Departamento de Informática
Universidad Carlos III de Madrid

Grado en Informática

Heurística y Optimización

26 de Junio de 2018

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

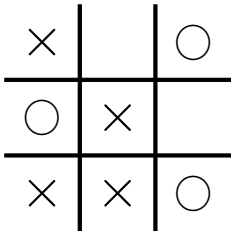
⑤ Escribe con claridad y en limpio, de forma ordenada y concisa

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 (1½ puntos)

Se pide emplear las técnicas de Programación Lineal para modelizar el cálculo de estrategias en el juego de las *tres en raya*, entre dos jugadores marcados como ◯ y ×. En particular, considérese exactamente la siguiente posición, en la que es el turno de ◯.



En todos los casos que siguen, se considera que una partida ganada puntúa con +1; si se pierde con 0 y, si se empata con $\frac{1}{2}$.

Se pide responder razonadamente las siguientes preguntas:

(a) **(½ puntos)** Diseñar una tarea de Programación Lineal cuyos puntos extremos factibles representen todas las combinaciones de jugadas legales de la posición indicada.

Es imprenscindible indicar claramente el significado de cada variable de decisión escogida, y el propósito de las restricciones.

(b) **(1 punto)** Diseñar una tarea de Programación Lineal que determine la jugada que debe hacer ◯ para maximizar su puntuación.

Es imprenscindible indicar claramente el significado de cada variable de decisión escogida, y el propósito de las restricciones y función objetivo diseñadas.

Pregunta 2 (2 puntos)

Considera la fórmula F_1 en Forma Normal Conjuntiva $F = \bigwedge_{i=1}^{6} C_i$ formada por las siguientes cláusulas:

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

Se pide responder razonadamente las siguientes preguntas:

(a) **(½ punto)** ¿La fórmula F_1 tiene literales puros? Si o no, y por qué.

(b) **(½ puntos)** Si los tuviera, podrían o bien eliminarse todas las cláusulas que los contengan o, simplemente, eliminar el literal puro preservando el resto de la cláusula. ¿Alguna de estas operaciones es incorrecta? Si o no y por qué.

(c) **(½ puntos)** Aplicar Davis-Putnam (DP) para encontrar un modelo que satisfaga la fórmula F_1 , en caso de que exista alguno.

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

(d) **(½ puntos)** Aplicar Davis–Putnam–Logemann–Loveland (DPLL) para encontrar, al menos, un modelo que satisfaga la fórmula F_1, en caso de que exista alguno.

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

Pregunta 3 (3 puntos)

Considerése la siguiente tarea problema de Programación Lineal:

$$\begin{array}{rcrcrcrcr} & & \multicolumn{7}{c}{\text{mín } z = -2x_1 - x_2 + 3x_3 - 5x_4} \\ & x_1 & + & 2x_2 & + & 2x_3 & + & 4x_4 & \leq & & 40 \\ - & 2x_1 & + & x_2 & - & x_3 & - & 2x_4 & \geq & - & 8 \\ & 4x_1 & - & 2x_2 & + & x_3 & - & x_4 & \leq & & 10 \\ & & & & & \multicolumn{5}{c}{\mathbf{x} \geq \mathbf{0}} & & \end{array}$$

Se pide responder razonadamente las siguientes preguntas:

(a) **(½ puntos)** Expresar el problema de Programación Lineal anterior en forma *estándar* de maximización.

(b) **(½ puntos)** Responde razonadamente la siguiente cuestión ¿es posible que este problema *no tenga solución*? Si o no y por qué.

(c) **(1 punto)** Resolver el problema de Programación Lineal obtenido en el primer apartado con el algoritmo SIMPLEX haciendo los preparativos que sean precisos para comenzar con la matriz identidad.

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

(d) **(½ puntos)** Interpretar el resultado y explicar qué conclusiones pueden extraerse de él.

(e) **(½ puntos)** Calcula la contribución por unidad de recurso del problema de Programación Lineal obtenido en el primer apartado al valor óptimo de la función objetivo.

Pregunta 4 (3½ puntos)

La Comunidad de Madrid está interesada en optimizar la gestión de sus embalses. Para ello, considérense todo el conjunto de embalses, para cada uno de los cuales se conoce el volumen de agua que contiene, y la capacidad máxima que puede acoger. Cada embalse da servicio a diferentes poblaciones y, si fuera preciso, es necesario mover agua de unos embalses a otros para garantizar la continuidad del servicio en todas las poblaciones de la Comunidad. Por lo tanto, un *requerimiento de servicio* consiste en una

especificación de volumen mínima de agua que debe llegar a un embalse específico. Con ello, el sistema debe determinar automáticamente los movimientos de agua necesarios desde cada embalse que satisfagan el requerimiento de servicio.

Se pide responder razonadamente la siguiente pregunta:

(a) **(½ puntos)** Considérese, idealmente, que cada embalse tiene una capacidad infinitamente alta y que, además, es posible mover agua desde cualquier embalse hasta cualquier otro.

¿Es posible resolver óptimamente este problema? En caso afirmativo, ¿cómo?.

Sin embargo, cada embalse sólo tiene compuertas para ceder una parte de su volumen a otros embalses, pero no a todos y, por supuesto, la capacidad de cada uno es finita.

Se pide responder razonadamente las siguientes preguntas:

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

(c) **(½ puntos)** Sabiendo que el coste del movimiento de aguas entre embalses es siempre el mismo, independientemente de la cantidad de agua, y los embalses origen y final, ¿qué algoritmo de búsqueda *no informada* sugerirías para minimizar el coste total para atender un determinado requerimiento de servicio?

(d) **(½ puntos)** Sugiere una función heurística $h(\cdot)$ que sea *admisible* e *informada* para el problema de minimizar el coste total del movimiento de aguas.

(e) **(½ puntos)** Suponiendo que se dispone de una función heurística $h(\cdot)$ admisible, sugiere un algoritmo de búsqueda *heurística* que sirva para resolver óptimamente el problema.

(f) **(½ puntos)** Considérese ahora que el coste de un movimiento de aguas entre dos embalses es exactamente igual al volumen de agua trasladado, independientemente de los embalses origen y final. ¿Sigue valiendo la función heurística obtenida en el apartado d) para la optimización del coste total? Si o no y por qué.

Por último, supóngase que la definición de requerimiento de servicio se extiende para considerar el volumen mínimo que debe llegar hasta una cantidad arbitraria de embalses, en vez de sólo uno.

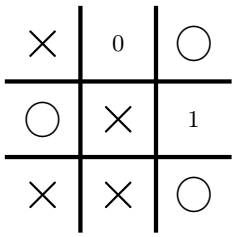
Se pide responder razonadamente la siguiente pregunta:

(g) **(½ puntos)** ¿Sigue valiendo la función heurística obtenida en el apartado d) para la optimización del coste total? Si o no y por qué.

Soluciones del examen de Heurística y Optimización
Junio 2018

Problema 1

1. En el primer apartado se pedía únicamente crear las restricciones que hicieran falta para crear una región factible cuyos puntos extremos representasen combinaciones de jugadas legales para ambos jugadores. En la posición dada en el enunciado sólo quedan dos posiciones por ocupar, numeradas como 0 y 1 en el diagrama siguiente:



Para representar las jugadas legales se emplearán dos variables de decisión, una por el movimiento que puede hacer cada jugador:

$$x_i = \begin{cases} 0, & \text{si el jugador } i \text{ juega en la posición 0} \\ 1, & \text{si el jugador } i \text{ juega en la posición 1} \end{cases}$$

Puesto que en la posición de la figura anterior es el turno de ◯, x_1 representará su jugada, y x_2, la de ×. Por ejemplo, si ◯ mueve en la posición que hay en medio de la fila superior, entonces $x_1 = 0$ y, necesariamente, $x_2 = 1$. Si, por el contrario, hubiera decidido hacer tres en raya moviendo en la posición libre que hay en la columna derecha, entonces $x_1 = 1$ y $x_2 = 0$.

Por lo tanto, son observaciones fundamentales las siguientes:

a) Las variables x_1 y x_2 son binarias, puesto que, como se ha visto, sólo pueden tomar los valores 0 y 1. Por lo tanto, $x_i \in \{0, 1\}$.

b) Si una variable toma el valor 0, la otra debe tomar el valor 1 y viceversa. Es decir, $x_1 + x_2 = 1$.

Por lo tanto, las restricciones quedan de la siguiente manera:

$$\begin{gathered} x_1 + x_2 = 1 \\ \mathbf{x} \in \{0, 1\} \end{gathered}$$

2. En el segundo apartado se pedía determinar la jugada del primer jugador (◯) que maximiza su puntuación. Por lo tanto, ahora sí que será preciso formular la función objetivo que servirá para encontrar la estrategia ganadora.

Resulta fácil observar que:

a) Si el primer jugador, ◯, juega en la primera fila ($x_1 = 0$) entonces, obligatoriamente el segundo jugador sólo podrá ocupar la posición de la columna derecha, $x_2 = 1$. La posición resultante serían tablas, y la puntuación de ◯ sería $\frac{1}{2}$.

b) Por el contrario, si ◯ ocupa la posición 1 ($x_1 = 1$), entonces hace tres en raya y ha ganado la partida, con una puntuación de $+1$.

Para representar la puntuación final del primer jugador, ◯, se emplea otra variable de decisión x_3 que, como se ha visto, sólo puede valer $\frac{1}{2}$ o $+1$. Además, de las observaciones anteriores resulta que el valor de x_1 determina inequívocamente el valor del resultado, x_3:

$$x_3 = \begin{cases} \frac{1}{2}, & \text{si } x_1 = 0 \\ 1, & \text{si } x_1 = 1 \end{cases}$$

y que se puede representar con la restricción lineal:

$$-x_1 \quad + \quad 2x_3 \quad = \quad 1$$

que debe sumarse a las restricciones del primer apartado, resultando entonces la tarea de Programación Lineal siguiente:

$$\begin{array}{rcccc} & & \max z = x_3 & & \\ x_1 & + & x_2 & = & 1 \\ - \; x_1 & & + \; 2x_3 & = & 1 \\ & x_1, x_2 \in \{0,1\}, x_3 \in \mathbb{R} & & & \end{array}$$

Nótese que:

a) Puesto que x_3 representa la puntuación obtenida por el primer jugador, la función objetivo simplemente maximiza su valor.

b) Además, x_3 no es una variable binaria (aunque sólo pueda tomar dos valores). Se trata de una variable de decisión ordinaria cuyo valor viene determinado por el de x_1.

Problema 2

1. Un literal ℓ es puro si y sólo si no aparece negado en ninguna parte de la fórmula F, esto es, si $\bar{\ell} \notin F$. 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.

2. Ćonsidérese una expresión de la lógica proposicional en Forma Normal Conjuntiva como la siguiente:

$$(x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_2) \wedge (x_1 \vee \bar{x}_3) \wedge (\bar{x}_2 \vee x_3)$$

En este caso, el literal $\ell = \bar{x}_2$ es puro, puesto que $\bar{\ell} = \bar{\bar{x}}_2 = x_2$ (por la regla de la doble negación), no aparece nunca en la fórmula anterior.

En este caso, se podría:

a) O bien eliminar todas las cláusulas que contengan el literal puro, $\bar{x}_2$, resultando entonces:

$$(x_1 \vee \bar{x}_3)$$

b) O bien eliminar únicamente el literal puro $\bar{x}_2$ preservando el resto de la cláusula quedando:

$$(x_1) \wedge (\bar{x}_1) \wedge (x_1 \vee \bar{x}_3) \wedge (x_3)$$

Resulta trivial observar que, mientras que la primera fórmula es satisfacible, la segunda resulta en una contradicción y, por lo tanto, hacer una cosa o la otra no preserva la satisfabilidad lógica de la fórmula original.

La explicacion es sencilla:

a) Si un literal es puro, basta con darle el valor de su signo ($\perp$ en el caso del ejemplo puesto que el literal ℓ consiste en una variable *negada*) para hacer cierta cada cláusula en la que aparezca, puesto que las cláusulas se definen precisamente como disyunciones.

Haciendo esta operación (en el ejemplo anterior, $x_2 = \perp$) resulta entonces la primera fórmula: $(x_1 \vee \bar{x}_3)$ que puede satisfacerse muy fácilmente haciendo que $x_1 = \top$ o $x_3 = \perp$.

b) Sin embargo, si un literal es puro, y se le da el valor contrario a su signo (esto es, $x_2 = \top$ sería la asignación contraria a su signo puesto que el literal puro consiste en su negación), entonces la fórmula se simplifica eliminando las ocurrencias del literal puro: $(x_1) \wedge (\bar{x}_1) \wedge (x_1 \vee \bar{x}_3) \wedge (x_3)$.

El motivo es que como cada cláusula es una disyunción, y el literal puro no se satisface precisamente por haberle dado el valor contrario a su signo, la única forma de que se satisfagan las cláusulas implicadas es satisfaciendo alguno del resto de literales que aparezcan en ella.

Pero, como pone de manifiesto el ejemplo mostrado, no hay ninguna garantía de que eso vaya a ser así.

Por lo tanto, la operación correcta consiste en eliminar completamente las cláusulas en las que aparece el literal puro. Análogamente, eliminar únicamente el literal puro preservando el resto de la cláusula es una operación incorrecta.

3. 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: $G_i \backslash \mathrm{Res}(G_i, x_j)$.

   Paso 0 $G_0 = \{C_1, C_2, C_3, C_4, C_5, C_6\}$

   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: $(\bar{x}_2 \vee x_3 \vee x_3 \vee \bar{x}_4)$ que, por la regla de idempotencia es lógicamente equivalente a $(\bar{x}_2 \vee x_3 \vee \bar{x}_4)$, de modo que:

   $$\mathrm{Res}(G_0, x_1) = \{C_3, C_4, C_5, C_6, C_7 : (\bar{x}_2 \vee x_3 \vee \bar{x}_4)\}$$

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

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

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

   A continuación, debe aplicarse la resolución respecto de x_2, que aparece afirmada en C_3 y C_4, y negada en C_7. Por lo tanto, la resolución producirá (tentativamente) dos cláusulas nuevas, las que resultan de las dos combinaciones de literal afirmado y negado: C_3 con C_7 y C_4 con C_7.

   La resolución entre C_3 y C_7 produce: $(\bar{x}_3 \vee x_5 \vee x_3 \vee \bar{x}_4))$, que es una tautología, puesto que contiene la disyunción del literal x_3 y su negado. Por lo tanto, esta cláusula no se añade a las resultantes de la aplicación del método de resolución.

   La resolución entre C_4 y C_7 produce: $(x_5 \vee x_3 \vee \bar{x}_4)$ que, al no ser una tautología, se añade como resultado de la resolución.

   Por lo tanto:

Paso	Variable	Cláusulas
3	x_4	$C_6 : (x_4 \vee \bar{x}_5)$
2	x_3	$C_5 : (\bar{x}_3 \vee x_4), C_8 : (x_3 \vee \bar{x}_4 \vee x_5)$
1	x_2	$C_3 : (x_2 \vee \bar{x}_3 \vee x_5), C_4 : (x_2 \vee x_5), C_7 : (\bar{x}_2 \vee x_3 \vee \bar{x}_4)$
0	x_1	$C_1 : (x_1 \vee \bar{x}_2 \vee x_3), C_2 : (\bar{x}_1 \vee x_3 \vee \bar{x}_4)$

Cuadro 1: Fase II del algoritmo de Davis-Putnam

$$\text{Res}(G_1, x_2) = \{C_5, C_6, C_8 : (x_3 \vee \bar{x}_4 \vee x_5)\}$$

de modo que:

$$G_1 \backslash \text{Res}(G_1, x_2) = G_1 \backslash \{C_5, C_6, C_8\} = \{C_3, C_4, C_7\}$$

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

La siguiente variable a considerar es x_3, que aparece afirmada en C_8, y negada en C_5, de modo que sólo hay que considerar un par de cláusulas resultando la cláusula nueva: $(\bar{x}_4 \vee x_5 \vee x_4)$ que, como ya ocurriera antes, es una tautología (puesto que el literal x_4 aparece afirmado y negado) y, por lo tanto, no se añade al resultado de la aplicación del método de resolución:

$$\text{Res}(G_2, x_3) = \{C_6\}$$

de donde resulta que:

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

Paso 3 $G_3 = \{C_6\}$

En este paso, los literales x_4 y $\bar{x}_5$ son puros, y el mismo resultado se obtendría con la resolución respecto de cualquiera de ellos. Se hará la resolución respecto de x_4 porque es el que corresponde en este paso. Como x_4 es un literal puro, simplemente se eliminan las cláusulas en las que aparece, dejando vacía la red de cláusulas de este paso:

$$\text{Res}(G_3, x_4) = \varnothing$$

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:

$$G_3 \backslash \text{Res}(G_3, x_4) = G_3 \backslash \varnothing = \{C_6\}$$

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.

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.

En primer lugar, la cláusula C_6 puede satisfacerse simplemente haciendo $x_4 = \top$.

Con ello, la cláusula C_5 se valida automáticamente en el siguiente paso, y ahora sólo es necesario validar la cláusula C_8 dando valor a x_3. Para ello, $x_3 = \top$.

Como $x_3 = \top$, entonces C_7 se valida inmediatamente en el tercer paso hacia atrás. Resta únicamente validar C_3 y C_4 dándole un valor a x_2. Puesto que ambas cláusulas contienen esta variable afirmada, basta con hacer $x_2 = \top$.

También porque $x_3 = \top$, las cláusulas C_1 y C_2 son ciertas, puesto que ambas contienen el literal x_3 sin negar.

En conclusión, componiendo los valores asignados a estas variables, resulta el siguiente modelo:

$$M = \{x_2 = \top, x_3 = \top, x_4 = \top\}$$

que, debe advertirse, no es el único cómo se podrá ver en el siguiente apartado. Es importante señalar que, con el modelo propuesto, el resto de variables que aparecen en la fórmula original, F, dada en el enunciado del problema (esto es x_1 y x_5) pueden tomar cualquier valor.

4. 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 3

Este problema es el mismo que se puso en la convocatoria de Junio de 2014, y que resulta de una pequeña variación del problema 2 de la serie de problemas 3.3a (página 82) de Hamdy A. Taha. Investigación de Operaciones. Sexta Edición, Prentice-Hall. 1998.

1. En el primer apartado se pedía expresar el problema de Programación Lineal del enunciado en *forma estándar* (de maximización), precisamente en preparación a su resolución con el algoritmo SIMPLEX.

   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, no verifica estas condiciones porque: uno, la función objetivo es de minimización; dos, la segunda restricción tiene un recurso negativo; por último, ninguna restricción es de la forma de igualdad.

   La siguiente formulación resuelve los dos primeros problemas:

   - Para cambiar el sentido de la función objetivo basta con multiplicarla por -1
   - Asimismo, para cambiar el signo del segundo miembro de alguna de las restricciones basta con multiplicarla por -1, cambiando entonces el signo de la desigualdad, si la hubiera.

   Por lo tanto, el problema de Programación Lineal queda de momento como sigue:

$$\begin{array}{c} \max z = 2x_1 + x_2 - 3x_3 + 5x_4 \\ \begin{array}{rcrcrcrcr} x_1 & + & 2x_2 & + & 2x_3 & + & 4x_4 & \leq & 40 \\ 2x_1 & - & x_2 & + & x_3 & + & 2x_4 & \leq & 8 \\ 4x_1 & - & 2x_2 & + & x_3 & - & x_4 & \leq & 10 \end{array} \\ \mathbf{x} \geq \mathbf{0} \end{array}$$

   Para resolver la segunda dificultad, conviene recordar aquí lo siguiente:

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

- 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.

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

$$\begin{array}{c} \max z = 2x_1 + x_2 - 3x_3 + 5x_4 \\ \begin{array}{rcrcrcrcrcrcrcl} x_1 & + & 2x_2 & + & 2x_3 & + & 4x_4 & + & x_5 & & & & & = & 40 \\ 2x_1 & - & x_2 & + & x_3 & + & 2x_4 & & & + & x_6 & & & = & 8 \\ 4x_1 & - & 2x_2 & + & x_3 & - & x_4 & & & & & + & x_7 & = & 10 \end{array} \\ \mathbf{x} \geq \mathbf{0} \end{array}$$

donde x_5, x_6 y x_7 son variables de holgura usadas para convertir las desigualdades en igualdades.

2. 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.

   En el caso de este problema en particular es imposible que el problema no sea resoluble puesto que no hay variables artificiales. En otras palabras, la ausencia de variables artificiales garantiza la solubilidad del problema original.

   De hecho, una solución factible consiste en hacer nulas (*no básicas*) todas las variables $x_i, 1 \leq i \leq 4$, y asignar a cada variable de holgura $x_i, 5 \leq i \leq 7$, el recurso de la restricción donde aparecen. Esto equivale a iniciar el algoritmo SIMPLEX con una base igual a la matriz identidad y así será, precisamente, como se iniciará el algoritmo en el siguiente apartado.

3. A continuación se aplica el algoritmo SIMPLEX siguiendo los pasos indicados anteriormente:

   Paso 0 Cálculo de una solución factible inicial

   a) Cálculo de las variables básicas
   Tal y como se ha indicado anteriormente (y como se requería explícitamente en el enunciado), la primera iteración se inicia con una base igual a la matriz identidad de dimensión 3 puesto que hay hasta tres restricciones. Por lo tanto, son variables básicas en el primer paso $\{x_5, x_6, x_7\}$.

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

$$x_0^* = B_0^{-1} b = b = \begin{pmatrix} 40 \\ 8 \\ 10 \end{pmatrix} \qquad z_0^* = c_{B_0}^T x_0^* = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 40 \\ 8 \\ 10 \end{pmatrix} = 0$$

b) Selección de la variable de entrada
Como en la primera iteración $y_i = a_i$, se omite el cálculo de los vectores columna y_i. El cálculo de los costes reducidos es, entonces:

$$z_1 - c_1 = c_{B_0} a_1 - c_1 = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix} - 2 = -2$$

$$z_2 - c_2 = c_{B_0} a_2 - c_2 = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 2 \\ -1 \\ -2 \end{pmatrix} - 1 = -1$$

$$z_3 - c_3 = c_{B_0} a_3 - c_3 = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + 3 = +3$$

$$z_4 - c_4 = c_{B_0} a_4 - c_4 = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 4 \\ 2 \\ -1 \end{pmatrix} - 5 = -5$$

Por lo tanto, entra la variable con el valor más negativo (puesto que es la que provocará el mayor crecimiento neto de la función objetivo), x_4.

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 para añadirse a la base y $0 \leq j < 3$ puesto que la base tiene dimensión 3:

$$\text{mín} \left\{ \frac{40}{4}, \frac{8}{2}, \frac{10}{\not -1} \right\}$$

y sale la variable x_6 que es la que se corresponde con la segunda fracción, precisamente la que tiene el menor valor. Nótese que se ha eliminado el tercer cociente, puesto que tiene un denominador negativo.

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

a) Cálculo de las variables básicas
Puesto que en el paso anterior se eligió la variable x_6 para abandonar la base y, en su lugar, se decidió introducir x_4, la nueva base estará formada por las variables básicas $\{x_4, x_5, x_7\}$.

$$B_1 = \begin{pmatrix} 4 & 1 & 0 \\ 2 & 0 & 0 \\ -1 & 0 & 1 \end{pmatrix} \qquad B_1^{-1} = \begin{pmatrix} 0 & \frac{1}{2} & 0 \\ 1 & -2 & 0 \\ 0 & \frac{1}{2} & 1 \end{pmatrix}$$

$$x_1^* = B_1^{-1} b = b = \begin{pmatrix} 4 \\ 24 \\ 14 \end{pmatrix} \quad z_1^* = c_{B_1}^T x_1^* = \begin{pmatrix} 5 & 0 & 0 \end{pmatrix} \begin{pmatrix} 4 \\ 24 \\ 14 \end{pmatrix} = 20$$

y, efectivamente, el valor de la función objetivo ha ascendido como corresponde en un problema de maximización.

b) Selección de la variable de entrada
Como la base actual ya no es la matriz identidad, se procede primero al cálculo de los vectores y_i de todas las variables no básicas:

$$y_1 = B_1^{-1}a_1 = \begin{pmatrix} 1 \\ -3 \\ 5 \end{pmatrix} \quad y_2 = B_1^{-1}a_2 = \begin{pmatrix} -\frac{1}{2} \\ 4 \\ -\frac{5}{2} \end{pmatrix}$$
$$y_3 = B_1^{-1}a_3 = \begin{pmatrix} \frac{1}{2} \\ 0 \\ \frac{3}{2} \end{pmatrix} \quad y_6 = B_1^{-1}a_6 = \begin{pmatrix} \frac{1}{2} \\ -2 \\ \frac{1}{2} \end{pmatrix}$$

A continuación se muestra el cálculo de los costes reducidos:

$$z_1 - c_1 = c_{B_1}y_1 - c_1 = \begin{pmatrix} 5 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ -3 \\ 5 \end{pmatrix} - 2 = +3$$
$$z_2 - c_2 = c_{B_1}y_2 - c_2 = \begin{pmatrix} 5 & 0 & 0 \end{pmatrix} \begin{pmatrix} -\frac{1}{2} \\ 4 \\ -\frac{5}{2} \end{pmatrix} - 1 = -\frac{7}{2}$$
$$z_3 - c_3 = c_{B_1}y_3 - c_3 = \begin{pmatrix} 5 & 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{2} \\ 0 \\ \frac{3}{2} \end{pmatrix} + 3 \quad = \frac{11}{2}$$
$$z_6 - c_6 = c_{B_1}y_6 - c_6 = \begin{pmatrix} 5 & 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{2} \\ -2 \\ \frac{1}{2} \end{pmatrix} - 0 = \frac{5}{2}$$

Por lo tanto, la variable x_2 entrará en la siguiente base.

c) Selección de la variable de salida
A continuación se calcula la variable que será reemplazada por x_2. Para ello, se calcula el mínimo de los cocientes x_1^*/y_2:

$$\text{mín}\left\{\frac{4}{-\frac{1}{2}}, \frac{24}{4}, \frac{14}{-\frac{5}{2}}\right\}$$

de modo que sale la segunda variable de la base, x_5 que, de hecho, es la única que no se deshecha por tener un denominador negativo.

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

a) Cálculo de las variables básicas
En el tercer paso del algoritmo SIMPLEX la base B_2 está formada por las variables $\{x_2, x_4, x_7\}$ de modo que:

$$B_2 = \begin{pmatrix} 2 & 4 & 0 \\ -1 & 2 & 0 \\ -2 & -1 & 1 \end{pmatrix} \qquad B_2^{-1} = \begin{pmatrix} \frac{1}{4} & -\frac{1}{2} & 0 \\ \frac{1}{8} & \frac{1}{4} & 0 \\ \frac{5}{8} & -\frac{3}{4} & 1 \end{pmatrix}$$
$$x_2^* = B_2^{-1}b = b = \begin{pmatrix} 6 \\ 7 \\ 29 \end{pmatrix} \quad z_2^* = c_{B_2}^T x_2^* = \begin{pmatrix} 1 & 5 & 0 \end{pmatrix} \begin{pmatrix} 6 \\ 7 \\ 29 \end{pmatrix} = 41$$

y, efectivamente, el valor de la función objetivo vuelve a ascender nuevamente.

b) Selección de la variable de entrada
Nuevamente, se procede primero al cálculo de los vectores y_i para todas las variables no básicas (es decir, aquellas susceptibles de formar parte de la nueva base, si la hubiera):

$$y_1 = B_2^{-1}a_1 = \begin{pmatrix} -\frac{3}{4} \\ \frac{5}{8} \\ \frac{25}{8} \end{pmatrix} \quad y_3 = B_2^{-1}a_3 = \begin{pmatrix} 0 \\ \frac{1}{2} \\ \frac{3}{2} \end{pmatrix}$$
$$y_5 = B_2^{-1}a_5 = \begin{pmatrix} \frac{1}{4} \\ \frac{1}{8} \\ \frac{5}{8} \end{pmatrix} \quad y_6 = B_2^{-1}a_6 = \begin{pmatrix} -\frac{1}{2} \\ \frac{1}{4} \\ -\frac{3}{4} \end{pmatrix}$$

Y ahora ya es posible calcular los costes reducidos de todas las variables no básicas como se indica a continuación:

$$z_1 - c_1 = c_{B_2}y_1 - c_1 = \begin{pmatrix} 1 & 5 & 0 \end{pmatrix} \begin{pmatrix} -\frac{3}{4} \\ \frac{5}{8} \\ \frac{25}{8} \end{pmatrix} - 2 = \frac{3}{8}$$
$$z_3 - c_3 = c_{B_2}y_3 - c_3 = \begin{pmatrix} 1 & 5 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ \frac{1}{2} \\ \frac{3}{2} \end{pmatrix} + 3 \quad = \frac{11}{2}$$
$$z_5 - c_5 = c_{B_2}y_5 - c_5 = \begin{pmatrix} 1 & 5 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{4} \\ \frac{1}{8} \\ \frac{5}{8} \end{pmatrix} - 0 \quad = \frac{7}{8}$$
$$z_6 - c_6 = c_{B_2}y_6 - c_6 = \begin{pmatrix} 1 & 5 & 0 \end{pmatrix} \begin{pmatrix} -\frac{1}{2} \\ \frac{1}{4} \\ -\frac{3}{4} \end{pmatrix} - 0 = \frac{3}{4}$$

De modo que no entra ninguna variable puesto que todos los costes reducidos son no nulos.

Por lo tanto, el algoritmo SIMPLEX ha concluido con la solución:

$$x^* = \begin{pmatrix} 0 \\ 6 \\ 0 \\ 7 \\ 0 \\ 0 \\ 29 \end{pmatrix}$$

para el que la función objetivo tiene un valor $z^* = 41$

4. 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 (como ya se había anticipado en la respuesta del segundo apartado) puesto que la solución no contiene valores positivos para alguna variable artificial. En particular, de hecho ni siquiera hay variables artificiales en el problema.
   - La solución es única porque los costes reducidos son todos estrictamente positivos. Eso significa que cualquier cambio en la base implicaría un decremento neto en el valor de la función objetivo mientras que el caso de *soluciones múltiples* ocurre cuando una cantidad infinita de soluciones tiene el mismo valor de la función objetivo.

- El valor de la función objetivo está acotado porque siempre se pudo aplicar la regla de salida con, al menos, alguna variable básica.

Interpretación de los recursos De los recursos se advierte:

- En la solución óptima sobran hasta 29 unidades del tercer recurso como lo atestigua el valor *óptimo* de la variable de holgura x_7. Es decir, se trata de un *recurso sobrante*.
- Por último, es muy conveniente estudiar la contribución por cada unidad de recurso al crecimiento de la función objetivo. Esto, en particular, se hace con el análisis del problema dual en el siguiente apartado.

5. Para la resolución del último apartado, basta con recordar la *interpretaciń 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. 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:

$$x'^* = c_B^T B^{-1} = (1 \quad 5 \quad 0) \begin{pmatrix} \frac{1}{4} & -\frac{1}{2} & 0 \\ \frac{1}{8} & \frac{1}{4} & 0 \\ \frac{5}{8} & -\frac{3}{4} & 1 \end{pmatrix} = \begin{pmatrix} \frac{7}{8} & \frac{3}{4} & 0 \end{pmatrix}$$

de donde resulta que la contribución del primer recurso al crecimiento de la función objetivo es $\frac{7}{8}$, mientras que el del segundo recurso es de $\frac{3}{4}$ por unidad de recurso. No es una casualidad que la contribución del tercer recurso al crecimiento de la función objetivo sea nulo puesto que, como ya se advirtió en el apartado anterior, se trata de un recurso sobrante.

Problema 4

1. ¡Por supuesto que es posible resolver óptimamente el problema propuesto!

   De hecho, el problema de este apartado resulta de *relajar* algunas restricciones del problema:

   a) Que cada embalse pueda estar conectado a cualquier otro, en vez de sólo a un subconjunto.

   b) Que los embalses puedan acoger una cantidad infinita de agua, en vez de una cantidad finita como ocurriría en la realidad.

De hecho, no es siquiera preciso considerar la segunda relajación para poder resolver óptimamente el problema. Considerando únicamente que es posible pasar agua desde cualquier embalse a cualquier otro, entonces para conseguir que llegue un determinado volumen a un embalse de destino, basta con seguir el siguiente procedimiento:

a) Ordenar todos los embalses de mayor a menor volumen de agua contenida

b) Trasvasar todo el agua desde el primer embalse (el que mayor capacidad tenga) al de destino.

c) Si el embalse de destino consigue así el volumen requerido, parar. En otro caso, volver al paso anterior considerando el siguiente embalse en la ordenación calculada en el primer paso.

En el problema se consideraban después varias funciones de coste:

a) En la primera (propuesta en el tercer apartado), el coste de trasvasar agua de un embalse a otro es siempre el mismo, independientemente del volumen y los embalses de origen y fin.

En este caso el coste de las operaciones necesarias para que un embalse de destino tenga un determinado volumen será el número diferente de embalses origen que se consideran. Precisamente, el procedimiento propuesto anteriormente minimiza el número de embalses a considerar por ordenarlos de mayor a menor volumen albergado.

b) En la segunda (propuesta en el sexto apartado), el coste total es igual al volumen trasvasado entre diferentes embalses.

En la relajación considerada en este apartado, el volumen a trasvasar es siempre el mismo, e igual a la diferencia entre el volumen objetivo del embalse de destino y el que tiene actualmente. Por lo tanto, el procedimiento anterior también resuelve óptimamente el mismo problema.

Por último, conviene advertir que en este problema relajado también pueden identificarse aquellos casos que son irresolubles: si la diferencia entre el volumen objetivo del embalse destino y el que alberga actualmente es mayor que la suma de los volúmenes albergados en el resto de embalses, entonces es imposible satisfacer el requerimiento de servicio.

2. 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 un conjunto de *embalses*, para los cuales se conoce su capacidad, el volumen de agua que albergan y aquellos a los que puede trasvasar agua.

Por lo tanto, cada estado vendrá determinado por la información de todos los embalses considerados en el problema, cada uno de los cuales se identifica como sigue:

Embalse Cada `embalse` se distingue por un identificador, `id`, que podría ser su nombre, por ejemplo. Además, para cada uno se conoce su `capacidad` máxima, y el `volumen` de agua que alberga en cada estado. Por último, para poder calcular las operaciones legales de trasvasar agua entre estados, cada `embalse` debe tener un último atributo `adyacentes` con referencias a todos aquellos embalses a los que puede trasvasar, efectivamente, parte de su volumen.

Además, el algoritmo que se diseñe para optimizar la gestión de todos los embalses, debe considerar el *requerimiento de servicio*:

Servicio Para el que sólo es preciso almacenar el identificador, `id`, del embalse objetivo, y el `volumen` de agua que se requiere para él.

De esta manera, un estado inicial consistirá en varias instancias de `embalse` y, una de `servicio` para la que el `volumen` que se requiere es estrictamente mayor que el que tiene actualmente. Análogamente, en un estado meta, el `volumen` que se requiere para el embalse destino es menor o igual que el que contiene.

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: `trasvasar` cuyos argumentos son el embalse `origen`, el embalse `destino`, y el `volumen`.

Precondiciones Que la cantidad que se desea trasvasar desde el embalse `origen` al `destino` no exceda el volumen actual del primero, ni la capacidad del segundo: $\texttt{origen.volumen} \leq \texttt{volumen} \wedge \texttt{destino.volumen} + \texttt{volumen} \leq \texttt{destino.capacidad}$.

Postcondiciones La ejecución del operador, efectivamente decrementa el volumen en el embalse `origen`, y lo aumenta en la misma cantidad, en el `destino`: $\texttt{origen.volumen} -= \texttt{volumen}$, y $\texttt{destino.volumen} += \texttt{volumen}$.

Coste Si se considera, como se indica en el tercer apartado, que el coste *es siempre el mismo* (independientemente del embalse origen, destino, y la cantidad trasvasada) entonces el coste es sencillamente $k = 1$.

Si, por el contrario, se considera el coste indicado en el sexto apartado, que es el volumen trasvasado, entonces $k = \texttt{volumen}$.

3. Si el coste de trasvasar agua es siempre el mismo, entonces la solución que minimiza el coste de la operación total para trasvasar agua entre embalses hasta que por fin se consigue el volumen objetivo en el embalse de destino será aquella que ejecute el operador `trasvasar` el menor número de veces.

   En el caso de problemas de optimización con costes unitarios, se considerarán los algoritmos de el primero en amplitud y de el primero en profundización iterativa:

   - El algoritmo del primero en amplitud es completo y, ademas, admisible cuando el coste de los operadores es siempre el mismo, como en este caso. Es decir, el coste será igual a la profundidad mínima a la que se genere un estado que contenga al embalse de destino con un volumen mayor o igual que el indicado en el *requerimiento de servicio*.

     Este algoritmo, sin embargo, tienen un consumo de memoria exponencial, aunque es particularmente bueno en el caso de que haya transposiciones.
   - El algoritmo de el primero en profundización iterativa realiza varias búsquedas de el primero en profundidad secuencialmente, incrementando la profundidad máxima en cada iteración. Por lo tanto, se trata de un algoritmo completo. Además, si el incremento de la profundidad máxima entre iteraciones es igual a la unidad, entonces la soluciones encontradas serán necesariamente óptimas en la profundidad de las soluciones —que, como ya se ha mencionado, resultan ser iguales al coste de las soluciones.

     Aunque este algoritmo tiene un consumo de memoria que es lineal en la profundidad a la que se encuentra la solución óptima, tiene dificultades en los espacios de estados con transposiciones, puesto que recorre todos los caminos que pasan por cada transposición.

   De hecho, este problema tiene una cantidad alta de transposiciones (esto es, es muy fácil generar el mismo estado con la ejecución de diferentes operaciones de `trasvasar`). Por lo tanto, el algoritmo recomendado es el algoritmo de el primero en amplitud.

4. Para obtener una función heurística, se sugiere el uso de la técnica de *relajación de restricciones*. En su aplicación, se observan las restricciones del problema y se relajan todas o un subconjunto de ellas hasta que es posible resolver el problema resultante de forma óptima. Como quiera que las restricciones del problema se encuentran típicamente en las *precondiciones* de los operadores del problema (estudiadas en el primer apartado), una relajación factible consiste en hacer que cada embalse pueda estar conectado a todos los demás, en vez de sólo aquellos que aparecen en su atributo `adyacentes`.

   La función que resuelve óptimamente el problema *relajado* resultante es, precisamente, el descrito en el primer apartado.

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.

   Como ya se indicó anteriormente, este problema tiene un gran número de transposiciones y, por ello, se sugiere el algoritmo A*.

6. ¡Por supuesto! Tal y como se explica en el primer apartado (que es precisamente el que explica como resolver óptimamente el problema relajado), el mismo cálculo vale para una función de coste u otra.

7. No, puesto que ahora es preciso considerar diferentes embalses. De hecho, basta con calcular la función heurística para todos los embalse de destino, y quedarse con el valor máximo obtenido para todos ellos puesto que, al menos, será preciso satisfacer el embalse de destino que más operaciones requiera.