Proyecto de curso

Subasta pública de acciones

Análisis de Algoritmos II

Escuela de Ingeniería de Sistemas y Computación

Profesor Juan Francisco Díaz Frias Profesor Jesús Alexander Aranda
Monitor Joan Sebastián Betancourt

Marzo de 2023

1. Introducción

El presente proyecto tiene por objeto verificar que los estudiantes han adquirido el siguiente resultado de aprendizaje: Aplica técnicas de algoritmos voraces y dinámicos, en la construcción de algoritmos, para solucionar problemas de naturaleza combinatoria. Para ello, los estudiantes deben demostrar que logran:

- Aplicar cada estrategia de diseño (fuerza bruta, voraz y programación dinámica) a un ejemplo práctico.
- Usar un enfoque de fuerza bruta para resolver un problema y determinar si la estrategia conlleva a una solución óptima.
- Usar un enfoque voraz para resolver un problema y determinar si la estrategia conlleva a una solución óptima.
- Usar programación dinámica para resolver un problema comprendiendo que dicha estrategia conlleva a una solución óptima.
- Reconocer las ventajas y desventajas de utilizar diferentes estrategias de diseño (fuerza bruta, voraz y programación dinámica).
- Comparar distintas estrategias (fuerza bruta, voraz y programación dinámica) para un problema dado a partir del análisis del respectivo análisis de complejidad y optimalidad.

Para ello el estudiante:

- Desarrolla un programa utilizando un lenguaje de programación adecuado para resolver en grupo un proyecto de programación planteado por el profesor.
- Escribe un informe de proyecto, presentando los aspectos más relevantes del desarrollo realizado, para que un lector pueda evaluar el proyecto.
- Desarrolla una presentación digital, con los aspectos más relevantes del desarrollo realizado, para sustentar el trabajo ante los compañeros y el profesor.

[1] Adaptado de un caso de la vida real

2. El problema de la Subasta Pública de Acciones[1]

2.1. El problema

El gobierno, en su política de privatizaciones, ha decidido sistematizar el proceso de venta de sus empresas de manera que este sea un proceso más rápido y transparente.

El mecanismo que el gobierno ha escogido es el de la *subasta pública de acciones*, consistente en determinar el número de acciones a la venta y el precio mínimo que se pide por cada acción, y colocarlas en subasta. La subasta consiste en que cada comprador, de manera secreta, comunica al gobierno el precio que quiere pagar por acción, un número máximo de acciones que puede comprar a ese precio, y un número mínimo de acciones por la cual haría el negocio. Una vez recibidas todas las ofertas, el gobierno debe decidir cuántas acciones le vende a cada comprador de manera que cumpla las restricciones del mismo, y que maximice el dinero obtenido en la operación. Si las ofertas no son suficientes para comprar todas las acciones, el gobierno compra al precio mínimo las acciones restantes.

2.2. Formalización

Sea A el número total de acciones a la venta y B el precio mínimo por acción. Una oferta es una tripleta (p, c, r) tal que $p \geq B \wedge 0 < c \leq A \wedge 0 \leq r \leq c$, donde p representa el precio por acción y c (respectivamente r) representa el número máximo (respectivamente, mínimo) de acciones que compraría el comprador.

Una asignación de acciones es una lista $X = [x_1, x_2, \ldots, x_k]$ tal que $x_i = 0 \vee r_i \leq x_i \leq c_i$ y tal que $\sum_{i=1}^{k} x_i = A$ (x_i es el número de acciones asignadas al comprador i de acuerdo a las restricciones de su oferta y todas las acciones son asignadas).

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

Entrada: A, B, n (número de ofertas de compradores), $(p_i, c_i, r_i), 1 \leq i \leq n+1$ (las n ofertas de los compradores ordenadas descendentemente en orden lexicográfico, y la oferta del gobierno $(B, A, 0)$)

Salida: Una asignación de acciones $X = [x_1, x_2, \ldots, x_n, x_{n+1}]$ tal que $vr(X)$ sea máximo.

2.3. ¿Entendimos el problema?

Suponga que tenemos la siguiente entrada:
$A = 1000, B = 100, n = 2,$
$[(500, 600, 100), (450, 800, 400), (100, 1000, 0)]$

- Describa cinco (5) asignaciones de acciones válidas y diferentes para esa entrada y calcule el valor recibido por cada una de ellas. Guarde los resultados en una tabla.
- Describa una asignación de acciones X tal que $vr(X)$ sea máximo.

3. El proyecto

Su proyecto consiste en explorar tres alternativas para solucionar el problema, una de fuerza bruta, una voraz y una usando programación dinámica, programarlas, analizarlas y hacer un informe al respecto.

3.1. Usando fuerza bruta

Considere el algoritmo que enumera todas las asignaciones de acciones de la forma

$$[y_1, y_2, \ldots, y_n, A - \sum_{i=1}^{n} y_i]$$

$$y_i \in \{0, c_i\} \wedge \sum_{i=1}^{n} y_i \leq A,$$

calcula el valor recibido por cada una de ellas y escoge la de valor máximo.

3.1.1. Entendimos el algoritmo

Enumere todas las asignaciones de acciones de la forma descrita para la entrada
$A = 1000, B = 100, n = 4,$
$[(500, 600, 400), (450, 400, 100), (400, 400, 100),$
$(200, 200, 50), (100, 1000, 0)]$
y calcule el valor recibido por cada una de ellas. Guarde los resultados en una tabla.

3.1.2. Complejidad

¿Cuál es el orden de complejidad del algoritmo? Argumente su respuesta.

3.1.3. Corrección

¿El algoritmo siempre da la respuesta correcta al problema? Argumente su respuesta.

3.2. Usando un algoritmo voraz

3.2.1. Describiendo el algoritmo

Describa un algoritmo voraz para abordar el problema (ustedes como grupo deciden su estrategia voraz; la idea es que sea la más sencilla posible que ojalá les permita encontrar el óptimo siempre o la mayoría de las veces)

3.2.2. Entendimos el algoritmo

Calcule la salida de su algoritmo para la entrada:

$A = 1000, B = 100, n = 4,$ $[(500, 600, 400), (450, 400, 100), (400, 400, 100),$ $(200, 200, 50), (100, 1000, 0)]$

y calcule el valor recibido la solución. ¿Es la solución óptima?

Describa al menos 4 entradas más, calcule la solución entregada por el algoritmo y verifique si es o no la óptima. Guarde los resultados en una tabla.

3.2.3. Complejidad

¿Cuál es el orden de complejidad del algoritmo? Argumente su respuesta.

3.2.4. Corrección

¿El algoritmo siempre da la respuesta correcta al problema? Argumente su respuesta. Si la respuesta es negativa, adicionalmente analice cuándo si y cuándo no da la respuesta correcta.

3.3. Usando programación dinámica

Una alternativa a las dos anteriores sería utilizar programación dinámica. Para hacerlo es necesario comprobar que el problema tiene la propiedad de subestructura óptima y dejarse guiar por ella.

3.3.1. Caracterizando la estructura de una solución óptima

Caracterice la estructura de una solución óptima y a partir de ella describa el conjunto de subproblemas para los cuáles será necesario calcular las soluciones óptimas. ¿Cuántos subproblemas hay?

3.3.2. Definiendo recursivamente el valor de una solución óptima

Defina recursivamente el costo de una solución óptima para cada subproblema definido en el punto anterior.

3.3.3. Describiendo el algoritmo para calcular el costo de una solución óptima

Describa el algoritmo que calcula el costo de una solución óptima al problema original, basado en el punto anterior.

3.3.4. Describiendo el algoritmo para calcular una solución óptima

Tome el algoritmo descrito en el punto anterior y adicione:

- Una estructura que permita almacenar datos para construir una solución óptima cuyo valor sea el calculado por el algoritmo.
- Un algoritmo que recorra esa estructura construyendo una solución de costo óptimo.

3.3.5. Complejidad

Complejidad en tiempo. Calcule la complejidad en tiempo del algoritmo en términos de n yA (o sea, contar el número de sumas que se realizan en el peor caso).

Complejidad en espacio. Calcule la complejidad en espacio del algoritmo en términos de n yA (o sea, contar el número de celdas que se utilizan en el peor caso).

¿Es útil en la práctica? Suponga que tiene un equipo que procesa 3×10^8 operaciones por minuto y que utiliza 4 bytes de almacenamiento por celda.

Si $A = 32 \times 10^6$ y $n = 2^{10}$ (datos del caso real):

- Mostrar que el tiempo que se tomará el equipo en resolver el problema es inaceptable en la práctica.

- Mostrar que el espacio que se tomará el equipo en resolver el problema es inaceptable en la práctica.

[**Sugerencia:**] Utilice las siguientes aproximaciones:
$n+1 \sim n,\ A+1 \sim A$
1 minuto $= \frac{1}{3^4 2^8 5^2}$ años
1 año $= 3^4 2^8 5^2$ minutos $= 518400$ minutos
2^{10} Bytes $= 1$ KB
2^{10} KB $= 1$ MB
2^{10} MB $= 1$ GB
10^3 MB ~ 1 GB

¿Qué hacer en la práctica? Para solucionar el problema anterior, alguien propuso que no se vendiera cualquier número de acciones sino por paquetes de 5000 acciones.

- Analice si el tiempo que se tomará el equipo en resolver el problema con esa modificación es aceptable en la práctica.
- Analice si el espacio que se tomará el equipo en resolver el problema con esa modificación es aceptable en la práctica.

3.4. Implementación

Cada grupo debe entregar el código correspondiente a:

- En el ambiente de programación de su preferencia, un programa que contenga una función por cada una de las soluciones diseñadas y analizadas (fuerza bruta, voraz y programación dinámica (original y modificada)). Cada función recibe A, B, n y $(p_i, c_i, r_i) 1 \leq i \leq n+1$ y devuelve una pareja $(X, vr(X))$ tal que $X = [x_1, x_2, \ldots, x_n, x_{n+1}]$ es la respuesta al problema. Esas funciones en su código deben llamarse `accionesFB, accionesV, accionesPD1` y `accionesPD2`.
- Usando la tecnología de su preferencia, una interfaz que permita leer entradas al problema desde un archivo de texto en un formato especificado, y escribir las salidas al problema en un archivo de texto en un formato especificado, así como visualizar las entradas, después de leerlas, y las salidas después de ser calculadas por cualquiera de las funciones solicitadas.

Todos los programas deben correr bajo un mismo ambiente; se debe entregar el programa **ejecutable** y el código fuente.

3.4.1. Formato de las entradas

Las entradas vendrán en un archivo de texto con $n+4$ líneas así:

```
A
B
n
p1,c1,r1
p2,c2,r2
.
.
.
pn,cn,rn
B,A,0
```

Es decir, la primera línea trae el número de acciones (un entero), la segunda el precio mínimo por acción (un entero), la tercera el número de compradores (un entero), las siguientes n líneas los tres enteros separados por comas, que representan las ofertas de cada comprador, y la última línea con la oferta del gobierno.

3.4.2. Formato de las salidas

Las salidas se deben producir en un archivo de texto con $n+2$ líneas así:

```
Costo
x1
x2
.
.
.
xn
resto
```

Es decir, la primera línea trae el costo de la solución, y las siguientes n líneas las acciones asignadas a cada oferente y la última línea el resto de acciones que son las asignadas al gobierno.

3.5. Entrega

La entrega se debe realizar vía el campus virtual en las fechas previstas para ello, por uno sólo de los integrantes del grupo.

La fecha de entrega límite es el 27 de abril de 2023 a las 23:59. La sustentación será en las sesiones del 4 y 9 de mayo de 2023.

Debe entregar un informe en formato pdf, los archivos fuente, un archivo Readme.txt que describa todos los archivos entregados y las instrucciones para ejecutar la aplicación. Todo lo anterior en un solo archivo empaquetado cuyo nombre contiene los apellidos de los autores y cuya extensión corresponde al modo de compresión. Por ejemplo Diaz.zip o Diaz.rar, o Diaz.tgz o ...

4. Sustentación y calificación

El trabajo debe ser sustentado por los autores en día y hora por definir. **La calificación del proyecto para el grupo** se hará teniendo en cuenta los siguientes criterios:

1. Informe (1/2) Debe responder a cada una de las solicitudes formuladas en el enunciado y debe contener al menos una sección dedicada a cada tipo de algoritmo implementado (fuerza bruta, voraz, programación dinámica (2)), además de una sección dedicada a comparar los resultados de las cuatro soluciones implementadas (para esto use tablas y diseñe casos de prueba y documente cómo los diseñó) y una sección dedicada a concluir sobre las ventajas y desventajas de usar en la práctica los diferentes enfoques.

2. Implementación (1/2). Esto quiere decir que el código entregado funciona por lo menos para las diferentes pruebas que tendremos para evaluarlo y corresponde con todo lo solicitado (incluyendo la interfaz de lectura y visualización solicitada, y la generación de casos de prueba)

En todos los casos la sustentación será pilar fundamental de la nota individual asignada a cada integrante del grupo. En la sustentación se demuestra la capacidad del grupo de navegar en el código y realizar cambios rápidamente en él, así como la capacidad de responder con solvencia a las preguntas que se le realicen.

Cada persona de cada grupo, después de la sustentación, tendrá asignado un número real (el factor de multiplicación) entre 0 y 1, correspondiente al grado de calidad de su sustentación. **Su nota definitiva será la calificación del proyecto para el grupo, multiplicada por ese valor.** Si su asignación es 1, su nota será la del proyecto. Pero si su asignación es 0.9, su nota será 0.9 por la nota del proyecto. La no asistencia a la sustentación tendrá como resultado una asignación de un factor de 0.

La idea es que lo que no sea debidamente sustentado no vale así funcione muy bien!!! Y que, del trabajo en grupo, es importante que todos aprendan, no sólo algunos.

Éxitos!!!