Problema G

Gramáticas e linguagens

Arquivo fonte: `grama.{c | cpp | java}`

Autor: Antonio Cesar de Barros Munari (Fatec Sorocaba)

A área de Linguagens Formais e Autômatos está no cerne da Teoria da Computação, e estuda as propriedades e conceitos que permitem determinar o que pode ser resolvido por meio de computadores e o que não pode. Um dos seus elementos fundamentais é o conceito de Gramática, um formalismo que indica como as palavras de uma linguagem são construídas. Conforme o tipo de gramática empregada será a complexidade das palavras que poderão ser formadas, e isso indica o tipo de linguagem com a qual estamos tratando. As linguagens mais simples são do chamado Tipo 3, as Linguagens Regulares; linguagens do Tipo 2, chamadas Linguagens Livres de Contexto, permitem construções mais complexas que as do Tipo 3; linguagens do Tipo 1, chamadas Linguagens Sensíveis ao Contexto permitem construções ainda mais sofisticadas e são apenas superadas nesse aspecto pelas linguagens do Tipo 0, as Linguagens Enumeráveis Recursivamente. O diagrama a seguir retrata a Hierarquia de Chomsky, que descreve o relacionamento entre esses tipos de linguagem. Perceba que as linguagens do Tipo 3 são um subconjunto das linguagens do Tipo 2 e estas, por sua vez, são parte das linguagens do Tipo 1 e assim por diante.

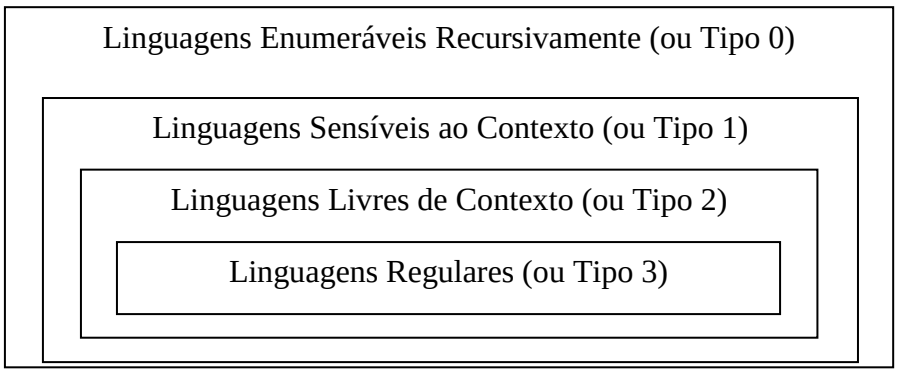

Fig 1: A hierarquia de Chomsky

Podemos identificar o tipo de uma gramática a partir da estrutura das regras utilizadas para formar as palavras. Uma regra tem sempre duas partes: o antecedente e o consequente. Ambos são strings que podem combinar três tipos de elementos: variáveis, terminais e o indicador de palavra vazia. Neste problema as variáveis serão sempre representadas por uma letra maiúscula, os terminais serão letras minúsculas e dígitos numéricos e o indicador de palavra vazia será o caracter '#'. Assim, por exemplo, o par (`aaXbc`, `aaa3`) é uma regra, onde o conteúdo antes da vírgula é o antecedente e o conteúdo após a vírgula é o consequente. Nesse exemplo encontramos no antecedente a variável *X* em meio a uma combinação dos terminais `a`, `b` e `c`, o consequente, por sua vez é composto apenas pela combinação dos terminais `aaa3`. Os parênteses não são considerados, pois servem apenas como delimitadores do par ordenado.

Para resolver este problema você não precisará entender como as regras são utilizadas para produzir as palavras de uma linguagem, precisa saber apenas que uma gramática precisa ter pelo menos uma regra embora geralmente possua várias. Além disso você precisará conhecer as características de cada tipo de gramática, que serão apresentadas resumidamente a seguir.

Se em todas as regras de uma gramática o antecedente é composto por um único caracter e esse caracter é uma variável; e além disso o consequente é formado sempre por um

único caracter, que é um terminal ou o indicador de palavra vazia ou então esse consequente é composto por dois caracteres sendo o primeiro um terminal e o segundo uma variável, essa é uma Gramática Linear à Direita e a linguagem por ela gerada é uma linguagem do Tipo 3.

Se todas as regras da gramática tiverem o antecedente composto por apenas um caracter, que seja uma variável e o consequente for composto pelo indicador de palavra vazia ou por alguma combinação qualquer de variáveis e terminais, essa é uma Gramática Livre de Contexto, e a linguagem que ela gera é uma linguagem do Tipo 2.

Uma gramática é chamada de Gramática Sensível ao Contexto se permite que o antecedente seja formado tanto por variáveis como por terminais (tem que ter pelo menos uma variável) e, além disso, o comprimento do antecedente é menor ou igual ao comprimento do consequente. Também é necessário que haja no máximo uma regra em que o consequente seja o indicador de palavra vazia; essa regra deve ser a primeira do conjunto de regras da gramática; o antecedente dela deve ser composto por um único caracter que é uma variável e essa variável não pode aparecer no consequente de nenhuma outra regra daquela gramática. Uma gramática com essas características permite gerar uma linguagem do Tipo 1.

Uma gramática válida que não se encaixe em nenhuma das situações anteriores é chamada de Gramática Irrestrita e gera linguagens do Tipo 0.

Entrada

O programa deverá processar vários casos de teste, onde cada caso é uma gramática a ser analisada. A primeira linha de um caso de teste é composta por um inteiro N ($1 \leq N \leq 20$) que indica a quantidade de regras da gramática. Seguem-se N linhas, cada uma com uma regra, no formato (A, C), onde A é uma string de até 20 caracteres indicando o antecedente da regra e C é uma string de até 20 caracteres indicando o seu consequente. Considere que apenas regras válidas serão fornecidas. Encerrar o processamento quando o programa ler um valor $N = 0$.

Saída

Para cada caso de teste, imprima "Tipo X" onde X é um número entre 0 e 3 que indica o tipo de linguagem que aquela gramática permite gerar. Atenção: você deve imprimir o tipo mais específico da linguagem gerada. Por exemplo, se uma gramática é Livre de Contexto, ela também se encaixa como sendo Sensível ao Contexto ou Enumerável Recursivamente (conforme podemos deduzir pela Hierarquia de Chomsky), mas seu programa deverá imprimir Tipo 2 (que é o mais específico nesse caso), e não Tipo 1 ou Tipo 0 na resposta.

Exemplos

Entrada:	**Saída:**
5 (P, aAbc) (A, aAbC) (A, #) (Cb, bC) (Cc, cc) 3 (S, aSbS) (S, bSaS) (S, #) 0	Tipo 0 Tipo 2