Departamento de Informática
Universidad Carlos III de Madrid

Grado en Informática

Heurística y Optimización

22 de Enero de 2019

Normas generales del examen

① El tiempo para realizar el examen es de **4 horas**

② No se responderá a ninguna pregunta sobre el examen transcurridos los primeros **30 minutos**

③ Cada pregunta debe responderse en páginas separadas en el mismo orden de sus apartados. Si no se responde, se debe entregar una página en blanco

④ Numera todas las páginas de cada ejercicio separadamente

⑤ Escribe con claridad y en limpio, de forma ordenada y concisa

⑥ Si se sale del aula, no se podrá volver a entrar durante el examen

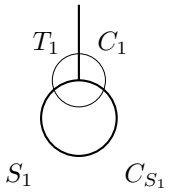
⑦ No se puede presentar el examen escrito a lápiz

Pregunta 1 (2 puntos)

Un sistema de distribución de aguas consiste en la combinación de diferentes elementos. Se pide modelizar con *Programación Lineal* su combinación, intentando minimizar la pérdida de agua y cumpliendo con las restricciones de capacidad máxima de cada tubería y depósito de aguas que forman parte del sistema.

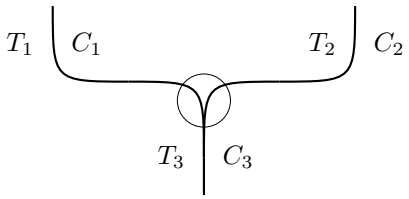
Se pide responder razonadamente las siguientes preguntas, explicando claramente en cada apartado las *variables de decisión* elegidas, y las *restricciones* necesarias que sirven para modelar cada componente:

(a) **(½ puntos)** La primera componente consiste en un depósito de aguas, mostrado con un círculo grande en el siguiente diagrama, al que llega agua desde una tubería denominada T_1, cuya capacidad máxima es C_1. El depósito de aguas, etiquetado como S_1, tiene una capacidad máxima de C_{S_1}. El pequeño círculo mostrado en la unión entre la tubería T_1 y el depósito de aguas, S_1 indica que ahí podría haber pérdidas si llega más volumen que el que puede acogerse en el depósito.



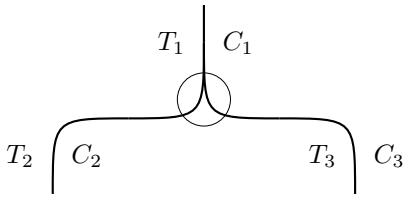
Dada una cantidad arbitraria de agua que llega por T_1, determinar las variables de decisión que explican el agua que queda en el depósito y la cantidad exacta que podría perderse, si se excede su capacidad, así como las restricciones que modelizan el comportamiento del depósito de aguas.

(b) **(½ puntos)** La segunda componente es una *bifurcación invertida*, a la que confluyen dos tuberías T_1 y T_2 con capacidades C_1 y C_2 respectivamente, para alimentar el caudal de la tubería T_3 cuya capacidad máxima es C_3. El pequeño círculo mostrado en la unión entre las tuberías T_1 y T_2 indica que ahí podría haber pérdidas si llega más volumen que el que puede acogerse en la tubería T_3.



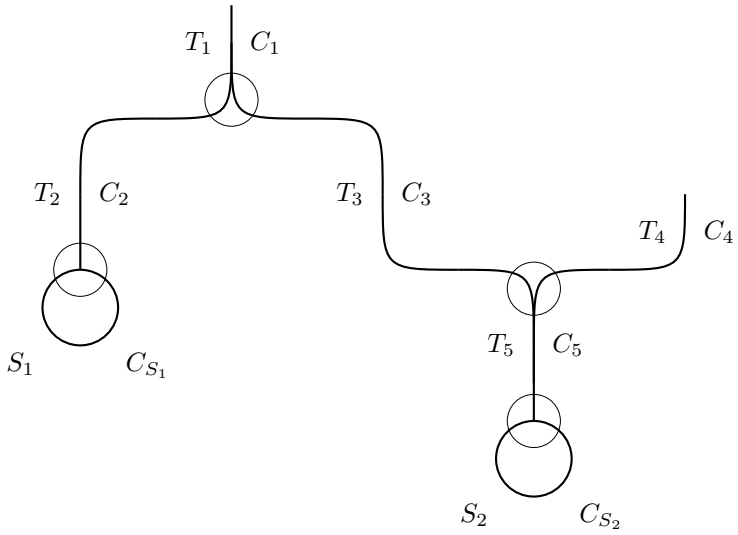
Dada una cantidad arbitraria de agua que llega por las tuberías T_1 y T_2, determinar las variables de decisión que explican el agua que habrá en T_3, y la cantidad exacta que podría perderse, si se excede su capacidad, así como las restricciones que modelizan el comportamiento de la bifurcación invertida.

(c) **(½ puntos)** La última componente que se considera consiste en una *bifurcación*, que distribuye el volumen que llega por una tubería T_1 en partes iguales a las tuberías T_2 y T_3, cuyas capacidades máximas son C_2 y C_3 respectivamente. El pequeño círculo mostrado en la unión entre las tuberías T_2 y T_3 indica que ahí podría haber pérdidas si llega más volumen que el que puede acogerse por éllas.



Dada una cantidad arbitraria de agua que llega por T_1, determinar las variables de decisión que explican el agua que habrá en las tuberías T_2 y T_3, y la cantidad exacta que podría perderse, si se excede la suma de sus capacidades, así como las restricciones que modelizan el comportamiento de la bifurcación.

(d) **(½ puntos)** Por último, se pide diseñar una tarea de Programación Lineal que determine el agua que debe llegar al sistema que se muestra en la siguiente figura a través de las tuberías T_1 y T_4, de modo que se maximiza el volumen que llega al depósito S_2, garantizando que no hay pérdidas en ninguna de las componentes de todo el sistema.



Pregunta 2 (2 puntos)

Considera la fórmula F_1 en Forma Normal Conjuntiva $F_1 = \bigwedge_{i=1}^{4} C_i$ formada por las siguientes cláusulas:

$$C_1{:}(x_1 \vee \bar{x}_2) \quad C_2{:}(x_2 \vee \bar{x}_3)$$
$$C_3{:}(x_1 \vee \bar{x}_3) \quad C_4{:}(x_2 \vee \bar{x}_2)$$

Se pide responder razonadamente las siguientes preguntas:

(a) **(1 punto)** Indica qué cláusulas pueden eliminarse de la fórmula F_1 porque son innecesarias para resolver la satisfabilidad de F_1, y el por qué.

Considérese ahora, en su lugar, la fórmula F_2 en Forma Normal Conjuntiva $F_2 = \bigwedge_{i=1}^{4} C_i$ formada por las siguientes cláusulas:

$$C_1{:}(x_1 \vee x_2 \vee \bar{x}_4) \quad C_2{:}(\bar{x}_1 \vee \bar{x}_2 \vee x_3)$$
$$C_3{:}(\bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_4) \quad C_4{:}(x_2 \vee x_4)$$

(b) **(½ puntos)** Aplicar Davis-Putnam (DP) para encontrar un modelo que satisfaga la fórmula F_2, en caso de que exista alguno.

Se exige aplicar DP usando las variables en orden ascendente de su subíndice

(c) **(½ puntos)** Aplicar Davis–Putnam–Logemann–Loveland (DPLL) para encontrar, al menos, un modelo que satisfaga la fórmula F_2, en caso de que exista alguno.

Se exige aplicar DPLL usando las variables en orden ascendente de su subíndice

Pregunta 3 (3 puntos)

Considerése el siguiente problema de Programación Lineal:

$$\text{máx } z = 2x_1 + x_3$$
$$\begin{array}{rcrcrcr} x_1 & - & 3x_2 & + & 2x_3 & \geq & -10 \\ x_1 & & & - & 2x_3 & \leq & 2 \\ 2x_1 & + & 6x_2 & + & 3x_3 & \geq & 7 \end{array}$$
$$\mathbf{x} \geq \mathbf{0}$$

Se pide responder razonadamente las siguientes preguntas:

(a) **(½ puntos)** Si se suman la primera y tercera restricción de la tarea de Programación Lineal anterior resulta $3x_1 + 3x_2 + 5x_3 \geq -3$. Por lo tanto, ¿son equivalentes la tarea anterior y la que se muestra a continuación? Sí o no, y por qué.

$$\text{máx } z = 2x_1 + x_3$$
$$\begin{array}{rcrcrcr} x_1 & & & - & 2x_3 & \leq & 2 \\ 3x_1 & + & 3x_2 & + & 5x_3 & \geq & -3 \end{array}$$
$$\mathbf{x} \geq \mathbf{0}$$

(b) **(½ puntos)** Expresar la primera tarea de Programación Lineal en forma *estándar* de maximización.

(c) **(½ puntos)** Preparar el problema de Programación Lineal que ha resultado del apartado anterior para su aplicación con el algoritmo SIMPLEX para garantizar que pueda comenzarse con la matriz identidad.

(d) **(1 punto)** Resolver el problema de Programación Lineal obtenido en el apartado anterior con el algoritmo SIMPLEX.

Es imprenscindible indicar claramente, en cada iteración: las variables escogidas en la base, su valor, y el valor de la función objetivo

(e) **(½ puntos)** Interpretar las soluciones halladas y explicar qué conclusiones pueden extraerse.

Pregunta 4 (3 puntos)

Un operador logístico dispone de una flota $\mathcal{C}$ de camiones, cada uno de los cuales puede transportar mercancías de uno o varios tipos, pero no de todos. Además, cada camión $c_i \in \mathcal{C}$ puede transportar mercancías hasta un peso máximo W_i. Dado un cojunto de mercancías, cada una de las cuales está caracterizada por su peso y tipo, se pide determinar qué mercancías se deben cargar en qué camiones para minimizar el número de las que no pueden cargarse en ningún camión.

Se pide responder razonadamente las siguientes preguntas:

(a) **(½ puntos)** Representar el problema como un *espacio de estados*

(b) **(½ puntos)** ¿Cuál es la profundidad máxima de un árbol de búsqueda desarrollado para resolver óptimamente este problema?

(c) **(½ puntos)** Si se desea calcular la solución óptima, ¿qué algoritmo de búsqueda de *fuerza bruta* sugerirías para su resolución?

(d) **(1 punto)** Sugiere una función heurística $h(\cdot)$ que sea *admisible* e *informada* para el problema de minimizar el número de mercancías que no pueden transportarse en ningún camión.

(e) **(½ puntos)** ¿Qué algoritmo de búsqueda *heurística* es el más indicado para resolver óptimamente el problema? ¿Por qué?

Soluciones del examen de Heurística y Optimización Enero 2018

Problema 1

1. En primer lugar, se representan con variables de decisión las diferentes cantidades de agua que intervienen en la componente del tanque de agua:

   $x_1 \rightleftharpoons$ Volumen de agua en la tubería T_1
   $x_2 \rightleftharpoons$ Volumen de agua en el tanque S_1
   $x_3 \rightleftharpoons$ Volumen de agua que se pierde en la unión de la tubería y el tanque

   Con las que ahora es posible desarrollar las siguientes restricciones:

   - En primer lugar, todas las variables de decisión deben ser no negativas puesto que:
     - Ni la tubería T_1 ni el tanque de agua S_1 pueden acoger un volumen negativo de agua.
     - Tampoco es posible perder una cantidad negativa de agua en la unión entre la tubería y el tanque de agua

     Por lo tanto:

     $$x_1, x_2, x_3 \geq 0$$

   - El volumen de agua que llega al depósito no puede ser nunca mayor que el volumen en la tubería:

     $$x_2 \leq x_1$$

   - Y, además, el tanque no puede exceder su capacidad:

     $$x_2 \leq C_{S_1}$$

   - Por último, se pedía determinar la cantidad exacta de agua que puede perderse en la unión entre la tubería T_1 y el tanque de agua S_1. Si y sólo si el volumen de agua en la tubería es mayor o igual que la capacidad del tanque, entonces la diferencia $(x_1 - C_{S_1})$ representa el volumen de agua que se desperdicia. Sin embargo, si $x_1 < C_{S_1}$ la diferencia anterior sería negativa (y representaría la cantidad adicional que podría acogerse en el tanque) y, obviamente, no es posible perder una cantidad negativa de agua.

     Por lo tanto, sea x_4 otra variable de decisión no negativa ($x_4 \geq 0$) que representa la cantidad de agua adicional que podría ser absorbida por el tanque, entonces la restricción:

     $$x_3 = x_1 - C_{S_1} + x_4$$

     representa en x_3 exactamente el volumen de agua desperdiciado en la unión entre la tubería y el tanque de agua: una cantidad positiva si hay efectivamente desperdicio, y cero si no la ha habido, si y sólo si x_4 toma el valor no negativo más pequeño posible y, por ello, se añade también:

     $$\text{mín}\, z = x_4$$

2. Como antes, se determinan primero las variables de decisión que representan los volúmenes de agua que hay en las diferentes partes de la bifurcación invertida como sigue:

   $x_1 \rightleftharpoons$ Volumen de agua en la tubería T_1
   $x_2 \rightleftharpoons$ Volumen de agua en la tubería T_2
   $x_3 \rightleftharpoons$ Volumen de agua en la tubería T_3
   $x_4 \rightleftharpoons$ Volumen de agua que se pierde en la unión de las tuberías T_1 y T_2

   Que se emplean convenientemente para la modelización de las siguientes restricciones:

- En primer lugar, todas las variables de decisión deben ser no negativas, puesto que no es posible ni acoger ni perder un volumen negativo de agua:

$$x_1, x_2, x_3, x_4 \geq 0$$

- Obviamente, la cantidad de agua que llega a la tubería T_3 no puede exceder la suma de los volúmenes de agua que llegan a las dos primeras tuberías:

$$x_3 \leq x_1 + x_2$$

- Y, por supuesto, tampoco puede exceder su capacidad:

$$x_3 \leq C_3$$

- Por último, razonando como antes se observa que $(x_1 + x_2 - C_3)$ es el volumen de agua que se desperdicia en la unión de las dos primeras tuberías, si y sólo sí esa diferencia es positiva. Para evitar que el resultado sea negativo se añade una variable adicional, $x_5 \geq 0$, que representa el volumen adicional de agua que puede ser absorbido por la tubería T_3:

$$x_4 = x_1 + x_2 - C_3 + x_5$$

de modo que ahora x_4 representa exactamente la cantidad de agua desperdiciada: 0 si no se pierde nada, y una cantidad positiva en caso de que sea así, si y sólo si, la variable x_5 toma el valor no negativo más pequeño posible y, por ello, se añade:

$$\text{mín}\, z = x_5$$

3. La bifurcación se modeliza con las siguientes variables de decisión (que, de hecho, son las mismas que las empleadas en la bifurcación invertida):

$x_1 \rightleftharpoons$ Volumen de agua en la tubería T_1
$x_2 \rightleftharpoons$ Volumen de agua en la tubería T_2
$x_3 \rightleftharpoons$ Volumen de agua en la tubería T_3
$x_4 \rightleftharpoons$ Volumen de agua que se pierde en la unión de las tuberías T_1 y T_2

De modo que las restricciones son las siguientes:

- Como en el caso de las componentes consideradas anteriormente, ninguna variable de decisión puede tomar valores negativos:

$$x_1, x_2, x_3, x_4 \geq 0$$

- Además, la cantidad de agua que llega hasta cualquiera de las tuberías T_2 o T_3 no puede ser mayor que la mitad del agua que corre por la primera tubería, puesto que ella distribuye su caudal en partes iguales:

$$\begin{aligned} x_2 &\leq \tfrac{1}{2}x_1 \\ x_3 &\leq \tfrac{1}{2}x_1 \end{aligned}$$

- Y tampoco es posible que ninguna de ellas exceda su capacidad:

$$\begin{aligned} x_2 &\leq C_2 \\ x_3 &\leq C_3 \end{aligned}$$

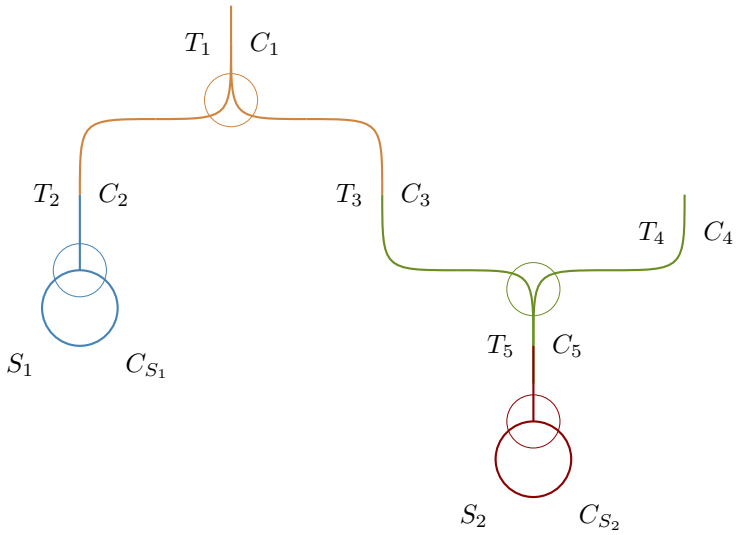
- Por último, se observa nuevamente que la diferencia $(\frac{1}{2}x_1 - C_2 + \frac{1}{2}x_1 - C_3) = (x_1 - C_2 - C_3)$ sería el volumen de agua desperdiciado en el punto en que se unen las tuberías T_2 y T_3, sólo si toma un valor positivo. Para evitar que la variable que representa la cantidad desperdiciada tome valores negativos, se añade otra variable, $x_5 \geq 0$, de modo que:

$$x_4 = x_1 - C_2 - C_3 + x_5$$

representa ahora en x_4 la cantidad de agua desperdiciada, o 0 si no se pierde nada, si y sólo si la variable x_5 toma el valor no negativo más pequeño posible:

$$\text{mín } z = x_5$$

4. El último apartado muestra un sistema de distribución de aguas que está formado por dos tanques de agua (S_1 y S_2), una bifurcación (a la que llega T_1), y una bifurcación invertida —en la que confluyen T_3 y T_4. Cada una de estas componentes se muestra a continuación con un color diferente:



Las variables de decisión necesarias para representar los diferentes volúmenes de agua que hay en el sistema, son las que siguen, en las que algunas se han renombrado para mejorar la legibilidad de la tarea de Programación Lineal resultante:

- Cada tubería T_i tiene un caudal que se representa con la variable x_i, $1 \leq i \leq 5$. Por lo tanto, $C_i, 1 \leq i \leq 5$ representa la capacidad máxima de agua que puede acogerse en el elemento i-ésimo.
- El volumen de agua en los tanques S_1 y S_2 se representa con las variables x_6 y x_7 respectivamente. Por ello, C_6 y C_7 representan la capacidad máxima de los tanques S_1 y S_2 respectivamente.
- El volumen de agua que se desperdicia en cada componente se representa con las variables y:
  - $y_1 \rightleftharpoons$ Volumen desperdiciado en la bifurcación de la tubería T_1
  - $y_2 \rightleftharpoons$ Volumen desperdiciado en la bifurcación invertida de las tuberías T_3 y T_4
  - $y_3 \rightleftharpoons$ Volumen desperdiciado en el tanque S_1
  - $y_4 \rightleftharpoons$ Volumen desperdiciado en el tanque S_2
- Por último, las variables z representan la cantidad de agua que debe sumarse (y que a continuación se dice que "sobra" en el sentido de que puede absorberse adicionalmente) para calcular correctamente la cantidad que se desperdicia en cada componente:
  - $z_1 \rightleftharpoons$ Sobrante de agua en la bifurcación de la tubería T_1
  - $z_2 \rightleftharpoons$ Sobrante de agua en la bifurcación invertida de las tuberías T_3 y T_4
  - $z_3 \rightleftharpoons$ Sobrante de agua en el tanque S_1
  - $z_4 \rightleftharpoons$ Sobrante de agua en el tanque S_2

Gracias a las cuales es posible ahora presentar todas las restricciones conjuntamente de las cuatro componentes del sistema de distribución de aguas:

$$
\begin{array}{rrrrrrrrrrrrrrrrrrrrr}
-\frac{1}{2}x_1 & +x_2 & & & & & & & & & & & & & & & & & \leq & 0 \\
-\frac{1}{2}x_1 & & +x_3 & & & & & & & & & & & & & & & & \leq & 0 \\
& x_2 & & & & & & & & & & & & & & & & & \leq & C_2 \\
& & x_3 & & & & & & & & & & & & & & & & \leq & C_3 \\
-x_1 & & & & & & & & +y_1 & & & & & & -z_1 & & & & = & -C_2 - C_3 \\
& & -x_3 & -x_4 & +x_5 & & & & & & & & & & & & & & \leq & 0 \\
& & & & x_5 & & & & & & & & & & & & & & \leq & C_5 \\
& & -x_3 & -x_4 & & & & & & +y_2 & & & & -z_2 & & & & & = & -C_5 \\
& -x_2 & & & & +x_6 & & & & & & & & & & & & & \leq & 0 \\
& & & & & x_6 & & & & & & & & & & & & & \leq & C_6 \\
& -x_2 & & & & & & & & & +y_3 & & & & -z_3 & & & & = & -C_6 \\
& & & & -x_5 & & +x_7 & & & & & & & & & & & & \leq & 0 \\
& & & & & & x_7 & & & & & & & & & & & & \leq & C_7 \\
& & & & -x_5 & & & & & & & +y_4 & & & & & -z_4 & = & -C_7
\end{array}
$$

donde cada grupo de restricciones que se refiere a cada componente del sistema mostrado en la figura anterior utiliza el mismo color para facilitar su identificación.

Además, el enunciado pedía expresamente que no hubiera pérdidas de agua en ninguna parte y, por lo tanto es preciso añadir las restricciones:

$$
\begin{array}{rrrrrrrrrrrrrrrrrrrrr}
y_1 & & & & & & & & & & & & & & & & & & = & 0 \\
& & y_2 & & & & & & & & & & & & & & & & = & 0 \\
& & & & y_3 & & & & & & & & & & & & & & = & 0 \\
& & & & & & y_4 & & & & & & & & & & & & = & 0
\end{array}
$$

A estas restricciones es preciso sumar las siguientes:

$$
\begin{array}{rcl}
\mathbf{x} & \geq & \mathbf{0} \\
\mathbf{y} & \geq & \mathbf{0} \\
\mathbf{z} & \geq & \mathbf{0}
\end{array}
$$

Por último, la función objetivo maximiza el caudal de agua que llega al depósito S_2:

$$\text{máx}\, z = x_7$$

Problema 2

1. El método de resolución elimina todas las cláusulas que contengan literales puros, y un literal ℓ es puro en una fórmula F de lógica proposicional en Forma Normal Conjuntiva si y sólo si su negación, $\bar{\ell}$, no aparece nunca en la fórmula F.

Puede comprobarse que los literales x_1 y $\bar{x}_3$ son puros en la fórmula F_1 y, por lo tanto, las cláusulas que los contienen (C_1, C_2 y C_3) pueden eliminarse. El resultado, que es la fórmula $F_1' = (x_2 \vee \bar{x}_2)$ es lógicamente equivalente a la dada en el enunciado.

Ahora bien, F_1' contiene ahora una única cláusula que es una tautología y, por lo tanto, puede eliminarse también puesto que las tautologías son siempre ciertas.

Por lo tanto, es posible eliminar todas las cláusulas, resultando entonces el conjunto vacío ($\varnothing$) que denota que la fórmula inicial F_1 es satisfacible.

2. Para aplicar Davis-Putnam, se escoge en cada iteración un literal y se aplica la resolución a la red de cláusulas actual respecto de ese literal. En cada paso, se guardan las variables usadas y las cláusulas involucradas de modo que si eventualmente resultara el conjunto vacío, $\varnothing$, se usa esa información para generar un modelo que valide la expresión inicial. Si, en otro caso, se obtiene la cláusula vacía, $\{\varnothing\}$, entonces se habrá probado que la fórmula original es insatisfacible, concluyendo así la aplicación del algoritmo —puesto que ahora se sabe que no hay ningún modelo que calcular.

   A continuación se muestran todos los pasos del algoritmo DP. En cada paso se muestran la red de cláusulas G_i, la variable empleada x_j (que será, como se solicita en el enunciado, la siguiente en orden ascendiente de su subíndice), y las cláusulas involucradas, que se calculan como la diferencia entre la red de cláusulas de cada paso y el resultado de aplicar resolución: $G_i \backslash \text{Res}(G_i, x_j)$.

   Paso 0 $G_0 = \bigwedge_{i=1}^{4} C_i$

   La primera variable a utilizar es x_1. Aplicando el método de resolución respecto de élla, se observa que el literal x_1 aparece afirmado en C_1, y negado en C_2 y C_3. Por lo tanto, deben considerarse dos pares de cláusulas: $\{C_1, C_2\}$ y $\{C_1, C_3\}$.

   Después de calcular la disyunción de los literales que acompañan a un literal y el otro en el primer par, resulta la cláusula $(x_2 \vee \bar{x}_4 \vee \bar{x}_2 \vee x_3)$, que es una tautología y, por lo tanto, no se produce.

   La misma operación realizada ahora con las cláusulas C_1 y C_3 produce una nueva cláusula: $C_5 : (x_2 \vee \bar{x}_4 \vee \bar{x}_3 \vee \bar{x}_4)$ que, por la regla de idempotencia es lógicamente equivalente a $C_5 : (x_2 \vee \bar{x}_3 \vee \bar{x}_4)$.

   $$\text{Res}(G_0, x_1) = \{C_4, C_5 : (x_2 \vee \bar{x}_3 \vee \bar{x}_4)$$

   Por lo tanto, las cláusulas involucradas en este paso se calculan con la expresión:

   $$G_0 \backslash \text{Res}(G_0, x_1) = G_0 \backslash \{C_4, C_5\} = \{C_1, C_2, C_3\}$$

   Paso 1 $G_1 = \bigwedge_{i=4}^{5} C_i$

   En el segundo paso de aplicación del algoritmo Davis-Putnam debe utilizarse x_2, que es puro en G_1, puesto que $\bar{x}_2 \notin G_1$. Por lo tanto, la aplicación del método de resolución elimina todas las cláusulas que contengan el literal x_2, que son las cláusulas C_4 y C_5.

   Por lo tanto:

   $$\text{Res}(G_1, x_2) = \varnothing$$

   con lo que se prueba que la fórmula original es satisfacible, pero antes de pasar al cálculo del modelo, se calculan también en este último paso las cláusulas involucradas:

   $$G_1 \backslash \text{Res}(G_1, x_2) = G_1 \backslash \varnothing = \{C_4, C_5\}$$

Puesto que la fórmula es lógicamente satisfacible, debe procederse ahora a la segunda fase del algoritmo de Davis-Putnam, que considera todas las variables empleadas y las cláusulas involucradas en cada paso en orden inverso al de su aplicación. La tabla 1 muestra esta información.

Paso	Variable	Cláusulas
1	x_2	$C_4 : (x_2 \vee x_4), C_5 : (x_2 \vee \bar{x}_3 \vee \bar{x}_4)$
0	x_1	$C_1 : (x_1 \vee x_2 \vee \bar{x}_4), C_2 : (\bar{x}_1 \vee \bar{x}_2 \vee x_3), C_3 : (\bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_4)$

Cuadro 1: Fase II del algoritmo de Davis-Putnam

En la segunda fase del algoritmo de Davis-Putnam, un modelo que valide una expresión satisfacible de la lógica proposicional puede encontrarse componiendo los modelos que satisfacen las cláusulas involucradas en cada paso dando valores a las variables empleadas y, sólo si fuera preciso, a otras. Por supuesto, en cada paso, deben tenerse en cuenta las asignaciones realizadas en los pasos anteriores.

Puesto que el literal x_2 es puro en la fórmula de lógica proposicional formarda por la conjunción de las cláusulas C_4 y C_5, basta con hacer $x_2 = \top$ para satisfacer ambas cláusulas, tal y como debe hacerse en el primer paso de la segunda fase del algoritmo Davis-Putnam.

En el siguiente paso se considera ahora la asignación de valor a la variable x_1 para satisfacer las cláusulas C_1, C_2 y C_3. Ahora bien, puesto que $x_2 = \top$, la cláusula C_1 ya está satisfecha. Puesto que la variable x_1 sólo aparece negada en las cláusulas restantes, C_2 y C_3, ambas pueden satisfacerse con $x_1 = \bot$.

Por lo tanto, el modelo resultante es:

$$M_1 = \{x_1 = \bot, x_2 = \top\}$$

para el que las variables x_3 y x_4 pueden tomar cualquier valor.

3. El siguiente árbol muestra la aplicación del algoritmo Davis-Putnam-Logemann-Loveland (DPLL) a la fórmula F_2 en orden creciente de los subíndices de las variables, tal y como se solicitaba en el enunciado:

   Como puede verse, hay en total hasta tres modelos diferentes, uno de los cuales es M_1, el modelo obtenido en el apartado anterior. Los otros dos son:

$$M_2{=}\{x_1 = \top, x_2 = \top, x_3 = \top, x_4 = \bot\}$$
$$M_3{=}\{x_1 = \top, x_2 = \bot, x_3 = \bot, x_4 = \top\}$$

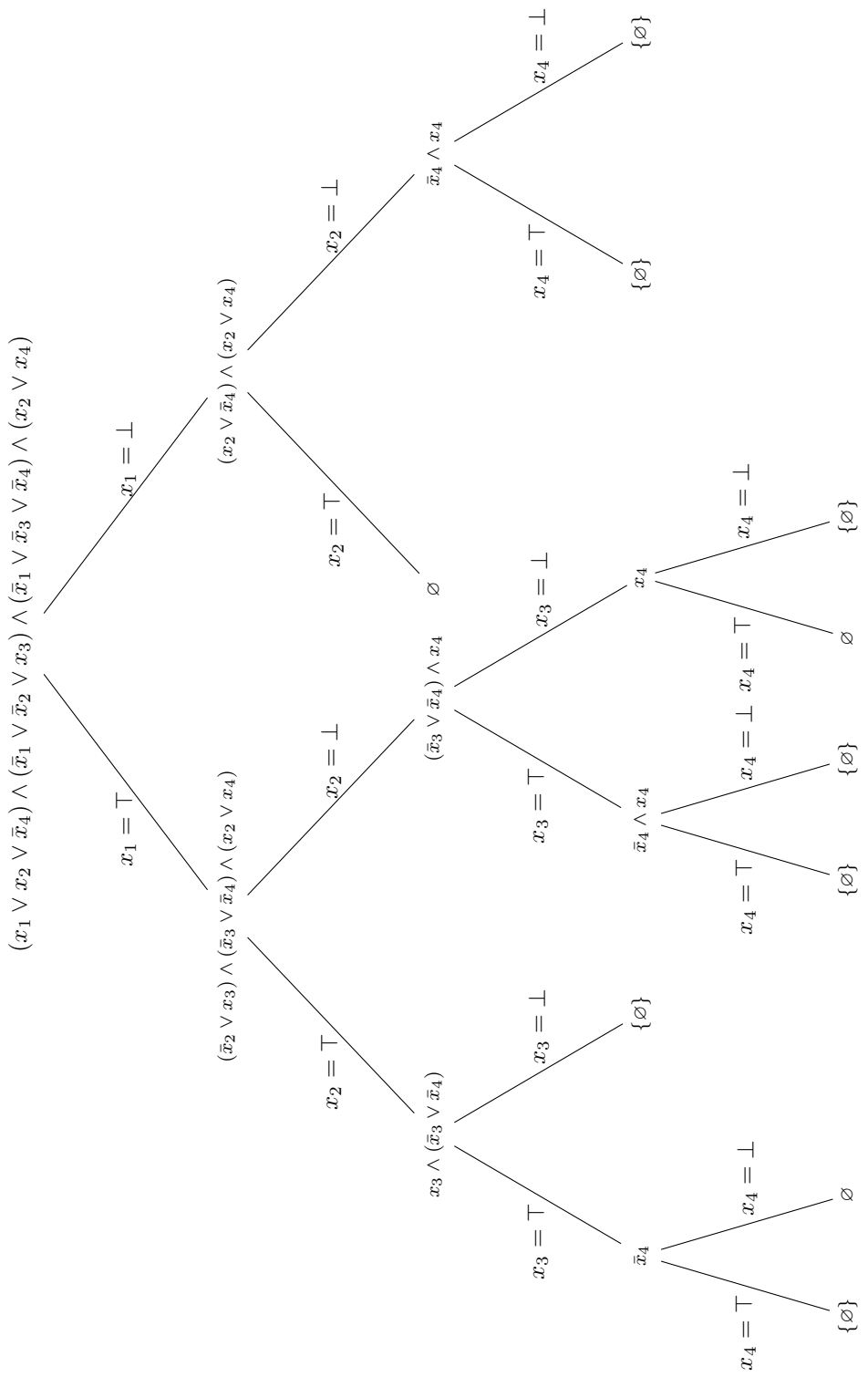
Problema 3

1. ¡Es una barbaridad creer que la suma de dos restricciones lineales es equivalente a éllas!

   Un buen argumento para demostrarlo es que dos restricciones lineales (cuya matriz de coeficientes tenga rango por filas igual a 2, es decir, donde una no es una combinación lineal de la otra) crean una región factible en el espacio n-dimensional (con n el número de variables de decisión involucradas) que no puede replicarse con un único hiperplano, el de la restricción que resultara de sumarlas.

   Como resultado de la observación anterior, es posible exponer otro argumento que puede verificarse manualmente, y es que habrá puntos $x \in X^n$ que, mientras que verifican la restricción que resulta de sumar dos restricciones, no pertenece a la región factible creada cuando las dos restricciones se consideran simultáneamente, o lo contrario.

   Por ejemplo, si se considera el punto n-dimensional $\mathbf{0} \in X^n$, puede observarse que verifica la suma de las restricciones, puesto que $3x_1 + 3x_2 + 5x_3$ en $x = \mathbf{0}$ vale precisamente 0 y, por ello $3x_1 + 3x_2 + 5x_3 \geq -3$. Sin embargo, el mismo punto no verifica la tercera restricción, $2x_1 + 6x_2 + 3x_3 \geq 7$, puesto que $0 \ngeq 7$ y, por lo tanto, $\mathbf{0} \in X^n$ no pertenece a la región factible creada por las dos restricciones, mientras que sí pertenece a la región factible que resulta de sumarlas.


$(x_1 \vee x_2 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (x_2 \vee x_4)$
$x_1 = \top$
$x_1 = \bot$
$(\bar{x}_2 \vee x_3) \wedge (\bar{x}_3 \vee \bar{x}_4) \wedge (x_2 \vee x_4)$
$(x_2 \vee \bar{x}_4) \wedge (x_2 \vee x_4)$
$x_2 = \top$
$x_2 = \bot$
$x_3 \wedge (\bar{x}_3 \vee \bar{x}_4)$
$(\bar{x}_3 \vee \bar{x}_4) \wedge x_4$
$\varnothing$
$\bar{x}_4 \wedge x_4$
$x_3 = \top$
$x_3 = \bot$
$x_3 = \top$
$x_3 = \bot$
$\bar{x}_4$
x_4
$\bar{x}_4 \wedge x_4$
$\{\varnothing\}$
$x_4 = \top$
$x_4 = \bot$
$x_4 = \top$
$x_4 = \bot$
$x_4 = \top$
$x_4 = \bot$
$\{\varnothing\}$
$\varnothing$
$\varnothing$
$\{\varnothing\}$
$\{\varnothing\}$
$\{\varnothing\}$
$\{\varnothing\}$

2. Un problema de programación lineal está en forma *estándar* si todas las restricciones son de igualdad, las variables de decisión son no negativas y, por último, el vector de constantes o recursos $\mathbf{b}$ no contiene términos negativos. Estará, además, en forma de maximización si la función objetivo maximiza y de minimización en otro caso. El problema, tal y como estaba enunciado, sólo verifica la segunda condición. Conviene aquí recordar:

   - Una restricción de la forma $\leq$ está acotada superiormente. Puesto que ninguna variable de decisión puede tomar valores negativos, es preciso *sumar* una *variable de holgura* para forzar la igualdad.
   - Análogamente, las restricciones de la forma $\geq$ están acotadas inferiormente de modo que, con variables de decisión que no pueden tomar valores negativos, es preciso *restar* una *variable de holgura* para forzar la igualdad.

   Además, las variables de holgura que se añadan a las restricciones para forzar igualdades, se añadirán a la función objetivo z con un coeficiente nulo.

   En primer lugar, se cambia el signo del recurso de la primera restricción para hacer que sea positivo. Para ello, basta con multiplicar los dos términos de la primera restricción por -1 (con lo que, además, se cambia el sentido de la desigualdad):

$$\begin{gathered}\text{máx}\, z = 2x_1 + x_3 \\ \begin{array}{rcrcrcr} -x_1 & + & 3x_2 & - & 2x_3 & \leq & 10 \\ x_1 & & & - & 2x_3 & \leq & 2 \\ 2\,x_1 & + & 6x_2 & + & 3x_3 & \geq & 7 \end{array} \\ \mathbf{x} \geq \mathbf{0}\end{gathered}$$

   A continuación se añaden variables de holgura para convertir todas las desigualdades en igualdades tal y como se explicaba anteriormente. Por lo tanto, el problema de Programación Lineal queda, como sigue, en forma estándar de maximización:

$$\begin{gathered}\text{máx}\, z = 2x_1 + x_3 \\ \begin{array}{rcrcrcrcrcr} -x_1 & + & 3x_2 & - & 2x_3 & + & x_4 & & & = & 10 \\ x_1 & & & - & 2x_3 & & & + & x_5 & = & 2 \\ 2\,x_1 & + & 6x_2 & + & 3x_3 & & & - & x_6 & = & 7 \end{array} \\ \mathbf{x} \geq \mathbf{0}\end{gathered}$$

3. El problema de Programación Lineal, tal y como se muestra en el apartado anterior, contiene dos columnas de la matriz identidad (de dimensión 3): $\mathbf{a_4}$ y $\mathbf{a_5}$. Para poder añadir la tercera columna que falta (y que consistirá primero de dos ceros y luego un 1), se añade una *variable artificial* x_7:

$$\begin{gathered}\text{máx}\, z = 2x_1 + x_3 - \infty x_7 \\ \begin{array}{rcrcrcrcrcrcr} -x_1 & + & 3x_2 & - & 2x_3 & + & x_4 & & & & & = & 10 \\ x_1 & & & - & 2x_3 & & & + & x_5 & & & = & 2 \\ 2\,x_1 & + & 6x_2 & + & 3x_3 & & & & & - & x_6 + x_7 & = & 7 \end{array} \\ \mathbf{x} \geq \mathbf{0}\end{gathered}$$

   que se ha añadido también a la función objetivo con un coeficiente $-\infty$, puesto que su sentido es puramente técnico (se ha usado sólo para forzar la aparición de la matrix identidad como base factible inicial) y, por lo tanto, no debería aparecer en la solución final del SIMPLEX —salvo que el problema sea infactible, en cuyo caso, alguna variable artificial tomará un valor estrictamente positivo.

4. El algoritmo del SIMPLEX consiste en la aplicación iterativa de tres pasos: cálculo de las variables básicas, selección de la variable de entrada y selección de la variable de salida hasta que se detecte alguna de las siguientes condiciones:

- El problema puede mejorar el valor de la función objetivo indefinidamente. Se dice entonces que el problema está *no acotado*. Este caso se detecta cuando todas las componentes y_i de la variable de decisión x_i elegida para entrar en la base son todos negativos o nulos.
- El problema es irresoluble. Esto ocurre cuando en el segundo paso, todos los costes reducidos son positivos y el primer paso asignó un valor no negativo a alguna variable artificial.
- Se alcanza una solución factible y puede demostrarse que no es posible mejorarla. Esta condición se detecta como en el segundo caso pero cuando las variables artificiales (si las hubiera) tienen valores nulos.
- El problema tiene soluciones infinitas. En este caso todos los costes reducidos calculados en el segundo paso son positivos y hay, al menos, uno con coste nulo. La arista que une el punto calculado en la iteración actual con aquél que se alcanzaría si entra una variable con coste reducido nulo (cuando no hay costes reducidos negativos) son todas soluciones óptimas de la tarea de Programación Lineal.

Paso 0 Cálculo de una solución factible inicial

a) Cálculo de las variables básicas
La primera iteración se inicia con una base igual a la matriz identidad de dimensión 3, tal y como se calculó ya en el apartado anterior. Por lo tanto, son variables básicas en este paso $\{x_4, x_5, x_7\}$, precisamente en este orden puesto que son precisamente las columnas $\mathbf{a_4}$, $\mathbf{a_5}$ y $\mathbf{a_7}$ las que recrean la matriz identidad de rango 3, I_3:

$$B_0 = I_3 \qquad B_0^{-1} = I_3$$

$$x_0^* = B_0^{-1}b = b = \begin{pmatrix} 10 \\ 2 \\ 7 \end{pmatrix} \quad z_0^* = c_{B_0}^T x_0^* = \begin{pmatrix} 0 & 0 & -\infty \end{pmatrix} \begin{pmatrix} 10 \\ 2 \\ 7 \end{pmatrix} = -7\infty$$

Obviamente, $-7\infty = -\infty$. Sin embargo, ∞ se trata aquí simbólicamente como una variable con un valor arbitrariamente grande. Por lo tanto, es buena práctica preservar los coeficientes para poder hacer comparaciones con otras expresiones que también contengan el mismo término.

b) Selección de la variable de entrada
En las expresiones siguientes el cálculo de los vectores y_i se ha embebido en el cálculo de los *costes reducidos* directamente (aunque en una iteración con una base igual a la matriz identidad, $y_i = a_i$):

$$z_1 - c_1 = c_{B_0}^T B_0^{-1} a_1 - c_1 = \begin{pmatrix} 0 & 0 & -\infty \end{pmatrix} I_3 \begin{pmatrix} -1 \\ 1 \\ 2 \end{pmatrix} - 2 = -2\infty - 2$$

$$z_2 - c_2 = c_{B_0}^T B_0^{-1} a_2 - c_2 = \begin{pmatrix} 0 & 0 & -\infty \end{pmatrix} I_3 \begin{pmatrix} 3 \\ 0 \\ 6 \end{pmatrix} - 0 = -6\infty$$

$$z_3 - c_3 = c_{B_0}^T B_0^{-1} a_3 - c_3 = \begin{pmatrix} 0 & 0 & -\infty \end{pmatrix} I_3 \begin{pmatrix} -2 \\ -2 \\ 3 \end{pmatrix} - 1 = -3\infty - 1$$

$$z_6 - c_6 = c_{B_0}^T B_0^{-1} a_6 - c_6 = \begin{pmatrix} 0 & 0 & -\infty \end{pmatrix} I_3 \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} + \infty = +2\infty$$

Tal y como se indicaba anteriormente, ∞ se ha utilizado como un símbolo cualquiera. También es posible usar una constante M muy alta (y, en particular, un valor de M mayor que la suma

de los valores absolutos de todos los coeficientes en la función objetivo). Usando ∞ como un símbolo cualquiera es posible saber qué valores son más grandes sustituyéndolo por valores arbitrariamente grandes.
En particular, sustituyendo ∞ por cualquier valor positivo arbitrariamente grande, el coste reducido de x_2 será siempre más negativo que el del resto de las variables no básicas y, por tanto, la variable elegida para entrar en la siguiente base será percisamente x_2.

c) Selección de la variable de salida
La regla de salida establece que debe salir aquella variable con el menor cociente x_i/y_{ij} (con valores y_{ij} estrictamente positivos) donde x_i es la variable elegida en el paso anterior (x_2):

$$\text{mín}\left\{\frac{10}{3}, \frac{2}{0}, \frac{7}{6}\right\}$$

y sale la variable x_7 que es la que se corresponde con la tercera fracción (puesto que ocupa la tercera posición en la base), precisamente la variable artificial.

Paso 1 Mejora de la solución actual (iteración #1)

a) Cálculo de las variables básicas
A continuación se mejora la calidad de la solución anterior. Las nuevas variables básicas son $\{x_2, x_4, x_5\}$:

$$B_1 = \begin{pmatrix} 3 & 1 & 0 \\ 0 & 0 & 1 \\ 6 & 0 & 0 \end{pmatrix} \qquad B_1^{-1} = \begin{pmatrix} 0 & 0 & \frac{1}{6} \\ 1 & 0 & -\frac{1}{2} \\ 0 & 1 & 0 \end{pmatrix}$$

$$x_1^* = B_1^{-1}b = \begin{pmatrix} \frac{7}{6} \\ \frac{13}{2} \\ 2 \end{pmatrix} \quad z_1^* = c_{B_1}^T x_1^* = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{7}{6} \\ \frac{13}{2} \\ 2 \end{pmatrix} = 0$$

Como puede verse, la función objetivo ha crecido(como debe ser, puesto que se trata de un problema de maximización) respecto de la iteración anterior de -7∞ a 0.

b) Selección de la variable de entrada
Como antes, el cálculo de los vectores columna $y_i = B_1^{-1}a_i$ se ha embebido en el cálculo de los costes reducidos, pero nótese que no es preciso hacerlos puesto que en todos los casos se emplea el vector c_{B_1} que, en esta iteración es $\mathbf{0}$:

$$\begin{aligned} z_1 - c_1 &= c_{B_1}^T B_1^{-1} a_1 - c_1 = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} B_1^{-1}a_1 - 2 = -2 \\ z_3 - c_3 &= c_{B_1}^T B_1^{-1} a_3 - c_3 = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} B_1^{-1}a_3 - 0 = 0 \\ z_6 - c_6 &= c_{B_1}^T B_1^{-1} a_6 - c_6 = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} B_1^{-1}a_6 - 0 = 0 \\ z_7 - c_7 &= c_{B_1}^T B_1^{-1} a_7 - c_7 = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} B_1^{-1}a_7 + \infty = +\infty \end{aligned}$$

y la variable x_1 debe entrar en la base puesto que es la única que tiene un coste reducido estrictamente negativo. Nótese que en este paso se ha calculado también el coste reducido de la variable artificial, x_7 que siempre dará un valor positivo arbitrariamente grande puesto que $c_7 = -\infty$ de modo que en el cálculo de su coste reducido sumará $+\infty$ y, por lo tanto, es imposible que vuelva a entrar nunca en la base. Por lo tanto, en lo sucesivo no se volverá a calcular.

c) Selección de la variable de salida
Como siempre, la variable de salida se calcula en atención al mínimo cociente x_1/y_{ij} (con valores y_{ij} estrictamente positivos) donde x_i es la variable elegida en el paso anterior para añadirse a la base (x_1) e y_{ij} son las componentes de su vector $\mathbf{y}$. Como en el paso anterior no se calculó ningún vector y_i porque se explotó la observación de que $c_{B_1} = \mathbf{0}$, se hace a continuación:

$$y_1 = B_1^{-1}a_1 = \begin{pmatrix} 0 & 0 & \frac{1}{6} \\ 1 & 0 & -\frac{1}{2} \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} -1 \\ 1 \\ 2 \end{pmatrix} = \begin{pmatrix} \frac{1}{3} \\ -2 \\ 1 \end{pmatrix}$$

Por lo tanto, el cálculo de la regla de salida queda como se muestra a continuación:

$$\text{mín}\left\{ \frac{\frac{7}{6}}{\frac{1}{3}}, \frac{\frac{13}{2}}{\not{-2}}, \frac{2}{1} \right\}$$

y sale la variable x_5, que es la que ocupa la tercera posición en la base actual.

Paso 2 Mejora de la solución actual (iteración #2)

a) Cálculo de las variables básicas
Las nuevas variables básicas son $\{x_1, x_2, x_4\}$

$$B_2 = \begin{pmatrix} -1 & 3 & 1 \\ 1 & 0 & 0 \\ 2 & 6 & 0 \end{pmatrix} \qquad B_2^{-1} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & -\frac{1}{3} & \frac{1}{6} \\ 1 & 2 & -\frac{1}{2} \end{pmatrix}$$

$$x_2^* = B_2^{-1}b = \begin{pmatrix} 2 \\ \frac{1}{2} \\ \frac{21}{2} \end{pmatrix} \quad z_2^* = c_{B_2}^T x_2^* = \begin{pmatrix} 2 & 0 & 0 \end{pmatrix} \begin{pmatrix} 2 \\ \frac{1}{2} \\ \frac{21}{2} \end{pmatrix} = 4$$

b) Selección de la variable de entrada
A continuación se muestra el cálculo de los costes reducidos de todas las variables no básicas, de las que se ha omitido únicamente, el de la variable artificial x_7 puesto que al hacer la diferencia con su coeficiente en la función objetivo resultará siempre un valor positivo arbitrariamente grande, tal y como se advirtió ya en el paso anterior:

$$z_3 - c_3 = c_{B_1}^T B_1^{-1} a_3 - c_3 = \begin{pmatrix} 2 & 0 & 0 \end{pmatrix} \begin{pmatrix} -2 \\ \frac{7}{6} \\ -\frac{15}{2} \end{pmatrix} - 1 = -5$$

$$z_5 - c_5 = c_{B_1}^T B_1^{-1} a_5 - c_5 = \begin{pmatrix} 2 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ -\frac{1}{3} \\ 2 \end{pmatrix} - 0 = 2$$

$$z_6 - c_6 = c_{B_1}^T B_1^{-1} a_6 - c_6 = \begin{pmatrix} 2 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ -\frac{1}{6} \\ \frac{1}{2} \end{pmatrix} - 0 = 0$$

de donde resulta que la variable x_3 debe entrar en la base, puesto que es la única que tiene un coste reducido negativo.

c) Selección de la variable de salida
El paso anterior mostraba el cálculo de todos los vectores $\mathbf{y}$, de modo que a continuación se usan los valores de y_3 en el denomiador de la regla de salida, puesto que la variable elegida para entrar es x_3:

$$\text{mín}\left\{ \frac{\not{2}}{\not{-2}}, \frac{\frac{1}{2}}{\frac{7}{6}}, \frac{\frac{21}{2}}{\not{-\frac{15}{2}}} \right\}$$

y sale la variable x_2, que es la única que podía hacerlo, puesto que las otras tienen denominadores nulos.

Paso 3 Mejora de la solución actual (iteración #3)

a) Cálculo de las variables básicas
Las nuevas variables básicas son $\{x_1, x_3, x_4\}$

$$B_3 = \begin{pmatrix} -1 & -2 & 1 \\ 1 & -2 & 0 \\ 2 & 3 & 0 \end{pmatrix} \qquad B_3^{-1} = \begin{pmatrix} 0 & \frac{3}{7} & \frac{2}{7} \\ 0 & -\frac{2}{7} & \frac{1}{7} \\ 1 & -\frac{1}{7} & \frac{4}{7} \end{pmatrix}$$

$$x_3^* = B_3^{-1}b = \begin{pmatrix} \frac{20}{7} \\ \frac{3}{7} \\ \frac{96}{7} \end{pmatrix} \quad z_3^* = c_{B_3}^T x_3^* = \begin{pmatrix} 2 & 1 & 0 \end{pmatrix} \begin{pmatrix} \frac{20}{7} \\ \frac{3}{7} \\ \frac{96}{7} \end{pmatrix} = \frac{43}{7}$$

y, como puede verse, el valor de la función objetivo ha vuelto a crecer nuevamente, de 4, a algo más de 6 unidades.

b) Selección de la variable de entrada
A continuación se muestra el cálculo de los costes reducidos de todas las variables no básicas, de las que se ha omitido únicamente, el de la variable artificial x_7 de nuevo::

$$z_2 - c_2 = c_{B_3}^T B_3^{-1} a_2 - c_2 = \begin{pmatrix} 2 & 1 & 0 \end{pmatrix} \begin{pmatrix} \frac{12}{7} \\ \frac{6}{7} \\ \frac{45}{7} \end{pmatrix} - 0 = \frac{30}{7}$$

$$z_5 - c_5 = c_{B_3}^T B_3^{-1} a_5 - c_5 = \begin{pmatrix} 2 & 1 & 0 \end{pmatrix} \begin{pmatrix} \frac{3}{7} \\ -\frac{2}{7} \\ -\frac{1}{7} \end{pmatrix} - 0 = \frac{4}{7}$$

$$z_6 - c_6 = c_{B_3}^T B_3^{-1} a_6 - c_6 = \begin{pmatrix} 2 & 1 & 0 \end{pmatrix} \begin{pmatrix} -\frac{2}{7} \\ -\frac{1}{7} \\ -\frac{4}{7} \end{pmatrix} - 0 = -\frac{5}{7}$$

de donde resulta que la variable x_6 debe entrar en la base, puesto que es la única que tiene un coste reducido negativo.

c) Selección de la variable de salida
En el cálculo de la variable de salida se observa ahora que:

$$\text{mín} \left\{ \frac{\frac{20}{7}}{\not{-}\frac{2}{7}}, \frac{\frac{3}{7}}{\not{-}\frac{1}{7}}, \frac{\frac{96}{7}}{\not{-}\frac{4}{7}} \right\}$$

de modo que el problema es *no acotado*, puesto que todos los denominadores son negativos. Con ello, concluye la aplicación del SIMPLEX.

5. La interpretación de un problema incluye varias consideraciones como son estudiar: si el problema es o no satisfacible, si la solución es única o hay varias soluciones o si está o no acotado. Además, debe estudiarse el uso de recursos: si sobra o no alguno y cual es su contribución al crecimiento de la función objetivo.

Interpretación de la solución De la solución se puede advertir lo siguiente:

- El problema es factible porque la solución no contiene valores positivos para ninguna variable artificial.
- El valor de la función objetivo no está acotado puesto que los denominadores calculados en el tercer paso de la última iteración son todos negativos.
- Por lo tanto, no existe una solución óptima a la tarea de Programación Lineal propuesta, puesto que existen siempre una cantidad infinita de puntos de la región factible que mejorarán el valor de la función objetivo de cualquier otro.

Interpretación de los recursos La interpretación de los recursos se hace, fundamentalmente, observando los valores de la variable de holgura en la solución óptima y la solución del problema dual. Sin embargo, en este caso no existe una solución óptima y, por lo tanto, no es posible llevar a cabo ninguna interpretación de los recursos.

Problema 4

1. El espacio de estados es una formalización que habilita la aplicación de algoritmos de búsqueda (informados o no) para la resolución de problemas. Consiste únicamente, en la definición de estados y operadores que sirven para transitar entre ellos. Relacionando, después, el conjunto de posibles estados con otro de vértices V y el de operadores (convenientemente instanciados) con otro de arcos E, resulta entonces de forma natural la definición de un grafo, el *grafo de búsqueda* que se recorrerá eficientemente con el uso de *árboles de búsqueda*. Este ejercicio propone ejercitar todos estos conceptos y, en el primer apartado, el del espacio de estados.

   Estados En este problema un estado se representa con información de las mercancías y la ocupación de los camiones. A continuación se presentan definiciones estructuradas para cada uno de estos conceptos:

   Mercancía Cada mercancía `m` está caracterizada con su peso y tipo con los atributos `m.peso` y `m.tipo` —donde el primero podría ser un número entero y, el segundo, una cadena de caracteres por ejemplo. Además, es necesario llevar la cuenta de aquellas mercancías que se han cargado ya en algún camión o no. Para ello, se usará el campo `m.camion` que valdrá originalmente -1 y, en otro caso, tendrá el identificador del camión donde se ha cargado.

   Camión Cada camión `c` de la flota está caracterizado por el peso máximo que puede transportar, `c.peso_maximo`, y un contenedor con los tipos que puede llevar, `c.tipos`. Además, tendrá también otro contenedor para almacenar instancias de las mercancías que contiene, `c.mercancias`. Por último, será necesario distinguir cada camión con un índice `c.id` para identificar correctamente el camión que transporta cada mercancía.

   En el estado inicial se verifica que $\texttt{m.camion} = -1$ para todos los camiones de la flota. De otra parte, en el estado final se verifica que $\texttt{m.peso} > \texttt{c.peso\_maximo} - \sum_{m' \in \texttt{c.mercancias}} \texttt{m}'\texttt{.peso}$ para todos los camiones `c` y mercancías `m`. Es decir, que ya no caben más mercancías en ningún camión o, si pudiera caber alguna: $\texttt{m.peso} \leq \texttt{c.peso\_maximo} - \sum_{m' \in \texttt{c.mercancias}} \texttt{m}'\texttt{.peso}$, entonces es que el tipo de la mercancía, `m.tipo` no está entre los que pueden llevarse en ese camión, $\texttt{m.tipo} \notin \texttt{c.tipos}$.

   Acciones En este problema hay una única acción, `cargar`, que toma como parámetros una mercancía `m`, y un camión `c`, y realiza la acción de carga.

   Para la identificación de las precondiciones de este operador, es importante observar que el tipo de la mercancía debe estar entre aquellos que pueden transportarse en el camión, $\texttt{m.tipo} \in \texttt{c.tipos}$. Además, la nueva carga no debe superar la carga máxima del camión: $\texttt{m.peso} + \sum_{m' \in \texttt{c.mercancias}} \texttt{m}'\texttt{.peso} \leq \texttt{c.peso\_maximo}$. Por supuesto, la mercancía `m` que se desea cargar no debe estar en ningún otro camión, $\texttt{m.camion} = -1$.

   Las postcondiciones describen el efecto de aplicar el operador, y consisten en añadir la nueva mercancía a las que hay en el camión elegido: $m \in \texttt{c.mercancias}$ y, además, en actualizar la información del camión que ahora acoge la mercancía, $\texttt{m.camion} = c.id$.

   Por último, es importante determinar el coste de cada una de estas acciones. Para ello, es importante observar que la métrica que se quiere minimizar es el número de objetos que no se han cargado, de modo que cada acción de carga tendría, intuitivamente hablando, un coste igual a -1. Con ello, el problema podría reformularse como un problema de maximización de objetos cargados en todos los camiones, cambiando el signo a $+1$. En su lugar, se sugiere enfocarlo, como se pedía en el enunciado, como un problema de minimización donde cada operador tendría coste 0, salvo el último (aquel que,

una vez aplicado, ya no puede seguirse de otras cargas porque el estado alcanzado es un estado final), cuyo coste sería igual al número de mercancías `m` que no han sido cargadas, esto es, con `m.camion` $= -1$.

2. Cualquier algoritmo de búsqueda debe considerar, como máximo, la carga de n mercancías, con n el número de objetos a cargar dado incialmente, independiente del tipo y naturaleza del algoritmo de búsqueda empleado, y esa será, por tanto, la profundidad máxima pedida.

   Esto es cierto sólo si se pide resolver el problema óptimamente (tal y como se hace en el enunciado), puesto que cualquier solución subóptima puede consistir en la carga de un número menor de mercancías en los camiones.

3. En este problema el algoritmo de búsqueda en amplitud no vale, puesto que la prueba de su admisibilidad se basa en la observación de que cualquier solución encontrada a nivel d es necesariamente mejor que cualquier otra encontrada a nivel d', si $d' > d$. En este caso ocurre literalmente al contrario, y las soluciones preferidas serán aquellas que consistan en secuencias de carga más largas.

   Por lo tanto, para resolver el problema óptimamente sólo se pueden considerar o el algoritmo del *primero en profundización iterativa* o el algoritmo de *ramificación y acotación en profundidad*:

   Primero en profundización iterativa Inicialmente se invoca el algoritmo del primero en profundidad con un umbral igual a la unidad, de modo que se enumeran todas las formas de cargar una, y sólo una mercancía en los camiones de la flota. Si algún nodo terminal generado de esta manera admite la carga de mercancías adicionales (lo que puede comprobarse fácilmente), entonces se incrementa el umbral de la siguiente iteración en otra unidad. Cuando en una iteración no sea posible extender ninguno de los nodos terminales, entonces la solución óptima fue calculada en la iteración anterior.

   La ventaja de este algoritmo es que tiene un consumo de memoria lineal. Sin embargo, debe re-expandir todas las *transposiciones*.

   Ramificación y acotación en profundidad El algoritmo de ramificación y acotación en profundidad aplicaría el algoritmo del primero en profundidad con un umbral de profundidad máxima igual a $+\infty$. Esto no provocará ningún error puesto que necesariamente se debe encontrar una solución inicial incluso si las mercancías se cargan en los camiones aleatoriamente —pero satisfaciendo las restricciones. A partir de ahí, utiliza el coste de la solución encontrada tentativamente para podar otras alternativas.

   Sin embargo, la acción de poda no será posible en este caso: si el algoritmo encuentra una solución a profundidad d es porque ha encontrado una manera de cargar hasta d paquetes. Sin una función heurística que guíe el proceso de búsqueda, cualquier nodo a una profundidad d' inferior, $d' < d$ debe ser expandido con la esperanza de encontrar una solución a una profundidad d'' mayor, $d'' > d$. Por otra parte, los nodos a profundidades mayores que la mejor solución encontrada hasta ahora deben expandirse siempre que sea posible, puesto que son soluciones mejores que la que se conocía.

   El algoritmo de ramificación y acotación en profundidad también tiene un consumo de memoria lineal, y como el algoritmo anterior, también debe re-expandir todas las *transposiciones*.

   En realidad, cualquiera de estos algoritmos podría recomendarse incluso si el segundo no puede podar nodos como hace habitualmente. De hecho, cualquiera de ellos examina, simplemente, todas las combinaciones de carga hasta que no pueden llenarse más los camiones para devolver la mejor configuración.

4. Una heurística muy sencilla que puede obtenerse con la técnica de *relajación de restricciones*, consiste en relajar el peso máximo que puede transportar cada camión.

   Ahora bien, asumir que cada camión puede cargar un peso arbitrario, produce una heurística no informada, en el que para todos los nodos dirá que el número de mercancías que se quedarán sin cargar es siempre igual a 0 —dado que haya un camión utilizable por cada tipo de mercancía.

En su lugar se sugiere relajar la restricción de los tipos que pueden cargarse en cada camión. Asumiendo, en su lugar, que cada camión puede cargar mercancías de cualquier tipo, entonces es posible estimar el número de mercancías que pueden cargarse en cada nodo con el siguiente procedimiento:

a) Se calcula primero el peso adicional que puede cargar cada camión:

$$W = \sum_{\forall c} \texttt{c.peso\_maximo} - \sum_{m' \in \texttt{c.mercancias}} \texttt{m'.peso}$$

b) Se ordenan todas las mercancías `m` que no se hayan cargado en ningún camion ($\texttt{m.camion} = -1$) en orden ascendente por su peso, `m.peso`

c) El valor de la heurística será el número de mercancías escogidas de la lista ordenada en el paso anterior tal que la suma de sus pesos no exceda el valor W calculado en el primer paso

5. La selección de un algoritmo de búsqueda informada para este caso depende, como en el apartado anterior, del número de transposiciones y también, naturalmente, de la dificultad de los problemas:

A* El algoritmo A* es admisible y garantiza, por lo tanto, que encontrará soluciones óptimas si la función heurística que lo guía también es admisible. Además, es un algoritmo rápido puesto que no reexpande nodos (y, con frecuencia, las ordenaciones de la lista abierta se pueden hacer en $O(1)$ con las estructuras de datos adecuadas si la función objetivo sólo toma valores enteros). Sin embargo, tiene un consumo de memoria exponencial.

IDA* El algoritmo IDA* reexpande nodos en caso de que haya transpociones pero no ordena nodos y, mucho más importante aún, tiene un consumo de memoria lineal en la profundidad de la solución (que en nuestro caso es siempre igual a N). Además, también es un algoritmo de búsqueda admisible.

En la selección de un algoritmo de búsqueda de fuerza bruta no se consideró el problema de las transposiciones, puesto que la única alternativa viable eran algoritmos de búsqueda del primero en profundidad. Ahora sí se hará porque el algoritmo A* sí es una estrategia viable.

De hecho, este problema tiene una cantidad exponencial de transposiciones puesto que la misma configuración de carga puede conseguirse simplemente permutando las aplicaciones del operador `cargar`.

Por ello, se recomienda el algoritmo A*, si bien su ejecución será como la del algoritmo de búsqueda de el mejor primero conocido como de *heurística pura* puesto que el coste de todos los operadores es igual a 0.