Unit-04

Infinite size

Pushdown Automata (PDA) :- Finite Automata + 1 STACK (memory)

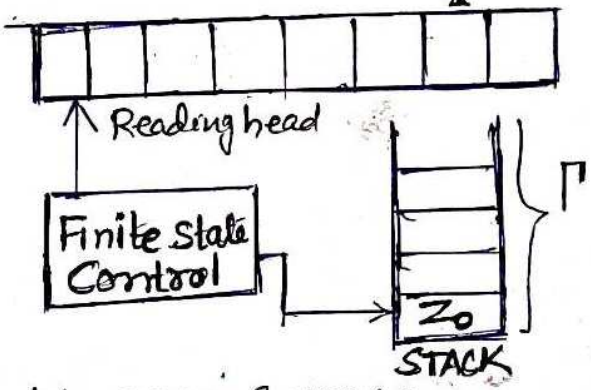
$$PDA = (Q, \Sigma, \delta, q_0, F) \cup (Z_0, \Gamma)$$

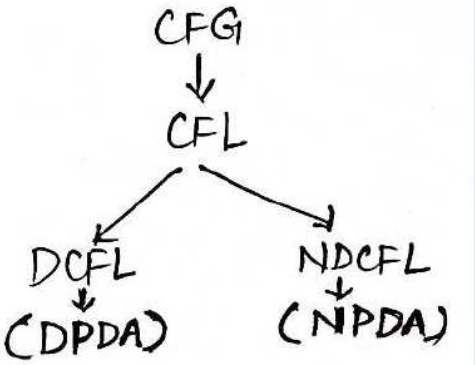
$$PDA = (Q, \Sigma, \delta, \Gamma, q_0, Z_0, F)$$

- Σ : I/p Alphabet
- Γ : set of pushdown symbols
- Z_0 : Initial pushdown symbol, $Z_0 \in \Gamma$
- Q : Collection of 7-tuples
- Set of states
- q_0 : Initial state, $q_0 \in Q$
- F : set of final states, $F \subseteq Q$
- δ : Transition function

$$\delta : Q \times (\Sigma \cup \varepsilon) \times \Gamma \rightarrow Q \times \Gamma^*$$

Model of Pushdown Automata :-





V.Imp.

Power (DPDA) < Power (NPDA)

Difference b/w DPDA & NPDA :-

Deterministic Pushdown Automata (DPDA)	Non Deterministic Pushdown Automata (NPDA)
1- Choice is not Allowed in DPDA	1- Choice Allowed in NPDA
2. Dead Configuration is Allowed in DPDA	2- Dead Configuration is Allowed in NPDA
3. Null moves Allowed Conditionally	3- Null Moves is allowed Unconditionally
4- $\delta : Q \times \Sigma \cup \varepsilon \times \Gamma \rightarrow Q \times \Gamma^*$	4- $\delta : Q \times \Sigma \cup \varepsilon \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$
5- it is less Powerful than NPDA	5- It is more Powerful Than DPDA
6- Language Accepted by DPDA is called DCFL	6- Language Accepted by NPDA is Called NDCFL
7- It is Possible to Convert every DPDA to a Corresponding NPDA	7- It is not Possible to Convert every NPDA to a Corresponding DPDA.

Instantaneous Description (ID):- Instantaneous Description (ID) is an informal notation of how a PDA "computes" a input string and make a Decision that string is accepted or rejected.

A- ID is a triple (q, w, α), where

1- q is the current state

2- w is the remaining input

3- α is the stack contents, top at the Left.

> **NOTE:** if acceptance is defined by empty stack then there is no meaning of final state and it is represented by ϕ

Turnstile Notation:

$\vdash$ sign is called a "turnstile notation" and represents one move

$\vdash^*$ sign represents a sequence of Moves

i.e $(P, b, T) \vdash (q, w, \alpha)$

This implies that while taking a transition from state P to state q, The input symbol b is consumed and the top of the STACK 'T' is replaced by a new string α.

Acceptance by PDA:

1- **Acceptance by final:** A PDA accepts a string when, after reading the entire string, The PDA is in a final state

Let $P = (Q, \Sigma, \delta, \Gamma, q_0, z_0, F)$ be a PDA The language accepted by The final state can be defined as

$$L(PDA) = \{ w \mid (q_0, w, z_0) \vdash^* (P, \varepsilon, X), P \in F, X \in \Gamma^* \}$$

2- **Acceptance by Empty stack:** Here a PDA accepts a string when, after reading the entire string, The PDA has emptied its stack.

$$N(PDA) = \{ w \mid (q_0, w, z_0) \vdash^* (P, \varepsilon, \varepsilon), P \in Q \}$$

Equivalence of Acceptng by final state and Empty STACK:

- if $L = N(P_1)$ for some PDA P_1, Then there is a PDA P_2 such that $L = L(P_2)$. That means the language accepted by empty stack PDA will also be accepted by final state PDA.
- if there is a language $L = L(P_1)$ for some PDA P_1 Then there is a PDA P_2 such that $L = N(P_2)$. That means language accepted by final state PDA is also accepted by empty stack PDA.

Representation of PDA :-

Transition function : $\delta : (Q \times \Sigma \cup \varepsilon \times \Gamma) \rightarrow Q \times \Gamma^*$

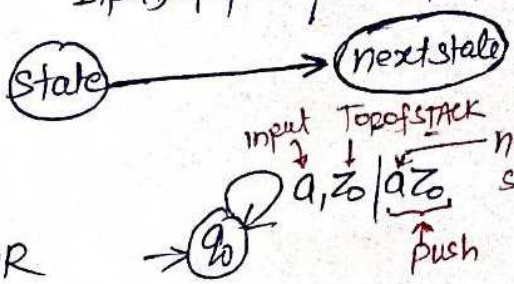
(State, I/p Read, STACK Top, next state, String Push OR POP in The STACK)

δ : (State, I/p symbol, Top of STACK symbol) ⊢ (next state, Top two symbols of STACK)

OR

Input, Top of STACK / new Top of STACK

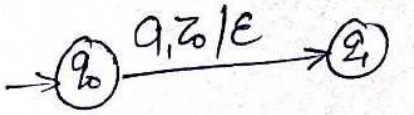
Representation of state Transition :



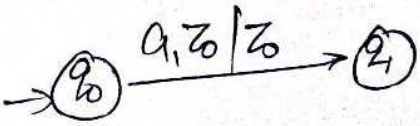
Push :

$\delta(q_0, a, z_0) \vdash (q_0, az_0)$ OR

POP : $\delta(q_0, a, z_0) \vdash (q_1, \varepsilon)$ OR



Read : $\delta(q_0, a, z_0) \vdash (q_1, z_0)$ OR



Language to PDA Designing :

1- $L = \{a^n b^n \mid n \geq 1\}$ Design PDA

Sol^n : $L = \{ab, a^2b^2, a^3b^3, a^4b^4, \ldots\infty\}$

Transition function of PDA : $\delta : Q \times \Sigma \cup \varepsilon \times \Gamma \rightarrow Q \times \Gamma^*$

$\delta(q_0, a, z_0) \vdash (q_0, az_0)$

$\delta(q_0, a, a) \vdash (q_0, aa)$

$\delta[q_0, b, a) \vdash (q_1, \varepsilon)$

$\delta(q_1, b, a) \vdash (q_1, \varepsilon)$

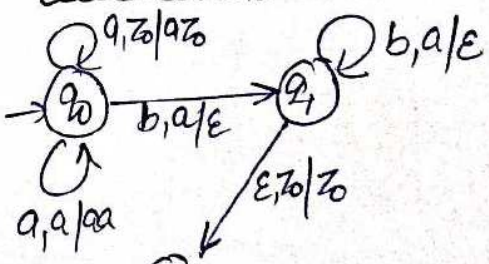
$\delta(q_1, \varepsilon, z_0) \vdash (q_f, z_0)$ By final state Acceptance $q_f \in F$

OR

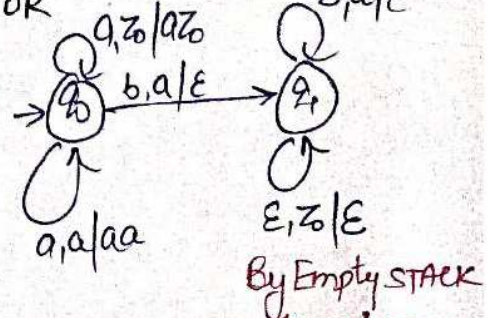
(q_1, ε) By Empty STACK Acceptance

OR

Transition Graph Representation :



OR



Now PDA is a Collection of 7 tuples $= (Q, \Sigma, \delta, \Gamma, q_0, z_0, F)$

$PDA = (\{q_0, q_1, q_f\}, \{a, b\}, \delta, \{a, z_0\}, q_0, z_0, q_f)$

Ans :

2- $L = \{a^n b^{2n} \mid n \geq 1\}$ Design PDA and Accept string aabbbb

Solution:- Given $L = \{a^n b^{2n} \mid n \geq 1$, means

$L = \{ab^2, a^2b^4, a^3b^6, a^4b^8, \ldots\ldots \infty\}$

$\delta(q_0, a, z_0) \vdash (q_0, az_0)$

$\delta(q_0, a, a) \vdash (q_0, aa)$

$\delta(q_0, b, a) \vdash (q_1, a)$

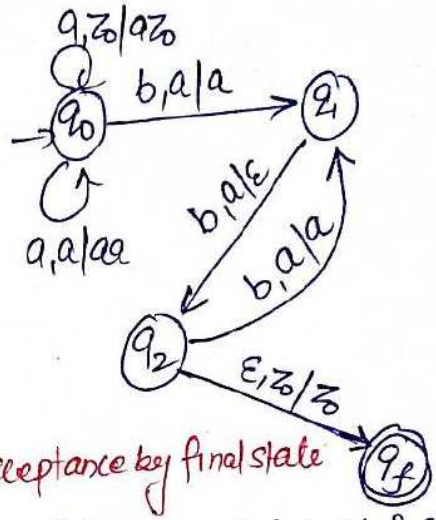
$\delta(q_1, b, a) \vdash (q_2, \varepsilon)$

$\delta(q_2, b, a) \vdash (q_1, a)$

$\delta(q_2, \varepsilon, z_0) \vdash (q_f, z_0), q_f \in F$

OR

Transition Graph:



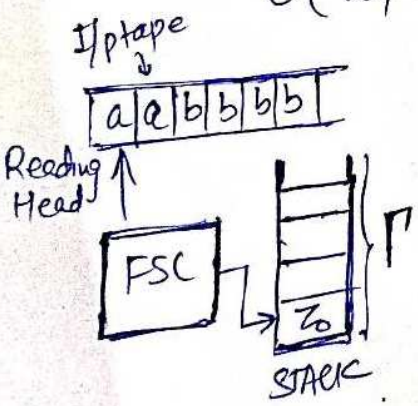
Acceptance by final state

$PDA = (\{q_0, q_1, q_2, q_f\}, \{a, b\}, \delta, \{a, z_0\}, q_0, z_0, q_f)$

Given string $\omega = aabbbb$,

Initial Configuration

$\delta(q_0, aabbbb, z_0) \vdash (q_0, abbbb, az_0)$



Initial Configuration

$\vdash (q_0, bbbb, aaz_0)$

$\vdash (q_1, bbb, aaz_0)$

$\vdash (q_2, bb, az_0)$

$\vdash (q_1, b, az_0)$

$\vdash (q_2, \varepsilon, z_0)$

$\vdash (q_f, \varepsilon, z_0)$

Accepted by final state

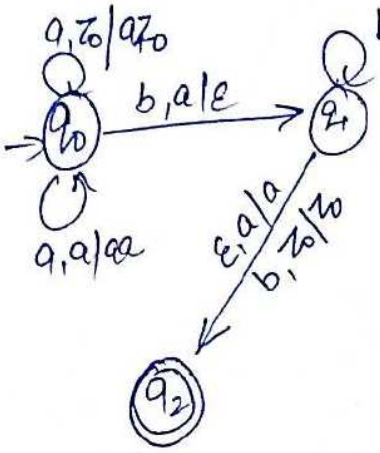
3- $L = \{ a^m b^n \mid m \neq n, m, n \geq 1 \}$ Design PDA

DCFL

Sol^n: Given Language $L = \{ ab^2, a^2b, ab^3, a^3b, a^2b^3, a^3b^2, \dots \}$

DPDA

δ:



Now PDA $= (Q, \Sigma, \delta, \Gamma, q_0, z_0, F)$

PDA $= (\{q_0, q_1, q_f\}, \{a, b\}, \delta, \{a, z_0\}, q_0, z_0, q_f)$

Ans:

4- $L = \{ a^m b^n \mid m = 2n+1, m, n \geq 1 \}$ Design PDA

DCFL

Sol^n: Given $L = \{ a^m b^n \mid m = 2n+1, m, n \geq 1 \}$

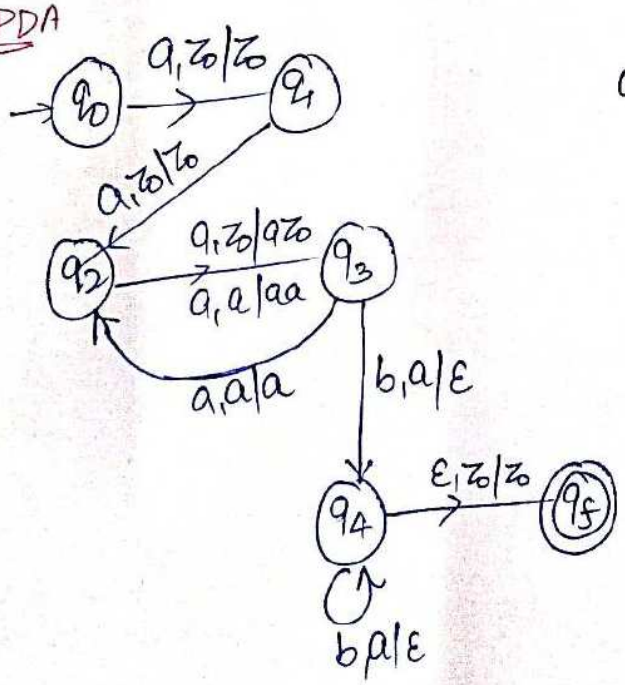
$L = \{ a^{2n+1} b^n \mid n \geq 1 \}$

$L = \{ a \cdot a^{2n} b^n \mid n \geq 1 \}$

V. Imp.

NOTE: Deterministic PDA (DPDA) is just like DFA, which has at most one choice to move for certain input. A PDA $= (Q, \Sigma, \delta, \Gamma, q_0, z_0, F)$ is DPDA if it satisfies condition given as follows for any $q \in Q$, $a \in (\Sigma \cup \varepsilon)$ and $Z \in \Gamma$, $\delta(q, a, Z)$ has at most one choice of Move

DPDA



OR

$\delta(q_0, a, z_0) \vdash (q_1, z_0)$

$\delta(q_1, a, z_0) \vdash (q_2, z_0)$

$\delta(q_2, a, z_0) \vdash (q_3, az_0)$

$\delta(q_3, a, a) \vdash (q_2, a)$

$\delta(q_2, a, a) \vdash (q_3, aa)$

$\delta(q_3, b, a) \vdash (q_4, \varepsilon)$

$\delta(q_4, b, a) \vdash (q_4, \varepsilon)$

$\delta(q_4, \varepsilon, z_0) \vdash (q_f, z_0)$

$q_f \in F$

Ans:

NOTE: According to DPDA Definition Question No. 1 to 4 are DPDA and corresponding language is DCFL

5- $L = \{ a^n b^n \cup a^n b^{2n} \mid n \geq 1 \}$ Design PDA

Sol^n: Given $L = \{ a^n b^n \cup a^n b^{2n} \mid n \geq 1 \}$

$L_{NDCFL} = \{ ab, a^2b^2, \cdots ab^2, a^2b^4, \cdots \}$

NPDA:

$\delta(q_0, a, z_0) \vdash (q_0, az_0)$

$\delta(q_0, a, a) \vdash (q_0, aa)$

$\delta(q_0, b, a) \vdash (q_1, \varepsilon), (q_2, a)$

$\delta(q_1, b, a) \vdash (q_1, \varepsilon)$

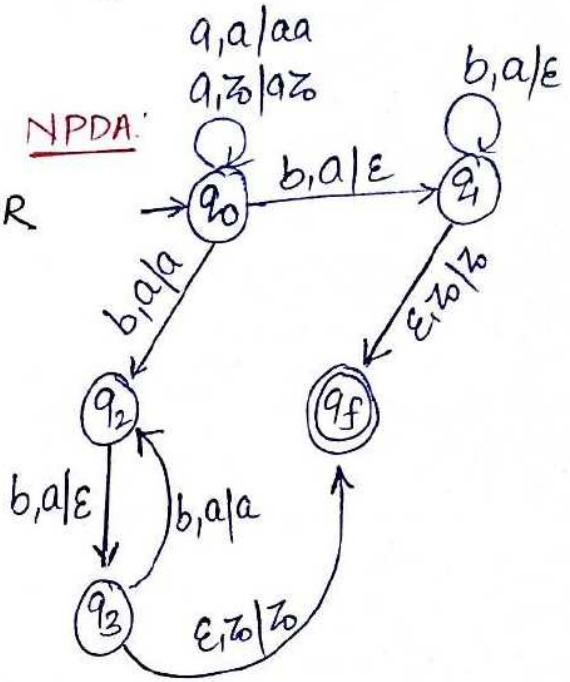
$\delta(q_1, \varepsilon, z_0) \vdash (q_f, z_0)$

$\delta(q_2, b, a) \vdash (q_3, \varepsilon)$

$\delta(q_3, b, a) \vdash (q_2, a)$

$\delta(q_3, \varepsilon, z_0) \vdash (q_f, z_0)$

OR



NPDA: $\delta : Q \times \Sigma \cup \varepsilon \times \Gamma \to Q^{Q \times \Gamma^*}$

NOTE:- like NFA, nondeterminishc PDA (NPDA) has finik Number of choices for its inputs. As we have discused in the mathematical description that transition function δ which maps from $Q \times (\Sigma \cup \varepsilon) \times \Gamma$ to (finite subset of) $Q \times \Gamma^*$

A ~~non~~ NPDA accepts an input if a sequence of choice leads to some final state or cause PDA to empty ~~stack~~ its STACK, since sometimes it has more than one choice to move further on a particular input ; it means PDA guesses the right choice always, otherwise it will fail and will be in hang state

6- Consider Previous Example (Question No-5) Check whether string $w = aabb$ is accepted or not?

Sol:- Question No 5, $L = \{a^n b^n \cup a^n b^{2n} \mid n \geq 1\}$ PDA Solve in Previous Pg

Given

$\delta(q_0, a, z_0) \vdash (q_0, az_0)$

$\delta(q_0, a, a) \vdash (q_0, aa)$

$\delta(q_0, b, a) \vdash (q_1, \varepsilon), (q_2, a)$

$\delta(q_1, b, a) \vdash (q_1, \varepsilon)$

$\delta(q_1, \varepsilon, z_0) \vdash (q_f, z_0)$

$\delta(q_2, b, a) \vdash (q_3, \varepsilon)$

$\delta(q_3, b, a) \vdash (q_2, a)$

$\delta(q_3, \varepsilon, z_0) \vdash (q_f, z_0)$

inihal Configuration $\delta(q_0, aabb, z_0)$

$(q_0, aabbz_0) \rightarrow (q_0, abb, az_0)$

$\downarrow$

(q_0, bb, aaz_0)

Non Deterministic

(q_1, b, az_0) → (q_1, ε, z_0) → (q_f, ε, z_0) Accepted, $q_f \in F$

(q_2, b, aaz_0) → (q_3, ε, az_0) → STOP, Rejected because Neither q_3 final state Nor STACK empty:

Yes string $w = aabb$ Accepted

7- $L = \{a^n b^n c^m \mid m, n \geq 1\}$ Design PDA

DCFL

Sol^n: Given language $L = \{abc, a^2b^2c, abc^2, a^2b^2c^2, abc^3, a^2b^2c^3, \ldots\}$

$\delta(q_0, a, z_0) \vdash (q_0, az_0)$

$\delta(q_0, a, a) \vdash (q_0, aa)$

$\delta(q_0, b, a) \vdash (q_1, \varepsilon)$

$\delta(q_1, b, a) \vdash (q_1, \varepsilon)$

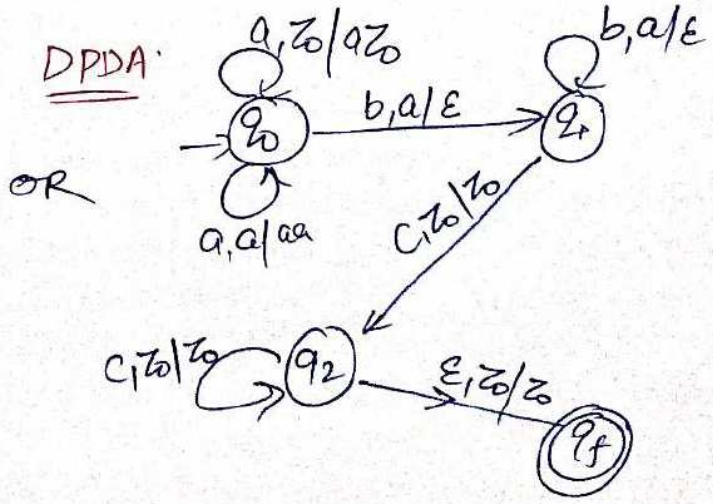
$\delta(q_1, c, z_0) \vdash (q_2, z_0)$

$\delta(q_2, c, z_0) \vdash (q_2, z_0)$

$\delta(q_2, \varepsilon, z_0) \vdash (q_f, z_0)$

DPDA

OR



8- $L = \underset{DCFL}{\{a^n b^m c^n \mid m,n \geq 1\}}$ Design PDA

Sol^n: Given $L = \{abc, ab^2c, a^2bc^2, a^2b^2c^2, ab^3c, a^2b^3c^2, \cdots\}$

DPDA:

$\delta(q_0, a, z_0) \vdash (q_0, az_0)$

$\delta(q_0, a, a) \vdash (q_0, aa)$

$\delta(q_0, b, a) \vdash (q_1, a)$

$\delta(q_1, b, a) \vdash (q_1, a)$

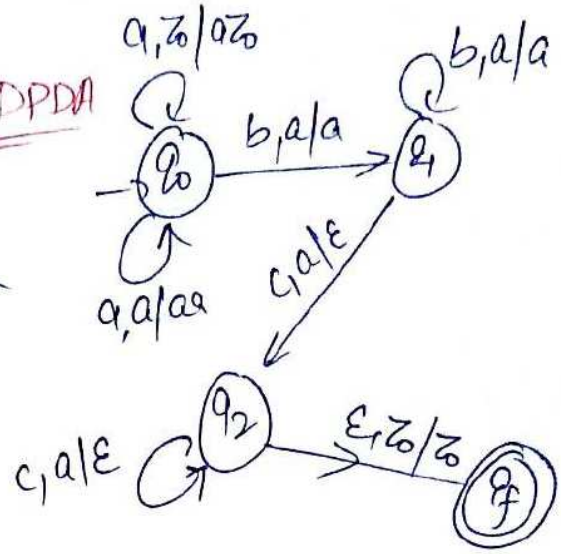
$\delta(q_1, c, a) \vdash (q_2, \varepsilon)$

$\delta(q_2, c, a) \vdash (q_2, \varepsilon)$

$\delta(q_2, \varepsilon, z_0) \vdash (q_f, z_0)$

OR

DPDA



DPDA $= (\{q_0, q_1, q_2, q_f\}, \{a, b, c\}, \delta, \{a, z_0\}, q_0, z_0, q_f)$

9- $L = \underset{DCFL}{\{a^m b^n c^n \mid m,n \geq 1\}}$ Design PDA.

DPDA

$\delta(q_0, a, z_0) \vdash (q_0, z_0)$

$\delta(q_0, b, z_0) \vdash (q_1, bz_0)$

$\delta(q_1, b, b) \vdash (q_1, bb)$

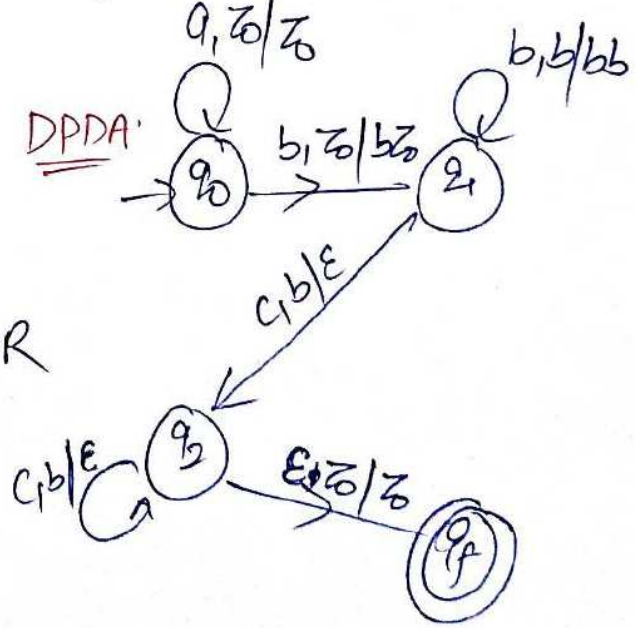
$\delta(q_1, c, b) \vdash (q_2, \varepsilon)$

$\delta(q_2, c, b) \vdash (q_2, \varepsilon)$

$\delta(q_2, \varepsilon, z_0) \vdash (q_f, z_0)$

OR

DPDA:



DPDA $= (\{q_0, q_1, q_2, q_f\}, \{a, b, c\}, \delta, \{z_0, b\}, q_0, z_0, q_f)$

Ans.

10- $L = \underset{DCFL}{\{} a^m b^n c^{m+n} \mid m,n \geq 1\}$ Design PDA.

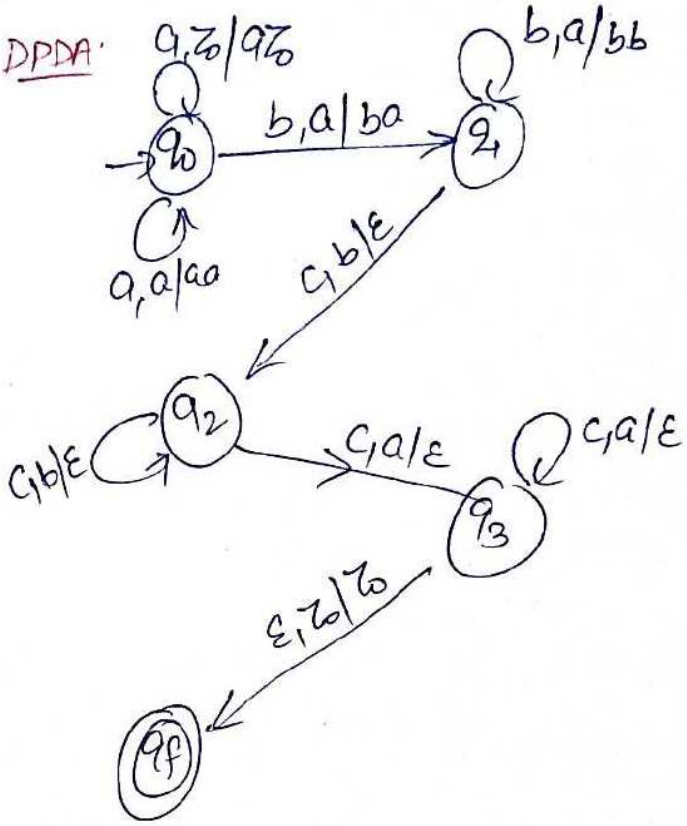
Sol^n: Given $L = \underset{DCFL}{\{} a^m b^n c^m . c^m \mid m,n \geq 1$

DPDA

$\delta(q_0, a, z_0) \vdash (q_0, az_0)$
$\delta(q_0, a, a) \vdash (q_0, aa)$
$\delta(q_0, b, a) \vdash (q_1, ba)$
$\delta(q_1, b, a) \vdash (q_1, bb)$
$\delta(q_1, c, b) \vdash (q_2, \varepsilon)$
$\delta(q_2, c, b) \vdash (q_2, \varepsilon)$
$\delta(q_2, c, a) \vdash (q_3, \varepsilon)$
$\delta(q_3, c, a) \vdash (q_3, \varepsilon)$
$\delta(q_3, \varepsilon, z_0) \vdash (q_f, z_0)$

OR

DPDA.



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

Sol^n: In Given Language No. of a = No. of C and No. of b = No of d
So you can not proof No. of a = No of C and No of b = No of d
by single STACK (PDA) There is no PDA for this language because
This language is not Context free language.

It is easy to recognise this language of you use a Queue (FIFO)
instead of STACK, use Two STACK, or use a Turing machine (all equivalent)

Turing m/c = FA + 2 STACK = FA + 1 Queue

Solution of this Question will discuss in 2 STACK PDA Chapter

12- $L = \{a^m b^n c^k \mid m=n \text{ OR } n=k, m,n,k \geq 1\}$, Design PDA.

NDCFL

OR

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

NDCFL — L_1 ∪ L_2

Solution: Given $L = \{a^m b^m c^k \cup a^m b^n c^n \mid m,n,k \geq 1\}$

NPDA:

$\delta(q_0, a, z_0) \vdash (q_0, az_0), (q_3, z_0)$

$\delta(q_0, a, a) \vdash (q_0, aa)$

$\delta(q_0, b, a) \vdash (q_1, \varepsilon)$

$\delta(q_1, b, a) \vdash (q_1, \varepsilon)$

$\delta(q_1, c, z_0) \vdash (q_2, z_0)$

$\delta(q_2, c, z_0) \vdash (q_2, z_0)$

$\delta(q_2, \varepsilon, z_0) \vdash (q_f, z_0)$

$\delta(q_3, a, z_0) \vdash (q_3, z_0)$

$\delta(q_3, b, z_0) \vdash (q_4, bz_0)$

$\delta(q_4, b, b) \vdash (q_4, bb)$

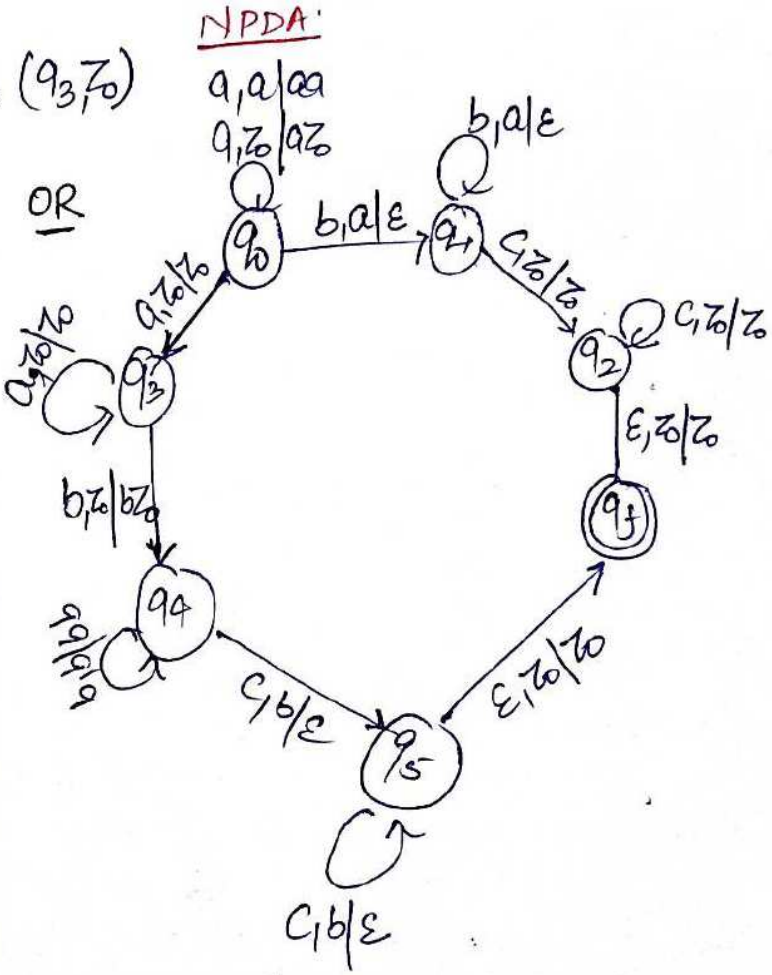
$\delta(q_4, c, b) \vdash (q_5, \varepsilon)$

$\delta(q_5, c, b) \vdash (q_5, \varepsilon)$

$\delta(q_5, \varepsilon, z_0) \vdash (q_f, z_0)$

NPDA:

OR



PDA is represented by 7-tuples $(Q, \Sigma, \delta, \Gamma, q_0, z_0, F)$

$PDA = (\{q_0, q_1, q_2, q_3, q_4, q_5, q_f\}, \{a, b, c\}, \delta, \{a, b, z_0\}, q_0, z_0, q_f)$

Ans

13- $L = \{ a^m b^n c^k \mid m=n \text{ and } n=k, m,n,k \geq 1 \}$

OR

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

Solution:- you can not proof No. of a = No. of b = No. of c by single STACK (PDA)
There is no PDA for this language because this language is not Context free Language.
It is easy to recognize this language if you use a Queue instead of STACK, use Two STACK, or use a Turing machine (all equivalent)

Turing machine = FA + 2 STACK = FA + 1 Queue

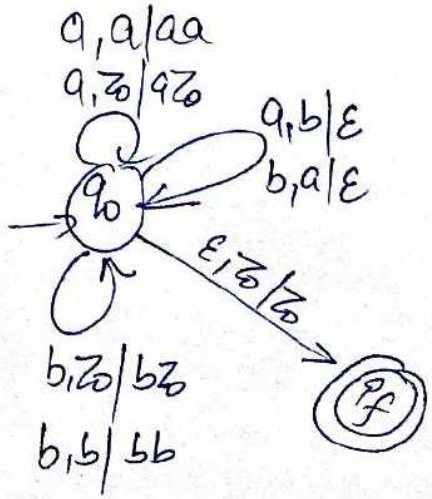
Solution of this Question will discussed in 2STACK-PDA Chapter

14- $L = \{ w \mid n_a(w) = n_b(w), w \in (a,b)^* \}$, Design PDA

Sol: $L = \{ \varepsilon, ab, abba, abab, aabb, baab, baba, \text{----} \}$

$\delta(q_0, a, z_0) \vdash (q_0, az_0)$
$\delta(q_0, b, z_0) \vdash (q_0, bz_0)$
$\delta(q_0, a, a) \vdash (q_0, aa)$
$\delta(q_0, b, b) \vdash (q_0, bb)$
$\delta(q_0, a, b) \vdash (q_0, \varepsilon)$
$\delta(q_0, b, a) \vdash (q_0, \varepsilon)$
$\delta(q_0, \varepsilon, z_0) \vdash (q_f, z_0)$

OR

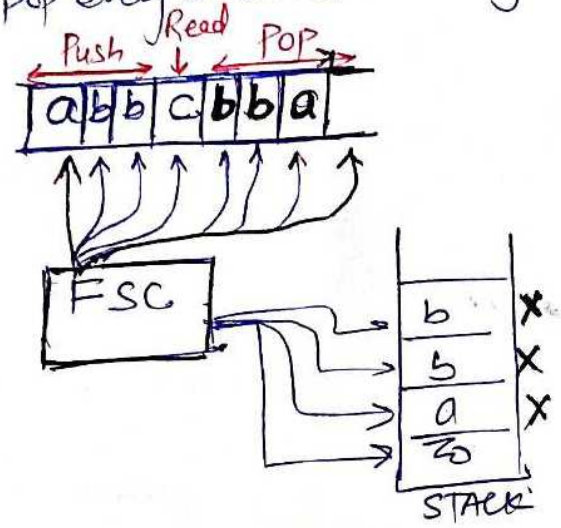


15- $L = \{wcw^R, w \in (a,b)^*\}$ Design PDA.

Sol^n: Given $L = \{c, aca, bcb, aacaa, bbcbb, abcba, bacab, \cdots\}$ is for contain Odd Length palindrom Strings

Some string will come followed by one 'c', followed by reverse of the string before 'c', so we get to know that 'c' will work as an alarm to starting poping STACK.

So we will pop every 'a' with 'a' and every 'b' with 'b'



$\delta(q_0, a, z_0) \vdash (q_0, az_0)$

$\delta(q_0, b, z_0) \vdash (q_0, bz_0)$

$\delta(q_0, a, a) \vdash (q_0, aa)$

$\delta(q_0, b, b) \vdash (q_0, bb)$

$\delta(q_0, a, b) \vdash (q_0, ab)$

$\delta(q_0, b, a) \vdash (q_0, ba)$

$\delta(q_0, c, a) \vdash (q_1, a)$

$\delta(q_0, c, b) \vdash (q_1, b)$

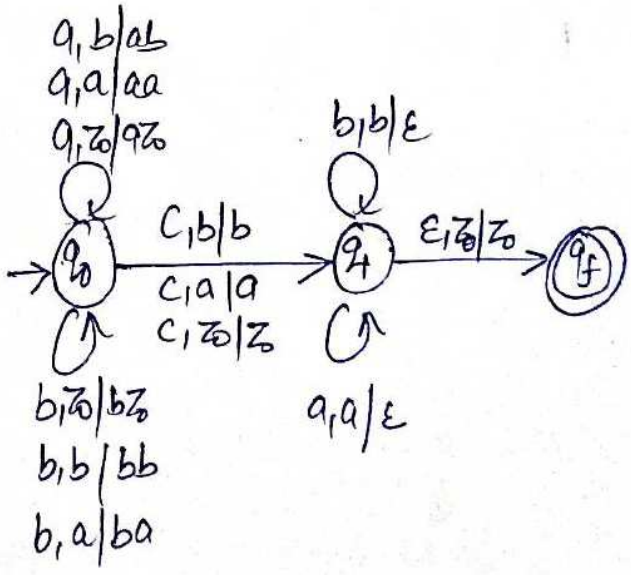
$\delta(q_1, a, a) \vdash (q_1, \varepsilon)$

$\delta(q_1, b, b) \vdash (q_1, \varepsilon)$

$\delta(q_1, \varepsilon, z_0) \vdash (q_f, z_0), q_f \in F$

$\delta(q_0, \varepsilon, z_0) \vdash (q_1, z_0$

OR



16- $L = \{ w.w^R, w \in (a,b)^* \}$ Design PDA

NDCFL

Sol^n: Given $L = \{ w.w^R, w \in (a,b)^* \}$

$L = \{ \varepsilon, aa, bb, abba, aaaa, bbbb, abbbba \cdots \}$ Contain Even Length Palindrom strings.

In this Question There is no C that marked the Centre of the String Therefore, the machine must "guess" when it has reached The middle of the input string and change from State q_0 to q_1 in a Nonterministic fashion.

NPDA:

$\delta(q_0, a, z_0) \vdash (q_0, az_0)$

$\delta(q_0, b, z_0) \vdash (q_0, bz_0)$

$\delta(q_0, a, a) \vdash (q_0, aa), (q_1, \varepsilon)$ /* Non deterministic $(q_0, a, a) \rightarrow (q_0, aa)$, (q_1, ε)

$\delta(q_0, b, b) \vdash (q_0, bb), (q_1, \varepsilon)$ /* Non deterministic $(q_0, b, b) \rightarrow (q_0, bb)$, (q_1, ε)

$\delta(q_0, a, b) \vdash (q_0, ab)$

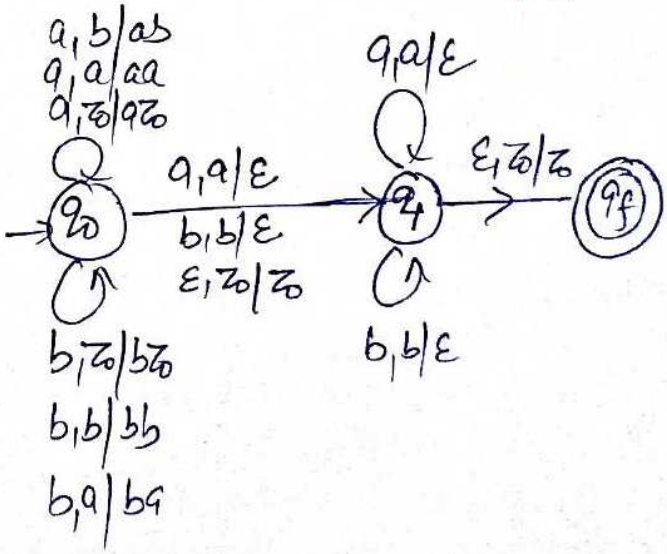
$\delta(q_0, b, a) \vdash (q_0, ba)$

$\delta(q_1, a, a) \vdash (q_1, \varepsilon)$

$\delta(q_1, b, b) \vdash (q_1, \varepsilon)$

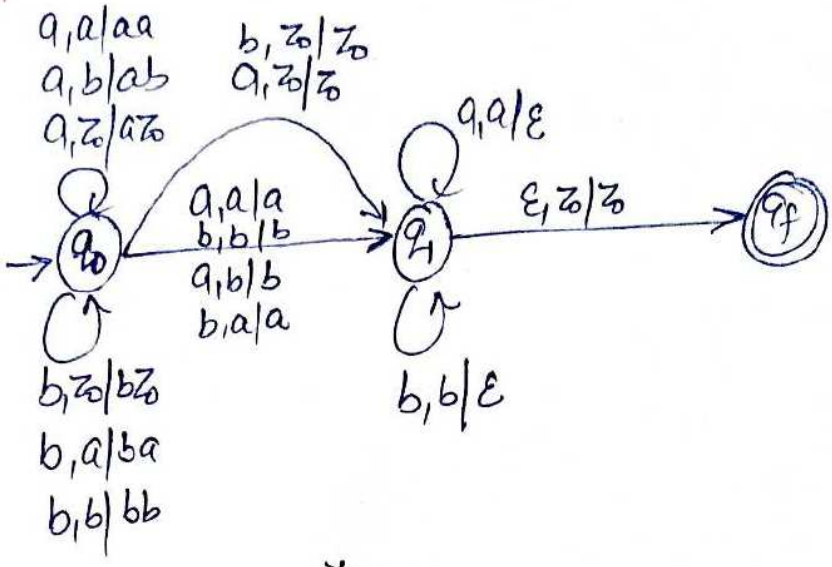
$\delta(q_1, \varepsilon, z_0) \vdash (q_f, z_0)$

$\delta(q_0, \varepsilon, z_0) \vdash (q_1, z_0)$



17- $L = \underset{NDCFL}{\{ w(a+b)w^R,\ w \in (a,b)^* \}}$ Design PDA.

Sol^n :- NPDA



18- $L = \{ w.w \mid w \in (a,b)^* \}$

Sol^n :- W.W Can not match by Single stack (PDA) for W.W Matching Need ~~two~~ STACK or Single Queue OR Matched by Turing machine Model

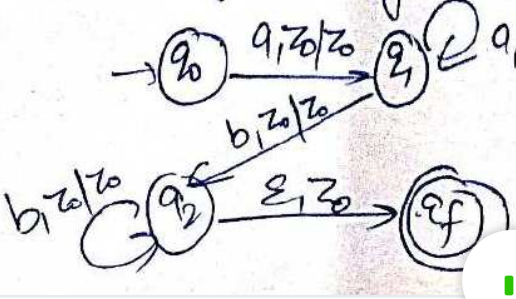
PDA Can not Design for $L = \{ w.w, w \in (a,b)^* \}$ because This is not Context free Language. we will Design Turing machine for This Question in Turing machine chapter

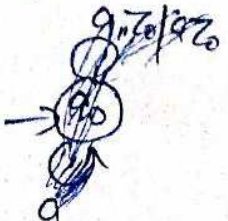
19- $L = \{ w_1 . w_2 \mid w_1, w_2 \in (a,b)^* \}$ Design PDA

Sol^n: $L = \{ w_1 . w_2 \mid w_1, w_2 \in (a,b)^* \} = (a+b)^*(a+b)^*$ is a Regular Language so you can ~~Does~~ ~~does~~ Design PDA ~~for~~ Every Regular.

20- $L = \{ a^m b^n \mid m, n \geq 1 \}$ Design PDA

Sol^n: This regular Language so you can Design PDA for Every Regular





operate STACK only Read operation

Context free Grammar to PDA:

CFG = (V, T, P, S)

↓

PDA = (Q, Σ, δ, Γ, q0, Z0, F)

Algo: PDA = ({q}, T, δ, {V ∪ T}, q, S, ϕ) By Empty STACK Acceptance

δ is defined by the following rules

PUSH: R_1: if $A \to \alpha$ is in P Then $\delta(q, \varepsilon, A) \vdash (q, \alpha)$

POP: R_2: $\forall a \in \Sigma$, $\delta(q, a, a) \vdash (q, \varepsilon)$

Ex:- Construct a PDA equivalent to the following CFG: $S \to OBB, B \to OS | 1S | 0$

Test whether $0 1 0^4$ is in N(PDA)

Soln: Given CFG:

$S \to OBB$

$B \to OS$

$B \to 1S$

$B \to 0$

PDA = ({q}, T, δ, {V ∪ T}, q, S, ϕ)

PDA = ({q}, {0,1}, δ, {S,B,0,1}, q, S, ϕ)

δ is defined by the following rules

R_1: if $A \to \alpha$ is in P Then $\delta(q, \varepsilon, A) \vdash (q, \alpha)$

$\delta(q, \varepsilon, S) \vdash (q, OBB)$

$\delta(q, \varepsilon, B) \vdash (q, 0S)$

$\delta(q, \varepsilon, B) \vdash (q, 1S)$

$\delta(q, \varepsilon, B) \vdash (q, 0)$

R_2: if $\forall a \in \Sigma, \delta(q, a, a) \vdash (q, \varepsilon)$

$\delta(q, 0, 0) \vdash (q, \varepsilon)$

$\delta(q, 1, 1) \vdash (q, \varepsilon)$

Testing $0 1 0^4$ is in N(PDA)

$\delta(q, 010000, S) \vdash (q, 010^4, OBB)$

$\vdash (q, 10^4, BB)$

/* POP */ $\vdash (q, 10^4, 1SB)$

$\vdash (q, 0^4, SB)$

$\vdash (q, 0^4, OBBB)$

/* pop */ $\vdash (q, 0^3, BBB)$

/* pop */ $\vdash (q, 0^3, OBB)$

$\vdash (q, 0^2, BB)$

/* pop */ $\vdash (q, 0^2, 0B)$

$\vdash (q, 0, B)$

/* pop */ $\vdash (q, 0, 0)$

Accepted By Empty STACK $\vdash (q, \varepsilon, \varepsilon)$ Accepted

Ex:- Design PDA for the following CFG:

$S \to \varepsilon$
$S \to SS$
$S \to (S)$

Soln:- $PDA = (Q, \Sigma, \delta, \Gamma, q_0, Z_0, F)$

$PDA = (\{q\}, \{(,)\}, \{S, (,)\}, q, S, \phi)$

NPDA:

$\delta: (q, \varepsilon, S) \vdash (q, (S))$
$\delta(q, \varepsilon, S) \vdash (q, SS)$
$\delta(q, \varepsilon, S) \vdash (q, \varepsilon)$
$\delta(q, (, () \vdash (q, \varepsilon)$
$\delta(q,),)) \vdash (q, \varepsilon)$

Ex:- $S \to aABB \mid aAA$, Design PDA
$A \to aBB \mid a$
$B \to bBB \mid A$

Soln:-
Push: R_1:
$\delta(q, \varepsilon, S) \vdash (q, aABB)$
$\delta(q, \varepsilon, S) \vdash (q, aAA)$
$\delta(q, \varepsilon, A) \vdash (q, aBB)$
$\delta(q, \varepsilon, A) \vdash (q, a)$
$\delta(q, \varepsilon, B) \vdash (q, bBB)$
$\delta(q, \varepsilon, B) \vdash (q, A)$

Pop: R_2:
$\delta(q, a, a) \vdash (q, \varepsilon)$
$\delta(q, b, b) \vdash (q, \varepsilon)$

$PDA = (\{q\}, \{a, b\}, \delta, \{a, b, S, A, B\}, q, S, \phi)$

Ans

Ex:- $S \to 0S1 \mid A$, Design PDA
$A \to 1A0 \mid S \mid \varepsilon$

Soln:- NPDA:

$\delta(q, \varepsilon, S) \vdash (q, 0S1)$
$\delta(q, \varepsilon, S) \vdash (q, A)$
$\delta(q, \varepsilon, A) \vdash (q, 1A0)$
$\delta(q, \varepsilon, A) \vdash (q, S)$
$\delta(q, \varepsilon, A) \vdash (q, \varepsilon)$
$\delta(q, 0, 0) \vdash (q, \varepsilon)$
$\delta(q, 1, 1) \vdash (q, \varepsilon)$, Ans:

Ex:- $S \to aSa \mid bSb \mid c$, Design PDA

Soln:-
$\delta(q, \varepsilon, S) \vdash (q, aSa)$
$\delta(q, \varepsilon, S) \vdash (q, bSb)$
$\delta(q, \varepsilon, S) \vdash (q, c)$
$\delta(q, a, a) \vdash (q, \varepsilon)$
$\delta(q, b, b) \vdash (q, \varepsilon)$
$\delta(q, c, c) \vdash (q, \varepsilon)$ Ans

PDA to CFG:- $PDA = (Q, \Sigma, \delta, \Gamma, q_0, z_0, F)$

$\downarrow$

$G = (V, T, P, S)$

Rules:

1- $V = [qzq']$, $q, q' \in Q$, $z \in \Gamma$

2- $T = \Sigma$

3- S production are given by $S \to [q_0 z_0 q]$, $\forall q \in Q$

4- P:

(i) **pop:** $\delta(q, a, z) \vdash (q_\alpha, \varepsilon)$ induced the Production

$[q, z, q_\alpha] \to a$

(ii) **Push & Read:**

$\delta(q, a, z) \vdash (q_1, z_1 z_2 \cdots z_m)$ induced many Production of the form

$[q z q'] \to a [q_1 z_1 q_2][q_2 z_2 q_3] \cdots [q_m z_m q']$, $q', q_2, q_3, q_m \in Q$

Ex:- Construct a CFG G which accept N(A), where

$A = (\{q_0, q_1\}, \{a, b\}, \delta, \{z_0, z\}, q_0, z_0, \phi)$, δ is given by

1- $\delta(q_0, b, z_0) \vdash (q_0, z z_0)$: Push

2- $\delta(q_0, \varepsilon, z_0) \vdash (q_0, \varepsilon)$: Pop

3- $\delta(q_0, b, z) \vdash (q_0, zz)$: Push

4- $\delta(q_0, a, z) \vdash (q_1, z)$: Read

5- $\delta(q_1, b, z) \vdash (q_1, \varepsilon)$: Pop

6- $\delta(q_1, a, z_0) \vdash (q_0, z_0)$: Read

Sol^n: $V = [qzq']$, $q, q' \in Q$, $z \in \Gamma$

$V = \{[q_0 z_0 q_0], [q_0 z_0 q_1], [q_0 z q_0], [q_0 z q_1], [q_1 z_0 q_0], [q_1 z_0 q_1], [q_1 z q_0], [q_1 z q_1]\}$

$S \to [q_0 z_0 q]$, $\forall q \in Q$

$S \to [q_0 z_0 q_0] / [q_0 z_0 q_1]$

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

1- $\delta(q_0, b, z_0) \vdash (q_0, z z_0)$: push

$[q_0 z_0 q_0] \to b [q_0 z q_0][q_0 z_0 q_0]$

$\to b [q_0 z q_1][q_1 z_0 q_0]$

$[q_0 z_0 q_1] \to b [q_0 z q_0][q_0 z_0 q_1]$

$\to b [q_0 z q_1][q_1 z_0 q_1]$

2- $\delta(q_0, \varepsilon, z_0) \vdash (q_0, \varepsilon)$: pop

$[q_0 z_0 q_0] \to \varepsilon$

3- $\delta(q_0, b, z) \vdash (q_0, zz)$: push

$[q_0 z q_0] \to b [q_0 z q_0][q_0 z q_0]$

$\to b [q_0 z q_1][q_1 z q_0]$

$[q_0 z q_1] \to b [q_0 z q_0][q_0 z q_1]$

$\to b [q_0 z q_1][q_1 z q_1]$

4- $\delta(q_0, a, z) \vdash (q_1, z)$: Read

$[q_0 z q_0] \to a [q_1 z q_0]$

$[q_0 z q_1] \to a [q_1 z q_1]$

5- $\delta(q_1, b, Z) \vdash (q_1, \varepsilon)$: Pop

$[q_1 Z q_1] \rightarrow b$

6- $\delta(q_1, q_1, z_0) \vdash (q_0, z_0)$: Read

$[q_1 z_0 q_0] \rightarrow a[q_0 z_0 q_0]$

$[q_1 z_0 q_1] \rightarrow a[q_0 z_0 q_1]$

Now Grammar $G = (V, T, P, S)$, where

$V = \{[q_0 z_0 q_0], [q_0 z_0 q_1], [q_0 Z q_0], [q_0 Z q_1], [q_1 z_0 q_0], [q_1 z_0 q_1], [q_1 Z q_0], [q_1 Z q_1]\}$

$T = \{a, b\}$

$S \rightarrow [q_0 z_0 q_0] \,/\, [q_0 z_0 q_1]$

P: $[q_0 z_0 q_0] \rightarrow b[q_0 Z q_0][q_0 z_0 q_0]$

$[q_0 z_0 q_0] \rightarrow b[q_0 Z q_1][q_1 z_0 q_0]$

$[q_0 z_0 q_1] \rightarrow b[q_0 Z q_0][q_0 z_0 q_1]$

$[q_0 z_0 q_1] \rightarrow b[q_0 Z q_1][q_1 z_0 q_1]$

$[q_0 z_0 q_0] \rightarrow \varepsilon$

$[q_0 Z q_0] \rightarrow a[q_1 Z q_0]$

$[q_0 Z q_1] \rightarrow a[q_1 Z q_1]$

$[q_0 Z q_0] \rightarrow b[q_0 Z q_0][q_0 Z q_0] \,/\, [q_0 Z q_1][q_1 Z q_0]$

$[q_0 Z q_1] \rightarrow b[q_0 Z q_0][q_0 Z q_1] \,/\, b[q_0 Z q_1][q_1 Z q_1]$

$[q_1 Z q_1] \rightarrow b$

$[q_1 z_0 q_0] \rightarrow a[q_0 z_0 q_0]$

$[q_1 z_0 q_1] \rightarrow a[q_0 z_0 q_1]$

2 STACK-PDA : Equivalent of Turing machine

$$P(\text{2-STACK PDA}) = P(\text{Turing Machine})$$

The Strength of Pushdown automata is increased by adding extra STACKs. A PDA with Two STACKs has the same Computation Power as for a Turing machine

$$\text{2 STACK PDA} = (Q, \Sigma, \delta, \Gamma_1, q_0, Z_0, Z_1, F, \Gamma_2)$$

$$\delta : Q \times (\Sigma \cup \varepsilon) \times \Gamma_1 \times \Gamma_2 \to Q \times \Gamma_1^* \times \Gamma_2^*$$

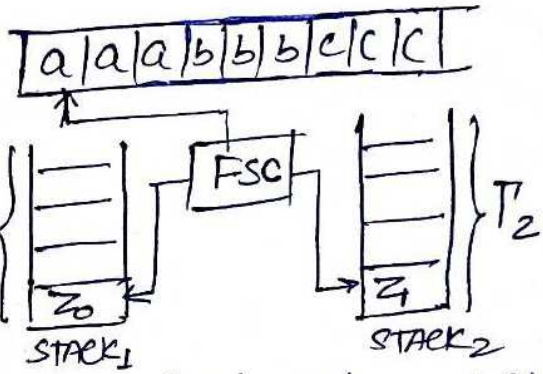
Ex:- $L = \{a^n b^n c^n \mid n \geq 1$, Design 2 STACK PDA$\}$

Soln: $L = \{abc, a^2b^2c^2, a^3b^3c^3, a^4b^4c^4, \dots\}$

Logic to Design 2-STACK PDA:-

- If a is read, Push a in STACK$_1$, No change in STACK$_2$
- If b is read, Pop a from Stack$_1$ and push b into stack$_2$ /*This ensures a=b*/
- If c is read, Pop b from STACK$_2$ /*ensure b=c*/

proved $a = b = c$



$\delta(q_0, a, Z_0, Z_1) \vdash (q_0, aZ_0, Z_1)$

$\delta(q_0, a, a, Z_1) \vdash (q_0, aa, Z_1)$

$\delta(q_0, b, a, Z_1) \vdash (q_1, \varepsilon, \cancel{bZ_1})$

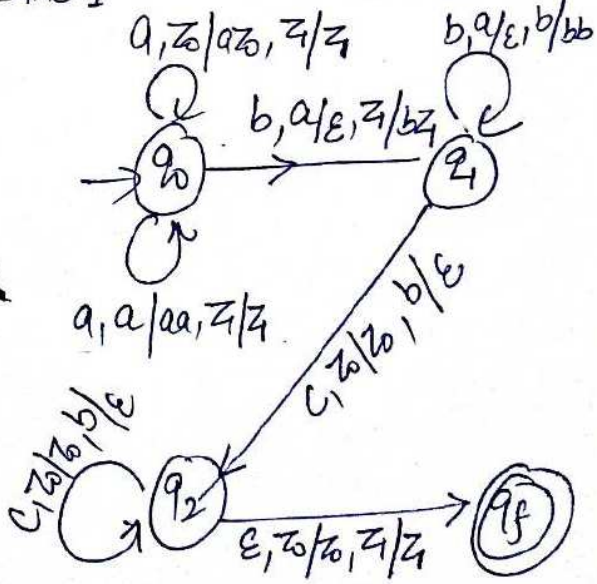
$\delta(q_1, b, a, \cancel{b}) \vdash (q_1, \varepsilon, \cancel{bb})$

$\delta(q_1, c, Z_0, b) \vdash (q_2, Z_0, \varepsilon)$

$\delta(q_2, c, Z_0, b) \vdash (q_2, Z_0, \varepsilon)$

$\delta(q_2, \varepsilon, Z_0, Z_1) \vdash (q_f, \varepsilon, \varepsilon)$

OR



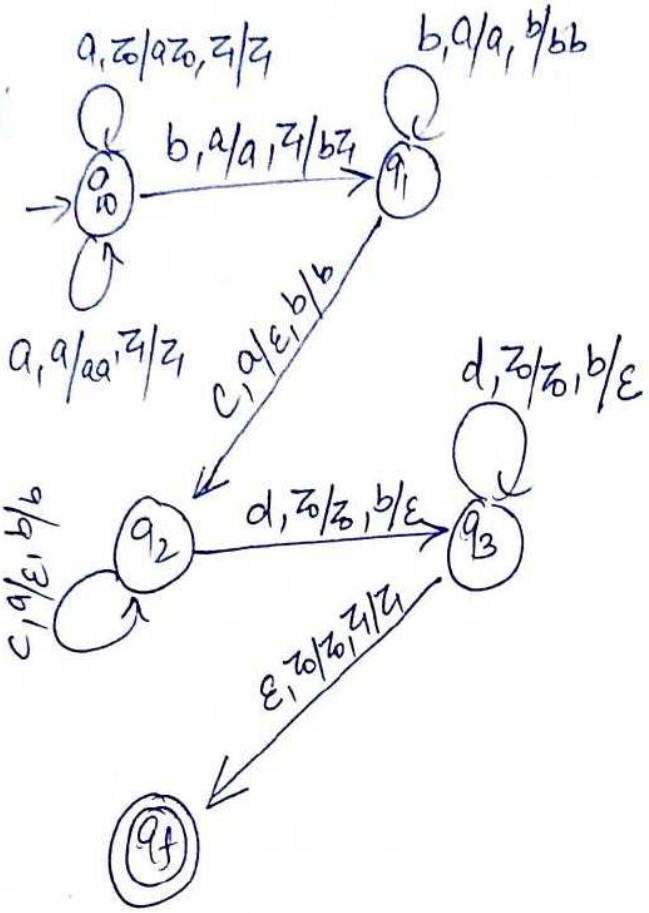
2 STACK PDA $= (\{q_0, q_1, q_2, q_f\}, \{a, b, c\}, \delta, \{Z_0, a\}, \{Z_1, b\}, Z_0, Z_1, q_0, q_f)$

Ans.

Ex:- Design 2STACK PDA for $L = \{a^m b^n c^m d^n \mid m, n \geq 1\}$

Sol^n:

2STACK-PDA



OR:

~~$\delta(q_0, z_0, z_1) \vdash (q_0, az_0, z_1)$~~

$\delta(q_0, a, z_0, z_1) \vdash (q_0, az_0, z_1)$

$\delta(q_0, a, a, z_1) \vdash (q_0, aa, z_1)$

$\delta(q_0, b, a, z_1) \vdash (q_1, a, bz_1)$

$\delta(q_1, b, a, b) \vdash (q_1, a, bb)$

$\delta(q_1, c, a, b) \vdash (q_2, \varepsilon, b)$

$\delta(q_2, c, a, b) \vdash (q_2, \varepsilon, b)$

$\delta(q_2, d, z_0, b) \vdash (q_3, z_0, \varepsilon)$

$\delta(q_3, d, z_0, b) \vdash (q_3, z_0, \varepsilon)$

$\delta(q_3, \varepsilon, z_0, z_1) \vdash (q_f, z_0, z_1)$

Pumping Lemma for CFL:- if L is/are CF

Pumping lemma is used to prove that a language is NOT CFL.

Pumping lemma:- if L is a CFL, Then $\exists P$ (Pumping Length) such that $\forall w \in L$, if $|w| \geq P$ Then $\exists u, v, x, y, z$ such that $w = uvxyz$

1. $|vxy| \leq P$
2. $|vy| > 0$
3. $uv^i xy^i z \in L, \forall i \geq 0$

NOTE:- To prove that a Language is not CFL using pumping lemma (for CFL) follow the steps given below: (we prove using Contradiction method)

→ Assume that L is CFL
→ It has to have a Pumping length (say P)
→ All strings longer than P can be Pumped $|w| \geq P$
→ Now find a string 'w' in L such that $|w| \geq P$
→ Divide w into uvxyz
→ Show that $uv^i xy^i z \notin L$ for some i
→ Then Consider the ways that w can be divided into uvxyz
→ Show that none of these can satisfy all the 3 pumping Conditions at the same time
→ w can not be pumped (Contradiction)

V. Imp:

L is a CFL → Pumping lemma hold

¬ Pumping lemma hold → ¬ L is a CFL

Pumping Lemma hold → L is a CFL or not CFL

Ex:. Show that $L = \{a^n b^n c^n \mid n \geq 0\}$ is not CFL using Pumping Lemma.

Soln: Assume that L is CFL, Then We Can Apply pumping Lemma Theorem

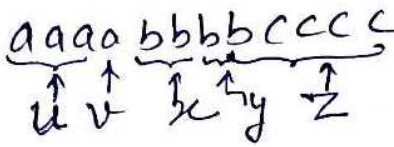
→ L must have a Pumping Length (Say P)

→ Now we take a string w Such that $w = a^P b^P c^P$, $|w| \geq P$, $3P \geq P$

→ We divide w into 5 parts uvxyz

let $P = 4$ so $w = a^4 b^4 c^4$

Case 1:. v and y each Contain only one type of symbol



, $|vxy| \leq P$, $|vy| > 0$

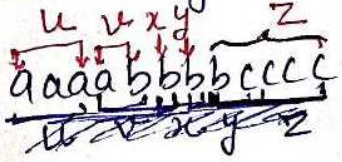
$|abbb| \leq 4$ $|ab| > 0$

$4 \leq 4$ TRUE $2 > 0$, TRUE

Then $uv^i xy^i z$, for $i = 2$

$uv^2xy^2z = aaaa^2bbb^2bcccc = a^5b^5c^4 \notin L$

Case 2:. Either v or y has more than one kind of Symbols



, $|vxy| \leq P$, $|vy| > 0$

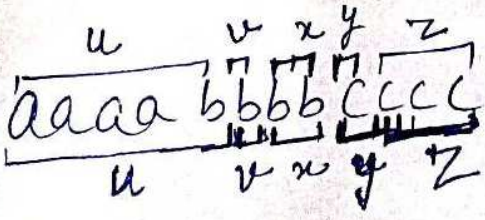
$|abbb| \leq 4$ $|abb| > 0$

$4 \leq 4$, TRUE $3 > 0$ TRUE

Then $uv^i xy^i z$, for $i = 2$

$uv^2xy^2z = aaaabab\,bb^2bcccc \Rightarrow a^4bab^5c^4 \notin L$

Case 3:



$|vxy| \leq P$ $|vy| > 0$

$|bbbc| \leq 4$ $|bc| > 0$

$4 \leq 4$, TRUE $2 > 0$ TRUE

Then $uv^i xy^i z$, for $i = 2$

$uv^2xy^2z = aaaabb^2bbc^2ccc \Rightarrow a^4b^5c^5 \notin L$

Contradicts in All Cases Hence L is not CFL

Ex:- Show that $L = \{ww \mid w \in (0,1)^*\}$ is not CFL

Sol^n: Assume L is a CFL Then we can apply pumping lemma.

Let Pumping Length is P

now we take a string w such that $w = 0^P 1^P 0^P 1^P$, $|w| \geq P$

$4P \geq P$ TRUE

We divide w into 5 parts uvxyz such that

$|vxy| \leq P$, $|vy| > 0$, Let $P = 5$

Case1:- 00000 | 1111 | 00000 | 1111 $\quad |vxy| \leq P, |vy| > 0$

u v x y z

$uv^2xy^2z = 00000\,1111\,1\,0\,0\,00001111 = 0^5 1^7 0 0^5 1^5 \notin L$

Case2: 00000 | 1111 | 00000 | 1111 $\quad |vxy| \leq P, |vy| > 0$

u v x y z

$uv^2xy^2z = 00000\,1111\,1\,00000\,1111 = 0^5 1^4 0^5 1^5 \notin L$

Case 3: 00000 | 1111 | 00000 | 1111 $\quad |vxy| \leq P, |vy| > 0$

u v x y z

$uv^2xy^2z = 00000\,11111\,11\,00000\,1111 = 0^5 1^8 0^5 1^5 \notin L$

Contradicts in all cases Hence L is not CFL

Ex:- $L = \{a^P$, P is prime No. not CFL$\}$

Sol^n: Assume L is CFL Then apply pumping lemma, Let

Pumping Length is m and Let P is a prime No

Such $P \geq m$. Choose a string w such that $|w| \geq m$, so $w = a^P$

We divide w into 5 parts $w = uvxyz$

$w = uvxyz = a^P$

$|w| = |uvxyz| = |uxz| + |vy| = P$, Let $|uxz| = q$ $\quad |vy| = r$

$uv^ixy^iz = |uxz| + i|vy|$, Let $i = q$

$= q + qr$

$= q(1+r)$ not prime No; means $uv^{i'}xy^{i'}z \notin L$

Contradicts our assumption Hence L is not CFL

Ex:- $L = \{a^{n^2} \mid n \geq 1 \text{ not CFL}\}$

Soln: let L is CFL, $L = \{a, a^2, a^3, a^4, \ldots\}$

Let Pumping length $p = 2$, $\omega = a^4$, $|\omega| \geq p$

$4 \geq 2$ TRUE

$\omega = uvxyz = aaaa = aa\,\varepsilon\,aa$, $|vxy| \leq p$

$u\ v\ x\ y\ z$

$2 \leq 2$, TRUE

$|vy| > 0$

$2 > 0$, TRUE

$uv^i xy^i z$, for $i = 0$

$uv^0xy^0z = aa^0\varepsilon a^0a = aa \notin L$

Contradicts our Assumption Hence L is not CFL.

Properties of Context free language (CFL):-

1- Closure Properties

2- Decision Properties

Closure Properties of CFL:- The CFL are closed Under some specific operations, closed means after doing that operation on a CFL the resultant language will also be a CFL.

Union:- if L_1 and L_2 are two CFL, their Union $L_1 \cup L_2$ will also be Context free language.

Proof:- $L_1 = CFL$, $L_2 = CFL$

$\uparrow$ $G_1 = CFG$, starting symbol S_1 $\quad$ $\uparrow$ $G_2 = CFG$, starting S_2

for $L_1 \cup L_2 \Rightarrow$ only Add starting Symbol (S)

$\boxed{S \to S_1 / S_2}$ CFG

Ex:- $L_1 = \{a^n b^n c^m \mid m,n \geq 0\}$, $L_2 = \{a^n b^m c^m \mid m,n \geq 0\}$

$L_3 = L_1 \cup L_2 = \{a^n b^n c^m \cup a^n b^m c^m \mid m,n \geq 0\}$ is Also CFL

Concatenation:- if L_1 and L_2 are two CFL, their Concatenation $L_1 \cdot L_2$ will also be CFL (Add $\boxed{S \to S_1 \cdot S_2}$, CFG production)

Ex:- $L_1 = \{a^n b^n \mid n \geq 0\}$, $L_2 = \{c^m d^m \mid m \geq 0\}$

$L_3 = L_1 \cdot L_2 = \{a^n b^n \cdot c^m d^m \mid m,n \geq 0\}$ is Also CFL

Transpose: $L = CFL$ Then L^T is also CFL

Ex: $L = \{a^n b^n \mid n \geq 0\}$ CFL, $L^T = \{b^n a^n \mid n \geq 0\}$ is also CFL.

Kleenstar: if $L_* = CFL$, its Kleene closure L_*^* will also be CFL.

for L^*, Add new start symbol $\boxed{S' \to SS' / \varepsilon}$ CFG production so L^* is also a CFL

Intersection: if L_1 and L_2 are two CFL, their intersection $L_1 \cap L_2$ need not be context free language. for Example.

$L_1 = \{a^n b^n \mid n \geq 1\}$, $L_2 = \{a^m b^n \mid n \geq 1\}$

$L_1 \cap L_2 = \{a^n b^n \mid n \geq 1\}$ is also CFL, But if

$L_3 = \{a^n b^n c^m \mid m, n \geq 1\}$ is a CFL and $L_4 = \{a^m b^n c^n \mid m, n \geq 1\}$ CFL

Then

$L_3 \cap L_4 = \{a^n b^n c^n \mid n \geq 1\}$ not CFL

Hence CFL is not closed Under intersection operation, means Intersection of CFL need not be CFL

Complement: L = CFL Then $\overline{L}$ = need not be CFL, means CFL is not closed Under Complement operation. for Example.

Proof: Assume Complement of CFL is CFL

if $L_1 \Rightarrow$ CFL Then $\overline{L_1}$ = CFL

if $L_2 \Rightarrow$ CFL Then $\overline{L_2}$ = CFL

Then

Then $\overline{\overline{L_1} \cup \overline{L_2}}$ = $L_1 \cap L_2$ = Already proved not closed

means Complement of CFL need not be CFL.

Some other operations also closed like

Homomorphism
Inverse Homomorphism
Substitution
Union with regular
Intersection with regular
Difference with regular. etc.

NOTE:

$L_{Regular} \cap L_{CFL}$ = CFL

Decision Properties of CFL: There are algorithms to decide of:

1- membership: String w is in CFL L
2- Emptiness: CFL L is empty
3- Finiteness: CFL L is finite

Finiteness: for a given CFG, There exist an Algo. to decide whether its language is finite or not

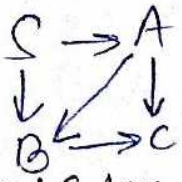
Steps:
1- Simplify CFG
2- Draw a directed Graph whose nodes are variables of the given Grammar
3- There exists an edge from node A to node B if there exist a Production of the form $A \to \alpha B \beta$, Now following 2 cases are possible

Case 1: if Directed Graph Contain a cycle Then the given Grammar is infinite

Case 2: if directed graph does not Contain any cycle Then The Grammar is finite

Ex:- $S \to AB \mid a$
$A \to Bc \mid b$
$B \to cc \mid c$

Draw a Directed Graph.



The Directed Graph does not Contain any cycle.
Therefore the Grammar is finite

Emptiness: Algo. to Decide whether CFL is empty or not.

Steps:-
1- Remove all the useless Symbol from the Grammar.
2- A useless Symbol is one that does not derive any string of terminals
3. if the start Symbol is found to be useless, Then language is empty otherwise not.

Membership Algo:- String w is in CFL L

CYK Algo:- CYK Algo. is a membership algo. of Context free Grammar.

- It is used to decide whether a given string belongs to the Language of Grammar or not.
- It is also known as CYK Algo. or Cocke-Younger-Kasami Algo. after its inventors.

NOTE:- CYK Algo. operates only on CFG given in Chomsky Normal form

NOTE:- The worst case running time of CYK Algo is $\theta(n^3 \cdot |G|)$, Here n = Length of parsed string and $|G|$ = size of the CNF Grammar G.

Algo:-

```
Begin
  for (i=1 to n do)
  Vi1 = {A | A→a is a production where i the symbol of x is a}
  for (J=2 to n do)
    for(i=1 to n-J+1 do)
    Begin
      VIJ = ϕ
      for K=1 to J-1 do
      VIJ = VIJ ∪ {A | A→BC is a Production where B is in
                  Vik and C is in V(i+K)(J-K)}
    End
End
```

Ex:- for a given grammar, check the acceptance of string w = baaba. using CYK Algo.

$S \rightarrow AB \mid BC$

$A \rightarrow BA \mid a$

$B \rightarrow CC \mid b$

$C \rightarrow AB \mid a$