PUSH DOWN AUTOMATA

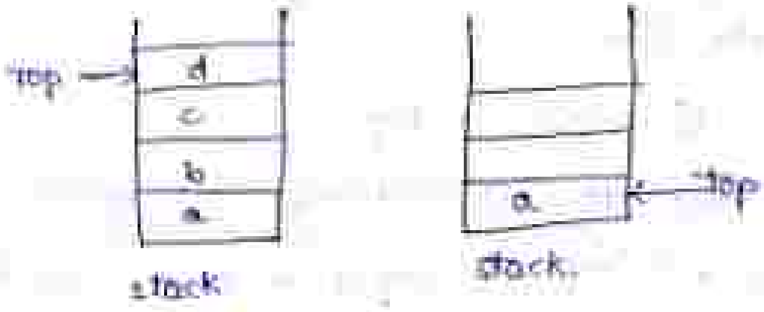
* Introduction * Basic model [Formal definition] * Graphical notation
* Instantaneous Description (ID) * Acceptance of PDA.

Introduction:-

* A PDA is a way to implement a CFG in a similar way we can design FA for Regular Grammar.
* PDA is more powerful than finite state machine.
* FSM has a very limited memory but a PDA has more memory.
* $\boxed{PDA = FSM + stack}$
* A stack is a way we arrange elements one on the top of stack.
* A stack does two basic operations
  - i) push :- A new element is added at the top of the stack.
  - ii) pop :- the top element of the stack is read and remove.

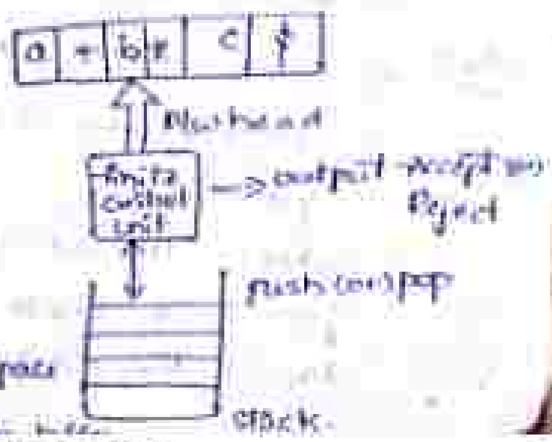
Ex:- push(a)
push(b)
push(c)
push(d)

pop()
pop()
pop()



* Basic model of PDA :-

PDA has three Components
i) input tape
ii) finite control unit
iii) stack



* A stack with infinite size
* It has unlimited amount of storage space
* Used to store data and remove the data ...

Formal definition:-

- mathematically a PDA is defined with 7 tuples like

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

$Q \rightarrow$ finite and non-empty set of states

$\Sigma \rightarrow$ finite and non-empty set of input symbols

$\Gamma \rightarrow$ finite and non-empty set of stack symbols

$\delta \rightarrow$ It is a transition function which is defined as

δ:

$$\boxed{Q \times \{\Sigma \cup \epsilon\} \times \Gamma^* \longrightarrow Q \times \Gamma^*}$$

$$\boxed{Q \times \Sigma^* \times \Gamma^* \longrightarrow Q \times \Gamma^*}$$

where 'δ' takes three tuples as i/p like $\delta(q, a, x)$

where i) q is a state in Q

ii) a is either an i/p symbol in Σ (or) a is also belongs ϵ

iii) x is a stack symbol i.e; member of Γ

iv) The o/p of δ is finite set of pairs like (p, γ)

where p: It is a new state.

γ: It is a set of stack symbols that replace 'x' at the top of the stack.

Ex:- 1) If $\gamma = \epsilon$ then the stack is pop

2) If $\gamma = x$ then the stack is unchange (the bypass operation)

3) If $\gamma = yz$ then x is replaced by z and 'y' is pushed on to the stack.

Ex:- 1) $\delta(q_0, a, z) = (q_1, yz)$

$\Rightarrow$ It indicates that from state q_0, reading i/p symbol 'a'

where, top of the stack z then the finite control goes to q_1 state and adding the element y to the top of the stack.

4) $\delta(q_1, a, Z) = (q_2, \epsilon)$

→ It indicates that 'Z' is removed from the stack and state is changed from q_1 to q_2

5) $\delta(q_1, a, Z) = (q_2, Z)$

⇒ It indicates that on reading symbol 'a' state is changing from q_1 to q_2 and there is no change in the stack (bypass operation)

q_0 → It is the initial state.

$\boxed{q_0 \in Q}$

Z_0 → It is the start stack symbol.

$\boxed{Z_0 \in \Gamma}$

F → It is the set of final (or) accepting state and

($F \subseteq Q$).

Graphical representation :-

The Graphical representation of PDA is "Transition diagram"

FA :-

$A \xrightarrow{a} B$

$\delta(A, a) = B$

PDA :-

$A \xrightarrow{a,\, b \to c} B$ (next state)

this symbols is pushed onto stack ↓ may be ε ↓ the symbol is pushed.

$\delta(A, a, b) = (B, C)$

Current state

i/p symbols may be ε

Symbol in top of the stack this symbol is poped ↓ ε means the stack neither pushed nor poped.

Instantaneous description :-

It is used to describe the configuration of PDA at given Instance.

ID remembers the state and stack content.

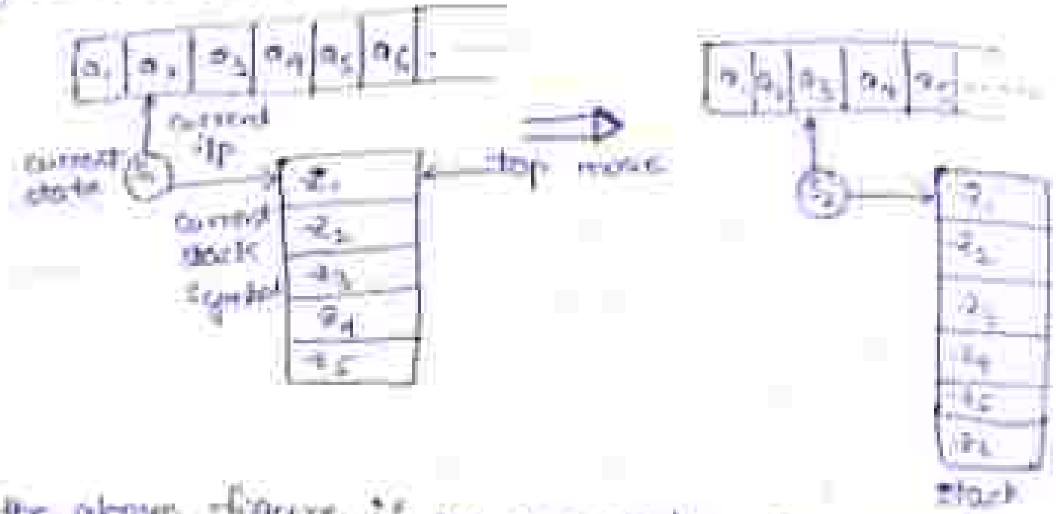
It was defined by Triple (q, w, γ) where

q → is a state.

w → input symbols of string

γ → is a string of stack symbols.

Example :- $\delta(q_0, z_0) = (q_1, ...)$



From the above figure, if we are reading the current i/p symbol 'a_2' at current state 's_1' and current stack symbol 'z_1' then after a move we will reach to state s_2 and there will come new symbol on the top of the stack. This description can be represented as.

1) push operation :-

$$\delta(q_0, a, z_0) = (q_1, az_0)$$

push 'a' onto the stack

current state (or) present state; current i/p symbol; current stack top symbol; change the state from q_0 to q_1.

2) pop operation :-

$$\delta(q_0, x, y) = (q_1, \epsilon)$$

current state; current i/p symbol; current stack top symbol; change the state from q_0 to q_1; pop the stack (or) removing the stack.

Acceptance of PDA :-

There are two ways to accept a language by PDA. They are i) Accepted by empty stack

ii) Accepted by final state

Accepted by empty stack :-

The given language accepted by empty stack to be define as $L(M) = \{w \mid \delta(q_0, w, z_0) \overset{*}{\vdash} (p, \epsilon, \epsilon)$ for some p in $Q\}$

that is, if stack becomes empty after scanning entire string then it is accepted by PDA otherwise not accepted.

Accepted by final string:-

The language accepted by final state is to be defined as the given language

$L(M) = \{ w \mid \delta(q_0, w, z_0) \vdash (p, \epsilon, \gamma)$ for some $p \in F$ and $\gamma \in \Gamma^* \}$

that is, eventhough stack is not empty, after scanning the string. if the finite control reaches to the final state then it is accepted. otherwise, not accepted.

Design of PDA :-

Types of PDA:-

i) Deterministic PDA :- if all derivations in the design has to give only single move

ii) Non Deterministic PDA :- if derivation generates more than one move in the designing of a particular task.

1) Design a PDA that accepts equal no. of A's and B's

Sol: δ: $\delta(q_0, a, z_0) = (q_1, az_0)$

$\delta(q_1, a, a) = (q_1, aa)$

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

$\delta(q_1, \epsilon, z_0) = (q_1, \epsilon)$

$\delta(q_1, a, z_0) = (q_1, az_0)$

∴ the PDA machine for the above language is defined as

$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ where $Q = \{q_0, q_1\}$

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

$\Gamma = \{z_0\}$

δ:

$q_0 = \{q_0\}$

$z_0 = \{z_0\}$

$F = \{ \}$

(or) Read A's $\rightarrow$ push operation, Read B's $\rightarrow$ push operation

Consider a String $w = \{abab\}$ Read a's

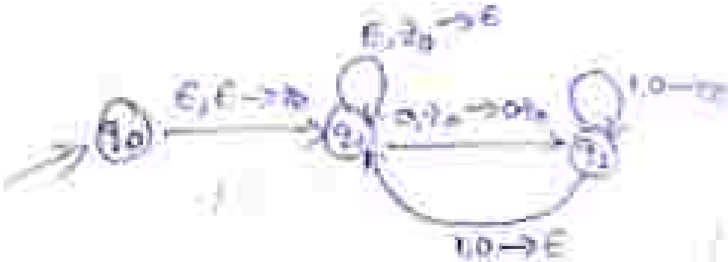
$\delta(q_0, abab, z_0) = \delta(q_1, bab, az_0)$

③ Design a PDA for the language $L = \{0^n 1^{2n} \mid n \geq 1\}$

sol:- $L = \{0^n 1^{2n} \mid n \geq 1\}$

Read one 0 → push

Read two 1's → pop



$\delta(q_0, \epsilon, \epsilon) = (q_1, z_0)$

$\delta(q_1, 0, z_0) = (q_2, 0z_0)$

$\delta(q_2, 0, 0) = (q_2, 00)$

$\delta(q_2, 1, 0) = (q_2, 00)$

$\delta(q_2, 1, 0) = (q_1, \epsilon)$

$\delta(q_1, \epsilon, z_0) = (q_1, \epsilon, \epsilon)$

$\delta(q_1, 1, 0) = (q_1, \epsilon)$

$\delta(q_1, 1, 0) = (q_1, 0z_0)$

④ Consider the string $w = \{001111\}$

$\delta(q_1, 001111, z_0) = \delta(q_2, 01111, 0z_0)$

$= \delta(q_2, 1111, 00)$

$= \delta(q_2, 111, 00)$

$= \delta(q_1, 11, 0z_0)$

$= \delta(q_1, 1, 0z_0)$

stack

$= \delta(q_1, \epsilon, z_0)$

$= \delta(q_1, \epsilon, \epsilon)$ //.

Design a PDA for the language $L = \{0^n 1^n \mid n \geq 1\}$

Read 0's → push

Read 1's → pop

$\delta(q_0, \epsilon, \epsilon) = (q_1, z_0)$

$\delta(q_1, 0, z_0) = (q_2, 0z_0)$

$\delta(q_2, 0, 0) = (q_2, 00)$

$\delta(q_2, 1, 0) = (q_3, \epsilon)$

$\delta(q_3, 1, 0) = (q_3, \epsilon)$

$\delta(q_3, \epsilon, z_0) = (q_3, \epsilon, \epsilon)$

* Design a PDA for the language $L = \{ww^R \mid w \in (a+b)^*\}$

ii) $L = \{w c w^R \mid w \in (a+b)^*\}$

sol) i) $L = \{ww^R \mid w \in (a+b)^*\}$

In this language contains palindrome string. if,

if $w = ab$, $w^R = ba$ then $ww^R = abba$ is a palindrome

* We can read no. of a's and b's and pushed them into stack until we can reach the mid position of i/p string.
* In the mid position we can't read any i/p and can't push onto stack.
* After mid position when we read a or b then pop them from the stack. This process is repeated until stack is empty.

$\delta(q_0, \epsilon, \epsilon) = (q_1, z_0)$

$\delta(q_1, a, z_0) = (q_1, az_0)$

$\delta(q_1, b, z_0) = (q_1, bz_0)$

$\delta(q_1, \epsilon, \epsilon) = (q_2, z_0)$

$\delta(q_2, a, a) = (q_2, \epsilon)$

$\delta(q_2, b, b) = (q_2, \epsilon)$

$\delta(q_2, \epsilon, z_0) = (q_f, \epsilon, \epsilon)$

ii) $L = \{ wcw^R \mid w = (a+b)^* \}$

$w = ab$
$w^R = ba$
$wcw^R = abcba$

$\delta(q_0, \epsilon, Z_0) = (q_1, Z_0)$
$\delta(q_1, a, Z_0) = (q_1, aZ_0)$
$\delta(q_1, b, Z_0) = (q_1, bZ_0)$
$\delta(q_1, a, a) = (q_1, aa)$
$\delta(q_1, a, b) = (q_1, ab)$
$\delta(q_1, b, a) = (q_1, ba)$
$\delta(q_1, b, b) = (q_1, bb)$

$\delta(q_1, c, Z_0) = (q_2, Z_0)$
$\delta(q_1, c, a) = (q_2, a)$
$\delta(q_1, c, b) = (q_2, b)$
$\delta(q_2, a, a) = (q_2, \epsilon)$
$\delta(q_2, b, b) = (q_2, \epsilon)$
$\delta(q_2, \epsilon, Z_0) = (q_3, \epsilon)$

Deterministic pushdown Automata:-

→ DPDA is 7 tuple like $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

where Q is finite and non empty set of states
Σ is finite and nonempty set of i/p Alphabet
Γ is finite set of stack symbols
δ is a mapping function used for mapping (or) moving from current state to next state. It is denoted as $\delta(q_0, x, Z_0) = (q, xp)$ where

q_0 is current state
x is current i/p symbol
Z_0 is current stack symbol
q is next state
xp shows top of the stack

if δ denotes a unique transition for each i/p then PDA is said to be deterministic PDA

Ex:- 1) $L = \{a^n b^n \mid n \geq 1\}$
2) $L = \{wcw^R \mid w = (a+b)^*\}$

Non deterministic PDA:-

It is a 7 tuple like $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ where
Q is finite and non empty set of states
Σ is finite and non empty set of i/p Alphabet
Γ is finite set of stack symbols.

δ is a mapping function used for moving from current state to next state and is defined as $\delta(q_0, x, z_0) = (q_1, \beta)$

q_0 is current state

x is current i/p symbol

z_0 is stack symbol

q_1 is next state

β is top of the stack

If δ denotes more than one transition for a particular i/p symbol then the PDA is said to be non-deterministic PDA

Ex: $L = \{ww^R \mid w \in (a+b)^*\}$

Context free grammar and push down automata:—

conversion of CFG to PDA

conversion of PDA to CFG

i) Conversion of CFG to PDA:—

* for constructing a PDA from given CFG it is necessary to convert this CFG to some normal form like GNF

* for converting given CFG to PDA, By this method the necessary condition is that the first symbol on RHS of production rule must be a terminal symbol. This rule that can be used to obtain PDA from CFG.

Algorithm:—

Rule 1 :- for non-terminal symbols, add following rule

$\delta(q, \epsilon, A) = (q, \alpha)$ where the production rule is $A \to \alpha$

Rule 2 :- for each terminal symbols, add following rule

$\delta(q, a, a) = (q, \epsilon)$ for every terminal symbol 'a' in given CFG.

Ex:- construct a PDA for the given CFG
$S \to 0BB$
$B \to 0S$
$B \to 1S$
$B \to 0$

Sol:- The given CFG $G = (V, T, P, S)$ where V = non-terminals $\{S, B\}$

$T = \{0, 1\}$

$P:$ $S \to 0BB$

$B \to 0S$

$B \to 1S$

$B \to 0$

$S = \{S\}$

Rule 1 : $A \to \alpha$

$\delta(q, \epsilon, A) = (q, \alpha)$

$S \to 0BB$

$\delta(q, \epsilon, S) = (q, 0BB)$

$B \to 0S$

$\delta(q, \epsilon, B) = (q, 0S)$

$B \to 1S$

$\delta(q, \epsilon, B) = (q, 1S)$

$B \to 0$

$\delta(q, \epsilon, B) = (q, 0)$

Rule 2:

Terminals

$\delta(q, a, a) = (q, \epsilon)$

$T = \{0, 1\}$

$\delta(q, 0, 0) = (q, \epsilon)$

$\delta(q, 1, 1) = (q, \epsilon)$

$\therefore$ the corresponding PDA for the given CFG is defined as

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

$Q = \{q\}$

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

$\Gamma = \{S, B, 0, 1\}$

δ = it is a transition symbol defined as

$\delta(q, \epsilon, S) = (q, 0BB)$

$\delta(q, \epsilon, B) = (q, 0S)$

$\delta(q, \epsilon, B) = (q, 1S)$

$\delta(q, \epsilon, B) = (q, 0)$

$\delta(q, 0, 0) = (q, \epsilon)$

$\delta(q, 1, 1) = (q, \epsilon)$

$q_0 = \{q\}$

$z_0 = \{z_0\}$

$F = \{\ \}$

Q) construct a PDA for the following CFG $S \to 0S1$
$S \to A$
$A \to 1A0 | S | \epsilon$.

Sol:- The given CFG is $S \to 0S1$
$S \to A$
$A \to 1A0$
$A \to S$
$A \to \epsilon$

elimination of ϵ-production:-

$A \to \epsilon$ ×
$S \to A$
$S \to \epsilon$ ×
$S \to 0S1$
$S \to 0S1$
$S \to 01$

$A \to 1A0$
$A \to 1\epsilon0$
$A \to 10$
$A \to S$
$A \to \epsilon$ ×

∴ $S \to 0S1 | 01$
$S \to A$
$A \to 1A0 | 10$
$A \to S$.

elimination of unit productions:-

$S \to A$ ×
$S \to 1A0 | 10$

$A \to S$ ×
$A \to 0S1 | 01$

∴ the resultant CFG is.

$S \to 1A0$
$S \to 10$
$A \to 0S1$
$A \to 01$

∴ the Simplified CFG is.

$S \to 1A0$
$S \to 10$
$A \to 0S1 | 1A0 | 10$
$A \to 01$
$S \to 0S1 | 01$

Method-1
$P \to 1$
$Q \to 0$

$S \to 1A0$
$S \to 1AQ$ ✓

$S \to 10$
$S \to 1Q$ ✓

$A \to 0S1$
$A \to 0SP$ ✓

$A \to 10$
$A \to 1Q$ ✓

$S \to 0S1$
$S \to 0SP$ ✓

$A \to 1A0$
$A \to 1AQ$ ✓

$A \to 01$
$A \to 0P$ ✓

$S \to 01$
$S \to 0P$ ✓

∴ the simplified CFG in GNF is
$S \to 1AQ$ $A \to 0SP$
$S \to 1Q$ $A \to 1Q$
$S \to 0SP$ $A \to 1AQ$
$S \to 0P$ $A \to 0P$
$P \to 1$
$Q \to 0$

Rule-1 ∴ the PDA is

$S \to 1AQ$
$\delta(q, \epsilon, S) = (q, 1AQ)$

$S \to 1Q$
$\delta(q, \epsilon, S) = (q, 1Q)$

$S \to 0SP$
$\delta(q, \epsilon, S) = (q, 0SP)$

$S \to 0P$
$\delta(q, \epsilon, S) = (q, 0P)$

$A \to 0SP$
$\delta(q, \epsilon, A) = (q, 0SP)$

$A \to 1Q$
$\delta(q, \epsilon, A) = (q, 1Q)$

$A \to 1AA$

$\delta(q, \epsilon, A) = (q, 1AA)$

$A \to 0P$

$\delta(q, \epsilon, A) = (q, 0P)$

$P \to 1$

$\delta(q, \epsilon, P) = (q, 1)$

$A \to 0$

$\delta(q, \epsilon, A) = (q, 0)$

Rule 2: Terminals

$\delta(q, a, a) = (q, \epsilon)$

$a = \{0, 1\}$

$\delta(q, 0, 0) = (q, \epsilon)$

$\delta(q, 1, 1) = (q, \epsilon)$

method-2

The Given CFG is $S \to 0S1$

$S \to A$

$A \to 1A0$

$A \to S$

$A \to \epsilon$

The resultant PDA is $S \to 0S1$

$\delta(q, \epsilon, S) = (q, 0S1)$

$S \to A$

$\delta(q, \epsilon, S) = (q, A)$

$A \to 1A0$

$\delta(q, \epsilon, A) = (q, 1A0)$

$A \to S$

$\delta(q, \epsilon, A) = (q, S)$

$A \to \epsilon$

$\delta(q, \epsilon, A) = (q, \epsilon)$

Construct PDA for the following CFG $S \to aABB \mid aAA$

$A \to aBB \mid a$

$B \to bBB \mid A$

sol:- The Given CFG is $S \to aABB$

$S \to aAA$

$A \to aBB$

$A \to a$

$B \to bBB$

$B \to A$

elimination of unit production:-

$B \to A$ ✗

$B \to aBB$

$B \to a$

$\therefore$ After eliminating unit production $B \to A$, the resultant CFG in GNF is $S \to aABB$

$S \to aAA$

$A \to aBB$

$A \to a$

$B \to bBB$

$B \to aBB$

$B \to a$

$S \rightarrow aABB$

$\delta(q, \epsilon, S) = (q, aABB)$

$S \rightarrow aAA$

$\delta(q, \epsilon, S) = (q, aAA)$

$A \rightarrow aBB$

$\delta(q, \epsilon, A) = (q, aBB)$

$A \rightarrow a$

$\delta(q, \epsilon, A) = (q, a)$

$B \rightarrow bBB$

$\delta(q, \epsilon, B) = (q, bBB)$

$B \rightarrow aBB$

$\delta(q, \epsilon, B) = (q, aBB)$

$B \rightarrow a$

$\delta(q, \epsilon, B) = (q, a)$

Conversion of PDA to CFG :-

- If $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is a PDA then there exists CFG G which is accepted by PDA (M)

Let G be a CFG which is generated by PDA. The G can be defined as $G = (V, T, P, S)$ where S is the start symbol and the set of non-terminals $V = \{S, q, q', Z_0\}$ where $S, q, q' \in Q$ and $Z_0 \in \Gamma$.

Now, we get set of production rules using the following algorithm.

Algorithm :-

Rule 1 :- The start symbol production rule can be $S \rightarrow [q, Z_0, q']$

where q indicates present state

q' indicates next state.

Z_0 is the stack symbol.

Rule 2 - If there exists a move of PDA as then the production rule can be return as $\delta(q, a, Z_0) = (q', \epsilon)$

$[q, Z_0, q'] \rightarrow a$

③ If there exists a move of PDA as $\delta(q, a, Z_0) = (q', Z_1 Z_2 \ldots)$
then the production rules can be written as

$$[q, Z_0, q'] \to a [q_1, Z_1, q_2][q_2, Z_2, q_3][q_3, Z_3, q_4] \ldots [q_m, Z_m, q']$$

Q: Construct a CFG from the following PDA $M = (\{q_0, q_1\}, \{0, 1\}, \{S, A\}, \delta, q_0, S, \phi)$ and

δ:

$\delta(q_0, 1, S) = (q_0, AS)$

$\delta(q_0, \epsilon, S) = (q_0, \epsilon)$

$\delta(q_0, 1, A) = (q_0, AA)$

$\delta(q_0, 0, A) = (q_1, A)$

$\delta(q_1, 1, A) = (q_1, \epsilon)$

$\delta(q_1, 0, S) = (q_0, S)$

Sol: Let we will construct a CFG $G = (V, T, P, S)$ where
$T = \{0, 1\}$
$V = \{S \cup [q_0, S, q_0], [q_0, S, q_1], [q_1, S, q_0], [q_1, S, q_1], [q_0, A, q_0], [q_0, A, q_1], [q_1, A, q_0], [q_1, A, q_1]\}$

Now, let us build the production rules as
using rule ② the production rules for start symbol is

$P_1: S \to [q_0, S, q_0]$

$P_2: S \to [q_0, S, q_1]$

using Rule ③ of the Algorithm for the $\delta(q_0, 1, S) = (q_0, AS)$

$P_3: [q_0, S, q_0] \to 1 [q_0, A, q_0][q_0, S, q_0]$

$P_4: [q_0, S, q_0] \to 1 [q_0, A, q_1][q_1, S, q_0]$

$P_5: [q_0, S, q_1] \to 1 [q_0, A, q_0][q_0, S, q_1]$

$P_6: [q_0, S, q_1] \to 1 [q_0, A, q_1][q_1, S, q_1]$

Now, for $\delta(q_0, \epsilon, S) = (q_0, \epsilon)$ using Rule ③ of Algorithm we get

$P_7: [q_0, S, q_0] \to \epsilon$

$P_8: [q_0, S, A] \to q_0/\phi$

Now for $\delta(q_0, 1, A) = (q_0, AA)$ using Rule ③ of Algorithm

$P_9: [q_0, A, q_0] \to 1 [q_0, A, q_0][q_0, A, q_0]$

Now for $\delta(q_0, 0, A) = (q_0, AA)$ string Rule-6 of Algorithm

$P_9: [q_0, A, q_0] \rightarrow 0[q_0, A, q_1][q_1, A, q_0]$

$P_{10}: [q_0, A, q_1] \rightarrow 0[q_0, A, q_0][q_0, A, q_1]$

$P_{11}: [q_0, A, q_1] \rightarrow 0[q_0, A, q_1][q_1, A, q_1]$

Now, for $\delta(q_0, 0, A) = (q_1, A)$ string

$P_{12}: [q_0, A, q_0] \rightarrow 0[q_1, A, q_0]$

$P_{13}: [q_0, A, q_1] \rightarrow 0[q_1, A, q_1]$

Now for $\delta(q_1, 1, A) = (q_1, \epsilon)$

$P_{14}: [q_1, A, q_1] \rightarrow 1$

Now for $\delta(q_1, 0, S) = (q_0, S)$

$P_{15}: [q_1, S, q_0] \rightarrow 0[q_0, S, q_0]$

$P_{16}: [q_1, S, q_1] \rightarrow 0[q_0, S, q_1]$

PDA with two stacks:

(a) PDA with one stack:

① $L = \{a^n b^n \mid n \geq 1\}$

consider the string $w = aabb$

read a's → push

read b's → pop

stack

a a b b $

when stack is empty then the string aabb is accepted

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

consider string $w = aabbcc$

read a's → push

read b's → pop

read c's → no change

stack

a a b b c c $

when read c there is no change in stack without completion of reading string 'w' the stack is empty so, string is not accepted

(b) PDA with two stacks:—

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

stack 1 stack 2

w = a a b b c c $

read a's → push (in stack₁)

read b's → push (in stack₂)

read c's → pop (a from stack₁ and 'b' from stack₂)

when two stacks are empty then string 'w' is accepted

∴ the PDA with two stacks is more powerful than a PDA with one stack.

FA + 0-stack = NFA or DFA

FA + 1-stack = PDA

FA + 2-stack = PDA with two stacks.

Applications of PDA:—

* used for deriving a string from the grammar.
* used for designing Top-down parser and bottom-up parser in compiler design.
* It works on regular grammar and Context-free grammar
* It accepts regular language and CFL.
* It has remembering capability by maintaining a stack.
* It is more powerful than FA.