Adivinar el número pensado por un usuario[*]

Gabriel Francisco Coronado Sánchez[a]

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

Abstract

En este documento se presenta la escritura formal del problema «Adivinar un número» junto con algunos algoritmos de solución.

Keywords: algoritmo, escritura formal, adivinanza, dividir y vencer, recurrencia.

1. Análisis del problema

Una persona puede pensar un número x que cumple con la condición:

$$min \leq x \leq max$$

donde min y max representan, respectivamente, el mínimo y el máximo del rango de números posibles por escoger. En consecuencia, cada número dentro del rango se define como un número mayor ($>$), menor ($<$) o igual ($=$) al número pensado. De estas definiciones, se formaliza lo siguiente:

1. x es un valor no conocido, que cumple con la condición $x \in \mathbb{Z}$.
2. min es un valor conocido que cumple con la condición $min \in \mathbb{Z} \cup (-\infty, x]$.
3. min es un valor conocido que cumple con la condición $min \in \mathbb{Z} \cup [x, \infty)$.
4. Los posibles valores por escoger deben ser enumerados (s_i) y existe un número finito $t = max - min + 1$ de ellos; entonces, esos datos deben estar representados en una secuencia de valores $S = \langle s_i \in \mathbb{Z} \cup [min, max] \rangle$, $i \in [1, t]$.
5. n es el número de posibles respuestas que serán dadas por el «adivinador» en un momento dado, definido por $n \in [1, t]$
6. Existe un número n de comparaciones c, donde c_i representa la comparación entre s_i y x, y se representa en la secuencia de caracteres $C = \langle c_i \in \{<, >, =\} \rangle$, $i \in [1, n]$.

[*] Este documento presenta la escritura formal del algorítmo de adivinanza de números.
Email address: `g-coronado@javeriana.edu.co` (Gabriel Francisco Coronado Sánchez)

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 entregadas por el «adivinador» y $R = \langle r_i \in \{<, >, =\}\rangle$, la secuencia de caracteres que representan la relación de las posibles respuestas con el número x.

2. *Salidas*: $S = \langle s_i \in \mathbb{Z} \cup [min, max]\rangle$ un número n de posibles soluciones y x la confirmación del número pensado por el «pensador».

3. Algoritmos de solución

3.1. Algoritmo adivinador

Este algoritmo de solución es el método de selección de posibles respuestas.

Algorithm 1 Algorithm to propose possible guesses for the selected number.

```
Require: n
Require: min
Require: max
procedure Guess(n, min, max)
    S
    |S| ← n
    t ← max − min + 1
    if t < n then
        |S| ← t
    end if
    r ← t ÷ n
    S ← GuessAux(S, min, max, r, 1, n)
    return S
end procedure

procedure GuessAux(S, min, max, r, b, e)
    n ← |S|
    if b > e then
        return S
    else
        q ← (b + e) ÷ 2
        min_q ← r ∗ q + min
        if q < n then
            max_q ← min_q + r − 1
        else
            max_q ← max
        end if
        S_q ← Rand{min, max}
        S ← GuessAux(S, min, max, b, q − 1)
        S ← GuessAux(S, min, max, q + 1, e)
        return S
    end if
end procedure
```

3.1.1. Invariante

En cada recurrencia, el arreglo S, con tamaño n, se llena con una secuencia ordenada de números no repetidos entre min y max. estos valores se le presentan al «pensador» para que de información sobre su relación con el número pensado x.

3.1.2. Análisis de complejidad

El algoritmo se puede representar de la forma

$$T(n) = 2T(\frac{n}{2}) + O(1)$$

Por la primera regla del teorema maestro:

$$1 \in O(n^{\log_2 2-\epsilon})$$

Si $\epsilon = 1$ entonces:

$$n^{\log_2 1} = n^0 = 1$$

Cumpliendo la condición. Por lo tanto, la complejidad del algoritmo es $\theta(n)$.

3.2. Algoritmo de análisis

Este algoritmo de solución es la solución para la comparación entre las posibles respuestas y el número elegido por el «pensador»

Algorithm 2 Algorithm to analize the guesses according to their relations with the selected number.

```
Require: S = ⟨s_i ∈ ℤ ∪ [min, max]⟩
Require: C = ⟨c_i ∈ {<, >, =}⟩
Require: min
Require: max
procedure ANALIZE(S, C, min, max)
    n ← |S|
    x ← min − 1
    return AnalizeAux(S, C, x, min, max, 1, n)
end procedure

procedure ANALIZEAUX(S, C, x, min, max, b, e)
    if b > e then
        return [min, max, x]
    else
        q ← (b + e) ÷ 2
        if C_q = " = " then
            x ← S_q
            return [min, max, x]
        else if C_q = " > " then
            max ← S_q − 1
            return AnalizeAux(S, C, x, min, max, b, q − 1)
        else if C_q = " < " then
            min ← S_q + 1
            return AnalizeAux(S, C, x, min, max, q + 1, e)
        end if
    end if
end procedure
```

3.2.1. Invariante

En cada recurrencia, se está relacionando el caracter C_i con el valor S_i. Si, con esa relación, se descubre que S_i es igual al valor seleccionado por el «pensador», se dice que $x = S_i$, y se actualiza el valor. De lo contrario, el valor del mínimo y el máximo se actualizan respectivamente si S_i es menor o mayor al número seleccionado, y se continua analizando hasta que todos los valores de S hayan sido revisados o se encuentre el valor de x.

3.2.2. Análisis de complejidad

Como solo se accede a una recurrencia por instancia, el algoritmo se puede representar de la forma

$$T(n) = T(\frac{n}{2}) + O(1)$$

Por la segunda regla del teorema maestro:

$$1 \in O(n^{\log_2 1})$$

, entonces:

$$n^{\log_2 1} = n^0 = 1$$

Cumpliendo la condición. Por lo tanto, la complejidad del algoritmo es $\theta(\log_2 n)$

3.3. Notas de implementación

1. Mientras que el «pensador» no indique que el número fue mencionado, debe preguntarsele repetidamente por la relación entre los números propuestos y su número.
2. el valor de n puede reducirse si se reducen también las posibles respuestas según las modificaciones de max y min.
3. Es necesario que se verifique que el pensador de un operador de comparación para cada número dado, y que sea congruente. Con la secuencia ordenada, se sabe que:
   a) un número no puede ser mayor a x si el siguiente es menor.
   b) un número no puede ser menor a x si el anterior es mayor.
   c) un número no puede ser igual a x si el anterior es mayor o el siguiente es menor.
   d) no puede haber más de un valor asignado a x.