Integrantes: Miguel Tomillo, Josué Mosquera, Jose Heredia

Curso: Cuarto

Paralelo: A

Algoritmo Torres de Hanoi

La torre de Hanoi en un problema matemático en el cual unas piezas deben ser trasladadas de una torre a otra con una cierta cantidad de combinaciones posibles, este problema puede ser traducido a código ya que para poder resolverlo se necesita un algoritmo recursivo, gracias a eso es que en el mundo de la programación es muy comúnmente planteada.

Algoritmo Torres de Hanói (Complejidad $\Theta(2^n - 1)$)

Entrada: Tres pilas de números *origen*, *auxiliar*, *destino*, con la pila *origen* ordenada

Salida: La pila *destino*

1. **si** *origen* $==\{1\}$ **entonces**
    1. **mover** el disco 1 de pila origen a la pila destino (insertarlo arriba de la pila destino)
    2. **terminar**
2. **si no**
    1. *hanoi*($\{1,..,n-1\}$,*origen*,*destino*, *auxiliar*) //mover todas las fichas menos la más grande (*n*) a la varilla auxiliar
3. **mover** disco *n* a *destino* //mover la ficha grande hasta la varilla final
4. *hanoi* (*auxiliar*, *origen*, *destino*) //mover todas las fichas restantes, 1...*n*–1, encima de la ficha grande (*n*)
5. **terminar**

Algoritmo de Hanói en lenguaje simple.

Durante la clase de Desarrollo móvil del día jueves 16 de diciembre de 2021 se encomendó de realizar este algoritmo con la diferencia que en esta ocasión serían utilizados 64 discos, utilizando para ello el lenguaje de programación Dart, el ejercicio logro resolverse de la siguiente manera:

- Se estableció un *const* o una constante la cual está establecida en 64
- Se utilizan 3 variables llamadas: TorreA, TorreB y TorreC para representar los pilares en el juego.
- Se define un contador para calcular los movimientos necesarios para completar el objetivo.
- También se utiliza recursividad ya que al ir moviendo piezas es necesario para cumplirlo.

A continuación, se muestra de forma grafica la escritura del algoritmo dentro de Dart.

```
void main() {
  const numeroDiscos = 64;
  var torreA = 'A';
  var torreB = 'B';
  var torreC = 'C';
  var contador = 0;

  void move(int numeroDiscos, i, j) {
    contador++;
    print(" " +
        '$contador' +
        " Movimiento:" +
        '$numeroDiscos' +
        " Disco de número:" +
        i +
        " -> " +
        j);
  }

  void hanoi(int numeroDiscos, torreA, torreB, torreC) {
    if (numeroDiscos == 1) {
      move(1, torreA, torreC);
    } else {
      //Mueve recursivamente la capa n-1 sobre la torre A de A a travez de C a B
      hanoi(numeroDiscos - 1, torreA, torreC, torreB);
      move(numeroDiscos, torreA, torreC);
      // coloca recursivamente la capa n-1 en el pilar B desde B a travez de A hasta C
      hanoi(numeroDiscos - 1, torreB, torreA, torreC);
    }
  }

  hanoi(numeroDiscos, torreA, torreB, torreC);
}
```

El resultado total de los movimientos que serian necesario para completar el juego es una cantidad demasiado grande que la mayoría de sistemas no pueden mostrarla pero se resumiría en (2^64-1) o 1.8446744e+19, pero para comprobar el funcionamiento del algoritmo se utilizaran 4 discos.

```
void main() {
  const numeroDiscos = 4;
  var torreA = 'A';
  var torreB = 'B';
  var torreC = 'C';
  var contador = 0;

  void move(int numeroDiscos, i, j) {
    contador++;
    print(" " +
        '$contador' +
        " Movimiento:" +
        '$numeroDiscos' +
        " Disco de número:" +
        i +
        " -> " +
        j);
  }

  void hanoi(int numeroDiscos, torreA, torreB, torreC) {
    if (numeroDiscos == 1) {
      move(1, torreA, torreC);
    } else {
      //Mueve recursivamente la capa n-1 sobre la torre A de A a travez de C a B
      hanoi(numeroDiscos - 1, torreA, torreC, torreB);
      move(numeroDiscos, torreA, torreC);
      // coloca recursivamente la capa n-1 en el pilar B desde B a travez de A hasta C
      hanoi(numeroDiscos - 1, torreB, torreA, torreC);
    }
  }

  hanoi(numeroDiscos, torreA, torreB, torreC);
}
```

Run

Console

```
 2 Movimiento:2 Disco de número:A -> C
 3 Movimiento:1 Disco de número:B -> C
 4 Movimiento:3 Disco de número:A -> B
 5 Movimiento:1 Disco de número:C -> A
 6 Movimiento:2 Disco de número:C -> B
 7 Movimiento:1 Disco de número:A -> B
 8 Movimiento:4 Disco de número:A -> C
 9 Movimiento:1 Disco de número:B -> C
10 Movimiento:2 Disco de número:B -> A
11 Movimiento:1 Disco de número:C -> A
12 Movimiento:3 Disco de número:B -> C
13 Movimiento:1 Disco de número:A -> B
14 Movimiento:2 Disco de número:A -> C
15 Movimiento:1 Disco de número:B -> C
```

Documentation

info