Taller 2 - Algoritmo juego de adivinanzas*

Michael Gonzalez[a], Santiago Castro[1]

[a] *Pontificia Universidad Javeriana, BogotÃ¡, Colombia*

Abstract

En este documento se presenta la forma de dividir y vencer aplicada en el algoritmo de adivinanza de un número cualquiera.

Keywords: algoritmo, escritura formal, .

1. Análisis del problema

El algoritmo tiene como entradas una secuencia S compuesta por numeros naturales, un indice de inicio B y uno de final E, ambos de igual manera seran numeros naturales.

El objetivo de este algoritmo es que mediante la interacion con el usuario, el algoritmo encuentre el numero el cual el esta pensando.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

2. Diseño del problema

El análisis anterior nos permite, usar herramientas como dividir y vencer, para este algoritmo, el usuario tendría 3 posibles respuestas : 'Es mayor' , 'Es menor' y por ultimo 'Es correcto', con estas 3 posibles opciones el algoritmo hallara, la cota superior (Es menor) la cota inferior (Es mayor) y eventualmente el número correcto.

Esta busqueda sera realizada por una busqueda N-aria, en la cual los pivotes seran encontrados por una N-division, es decir 1/N y N sera el numero de respuestas que el usuario desee recibir.

1. *Entradas*: $X = \langle s_{\in}\mathbb{N}, b_{\in}\mathbb{N}, e_{\in}\mathbb{N}\rangle$ S una secuencia de elementos del conjunto $\mathbb{N}$, que pertenece a los numeros naturales y dos indices, inicio y fin que pertenecen a los numeros naturales.

*Este documento presenta la escritura formal de un algoritmo.

Email addresses: michael.gonzalez@javeriana.edu.co (Michael Gonzalez), castromsantiago@javeriana.edu.co (Santiago Castro)

2. *Salidas*: $n_{\in}\mathbb{N}$ Un numero N que pertenece a los naturales y que cumple con la condicion de ser el numero que el usuario penso.
   el promedio.

3. Algoritmos de solución

3.1. Algoritmo evidente

Este algoritmo de solución es una traducción literal de las deficiones de lo que se quiere resolver.

Algorithm 1 Adivina el número que el usuario pensó.

Require: $\langle n \in \mathbb{N} \rangle$
Require: $\langle ci \in \mathbb{N}; default : 0 \rangle$
Require: $\langle p \in \mathbb{N}; default : 100 \rangle$
Require: La función *ObtenerCercanias* permite la obtención de una lista de cercanias a partir de la lista de pivotes. La lista resultante ordenada solo guarda valores hasta que el usuario determine un pivote como cota superior o como el valor buscado, de lo contrario se recorre toda la secuencia, permitiendo la obtención de un nuevo rango.

```
1:  procedure ADIVINAR(n, ci = 0, p = 100)
2:      i ← 0
3:      v ← ci
4:      qs ← ∅
5:      while v < (n + 1) * p do
6:          qs_i ← v
7:          i ← v + p
8:          i ← i + 1
9:      end while
10:     cs ← ObtenerCercanias(qs)
11:     if |cs|>1 then
12:         ci ← qs_{|cs|−1} + 1
13:     else
14:         ci ← ci + 1
15:     end if
16:     if cs_0 =′ igual′ then
17:         return qs_{|cs−1|}
18:     else
19:         return AdivinarAux(n, ci, qs_{|cs|−1} − 1)
20:     end if
21:     return Adivinar(n, qs_{|cs−1|+1}, p * 10)
22: end procedure
```

Algorithm 2 Evident algorithm to compute average and non-biased standard devitation from a sequence of numbers.

Require: $X = \langle x_i \in \mathbb{T} \rangle$
Require: The arithmetic operations $+, -, \cdot$ should be defined in $\mathbb{T}$.
Require: The arithmetic operation $\mathbb{T} \div \mathbb{N}^+ \to \mathbb{T}$.

```
1: procedure AdivinarAux(n, b, e)
2:     d ← ⌊(e − b + 1) ÷ 2⌋
3:     qs ← ∅
4:     for i ← 1 to n do
5:         qs_i ← b + d ∗ i
6:     end for
7:     if b >= e then
8:         return b
9:     end if
10:    cs ← ObtenerCercanias(qs)
11:    if |cs|>1 then
12:        ci ← qs_{|cs|−1} + 1
13:    else
14:        ci ← ci + 1
15:    end if
16:    if cs_0 =′ igual′ then
17:        return qs_{|cs−1|}
18:    else if cs_{|cs|−1} then
19:        return AdivinarAux(n, ci, qs_{|cs|−1} − 1)
20:    else
21:        return AdivinarAux(n, qs_{|cs|−1} + 1, e)
22:    end if
23:    return Adivinar(n, qs_{|cs−1|+1}, p ∗ 10)
24: end procedure
```

3.1.1. Invariante

En cada recursion la cota superior, o la cota inferior cambian segun el resultado del usuario, para dar mas presicion a la adivinanza.

3.1.2. Análisis de complejidad

Por inspección de código el algoritmo es $O(n)$.