Proyecto de curso

Repartición Óptima de Cupos

Análisis de Algoritmos II

Escuela de Ingeniería de Sistemas y Computación

Profesor Jesús Alexander Aranda Monitor Mauricio Muñoz

Septiembre de 2025

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.

2. El problema de la repartición óptima de cupos

2.1. El problema

Un motivo para una frecuente, y aparentemente justa, insatisfacción estudiantil se presenta cada semestre entre los estudiantes que se matriculan al final, pues no encuentran cupos para las materias que quieren ver. Cada semestre aparecen nuevas estrategias para resolver las injusticias que se descubren de proceso en proceso: matricular por código de los antiguos a los más nuevos, matricular por código de los nuevos a los más antiguos, matricular por promedio general, ...; sin embargo siempre aparecen los insatisfechos, explicando con buenas razones por qué fue injusta la estrategia, y proponiendo otra manera más adecuada de repartir los cupos.

2.2. Formalización

Formalmente el problema de *la repartición óptima de cupos* se puede formular así:

1. **Sea:**
   - $P = \{1, 2, 3, 4, 5\}$, el conjunto de prioridades de materias disponibles.
   - $M = \{(M_i, m_i) : i = 1 \dots k\}$ el conjunto de materias disponibles, donde M_i es el código de la materia i y m_i es cupo de estudiantes para M_i.
   - $E = \{(e_j, ms_j) : j = 1 \dots r\}$, es el conjunto de las solicitudes de matriculación de materias por cada estudiante, donde e_j es el código del estudiante j, y $ms_j = \{(s_{jl}, p_{jl}) : 1 \leq l \leq 7\}$, es el conjunto de materias solicitadas por el estudiante j y priorizadas por él (s_{jl} es el código de la l-ésima materia solicitada por el estudiante j y $p_{jl} \in P$ es la prioridad de s_{jl}).

     Además, deben cumplirse las siguientes restricciones:
     - $\sum_{l=1}^{|ms_j|} p_{jl} \leq \gamma(|ms_j|)$ para $j = 1..r$, donde $\gamma(X) = 3X - 1$ (es decir, cada estudiante e_j repartió su capacidad de priorizar $\gamma(|ms_j|)$ entre las asignaturas solicitadas por él).
     - Para todo $j = 1..r$, $s_{jl} \neq s_{jq}$ si $l \neq q$ (es decir, en la solicitud de cada estudiante e_j no hay materias repetidas).
   - $A = \{(e_j, ma_j) : j = 1 \dots r\}$ una posible solución del problema, es decir, un conjunto de materias asignadas por estudiante, donde $ma_j \subseteq ms_j$. Además A no debe sobrepasar el cupo en ninguna materia.
   - La función de medición de insatisfacción del estudiante j, es:

$$f_j = (1 - \frac{|ma_j|}{|ms_j|}).(\frac{\sum_l \{p_{jl} : ((s_{jl}, p_{jl}) \in ms_j) \wedge ((s_{jl}, p_{jl}) \notin ma_j)\}}{\gamma(|ms_j|)})$$

     .
   - La función de insatisfacción general de los estudiantes, dados una entrada $\langle M, E \rangle$ y una solución A, es:

$$\mathcal{F}_{\langle M,E \rangle}(A) = \frac{\sum_{j=0}^{r} f_j}{r}$$

     .
2. **El problema**
   - **Entrada:** La tupla $\langle M, E \rangle$
   - **Salida:** Un conjunto A tal que $\mathcal{F}_{\langle M,E \rangle}(A)$ sea mínima.

2.3. ¿Entendimos el problema?

Supongamos que:

- $M = \{(M_1, 3), (M_2, 4), (M_3, 2)\}$
- $E = \{(e_1, ms_1), (e_2, ms_2), (e_3, ms_3), (e_4, ms_4), (e_5, ms_5)\}$, donde:
  - $ms_1 = \{(M_1, 5), (M_2, 2), (M_3, 1)\}$
  - $ms_2 = \{(M_1, 4), (M_2, 1), (M_3, 3)\}$
  - $ms_3 = \{(M_2, 3), (M_3, 2)\}$
  - $ms_4 = \{(M_1, 2), (M_3, 3)\}$
  - $ms_5 = \{(M_1, 3), (M_2, 2), (M_3, 3)\}$
- Describa cinco (5) posibles soluciones válidas para esa entrada y calcule la función de insatisfacción general para cada una de ellas. Guarde los resultados en una tabla.
- Describa una solución válida A tal que $\mathcal{F}_{\langle M,E\rangle}(A)$ sea mínima.

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 genera todas las soluciones posibles y entre ellas escoge la mejor.

3.1.1. Complejidad

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

3.1.2. 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 presentada en 2.3 y calcule la insatisfacción de 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 k yr (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 k yr (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 $k = 2 \times 10^3$ y $n = 32 \times 10^3$ (datos del caso real):

- Argumente si el tiempo que se tomará el equipo en resolver el problema es aceptable o inaceptable en la práctica.

- Argumente si el espacio que se tomará el equipo en resolver el problema es aceptable o 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

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). Cada función recibe k, r y M, E y devuelve una pareja $(A, \mathcal{F}_{\langle M,E\rangle}(A))$ tal que A es la respuesta al problema. Esas funciones en su código deben llamarse `rocFB, rocV` y `rocPD`.
- 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:

```
k
M1,m1
M2,m2
.
.
.
Mk,mk
r
e1,s1
m11,p11
m12,p12
.
.
.
m1s1,p1s1
e2,s2
m21,p21
m22,p22
```

```
.
.
.
m2s2,p2s2
.
.
.
er,sr
mr1,pr1
mr2,pr2
.
.
.
mrsr,psr
```

Es decir, la primera línea trae el número de materias (un entero k); las siguientes k líneas traen los códigos de las materias y su cupo separados por coma, una línea por materia; la siguiente línea trae el número de estudiantes (r). Luego vienen r bloques de líneas, una por cada estudiante, así: la primera línea del bloque $j \in 1..r$ trae el código del estudiante (e_j) y el número de materias solicitadas por ese estudiante (s_j) separados por una coma; luego vienen s_j líneas, cada una con un código de asignatura solicitado, una coma, y enseguida la prioridad ($m_{j,l}, p_{j,l} : l \in 1..s_j$)

3.4.2. Formato de las salidas

Las salidas se deben producir en un archivo de texto con $r + 1$ líneas así:

```
Costo
e1,a1
m11
m12
.
.
.
m1a1
e2,a2
m21
m22
.
.
.
m2a2
.
.
.
er,ar
mr1
mr2
.
.
.
mrar
```

Es decir, la primera línea trae el costo de la solución; luego vienen r bloques de líneas, una por cada estudiante, así: la primera línea del bloque $j \in 1..r$ trae el código del estudiante (e_j) y el número de materias asignadas a ese estudiante (a_j) separados por una coma; luego vienen a_j líneas, cada una con un código de asignatura asignado $(m_{j,l} : l \in 1..a_j)$.

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 14 de octubre de 2025 a las 23:59. La sustentación será definido posteriormente a la fecha de entrega

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 ArandaMuñoz.zip o ArandaMuñoz.rar, o ArandaMuñoz.tgz o ...

4. Sustentación y calificación

El trabajo debe ser sustentado por los autores en el día y hora especificado para su grupo, dicha fecha será comunicada por medio del campus virtual. **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), además de una sección dedicada a comparar los resultados de las tres 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!!!