Assignment-3

Context-Free Language (CFL) reachability problem

Problem Statement:

Given a **directed graph** G = (V, E) and a **context-free grammar** (CFG) in Chomsky normal form, the tool should determine whether there is a **path** from a given **source node** to **a target node** such that the sequence of edge labels along the path forms a **string derivable** from the CFG.

We will refer to this tool as **CFL-Reachability Solver**.

CFL-Reachability Solver should process the **graph** and the **grammar** to **determine reachability** based on the **context-free language** defined by the **CFG**.

Input

- A directed graph
- A context-free grammar (CFG) in Chomsky Normal Form (CNF), defining the language of valid paths.
- Source node and a target node.

Example:

  - S -> A B; A -> a; B -> BC| b; C -> c **# CFG**
  - {'v1': [('v2', 'a')], 'v2': [('v3', 'b')], 'v3': [('v4', 'c')]} **#Graph**
  - v1 **#Source node**
  - v4 **#Target node**

Output

YES or **NO**

YES if there exists a path from source node to target node such that the sequence of edge labels belongs to the language of the CFG;

otherwise, output **NO**

Example:

CFG:

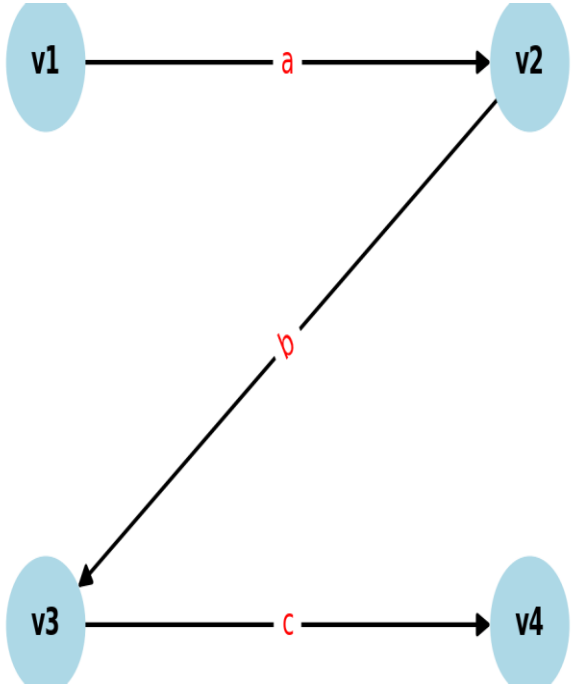
S -> A B; A -> a; B -> BC| b; C -> c

Graph:

{'v1': [('v2', 'a')], 'v2': [('v3', 'b')], 'v3': [('v4', 'c')]}

Source node: v1

Target node : v4



Output:

YES

How to solve ??

Given a CFL **A** and a string $\boldsymbol{x} \in \Sigma^*$

how do we tell whether $\boldsymbol{x}$ is in A?

Cocke–Kasami–Younger (CKY) Algorithm

The **CKY Algorithm** is a **dynamic programming algorithm** used to determine whether a given string can be generated by a **context-free grammar (CFG)**.

Input

A string x = $x_1x_2...x_n$ and a CFG in CNF.

Output

A parsing table that determines if **x** is in the language.

Steps

- Initialize a table where **entry (i, j)** represents the non-terminals that derive substring $x_i...x_j$.
- Fill in the table **diagonally** using dynamic programming.
- If the start symbol appears in the **cell(Bottom-Left),** the string is in the language.

Consider the following grammar

S->AB|BA|SS|AC|BD

A->a

B->b

C->SB

D->SA

Language generated by grammar ???

The set of all **non-null strings** with **equally many a's and b's**

Input:

Grammar:

S->AB|BA|SS|AC|BD

A->a

B->b

C->SB

D->SA

String:

x=|a|b|

0 1 2

Output:

0		
A	1	
S	B	2

- For **each character** of the string, the algorithm populates the table with the **non-terminal symb** car that character based on the production rules. $A \to a$ $B \to b$
- The algorithm iteratively **fills out the table** for substrings of **increasing length**, checking if there exists a split of the substring into two parts, and if so, whether the **non-terminals from the split parts** can be combined to **form the sub-string**.
- This process is repeated for substrings of length 2, then length 3, and so on, up to the length of the entire string.

$$x_{0,3} = x_{0,1}x_{1,3} = x_{0,2}x_{2,3} \qquad x_{1,4} = x_{1,2}x_{2,4} = x_{1,3}x_{3,4}$$

How to check string is derived from Grammar?

- At the end of the process, if the **start symbol** (S) appears in the **table entry(Bottom-left)** corresponding to the entire string, then the **string can be derived from the grammar**.
- If the start symbol does not appear, then the string cannot be derived.

Grammar:

S->AB|BA|SS|AC|BD

A->a

B->b

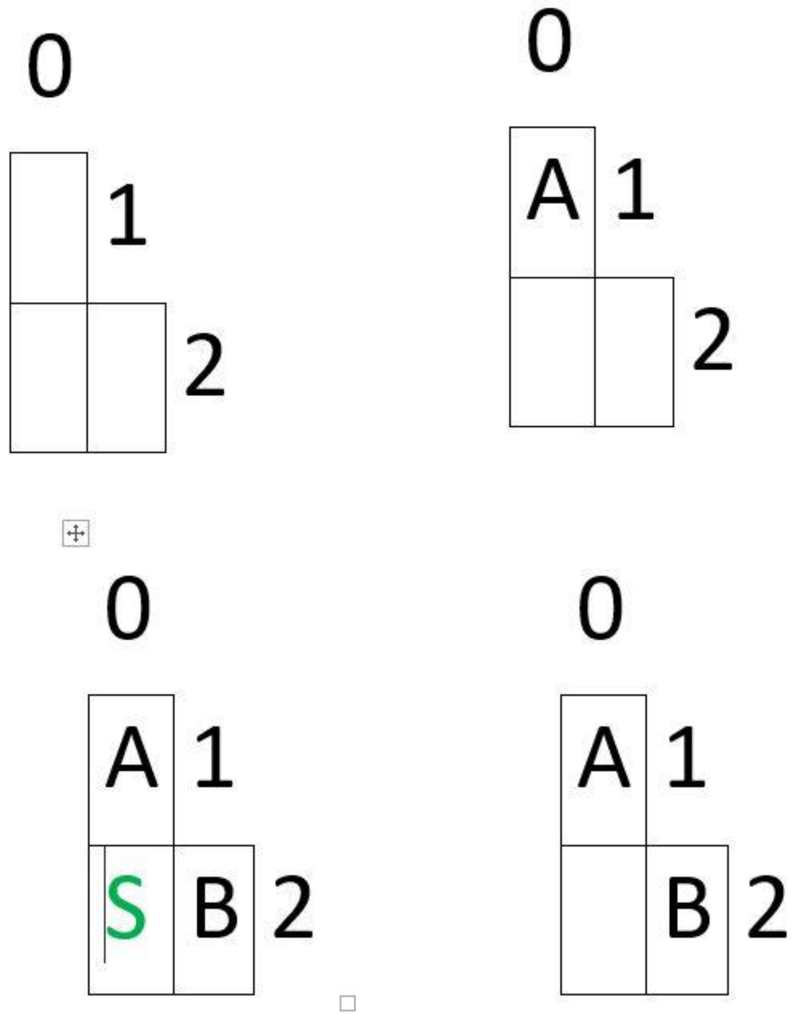
C->SB

D->SA

String:

x=|a|b|

0 1 2



Grammar:

S->AB|BA|SS|AC|BD

A->a

B->b

C->SB

D->SA

String:

x=|a|b|b|a|

0 1 2 3 4

0

A				1
S	B			2
C	Ø	B		3
S	Ø	S	A	4

0

				1
				2
				3
				4

0

A				1
	B			2
		B		3
			A	4

0

A				1
S	B			2
C	Ø	B		3
	Ø	S	A	4

0

A				1
S	B			2
	Ø	B		3
		S	A	4

Grammar:

S->AB|BA|SS|AC|BD

A->a

B->b

C->SB

D->SA

String:

x=|a|a|b|b|a|b|

0 1 2 3 4 5 6

0						
A	1					
∅	A	2				
∅	S	B	3			
S	C	∅	B	4		
D	S	∅	S	A	5	
S	C	∅	C	S	B	6