Derivation Trees / Parse Tree

→ Syntax analysis is also called Syntax analyzer / parser

An Ordered rooted tree that graphically represents of how strings are derived from a CFG.

$G = \{ V, T, P, S \}$

P: S → 0B

A → 1AA / ε

B → 0AA

A → 1AA

A → ε

$V = (NT) = \{S, A, B\}$

$T = \{0, 1\}$

$S = \{S\}$

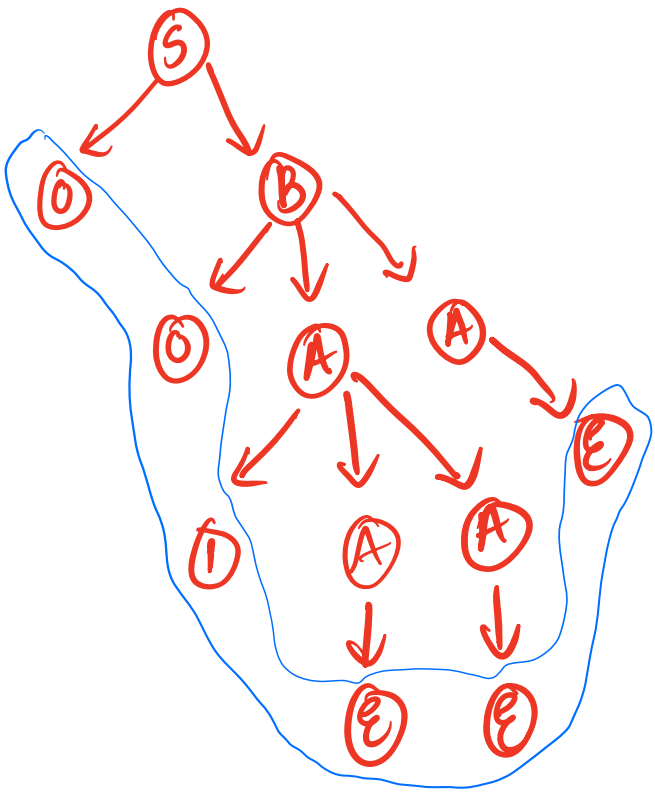
$S \rightarrow 0B$

$A \rightarrow 1AA$

$A \rightarrow \varepsilon$

$B \rightarrow 0AA$

"001" Input string



Root vertex: Should be start symbol

Vertex: Non-Terminals

Leaves: Terminals or ε.

2 Types of Derivation Trees:

Left DT (LDT)

A LDT obtained by applying production to left most variable in each step

Right DT (RDT)

" Right most variable,

→ Generate a string 'aabaa' from the

G: $S \to aAS / ass / \varepsilon$

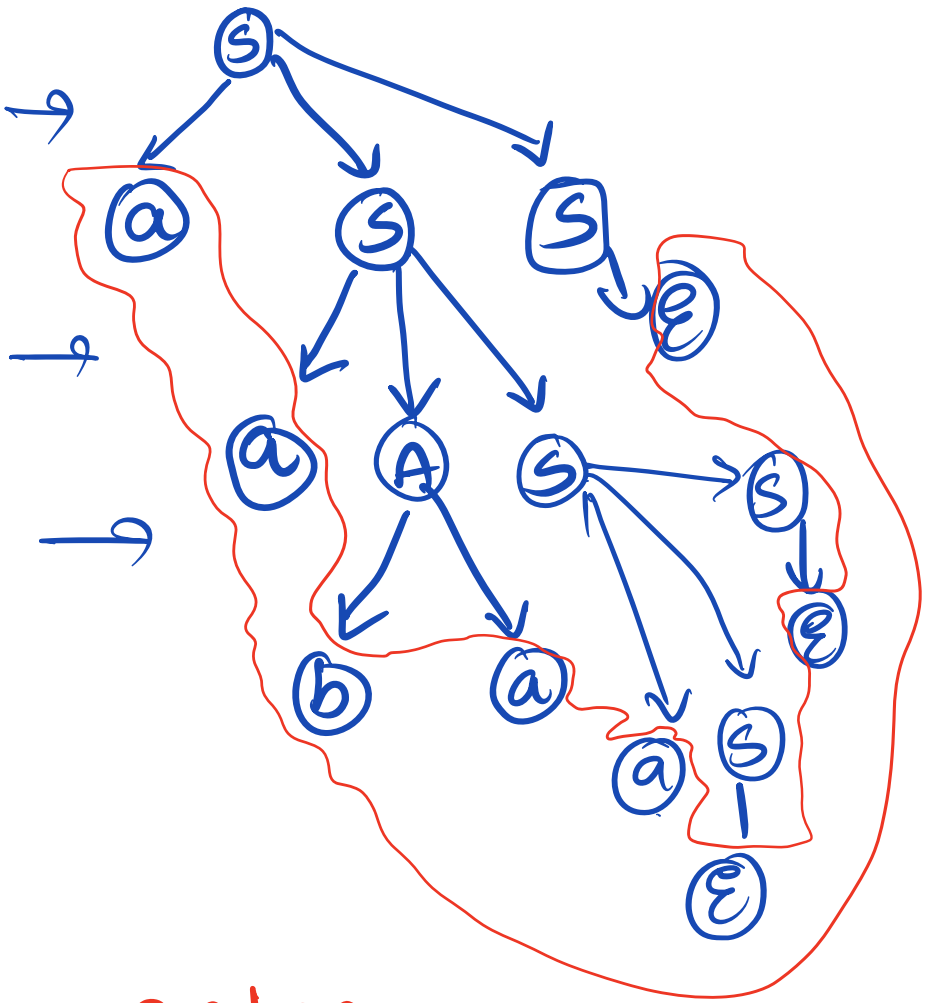
$A \to sbA / ba$

1. $S \to aAS$ ✓
2. $S \to ass$
3. $S \to \varepsilon$

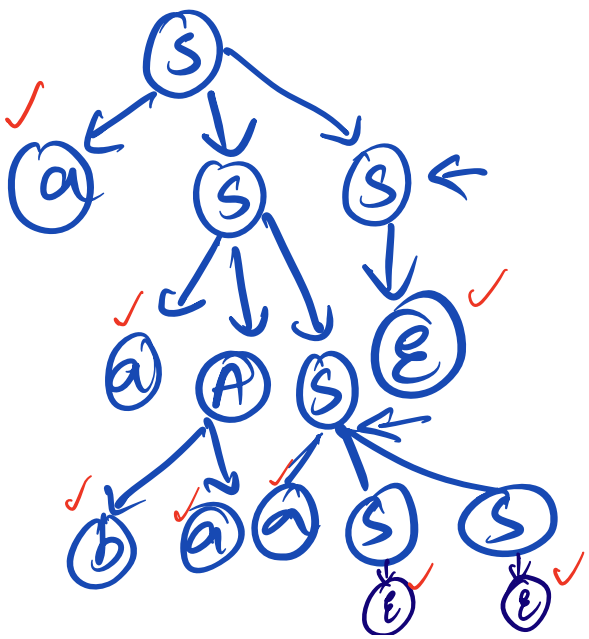
4. $A \to sbA$
5. $A \to ba$

✓✓✓✓✓

aabaa



aabaa



aabaaεε

aabaa

L → R

Ambiguous Grammar

If there exists 2 or more derivation trees for a string.

2 or more LDT ✓	2 or more RDT ✓

Example:

$G_1 = \{\{S\}, \{a+b, +, *\}, P, S\}$

$P = S \rightarrow S+S \mid S*S \mid a \mid b$.

" $a + a * b$ "

Sol^n

P:
1. $S \rightarrow S+S$ –
2. $S \rightarrow S*S$ ✓
3. $S \rightarrow a$ ✓
4. $S \rightarrow b$ ✓

NT = {S}
T = {a, b, +, *}
S = {S}

LDT ① 1. $S \to \underset{L}{S} + \underset{R}{S}$

$\to a + S$

Left $\to a + S * S$

$\to a + a * S$

$\to a + a * b$

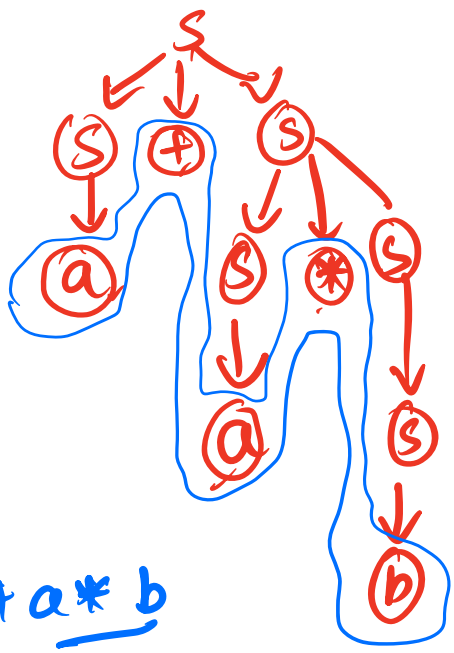
LDT ② 2. $S \to S * S$

$S \to S + S * S$

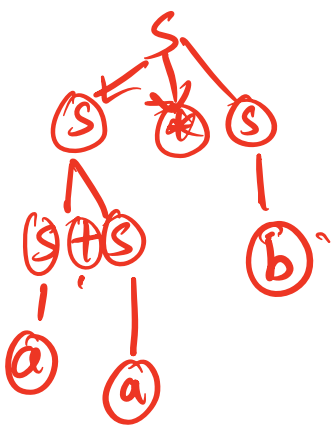
$S \to a + S * b$

$S \to a + a * S$

$S \to a + a * b$



a + a * b

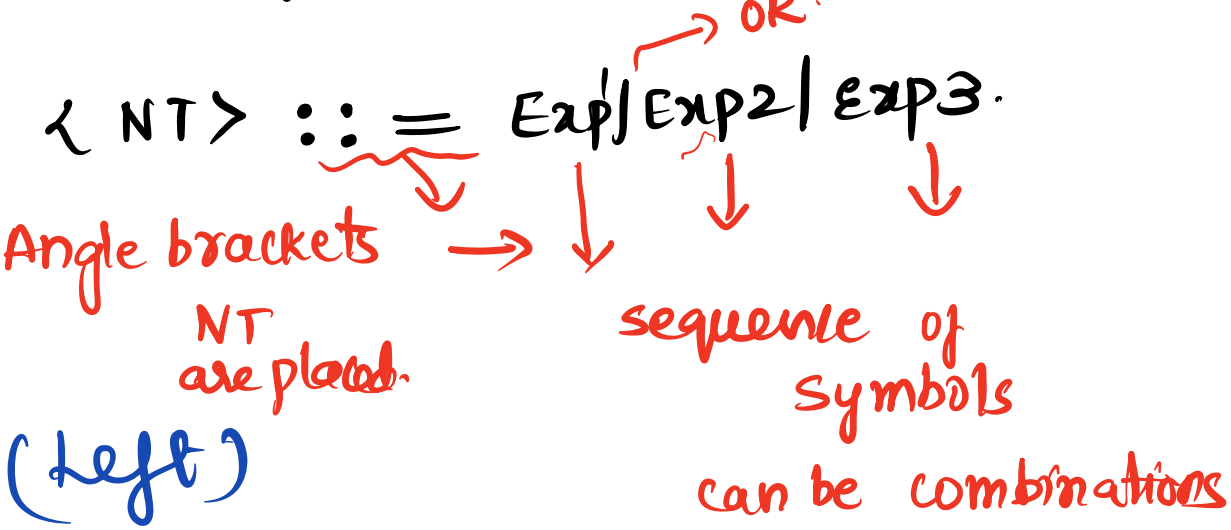


Ambiguous Grammar

Backus – Naur Form (BNF Notation)

→ technique for CFG

→ specifying the syntax of language.



Rules:

① Symbols are written with [ ]

② Repeating the symbol 0 or more times (*) (ab)*

③ " for 1 or more times (+) (ab)+

④ Alternative rules are separated by vertical bar

$S \rightarrow a \mid b$

⑤ Group of items must be enclosed within bracket.

Example:

$A \rightarrow id1 \mid 2 \mid id2$

A : factor

<factor> ::= <identifier> /

<constant> /

<identifier2>

② $S \rightarrow a \mid (C-d)$

$\langle S \rangle ::= \langle a \rangle \mid \langle C \rangle [-] \langle d \rangle$

CHOMSKY NORMAL FORM (CNF)

Restriction:

The length of RHS, that is the elements in RHS should be either 2 variables or a terminal.

2 N.T

1 T

CFG is in CNF if the products are in

1. $A \rightarrow a$ (1 T) ✓
2. $A \rightarrow BC$ (2 N.T) ✓

2 N.T

3. $B \rightarrow ab \mid Bd$

3. $B \rightarrow ab$ (NO) ✗

4. $B \rightarrow Bd$ (NO)

$B \rightarrow B\ d$

(1. NT) (T)

upper lower

Examples:

Gr1:

1. $S \rightarrow BE$
2. $S \rightarrow d$
3. $B \rightarrow a$
4. $E \rightarrow b$

CNF ? YES ✓

Gr2:

$S \rightarrow bB$ ✓ ~ ✗

$B \rightarrow e$ ✓

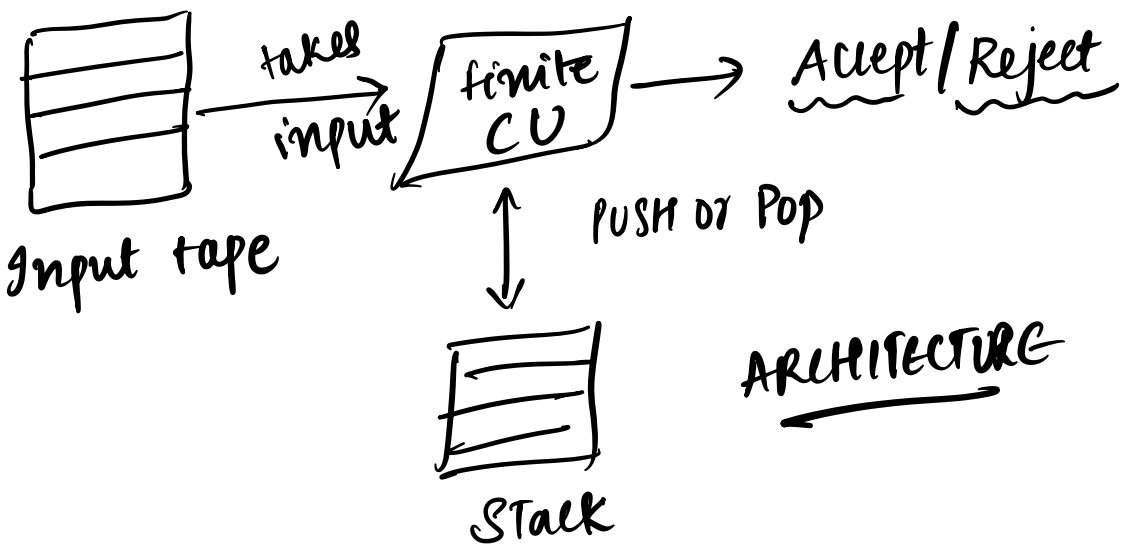
$D \rightarrow f$ ✓

NO ✓

Push Down Automata: (PDA)

PDA has 3 components:

1. An input Tape
2. A finite control unit
3. A stack with infinite size



ARCHITECTURE

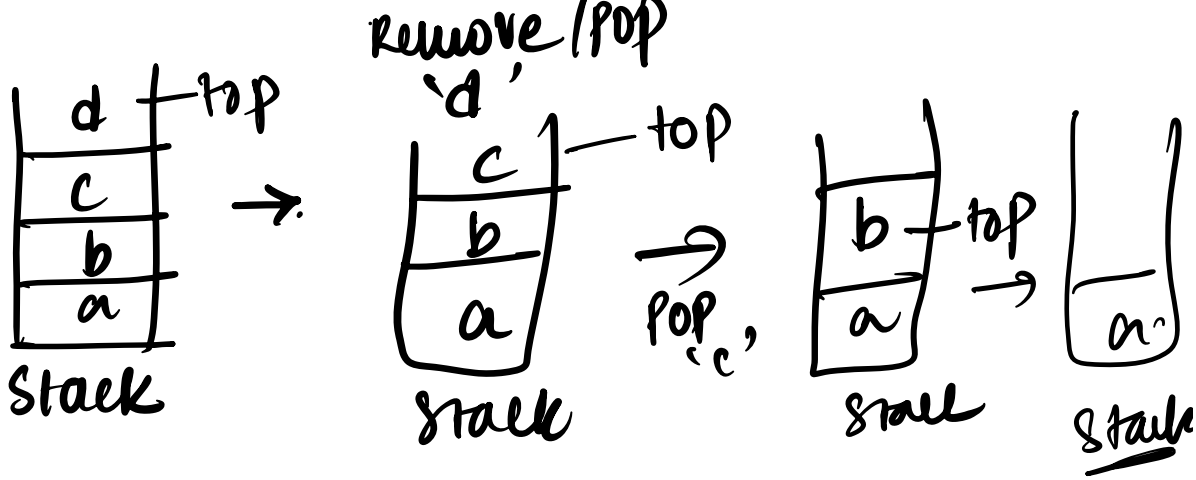
Definition:

A way to implement CFG is a similar way we design FSA for Regular Grammars.

* PDA has more memory

PDA = FSA + A Stack

Pile of Books: Example:



PDA has 7 tuples.

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

$$\text{PDA} = \underbrace{Q, q_0, \Sigma, \delta, F}_{\text{FSA}} + \underbrace{\Gamma, z_0}_{\text{stack}}$$

Q = A finite set of States

Σ = " " of inputs

q_0 = start state

δ = Transition function

F = Final State

(Gama) Γ = A finite stack Alphabet

z_0 = the start stack symbol.

z_0

δ : argument as a input of a triple.

$$\delta(q, a, X)$$

where
- q : state in Q
- a : input symbol in Σ or $a = \varepsilon$
- X : X is a stack symbol, member of Γ (upper case)

output of δ = finite set (P, γ) (lower case Gama)
- P : new state
- γ : string of stack symbols, that replaces 'X' at top of the stack.

Example:

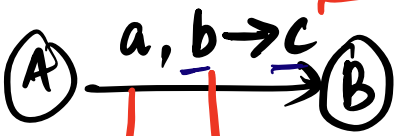
if $\gamma = \varepsilon$, then stack is popped

$\gamma = X$, then " is unchanged

$\gamma = YZ$, then X is replaced by Z & Y is pushed onto stack

PDA: Graphical Notation:

FSA: 

PDA: 

Pushed onto stack

'ε' means nothing is pushed.

Input symbol

↓

May be ε

something

Symbol on top of stack (this symbol is popped)

If 'ε' means that stack is neither popped nor pushed.

Example:

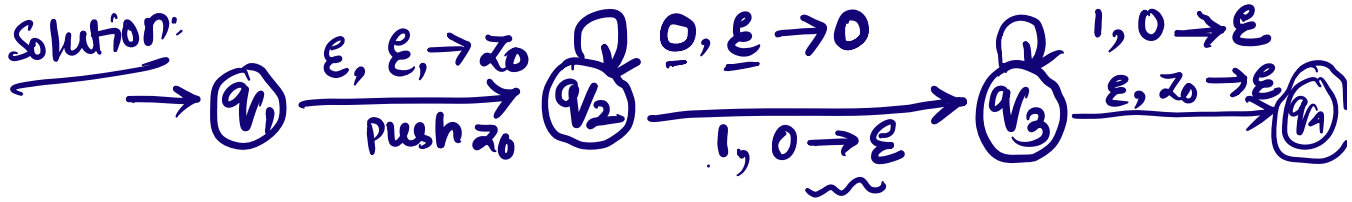
construct a PDA, that accepts

$L = \{0^n 1^n \mid n \geq 0\}$

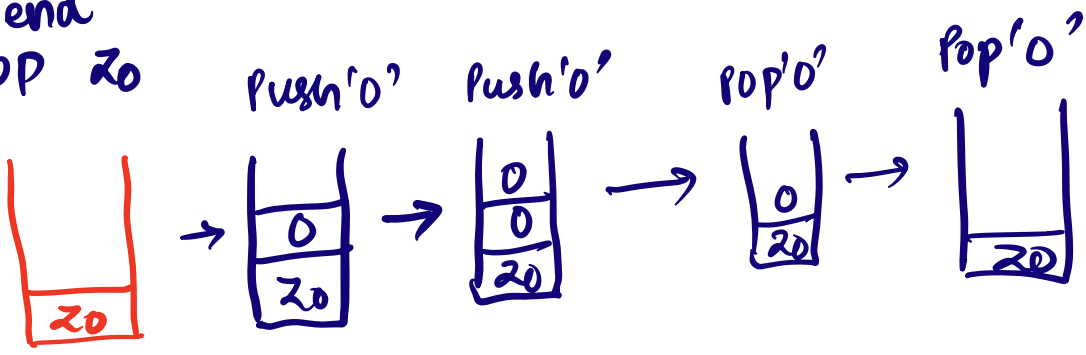
$n = 2 \Rightarrow 0^2 1^2$

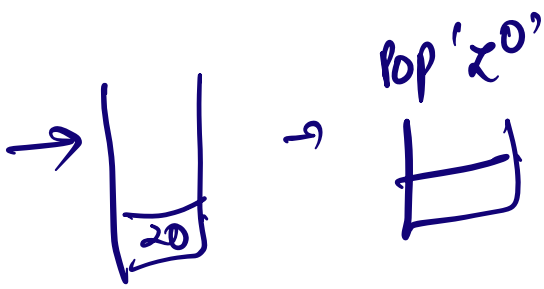
$\Rightarrow 00\ 11$

Solution:



Always
Push z_0
& at end
POP z_0





0 0 1 1

0 0

✗ ✗

string is accepted

i. If we reach a final state

ii. whenever stack is empt.

first symbol , second ⟶ third

- first symbol → Input
- second → top most symbol or symbol to be poped
- third → to be pushed.

Question: PDA that accepts '00001111'?

Same solution as the above as

$0000111 = 0^4 1^4 = 0^n 1^n$ where $n = 4$.

Example 2:

Construct a PDA that accepts even palindrome of form $L = \{ W W^R / W = (a+b)^+ \}$

Solution:

$W = (a+b)^+$

word = It cannot be empty, atleast 1 symbol has to be there.

Palindrome:

A word / sequence that reads the same backward or forward.

Example:

NOON

RACECAR

TACOCAT

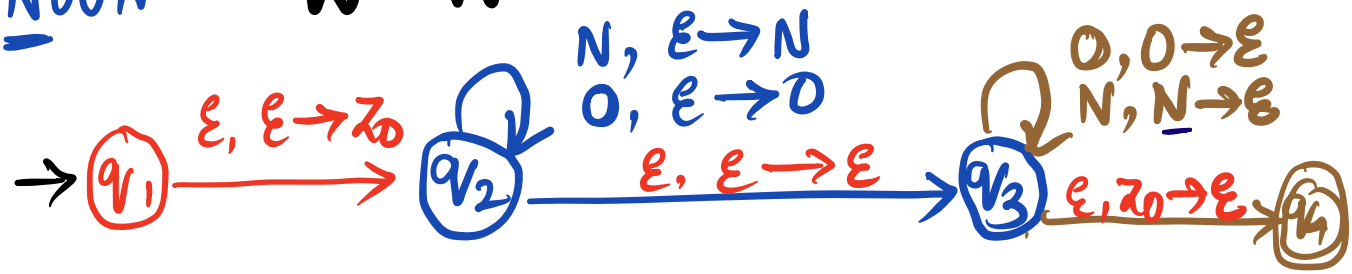
WOW

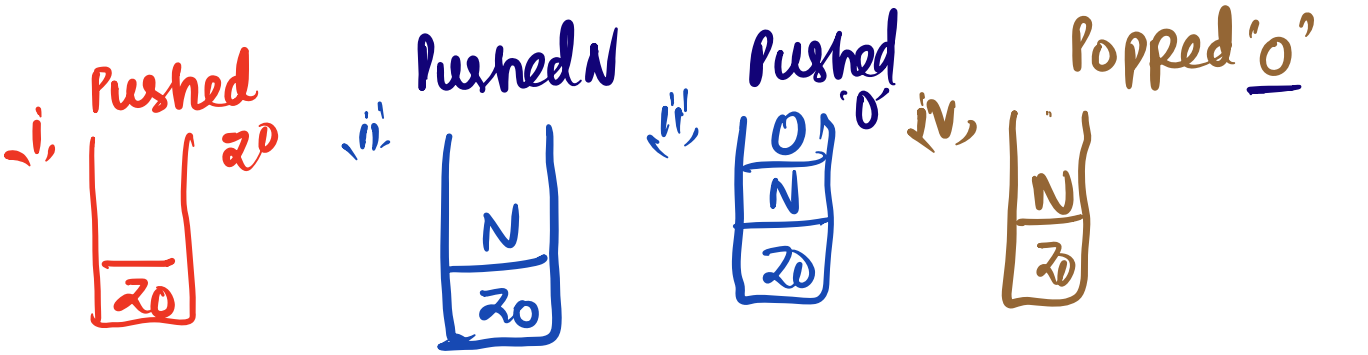
12321

NOON

N O O N
W W^R

W - word
W^R - Reverse of word.





O O 1 1

O, ε → O

pop/ top of stack

push

O / Z0

, 1, O → ε

push Nothing