Taller 2 - Juego de Adivinanzas*

Juan Camilo Sánchez Urrego[1], Sofía Cespedes Vargas[1]

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

Abstract

En este documento se busca analizar y resolver el problema de "Adivinar Números". Concretamente, adivinar un número en concreto que un usuario haya pensado previamente. Además, de buscar resolver dicho problema utilizando la estrategia de dividir y vencer.

Keywords: algoritmo, pensador, adivinador, adivinar, juego.

1. Análisis del problema

Inicialmente es necesario comprender el contexto del problema a resolver, en este caso el juego "adivinar un número" el cual implica dos participantes: el "pensador" y el "adivinador". El "pensador" elige un número natural y el "adivinador" intenta adivinarlo haciendo preguntas estratégicas, donde el objetivo es que el "adivinador" acierte el número pensado por el "pensador" utilizando la menor cantidad de intentos posible. el juego consta de unas reglas fundamentales para el funcionamiento del juego las cuales son:

1. **E**l "pensador" elige un número natural desconocido para el "adivinador".
2. **E**n cada turno, el "adivinador" proporciona un conjunto de números al "pensador". El "pensador" compara cada número del conjunto con el número pensado y responde si es mayor, menor o igual.
3. **E**l "adivinador" utiliza esta información para restringir las posibilidades y hacer un nuevo intento de adivinar el número

Ahora bien cuando se plantea una solución acorde al algoritmo es necesario tener en cuenta algunos factores tales como la eficiencia pues se busca un algoritmo que minimice el número de intentos requeridos para adivinar el número y la complejidad dado que esta evalúa la complejidad temporal y espacial del algoritmo propuesto. Por otro lado es factible tener en cuenta y analizar el peor de los casos ya que nos permite evaluar el rendimiento del algoritmo propuesto en el peor momento por ejemplo la máxima cantidad de intentos que el "adivinador" podría necesitar para adivinar el número.

*Este documento presenta la escritura formal de un algoritmo.

Email addresses: sanchezjcamilo@javeriana.edu.co (Juan Camilo Sánchez Urrego), sofia.cespedes@javeriana.edu.co (Sofía Cespedes Vargas)

- Es necesario estructurar la función del problema, pues nos ayudara a comprenderlo de manera conceptual. $f(a_i)$: Función que devuelve la respuesta del "pensador" para el número a_i en el turno i.

- El "pensador" selecciona un número natural n.

- El "adivinador" hace un conjunto de preguntas $\{a_1, a_2, ..., a_k\}$, donde k es el tamaño del conjunto.

- El "pensador" proporciona las respuestas $\{r_1, r_2, ..., r_k\}$.

- Para cada a_i, el "pensador" devuelve r_i, indicando si a_i es mayor, menor o igual a n.

2. Diseño del problema

Para Diseñar el problema es necesario definir las entradas , el proceso y las salidas.

1. *Entradas*:
   a) Número Pensado (n): Es el número natural que el "pensador"selecciona al comienzo del juego. Esta es la información que el .ªdivinadorïntentará adivinar.
   b) Conjunto de Números $\{a_1, a_2, ..., a_k\}$ En cada turno, el .ªdivinador"proporciona un conjunto de números al "pensador". Estos números son los que el .ªdivinador.está considerando como posibles candidatos para el número pensado n.
2. *Salidas*:
   a) Respuestas del Pensador $\{r_1, r_2, ..., r_k\}$: Para cada número a_i en el conjunto proporcionado por el .ªdivinador", el "pensador"devuelve una respuesta indicando si a_i es mayor, menor o igual al número pensado n.
      - $r_i = -1$: a_i es menor que n.
      - $r_i = 0$: a_i es igual a n.
      - $r_i = 1$: a_i es mayor que n.

3. Algoritmo de solución

3.1. Procedimiento

Algorithm 1 Algoritmo que calcula el promedio utilizando un algoritmo de suma iterativo.

Require: $q \in \mathbb{N}$
Require: $r \in \mathbb{Z}$
Require: $b \in \mathbb{N}$
Require: $e \in \mathbb{N}$

```
1:  procedure ADIVINADOR(q, r, b, e)
2:      if r = 0 then
3:          return ⟨q, b, e⟩
4:      end if
5:      if r = −1 then
6:          e ← q − 1
7:          q ← ⌊(b + e)/2⌋
8:      end if
9:      if r = 1 then
10:         b ← q + 1
11:         q ← ⌊(b + e)/2⌋
12:     end if
13:     return ADIVINADOR(q, 0, b, e)
14: end procedure
```

3.2. Invariante

1. **Inicialización:** Al iniciar la variable q contiene el valor que se respondió en la jugada anterior, r es la respuesta del "pensador" de si el número es mayor o menor, o en caso de ser la primera jugada puede ser 0, b contiene el limite inferior del rango acotado anteriormente, e contiene el limite superior del rango acotado.
2. **Mantenimiento:** Después de pasar por los condicionales, q contiene el nuevo número para probar, $r = 0$ y b y e tienen el rango acotado con la nueva información.
3. **Finalización:** Eventualmente el "pensador" tendrá que responder que se ha adivinado el número correctamente.

3.3. Análisis de complejidad del Recursivo

Se plantea la siguiente ecuación de complejidad temporal del algoritmo:

$$T(n) = aT\left(\frac{n}{b}\right) + f(n) \tag{1}$$

$$f(n) = 1,\ a = 1,\ b = 2$$

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

Caso 1

$$f(n) \in O(n^{\log_b a - \epsilon}) \tag{2}$$

$$1 = n^{\log_2 1 - \epsilon}$$

$$0 = \log_2(1 - \epsilon) \wedge \epsilon = 0 \rightarrow 0 = 0$$

Este caso no se puede cumplir debido a que:

$$\epsilon > 0$$

Caso 2

$$f(n) \notin \Theta(n^{\log_b a}) \tag{3}$$

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

Caso 3

$$f(n) \in \Omega(n^{\log_b a + \epsilon}) \tag{4}$$

$$1 = n^{\log_2(1+\epsilon)}$$

$$0 = \log_2(1 + \epsilon)$$

$$\epsilon = 0 \wedge \log_2(1 + \epsilon) \rightarrow 0 = 0$$

Este caso no se puede cumplir debido a que:

$$\epsilon > 0$$

Orden de Complejidad - Caso 2

$$T(n) \in \Theta(n^{\log_2 1} \log_2 n)$$

$$T(n) \in \Theta(\log_2 n)$$