

Instituto Tecnológico de Costa Rica

Escuela de Computación

Análisis de Algoritmos

Primer Proyecto Programado

Complejidad de algoritmos

Prof. Jorge Vargas

Estudiantes:

* Monserrath Alexander Crooks
* Jose Alejandro Aguirre Torrez
* Luis Antonio Mora Hernández

Fecha de entrega: 4/05/2017

# Torres de Hanoi

Las Torres de Hanoi es un [rompecabezas](https://es.wikipedia.org/wiki/Rompecabezas) o [juego matemático](https://es.wikipedia.org/w/index.php?title=Juego_matem%C3%A1tico&action=edit&redlink=1) inventado en [1883](https://es.wikipedia.org/wiki/1883) por el [matemático](https://es.wikipedia.org/wiki/Matem%C3%A1tico) [francés](https://es.wikipedia.org/wiki/Francia) [Édouard Lucas](https://es.wikipedia.org/wiki/Fran%C3%A7ois_%C3%89duard_Anatole_Lucas).Este [juego de mesa](https://es.wikipedia.org/wiki/Juego_de_mesa) solitario consiste en un número de discos de radio creciente que se apilan insertándose en una de las tres estacas de un tablero. El objetivo del juego es crear la pila en otra de las estacas siguiendo ciertas reglas. El problema es muy conocido en la [ciencia de la computación](https://es.wikipedia.org/wiki/Ciencia_de_la_computaci%C3%B3n) y aparece en muchos libros de texto como introducción a la teoría de [algoritmos](https://es.wikipedia.org/wiki/Algoritmo).

La fórmula para encontrar el número de movimientos necesarios para transferir *n* discos del poste A al poste C es: 2*n* - 1.

El juego, en su forma más tradicional, consiste en tres varillas verticales. En una de las varillas se apila un número indeterminado de discos (elaborados de [madera](https://es.wikipedia.org/wiki/Madera)) que determinará la complejidad de la solución, por regla general se consideran ocho discos. Los discos se apilan sobre una varilla en tamaño decreciente de abajo a arriba. No hay dos discos iguales, y todos ellos están apilados de mayor a menor radio -de la base de la varilla hacia arriba- en una de las varillas, quedando las otras dos varillas vacantes. El juego consiste en pasar todos los discos de la varilla ocupada (es decir la que posee la [torre](https://es.wikipedia.org/wiki/Torre)) a una de las otras varillas vacantes. Para realizar este objetivo, es necesario seguir tres simples reglas:

1. Sólo se puede mover un disco cada vez.
2. Un disco de mayor tamaño no puede descansar sobre uno más pequeño que él mismo.
3. Sólo puedes desplazar el disco que se encuentre arriba en cada varilla.

Existen diversas formas de realizar la solución final, todas ellas siguiendo estrategias diversas.

|  |
| --- |
| **Algoritmo** Torres de Hanói (Complejidad {\displaystyle \Theta (2^{n})}) |
|  |

Los números en vertical representan el tiempo de ejecución en segundos y los horizontales el tamaño de la entrada.

Torres de Hanoi en Python:

K = 1.01 : Valor de K más pequeño que cumple los resultados esperados.

# Búsqueda de Fibonacci

En [ciencia de la computación](https://es.wikipedia.org/wiki/Ciencia_de_la_computaci%C3%B3n), la técnica de búsqueda de Fibonacci es un método de búsqueda en un [array ordenado](https://es.wikipedia.org/w/index.php?title=Array_ordenado&action=edit&redlink=1) usando un [algoritmo de divide y vencerás](https://es.wikipedia.org/w/index.php?title=Algoritmo_de_divide_y_vencer%C3%A1s&action=edit&redlink=1) que disminuye las ubicaciones posibles con la ayuda de los [números de Fibonacci](https://es.wikipedia.org/wiki/N%C3%BAmeros_de_Fibonacci). Comparado con la [búsqueda binaria](https://es.wikipedia.org/w/index.php?title=Algoritmo_de_b%C3%BAsqueda_binaria&action=edit&redlink=1), Fibonacci busca las ubicaciones cuyas direcciones tienen poca dispersión. Por lo tanto, cuando los elementos se buscan, tiene un acceso a memoria no uniforme (el tiempo necesario para acceder a la ubicación de almacenamiento varía en dependencia de la ubicación previamente accedida), la búsqueda de Fibonacci tiene una ventaja sobre la búsqueda binaria en disminuir ligeremante el tiempo promedio necesario para acceder a la ubicación de almacenamiento.

La búsqueda de Fibonacci tiene complejidad *O*(log(*x*)) (Ver [notación de O Grande](https://es.wikipedia.org/w/index.php?title=Notaci%C3%B3n_de_O_Grande&action=edit&redlink=1)). La búsqueda de Fibonacci fue concebida por primera vez por [Kiefer](https://es.wikipedia.org/w/index.php?title=Kiefer&action=edit&redlink=1)(1953) como una búsqueda [minimax](https://es.wikipedia.org/wiki/Minimax) para el máximo (mínimo) de una función unimodal en un intervalo.



Complejidad del algoritmo:

Los números en vertical representan el tiempo de ejecución en segundos y los horizontales el tamaño de la entrada.

Búsqueda de Fibonacci en Python:

K = 1.0001 : Valor de K más pequeño que cumple los resultados esperados.

# Multiplicación de Matrices

En [matemática](https://es.wikipedia.org/wiki/Matem%C3%A1tica), la multiplicación o producto de matrices es la [operación](https://es.wikipedia.org/wiki/Operaci%C3%B3n_matem%C3%A1tica) de [composición](https://es.wikipedia.org/wiki/Funci%C3%B3n_compuesta) efectuada entre dos [matrices](https://es.wikipedia.org/wiki/Matriz_%28matem%C3%A1tica%29), o bien la [multiplicación](https://es.wikipedia.org/wiki/Multiplicaci%C3%B3n) entre una matriz y un [escalar](https://es.wikipedia.org/wiki/Escalar_%28matem%C3%A1tica%29) según unas determinadas reglas.

Al igual que la multiplicación [aritmética](https://es.wikipedia.org/wiki/Aritm%C3%A9tica), su definición es instrumental, es decir, viene dada por un [algoritmo](https://es.wikipedia.org/wiki/Algoritmo_de_multiplicaci%C3%B3n) capaz de efectuar. El [algoritmo](https://es.wikipedia.org/wiki/Algoritmo) para la multiplicación matricial es diferente del que resuelve la multiplicación de dos números. La diferencia principal es que la multiplicación de matrices no cumple con la propiedad de [conmutatividad](https://es.wikipedia.org/wiki/Conmutatividad).

Complejidad del algoritmo:

Los números en vertical representan el tiempo de ejecución en segundos y los horizontales el tamaño de la entrada.

K = 1.0001 : Valor de K más pequeño que cumple los resultados esperados.

# Conclusiones

AL realizar las distintas pruebas en todos los programas se obtuvieron las siguientes observaciones:

* El ruido generado por las aplicaciones y demás procesos que están corriendo a la hora de utilizar los programas, pueden no afectar muy significativamente cuando las entradas son pequeñas, pero al ir creciendo, logró notarse que la diferencia entre los tiempos de ejecución era considerablemente.
* La diferencia entre los tiempos de ejecución entre los lenguajes Python y C son bastante notables, siendo C el que presenta un mejor rendimiento a la hora de realizar las operaciones.
* Es evidente que la forma de procesar los datos entre Python y C es completamente distinta, siendo Python el que agrega más “ruido interno” al realizar las distintas operaciones y los cálculos.