Instituto Politécnico Nacional
Escuela Superior de Cómputo


**Análisis de Algoritmos, Sem.: 2022-1, 3CV11, Práctica 6, 3 diciembre 2021**

# Práctica 6: Programación Dinámica

**Vanegas García Andrés, Vaquera Aguilera Ethan Emiliano.**

*avanegasg*1601@*alumno.ipn.mx*, *evaqueraa*@*alumno.ipn.mx*

**Resumen:** Para la practica en cuestión se realizo el análisis de un problema que incumbía en compara dos archivos con código fuente, el cual debemos obtener la subcadena comun mas larga y el porcentaje de recurrencia entre los dos códigos, para resolverlo se aplico la técnica de Programación dinámica y el algoritmo de la subsecuencia común mas larga . Después de haberlo resuelto se realiza el respectivo análisis a priori donde se profundiza sobre la aplicación de la técnica, funcionamiento y orden de complejidad .

**Palabras Clave**

**C:** C es un lenguaje de programación estructurado, usado por ya poco más de 52 años, esto brindándole un amplio repertorio de bibliotecas, además de un gran soporte de la comunidad que lo utiliza. C al ser un lenguaje de programación estructurado comparte similitudes con otros lenguajes desarrollados en la época además de que es un lenguaje de alto nivel, que de igual manera pude ser utilizado a bajo nivel para programar en ensamblador y rutinas de memoria.

**Complejidad de un Algoritmo:** Este es el punto en el que se mide la complejidad de resolución de un algoritmo por medio de una computadora, poniendo énfasis en el uso de la memoria de este sistema además de, el tiempo que este toma para desarrollarlo. Esta desde el inicio de la computación es una problemática que ha ido evolucionando a modo de que hoy en día se utiliza el análisis de algoritmos para so resolución y determinar la complejidad de un algoritmo en especifico.

**Análisis a priori y posterior:** Según lo comprendido en clase. El análisis a priori en computación es analizar un algoritmo, de manera formalmente teórica y matemáticamente, primero haciendo un análisis de sus posibles pseudocódigos y después en base a esto realizar una formulación del comportamiento de este en base a una función matemática.

Por otro lado el análisis a posterior, es la recolección de datos por medio de la experimentación, que en el caso de un algoritmo es la programación de este para así tomar una serie de pruebas en base a las veces ejecutadas y la cantidad de veces que este algoritmo tarda para terminar.

**Iteración:** Una iteración es la forma de hacer una serie de pasos repetidas veces hasta llegar al resultado esperado en un rango esperado sin sobrepasar este, las iteraciones pueden ir de cualquier manera en la vida real siendo como realizar un ejercicio matemático de una raíz, una potencia numérica o un logaritmo, y en computación serian las sentencias de while, for, do-while, etc.

**Recursividad:** La recursividad consiste mas que nada en la realización iterativa de un mismo lenguaje descriptivo, pero sin tener que mandar a llamar a una función iterativa. En palabras mas simples se trata de el desarrollo de una función que se llama a si misma, y después de llegar a un resultado esta regresa de vez en vez al inicio y finalmente otorgar el resultado final.

**Divide y vencerás:** Es una técnica para diseñar algoritmos que consiste en dividir el problema general en otros subproblemas de menor tamaño y del mismo tipo. Si un subproblema sigue siendo lo demasiado grande, esta técnica se emplea repetidas veces hasta que cada subproblema sea considerablemente pequeño para ser resuelto de una forma directa.

**Solución Factible :** Una solución es factible si satisface todas las restricciones sobre el problema en cuestión.

**Solución Óptima :** Una solución óptima es una solución factible que da un valor mas favorable, maximiza o minimiza una función objetivo.

**Programación Dinámica:** Es un método que resuelve problemas combinando las soluciones de subproblemas y guarda esa solución en una tabla, normalmente esta técnica se utiliza para resolver problemas de optimización.

# 1 Introducción

Programación dinámica es una técnica muy versátil que nos puede ayudar a resolver problemas de decisión en varios pasos, pero la realizad es que puede aplicarse en una gran variedad de problemas de diferentes tiempos. Con esta técnica podemos obtener una solución óptima realizando un sofisticado análisis sobre el problema en cuestión.

En esta practica le daremos una solución un problema un poco complicado, la comparación entre dos códigos fuente puede ser un poco complicado y tardado si uno lo realiza sin utilizar alguna herramienta que nos ayude a calcular un porcentaje de similitud. A lo largo de esta practica se ira desarrollando un algoritmo usando la técnica de programación dinámica, después realizaremos su implementación en un programa desarrollado en lenguaje C que muestre el tamaño de la subcadena común mas larga y una porcentaje de similitud entre los dos archivos de código fuente, consideran.

Uno de los objetivos de esta practica es visualizar un ejemplo de como se debe de implementar programación dinámica para resolver problemas de optimización. Además se puede observar la similitud y diferencia con otras técnicas como divide y vencerás, donde igualmente se divide el problema en subproblema o incluso su diferencia con greedy, una técnica que también se emplea para resolver problemas de optimización. Además, con su análisis a priori se mostrara mas detalladamente como se comporta una solución a este problema utilizando programación dinámica.

# 2 Conceptos Básicos

**Θ:** La sigla griega Teta (Θ) es la denotación que describe de manera clara, el comportamiento similar de las funciones matemáticas del peor y mejor caso, esto siendo una referencia clara para hacer distinguir de Ω y $O$, que tiene definiciones diferentes.

**Definición formal:** Θ(g(n))= $\{f(n) : existe\, n, c_1, c_2, > 0\, y\, n_0 > 0 \,||\, 0 \leq c_1 g(n \leq f(n) \leq c_2 g(n)\, \forall\, n \geq n_0\}$

**$O$:** Big $O$ es la denotación usada para el conjunto de funciones que se acotan por encima de f(n), describiendo de manera clara el comportamiento da la función del peor caso, y denota los puntos experimentales de esta también.

**Definición Formal:** $O(g(n)) = \{f(n) : existen, c > 0\, \&\, n_0 > 0\, constantes \,||\, 0 \leq f(n) \leq cg(n)\, \forall\, n \geq n_0\}$

**Ω:** La denotación Omega (Ω) es la denotación usada para acotar por abajo a la función f(n), siendo que es usada para describir al conjunto de funciones que están por debajo de la función experimental f(n), y de igual manera sirve para describir los puntos experimentales de igual manera.

**Definición Formal:** $\Omega(g(n)) = \{f(n) : existen, c > 0\, \&\, n_0 > 0\, constantes \,||\, cg(n) \leq f(n)\, \forall\, n \geq n_0\}$

**Recursividad:** La recursividad es la serie de procesos definidos, en la cual se quiere la sucesión de varios factores y en vez de usar una función de la clase iterativa, se usan los mismos procesos ya definidos para crear esa sucesión de procesos, Imaginemos una muñeca de Matryoshkas (tipo de muñecas rusas), cada que abrimos una para sacar una nueva estas son mas pequeñas y además, todas y cada una de de ellas son iguales, usando ese ejemplo podemos definir un poco mejor la función recursiva, donde cada vez que mandamos a llamar a la función los valores de esta pueden ir disminuyendo o aumentando, dependiendo de lo que se quiera solucionar, hasta llegar a una condición de frontera, y de igual manera a pesar de que los valores cambien, lo que nunca se modifica son los procesos de la función recursiva que se mantiene de la misma manera.

**Algorithm 1** Algoritmo Ejemplo Recursivo

```
function RECURSIVO(n)
    a ← n
    if a ≠ 0 then
        return Recursivo(a ← a − 1)
    else
        return a
    end if
end function
```

# 3 Experimentación y Resultados

Para la practica en cuestión se realizo un analizador de archivos, si lo deseamos llamar de esa manera. En principio el análisis del problema a resolver consistió en que se tienen dos archivos y se desea comparar estos dos para obtener la subsecuancia concurrente mas larga, para ello lo primero y único que obtenemos como entrada son los dos archivos que deseamos comprar, después de eso mediante el algoritmo LCS implementado para este programa, (un algoritmo de programación dinámica) deberemos de obtener la mayor subcadena que sea igual entre ambos archivos, siendo que al final del programa deberemos de obtener un porcentaje de recurrencia que nos dirá que tan parecidos son los archivos.
Como cuestiones especiales tenemos que el programa no toma en cuanta los espacios, esto por que lo que se desea comparar son las cadenas de caracteres para determinar si estas son parecidas, ya que el uso de espacios en el programa de primer instancia es muy complicado determinar los espacios de la lectura de un archivo y además de que el uso de los espacios puede incurrir en que el programa determine que la subsecuancia mas larga difiere por mas ya que estos son tomados en cuenta y provocan un error.

**Análisis a priori:** Para el análisis de la LCS solo se utilizo una breve cantidad de código que esta analizado a continuación en la imagen 1.

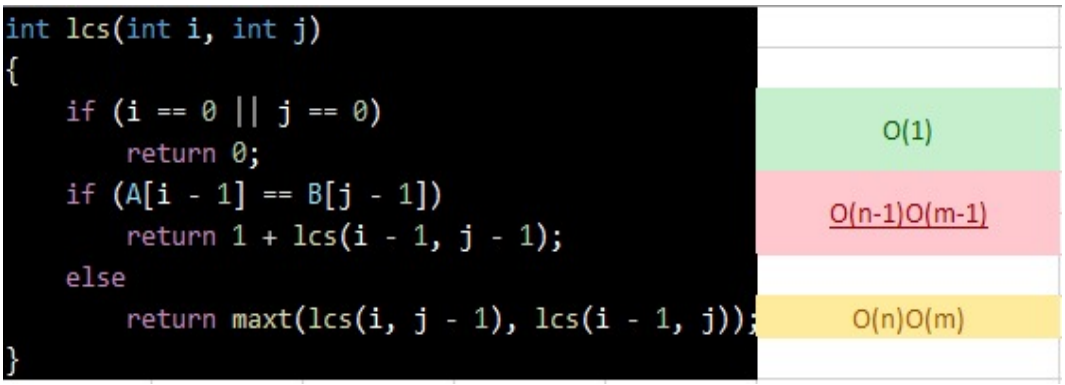



Figura 1. Análisis a priori algoritmo LCS recursivo

Como se nota en el código podemos ver una función que se comporta de manera contante pero esta se ejecuta una cantidad de n y m veces, la función en cuestión se encarga de la comparación de la subsecuencias mas grandes y retorna ya al final de que los caracteres no sean exactos iguales.

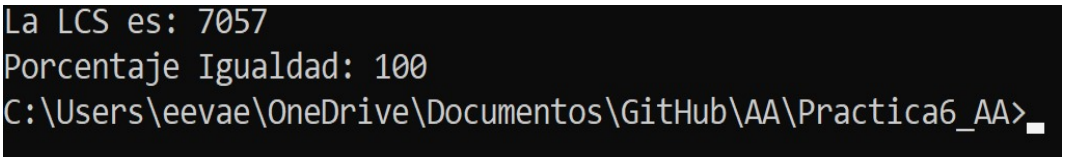



Figura 2. Ejemplo de ejecución del algoritmo LCS dinámico

Como podemos ver en la figura 2 tenemos la subsecuancia siguiente con el porcentaje de parecido, donde tenemos dos archivos exactamente iguales.
Y ahora en la figura 3 tenemos el ejemplo de dos que son diferentes.

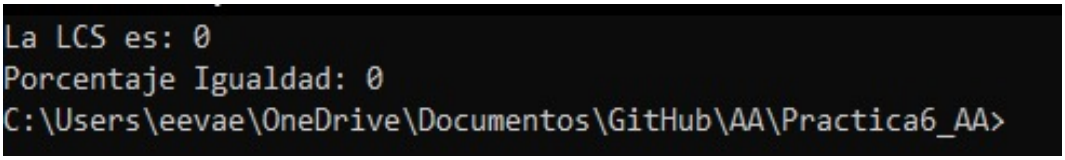



Figura 3. Ejemplo de ejecución del algoritmo LCS dinámico

Adjunto al programa estarán los archivos que se usaron para estas pruebas.

**Características PC Venegas Garcia Andres**

-Procesador Intel i5-10400f a 2.9 Ghz six core
-16 Gb de Ram
-Nvidia Geforce GTX 970

**Características PC Vaquera Aguilera Ethan Emiliano**

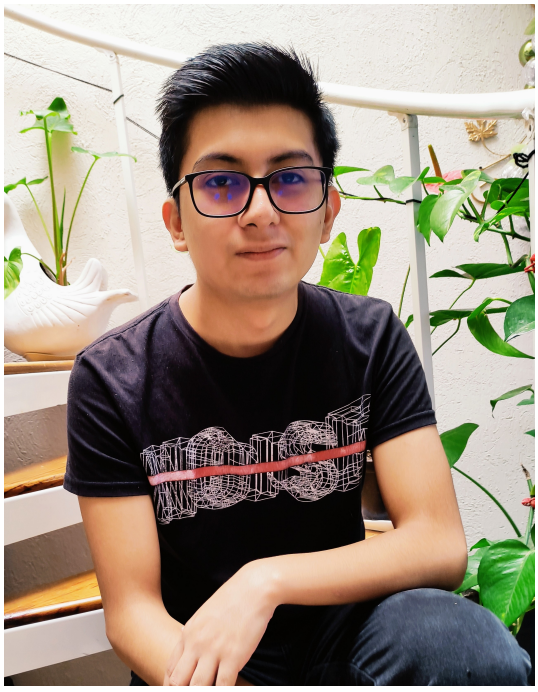
-Procesador ryzen 7-2700 a 3.2 Ghz octa core
-16 Gb de Ram
-AMD Radeon RX 5600XT

# 4 Conclusiones

**Conclusión General:** Aunque es un algoritmo sencillo y corto, se puede resolver un problema que puede ser largo sin la utilización de una herramienta como esta. Esto demuestra que esta técnica puede ser fácil implementarla y utilizar muy poco código, el verdadero reto esta en el análisis les problema que deseamos solucionar. Además esta practica permite observar diferencias que tiene con algunas otras técnicas.

**Conclusión Individual Vanegas García Andrés:** Durante el desarrollo de esta practica pude reafirmar conocimientos teóricos que aprendimos en clase con la investigación e implementación de esta técnica, aunque este algoritmo fue sencillo tiene una complejidad el resolverlo y analizarlo. También se pudo observar un poco la diferencia que tiene con otras técnicas como greedy o divide y vencerás.

Figura 3. Vanegas García Andrés



**Conclusión Individual Vaquera Aguilera Ethan Emiliano:** Por mi lado la practica se me hizo interesante ya que esta es fue sencilla hasta cierto puto donde si en efecto me tope con problemas de implementación que son algo difíciles de soluciona, pero lo demás de como es que funciona un algoritmo de este tipo es sencillo pero muy complicado al mismo tiempo ya que requiere mucha lógica antes de entrar en temas de implementación.


Figura 4. Vaquera Aguilera Ethan Emiliano

### 4.1 Anexos

## 5 Bibliografía

programmerclick.com. (2021). Complejidad temporal y complejidad espacial. 2021, de programmerclick.com Sitio web: https://programmerclick.com/article/89671544092/

Agrawal, Manindra; Kayal, Neeraj; Saxena, Nitin: "PRIMES is in P". Annals of Mathematics 160 (2004), no. 2, pp. 781–793.
J.M.+Gimeno+y+J.L.+González. (2021). Recursividad. 2021, de web Sitio web: http://ocw.udl.cat/enginyeria-i-arquitectura/programacio-2/continguts-1/2-recursividad.pdf
-. (2021). Algoritmo Greedy. 2021, de Universidad de Granada Sitio web: https://elvex.ugr.es/decsai/algorithms/slides/4
Maurette, M., Ojea, I. (2006). Programación Dinámica. Universidad de Buenos Aires Sitio Web: http://cms. dm. uba. ar/materias/1ercuat2009/optimizacion/Maurette$_Oje$