Algorithme de Cocke-Younger-Kasami

Source : https://fr.wikipedia.org/wiki/Algorithme_de_Cocke-Younger-Kasami

En informatique théorique et en théorie des langages, l'**algorithme de Cocke-Younger-Kasami** (**CYK**) est un algorithme d'analyse syntaxique pour les grammaires non contextuelles, publié par Itiroo Sakai en 1961. Il permet de déterminer si un mot est engendré par une grammaire, et si oui, d'en donner un arbre syntaxique. L'algorithme est nommé d'après les trois personnes qui l'ont redécouvert indépendamment, J. Cocke, dont l'article n'a jamais été publié, D. H. Younger et T. Kasami qui a publié un rapport interne aux US-AirForce.

L'algorithme opère par analyse ascendante et emploie la programmation dynamique. L'algorithme suppose que la grammaire est en forme normale de Chomsky. Cette restriction n'est pas gênante dans la mesure où toute grammaire non contextuelle admet une grammaire en forme normale de Chomsky équivalente.

Principe

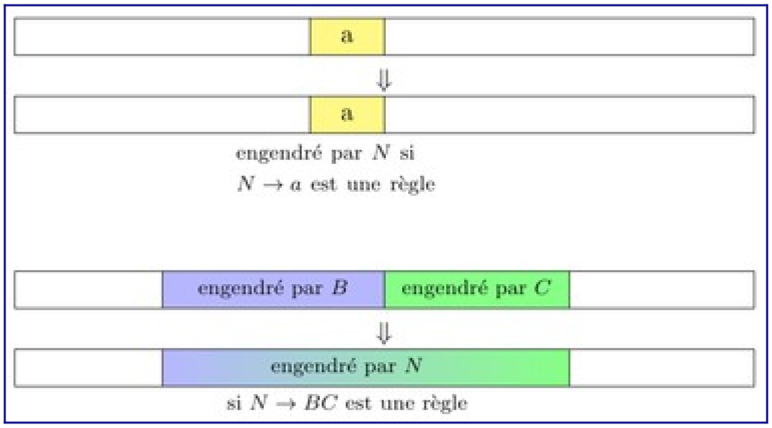
Sans perte de généralité, on suppose que la grammaire **G** n'engendre pas le mot vide **ε.** Ainsi, on peut supposer que la grammaire **G** est sous forme normale de Chomsky et qu'elle ne contient pas de règles de la forme **N** → **ε** (on parle de grammaire propre, grammaire non contextuelle).

Soit **m** un mot non vide à analyser. L'algorithme emploie la programmation dynamique. Les sous-problèmes sont les suivants : **P[i,j]** est l'ensemble des non-terminaux qui engendrent le mot **m[i..j]** pour tout **i, j** tels que **1** ≤ **i** ≤ **j** ≤ **|m|** où **|m|** est la longueur du mot .

On peut calculer les ensembles **P[i,j]** par récurrence sur **|j-i|**.

- Cas de base : **P[i,j]** est l'ensemble des non-terminaux **N** tel que **N** → **m[i]** est une règle de la grammaire.
- Cas récursif : Si **i** < **j**, **P[i,j]** est l'ensemble des non-terminaux **N** tels qu'il existe une règle **N** → **BC** où **B** et **C** sont des non-terminaux et un entier **k** ∈ **{i, …, j-1}** tels que **B** est dans **P[i,k]** et **C** est dans **P[k+1,j]**.

La figure montre le cas de base et le cas récursif.



On en déduit un algorithme de programmation dynamique qui calcule tous les ensembles **P[i,j].** Le mot **m** est engendré par la grammaire si et seulement si **S** est dans **P[1, |m|]** où **S** est l'axiome de la grammaire et **|m|** est la longueur du mot .

Exemple

Considérons la grammaire suivante en forme normale de Chomsky :

$$\begin{array}{lcl} S & \rightarrow & GN\,GV \\ GV & \rightarrow & GV\;C \\ GV & \rightarrow & V\;GN \\ GV & \rightarrow & mange \\ C & \rightarrow & P\;GN \\ GN & \rightarrow & Det\;N \\ GN & \rightarrow & elle \\ V & \rightarrow & mange \\ P & \rightarrow & avec \\ N & \rightarrow & poisson \\ N & \rightarrow & fourchette \\ Det & \rightarrow & du \\ Det & \rightarrow & une \end{array}.$$

où l'ensemble des non-terminaux est **{S, GV, C, GN, V, P, Det}** et l'ensemble des terminaux (lettres) est **{elle, posson, fourchette, mange, du, avec, une}**. Ici, « elle » s'appelle une lettre (bien que c'est un mot) et une phrase comme « elle mange du poisson avec une fourchette » s'appelle un mot.

Maintenant, analysons le mot **m** qui est la phrase « elle mange du poisson avec une fourchette » avec l'algorithme CYK. Dans la table suivante, on indique les valeurs de **P[i,j]**:

P[1, 7] = {S}						
P[1, 6] = ø	P[2, 7] = {GV}					
P[1, 5] = ø	P[2, 6] = ø	P[3, 7] = ø				
P[1, 4] = S	P[2, 5] = ø	P[3, 6] = ø	P[4, 7] = ø			
P[1, 3] = ø	P[2, 4] = {GV}	P[3, 5] = ø	P[4, 6] = ø	P[5, 7] = {C}		
P[1, 2] = {S}	P[2, 3] = ø	P[3, 4] = {GN}	P[4, 5] = ø	P[5, 6] = ø	P[6, 7] = {GN}	
P[1, 1] = {GN}	P[2, 2] = {GV}	P[3, 3] = {Det}	P[4, 4] = {N}	P[5, 5] = {P}	P[6, 6] = {Det}	P[7, 7] = {N}
elle	mange	du	poisson	avec	une	fourchette

Le mot « elle mange du poisson avec une fourchette » est reconnu car l'axiome S est dans P[1,7].

Pseudo-code

Voici un pseudo-code inspiré de l'analyse de la section précédente :

```
Pour i = 1 à |m|
    P[i,i] := ensemble des non-terminaux N tel que N → m[i] est une règle de la grammaire
Pour d = 1 à |m| − 1
    Pour i = 1 à |m|-d
         j := i+d
         P[i,j] := ensemble des non-terminaux N tels qu'il existe une règle N → BC et un entier
k ∈ {i,...,j − 1}. tels que
                                B est dans P[i,k] et C est dans P[k + 1,j]

Retourne oui si S est dans P[1,|m|] ; non sinon.
```