Proyecto 1
Programación dinámica y voraz
Análisis de algoritmos II

Carlos Andres Delgado S, Msc
carlos.andres.delgado@correounivalle.edu.co

Septiembre 2024

Proyecto 1: Programación dinámica y voraz

En este proyecto deberá resolver dos problemas utilizando:

1. Estrategia por fuerza bruta (solución ingenua)
2. Estrategia de programación dinámica
3. Estrategia de programación voraz

1. LA TERMINAL INTELIGENTE

Una terminal inteligente tiene la capacidad de cambiar una cadena de caracteres, desplegada en la pantalla, en otra. Los cambios de la cadena se hacen por medio de las cinco operaciones que se muestran en la siguiente tabla, cada operación tiene un costo asignado:

Operación	Descripción	Costo
advance	Mueve el cursor un carácter a la derecha	a
delete	Borra el carácter bajo el cursor y mueve el cursor al siguiente carácter	d
replace	Reemplaza el carácter bajo el cursor por otro y mueve el cursor una posición a la derecha	r
insert	Inserta un nuevo carácter antes del carácter que está bajo el cursor. El cursor se mantiene su posición	i
kill	Borra los caracteres desde el que está bajo el cursor, incluyéndolo, hasta el final de la línea	k

Considere el siguiente ejemplo donde se transforma la cadena `algorithm` a la cadena `altruistic`:

Operación	Cadena
-	algorithm
advance	algorithm

advance	algorithm	1
replace with t	altorithm	3
delete	altrithm	2
advance	altrithm	1
insert u	altruithm	2
advance	altruithm	1
insert s	altruisthm	2
advance	altruisthm	1
insert i	altruistihm	2
insert c	altruistichm	2
kill	altruistic	1

costo total = 5*5 +2+3+4*8+1 = 63

El costo de la transformación es $5a + d + r + 4i + k$. Pueden existir diferentes secuencias de las operaciones que producen una misma transformación y que tienen costo diferente.

1.1 Entender el problema

- Muestre dos soluciones que permitan transformar la cadena `ingenioso` en `ingeniero`. Indique el costo de cada solución.
- Muestre dos soluciones que permitan transformar la cadena `francesa` en `ancestro`. Indique el costo de cada solución.

1.2 Caracterizar la estructura de una solución óptima

Minimizar el número de operaciones costosas (como inserciones y eliminaciones).

Sea $x[1..n]$ la cadena a transformar en $y[1..n]$, caracterice la estructura de una solución óptima, esto es, indique cómo la solución óptima está compuesta de otras soluciones óptimas.

Utilizar la operación "kill" de manera estratégica para eliminar varios caracteres de una vez cuando sea beneficioso.

1.3 Definir recursivamente el valor de una solución óptima

Utilice la recurrencia. Explique con ejemplos cada uno de los valores que calcula para la matriz M. Puede suponer los siguientes costos para las operaciones: $a = 1$, $d = 2$, $r = 3$, $i = 2$, $k = 1$.

1.4 Calcular el valor de una solución óptima

- Explique cuál es la forma correcta de completar la matriz M. Indique en qué posición de la matriz quedará la solución al problema original.
- Desarrolle un algoritmo para calcular el valor de la solución óptima. Indique su complejidad.
- Implemente el algoritmo.

1.5 Construir una solución óptima

- Desarrolle un algoritmo que despliegue la secuencia de operaciones óptima que permite transformar una cadena $x[1..n]$ en $y[1..n]$. Indique su complejidad.
- Implemente el algoritmo. El programa debe permitir modificar los costos de las operaciones e ingresar las cadenas x y y.

1.3 Definir recursivamente el valor de una solución óptima

Utilice la recurrencia. Explique con ejemplos cada uno de los valores que calcula para la matriz M. Puede suponer los siguientes costos para las operaciones: $a = 1$, $d = 2$, $r = 3$, $i = 2$, $k = 1$.

1.4 Calcular el valor de una solución óptima

- Explique cuál es la forma correcta de completar la matriz M. Indique en qué posición de la matriz quedará la solución al problema original.
- Desarrolle un algoritmo para calcular el valor de la solución óptima. Indique su complejidad.
- Implemente el algoritmo.

Sea M[i][j] el costo mínimo para transformar los primeros i caracteres de la cadena origen en los primeros j caracteres de la cadena destino.
La recurrencia sería:
M[i][j] = min(
M[i-1][j-1] + costo_reemplazo si origen[i] != destino[j],
M[i-1][j-1] si origen[i] == destino[j],
M[i][j-1] + costo_insercion,
M[i-1][j] + costo_eliminacion,
min(M[i-k][j] + costo_kill) para k = 1 hasta len(origen) - i
)

dado este costo de transformacion : El costo de la transformación es 5a + d + r + 4i + k.

2. EL PROBLEMA DE LA SUBASTA PÚBLICA

El mecanismo utilizado por el gobierno para realizar las subastas de acciones es el siguiente:

- El gobierno pone en subasta un total de A acciones a un precio mínimo de B.
- Cada oferente indica el precio p_i a pagar por acción, el número mínimo m_i de acciones a comprar y el número máximo M_i que podría comprar. Cada oferta se puede ver como una tripleta $< p_i, m_i, M_i >$.
- El gobierno ofrece comprar las acciones que sobren a un precio B.
- Entre los diferentes oferentes, el gobierno selecciona cuántas acciones comprar a cada uno, lo cual se representa como una lista $X =< x_1, x_2, ..., x_k >$, de tal forma que $x_i = 0$ o $m_i \leq x_i \leq M_i$. La lista X además debe cumplir que $\sum_{i=1}^{k} x_i = A$, esto es, se deben vender todas las acciones.

El valor recibido por una asignación de acciones X es $vr(X)$, lo cual se define como $vr(X) = \sum_{i=1}^{k} x_i p_i$.

2.1 Entender el problema

Muestre dos asignaciones de las acciones para $A = 1000$, $B = 100$, $n = 2$, la oferta $< 500, 100, 600 >$, la oferta $< 450, 400, 800 >$ y la oferta del gobierno $< 100, 0, 1000 >$. Indique el valor vr para la solución.

2.2 Una primera aproximación

Considere el algoritmo que lista las posibles asignaciones:

$$< y_1, y_2, ..., \sum_{i=1}^{n-1} y_i = A >$$

donde $y_i \in \{0, M_i\}$, luego calcula el vr para cada asignación y selecciona la mejor.

- Utilice el algoritmo anterior para la entrada $A = 1000$, $B = 100$, $n = 4$, $< 500, 400, 600 >$, $< 450, 100, 400 >$, $< 400, 100, 400 >$, $< 200, 50, 200 >$, la oferta del gobierno $< 100, 0, 1000 >$.
- Indique si el algoritmo anterior siempre encuentra la solución óptima.

2.3 Caracterizar la estructura de una solución óptima

Caracterice la estructura de una solución óptima, esto es, indique la forma de los subproblemas y cómo la solución óptima la componen.

2.4 Definir recursivamente el valor de una solución óptima

Muestre la recurrencia que define el costo de la solución óptima.

2.5 Calcular el valor de una solución

- Por medio de un algoritmo, muestre cómo se calcula el costo de la solución óptima a través de subproblemas hasta llegar a dar la solución del problema original.
- Indique la complejidad del algoritmo.
- Implemente el algoritmo.

2.6 Construir una solución óptima

- Desarrolle un algoritmo que permita conocer la asignación de acciones $X =< x_1, x_2, ..., x_n, x_{n+1} >$ tal que $vr(X)$ sea máximo. Indique su complejidad.
- Implemente el algoritmo. El programa debe permitir ingresar los valores para A, B, n y cada oferta $< p_i, m_i, M_i >$.

1. Entrega

Se debe entregar un repositorio de Github en el enlace dispuesto en el campus vitual, teniendo como plazo máximo el 01 de Octubre de 2024 a las 23:59. El repositorio debe contener un archivo README.md con la información sobre los integrantes del grupo y cómo ejecutar el código generado. Además, se debe entregar un archivo PDF que contenga las respuestas a cada uno de los puntos planteados en el enunciado y las complejidades computacionales de las 3 soluciones: ingenua, dinámica y voraz.

En caso de ser privado el repositorio, se debe dar acceso al docente con su correo institucional.

Utilice una estructura sencilla que sea fácil de identificar cual es el código fuente de sus soluciones.

Se permiten entregas tardías pero con una penalización de 0.15 por hora o fracción, por ejemplo, si entrega 1 minuto después de la fecha límite, la nota máxima será de 4.85, si entrega 2 horas después, la nota máxima será de 4.70 y así sucesivamente.

El proyecto puede ser realizado en grupos de máximo de 4 personas, la cuales deben inscribirse en el campus virtual, hasta el día 24 de Septiembre de 2024. El enlace no le permitirá enviar la entrega si no realiza este paso.

2. Rubricas de evaluación

Criterio	**Nivel 0 (0 puntos)**	**Nivel 1 (5 puntos)**	**Nivel 2 (10 puntos)**	**Nivel 3 (15 puntos)**

Solución de fuerza bruta problema de Terminal Inteligente	No realiza el punto	La solución está incompleta o incorrecta	La solución es correcta, pero la estimación de la complejidad es errónea o falta	La solución por fuerza bruta soluciona correctamente el problema y la estimación de complejidad computacional es correcta
Solución de programación dinámica Terminal Inteligente	No realiza el punto	La subestructura óptima es incorrecta o falta	La subestructura óptima es correcta, pero la implementación no es fiel a la especificación	La solución por programación dinámica de Terminal Inteligente tiene una subestructura óptima correcta para el problema y la implementación es correcta y fiel a la especificación
Solución voraz Terminal Inteligente	No realiza el punto	La decisión voraz no es clara o la implementación es incompleta	La decisión voraz es correcta, pero la implementación no es fiel a la especificación	Se muestra claramente cuál es la decisión voraz, la implementación es totalmente funcional y fiel a la especificación

Solución de fuerza bruta problema de subasta pública	No realiza el punto	La solución está incompleta o incorrecta	La solución es correcta, pero la estimación de la complejidad es errónea o falta	La solución por fuerza bruta soluciona correctamente el problema y la estimación de complejidad computacional es correcta
Solución de programación dinámica subasta pública	No realiza el punto	La subestructura óptima es incorrecta o falta	La subestructura óptima es correcta, pero la implementación no es fiel a la especificación	La solución por programación dinámica de Terminal Inteligente tiene una subestructura óptima correcta para el problema y la implementación es correcta y fiel a la especificación
Solución voraz subasta pública	No realiza el punto	La decisión voraz no es clara o la implementación es incompleta	La decisión voraz es correcta, pero la implementación no es fiel a la especificación	Se muestra claramente cuál es la decisión voraz, la implementación es totalmente funcional y fiel a la especificación

Informe respuestas a las preguntas	No realiza el punto	Responde algunas preguntas, pero no de forma clara o precisa	Responde la mayoría de las preguntas, pero faltan algunos detalles o argumentos matemáticos	Se responden todas las preguntas de forma clara y concisa, utilizando notación matemática y argumentando correctamente de acuerdo con los problemas
Soporte de complejidad computacional	No realiza el punto	Se realizan pocas mediciones o no se gráfica los resultados	Se realizan mediciones y gráficos, pero no se explican correctamente o faltan casos significativos	Se realizan mediciones de tiempo con al menos 5 casos de crecimiento, se toman tiempos promedios con al menos 50 ejecuciones, se presentan gráficos que comparan la complejidad teórica con la experimental, y se discuten los resultados
Informe complejidades y evidencias	No realiza el punto	Estima las complejidades de forma incorrecta o incompleta	Estima correctamente algunas complejidades, pero faltan argumentos o evidencias claras	Se estiman correctamente todas las complejidades (ingenua, dinámica y voraz), se usan notación matemática y capturas de pantalla de ejemplos no triviales

Interfaz gráfica	No realiza el punto	La interfaz gráfica es incompleta o no funcional	La interfaz gráfica es funcional, pero no es fácil de usar o faltan algunas características	Se realiza una interfaz gráfica que permite seleccionar el problema, ingresar datos fácilmente, ejecutar las soluciones ingenua, voraz y dinámica, y mostrar las soluciones de manera clara

3. Evaluación

La calificación se hará en base al puntaje de las rubricas de evaluación. La calificación final se obtendrá de la siguiente forma:

$$\text{Calificación final} = \frac{\text{Puntaje obtenido}}{150} \times 5$$

En caso de que no se implemente algunos de los puntos o no se entregue el código fuente, máximo se calificará nivel 1 en todas. En el caso que el informe no se entrega se calificarán sobre nivel 2 como máximo. Así mismo, en caso de no entregar el soporte respectivo del análisis de complejidad computacional o su análisis respectivo, se calificará sobre nivel 2 como máximo.

Este proyecto debe ser sustentado en clase, la calificación de la sustentación es un valor entre 0 y 1 que se multiplica por la nota grupal, por ejemplo, si su factor es 0.8 y su grupo obtuvo 4.5 en el proyecto, su nota final será de 4.5*0.8=3.6.