Algoritmo para el juego de «Adivinar un número» con la estrategia de Dividir y vencer

Nicolás Reyes Díaz[1]

[a]*Pontificia Universidad Javeriana, Bogotá, Colombia*

Abstract

En este documento se presenta la descripción formal del algoritmo para solucionar el juego de adivinar un número

Keywords: algoritmo, escritura formal, juego, adivinar, adivinador, pensador, dividir y vencer

1. Análisis del problema

El juego de "adivinar un número" se juega entre dos personas: una de ellas piensa en un número natural y la otra intenta adivinarlo dando algunas posibles respuestas. En cada turno, el "adivinador" da un conjunto de n números y el "pensador" indica para cada número, si es mayor, menor o igual. Para este algoritmo, pensando en una implementación posterior en un lenguaje de programación, se considera al conjunto de los números naturales como:

$$N = \{1, 2, 3, 4, 5, 6..,2147483647\}$$

Vistas las reglas del juego, se aclara que el algoritmo presentado a continuación tiene el propósito de tomar el rol de «adivinador», es decir, que éste hará la búsqueda del número que el «pensador» elija.

2. Diseño del problema

El análisis anterior nos permite diseñar el problema: definir las entradas y salidas de un posible algoritmo de solución, que aún no está definido.

1. *Entradas*: n, el número de posibles respuestas que puede presentar el «adivinador» por turno. $k \in N$, el número elegido por el «pensador»
2. *Salidas*: $k \in N$, el número elegido por el «pensador»

⋆Este documento presenta la escritura formal de un algoritmo para el juego de adivinar un número.

3. Algoritmos de solución

3.1. Algoritmo con la estrategia de «Dividir y Vencer»

Este algoritmo de solución se basa en la estrategia de «Dividir y Vencer» para solucionar el problema. Adicionalmente, se da por sentado que en la primerta iteración del algoritmo, el inf y sup serán 1 y 2147483647 respectivamente.

Algorithm 1 Divide and conquer based algorithm to guess number

Require: $n \in N, k \in N, inf, sup$
1: **procedure** GUESSNUMBER(n, k, inf, sup)
2: **if** $sup < inf$ **then**
3: **return** 0
4: **end if**
5: $max \leftarrow sup$
6: $min \leftarrow inf$
7: $cont \leftarrow min$
8: $p \leftarrow (sup - inf) \div (n + 1)$
9: **for** $i \leftarrow 1$ to $n + 1$ **do**
10: $cont \leftarrow cont + p$
11: **if** $cont = k$ **then**
12: **return** k
13: **else if** $cont > k$ and $cont < max$ **then**
14: $max \leftarrow cont$
15: **else if** $cont < k$ and $cont > max$ **then**
16: $min \leftarrow cont$
17: **end if**
18: **end for**
19: **return** $[GuessNumber(n, k, min, max)]$
20: **end procedure**

3.1.1. Invariante

En cada llamado recursivo se tiene que el número buscado debe estar en el intervalo (min,max). También n y k son números que no varían a lo largo del algoritmo.

3.1.2. Análisis de complejidad

Por inspección de código simple podemos llegar a que la complejidad temporal del algoritmo es $T(n) = T(n/n + 1) + O(n)$, Cuando $n \geqq 1$. De esta forma, se comprobaron todos los casos del teorema maestro, hasta llegar al tercero, dando este desarrollo:

$n = n^{log_{n+1}(1+\epsilon)}$

$log_{n+1}(1 + \epsilon) = 1$

$(1 + \epsilon) = n + 1$
$n = \epsilon$
Reemplazando en la ecuación original:
$n = n^{log_{n+1}(1+n)}$
Desarrollando:
$log_{n+1}(n + 1) = 1$
$n + 1 = n + 1$
Así, se cumple el caso 3 y la complejidad del algoritmo es de:
$\Theta(n) = n$

4. Notas de implementación

A la hora de hacer el programa, será recomendable tener una interfaz de usuario para indicar al algoritmo tanto el número a adivinar, como las respuestas permitidas por turno. También para darle la opción al usuario, con el rol del «pensador», de indicarle al algoritmo si ha acertado, si el número es menor o mayor. También pensando en que es un juego, se puede desarrollar un contador, para ver cuántos intentos le lleva al algoritmo adivinar el número.