $L = \{ \underbrace{a^n b^n} c^m / n, m \geq 1 \}$

→ simply read c's

* APPROACH → a's & b's have to be compared & have to be equal.



→ Entire a's will be popped off by the time you reach here.

All a's and b's have been completely matched out.

∵ we do not want to do anything with the stack as 'c' does not require counting.

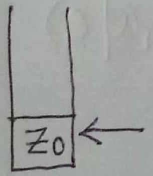
* An important note :-

In all of our examples till now, we are refering to our F.A as deterministic PDA.

But, in contrast to a DFA,

for every state, for every configuration, we are not providing a transition.

Eg:- if input is bc.
↑



for (b, Z₀ , nothing is defined. This is known as a dead configuration.

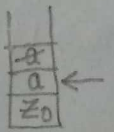
↓

for any string which is not in the language, we come across a dead configuration.

∴ for the dead configuration, the PDA will 'halt' & say that it is not accepted.

for 'bc', it will halt in state q_0 itself.

or aabcc,

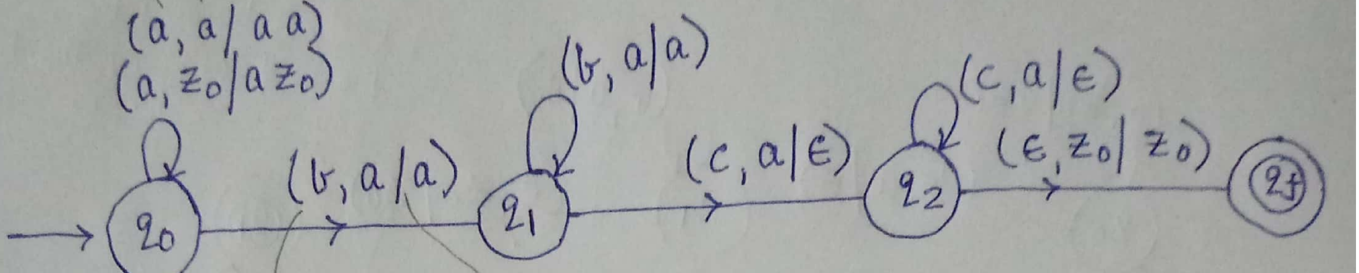


(c, a ...)

halts in state q_1

Eg:- $L = \{a^n b^m c^n \mid n, m \geq 1\}$

* APPROACH → Match a's & c's & leave b's like that.

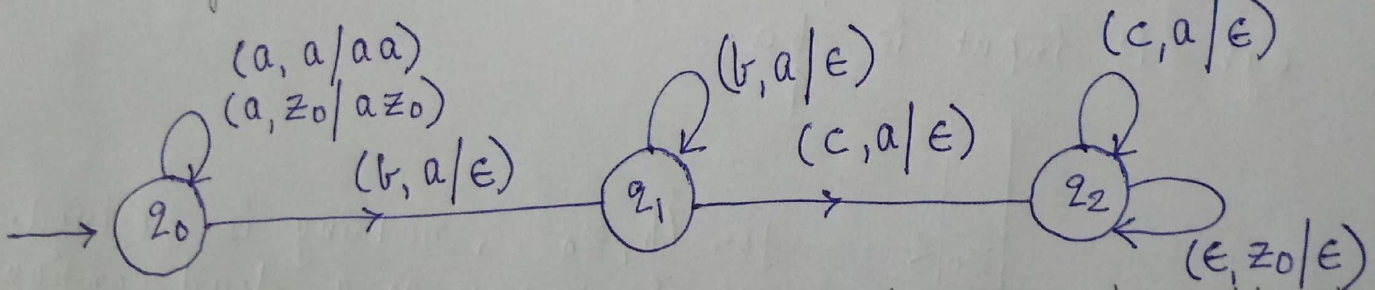


by this time, all a's will be pushed because of previous 2 moves.

∵ we don't want to push or POP for b's as it does not require counting. Hence, leave 'a' as it is.

Eg:- $\{\underbrace{a^{m+n}} b^m c^n \mid m, n \geq 1\} = L$

Count all a's and match them against b's & c's.



(Acceptance by empty stack)

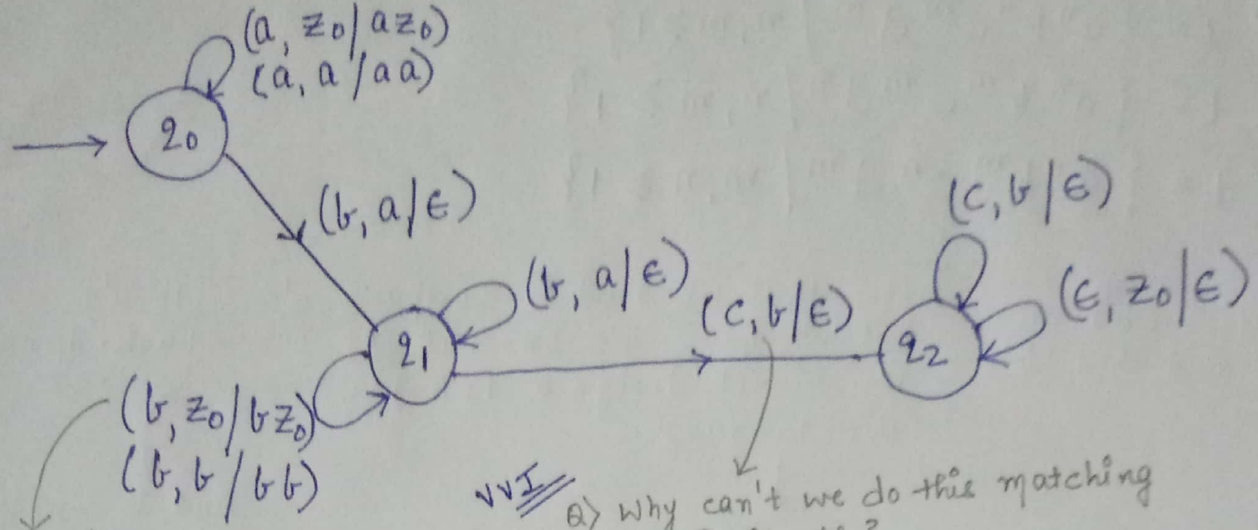
Eg:- $L = \{a^n b^{m+n} c^m \mid n, m \geq 1\}$

$\Rightarrow L = \{\underbrace{a^n b^n}\, \underbrace{b^m c^m} \mid n, m \geq 1\}$

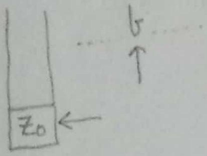
push them on stack

APPROACH →

whenever we see a 'b' we match them against a's & then, we push remaining b's on to the stack. & then, match them against c's.



During this state, all a's have been matched and extra b's are coming in.



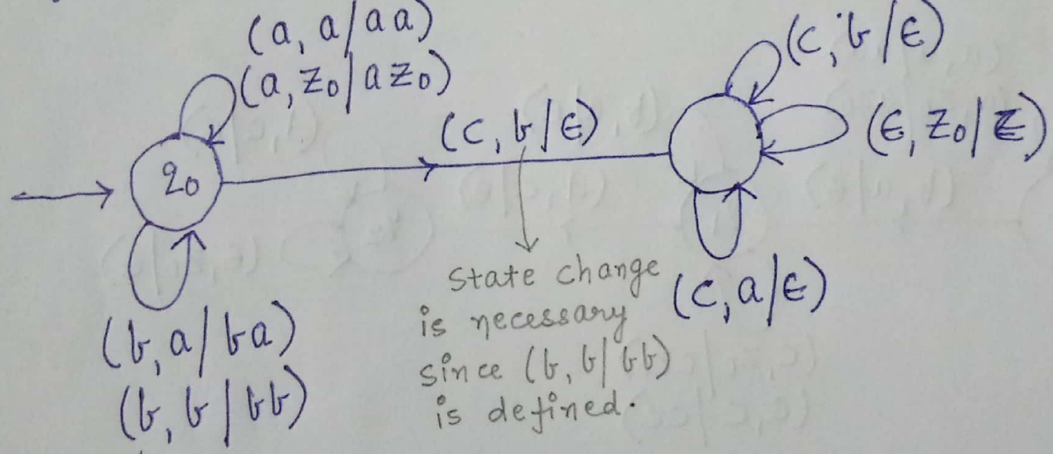
VVI Q) Why can't we do this matching in q_1 itself?

A) Because after 'c' a 'b' will come (due to this $(b, b/bb)$ transition) and it will be pushed.

Since, $(b, a/\epsilon)$ is defined for q_1 it will pop 'a'. But, $(b, b/bb)$ will push 'b' even after seeing a 'c' as an i/p.

Q) $L = \{ a^n b^m c^{m+n} ; n, m \geq 1 \}$

A)

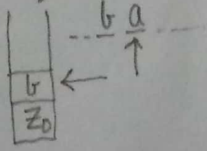


State change is necessary since $(b, b/bb)$ is defined.

VVI * Is there any problem with this?

what if after 'b' an 'a' comes?

A) Doesn't harm, since, we do not have $(a, b/\ldots)$ as our transition on q_0.



∴ We do not accept it & reach dead config. due to absence of the transition $(a, b/\ldots)$.

1. Q> $L = \{a^n b^n c^m d^m / n, m \geq 1\}$

2. Q> $L = \{a^n b^m c^m d^n / n, m \geq 1\}$

*Q> $L = \{\underbrace{a^n b^m}_{} c^n d^m / m, n \geq 1\}$

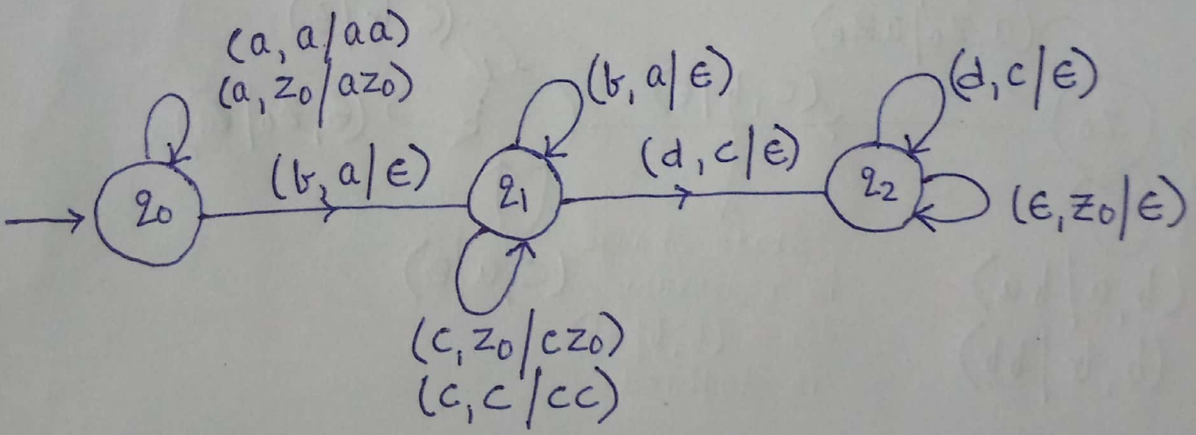
first we PUSH a's and then 'b's

After that we need to compare c's with a's but we have b's on TOP of our stack. Hence, We cannot design a PDA for this language.

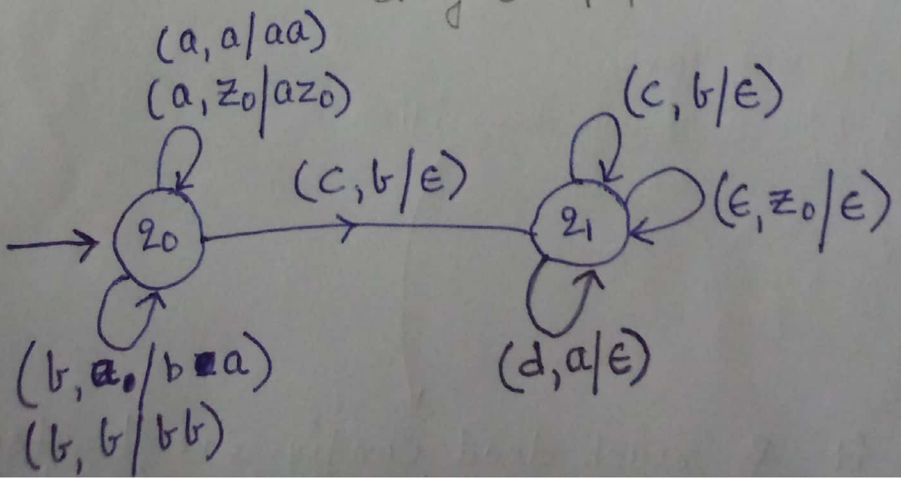
∵ the language is not context free.

And, to compare a's, we need to go to the bottom of the stack but then, in a stack we can only see the TOP.

1. A> APPROACH:→ PUSH a's, then, match them against b's.
PUSH c's, then, match them against d's.



2. B> APPROACH :→ PUSH a's, then, PUSH b's after that for 'c' pop b's and for every 'd' pop a's.

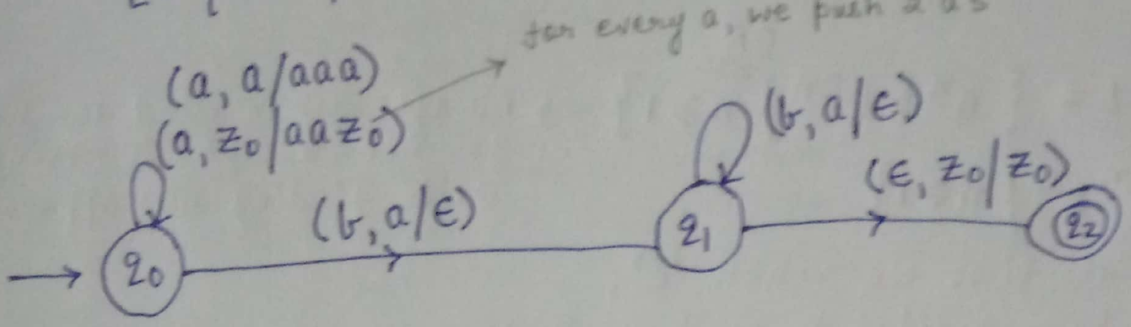


6) $L = \{a^n b^{2n} / n \geq 1\}$

A) *Approach ①:→ for every 'a' we could push 2 a's, and whenever we see a 'b', we match the 'a' off.

Enumerating the language, We have,

$L = \{abb, aabbbb, \ldots\}$

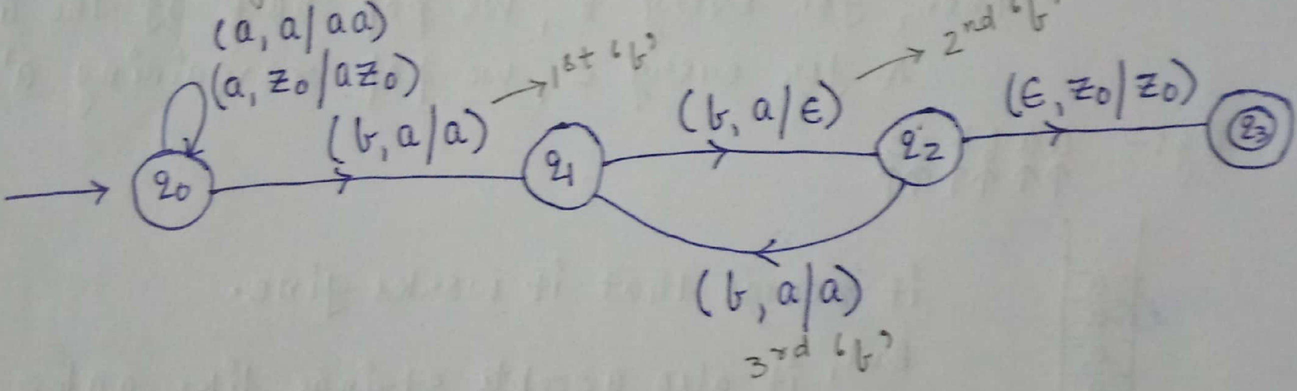


* Approach ② :-

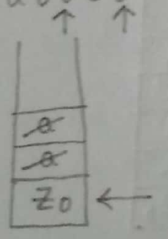
for every 'a', we push a single 'a' and for every 2 b's, we POP an 'a'. But, we cannot see 2 b's at the same time, we will see 1st 'b' & then 2nd 'b',

for 1st b → don't do anything.

for 2nd b → pop off an 'a'.

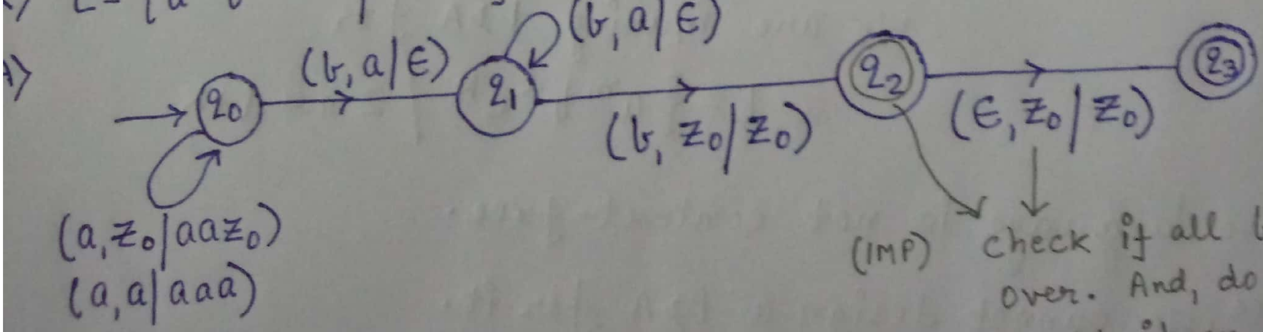


Eg:- aabbbbε

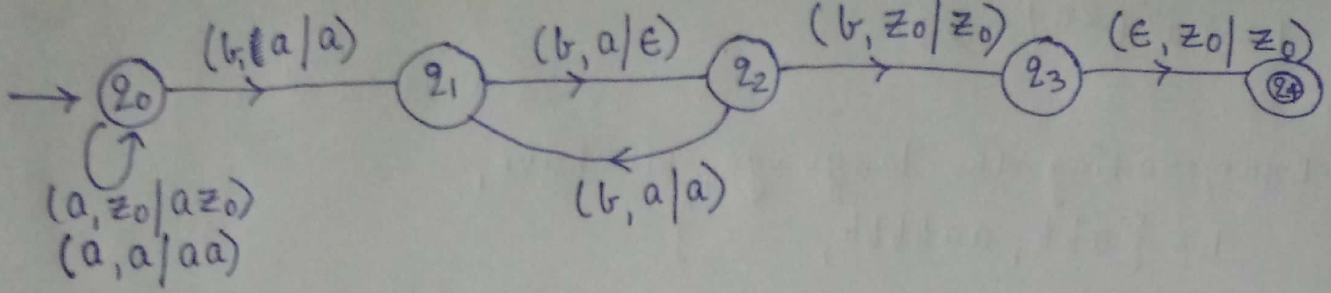


) $L = \{a^n b^{2n+1} | n \geq 1\}$

A)



(IMP) check if all b's are over. And, do not directly accept it on q2.



OR

given → $L = \{a^n b\, b^{2n} / n \geq 1\}$ → Always keep in mind the D.C for non-acceptable strings while finding new approaches

*

Q) $L = \{a^n b^n c^n / n \geq 1\}$

A) Now, by the time we reach 'c', we have nothing to compare in the stack since every 'a' has been popped off.

Hence, APPROACH-1 fails.

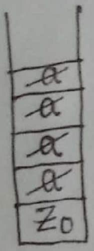
Another approach :-

for every a, we push 2 a's

and, then, for every b, we pop half of the a's.

& for every c, we pop remaining a's.

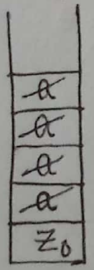
Eg:- aabbcc
↑↑ ↑↑ ↑↑



it seems that it works fine.

But, it also accepts strings like aabccc.
↑↑↑↑↑↑

i.e



In fact, what happens is,

We are making PDA for

$L = \{a^m b^l c^k \mid 2m = l + k$

The above language is not context-free.

↳ since, we cannot design a PDA for it.

Q) $L = \{ w c w^R : w \in (a, b)^+ \}$

A) Eg:- Set of all odd length palindromes

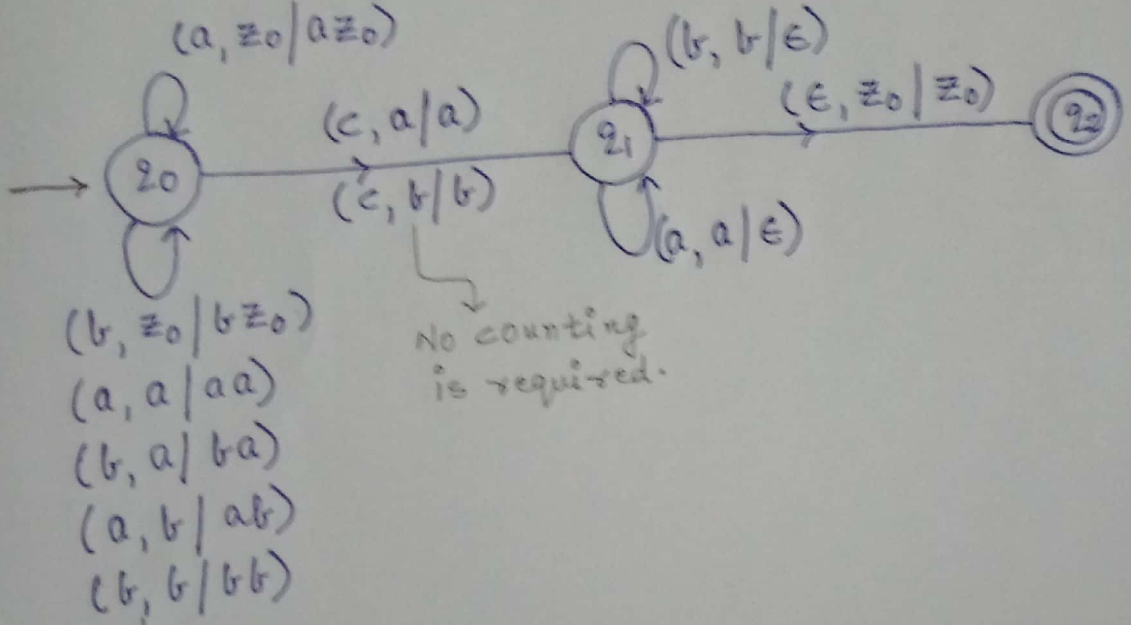
abcba
abbcbba

↓ palindromes whose center is known.

APPROACH:-

Whenever you see 'w', push it on stack & whenever we see w^R, we match it correspondingly. (Our center 'c' tells when w^R will start.)

↳ IMP



↳ No counting is required.

↳ the word will be PUSHED on to the stack.

Q) $L = \{ w w^R : w \in (a+b)^+ \}$

A) Set of all even length palindromes.

↳ center is not known.