Departamento de Informática
Universidad Carlos III de Madrid

Grado en Informática

Heurística y Optimización

20 de Enero de 2017

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 (1 puntos)

Macarena dispone de N cajas y quiere apilarlas todas verticalmente. Cada caja tiene un peso $w_i, 0 \leq i < N$ y una capacidad para soportar peso s_i diferente, pero todas son indistintas en cuanto a su contenido. Cada caja debe soportar su propio peso y el de aquellas que se dispongan sobre élla, de modo que algunas combinaciones son imposibles. Macarena desea maximizar el número de cajas que puede disponer en total —independientemente del peso de cada una.

Se pide responder razonadamente las siguientes preguntas:

(a) **(½ puntos)** Considérese primero el caso en que todas las cajas tienen exactamente la misma capacidad para soportar peso $s_i, 0 \leq i < N$.

Se pide resolver el problema de maximizar el número de cajas que pueden apilarse entre un conjunto de N cajas con la técnica de *Programación Dinámica*, teniendo en cuenta que la capacidad de todas las cajas es la misma.

(b) **(½ puntos)** Considérese ahora el caso en que cada caja tiene una capacidad para soportar peso diferente. ¿Es posible aplicar el algoritmo diseñado en el apartado anterior para resolver este problema óptimamente?

Pregunta 2 (1 puntos)

Adriana debe irse de viaje y tiene que hacer el equipaje. Le gustaría llevarse consigo hasta N items, cada uno de ellos con un volumen $v_i, 0 \leq i < N$ diferente, pero su maleta tiene una capacidad limitada igual a C. Sabiendo que Adriana le concede a cada item un valor u_i (de modo que los más preferidos son aquellos con un valor mayor), se desea determinar los objetos que puede llevarse consigo que maximicen el valor de la colección elegida, mientras que no se excede el volumen de la maleta.

Se pide responder razonadamente las siguientes preguntas:

(a) **(½ puntos)** Modelizar el enunciado como un problema de *Programación Lineal*.

Indica claramente el significado de las variables de decisión elegidas, y el significado de cada restricción y de la función objetivo.

(b) **(½ puntos)** En otro viaje, Adriana considera dos objetos diferentes n_1 y n_2 que, o necesariamente deben ir juntos, o no deben estar ninguno en la selección. Por otra parte, considera otros dos objetos n_3 y n_4, de modo que si se incluye uno, entonces no debe estar el otro.

¿Es necesario hacer alguna modificación en el modelo del apartado anterior para verificar estas restricciones adicionales? Si o no y por qué.

Si fuera preciso hacer alguna modificación detállala claramente, incluso reformulando el modelo si fuera preciso.

Pregunta 3 (1½ puntos)

Considera la fórmula F_1 en Forma Normal Conjuntiva formada por las siguientes cláusulas:

$$\begin{array}{ll} C_1{:}(x_1 \vee x_4 \vee x_5) & C_5{:}(x_3 \vee \bar{x}_4) \\ C_2{:}(\bar{x}_1 \vee \bar{x}_5 \vee x_2) & C_6{:}(\bar{x}_4) \\ C_3{:}(\bar{x}_2 \vee x_3 \vee x_4) & C_7{:}(\bar{x}_3 \vee \bar{x}_5) \\ C_4{:}(\bar{x}_3 \vee x_4) & C_8{:}(x_5) \end{array}$$

Se pide responder razonadamente las siguientes preguntas:

(a) **(½ punto)** ¿La fórmula F_1 tiene literales puros? Si fuera así, ¿qué fórmula F_1' resultaría de hacer la resolución respecto de ellos?

(b) **(½ puntos)** Aplicar Davis-Putnam (DP) para encontrar un modelo que satisfaga la fórmula F_1' obtenida en el paso anterior, 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 enumerar todos los modelos que satisfagan la fórmula F_1', en caso de que exista alguno.

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

Pregunta 4 (3½ puntos)

Considerése el siguiente problema de Programación Lineal:

$$\begin{array}{rrcrcl} & \multicolumn{5}{c}{\max z = x_1 - 2x_2} \\ & x_1 & + & 3x_2 & \geq & 2 \\ - & 2x_1 & + & 3x_2 & \leq & 4 \\ & 7x_1 & - & 5x_2 & \leq & 8 \\ & \multicolumn{5}{c}{\mathbf{x} \geq \mathbf{0}} \end{array}$$

Se pide responder razonadamente las siguientes preguntas:

(a) **(½ puntos)** Resolver el problema utilizando el método de *resolución gráfica*

(b) **(½ puntos)** Resolver el mismo problema utilizando el método de *resolución gráfica*, pero considerando la restricción adicional de que x_1 y x_2 deben tomar el mismo valor.

Y ahora, considerése otro problema de Programación Lineal:

$$\begin{array}{rrcrcrcl} & \multicolumn{7}{c}{\max z = x_1 - 2x_2 + 3x_3} \\ & x_1 & - & 2x_2 & + & 3x_3 & \leq & 2 \\ - & 2x_1 & + & 3x_2 & + & 5x_3 & \geq & 4 \\ & 3x_1 & - & 5x_2 & - & 7x_3 & \leq & 8 \\ & \multicolumn{7}{c}{\mathbf{x} \geq \mathbf{0}} \end{array}$$

Se pide responder razonadamente las siguientes preguntas:

(c) **(½ puntos)** Expresar esta tarea de Programación Lineal en forma *estándar* de maximización de modo que, además, sea posible iniciar la aplicación del algoritmo SIMPLEX con una base igual a 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.

(f) **(½ puntos)** Calcula la contribución por unidad de recurso al crecimiento de la función objetivo.

Pregunta 5 (3 puntos)

Una empresa dispone de una servicio de impresión que consta de una única impresora. La impresora acepta hasta N peticiones, para cada una de las cuales sabe el tiempo exacto que tardará en llevarlas a cabo, $t_i, 0 \leq i < N$. Además, cada petición está detallada con una hora a partir de la cual se recogerá el trabajo, e_i. Si algún trabajo acabara antes de la hora indicada, entonces es preciso guardarlo incurriendo entonces en un coste c_i.

Dadas N peticiones de impresión, se desea crear un sistema automático que determine la hora a la que debe iniciarse la impresión y el orden a seguir, de modo que se minimice la suma de todas las penalizaciones de aquellos trabajos que acaben antes de su hora indicada. Para ello, debe tenerse en cuenta que una vez que la impresora empieza a imprimir trabajos, ya lo hace ininterrumpidamente.

Se pide responder razonadamente las siguientes preguntas:

(a) **(1 punto)** Representar el problema como un *espacio de estados*

(b) **(½ puntos)** ¿Cuál es el tamaño del espacio de estados?

(c) **(½ puntos)** ¿Qué profundidad tendría un árbol de búsqueda desarrollado por un algoritmo de búsqueda de fuerza bruta para resolver óptimamente este problema? ¿Y cuál sería su factor de ramificación?

(d) **(½ puntos)** Habida cuenta de que queremos encontrar soluciones óptimas, ¿qué algoritmo de búsqueda *no informada* sugerirías para su resolución?

(e) **(½ puntos)** Suponiendo que se dispone de una función heurística $h(n)$ admisible, ¿qué algoritmo de búsqueda *heurística* es el más indicado para resolver este problema óptimamente? ¿Por qué?

Soluciones del examen de Heurística y Optimización Enero 2017

Problema 1

Este problema está tomado de una pregunta realizada en `stackexchange.com`[1]. En lo sucesivo, sin embargo, se considerará únicamente el caso en el que la capacidad de todas las cajas es la misma, en vez del caso general donde cada caja puede tener una capacidad diferente.

1. En el primer apartado, se asume que la capacidad para soportar peso de todas las cajas es idéntico y, por lo tanto, igual a un valor constante $s_i = k, 0 \leq i < N$. El problema, por lo tanto, consiste en determinar el máximo número de cajas que pueden apilarse de modo que el peso de la pila soportado por cada una no exceda su capacidad, k.

   De hecho, cualquier pila de cajas con un peso menor o igual que k es una solución correcta, puesto que eso significa que todas las cajas elegidas pueden soportar su propio peso más el de aquellas dispuestas sobre éllas. En particular, la solución óptima consistirá entonces en una pila de cajas cuyo peso máximo acumulado no sea mayor que k, exactamente igual que en el *problema de la mochila* que, como este problema, está acotado por una capacidad máxima.

   Por otra parte, las cajas son indistintas y se puede elegir cualquiera de ellas. Por ello, este problema es equivalente al *problema de la mochila* donde la utilidad de cada item (o caja) es el mismo, e igual a la unidad. Efectivamente, si la utilidad de los objetos escogidos es siempre la misma, entonces maximizar la utilidad es estrictamente equivalente a maximizar el número de cajas.

   Por lo tanto, este problema se resuelve aplicando el algoritmo del *problema de la mochila* donde el peso de cada objeto es el de caja, y donde la capacidad de la mochila es igual a k, la capacidad para soportar peso de todas las cajas.

2. Si cada caja tiene una capacidad para soportar pesos diferentes, entonces el problema ya no es como el de la mochila y, por lo tanto, la solución anterior no puede aplicarse.

Problema 2

El enunciado del problema es, exactamente, el *problema de la mochila* (que ya se había considerado en la solución del primer ejercicio). En este ejercicio, en vez de resolverlo con el consabido algoritmo de *Programación Dinámica*, se sugiere su modelización con técnicas de *Programación Lineal Entera*.

1. Como en cualquier problema de modelización de *Programación Lineal*, los pasos consisten en determinar las *variables de decisión*, la representación de las *restricciones* y, por último, de la *función objetivo*:

   Variables de decisión Para la modelización que sigue se sugiere el uso de variables que determinen si cada objeto se selecciona para ser introducido en la maleta o no:

   $$x_i = \begin{cases} 1 & \text{si el objeto } i\text{-ésimo es escogido} \\ 0 & \text{en caso contrario} \end{cases}$$

   Puesto que todas las variables de decisión sólo pueden escoger uno entre los valores $\{0, 1\}$, se trata de un problema de *Programación Lineal Entera Pura* con variables *binarias*.

   Restricciones La única restricción del problema es que la suma del volumen de todos los ítems escogidos no superen el volumen total de la mochila, C. Si x_i es no nulo sólo cuando es escogido, entonces $\sum_{i=1}^{N} x_i v_i$ es la suma del volumen de todos los objetos introducidos en la maleta. Por lo tanto, la única restricción del problema es:

   $$\sum_{i=1}^{N} x_i v_i \leq C$$

[1] Ver `http://stackoverflow.com/questions/41361652/object-stacking-dynamic-programming`

donde C es la capacidad total de la mochila.

Como de costumbre, debe añadirse que las variables de decisión tomen valores no negativos pero, en este caso se exigirá que sean binarias:

$$x_i \in \{0, 1\}$$

Función objetivo Por último, el objetivo del problema consiste en maximizar la utilidad de todos los objetos escogidos. Razonando como antes resulta:

$$\text{máx}\, z = \sum_{i=1}^{N} x_i u_i$$

Por lo tanto, el modelo queda como sigue:

$$\begin{array}{rcl} \text{máx}\, z = \sum_{i=1}^{N} x_i u_i & & \\ \sum_{i=1}^{N} x_i v_i & \leq & C \\ x_i \in \{0, 1\} & & \end{array}$$

2. Para la modelización de reglas del tipo *condicional* se suelen usar variables binarias, de hecho como las mismas que se han definido en el modelo del primer apartado.

   A propósito de la primera condición, es posible obligar al modelo de Programación Lineal a que escoja n_1 y n_2 al mismo tiempo o no, simplemente haciendo que tomen el mismo valor, $x_1 = x_2$ o, equivalentemente:

   $$x_1 - x_2 = 0$$

   A propósito de la segunda condición, basta con hacer que si una variable (por ejemplo x_3) toma el valor 1, la otra tome el valor 0 (en nuestro ejemplo, x_4), $x_3 = 1 - x_4$, que se reescribe como:

   $$x_3 + x_4 = 1$$

   Nótese que las expresiones anteriores tienen el comportamiento deseado porque las variables empleadas son binarias. En consecuencia, el nuevo modelo de Programación Lineal es:

   $$\begin{array}{rcl} \text{máx}\, z = \sum_{i=1}^{N} x_i u_i & & \\ \sum_{i=1}^{N} x_i v_i & \leq & C \\ x_1 - x_2 & = & 0 \\ x_3 + x_4 & = & 1 \\ x_i \in \{0, 1\} & & \end{array}$$

Problema 3

1. Un literal ℓ es puro si y sólo si no aparece negado en ninguna parte de la fórmula F_1, esto es, si $\bar{\ell} \notin F_1$. Ahora bien, es fácil verificar que todas las variables aparecen afirmadas y negadas en la fórmula de lógica proposicional dada en el enunciado y, por lo tanto, no existen literales puros.

   Puesto que no hay literales puros, no puede aplicarse la resolución pedida, y en lo sucesivo se dirá que $F_1' = F_1$.

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 $G_0 = \{C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8\}$

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. Por lo tanto, después de calcular la disyunción de los literales que acompañan a un literal y el otro en cada cláusula respectivamente se obtiene: $(x_4 \vee x_5 \vee \bar{x}_5 \vee x_2)$ que, de hecho, es una *tautología* (es decir, una expresión que siempre se verifica), puesto que contiene un literal, x_5, y su negado, $\bar{x}_5$. En este caso, el método de resolución no produce ninguna cláusula nueva, resultando entonces el resto de cláusulas:

$$\text{Res}(G_0, x_1) = \{C_3, C_4, C_5, C_6, C_7, C_8\}$$

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

$$G_0 \backslash \text{Res}(G_0, x_1) = G_0 \backslash \{C_3, C_4, C_5, C_6, C_7, C_8\} = \{C_1, C_2\}$$

Paso 1 $G_1 = \{C_3, C_4, C_5, C_6, C_7, C_8\}$

En este paso, la red de cláusulas consiste en las cláusulas que resultaron de hacer la resolución en el paso anterior, esto es, $\{C_3, C_4, C_5, C_6, C_7, C_8\}$, y la siguiente variable a usar será x_2. Ahora bien, $\bar{x}_2$ es ahora un literal puro y, por lo tanto, se hará la resolución respecto de él:

$$\text{Res}(G_1, \bar{x}_2) = \{C_4, C_5, C_6, C_7, C_8\}$$

y que resulta, simplemente, en la eliminación de la única cláusula que lo contenía, C_3.

En este paso, las cláusulas involucradas son:

$$G_1 \backslash \text{Res}(G_1, \bar{x}_2) = G_1 \backslash \{C_4, C_5, C_6, C_7, C_8\} = \{C_3\}$$

Paso 2 $G_2 = \{C_4, C_5, C_6, C_7, C_8\}$

La siguiente variable a usar es, en orden creciente del subíndice, x_3, que aparece afirmada en C_5 y negada en C_4 y C_7. Por lo tanto, hay que calcular la resolución respecto de dos pares de cláusulas: C_5 y C_4; por una parte, y C_5 y C_7 por la otra:

- Considerando el par de cláusulas C_5 y C_4, la disyunción de los literales que acompañan a x_3 afirmado y negado en cada cláusula respectivamente dan $(x_4 \vee \bar{x}_4)$, que es una tautología y, por lo tanto, no se produce.
- Considerando, en segundo lugar, el par de cláusulas C_5 y C_7, resulta la disyunción de literales $(\bar{x}_4 \vee \bar{x}_5)$, que se debe añadir como una cláusula nueva, C_9.

$$\text{Res}(G_2, x_3) = \{C_6, C_8, C_9 : (\bar{x}_4 \vee \bar{x}_5)\}$$

En este paso, las cláusulas involucradas son:

$$G_2 \backslash \text{Res}(G_2, x_3) = G_2 \backslash \{C_6, C_8, C_9\} = \{C_4, C_5, C_7\}$$

Paso 3 $G_3 = \{C_6, C_8, C_9\}$

En este paso, se usa el literal x_4, pero se observa que el literal $\bar{x}_4$ es puro, puesto que no aparece su negación, x_4 en la red de cláusulas actual. Por lo tanto, la resolución de G_3 respecto de $\bar{x}_4$ resulta simplemente en eliminar las cláusulas que lo contienen:

$$\text{Res}(G_3, \bar{x}_4) = \{C_8\}$$

de modo que las cláusulas empleadas en este paso son:

$$G_3 \backslash \text{Res}(G_3, \bar{x}_4) = G_3 \backslash \{C_8\} = \{C_6, C_9\}$$

Paso 4 $G_4 = \{C_8\}$

En el último paso, el siguiente literal a usar es x_5, que es puro en la red de cláusulas actual. Por lo tanto, la resolución respecto de él simplemente elimina la única cláusula que lo contiene, y como no hay más, resulta el vacío:

$$\text{Res}(G_4, x_5) = \varnothing$$

que, como se ha dicho antes, significa que la fórmula inicial es satisfacible y, por lo tanto, en la segunda fase de Davis-Putnam será preciso calcular un modelo que la satisfaga. Sin embargo, antes de continuar, es preciso calcular la única cláusula involucrada como sigue:

$$G_4 \backslash \text{Res}(G_4, x_5) = G_4 \backslash \varnothing = \{C_8\}$$

Puesto que se ha probado que la fórmula proposicional F_1' es satisfacible, a continuación se calcula un modelo, tal y como se pedía en la pregunta. Para ello, se muestran en orden inverso los literales empleados y las cláusulas involucradas en cada paso:

Paso	Variable	Cláusulas involucradas
4	x_5	$\{C_8\}$
3	$\bar{x}_4$	$\{C_6, C_9\}$
2	x_3	$\{C_4, C_5, C_7\}$
1	$\bar{x}_2$	$\{C_3\}$
0	x_1	$\{C_1, C_2\}$

Ahora es posible crear un modelo M que satisfaga la fórmula proposicional original, dando valor a la variable indicada en cada paso de modo que se satisfagan, por separado, las cláusulas de cada paso —y otras, que apareciesen en las mismas cláusulas, si fuera preciso:

a) Primero, se considera en el paso 4 la asignación de valor a la variable x_5 para que se satisfaga la cláusula C_8. Para ello, basta con hacer $x_5 = \top$.

b) En segundo lugar, es preciso asignar un valor a x_4. Puesto que el literal $\bar{x}_4$ es puro, entonces haciendo $x_4 = \bot$ se satisfacen automáticamente C_6 y C_9.

c) A continuación, es preciso satisfacer las cláusulas C_4, C_5 y C_7 asignando valor a x_3, y otras variables en esas cláusulas si fuera preciso. Ahora bien, como $x_4 = \bot$, C_5 ya está satisfecha. Como x_3 aparece sólo negada en C_4 y C_7, se hace $x_3 = \bot$.

d) Para satisfacer C_3 en el siguiente paso, se observa que como x_3 y x_4 tienen el valor falso ($\bot$), es preciso dar valor a x_2 (como se indica en la tabla superior) para satisfacer la cláusula. Puesto que x_2 es un literal puro en C_3, basta con hacer $x_2 = \bot$.

e) Por último, es preciso satisfacer C_1 y C_2. Como en el primer paso se hizo $x_5 = \top$, C_1 ya está satisfecha. Por otra parte, $x_5 = \top$ no sirve para satisfacer C_2. Además, como se decidió hacer $x_2 = \bot$ en el paso anterior, entonces es preciso darle un valor a x_1 para satisfacer C_2: $x_1 = \bot$.

con lo que resulta el siguiente modelo:

$$M = \{x_1 = \bot, x_2 = \bot, x_3 = \bot, x_4 = \bot, x_5 = \top\}$$

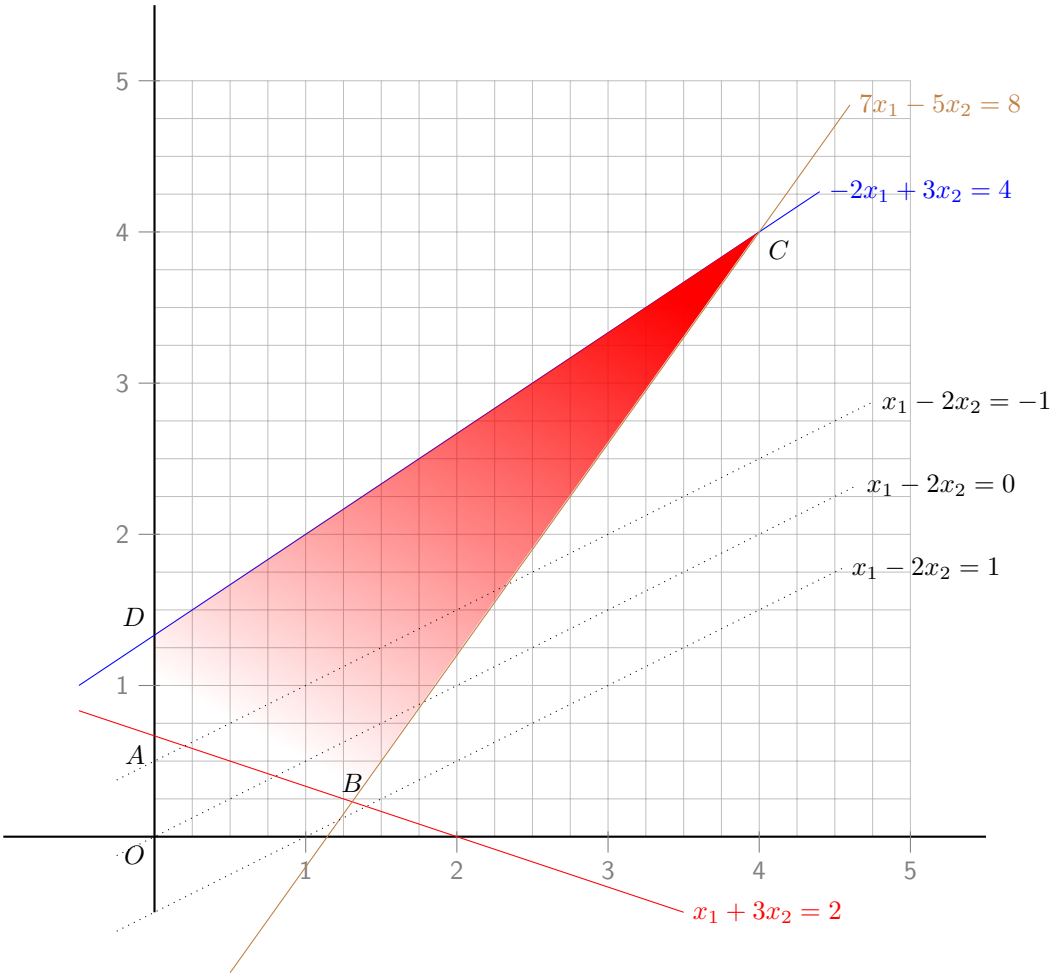
3. El siguiente árbol muestra la aplicación del algoritmo Davis-Putnam-Logemann-Loveland (DPLL) a la fórmula F_1' obtenida en el primer apartado —y que es igual a la original. Para mejorar la legibilidad, en la figura se han omitido las instanciaciones de variables unitarias en los dos últimos niveles, x_4 y x_5 que, como en el resto, se ordenan de la misma manera: el sucesor izquierdo se corresponde con la variable afirmada, y el derecho con la misma variable negada.

   Como puede verse, hay un único modelo, que es exactamente igual al mismo expuesto en el apartado anterior.

   El árbol de resolución anterior evidencia el efecto de la ordenación de sucesores. Como se ve, la única solución encontrada es precisamente el nodo más a la derecha de todos. Por lo tanto, si cada sucesor izquierdo hubiera sido la variable que correspondiese en cada nivel negada, y después afirmada, el mismo nodo se habría encontrado como el nodo más a la izquierda de todos, y por lo tanto, muchísimo antes.

!
Problema 4

1. La siguiente figura muestra la región factible (en rojo) que resulta de la intersección de las regiones factibles de cada restricción (cuyas fronteras están marcadas en rojo, azul y marrón).



   Los puntos A, B, C y D se calculan como la intersección de las rectas correspondientes y resultan, por lo tanto, de la resolución de sistemas de ecuación compatibles determinados de dos variables con dos ecuaciones (cuyo cálculo se omite aquí):

$(x_1 \vee x_4 \vee x_5) \wedge (\overline{x}_1 \vee \overline{x}_5 \vee x_2) \wedge (\overline{x}_2 \vee x_3 \vee x_4) \wedge (\overline{x}_3 \vee x_4) \wedge (x_3 \vee \overline{x}_4) \wedge (\overline{x}_4) \wedge (\overline{x}_3 \vee \overline{x}_5) \wedge (x_5)$
$x_1 = \top$
$x_1 = \bot$
$(\overline{x}_5 \vee x_2) \wedge (\overline{x}_2 \vee x_3 \vee x_4) \wedge (\overline{x}_3 \vee x_4) \wedge (x_3 \vee \overline{x}_4) \wedge (\overline{x}_4) \wedge (\overline{x}_3 \vee \overline{x}_5) \wedge (x_5)$
$(x_4 \vee x_5) \wedge (\overline{x}_2 \vee x_3 \vee x_4) \wedge (\overline{x}_3 \vee x_4) \wedge (x_3 \vee \overline{x}_4) \wedge (\overline{x}_4) \wedge (\overline{x}_3 \vee \overline{x}_5) \wedge (x_5)$
$x_2 = \top$
$x_2 = \bot$
$x_2 = \top$
$x_2 = \bot$
$(x_3 \vee x_4) \wedge (\overline{x}_3 \vee x_4) \wedge (x_3 \vee \overline{x}_4) \wedge (\overline{x}_3 \vee \overline{x}_5) \wedge (x_5)$
$(\overline{x}_5) \wedge (\overline{x}_3 \vee x_4) \wedge (x_3 \vee \overline{x}_4) \wedge (\overline{x}_3 \vee \overline{x}_5) \wedge (x_5)$
$(x_4 \vee x_5) \wedge (x_3 \vee x_4) \wedge (\overline{x}_3 \vee x_4) \wedge (x_3 \vee \overline{x}_4) \wedge (\overline{x}_4) \wedge (\overline{x}_3 \vee \overline{x}_5) \wedge (x_5)$
$(x_4 \vee x_5) \wedge (\overline{x}_3 \vee x_4) \wedge (x_3 \vee \overline{x}_4) \wedge (\overline{x}_4) \wedge (\overline{x}_3 \vee \overline{x}_5) \wedge (x_5)$
$x_3 = \top$
$x_3 = \bot$
$x_3 = \top$
$x_3 = \bot$
$x_3 = \top$
$x_3 = \bot$
$x_3 = \top$
$x_3 = \bot$
$(x_3 \vee x_4) \wedge (\overline{x}_3 \vee x_4) \wedge (x_3 \vee \overline{x}_4) \wedge (\overline{x}_4) \wedge (\overline{x}_3 \vee \overline{x}_5) \wedge (x_5)$
$(\overline{x}_5) \wedge (x_3 \vee x_4) \wedge (\overline{x}_3 \vee x_4) \wedge (x_3 \vee \overline{x}_4) \wedge (\overline{x}_4) \wedge (\overline{x}_3 \vee \overline{x}_5) \wedge (x_5)$
$(x_4 \vee x_5) \wedge (x_3 \vee x_4) \wedge (\overline{x}_3 \vee x_4) \wedge (x_3 \vee \overline{x}_4) \wedge (\overline{x}_4) \wedge (\overline{x}_3 \vee \overline{x}_5) \wedge (x_5)$
$(x_4 \vee x_5) \wedge (\overline{x}_3 \vee x_4) \wedge (x_3 \vee \overline{x}_4) \wedge (\overline{x}_4) \wedge (\overline{x}_3 \vee \overline{x}_5) \wedge (x_5)$
$(x_4)\wedge$ $(\overline{x}_4)\wedge$ $(\overline{x}_5)$
$(\overline{x}_5)\wedge$ $(x_4)\wedge$ $(\overline{x}_4)\wedge$ $(\overline{x}_5)\wedge$ (x_5)
$(x_4)\wedge$ $(\overline{x}_4)\wedge$ (x_5)
$(x_4)\wedge$ $(\overline{x}_4)\wedge$ (x_5)
$(x_4 \vee x_5)\wedge$ (x_4) $(\overline{x}_4)\wedge$ $(\overline{x}_5)$ (x_5)
$(x_4 \vee x_5)\wedge$ $(x_4)\wedge$ $(\overline{x}_4)\wedge$ (x_5)
$(x_4 \vee x_5)\wedge$ $(\overline{x}_4)$ $(\overline{x}_4)\wedge$ (x_5)
$(x_5)\wedge$ (x_5)
$\{\varnothing\}$ $\{\varnothing\}\{\varnothing\}$ $\{\varnothing\}$ $\{\varnothing\}\{\varnothing\}$ $\{\varnothing\}$ $\{\varnothing\}\{\varnothing\}$ $\{\varnothing\}$ $\{\varnothing\}\{\varnothing\}$ $x_4 = \bot$
$\{\varnothing\}$ $\{\varnothing\}$
$\varnothing$ $\{\varnothing\}$

$$A(0, \tfrac{2}{3})$$
$$B(\tfrac{17}{13}, \tfrac{3}{13})$$
$$C(4, 4)$$
$$D(0, \tfrac{4}{3})$$

Tal y como asegura el teorema fundamental del *simplex*, la solución óptima será un *punto extremo* de la región factible. Para ver cuál será, la misma figura muestra en negro varias curvas de isobeneficio (denominadas así porque la función objetivo es de maximización). Como se puede ver, según aumenta el valor de z, el último punto que se toca de la región factible es el punto B que será la solución buscada con un valor de la función objetivo z menor que 1.

Una forma alternativa de calcular la solución consiste en evaluar la función objetivo en cada punto extremo:

$$z(A) = c^T A = (1 \quad -2) \begin{pmatrix} 0 \\ \frac{2}{3} \end{pmatrix} = -\tfrac{4}{3}$$

$$z(B) = c^T B = (1 \quad -2) \begin{pmatrix} \frac{17}{13} \\ \frac{3}{13} \end{pmatrix} = \tfrac{11}{13}$$

$$z(C) = c^T C = (1 \quad -2) \begin{pmatrix} 4 \\ 4 \end{pmatrix} = -4$$

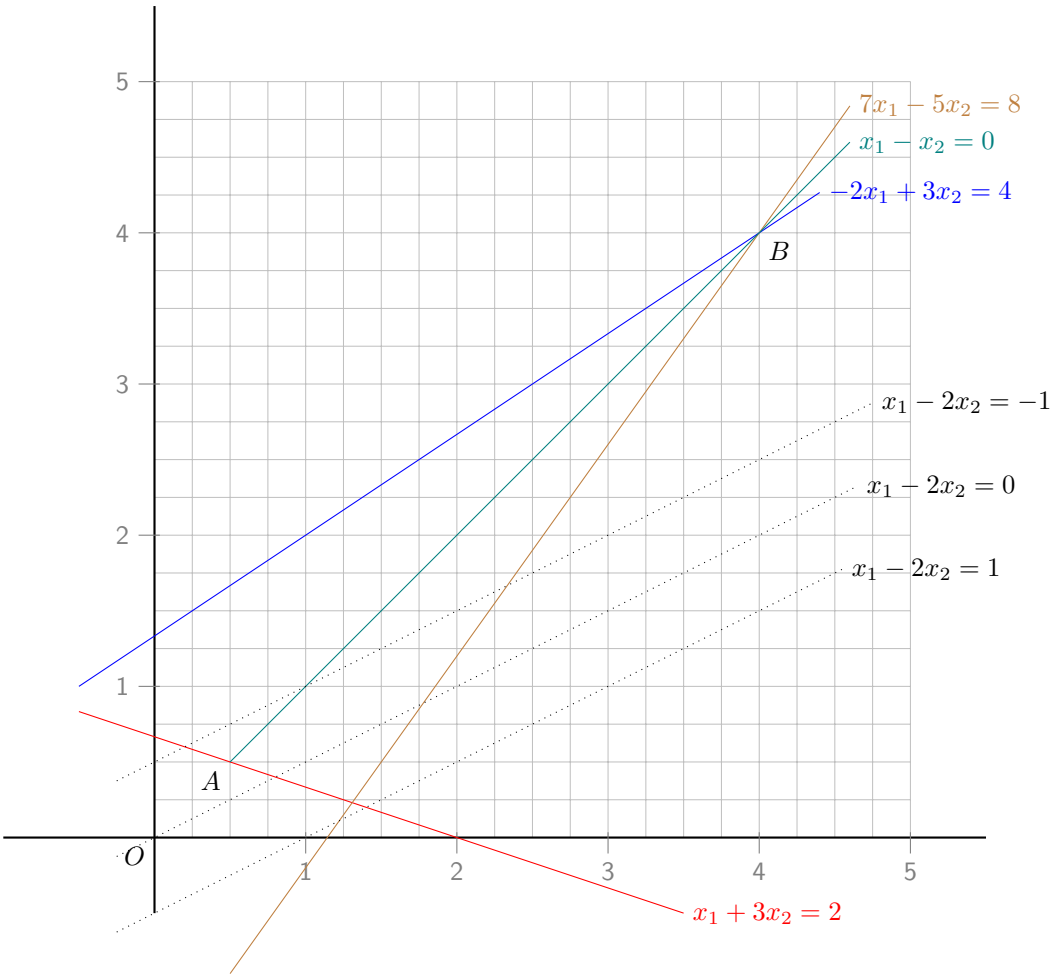
$$z(D) = c^T D = (1 \quad -2) \begin{pmatrix} 0 \\ \frac{4}{3} \end{pmatrix} = -\tfrac{8}{3}$$

Como puede verse, el punto extremo con el mayor valor de la función objetivo es el punto B, de modo que la solución óptima del problema planteado es:

$$x^* = \begin{pmatrix} \frac{17}{13} \\ \frac{3}{13} \end{pmatrix}$$

con un valor de la función objetivo $z^* = \frac{11}{13}$.

2. La siguiente figura muestra la región factible que resulta de sumar la restricción $x_1 = x_2$. Como puede verse, la nueva región no es otra cosa que la *intersección* de la que se calculó en el anterior apartado, con la recta $x_1 - x_2 = 0$. Por lo tanto, la región factible consiste únicamente en el segmento $\overline{AB}$.



Tal y como muestran las curvas de isocoste, la función objetivo será máxima en el punto A, que se calcula como la intersección de $x_1 + 3x_2 = 2$, y la nueva restricción $x_1 = x_2$:

$$\begin{array}{rcr} x_1 & + & 3x_2{=}2 \\ x_1 & - & x_2{=}0 \end{array}$$

y que se resuelve como el producto de la inversa de la matriz de coeficientes por el vector columna de términos independientes:

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

3. El tercer apartado pedía transformar el problema de programación lineal del enunciado en forma *estándar* de maximización.

   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, ya verifica todas estas condiciones salvo la primera. 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.

y, en cualquier caso, las variables de holgura se añaden a la función objetivo con coeficiente nulo.

Por lo tanto, el problema de Programación Lineal queda, como sigue, en forma estándar de maximización:

$$\begin{array}{c} \text{máx } z = x_1 - 2x_2 + 3x_3 \\ \begin{array}{rcrcrcrcrcrcl} x_1 & - & 2x_2 & + & 3x_3 & + & x_4 & & & & & = & 2 \\ - & 2x_1 & + & 3x_2 & + & 5x_3 & & - & x_5 & & & = & 4 \\ & 3x_1 & - & 5x_2 & - & 7x_3 & & & & + & x_6 & = & 8 \end{array} \\ \mathbf{x} \geq \mathbf{0} \end{array}$$

Ahora bien, el enunciado pedía, además, transformar el problema de modo que fuera posible iniciar la aplicación del algoritmo SIMPLEX con una base igual a la matriz identidad. Actualmente, los vectores columna $\mathbf{a_4}$ y $\mathbf{a_6}$ son vectores columna de la matriz identidad, pero falta aún un vector que tenga un 1 en la segunda posición. Para conseguirlo, simplemente se añade una *variable artificial* a la segunda restricción:

$$\begin{array}{c} \text{máx } z = x_1 - 2x_2 + 3x_3 - \infty x_7 \\ \begin{array}{rcrcrcrcrcrcl} x_1 & - & 2x_2 & + & 3x_3 & + & x_4 & & & & & = & 2 \\ - & 2x_1 & + & 3x_2 & + & 5x_3 & & - & x_5 & + & x_7 & = & 4 \\ & 3x_1 & - & 5x_2 & - & 7x_3 & & & & + & x_6 & = & 8 \end{array} \\ \mathbf{x} \geq \mathbf{0} \end{array}$$

que se añade, además, a la función objetivo con una penalización infinitamente alta.

Como puede verse, la tarea de programación lineal resultante: uno, está en forma estándar de maximización; segundo, los vectores columna $\mathbf{a_4}$, $\mathbf{a_6}$ y $\mathbf{a_7}$ ya forman una base que es igual a la matriz identidad.

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 tiene soluciones infinitas. Este caso se detecta cuando todos los costes reducidos son no negativos, y hay al menos uno que es nulo. Además, no tiene que haber variables artificiales en la base actual.
   - El problema es irresoluble. Esto ocurre cuando en el segundo paso, todos los costes reducidos son no negativos 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.

   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 $\{x_4, x_7, x_6\}$.
Nótese la importancia del orden en el que se indican las variables básicas, y es que ése es el único orden de los vectores columna $\mathbf{a_i}$ que pueden formar la matriz identidad.

$$B_0 = I_3 \qquad B_0^{-1} = I_3$$
$$x_0^* = B_0^{-1}b = b = \begin{pmatrix}2\\4\\8\end{pmatrix} \quad z_0^* = c_{B_0}^T x_0^* = \begin{pmatrix}0 & -\infty & 0\end{pmatrix}\begin{pmatrix}2\\4\\8\end{pmatrix} = -4\infty$$

b) Selección de la variable de entrada
En las expresiones siguientes el cálculo de los vectores y_i, $y_i = B_0^{-1}a_i$, se ha embebido en el cálculo de los *costes reducidos* directamente (aunque en una iteración con una base igual a la matriz identidad, $y_i = a_i$):

$$z_1 - c_1 = c_{B_0}^T B_0^{-1} a_1 - c_1 = \begin{pmatrix}0 & -\infty & 0\end{pmatrix} I_3 \begin{pmatrix}1\\-2\\3\end{pmatrix} - 1 = 2\infty - 1$$
$$z_2 - c_2 = c_{B_0}^T B_0^{-1} a_2 - c_2 = \begin{pmatrix}0 & -\infty & 0\end{pmatrix} I_3 \begin{pmatrix}-2\\3\\-5\end{pmatrix} + 2 = -3\infty + 2$$
$$z_3 - c_3 = c_{B_0}^T B_0^{-1} a_3 - c_3 = \begin{pmatrix}0 & -\infty & 0\end{pmatrix} I_3 \begin{pmatrix}3\\5\\-7\end{pmatrix} - 3 = -5\infty - 3$$
$$z_5 - c_5 = c_{B_0}^T B_0^{-1} a_5 - c_5 = \begin{pmatrix}0 & -\infty & 0\end{pmatrix} I_3 \begin{pmatrix}0\\-1\\0\end{pmatrix} - 0 = +\infty$$

En los cálculos anteriores, ∞ 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 es suficiente para penalizar las variables artificiales). Usando ∞ como un símbolo cualquiera es posible saber qué valores son más grandes sustituyéndolo por valores arbitrariamente grandes.
En este caso particular, la variable no básica con el valor más negativo es x_3, asi que ésta será la variable que entre en la siguiente iteración.

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} donde x_i es la variable elegida en el paso anterior (x_3) y el vector $\mathbf{y_i}$ será entonces el vector columna $\mathbf{a_3}$ porque, como se advirtió en el punto anterior, con una base igual a la matriz identidad $\mathbf{y_i} = \mathbf{a_i}$:

$$\text{mín}\left\{\frac{2}{3}, \frac{4}{5}, \frac{8}{\not{-7}}\right\}$$

donde el último cociente ha sido desestimado porque tiene un denominador negativo. El menor de los dos primeros cocientes es el que se corresponde con la primera de las variables básicas de esta iteración, x_4, que será, por tanto, la variable que abandonará la base.

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 $\{x_3, x_6, x_7\}$.
Nótese que ahora no se ordenan las variables básicas con ningún criterio específico, puesto que cualquier base, que resultara de cualquier ordenación, será igualmente válida.

$$B_1 = \begin{pmatrix} 3 & 0 & 0 \\ 5 & 0 & 1 \\ -7 & 1 & 0 \end{pmatrix} \qquad B_1^{-1} = \begin{pmatrix} \frac{1}{3} & 0 & 0 \\ \frac{7}{3} & 0 & 1 \\ -\frac{5}{3} & 1 & 0 \end{pmatrix}$$

$$x_1^* = B_1^{-1}b = b = \begin{pmatrix} \frac{2}{3} \\ \frac{38}{3} \\ \frac{2}{3} \end{pmatrix} \quad z_1^* = c_{B_1}^T x_1^* = \begin{pmatrix} 3 & 0 & -\infty \end{pmatrix} \begin{pmatrix} \frac{2}{3} \\ \frac{38}{3} \\ \frac{2}{3} \end{pmatrix} = 2 - \frac{2\infty}{3}$$

b) Selección de la variable de entrada
A continuación se muestra el cálculo de *costes reducidos* para todas las variables no básicas. En primer lugar, se muestra el cálculo de todos los vectores columna $\mathbf{y_j} = B_1^{-1}\mathbf{a_j}$:

$$y_1 = \begin{pmatrix} \frac{1}{3} \\ \frac{16}{3} \\ -\frac{11}{3} \end{pmatrix} \qquad y_2 = \begin{pmatrix} -\frac{2}{3} \\ -\frac{29}{3} \\ \frac{19}{3} \end{pmatrix}$$

$$y_4 = \begin{pmatrix} \frac{1}{3} \\ \frac{7}{3} \\ -\frac{5}{3} \end{pmatrix} \qquad y_5 = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}$$

y que se usan a continuación:

$$z_1 - c_1 = c_{B_1}^T y_1 - c_1 = \begin{pmatrix} 3 & 0 & -\infty \end{pmatrix} \begin{pmatrix} \frac{1}{3} \\ \frac{16}{3} \\ -\frac{11}{3} \end{pmatrix} - 1 = \frac{11}{3}\infty$$

$$z_2 - c_2 = c_{B_1}^T y_2 - c_2 = \begin{pmatrix} 3 & 0 & -\infty \end{pmatrix} \begin{pmatrix} -\frac{2}{3} \\ -\frac{29}{3} \\ \frac{19}{3} \end{pmatrix} + 2 = -\frac{19}{3}\infty$$

$$z_4 - c_4 = c_{B_1}^T y_4 - c_4 = \begin{pmatrix} 3 & 0 & -\infty \end{pmatrix} \begin{pmatrix} \frac{1}{3} \\ \frac{7}{3} \\ -\frac{5}{3} \end{pmatrix} - 0 = \frac{5}{3}\infty + 1$$

$$z_5 - c_5 = c_{B_1}^T y_5 - c_5 = \begin{pmatrix} 3 & 0 & -\infty \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} - 0 = \infty$$

La única variable con un coste reducido negativo es x_2 que será, por lo tanto, la variable elegida para entrar en la base en la siguiente iteración.
Aunque la variable x_4 no podía volver a entrar, porque había salido en la iteración anterior, se ha calculado su coste reducido para asegurarse de que efectivamente tiene un valor no negativo —para cualquier valor arbitrariamente grande de ∞.

c) Selección de la variable de salida
Nuevamene, la variable de salida se calcula en atención al mínimo cociente x_i/y_{ij} donde x_i es la variable elegida en el paso anterior para añadirse a la base (x_2) e y_{ij} son las componentes de su vector $\mathbf{y}$ también calculados en el paso anterior:

$$\text{mín}\left\{\cancel{\frac{\frac{2}{3}}{\frac{2}{3}}}, \cancel{\frac{\frac{38}{3}}{\frac{29}{3}}}, \frac{\frac{2}{3}}{\frac{19}{3}}\right\}$$

Como se ve, lo dos primeros cocientes tienen denominadores negativos y, por ese motivo, se desechan. Por lo tanto, la variable que sale es la última, x_7, precisamente la variable artificial.

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

a) Cálculo de las variables básicas
Las nuevas variables básicas son $\{x_2, x_3, x_6\}$

$$B_2 = \begin{pmatrix} -2 & 3 & 0 \\ 3 & 5 & 0 \\ -5 & -7 & 1 \end{pmatrix} \qquad B_2^{-1} = \begin{pmatrix} -\frac{5}{19} & \frac{3}{19} & 0 \\ \frac{3}{19} & \frac{2}{19} & 0 \\ -\frac{4}{19} & \frac{29}{19} & 1 \end{pmatrix}$$

$$x_2^* = B_2^{-1}b = b = \begin{pmatrix} \frac{2}{19} \\ \frac{14}{19} \\ \frac{260}{19} \end{pmatrix} \quad z_2^* = c_{B_2}^T x_2^* = \begin{pmatrix} -2 & 3 & 0 \end{pmatrix} \begin{pmatrix} \frac{2}{19} \\ \frac{14}{19} \\ \frac{260}{19} \end{pmatrix} = 2$$

b) Selección de la variable de entrada
A continuación se calculan todos los costes reducidos. En este paso, se obvia el cálculo del coste reducido de la variable artificial puesto que al restar su coeficiente en la función objetivo saldría un valor positivo arbitrariamente grande.
Primero, se calculan todos los vectores columna $\mathbf{a_j} = B_2^{-1}\mathbf{a_j}$:

$$y_1 = \begin{pmatrix} -\frac{11}{19} \\ -\frac{3}{19} \\ -\frac{5}{19} \end{pmatrix}$$

$$y_4 = \begin{pmatrix} -\frac{5}{19} \\ \frac{3}{19} \\ -\frac{4}{19} \end{pmatrix}$$

$$y_5 = \begin{pmatrix} -\frac{3}{19} \\ -\frac{2}{19} \\ -\frac{29}{19} \end{pmatrix}$$

y que se emplean en el cálculo de costes reducidos $(z_j - c_j)$ que se muestra a continuación:

$$z_1 - c_1 = c_{B_2}^T y_1 - c_1 = \begin{pmatrix} -2 & 3 & 0 \end{pmatrix} \begin{pmatrix} -\frac{11}{19} \\ -\frac{3}{19} \\ -\frac{5}{19} \end{pmatrix} - 1 = 0$$

$$z_4 - c_4 = c_{B_2}^T y_4 - c_4 = \begin{pmatrix} -2 & 3 & 0 \end{pmatrix} \begin{pmatrix} -\frac{5}{19} \\ \frac{3}{19} \\ -\frac{4}{19} \end{pmatrix} - 0 = 1$$

$$z_5 - c_5 = c_{B_2}^T y_5 - c_5 = \begin{pmatrix} -2 & 3 & 0 \end{pmatrix} \begin{pmatrix} -\frac{3}{19} \\ -\frac{2}{19} \\ -\frac{29}{19} \end{pmatrix} - 0 = 0$$

Como se ve, no hay ningún coste reducido negativo. Ahora bien, se observa que hay costes reducidos nulos, lo que indica que hay puntos adyacentes con el mismo valor de la función ob-

jetivo que el punto extremo calculado en esta iteración. Corresponde ahora calcular cualquier punto adyacente y, por lo tanto, se prosigue con la aplicación del SIMPLEX.
En este paso se considerará, simultaneamente, que podrían entrar tanto x_1 como x_5 —y el resultado no variará como se verá a continuación.

c) Selección de la variable de salida
Como de costumbre, se calcula el menor cociente x_i/y_{ij}, después de deshechar todos aquellos que tengan un denominador nulo o negativo.
Ahora bien, tanto x_1 como x_5 tienen todas las componentes de sus vectores $\mathbf{a_j}$ negativas, de modo que no es posible realizar este cálculo, y se trata, por lo tanto, de un problema *no acotado* que no tiene solución única.

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 no acotado (esto es, la función objetivo puede crecer indefinidamente), porque todas las componentes del vector columna a_j de las únicas variables básicas (x_1 y x_5) que podrían entrar en la base sin reducir el valor de la función objetivo, son negativas.
   - Por lo tanto, no existe una solución única.

   Interpretación de los recursos La importancia de los recursos puede estudiarse con la resolución del problema dual puesto que el valor óptimo de la variable dual i-ésima representa la contribución al crecimiento de la función objetivo por unidad de recurso i-ésimo. Esto es exactamente lo que se hará en el siguiente apartado.

6. Para la resolución de este apartado, basta con recordar la *interpretación económica* de las soluciones de un problema dual que advierte que:

   > La variable dual $x_i'^*$ indica la contribución por unidad del recurso i-ésimo b_i a la variación el en valor óptimo z^* actual del objetivo

   Puesto que el enunciado requiere la contribución unitaria de todos los recursos, entonces es preciso calcular la solución completa del problema dual.

   Para ello, es posible iniciar la aplicación de otro SIMPLEX al problema dual (en forma *simétrica*) del primal. Sin embargo, en su lugar es preferible hacer uso del siguiente resultado teórico:

   > Si el problema de programación lineal en forma simétrica tiene una solución óptima correspondiente a una base $\mathbf{B}$, entonces $\mathbf{x'}^T = \mathbf{c_B^T B^{-1}}$ es una solución óptima para el problema dual

   En este teorema, los términos usados para el cálculo de la solución óptima del problema dual se refieren al problema primal, salvo que se indique explícitamente lo contrario. Por lo tanto c_B^T es el vector de costes de las variables básicas en la solución del problema primal y B la base usada para el cálculo de la misma solución —que se mostró en el último paso de aplicación del SIMPLEX. Por el contrario, x'^T es la solución del problema dual.

   En particular:

$$x'^* = c_B^T B^{-1} = \begin{pmatrix} -2 & 3 & 0 \end{pmatrix} \begin{pmatrix} -\frac{5}{19} & \frac{3}{19} & 0 \\ \frac{3}{19} & \frac{2}{19} & 0 \\ -\frac{4}{19} & \frac{29}{19} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$$

El resultado debe leerse como que el incremento por unidad del primer recurso generará un crecimiento de la función objetivo igual a una unidad, mientras que los recursos de las últimas dos restricciones no harán crecer la función objetivo aunque se dispusiera de más de ellos.

Problema 5

Este problema se conoce como *Simple Machine Scheduling*, si bien en su versión formal existen penalizaciones tanto si el trabajo se acaba antes de una hora marcada (como se indica en el enunciado del problema), o si se acaba después de otra hora predefinida.

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 Un estado en este problema está constituido por una *permutación* de trabajos que ya han sido asignados a la cola de impresión a partir de una hora inicial, y un conjunto (por lo tanto, no ordenado) de tareas pendientes de ordenar en la cola de impresión.

   El concepto esencial que debe modelarse, por lo tanto, es `Trabajo`:

   Trabajo Un `trabajo` consiste en una petición de impresión que se envía al sistema. Cada una de ellas debe estar etiquetada con un identificador `id` que sirva para distinguirlo de otros. Además, tal y como advertía el enunciado, para cada uno de ellos se conocen su tiempo de ejecución, $\mathtt{t}_i$, y el primer instante de tiempo en el que deberían imprimirse, $\mathtt{e}_i$, si no se quiere incurrir en unan penalización, $\mathtt{c}_i$ que también se especifica con cada petición o `trabajo`.
   Pero además, como el sistema debe determinar el orden de ejecución de todos los trabajos, debe habilitarse un atributo `tiempo` que sirva para indicar el momento en el que se inicia su ejecución —de modo que se sabe que su conclusión ocurrirá exactamente en $\mathtt{tiempo}+\mathtt{t}_i$.

   Tal y como se advertía antes, cada estado debe mantener información actualizada de los trabajos que ya han sido asignados, y de aquellos que aún están pendientes de ordenarse en la cola de impresión. Para ello, se sugiere el uso de dos listas:

   Ordenados Esta lista contiene todos los trabajos que ya han sido asignados a la cola de impresión. Una invariante importante es que todos los trabajos en esta lista deben tener su atributo `tiempo` convenientemente instanciado.
   Inicialmente $\mathtt{Ordenados}=\varnothing$.

   Pendientes Contiene toda la lista de trabajos que están pendientes de ser ordenados por el sistema automático de optimización. También aquí hay una invariante importante, y es que para todos ellos, $\mathtt{tiempo}=-1$, o cualquier otro valor imposible que indique que aún no están asignados.
   Una vez que se hayan satisfecho todas las peticiones será cierto que $\mathtt{Pendientes}=\varnothing$.

   Precisamente, la gestión de estas listas se llevará a cabo con el operador que se detalla a continuación:

   Operadores En la definición de operadores es importante describir sus precondiciones/postcondiciones y, además, el coste que tienen. En este problema hay un único operador:

   Ordenar Este operador recibe un `trabajo`, y lo asigna a continuación del último. En lo sucesivo se asume que el primer trabajo se ejecuta en el instante de tiempo $\mathtt{tiempo}=0$ —y que todos los tiempos, de hecho, son relativos al inicio de la impresión de todos los trabajos.

   Precondiciones $\mathtt{trabajo}\in\mathtt{Pendientes}$ o, equivalentemente, que $\mathtt{trabajo.tiempo}=-1$.

Postcondiciones El operador debe borrar `trabajo` de la lista de trabajos `Pendientes`, y debe insertarlo, en su lugar, en la lista `Ordenados`. Al hacerlo, debe actualizar el instante en el que se inicia su impresión a 0 si no hay ningún trabajo en `Ordenados`, o a $\texttt{trabajo}'.\texttt{tiempo} + \texttt{trabajo}'.t_i$, donde $\texttt{trabajo}'$, el último trabajo que haya actualmente en `Ordenados`, en otro caso.

Coste El coste de este operador se debe calcular como sigue:

$$k = \begin{cases} 0 & \text{si } \texttt{trabajo.tiempo} + t_i \geq \texttt{trabajo}.e_i \\ \texttt{trabajo}.c_i & \text{en otro caso} \end{cases}$$

de donde se advierte que se trata de un problema de *costes arbitrarios*.

2. Formalmente hablando, el tamaño del espacio de estados se calcula como el número diferente de estados que pueden definirse. La definición del espacio de estados presentada en el apartado anterior sugiere crear estados incrementalmente, de modo que muchos de ellos contienen algunos trabajos ya ordenados en la cola de impresión, y otros no.

   Sin embargo, es fácil advertir que el número de soluciones diferentes son todas las permutaciones posibles de trabajos, N. De hecho, $N!$ domina el número de estados que no están completamente instanciados.

3. Tal y como se decía en el apartado anterior, el operador `Ordenar` definido en el primer apartado sugiere crear las soluciones incrementalmente. Por lo tanto, cualquier solución consistirá en un camino desde la raíz de un árbol de búsqueda (donde $\texttt{Ordenados} = \varnothing$), hasta otra donde $|\texttt{Ordenados}| = N$, el número de trabajos. Puesto que cada invocación del operador `Ordenar` sólo añade un trabajo (el que recibe como argumento) a la cola de impresión, entonces la profundidad será necesariamente N.

   Por otra parte, el factor de ramificación, b, se define como el número medio de sucesores. Inicialmente, el nodo raíz tendrá hasta N sucesores, puesto que puede decidir colocar en primer lugar a uno cualquiera de los trabajos recibidos. A profundidad $d = 1$, ya se habrá colocado un trabajo, de modo que cualquier nodo a esta profundidad tendrá hasta $(N - 1)$ sucesores. En general, cualquier nodo a profundidad d tendrá hasta $(N - d)$ sucesores.

   Puesto que el número de sucesores de todos los nodos a la misma profundidad es el mismo, basta con considerar cualquier solución (o camino de longitud N desde la raíz del árbol de búsqueda) para calcular el número medio de sucesores:

$$b = \frac{N + (N-1) + (N-2) + \cdots + 2 + 1}{N} = \frac{N\frac{(1+N)}{2}}{N} = \frac{1+N}{2}$$

   donde se ha dividido convenientemente por el número de nodos en cualquier camino, N, el número de sucesores generado en cada nodo que hay en el mismo camino.

4. Tal y como se señaló en el primer apartado, este es un problema de *costes arbitrarios*. Por lo tanto, se trata de resolver un problema de *minimización* en un grafo de estados donde los arcos tienen asignados costes diferentes. Los algoritmos estudiados con este propósito son fundamentalmente dos:

   Ramificación y acotación en profundidad (DFBnB) Tiene un coste de memoria lineal pero puede re-expandir muchos nodos en caso de que el dominio presente muchas *transposiciones* (como ocurre con todos los algoritmos de *el primero en profundidad*).

   Dijkstra Consiste en un algoritmo de *el mejor primero* donde la función de evaluación, $f(n)$ es, simplemente, el coste del camino desde el estado inicial hasta n, $g(n)$. Tiene un coste de memoria exponencial pero garantiza que cada vez que expande un nodo habrá encontrado la solución óptima hasta él puesto que los nodos se expanden en orden creciente de su valor de $f(n)$ —o, equivalentemente, de $g(n)$.

En este problema en particular, no es posible que haya transposiciones, puesto que es precisamente el orden en el que se ejecuta el operador `Ordenar` el que genera ordenaciones diferentes —y, por lo tanto, estados diferentes. Además, todas las soluciones están necesariamente a la misma profundidad como se vió en el tercer apartado y, por lo tanto, el algoritmo de *Ramificación y Acotación en Profundidad* garantiza que siempre encontrará una solución que sea susceptible de mejorarse (hasta encontrar una solución óptima) con un consumo de memoria lineal.

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.

   Por lo tanto, mientras que las instancias más sencillas se podrían resolver con el algoritmo A*, lo cierto es que el algoritmo IDA* es una elección mucho más razonable puesto que, como se indicó en el apartado anterior, no hay transposiciones en este dominio.