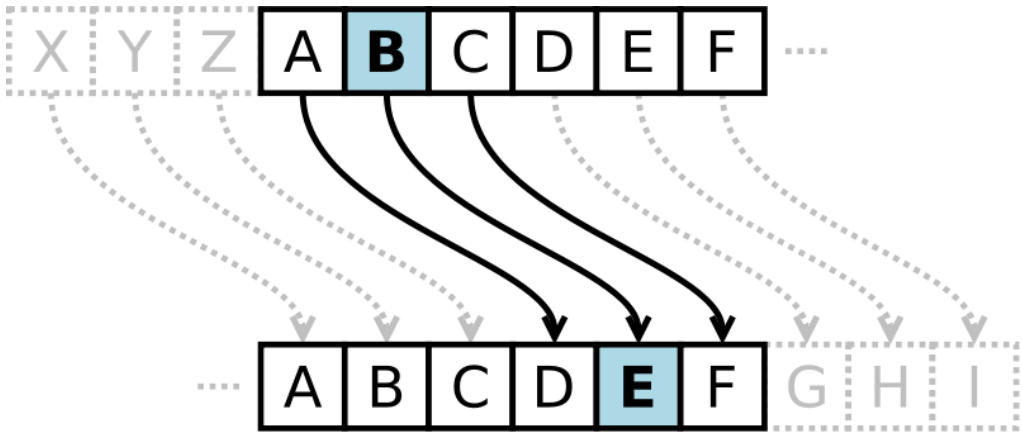
Programação Funcional
Aula 5 — A cifra de César

Pedro Vasconcelos
DCC/FCUP

2022

A cifra de César

- Um dos métodos mais simples para cifrar mensagens
- Utilizada pelo imperador Júlio César (100 AC–44 AC)
- Cada letra é substituida pela que dista k posições no alfabeto
- Quando ultrapassa a letra 'Z', volta à letra 'A'
- Exemplo (para $k = 3$)



- `https://pt.wikipedia.org/wiki/Cifra_de_C%C3%A9sar`

Problema

Escrever uma função

```
cifrar :: Int -> String -> String
```

para implementar a cifra de César com um deslocamento dado (exercício 3.8).

O módulo *Data.Char*

Vamos usar algumas funções sobre caracteres definidas no módulo *Data.Char*:

```
ord :: Char -> Int  -- código Unicode dum caracter
chr :: Int -> Char  -- caracter a partir do código Unicode
```

Para usar este módulo, colocamos a seguinte declaração no início do programa:

```
import Data.Char
```

Resolução

Começamos por definir duas funções de conversão entre as letras A...Z e os inteiros no intervalo 0...25.

```
-- converter letra para inteiro 0..25
letraInt :: Char -> Int
letraInt x = ord x - ord 'A'

-- converter inteiro 0..25 para letra
intLetra :: Int -> Char
intLetra n = chr (n + ord 'A')
```

Atenção: estas funções assumem que os seus argumentos estão nos intervalos certos!

Resolução (cont.)

Definimos agora uma função para deslocar *k* posições no alfabeto as letras minúsculas; outros caracteres ficam inalterados.

```
deslocar :: Int -> Char -> Char
deslocar k x
     | maiúscula x = intLetra ((letInt x + k)`mod`26)
     | otherwise   = x

maiúscula :: Char -> Bool
maiúscula x = x>='A' && x<='Z'
```

Resolução (cont.)

A cifra de César é definida aplicando a função *deslocar* a cada caracter da cadeia dada.

```
cifrar :: Int -> String -> String
cifrar k xs = [deslocar k x | x<-xs]
```

Resolução (cont.)

Também podemos usar deslocamentos negativos; por exemplo, para descodificar uma mensagem cifrada com *cifrar k* usamos *cifrar* $(-k)$:

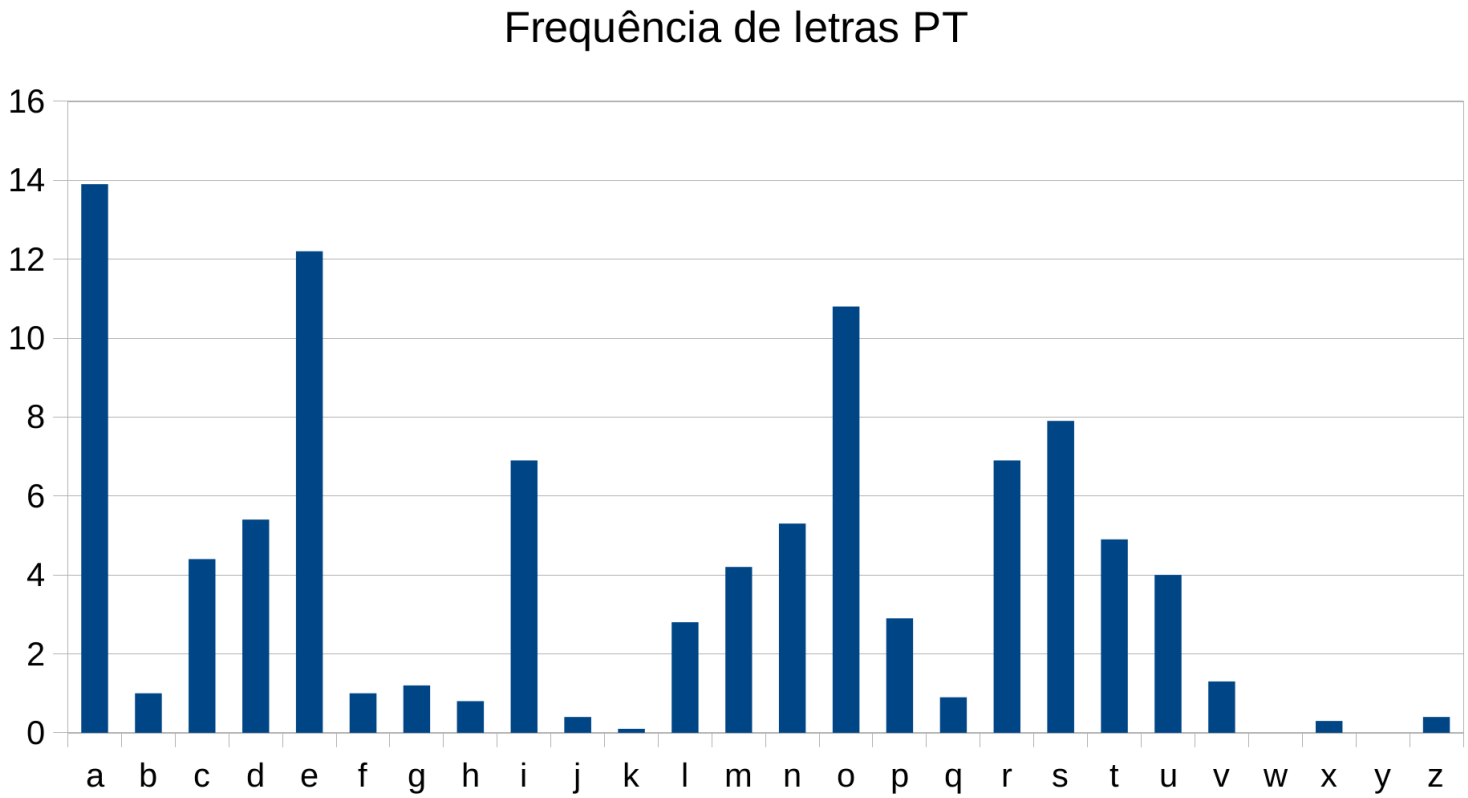
```
> cifrar 3 "HASKELL IS COOL"
"KDVNHOO LV FRRO"
> cifrar (-3) "KDVNHOO LV FRRO"
"HASKELL IS COOL"
```

Quebrar a cifra

Vamos agora ver como quebrar a cifra, isto é, determinar qual o deslocamento usado para cifrar uma mensagem.

Quebrar a cifra (cont.)

As letras do alfabeto têm frequências relativas características de cada língua; para o Português (em percentagens):



Fonte: `https://www.dcc.fc.up.pt/~rvr/naulas/tabelasPT/`

Quebrar a cifra (cont.)

Plano:

1. Calcular as frequências relativas no texto cifrado
2. Deslocar a tabela $0 \ldots 25$
3. Escolher o deslocamento que melhor corresponde à distribuição do Português

Este método é muito eficaz se a mensagem não for muito curta.

Calcular frequências relativas

```
freqs :: String -> [Float]
freqs xs = [porcento (ocorrências x xs) n | x<-['A'..'Z']]
    where n = contarMaiúsculas xs

ocorrências :: Char -> String -> Int
ocorrências y xs = length [x | x<-xs, x==y]

contarMaiúsculas :: String -> Int
contarMaiúsculas xs = length [x | x<-xs, maiúscula x]

porcento :: Int -> Int -> Float
porcento num denom = (fromIntegral num /
                      fromIntegral denom) * 100
```

Comparar distribuições

Para comparar frequências observadas

$$[o_1, \ldots, o_n]$$

com frequências esperadas

$$[e_1, \ldots, e_n]$$

vamos empregar um método estatístico: o teste de Chi-quadrado.

Comparar distribuições (cont.)

Teste de Chi-quadrado

Quanto menor for o valor de

$$\chi^2 = \sum_{i=1}^{n} \frac{(o_i - e_i)^2}{e_i}$$

maior será a correspondência entre distribuições.

Uma função para calcular esta medida:

```
chiquad :: [Float] -> [Float] -> Float
chiquad os es = sum [((o-e)^2)/e | (o,e)<-zip os es]
```

Encontrar o deslocamento

```
quebrar :: String -> Int
quebrar xs = k
  where
    obs = freqs xs
    chitab = [chiquad (rodar k obs) tabelaPT | k<-[0..25]]
    k = head (indices (minimum chitab) chitab)

tabelaPT :: [Float] -- Frequências das letras PT
tabelaPT = [ 13.9, 1.0, ... ]

rodar :: Int -> [a] -> [a] -- Rotação circular
rodar k xs = drop k xs ++ take k xs

indices :: Eq a => a -> [a] -> Int
-- Indices das ocorrências (ver aula 4)
```