Estudio de Códigos - Nelson Rodríguez

# Código 1: Secuencias de Fibonacci

Este código presenta tres enfoques para calcular la serie de Fibonacci:

- Recursiva: simple de entender, aunque ineficiente; su aumento es exponencial.

- La memoización: utiliza caché para guardar los resultados y mejorar la eficiencia.

- Iterativa: es la más rápida y requiere menos memoria.

Se verifica que las tres producen los mismos resultados con valores bajos (n<10).

# Código 2: Valoración de la memoria y los tiempos

Cada versión de Fibonacci es analizada por este código en relación al tiempo que tarda en ejecutarse y la cantidad máxima de memoria empleada. Lo que los resultados sugieren es que:

- La recursiva, conforme n aumenta, consume mucho tiempo y memoria (por ejemplo: para n=37 puede tardar varios segundos).

- La versión con almacenamiento es rápida y necesita poca memoria.

- La versión iterativa ofrece rapidez y estabilidad, manteniendo una memoria casi constante.

Esto demuestra que la elección del algoritmo influye en el rendimiento.

# Código 3: Gráfico comparativo hecho con Plotly

Este código, utilizando Plotly, presenta los resultados alcanzados en una gráfica. En el gráfico se observa que la versión recursiva utiliza recursos a un ritmo considerablemente más alto que las otras, que se mantienen casi constantes. La representación gráfica ayuda a comprender de forma más eficaz la diferencia entre las complejidades O(n) y O (log n).

# Resumen:

Los tres códigos ilustran de forma práctica cómo diferentes implementaciones de un mismo algoritmo pueden variar en cuanto a memoria y tiempo. Es esencial elegir la opción más adecuada, en función del problema y los recursos disponibles.

# Código log: Gráfico comparativo

El segundo gráfico emplea una escala logarítmica en sus ejes. Esto permite que la distinción entre los algoritmos sea más evidente: la búsqueda lineal crece en proporción al tamaño de la lista, mientras que la búsqueda binaria lo hace a un ritmo mucho más lento, siguiendo una tendencia logarítmica. Esta escala facilita observar que la búsqueda binaria mantiene su efectividad, incluso en listas extensa.

# Código: grafica 1°

Esta grafica se representa en la Notación Big -0: medición la complejidad en la cual esta se identifica con el esquema:

CUASILINEAL:

0(n Long n) - cuasilineal

Común en el algoritmo eficientes de ordenación.

Ya que en esta su curvatura es ascendente continua se podría confundir con la 0(long-0) –cuadrática ya que esta tiene a subir de manera vertical por lo contrario la cuasilineal esta asciende de manera progresiva.

Analizando un poco más afondo el esquema se puede identificar que esta de cae en cierto punto de la gráfica para ser más específicos en la parte de la numeración (32) esta deciende de manera irregular ya que necesita un proceso diferente y menos consume pero a su vez mayor tiempo de análisis.

# Código: grafica 2°

En esta grafica tiene cierta semejanza o similitud con la anterior lo que la diferencia es que cierto punto esta tiende a subir de manera vertical haciendo que esta no tome la misma identificación de grafica que su contra parte, por lo tanto en este esquema se identifica como:

0(Nˆ2) – Cuadrática.

El tiempo crece con el cuadrado del tamaño de entradas.

Analizando el esquema se puede ver que en cierto punto de ella esta hace un cambio drástico haciendo que su trayectoria sea más vertical.

Mirándola desde la parte del código se puede identificar que tuvo un consumo más de tiempo que las anteriores haciendo que tenga que hacer un incremento.