# Taller 2

Julian Arana[a]

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

**Abstract**

En este algoritmo veremos el desarrollo del algoritmo que permite hacer el juego de adivina el numero donde el usuario lo asigna y el algoritmo lo adivina.

*Keywords:* algoritmo, escritura formal, adivinar, dividir y vencer.

## 1. Analisis del problema

En este problema tenemos 2 actores, el software dara un numero aleatorio entre 1 y 1000. El numero que de sera el limite superior que se tendra, por ejemplo si el numero es 500 el rango del numero que el usuario puede poner es de 1 a 500.

Un ejemplo de como deberia funcionar:

$$|1|2|3|4|5|6|7|8|9|10|$$

Numero a adivinar: 8.

Dividir el algoritmo a la mitad, si el numero a adivinar es menor al de adivinar, se usara como limite inferior el numero en la mitad, si es al reves, se usa este numero como el limite superior.

$$|5|6|7|8|9|10| \rightarrow |8|9|10| \rightarrow |8|9| \rightarrow |8|$$

Numero encontrado.

## 2. Diseno del problema

El annlisis anterior nos permite disenar el problema: definir las entradas y salidas de un posible algoritmo de solucion, que aun no esta definido.

1. *Entradas*: Los limites superior [e] e inferior [b] $\rightarrow [b, e], Elnumeroaadivinara.$
2. *Salidas*: Un mensaje diciendo si el numero adivinado es mayor, menor o igual al ingresado por el usuario.

*En este documento se realiza el taller 2 de analisis de algoritmos, en el cual se nos pidio elaborar un algoritmo de dividir y vencer para hacer un juego de adivina el numero donde el usuario lo asigna y el algoritmo lo adivina.

*Email address:* `juliana-aranag@javeriana.edu.co` (Julian Arana)

## 3. Algoritmos de solucion

### *3.1. Algoritmo evidente*

Este algoritmo de solucion es una traduccion literal de las deficiones de lo que se quiere resolver.

**Algorithm 1** Algoritmo de bÃºsqueda binaria

```
procedure ENC_AUX(b, e, a)
    if b > e then
        return 0
    else
        q ← round((b + e)/2)
        print .El nÃºmero adivinado por el algoritmo es: q"
        if q > a then
            print .El nÃºmero es menor"
            return ENC_AUX(b, q, a)
        else if q < a then
            print .El nÃºmero es mayor"
            return ENC_AUX(q, e, a)
        else
            print .El nÃºmero es correcto"
            return q
        end if
    end if
end procedure
```

**Algorithm 2** FunciÃ³*nparaencontrarelnmero*

```
procedure ENCONTRAR
    rango ← random.randint(1, 1000)
    print .El rango del numero va desde 0 a rango"
    numero ← int(input(”Ingreseelnumeroaadivinar”))
    while numero > rango do
        numero ← int(input(”Elnumeroingresadoesmayoralrangodado.Seleccioneotronmero”))
    end while
    adivinado ← enc_aux(0, rango, numero)
end procedure
```

### Analisis de Complejidad

El algoritmo implementa la estrategia de dividir y vencer para adivinar un numero en un rango dado. Supongamos que el rango dado es 1 a $n$ y el numero a adivinar es $x$.

Sea $T(n)$ la complejidad temporal del algoritmo para un rango de tamaÃ±*on*.

*Analisis*

El algoritmo divide el rango a la mitad en cada paso y realiza una comparacion. Por lo tanto, la complejidad temporal puede expresarse de la siguiente manera:

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

Donde $O(1)$ representa el tiempo constante para realizar la comparacion y otros calculos.

Podemos ver que la complejidad es $O(\log n)$, ya que el algoritmo divide repetidamente el rango por la mitad hasta encontrar el numero deseado.

La complejidad temporal del algoritmo es $O(\log n)$.