juego de adivinar un número: diseño, implementación y explicación ⋆

Andrés Felipe Vaca Lago[a]

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

Abstract

En este documento se presenta la escritura formal e implementación de un algoritmo el cual permita a un usuario jugar con el rol de "pensador"del juego, mientras que la maquina intenta adivinar el numero.

Keywords: algoritmo, escritura formal, secuencia, dividir y vencer.

1. Análisis del problema

1.1. Interacción inicial

El usuario debe poder indicar el rango en el cual se quiere jugar el juego.

1.2. Dividir y vencer

El algoritmo debe abordar el dividir y vencer para lograr el objetivo de dividir el rango de posibles números hasta llegar al numero deseado.

Debe presentar un numero posible y el usuario debe indicar si el numero es mayor, menor o si el numero fue adivinado, basándose en estas acciones el algoritmo debe acercarse cada vez mas a la respuesta.

1.3. Interacción

El rol del pensador (usuario) debe pensar en un numero natural antes de que el juego sea iniciado.

El pensador debe dar las pistas si el numero pensado es mayor o menor cada vez que el algoritmo adivine un numero.

1.4. Ciclo recursivo

Necesariamente, el programa debe mantenerse en un ciclo recursivo hasta que el numero sea encontrado o en su defecto la maquina ser derrotada.

⋆Este documento presenta la escritura formal de un algoritmo.

Email address: afelipe-vaca@javeriana.edu.co (Andrés Felipe Vaca Lago)

1.5. Interfaz de usuario

La interfaz de usuario debe de una forma intuitiva mostrar las acciones posibles por parte del usuario.

En cada uno de los intentos debe mostrar información relevante en cuanto al juego como el numero a adivinar y el numero de intentos cuando el numero haya sido eliminado.

2. Diseño del problema

El problema inicial se abordará de la siguiente manera:

1. Se consulta el rango de valores en el que se quiere jugar el juego.
2. Se preguntan números hasta que el algoritmo logre .[a]divinar.[el] numero que el usuario esta pensando.
3. Mediante el algoritmo de búsqueda binaria el algoritmo va acortando las opciones del rango de números posibles según la respuesta del usuario en cada pregunta.
4. Cuando el algoritmo mediante dividir y vencer logre dar con dicho numero, imprimirá el numero que se adivinó y el numero de intentos en el que se adivinó.

3. Algoritmos de solución

3.0.1. Algoritmo de dividir y reinar en Pseudo-código

```
procedure busquedaBinariaRecursiva(numeros, objetivo, bajo, alto)
    if bajo > alto then
        return -1  // No encontrado
    end if

    medio ← bajo + (alto - bajo) ÷ 2

    if numeros[medio] = objetivo then
        return medio
    elif numeros[medio] < objetivo then
        return busquedaBinariaRecursiva(numeros, objetivo, medio + 1, alto)
    else
        return busquedaBinariaRecursiva(numeros, objetivo, bajo, medio - 1)
    end if
end procedure

procedure busquedaBinaria(numeros, objetivo)
    return busquedaBinariaRecursiva(numeros, objetivo, 0, length(numeros) - 1)
end procedure
```

3.0.2. Algoritmo de interfaz

```
procedure jugarJuego(const vector<int>& numeros, int bajo, int alto)
    if bajo > alto then
        print << "El número no fue adivinado." << endl
        return
    end if

    int medio ← bajo + (alto - bajo) ÷ 2

    print << "¿Tu número es mayor (1), menor (2) o igual (3) a " << numeros[medio] << "? "
    int intento
    get >> intento

    if intento = 1 then
        jugarJuego(numeros, medio + 1, alto)
    else if intento = 2 then
        jugarJuego(numeros, bajo, medio - 1)
    else if intento = 3 then
        print << "El número fue adivinado." << endl
    else
        print << "Entrada inválida. Saliendo." << endl
    end if
end procedure
```

3.0.3. Análisis de complejidad

Ya que en ambos casos se utiliza una búsqueda binaria, ambos algoritmos tienen una complejidad de $O(\log n)$.

4. Implementación

En este caso, se decidió utilizar el lenguaje c++ para la implementación de los algoritmos.

4.1. Código

```
#include <iostream>
#include <vector>

using namespace std;

int busquedaBinariaRecursiva(const vector<int>& numeros, int objetivo, int bajo, int alto) {
    if (bajo > alto) {
        return -1; // No encontrado
    }
```

```
    int medio = bajo + (alto - bajo) / 2;

    if (numeros[medio] == objetivo) {
        return medio;
    } else if (numeros[medio] < objetivo) {
        return busquedaBinariaRecursiva(numeros, objetivo, medio + 1, alto);
    } else {
        return busquedaBinariaRecursiva(numeros, objetivo, bajo, medio - 1);
    }
}

int busquedaBinaria(const vector<int>& numeros, int objetivo) {
    return busquedaBinariaRecursiva(numeros, objetivo, 0, numeros.size() - 1);
}

void jugarJuego(const vector<int>& numeros, int bajo, int alto) {
    if (bajo > alto) {
        cout << "El número no fue adivinado." << endl;
        return;
    }

    int medio = bajo + (alto - bajo) / 2;
    int intentoCounter = 1;  // Counter for attempts

    while (true) {
        cout << "¿Tu número es mayor (1), menor (2) o igual (3) a " << numeros[medio] << "? ";
        int intento;
        cin >> intento;

        if (intento == 1) {
            bajo = medio + 1;
        } else if (intento == 2) {
            alto = medio - 1;
        } else if (intento == 3) {
            cout << "El número fue adivinado en " << intentoCounter << " intentos." << endl;
            break;
        } else {
            cout << "Entrada inválida. Saliendo." << endl;
            return;
        }

        medio = bajo + (alto - bajo) / 2;
        intentoCounter++;
    }
}
```

```
int main() {
    int eleccion;
    cout << "Elige el rango del número:" << endl;
    cout << "1. 1 a 100" << endl;
    cout << "2. 1 a 1000" << endl;
    cout << "3. 1 a 10000" << endl;
    cout << "Ingresa tu elección (1, 2 o 3): ";
    cin >> eleccion;

    int rango;
    switch (eleccion) {
        case 1:
            rango = 100;
            break;
        case 2:
            rango = 1000;
            break;
        case 3:
            rango = 10000;
            break;
        default:
            cout << "Elección inválida. Saliendo." << endl;
            return 1;
    }

    vector<int> numeros;
    for (int i = 1; i <= rango; ++i) {
        numeros.push_back(i);
    }

    jugarJuego(numeros, 0, numeros.size() - 1);

    return 0;
}
```

5. Utilización

Acá podemos ver algunos ejemplos de uso del código y su desempeño adivinando los números:

5.1. Para adivinar el numero 34

```
Output:
Elige el rango del número:
1. 1 a 100
```

```
2. 1 a 1000
3. 1 a 10000
Ingresa tu elección (1, 2 o 3): 1
¿Tu número es mayor (1), menor (2) o igual (3) a 50? 2
¿Tu número es mayor (1), menor (2) o igual (3) a 25? 1
¿Tu número es mayor (1), menor (2) o igual (3) a 37? 2
¿Tu número es mayor (1), menor (2) o igual (3) a 31? 1
¿Tu número es mayor (1), menor (2) o igual (3) a 34? 3
El número fue adivinado en 5 intentos.
```

5.2. Para adivinar el numero 254

```
Output:
Elige el rango del número:
1. 1 a 100
2. 1 a 1000
3. 1 a 10000
Ingresa tu elección (1, 2 o 3): 2
¿Tu número es mayor (1), menor (2) o igual (3) a 500? 2
¿Tu número es mayor (1), menor (2) o igual (3) a 250? 1
¿Tu número es mayor (1), menor (2) o igual (3) a 375? 2
¿Tu número es mayor (1), menor (2) o igual (3) a 312? 2
¿Tu número es mayor (1), menor (2) o igual (3) a 281? 2
¿Tu número es mayor (1), menor (2) o igual (3) a 265? 2
¿Tu número es mayor (1), menor (2) o igual (3) a 257? 2
¿Tu número es mayor (1), menor (2) o igual (3) a 253? 1
¿Tu número es mayor (1), menor (2) o igual (3) a 255? 2
¿Tu número es mayor (1), menor (2) o igual (3) a 254? 3
El número fue adivinado en 10 intentos.
```

5.3. Para adivinar el numero 3956

```
Output:

Elige el rango del número:
1. 1 a 100
2. 1 a 1000
3. 1 a 10000
Ingresa tu elección (1, 2 o 3): 3
¿Tu número es mayor (1), menor (2) o igual (3) a 5000? 2
¿Tu número es mayor (1), menor (2) o igual (3) a 2500? 1
¿Tu número es mayor (1), menor (2) o igual (3) a 3750? 1
¿Tu número es mayor (1), menor (2) o igual (3) a 4375? 2
¿Tu número es mayor (1), menor (2) o igual (3) a 4062? 2
¿Tu número es mayor (1), menor (2) o igual (3) a 3906? 1
¿Tu número es mayor (1), menor (2) o igual (3) a 3984? 2
¿Tu número es mayor (1), menor (2) o igual (3) a 3945? 1
```

```
¿Tu número es mayor (1), menor (2) o igual (3) a 3964? 2
¿Tu número es mayor (1), menor (2) o igual (3) a 3954? 1
¿Tu número es mayor (1), menor (2) o igual (3) a 3959? 2
¿Tu número es mayor (1), menor (2) o igual (3) a 3956? 3
El número fue adivinado en 12 intentos.
```

La secuencia de entrada X debe ser representada con alguna estructura de datos lineal que se pueda iterar de forma, al menos, unidireccional. El tipo de dato de esta secuencia debe soportar los operadores aritméticos necesarios.