



INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

SEMESTRE 2023-1

PRACTICA 6

GRUPO: 3CV11

MATERIA: ANÁLISIS DE ALGORITMOS

ALUMNOS:

ISAAC SÁNCHEZ VERDIGUEL

ISANCHEZV1603@ALUMNO.IPN.MX

AXEL TREVIÑO PALACIOS

ATREVINOP1500@ALUMNO.IPN.MX

INSTITUTO POLITÉCNICO NACIONAL



ESCOM

MAESTRO:

BENJAMIN LUNA BENOSO

31 Diciembre 2022

Índice general

1	Introducción	2
1.1	Resumen	2
1.2	Introducción	2
2	Desarrollo	3
2.1	Conceptos Básicos	3
2.2	Algoritmo LCS	4
2.2.1	Algoritmo LCS	4
3	Experimentación y Resultados	5
3.1	Algoritmo: LCS	5
3.1.1	Análisis a Priori	5
3.1.2	Análisis a Posteriori	6
3.2	Pantallas de Ejecución del Algoritmo	7
4	Conclusiones	8
4.1	Conclusiones Generales	8
4.2	Isaac Sánchez - Conclusiones	9
4.3	Axel Trevino - Conclusiones	10

Índice de figuras

3.1	Análisis a Priori: LCS	5
3.2	Análisis a Posteriori: LCS	6
3.3	Ejecución de LCS	7
4.1	Isaac Sánchez	9
4.2	Axel Treviño	10

1 | Introducción

1.1. Resumen

La práctica consta de implementar un algoritmo que permita comparar dos archivos de texto plano mediante **Programación Dinámica** haciendo uso del algoritmo de la subsecuencia común más larga. Siendo desarrollada en un ambiente de programación con **Python** y **Linux**.

Palabras Clave: Python, Programación Dinámica, Recursividad.

1.2. Introducción

Los algoritmos son una parte fundamental de la ciencia de la computación, ya que estos al ser computables pueden dar solución o una idea más concreta acerca de la solución de un problema.

Un algoritmo no siempre dará una solución correcta, lo cual jamás será malo, porque esto nos ayudará a poder minimizar su radio de error. Una característica casi obligatoria para el buen funcionamiento de un algoritmo es su **rendimiento y eficacia**. El rendimiento adecuado se encuentra en la solución más rápida y menos costosa [Cormen, 2009].

Un problema desarrolla complicaciones debido al tiempo y el uso de sus recursos, una alternativa para poder amplificar soluciones es la Programación Dinámica que se inspira del camino más corto dentro de una solución óptima utilizando técnicas como la subdivisión de problemas reduciendo de esta manera el tiempo de ejecución [Wikipedia, 2022]. En esta práctica se analiza principalmente este paradigma mientras se analiza un algoritmo que permita el análisis de la subsecuencia común más larga de dos archivos planos de texto que contienen cadenas de texto.

2 | Desarrollo

2.1. Conceptos Básicos

La **complejidad temporal**, dentro del análisis de algoritmos, es el número de operaciones que ejecuta un algoritmo en cierto tiempo. Su denotación es $T(n)$ y puede ser analizada mediante dos tipos de análisis:

- Análisis de priori: entrega una función que muestra el tiempo de cálculo de un algoritmo.
- Análisis a posteriori: es la prueba en tiempo real del algoritmo, midiendo su costo mediante valores de entrada.

El análisis de complejidad temporal define que un algoritmo alcanza su máximo potencial cuando los valores de entrada son mayores al tiempo estimado de ejecución, siendo que es factible poder completar sus ejecuciones en menor tiempo posible.

Programación Dinámica La programación dinámica es un proceso algorítmico que los informáticos y programadores emplean para abordar las dificultades de optimización. Cuando la programación dinámica se incorpora, el algoritmo utilizado para abordar problemas de codificación difíciles los descompone en subproblemas. Una solución optimizada para cada subcuestión puede entonces aplicarse a todo el escenario, dependiendo del tipo de solución que obtengan de cada subcuestión del código. Además, la programación dinámica optimiza la recursividad simple con las soluciones recursivas que los programadores obtienen mediante los cálculos de los subproblemas del problema [Historiadelaempresa, 2022].

Subsecuencia Común Más Larga El problema de la subsecuencia común más larga (LCS) es encontrar la subsecuencia más larga presente en dos secuencias dadas en el mismo orden, es decir, encontrar la secuencia más larga que se puede obtener de la primera secuencia original eliminando algunos elementos y de la segunda secuencia original eliminando otros elementos [TECHIE DELIGHT, 2022].

2.2. Algoritmo LCS

2.2.1. Algoritmo LCS

Pseudocódigo Algoritmo LCS

El algoritmo lo realiza una comparación de dos archivos que contiene cada uno una cadena de caracteres en el cual se busca la Subsecuencia Común Más Larga (LCS) de forma que se llaman los archivos y se obtiene la sentencia, calculando de igual manera su longitud. Al ser llamada la función hace una comparación de longitudes para verificar que no sea cero (en caso de ser cero se retorna un cero), si existe una secuencia se acorta la misma eliminando el último elemento de la secuencia y repitiendo el proceso de forma recursiva, en caso contrario se encuentra el máximo de ambas y arroja el resultado.

Algorithm 1: LCS

Result: *LCS*

```
if  $len1 == 0$  or  $len2 == 0$  then  
    return 0;  
else  
    if  $sec1[len1-1] == sec2[len2-1]$  then  
        return  $1 + lcs(sec1, sec2, len1-1, len2-1)$ ;  
    else  
        return  $\max(lcs(sec1, sec2, len1, len2-1), lcs(sec1, sec2, len1-1, len2))$ ;
```

3 | Experimentación y Resultados

Aquí se presentaran los resultados del **Análisis a Priori y Posteriori** del algoritmo de LCS.

3.1. Algoritmo: LCS

El algoritmo fue ejecutado en el lenguaje de programación **Python** en un entorno de **Linux**. A continuación se muestra el análisis de priori y posteriori.

3.1.1. Análisis a Priori

La figura 3.1 presenta el análisis a priori realizado sobre el pseudocódigo del algoritmo de LCS. Concluyendo que el algoritmo presenta $T(n) = \theta(n * m)$ simplificando a $\theta(n^2)$

Algorithm 1: LCS

Result: *LCS*

```
if len1 == 0 or len2 == 0 then —————  $\Theta(1)$ 
|   return 0;
else
|   if sec1[len1-1] == sec2[len2-1] then
|       return 1 + lcs(sec1, sec2, len1-1, len2-1);
|   else
|       return max(lcs(sec1, sec2, len1, len2-1), lcs(sec1, sec2, len1-1, len2)); ———  $\Theta(m*n)$ 
```

Figura 3.1: Análisis a Priori: LCS

3.1.2. Análisis a Posteriori

En el análisis posteriori se verifica que el análisis a priori demostró que la complejidad del peor caso es $T(n) = \theta(n^2)$ dando un poco más la complejidad. En la figura 3.2 se muestra que el análisis a priori tuvo problemas al momento de ser planteado ya que la complejidad varió mínimamente.

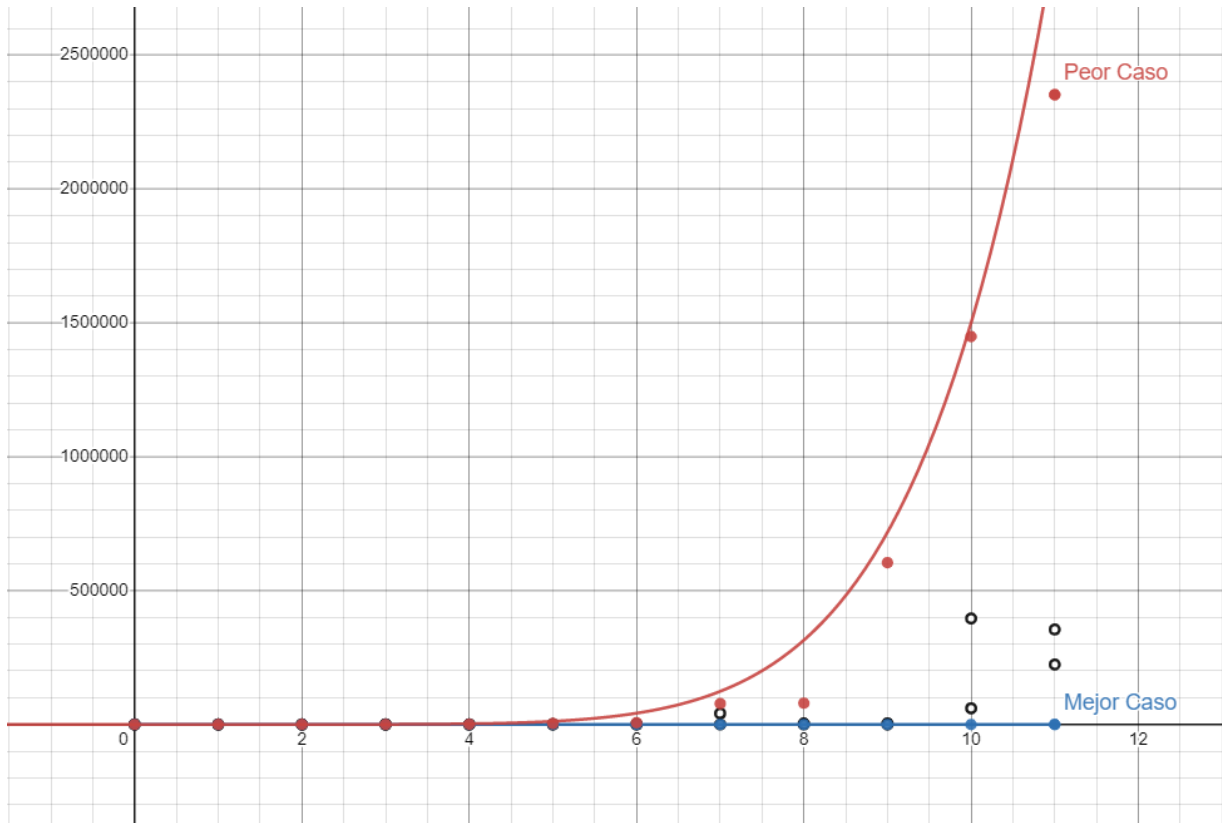
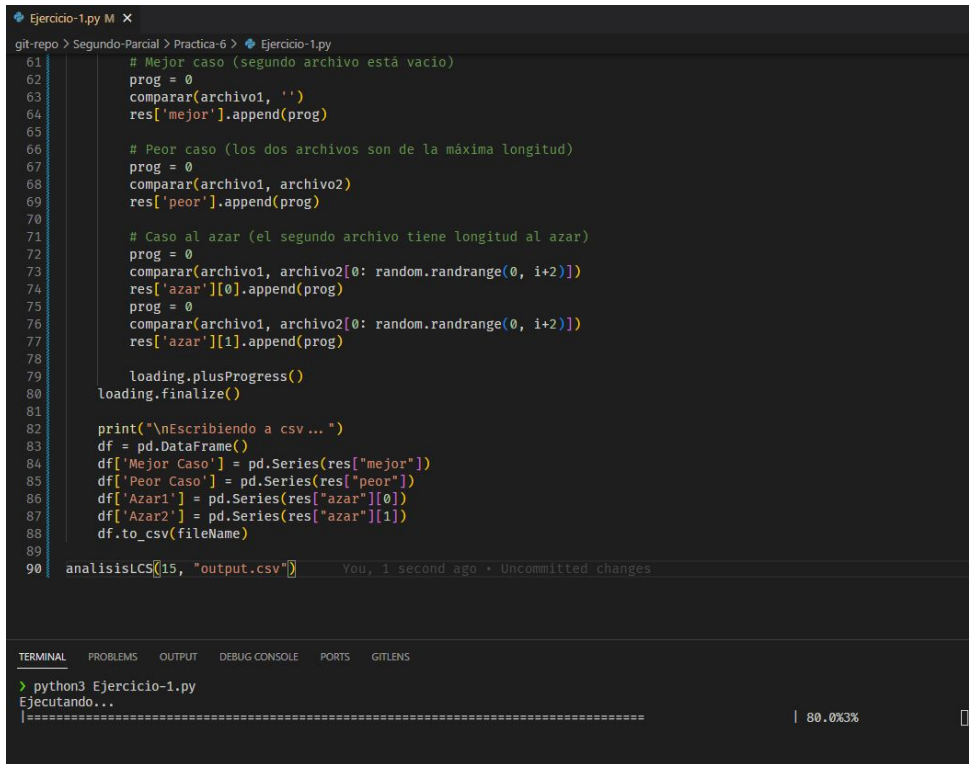


Figura 3.2: Análisis a Posteriori: LCS

3.2. Pantallas de Ejecución del Algoritmo

Se muestra en la figura 3.3 la ejecución del algoritmo, demostrando la velocidad del algoritmo.



```
Ejercicio-1.py M X
git-repo > Segundo-Parcial > Practica-6 > Ejercicio-1.py
61 # Mejor caso (segundo archivo está vacío)
62 prog = 0
63 comparar(archivo1, '')
64 res['mejor'].append(prog)
65
66 # Peor caso (los dos archivos son de la máxima longitud)
67 prog = 0
68 comparar(archivo1, archivo2)
69 res['peor'].append(prog)
70
71 # Caso al azar (el segundo archivo tiene longitud al azar)
72 prog = 0
73 comparar(archivo1, archivo2[0: random.randrange(0, i+2)])
74 res['azar'][0].append(prog)
75 prog = 0
76 comparar(archivo1, archivo2[0: random.randrange(0, i+2)])
77 res['azar'][1].append(prog)
78
79 loading.plusProgress()
80 loading.finalize()
81
82 print("\nEscribiendo a csv...")
83 df = pd.DataFrame()
84 df['Mejor Caso'] = pd.Series(res["mejor"])
85 df['Peor Caso'] = pd.Series(res["peor"])
86 df['Azar1'] = pd.Series(res["azar"][0])
87 df['Azar2'] = pd.Series(res["azar"][1])
88 df.to_csv(fileName)
89
90 analisisLCS(15, "output.csv") You, 1 second ago • Uncommitted changes

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE PORTS GITLENS
> python3 Ejercicio-1.py
Ejecutando...
|=====| 80.0%3%
```

Figura 3.3: Ejecución de LCS

4 | Conclusiones

4.1. Conclusiones Generales

Pese a ser una práctica sencilla, tuvimos problemas dentro del análisis a Priori, siendo que no entendimos de forma correcta el analizar un algoritmo que entre en el paradigma de la Programación Dinámica, creemos que es fue nuestro único punto del cuál tuvimos más problemas. La teoría de la Programación Dinámica es eficiente para poder tratar con problemas que no se encuentra una solución rápida. Nos resultó de ayuda el algoritmo de la subsecuencia común más larga para poder entender su funcionamiento.

4.2. Isaac Sánchez - Conclusiones

El algoritmo de la subsecuencia común más larga me pareció de forma inteligente una buena introducción para el desarrollo de soluciones que requieran una eficacia mejor. Mi problema surgió en el análisis del algoritmo, que fue resuelto después de verificar diferentes fuentes en las cuales me apoye para dar una conclusión más certera a mi análisis.



Figura 4.1: Isaac Sánchez

4.3. Axel Trevino - Conclusiones

No veo diferencias entre divide y vencerás y programación dinámica, pero al menos tiene un nombre bonito



Figura 4.2: Axel Treviño

Bibliografía

- [Cormen, 2009] Cormen, T. H. (2009). *Introduction to Algorithms*. The MIT Press, London.
- [Historiadelapresa, 2022] Historiadelapresa (2022). ¿qué es la programación dinámica? (métodos y ejemplos). <https://historiadelapresa.com/programacion-dinamica>. [Online; accessed 27-December-2022].
- [TECHIE DELIGHT, 2022] TECHIE DELIGHT (2022). Problema de la subsecuencia común más larga. <https://www.techiedelight.com/es/longest-common-subsequence/>. [Online; accessed 27-December-2022].
- [Wikipedia, 2022] Wikipedia (2022). Programacion dinamica. https://es.wikipedia.org/wiki/Programaci%C3%B3n_din%C3%A1mica. [Online; accessed 27-December-2022].