Problema B - Teoria da Computação 2022/2023

Algoritmo CYK (de Cocke, Younger e Kasami)

Problema

O objetivo deste exercício é a implementação de um algoritmo polinomial de reconhecimento de palavras por uma gramática algébrica na sua forma normal de Chomsky. Este algoritmo foi desenvolvido independentemente por J. Cocke, D. H. Younger e T. Kasami, em 1965 e permite saber se uma palavra w pertence à linguagem gerada por uma gramática algébrica em forma normal de Chomsky G.

Este algoritmo pertence a família dos algoritmos dinâmicos (dinâmica de *programação dinâmica*) e tem uma complexidade de $\mathcal{O}(|w|^3)$.

Este realiza uma análise da palavra de forma ascendente (bottom-up). O algoritmo começa com a palavra por analisar e tente deduzir que produção contribui diretamente a geração desta última. O resultado é uma matriz triangular. Se o símbolo inicial está na célula de topo da matriz então a palavra foi reconhecida, senão a palavra não pertence a linguagem gerada pela gramática em forma normal de Chomsky.

O algoritmo é descrito com todo o rigor e detalhe nos apontamentos das aulas teóricas e existem inúmeras fontes de informação na web e na bibliografia indicada nas aulas. Pelo que este enunciado não apresenta o algoritmo mas sim dois exemplos elucidativos, um simples, e outro mais geral. O leitor é então fortemente aconselhado a consultar os apontamentos para ter os detalhes todos do algoritmo. Assim sendo:

Seja $G = \{\{S, A\}, \{a, b\}, P, S\}$ com $P = \{S \to AA, S \to AS, S \to b, A \to AS, A \to SA, A \to a\}$

Será que $abaab$ a linguagem gerada por G?

O algoritmo começa por inicializar uma matriz triangular M de base de tamanho $|abaab| = 5$ que será preenchida de baixo para cima com não-terminais:

a	b	a	a	b

A linha $\boxed{a \mid b \mid a \mid a \mid b}$ vai servir de base a inicialização do processo de preenchimento da matriz. Esta inicialização consiste em preencher a ultima linha com os não terminais que geram os terminais situados imediatamente por baixo.

De reparar que cada célula da matriz contém um conjunto (lista) de não terminais.

Obtemos então a matriz seguinte:

A	S	A	A	S
a	b	a	a	b

Repare que se vários não terminais gerassem a então estes estariam todos nas células por cima das células contendo a.

O processo de preenchimento funciona da linha $i = 4$ até a linha 1. Para preencher uma célula $M[i, j]$ o algoritmo vai tentar utilizar os não terminais presentes nas linhas anteriores e nas colunas a direita (ou seja os não terminais presentes em $M[k, l]$ com $i < k \leq 5$ e $j \leq l \leq 5$).

Um não terminal N é colocado na matriz em $M[i, j]$ a custa de dois outros não terminais P e Q se

- $N \to P\ Q$ é uma produção.
- P está a esquerda de Q e ambos estão ”por baixo” e a direita de N.

Dada uma linha i, o processo vare a linha toda. Repare que este processo pode não poder preencher algumas células, estas ficando assim vazias.

Se repetimos este processo de baixo para cima até chegar ao topo da matriz então podemos determinar se a palavra analisada pertence a linguagem. Para tal basta ver se S está na célula $M[1, 1]$.

Vejamos o processo linha a linha:

A, S	A	S	A, S	
A	S	A	A	S
a	b	a	a	b

$M[4, 4] = \{A, S\}$ porque $M[5, 4] = A$, $M[5, 5] = S$, $A \to AS$ e $S \to AS$ são produções.
$M[4, 3] = S$ porque temos $M[5, 3] = A$ e $M[5, 4] = A$ e que $S \to AA$ é produção.
$M[4, 2] = A$ porque $M[5, 2] = A$, $M[5, 3] = A$, $A \to AA$ é produção.
$M[4, 1] = \{A, S\}$ porque $M[5, 1] = A$, $M[5, 2] = S$, $A \to AS$ e $S \to AS$ são produções.
As linhas seguintes preenchem-se de forma similar.

A, S	A, S	A		
A, S	A	S	A, S	
A	S	A	A	S
a	b	a	a	b

A, S	A, S			
A, S	A, S	A		
A, S	A	S	A, S	
A	S	A	A	S
a	b	a	a	b

A, S				
A, S	A, S			
A, S	A, S	A		
A, S	A	S	A, S	
A	S	A	A	S
a	b	a	a	b

Neste ponto ja podemos concluir: *abaab* pertence a linguagem gerada por G.

Este exemplo é simples porque todas as células da matriz foram preenchidas. Isto nem sempre acontece, como o podemos ver no exemplo seguinte.

Seja $G' = \{\{S, A\}, \{a, b\}, P, S\}$ com $P = \{S \to A\ B,\ S \to b,\ A \to a,\ B \to S\ A\}$

S				
	$\boxed{B}$			
	$\boxed{S}$			
		B		
A	A	S	A	$\boxed{A}$
a	a	b	a	a

De reparar que neste exemplo não se consegue preencher todas as células da matriz. No entanto as células preenchidas levam mesmo assim a que se coloque o símbolo inicial no topo da matriz. A palavra é assim aceite. Na linha 4 só se consegue colocar o B graças a regra $B \to S\ A$. De seguida só se consegue colocar o S na linha 3 porque se tem a regra $S \to A\ B$. Neste caso o A provem de $M[5, 2]$ e B de $M[4, 3]$. O caso realçado na matriz corresponde ao caso da aplicação da regra $B \to S\ A$ com A retirado de $M[5, 5]$. Finalmente consegue-se colocar S em $M[1, 1]$ graças ao A de $M[5, 1]$ e o B de $M[2, 2]$.

Input

Na primeira linha é introduzida a palavra por reconhecer. Esta palavra só é constituída por caracteres do alfabeto $\Sigma = \{a, \ldots, z\}$ e tem por comprimento máximo 50 caracteres. Assume-se que o conjunto de não-terminais é $N = \{A, B, \ldots S, \ldots, Z\}$ em que se destaca o símbolo S que assumiremos como sendo sempre o símbolo inicial.

A segunda linha apresenta o número m de regras de produções da gramática considerada. As restantes m linhas introduzem as m regras de produção. Cada uma destas produções tem o formato seguinte `N -> a1 a2 a3 ... an` com N não terminal e $a_i \in (N \cup \Sigma)$ e cada a_i é separado do a_j seguinte por um espaço único.

Output

O output é constituído por uma linha contendo:

- A palavra "YES" se a palavra é gerada pela gramática.
- A palavra "'NO" se a palavra não é gerada (reconhecida) pela gramática

Seguida da matriz triangular inferior, usada para reconhecer (ou não) a palavra.
NOTAS:

- cada célula da matriz é separada da célula adjacente à direita por 2 tabs (`\t\t`).
- Sempre que uma célula contém mais do que um não terminal, estes são separados por um só espaço, e estes estão ordenados por ordem alfabética.

Sample Input 1

```
abbabba
7
S -> S F
S -> a
A -> C C
A -> S S
A -> C S
C -> b
F -> A S
```

Sample Output 1

```
YES
S
    F
        A
S             S
    F             F
    A     A         A     A
S     C     C     S     C     C     S
a     b     b     a     b     b     a
```

Sample Input 2

```
aaabbabaaaabba
8
S -> S F
S -> a
A -> C G
A -> S S
A -> C S
C -> b
F -> A S
G -> C A
```

Sample Output 2

```
NO
```

```
F
    A
        S
            F
            A
            G     F G     F
                A     A     A
                    S         S
F             G             F G     F     F             G
A     A             A         A     A     A     A             A
S     S     S     C     C     S     C     S     S     S     S     C     C     S
a     a     a     b     b     a     b     a     a     a     a     b     b     a
```