Práctica 3. Longitud constante - D.Kraft

19 de octubre del 2022 Marcos Hidalgo Baños

a) Algoritmo para la construcción de códigos de longitud constante.

Este ejercicio es bastante sencillo de implementar (se realiza en una sola línea), aunque emplea alguna función interesante descrita al final del documento.

```
codLongitudConstante <- function(mensaje, simbolos, m) {
  # @param mensaje, numero constante de mensajes por unidad de tiempo
  # @param simbolos, alfabeto de codificacion con D simbolos
  # @param m, tamaño de la m-upla de mensajes

  return(ceiling(m*log2(length(mensaje))/log2(length(simbolos))))
  # Expresion matematica para codificar la m-tupla de mensajes
}
```

```
# Pruebas
cifrasDecimales = c(0:9)
simbolos = c(0,1)
codLongitudConstante(cifrasDecimales, simbolos, 1)
```

Pruebas realizadas con los ejemplos de las transparencias.

```
> # Pruebas
> cifrasDecimales = c(0:9)
> simbolos = c(0,1)
> codLongitudConstante(cifrasDecimales, simbolos, 1)
[1] 4
```

```
> # Pruebas
> cifrasDecimales = c(0:9)
> simbolos = c(0,1)
> codLongitudConstante(cifrasDecimales, simbolos, 3)
[1] 10
```

(Notar que la entrada para ambas pruebas difiere)

b) Fórmula para calcular la longitud media del código.

En esta implementación partimos de un mensaje codificado previamente.

```r
longitudMedia <- function (codificacion, pk) {
  # @param codificacion, vector de mensajes codificados
  # @param pk, vector de probabilidades
  long = 0 # Variable acumulativa del resultado

  for (i in 1:length(codificacion)) {
    # Recorremos los valores de la codificacion para el sumatorio
    long = long + pk[i]*nchar(codificacion[i])
    # Expresion para el calculo de la longitud media
  }
  return(long)
}
```

Pruebas realizadas con los ejercicios de la relación.

```r
# Pruebas (ejercicio de clase, relacion de ejercicios)
mensajeCodificado = c('1','01','0000','0001','00100','00101','00110','00111')
pk = c(0.4, 0.2, 0.1, 0.1, 0.05, 0.05, 0.05, 0.05)
longitudMedia(mensajeCodificado, pk)
```

```
> # Pruebas (ejercicio de clase, relacion de ejercicios)
> mensajeCodificado = c('1','01','0000','0001','00100','00101','00110','00111')
> pk = c(0.4, 0.2, 0.1, 0.1, 0.05, 0.05, 0.05, 0.05)
> longitudMedia(mensajeCodificado, pk)
[1] 2.6
```

Observación

Para la implementación en R, el mensaje codificado debe ser un **vector de Strings** para poder obtener su longitud total.

Si no fuese así, el valor ‘0001’ sería introducido como ‘1’ y daría lugar a un cálculo incorrecto.

c) Fórmula para verificar la desigualdad de Kraft.

Para comprobar que una entrada cumple la desigualdad, basta con analizar elemento a elemento hasta que no se cumpla la condición y salir antes de tiempo.

```
desigualdadKarft <- function(S, simbolos, longitudes) {
  # @param S, vector de probabilidades P(A)
  # @param simbolos, alfabeto de codificacion con D simbolos
  # @param longitudes, vector de longitudes variables
  i = 1
  res = 0

  while (i <= length(S)) {
    # Mientras que haya mensajes...
    res = res + length(simbolos)^((-1)*longitudes[i]*i)
    # Expresion que debe cumplirse para la desigualdad de Kraft
    if (res > 1) {
      # ... comprobamos que no des-cumpla la propiedad
      break
      # Si es el caso, dejamos de comprobar
    }
    i = i + 1
    # Actualizamos i para que apunte al siguiente mensaje
  }
  return(i > length(S))
  # Si hemos salido antes de tiempo, esto deberia ser falso
  # Si se cumple la propiedad, esto es verdadero siempre
}
```

```
# Pruebas
mensajes = c(1:6)
simbolos = c(0,1)
longitudes = c(2, 2, 3, 4, 4, 5)
desigualdadKarft(mensajes, simbolos, longitudes)
```

Pruebas realizadas con los ejemplos de las transparencias.

```
> # Pruebas
> mensajes = c(1:6)
> simbolos = c(0,1)
> longitudes = c(2, 2, 3, 4, 4, 5)
> desigualdadKarft(mensajes, simbolos, longitudes)
[1] TRUE
```

Si la entrada no cumpliese la desigualdad de Kraft, el programa devolvería FALSE por la razón comentada en el código.

Funciones auxiliares.

➔ ceiling (x)

Función para redondear al siguiente entero más próximo de x.

➔ nchar (x)

Obtiene el tamaño del String pasado por parámetro.

➔ length (x)

Equivalente a la función nchar para cualquier tipo de vector de valores.

➔ log2 (x)

Función para el cálculo del logaritmo en base dos de un valor x.

- Marcos Hidalgo Baños -