Capítulo 7

Gramáticas Livres de Contexto (CFG)

As Gramáticas Livres de Contexto (ou CFGs, do inglês *Context-Free Grammars*) são uma das formas de representação de Linguagens Livres de Contexto (ou CFLs, do inglês *Context-Free Languages*), sendo a outra forma autómatos de pilha, abordados no próximo capítulo. As CFGs permitem especificar linguagens que as expressões regulares e autómatos vistos até agora não permitem, como por exemplo as que foram vistas no final do último capítulo, nas provas de não regularidade.

As gramáticas livres de contexto podem ser vistas como um conjunto de regras, ou produções, no formato H $\rightarrow$ B, em que H (a cabeça da regra) representa uma variável, e B (o corpo da regra) representa um conjunto de terminais (símbolos da linguagem) e não-terminais (variáveis) pelos quais a cabeça pode ser substituída.

Começando pela variável de arranque da gramática, podem fazer-se substituições de cada variável por uma das suas produções; as cadeias obtidas constituídas apenas por terminais são aquelas que constituem a linguagem.

Vamos ver um exemplo da gramática para a linguagem L = $\{vv^R : v \in \{0,1\}^* \}$, que vimos já no capítulo anterior não ser uma linguagem regular.

A gramática pode ser constituída por apenas uma variável, com três produções, duas capazes de, recursivamente, produzir as cadeias desejadas, e uma que funciona como base de recursão:

$S \rightarrow 0S0 \mid 1S1 \mid \varepsilon$

A definição formal de uma gramática é feita com um tuplo de 4 elementos G = (V, T, P, S), em que V representa o conjunto de símbolos não-terminais (variáveis), T o conjunto de símbolos terminais (os símbolos do alfabeto), P o conjunto de produções e S $\in$ V a variável de arranque da gramática. Para a gramática apresentada, a sua definição formal será G = ({S}, {0, 1}, P, S),

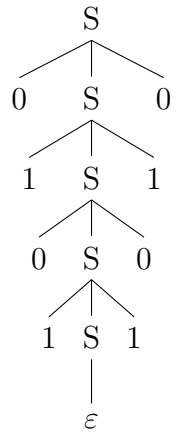
em que P representa as produções acima.

Se começarmos com o símbolo S, podemos produzir a cadeia vazia (ε), ou cadeias que iniciam e terminam no mesmo símbolo, e que no meio podem voltar a gerar novas cadeias nesse formato.

Para obter, por exemplo, a cadeia 01011010, começamos pela variável de arranque da gramática, S, substituindo a cada passo de derivação uma variável pelo corpo de uma das suas produções. Assim, para obter a cadeia desejada, temos:

$S \Rightarrow 0S0 \Rightarrow 01S10 \Rightarrow 010S010 \Rightarrow 0101S1010 \Rightarrow 0101\varepsilon1010$

Esta derivação pode também ser representada pela seguinte árvore de análise, em que os nós internos representam as variáveis da linguagem, e as folhas os terminais (cada expansão de um nó interno nos seus filhos corresponde a um passo da derivação, com os filhos sendo os símbolos presentes na produção, por ordem, da esquerda para a direita):



A colheita da árvore (leitura das folhas da esquerda para a direita) produz a cadeia que a árvore representa.

As derivações podem ainda ser classificadas como derivação mais à esquerda (*leftmost*) ou mais à direita (*rightmost*). Numa derivação mais à esquerda, a cada passo de derivação, é sempre escolhida para substituição a variável mais à esquerda na cadeia intermédia. Por outro lado, numa derivação mais à direita, em cada passo de derivação é escolhida sempre a variável mais à direita na cadeia intermédia. O resultado final é o mesmo (a cadeia pretendida), sendo uma distinção apenas no método de a obter. No exemplo apresentado, ambas as derivação mais à esquerda e mais à direita teriam o mesmo aspeto, já que existe sempre apenas uma variável a substituir.

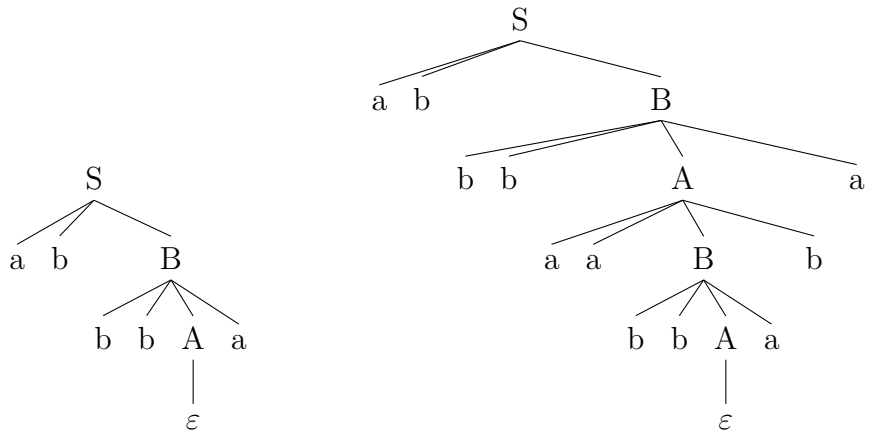
Exercício 1

Considere a seguinte gramática, em que S é a variável de arranque, e apresente uma derivação mais à esquerda para as cadeias *abbba* e *abbbaabbaba*, assim como as respetivas árvores de análise.

$S \to abB$
$A \to aaBb \mid \varepsilon$
$B \to bbAa$

Iniciando por S, podemos fazer a seguinte derivação para a cadeia *abbba*:
$S \underset{lm}{\Longrightarrow} abB \underset{lm}{\Longrightarrow} abbbAa \underset{lm}{\Longrightarrow} abbb\varepsilon a$

Para a cadeia *abbbaabbaba*, temos a seguinte derivação:
$S \underset{lm}{\Longrightarrow} abB \underset{lm}{\Longrightarrow} abbbAa \underset{lm}{\Longrightarrow} abbbaaBba \underset{lm}{\Longrightarrow} abbbaabbAaba \underset{lm}{\Longrightarrow} abbbaabb\varepsilon aba$
Podemos agora também desenhar as respetivas árvores de análise:



Exercício 2 (Desafio 2014/15)

Obtenha uma CFG para a linguagem L = $\{a^n b^m c^k : n, m > 0, k = n + m, n$ par, m ímpar$\}$. Obtenha ainda a derivação mais à esquerda e uma árvore de análise para a cadeia *aaaabbbccccccc*.

Para obter cadeias neste formato, torna-se necessário garantir que existe pelo menos um a e um b, e que o número de c's é igual ao número de a's mais o número de b's. De forma a garantir a paridade desejada, acrescentam-se

sempre símbolos aos pares, com exceção da produção base que contém b, de forma a obter um número ímpar. Uma possível solução é a seguinte:

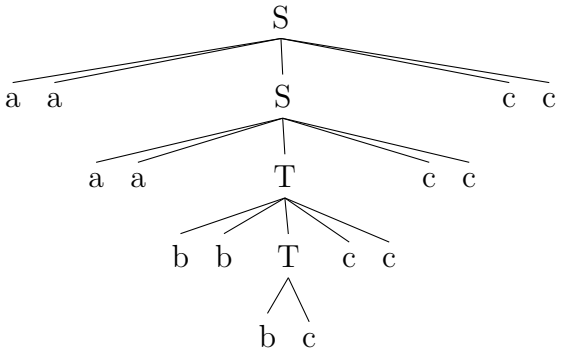
$S \rightarrow aaScc \mid aaTcc$
$T \rightarrow bbTcc \mid bc$

Com a variável S, variável de arranque desta gramática, garante-se que o número de a's é igual ao número de c's e que existem pelo menos dois a's, podendo existir mais, sempre em número par. A 'base de recursão' de S é quando é aplicada a produção $aaTcc$, que nos leva à variável T, a qual assegura um número igual de b's e de c's, assim como garante, com a sua 'base de recursão', a existência de pelo menos um b (assim como também assegura que o número total de b's é ímpar). No total, o número de c's é igual à soma do número de a's e de b's.

Vamos então fazer a derivação mais à esquerda para a cadeia pretendida, começando pela variável de arranque, S, e fazendo a cada passo de derivação a substituição de uma variável pelo corpo de uma das suas produções:

$$S \underset{lm}{\Longrightarrow} aaScc \underset{lm}{\Longrightarrow} aaaaTcccc \underset{lm}{\Longrightarrow} aaaabbTcccccc \underset{lm}{\Longrightarrow} aaaabbbccccccc$$

A partir da derivação podemos então criar a árvore de análise da cadeia, começando pela raíz (variável de arranque da gramática) e fazendo as substituições conforme a derivação.



Como podemos ver, a colheira da árvore produz a cadeia pretendida.

Exercício 3 (Desafio 2014/15)

Considere a seguinte CFG, com T = {t, s, c, d}, V = {S, W, C, D}, e variável inicial S.

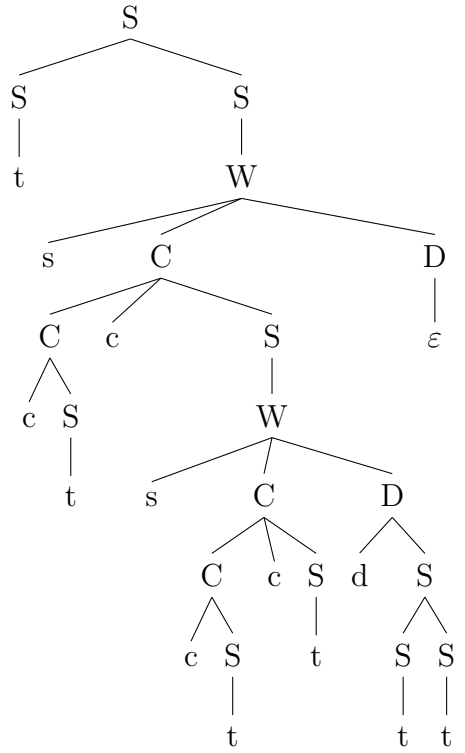
$S \to W \mid t \mid SS$
$W \to sCD$
$C \to cS \mid CcS$
$D \to dS \mid \varepsilon$

Obtenha uma derivação mais à esquerda para a cadeia *tsctcsctctdtt*, assim como a respetiva árvore de análise. Esta gramática é ambígua? Prove-o.

Podemos então começar por obter uma derivação para a cadeia, começando pela variável S, e verificando qual das possibilidades de derivação permite obter a cadeia pretendida.

$$S \underset{lm}{\Longrightarrow} SS \underset{lm}{\Longrightarrow} tS \underset{lm}{\Longrightarrow} tW \underset{lm}{\Longrightarrow} tsCD \underset{lm}{\Longrightarrow} tsCcSD \underset{lm}{\Longrightarrow} tscScSD \underset{lm}{\Longrightarrow} tsctcSD \underset{lm}{\Longrightarrow} tsctcWD \underset{lm}{\Longrightarrow} tsctcsCDD \underset{lm}{\Longrightarrow} tsctcsCcSDD \underset{lm}{\Longrightarrow} tsctcscScSDD \underset{lm}{\Longrightarrow} tsctcsctcSDD \underset{lm}{\Longrightarrow} tsctcsctctDD \underset{lm}{\Longrightarrow} tsctcsctctdSD \underset{lm}{\Longrightarrow} tsctcsctctdSSD \underset{lm}{\Longrightarrow} tsctcsctctdtSD \underset{lm}{\Longrightarrow} tsctcsctctdttD \underset{lm}{\Longrightarrow} tsctcsctctdtt$$

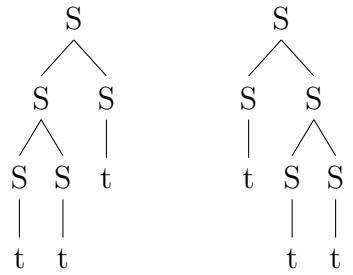
Agora, podemos também desenhar a árvore:



Esta derivação e árvore são um pouco mais complexas de obter do que as anteriores, dado o comprimento da cadeia e número de produções, sendo

necessário ter um pouco mais de atenção a quais as produções que podem produzir cada terminal e a forma como elas se relacionam entre si, de forma a encontrar o 'caminho certo'.

Em relação à ambiguidade, sim, podemos afirmar que esta gramática é ambígua, uma vez que é possível pensar numa cadeia da linguagem para a qual existem pelo menos duas árvores de análise diferentes. A título de exemplo, a cadeia *ttt* possui as seguintes duas árvores de análise:



Exercício 4

Obtenha uma CFG para a linguagem L = $\{w_1 c w_2 : w_1, w_2 \in \{a,b\}^+, |w_1| = |w_2|, w_1 \neq w_2^R\}$

Para obter uma solução para este problema, temos de garantir ao mesmo tempo que as 'partes exteriores' das cadeias têm o mesmo comprimento, mas não são o reverso uma da outra, o que significa que é necessário detetar essa diferença. Uma possível solução seria:
$S \rightarrow aSa \mid bSb \mid aTb \mid bTa$
$T \rightarrow aTa \mid aTb \mid bTa \mid bTb \mid c$

Com esta solução, e sendo S a variável de arranque da gramática, ficamos em S enquanto o início e o final da cadeia forem o reverso um do outro; quando se detetam símbolos distintos, que quebram esse padrão, passamos para T, onde se pode continuar a inserir qualquer par de símbolos, um de cada lado do *c* final, de forma a garantir comprimento igual de cada lado; no final, T é substituído por um c, terminando a cadeia.

Exercício 5

Obtenha uma CFG para a linguagem L=$\{a^p b^q c^r : p, q, r \geq 0, p = q \vee q \neq r\}$. A gramática obtida é ambígua? Justifique. Obtenha ainda uma derivação mais à direita para a cadeia *aabbccc*.

Como temos um 'ou' na definição da linguagem L, podemos separar a construção em duas sub-linguagens diferentes (L1 e L2), e no final a linguagem L será a união dessas linguagens. Vamos então ver cada caso.

Para L1=$\{a^p b^q c^r : p, q, r \geq 0, p = q\}$, temos apenas de garantir que o número de a's e de b's é igual. Para isso, podemos usar a seguinte gramática:

$S1 \rightarrow XC$

$X \rightarrow aXb \mid \varepsilon$

$C \rightarrow cC \mid \varepsilon$

Para L2=$\{a^p b^q c^r : p, q, r \geq 0, q \neq r\}$, temos de assegurar que o número de b's e de c's é diferente, o que pode ser conseguido assegurando que o número de b's é menor do que o número de c's, ou vice-versa. Assim, uma gramática possível será:

$S2 \rightarrow AY \mid AZ$

$A \rightarrow aA \mid \varepsilon$

$Y \rightarrow bYc \mid bY \mid b$

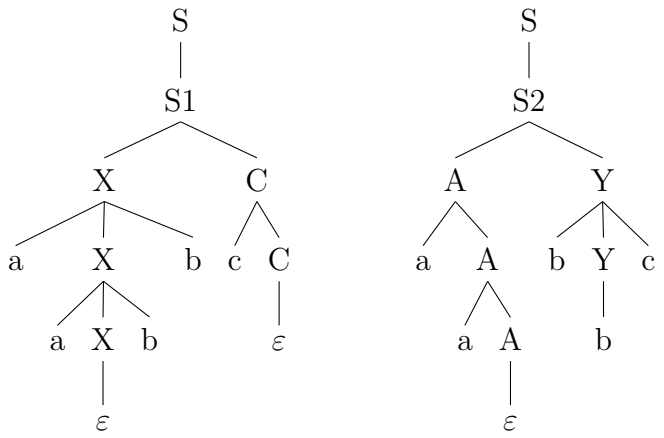
$Z \rightarrow bZc \mid Zc \mid c$

A variável A permite a existência de um qualquer número de a's. A variável Y permite assegurar a existência de pelo menos mais um b do que c's na cadeia, enquanto que a variável Z assegura o oposto, isto é, que o número de c's na cadeia é superior ao número de b's em pelo menos uma unidade.

Para obter a gramática final, serão necessárias as produções acima, mais a seguinte, para juntar as duas opções:

$S \rightarrow S1 \mid S2$

A gramática obtida é ambígua. Para o provar, podemos dar o exemplo de uma cadeia que tenha mais do que uma árvore de análise diferente. Por exemplo, a cadeia *aabbc* tem as seguintes duas possibilidades de interpretação:



Em relação à derivação mais à direita para a cadeia *aabbccc*, e para a gramática apresentada acima, teremos a seguinte derivação:
$S \underset{rm}{\Longrightarrow} S1 \underset{rm}{\Longrightarrow} XC \underset{rm}{\Longrightarrow} XcC \underset{rm}{\Longrightarrow} XccC \underset{rm}{\Longrightarrow} XcccC \underset{rm}{\Longrightarrow} Xccc\varepsilon \underset{rm}{\Longrightarrow} aXbccc \underset{rm}{\Longrightarrow} aaXbbccc \underset{rm}{\Longrightarrow} aa\varepsilon bbccc$

Exercício 6 (Exercício 2015/16)

Obtenha uma gramática para a linguagem $L1 = \{a^i b^j c^k : i, j, k \geq 0\}$, e para a linguagem $L2 = \{a^i b^j c^k : i, j, k \geq 0, j \geq i + k\}$. Consegue obter uma expressão regular para a linguagem L1? E para a linguagem L2?

Para a linguagem L1, podemos facilmente dividir a gramática em 3 componentes, cada uma para gerar cada um dos símbolos do alfabeto, ficando assim com a seguinte gramática:
$S \rightarrow ABC$
$A \rightarrow aA \mid \varepsilon$
$B \rightarrow bB \mid \varepsilon$
$C \rightarrow cC \mid \varepsilon$

Para a linguagem L2, podemos constatar que esta é um sub-conjunto de L1, mas que para gerar estas cadeias temos agora de relacionar o número de símbolos gerados. Uma possível solução será:
$S \rightarrow ABC$

$A \to aAb \mid \varepsilon$
$B \to bB \mid \varepsilon$
$C \to bCc \mid \varepsilon$

Com esta gramática, a partir da variável A consegue-se gerar um número igual de a's e de b's; com a variável C, consegue-se gerar um número igual de b's e de c's; com estas duas variáveis consegue-se garantir que o número de b's é igual à soma de a's e de c's, sendo que com a variável B se conseguem gerar b's adicionais, garantindo assim a condição de que $j \geq i + k$.

Para a linguagem L1, consegue-se facilmente pensar numa expressão regular, uma vez que se pretende gerar um qualquer número de cada um dos símbolos, ficando com L1 = a*b*c*.

Para L2, já não é possível encontrar uma expressão regular, uma vez que o número de símbolos se relacionam entre si. Podemos utilizar o lema da bombagem para provar que a linguagem não é regular, sendo portanto impossível de criar uma expressão regular para a mesma.

Exercício Proposto 1 Obtenha uma CFG para a linguagem L = $\{a^p b^q c^r : r = |p - q|\}$.

Exercício Proposto 2 Obtenha uma CFG para a linguagem L = $\{abcb^R : a, b, c \in \{0, 1\}^+\}$.

Gramáticas Regulares

Uma vez que as Linguagens Livres de Contexto englobam as Linguagens Regulares, podemos usar CFGs para representar linguagens regulares.

Surge então aqui o conceito de gramática regular: uma gramática diz-se ser regular se for consistentemente linear à esquerda ou consistentemente linear à direita. Uma gramática é linear quando todas as suas produções têm apenas uma variável no seu corpo. Para que a gramática seja consistentemente linear à direita, todas as produções são da forma $A \to xB$ ou $A \to x$, em que x representa um conjunto de símbolos terminais e B uma variável (não-terminal). Por outro lado, uma gramática diz-se consistentemente li-

near à esquerda se todas as suas produções são da forma $A \to Bx$ ou $A \to x$.

Para fazer a passagem de uma linguagem regular (representada por um DFA) para uma CFG, podemos aplicar os seguintes passos de conversão:

- Os estados do DFA tornam-se nas variáveis da gramática (estado inicial será o símbolo inicial da gramática)
- Os símbolos do alfabeto tornam-se nos terminais da gramática
- Para cada transição δ é criada uma produção: $\delta(q_1, a) = q_2$ dá origem a uma produção $Q_1 \to aQ_2$; Para cada estado final do DFA, acrescenta-se uma nova produção epsilon (por exemplo, se q_2 é um estado final, acrescenta-se uma produção $Q_2 \to \varepsilon$).

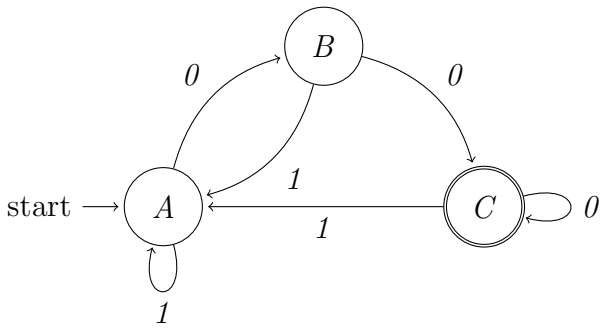
Podemos também aplicar esta tranformação a partir de um ε-NFA, adicionando ainda uma produção unitária para cada transição vazia. Por exemplo, para uma transição $\delta(q_1, \varepsilon) = q_2$ adiciona-se uma produção $Q_1 \to Q_2$.

Conversamente, podemos também fazer a passagem de uma gramática regular para um autómato que represente essa mesma linguagem, aplicando as regras de conversão em sentido contrário.

Exercício 7

Considere a linguagem das cadeias binárias que quando interpretadas como números em decimal são divisíveis por quatro. Desenhe um autómato para esta linguagem e faça a conversão para uma gramática equivalente. Indique ainda uma derivação e árvore de análise para a cadeia *10100*.

Uma possibilidade de reconhecer números divisíveis por quatro é reconhecer cadeias que terminam com dois zeros. Podemos fazê-lo facilmente com o seguinte autómato:



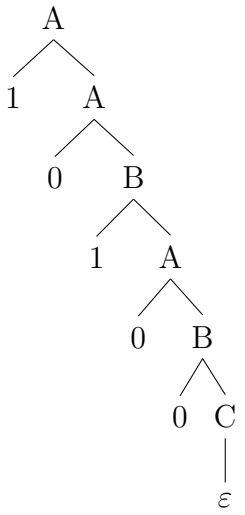
Para fazer a passagem para uma CFG, temos apenas de aplicar as regras. Identificamos então os estados como sendo as variáveis da gramática, os símbolos do alfabeto como sendo os símbolos terminais da gramática e as transições como sendo as produções da gramática. Ficamos então com as seguintes produções:
$A \to 1A \mid 0B$
$B \to 0C \mid 1A$
$C \to 0C \mid 1A \mid \varepsilon$

Como a conversão foi feita a partir de um DFA, não existe ambiguidade na gramática, havendo sempre apenas uma derivação possível a cada passo.

Para a cadeia 10100, temos a seguinte derivação:

$A \Rightarrow 1A \Rightarrow 10B \Rightarrow 101A \Rightarrow 1010B \Rightarrow 10100C \Rightarrow 10100\varepsilon$

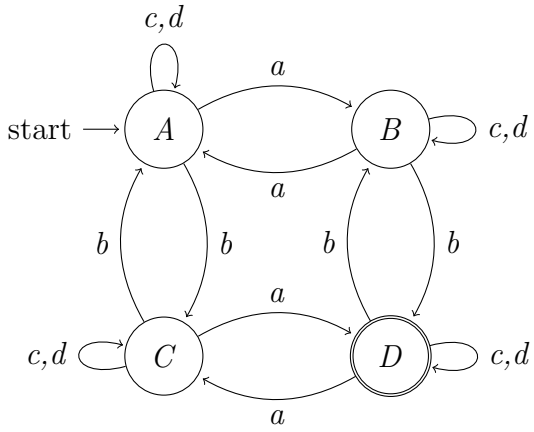
A árvore de análise correspondente seria a seguinte:



Exercício 8

Partindo da seguinte gramática, em que A é a variável de arranque, obtenha o DFA para a linguagem por ela definida e descreva a linguagem aceite.
$A \to aB \mid bC \mid cA \mid dA$
$B \to aA \mid bD \mid cB \mid dB$
$C \to aD \mid bA \mid cC \mid dC$
$D \to aC \mid bB \mid cD \mid dD \mid \varepsilon$

Sendo a gramática linear à direita, podemos então aplicar as regras de conversão de autómato para gramática em sentido inverso, e obter o autómato para esta linguagem. Iremos ter então quatro estados, conseguindo também identificar 4 símbolos no alfabeto, e um estado final.



Olhando para o autómato, torna-se mais simples de perceber que a linguagem definida tanto pela gramática como pelo autómato é a linguagem das cadeias no alfabeto a,b,c,d em que tanto o número de a's como o número de b's são ímpares.

Exercício Proposto 3 É também possível fazer a conversão de uma linguagem regular para uma gramática livre de contexto partindo de uma expressão regular. Consegue determinar um algoritmo que o permita fazer?