CS & IT ENGINEERING

Theory of Computation

Push Down Automata

Lecture No. 01

By- DEVA Sir

01 Revision of Regulars

02 CFL

03

04

05

Regular Languages

1. Regular Expression
2. FA
3. RG : LLG / RLG
4. Regulars & Non Regulars
5. closure properties

Regular Expressions:

① $(a+b)^*$
$= (a^*+b^*)^*$
$= (a^*+b^*)^+$
$= (a^+ + b^+)^*$
$= (a+b)^+ + \varepsilon$
$= (a^*b^*)^*$
$= (a^*b^*)^+$
$= (b^*a^*)^*$
$= (b^*a^*)^+$
$= b^*(ab^*)^*$
$= a^*(ba^*)^*$
$= (b^*a)^*b^*$
$= (a^*b)^*a^*$

② aX

③ abX

④ $aaaX$

⑤ Xa

⑥ Xab

⑦ $Xaaa$

⑧ XaX

⑨ $XabX$

⑩ $Xaaa\,X$

⑪ $(a+b)^2 = x^2$

⑫ $(\varepsilon+a+b)^2$

⑬ $(a+b)^2\,X$

⑭ a^*b^*

⑮ b^*a^*

⑯ $(aa)^*$

⑰ $a(aa)^*$
$= (aa)^*a$

⑱ $b^*ab^*ab^*$

⑲ $b^*(a+\varepsilon)b^*(a+\varepsilon)b^*$

⑳ $XaXaX$

㉑ $[(a+b)^2]^*$

㉒ $(x^2)^*x$

㉓ $(b^*ab^*ab^*)^* + b^*$
$= b^*(b^*ab^*ab^*)^*$
$= (b^*ab^*ab^*)^*b^*$
$= b^*(b^*ab^*ab^*b^*)^*b^*$
$= (b^*ab^*a)^*b^*$
$= b^*(ab^*ab^*)^*$

㉔ $b^*(b^*ab^*ab^*)^*b^*ab^*$

㉕ $(b^*ab^*ab^*ab^*)^* + b^*$

㉖ $\phi^0 = \varepsilon$

㉗ $\phi^* = \varepsilon$

㉘ $aa^* = a^+$

㉙ a^*
$= (a^*)^*$
$= (a^*)^+$
$= (a^+)^*$
$= a^+ + \varepsilon$
$= (a^*)^{100}$

㉚ a^+
$= aa^*$
$= a^*a$
$= aa^*a^*$
$= a^*aa^*$
$= (a^*)^{100}a$
$= a^+a^*$
$= a^*a^+$

$X = (a+b)^*$

$x = (a+b)$

① $(a+b)^*$

$S \to aS \mid bS \mid \varepsilon$

$S \to Sa \mid Sb \mid \varepsilon$

② Reg Exp

DFA

NFA

LLG

RLG

③

④

⑤

30 Languages

Page 1 ⇒ Reg Exp

Page 2 & 3 ⇒ DFA
NFA

Page 4 & 5 ⇒ LLG
RLG

Page 6 & 7 ⇒ Reg & Non Reg

Page 8 ⇒ Closure properties

Max 10 Pages

Equivalence classes [Myhill-Nerode equivalence classes]

I) **No. of states in min DFA for Reg = No. of equivalence classes for Reg**

→ Find no. of equivalence class for $L = ab(a+b)^*$.

Regular lang.

= 4 equivalence classes

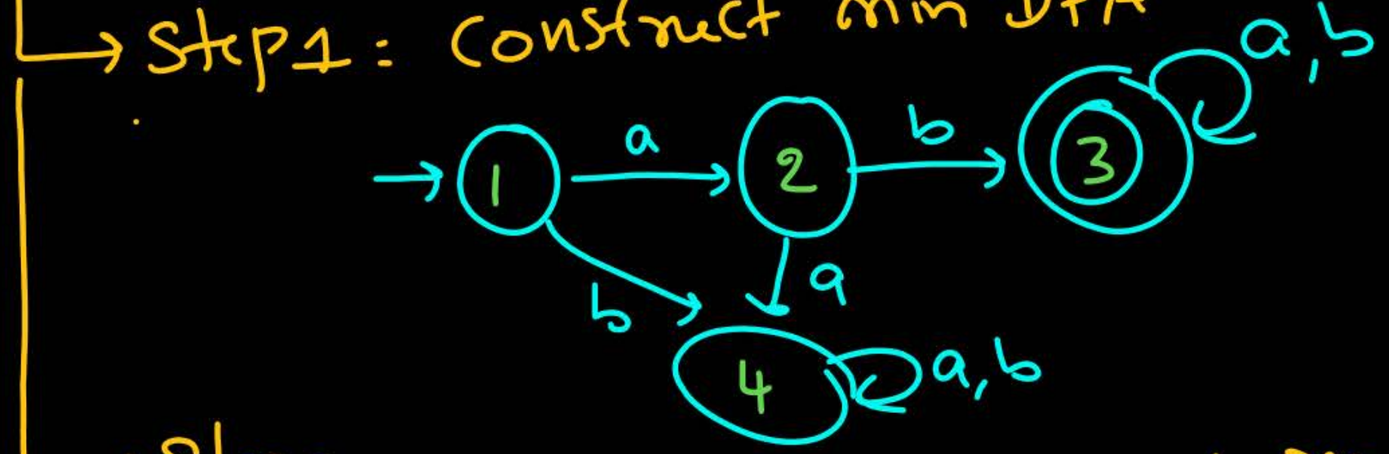
[1] = ε

[2] = a

[3] = $ab(a+b)^*$

[4] = $(b+aa)(a+b)^*$

→ Step 1 : Construct min DFA



→ Step 2 : Each state represent one equivalence class.

$[1] \cup [2] \cup [3] \cup [4] = (a+b)^*$

$[1] \cap [2] = \phi$

$[1] \cap [3] = \phi$

$[1] \cap [4] = \phi$

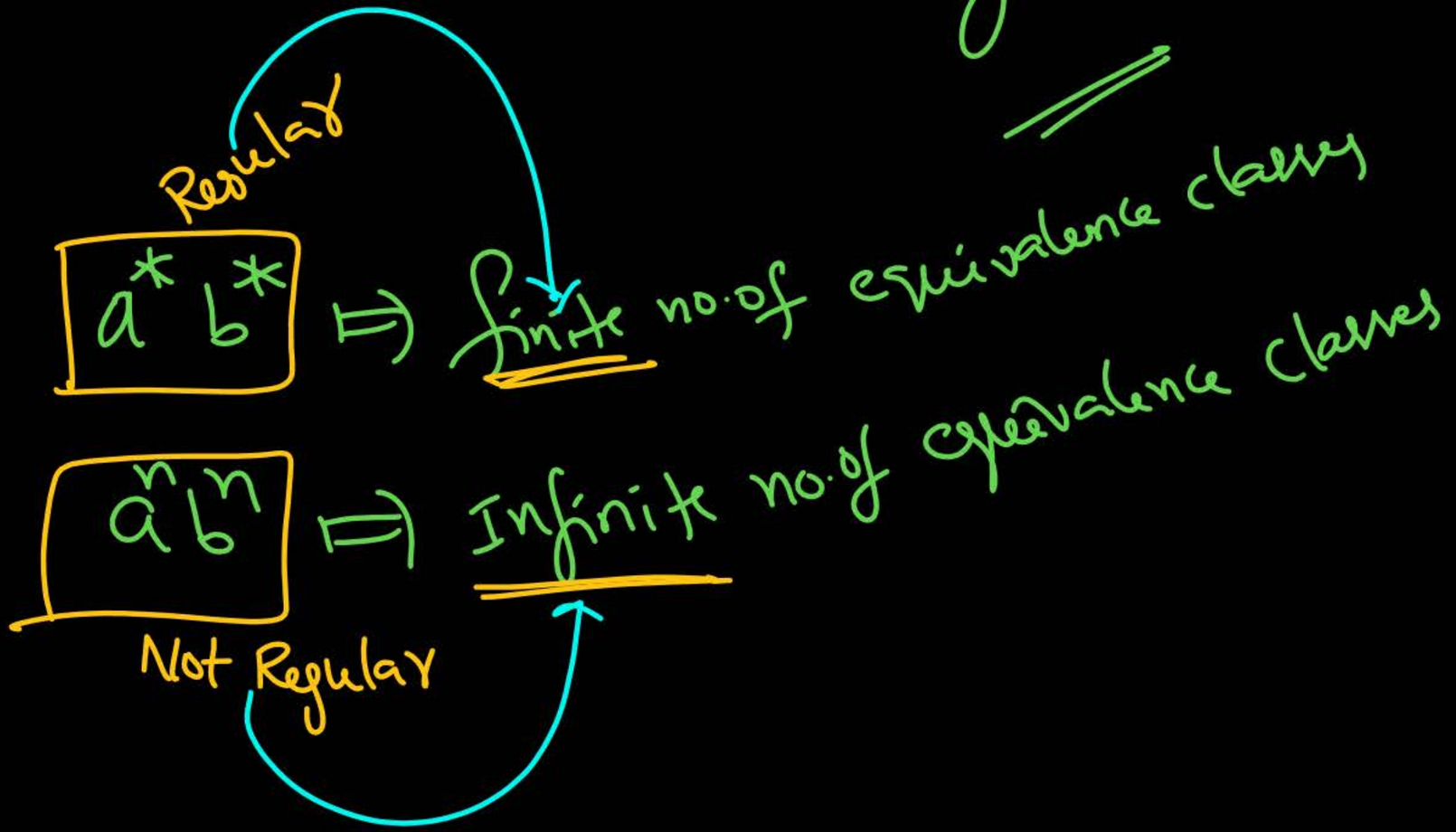
$[2] \cap [3] = \phi$

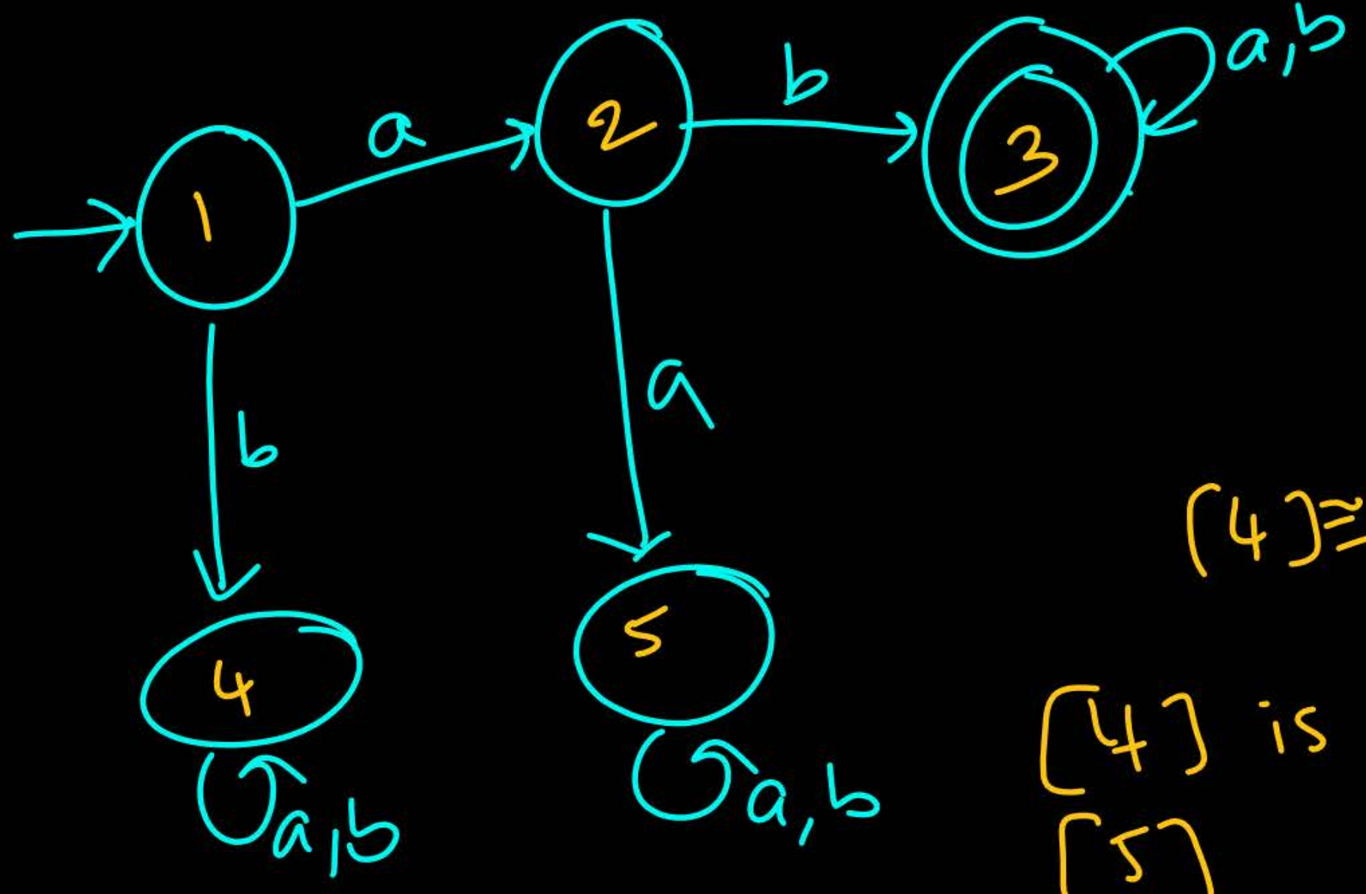
$[2] \cap [4] = \phi$

$[3] \cap [4] = \phi$

II) No. of equivalence classes for Non regular languages

= Infinite





$[4] \equiv [5] \equiv [45]$

[4] is not equivalence class

[5] " "

[45] is equivalence class

X and Y are 2 different equivalence classes

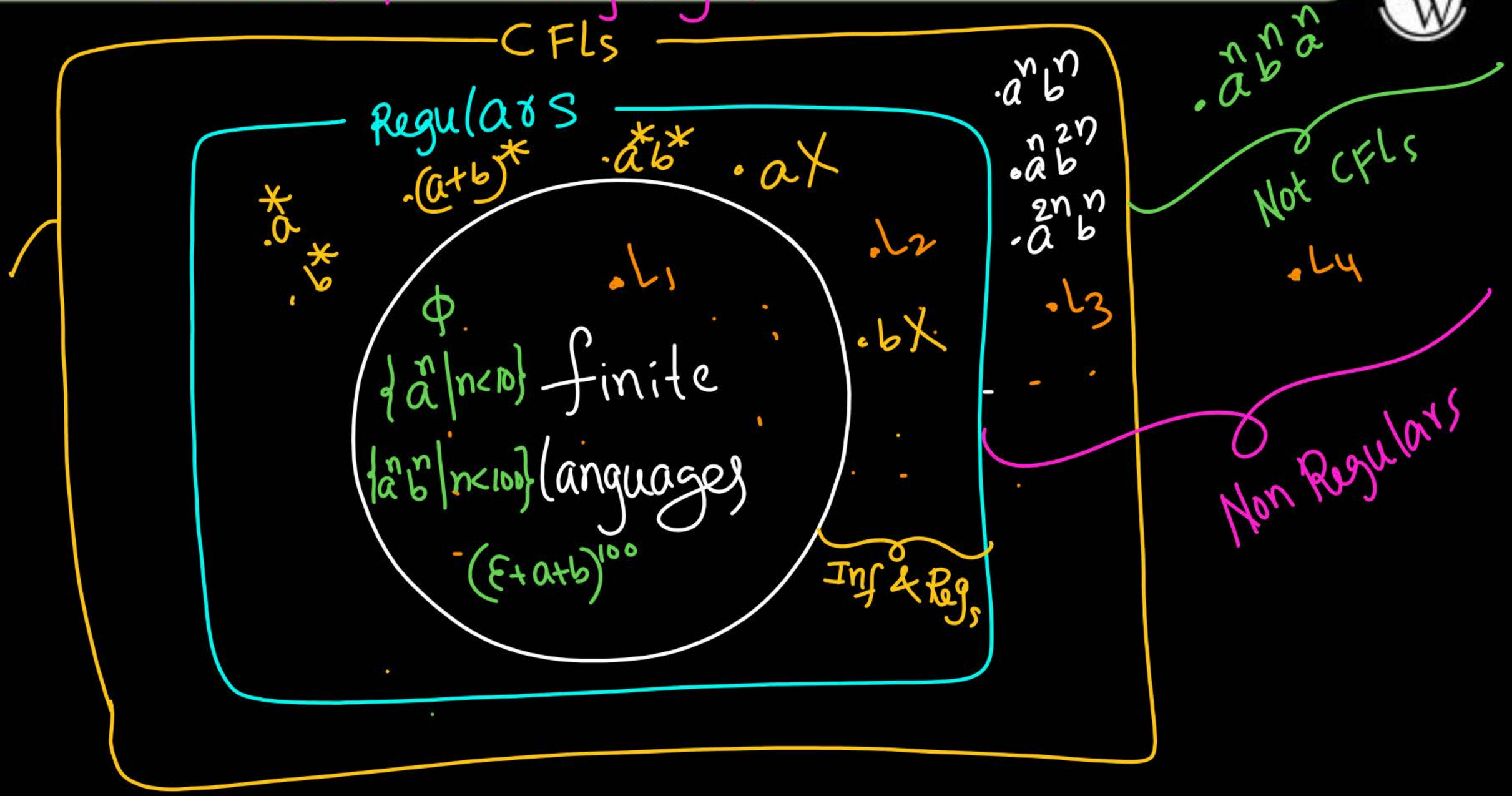
iff

Some string $\in \Sigma^*$ $\Rightarrow$ one of Xw and Yw is valid and other is invalid

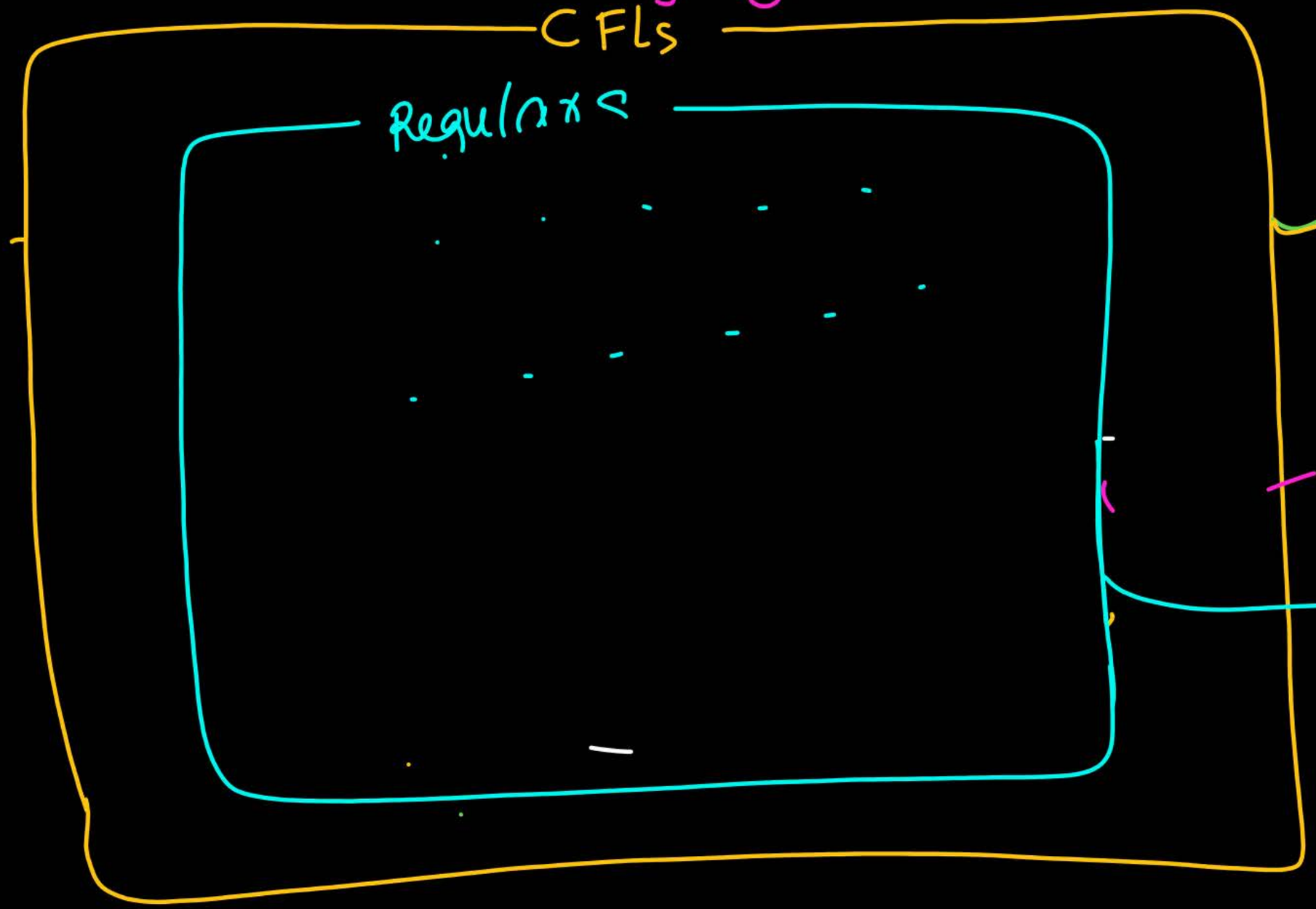
$\in L$ $\notin L$

Learn	Apply
With limitation	With no limitation

Context Free Languages :

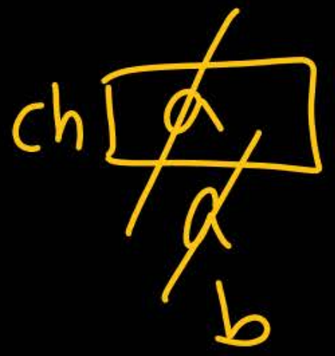


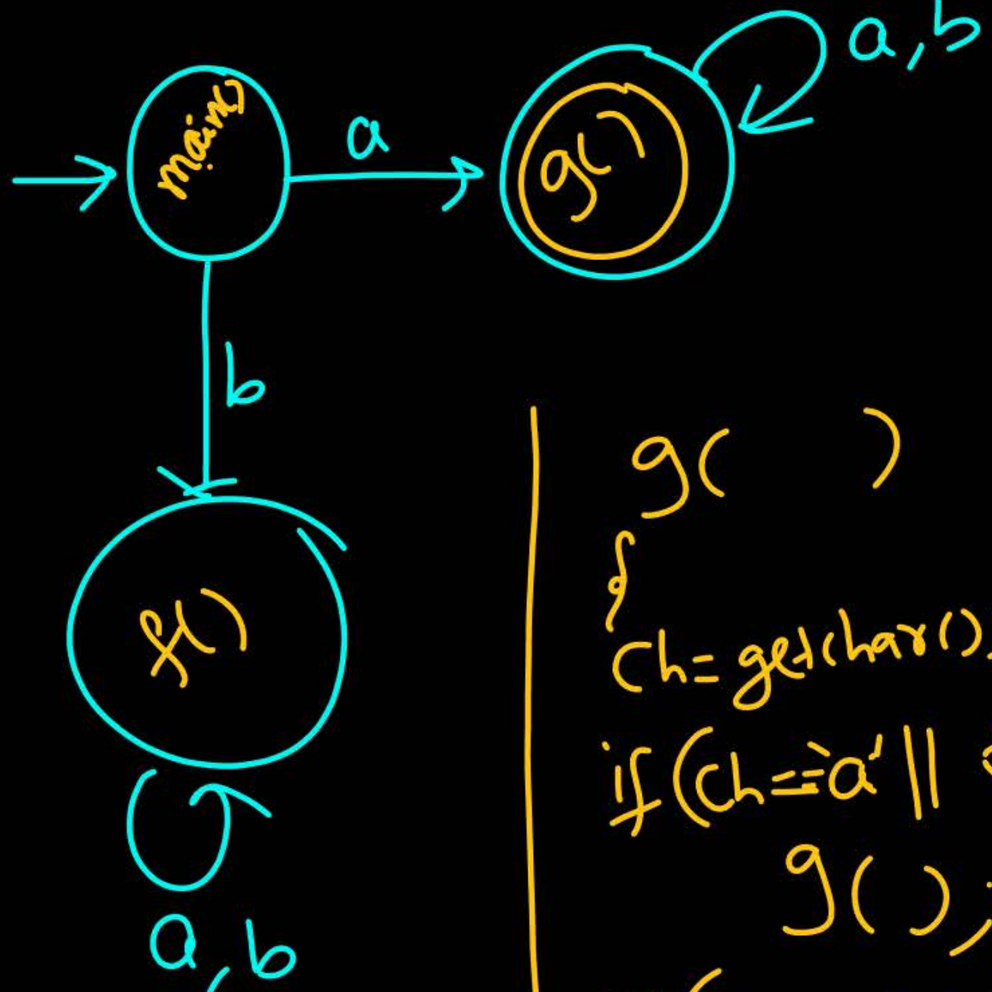
Context Free Languages :



Using 1 stack, we can solve any CFL.

O(1) Space
Every regular do not require memory to Solve



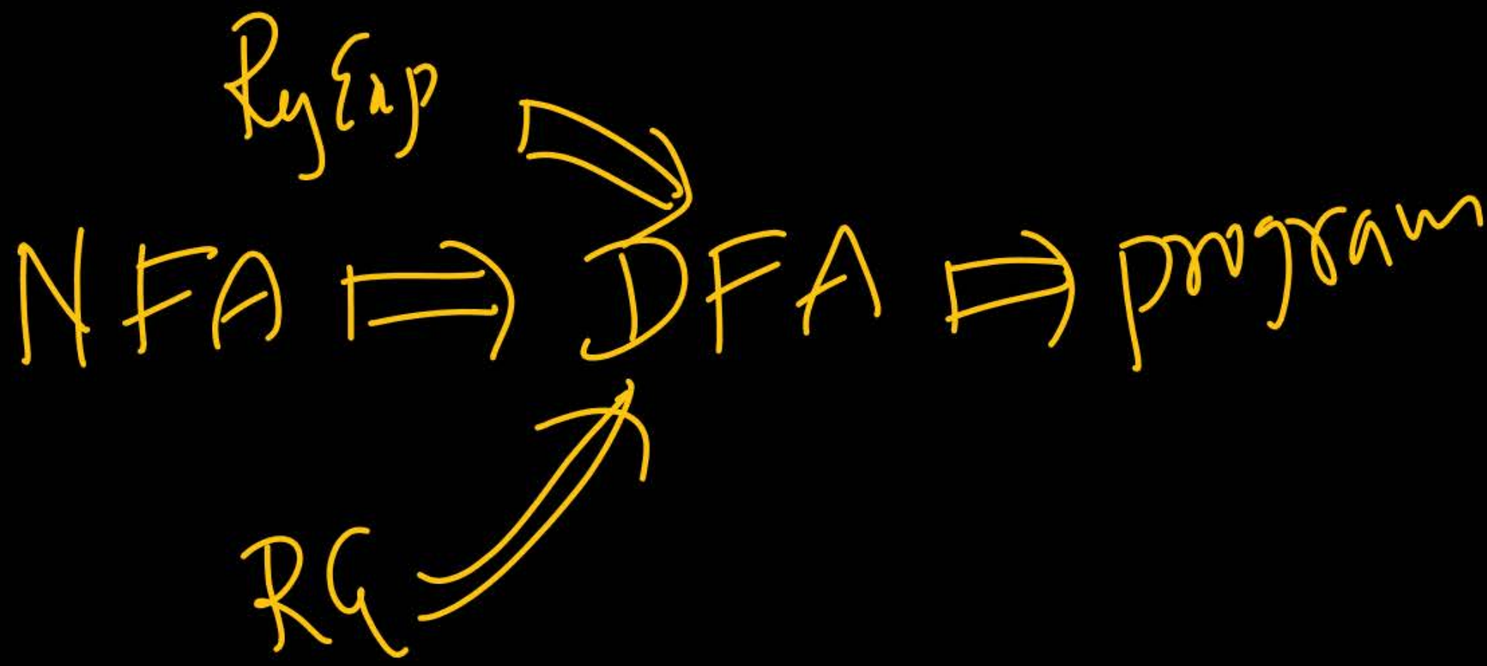


O(1)
Space

```
g( )
{
ch=getchar();
if(ch=='a' || ch=='b')
      g();
if(ch=='\n')
      "Accept";
}
```

char ch; Space

```
main( )
{
  ch=getchar( )
  if(ch=='a')
        g();
  if(ch=='b')
        f();
  if(ch=='\n')
        "Not Accept"
}
```


RyExp
NFA
DFA
program
RG

I) L_1 is finite language
(Reg) (CFL)

II) L_2 is Regular but not finite (Inf & Reg)
(CFL)

III) L_3 is CFL but not regular (CFL & not reg)

IV) L_4 is not CFL
(not reg)
(not finite)

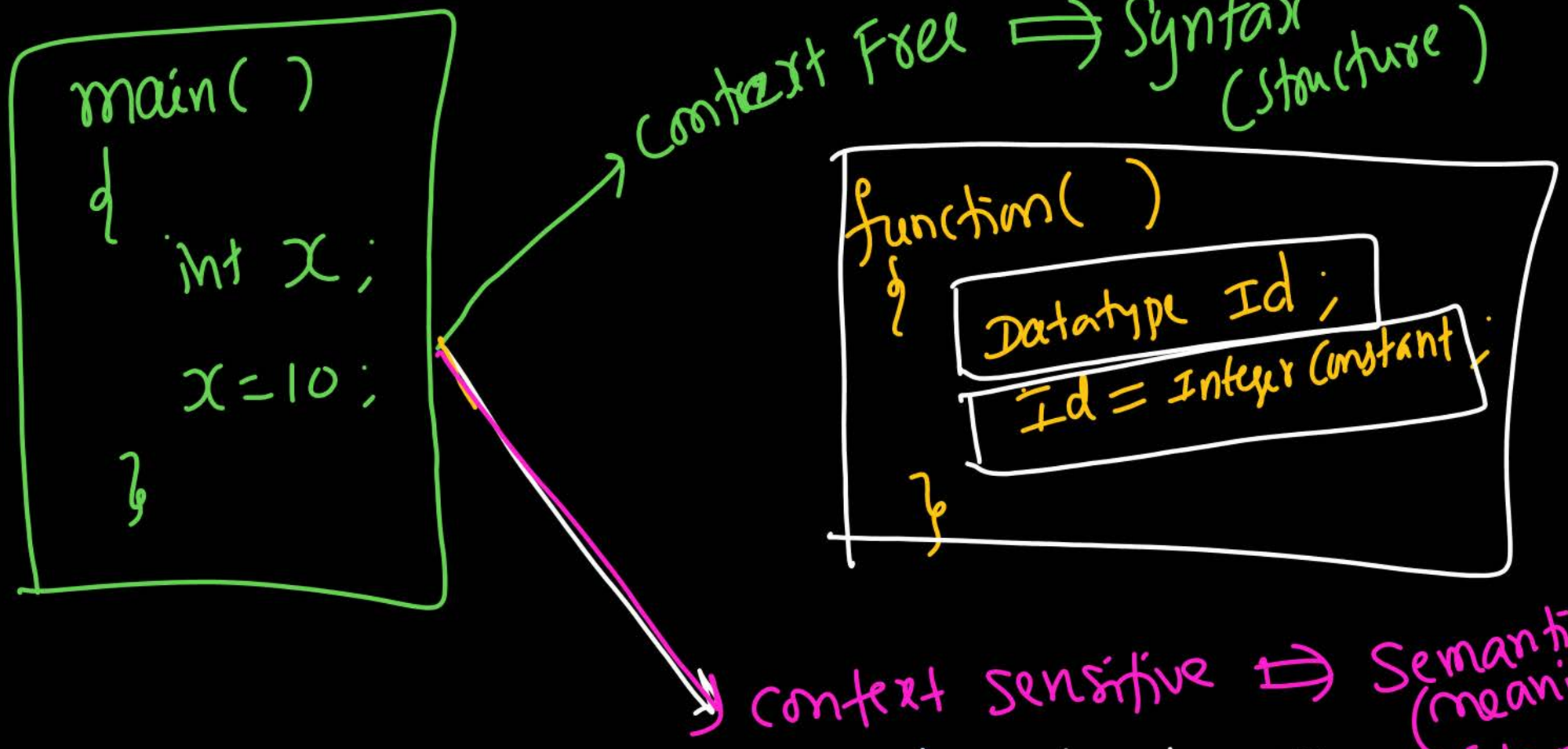
L is not reg & not CFL

L is not CFL

I) Every Finite Language is Regular.

II) Every Regular Language is CFL.

III) CFL may or may not regular.


main( )
{
int x;
x=10;
}
Context Free ⇒ Syntax (Structure)
function( )
{
Datatype Id;
Id = Integer Constant;
}
context sensitive ⇒ Semantic (meaning) (type)
checks type in every Statement

Regular Language

CFL

CSL

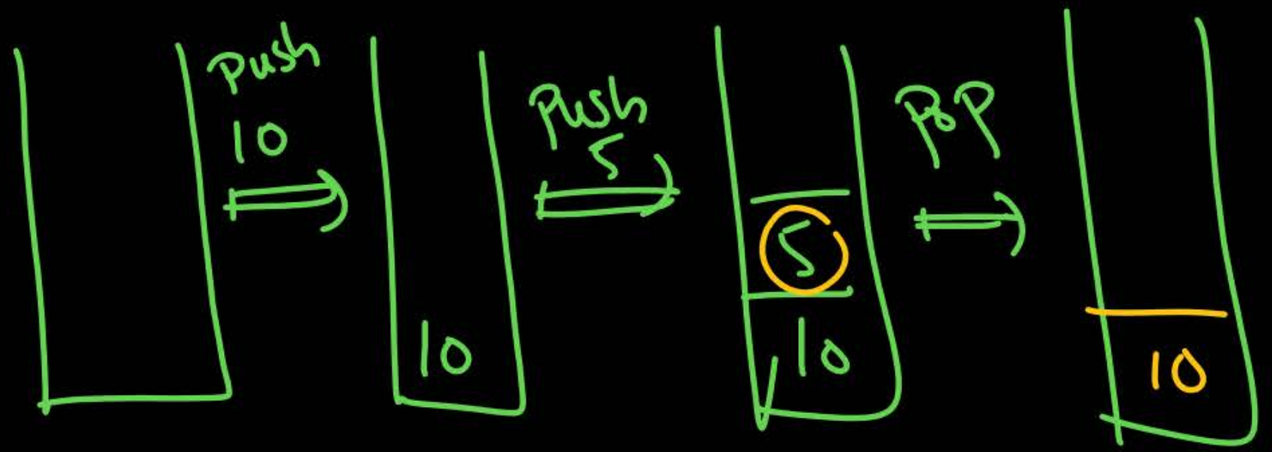
Words ⟹ Structure ⟹ meaning

Tokens ⟹ Syntax ⟹ Semantic

Words ⟹ Sentences ⟹ English language meaning

Stack [LIFO]

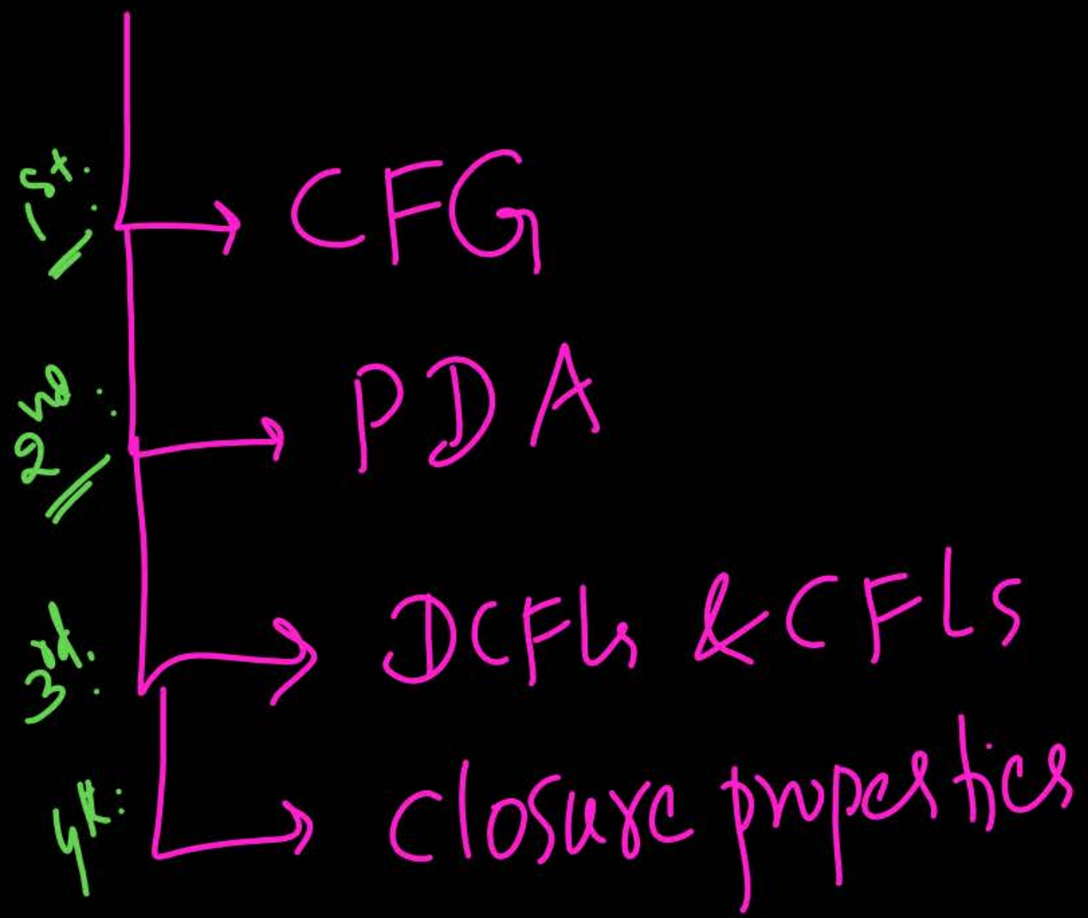
- It is data structure
- Push & Pop
- Insertion & Deletion can happen only at one end



CFL

- Every regular application
- Messages
- Call history
- Web pages history
- DFS
- Balanced parentheses
- palindromes

- Recursion
- Activation Record
- Calling Sequence (Call Tree)



11Am - 1Pm

morning

Sat & Sun

Summary

- Regulars Revision ✓
- CFLs ?


THANK YOU
GW
SOLDIERS !