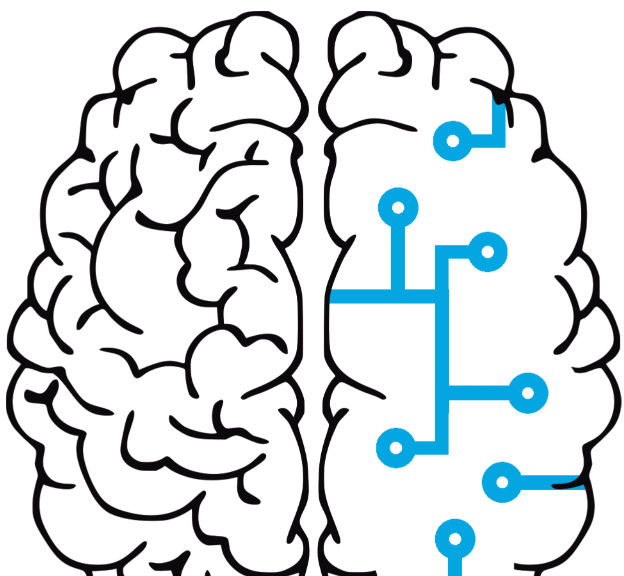
115

ENRIQUE DOMÍNGUEZ MERINO
EZEQUIEL LÓPEZ RUBIO
MIGUEL ÁNGEL MOLINA CABELLO
JOSÉ ANTONIO MONTENEGRO MONTES
FRANCISCO VILLALBA SÁNCHEZ

# *EJERCICIOS DE SISTEMAS INTELIGENTES*



UNIVERSIDAD DE MÁLAGA / MANUALES

El objetivo de este manual es proporcionar al estudiante el material de apoyo necesario para la asignatura de Sistemas Inteligentes, proponiendo y resolviendo distintos tipos de ejercicios teóricos y prácticos, mediante un breve recorrido por los distintos campos de la Inteligencia Artificial.

ENRIQUE DOMÍNGUEZ MERINO
EZEQUIEL LÓPEZ RUBIO
MIGUEL ÁNGEL MOLINA CABELLO
JOSÉ ANTONIO MONTENEGRO MONTES
FRANCISCO VILLALBA SÁNCHEZ

# EJERCICIOS DE SISTEMAS INTELIGENTES

UNIVERSIDAD DE MÁLAGA / MANUALES
2018

© UMA editorial. Universidad de Málaga
Bulevar Louis Pasteur, 30 (Campus de Teatinos) 29071 Málaga
www.uma.es/servicio-publicaciones-y-divulgacion-cientifica

Diseño de colección: J. M. Mercado Maquetación: Los autores

Colección: Manuales

ISBN: 978-84-17449-60-5

Este libro está editado en papel

Esta editorial es miembro de la UNE, lo que garantiza la difusión y comercialización de sus publicaciones a nivel nacional.

# Índice general

# Prefacio

Este manual ha sido fruto de la recopilación de numerosos ejercicios de clase y de exámenes durante varios años impartiendo la asignatura de Sistemas Inteligentes en los grados de Ingeniería Informática, Ingeniería del Software, Ingeniería de Computadores e Ingeniería de la Salud de la E.T.S.I. Informática de la Universidad de Málaga.

El objetivo de este manual es proporcionar al estudiante el material de apoyo que se ha ido proveyendo a los estudiantes en años anteriores a través de diferentes medios, en un único recurso para su comodidad y facilidad de uso.

Todo este trabajo se ha realizado gracias a la colaboración de los profesores implicados, los cuales también quieren agradecer el apoyo y colaboración de los profesores José Luis Pérez de la Cruz, Lawrence Mandow y Amparo Ruiz Sepúlveda, que de algún modo han facilitado y contribuido en la elaboración de este manual. Del mismo modo, deseamos dedicar esta obra a José Muñoz Pérez, profesor jubilado de nuestra Escuela, quien ha sido un ejemplo para todos nosotros.

# Tema 1

# Búsqueda

Para los problemas de búsqueda 1 al 7 planteados más abajo, contesta a las siguientes cuestiones y razona tus respuestas:

-Propón una representación adecuada de los estados.

-Describe el espacio de búsqueda.

-Detalla las acciones disponibles y sus costes.

-¿Cuál es la condición que debe cumplir un estado para ser final?

-¿Es el problema más adecuado para la búsqueda en árbol o para la búsqueda en grafo?

-¿Cuántos estados finales hay?

-¿Es posible dar un heurístico admisible/consistente para el problema? Si la respuesta es afirmativa, proporciona uno y demuestra su admisibilidad/consistencia.

-¿Qué algoritmo(s) de búsqueda serían adecuados para resolver el problema?

**Ejercicio 1.1 Dos amigos.** Una persona sale de Gerona y otra de Cádiz. En cada etapa, ambas personas pueden moverse desde la capital de una provincia peninsular de España a la capital de una provincia vecina. La tarea es encontrar una ruta para cada persona de manera que se encuentren en alguna capital lo antes posible. El viaje hacia la siguiente capital lo empiezan ambas personas simultáneamente, y ambas deben llegar a sus destinos antes de empezar con la siguiente etapa. Suponemos que sabemos el tiempo necesario para viajar desde cualquier capital a cualquier otra capital vecina.

**Solución**

**Dos amigos.** Un estado es un par ordenado $(a,b)$, donde $a$ es la posición actual de la primera persona, y $b$ es la posición actual de la segunda persona. Ambas posiciones deben ser capitales de provincias peninsulares españolas.

El espacio de estados es el conjunto de todos los pares ordenados con la estructura anterior. Como hay 47 provincias peninsulares españolas, esto implica que el espacio de estados tiene tamaño $47^2$.

Si el estado actual es $(a,b)$ las acciones disponibles son moverse a otro estado $(c,d)$ tal que $a$ y $c$ son capitales de provincias vecinas, y lo mismo ocurre con $b$ y $d$. El coste de la etapa es $\text{máx}(distancia(a,c), distancia(b,d))$, dado que la persona que llega antes a su destino debe esperar a la otra.

Un estado es final si $a{=}b$, es decir, si ambas personas están en el mismo sitio.

Siempre es posible volver a un estado previamente visitado, con lo cual necesitamos usar búsqueda en grafo para este problema.

Hay 47 estados finales de la forma $(a,a)$, donde $a$ es la capital de una provincia peninsular española.

Un heurístico consistente es $h(a,b){=}distancia(a,b)/2$, dado que la distancia entre los dos viajeros no puede reducirse en una etapa en más del doble del coste de la etapa:

$$distancia(a,b)\text{-}distancia(c,d) \leq 2\cdot\ \text{máx}(distancia(a,c),distancia(b,d)) \Rightarrow$$
$$distancia(a,b)/2 \leq \text{máx}(distancia\,a,c),distancia\,b,d))+distancia(c,d)/2 \Rightarrow$$
$$h(a,b) \leq coste((a,b),(c,d))+\ h(c,d)$$

El algoritmo A* sería particularmente adecuado para este problema, dado el heurístico consistente que acabamos de presentar.

**Ejercicio 1.2 Cruzar el puente.** Un grupo de 5 personas quiere cruzar un viejo y estrecho puente por la noche. Es necesario usar una linterna para cruzarlo. Sólo hay una linterna disponible, cuya batería está baja: sólo quedan 5 minutos antes de que se apague. Las personas necesitan 10, 30, 60, 80 y 120 segundos para cruzar el puente, respectivamente. El puente sólo puede sostener a 2 personas como mucho al mismo tiempo, y la velocidad a la que se cruza es la de la persona más lenta. La linterna no se puede lanzar de un lado al otro, así que es necesario que una persona cruce el puente de nuevo para llevarla al otro lado. La tarea es encontrar el modo más rápido de que todo el mundo llegue al lado contrario, si esto es posible.

**Solución**

**Cruzar el puente** Un estado es una terna ($A$, $B$, $c$), donde $A$ es el conjunto de personas en el lado inicial del puente, $B$ es el conjunto de personas en el lado final del puente, y $c$ es la ubicación (inicial/final) de la linterna. Notaremos a cada persona con el número de segundos que necesita para cruzar el puente.

El espacio de estados es el conjunto de todas las ternas con la estructura anteriormente citada, donde los dos conjuntos forman una partición del conjunto {10, 30, 60, 80, 120}, y $c \in$ {inicial, final}.

Si el coste de camino $g(n)$ desde el estado inicial ({10, 30, 60, 80, 120}, $\emptyset$, inicial) al estado actual $n$ es mayor o igual que 300 segundos, entonces no hay ninguna acción disponible (porque el tiempo se ha agotado). En otro caso, dado un estado consideraremos acciones consistentes en transferir una o dos personas desde el lado donde está la linterna hasta el otro lado. El resultado de una acción es un estado en el que la gente que ha cruzado el puente ha sido movida de un conjunto al otro, y la linterna ha cambiado de posición. El coste de una acción es el mayor de los valores de las personas que han cruzado, es decir, el número de segundos que tarda en cruzar la persona más lenta. Sin embargo, si el coste de camino desde el estado inicial hasta el estado resultante es estrictamente mayor que 300 segundos, entonces la acción no está disponible porque no hay tiempo para ejecutarla.

Para saber si un estado es final comprobamos si $B$ tiene 5 elementos, es decir, si todas las personas están en el lado final del puente.

Es posible volver a un estado considerado anteriormente haciendo que algunas personas vayan y vuelvan por el puente. Sin embargo, esto consume tiempo, así que en algún momento el conjunto de acciones disponibles se reducirá al conjunto vacío. Así pues, es posible usar la búsqueda en árbol puesto que el árbol de búsqueda es finito, y no demasiado grande. Si queremos usar la búsqueda en grafo, debemos asegurarnos de que los nodos no sean expandidos hasta que se haya encontrado el camino óptimo hasta ellos (usando Dijkstra o A*), ya que un estado puede alcanzarse mediante diferentes caminos, cada uno con su propio coste.

Sólo hay un estado final: ($\emptyset$, $\{10, 30, 60, 80, 120\}$, final). Nótese que no hay manera de que la linterna acabe en el lado inicial del puente si todo el mundo está en el lado final.

Se puede calcular un heurístico consistente ordenando los números del primer conjunto de estados en orden decreciente, y a continuación sumando los números que están en posiciones impares. Esto es consistente porque en un solo paso solo podemos transferir dos personas al lado final. Además, si la linterna está en el lado final y hay todavía gente en el lado inicial, podemos también sumar el número más pequeño del segundo conjunto del estado. Esto sigue siendo consistente porque si la linterna está en el lado final y no hemos llegado todavía al estado final sólo podemos continuar enviando al menos una persona al lado inicial.

Para este problema podemos usar A* o Dijkstra, dado que el espacio de búsqueda es relativamente pequeño. La profundización progresiva también funcionará, ya que es de esperar que haya una solución no demasiado profunda.

**Ejercicio 1.3 Reparto de pan.** Una empresa de panadería tiene una furgoneta para realizar el reparto diario de pan, que puede cargar hasta $P$ panes. Debe servir a $N$ restaurantes, y el i-ésimo restaurante necesita $c_i$ panes diariamente. La cantidad total de panes que repartir excede a $P$, pero hay panaderías de la empresa repartidas por toda la ciudad, donde la furgoneta puede recargar si es necesario. Supondremos que la furgoneta empieza el recorrido en alguna de las panaderías. La tarea es encontrar la ruta más corta para servir a todos los restaurantes, incluyendo las visitas a las panaderías que sean necesarias. Supondremos conocidas las distancias entre cualquier par de puntos de la ciudad.

**Solución**

**Reparto de pan.** Un estado es un vector ($A$, $B$, $c$, $d$), donde $A$ es el conjunto de restaurantes ya visitados, $B$ es el conjunto de restaurantes por visitar, $c$ es la posición actual de la furgoneta (que puede ser una panadería o un restaurante), y $d$ es el número de panes que hay en la furgoneta. El estado inicial es ($\emptyset$, *Todos-Los-Restaurantes*, *ningún-lugar*, 0), where *ningún-lugar* es una posición especial para indicar que todavía no hemos decidido dónde empezar.

El espacio de estados es el conjunto de todos los vectores que tienen la estructura mencionada más arriba.

Si $c$=*ningún-lugar* (estado inicial), las acciones disponibles son asignar el valor de $c$ a alguna panadería y recargar la furgoneta (poner $d$=$P$). El coste es cero porque podemos empezar donde queramos. En otro caso, dado un estado las acciones disponibles son:-

Ir a un restaurante $j \in B$ tal que $d \geq c_j$. Entonces ponemos $c$=$j$; se mueve $j$ desde $B$ hasta $A$; y se asigna $d$=$d$- $c_j$.-

Si la furgoneta no está llena ($d$<$P$), podemos ir a una panadería $k$ y poner $d$=$P$. En tal caso ponemos $c$=$k$ y $d$=$P$, y se dejan $A$ y $B$ sin cambios.

El coste de una acción es la distancia desde la anterior posición de la furgoneta $c$ a la posición que vamos a visitar.

Para saber si un estado es final comprobamos si $B$ está vacío.

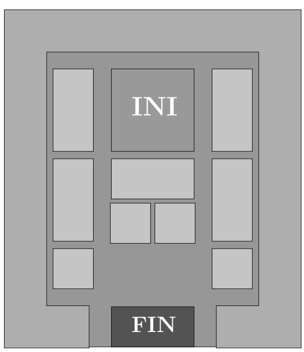
El problema es adecuado para la búsqueda en árbol, ya que no es posible volver a un estado considerado anteriormente. Esto se debe a que los elementos no se pueden mover desde $A$ hasta $B$, y no podemos visitar dos panaderías en medio de dos restaurantes.

Todo posible vector de la forma $(A, \emptyset, c, d)$ es un estado final. En este problema hay al menos $P+1$ estados finales, ya que tenemos $P+1$ valores posibles de $d$.

Un heurístico admisible es la distancia desde la posición actual $c$ hasta el restaurante más lejano de $B$. Esto es admisible porque tenemos que llegar a ese restaurante en cualquier caso. Un heurístico mejor es el coste del árbol de unión mínimo del conjunto $B \cup \{c\}$. Esto es admisible porque el árbol de unión da el coste de la manera más barata de conectar todas estas posiciones, ignorando el hecho de que podríamos tener que visitar algunas panaderías para obtener más panes.

El algoritmo A* sería particularmente adecuado para este problema, dado el heurístico consistente que acabamos de presentar, y el gran tamaño del espacio de búsqueda.

**Ejercicio 1.4 Klotski.** Dada la configuración de bloques del diagrama siguiente, el bloque más grande (INI) debe moverse hasta la posición central inferior (FIN) con el menor número de movimientos de bloques posible. No está permitido quitar bloques; solamente se pueden deslizar los bloques horizontal y verticalmente.




Klotski

**Solución**

**Klotski.** Un estado es una matriz 4x4 con la distribución actual de los bloques. Cada bloque se identifica mediante un número, pongamos 1...10. Podemos suponer que el bloque rojo es el número 1. Las celdas vacías contienen ceros.

El espacio de estados es el conjunto de todas las matrices 4x4 tales que todas sus celdas contienen números naturales en el intervalo 0...10, y tales que las formas de los bloques se respetan.

Las acciones disponibles en un cierto estado consisten en mover un bloque una celda hacia la izquierda, hacia la derecha, hacia arriba o hacia abajo, siempre que haya espacio para el movimiento. El coste de cada etapa es siempre 1.

Para ver si un estado es final comprobamos si el bloque INI está en la parte central inferior, esto es, si las celdas (3,2), (3,3), (4,2) y (4,3) de la matriz valen 1.

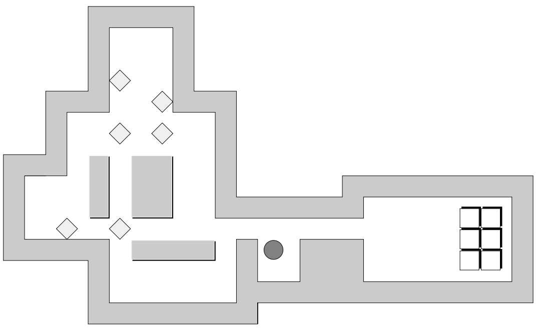
Siempre es posible volver a un estado previamente visitado, con lo cual necesitamos búsqueda en grafo para este problema.

Toda matriz del espacio de búsqueda tal que el bloque INI esté en la parte central inferior es un estado final. Por tanto, tenemos muchos estados finales.

Un heurístico consistente es la distancia Manhattan desde la posición actual de la celda más arriba y a la izquierda con valor 1 hasta la posición (3,2), que es la posición deseada de la parte superior izquierda del bloque rojo. En un paso esta función no puede decrecer en más de uno, que es el coste del paso, así que este heurístico es consistente.

El algoritmo A* sería particularmente adecuado para este problema, dado el heurístico consistente que acabamos de presentar.

**Ejercicio 1.5 Sokoban.** El agente (círculo) debe empujar los diamantes para llevarlos hasta los almacenes (señalados con recuadros huecos). Sólo se puede empujar un diamante cada vez. El agente no puede pasar por encima de los diamantes ni de las paredes. El problema está resuelto cuando todos los diamantes están en almacenes, y buscamos la solución con el menor número de movimientos del agente.



Sokoban

**Solución**

**Sokoban.** Un estado es una matriz del tamaño del mundo donde se mueve el agente, donde cada celda puede tener uno de estos valores: *vacía*, *diamante*, *agente*. Las coordenadas de las paredes y de los almacenes para los diamantes nunca cambian, así que no tiene sentido almacenarlas en el estado; las mantendremos en otro lugar.

El espacio de búsqueda es el conjunto de todas las matrices del tamaño del mundo tales que toda celda tiene uno de los valores permitidos, tales que sólo hay una celda marcada como agente, y que el número de diamantes sea el mismo que en el estado inicial.

Las acciones disponibles en un cierto estado consisten en mover el agente una celda a la izquierda, a la derecha, hacia arriba o hacia abajo, suponiendo que haya espacio para el movimiento. Si el movimiento implica empujar un diamante, debe haber una celda vacía detrás del diamante. El coste de paso es siempre 1.

Para ver si un estado es final comprobamos si todos los diamantes están en almacenes.

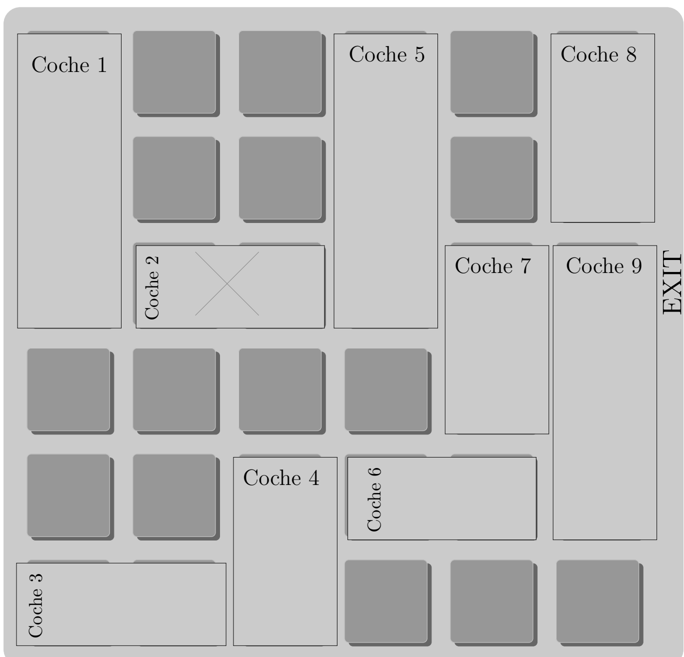
Hay muchas maneras de volver a un estado previamente visitado, así que necesitamos la búsqueda en grafo para este problema.

Los estados finales tienen todos los diamantes en almacenes, y el agente debe estar adyacente a uno de esos lugares. Por tanto, podemos tener varios estados finales, dependiendo de las coordenadas de los almacenes.

Un heurístico consistente es la suma de las distancias Manhattan desde cada diamante hasta el almacen más cercano. En un paso esta función no puede decrecer en más de uno, que es el coste del paso, así que el heurístico es consistente.

El algoritmo A* sería particularmente adecuado para este problema, dado el heurístico consistente que acabamos de presentar.

**Ejercicio 1.6 Hora punta.** El objetivo es sacar el coche marcado con X de una rejilla 6x6 llena de automóviles, moviendo los otros vehículos fuera de su camino. Los coches y camiones sólo pueden moverse en línea recta sobre la rejilla (hacia atrás o hacia delante). No se pueden rotar. Buscamos la solución con el menor número de movimientos; puedes suponer que los coches y camiones sólo pueden moverse un recuadro cada vez.



Hora punta

**Solución**

**Hora punta.** Un estado es una matriz 6x6 con la distribución actual de los vehículos. Cada vehículo puede identificarse mediante un número, pongamos 1...$m$. Podemos suponer que el coche X es el número 1. Las celdas vacías contendrán ceros.

El espacio de estados es el conjunto de todas las matrices 6x6 tales que todas las celdas contienen números naturales del intervalo 0... $m$, y tales que las formas y orientaciones de los vehículos se respetan.

Las acciones disponibles en un cierto estado son desplazar un vehículo hacia la izquierda, hacia la derecha, hacia arriba o hacia abajo, suponiendo que haya sitio para el movimiento, y que el vehículo no se mueva de lado. El coste de paso es siempre 1.

Para ver si un estado es final comprobamos si el coche X está en la salida, es decir, si las celdas (3,5) y (3,6) de la matriz valen 1.

Hay muchas maneras de volver a un estado previamente visitado, así que necesitamos la búsqueda en grafo para este problema.

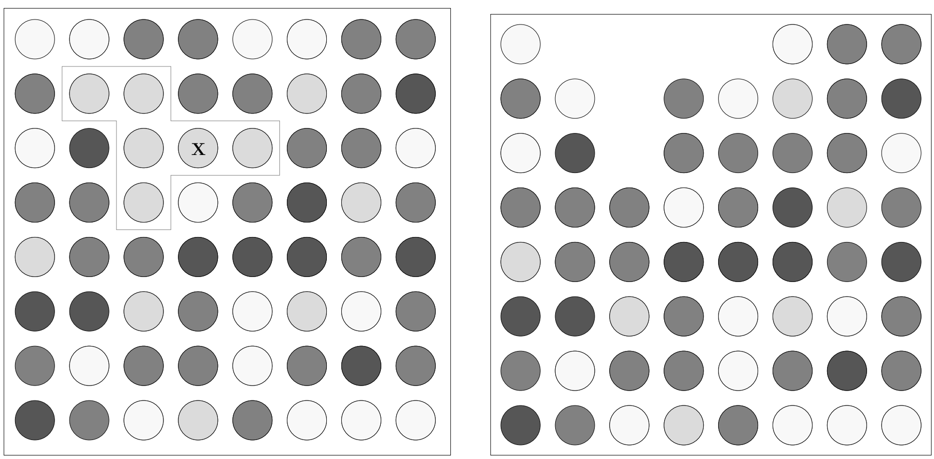
Toda matriz en el espacio de búsqueda tal que el coche X está en la salida es un estado final. Por tanto, tenemos muchos estados finales.

Un heurístico consistente es la distancia Manhattan desde la posición actual de la celda más arriba a la izquierda con valor 1 hasta la posición (3,5), que es la posición deseada de la parte izquierda del coche X. En un paso esta función no puede decrecer en más de uno, que es el coste del paso, así que el heurístico es consistente. Nótese que el coche X no puede moverse fuera de la fila 3, así que en realidad la distancia de Manhattan se reduce a una distancia en columnas.

El algoritmo A* sería particularmente adecuado para este problema, dado el heurístico consistente que acabamos de presentar.

**Ejercicio 1.7 Juegos de burbujas.** Tenemos una matriz con una burbuja en cada celda. Cada burbuja puede ser de cualquiera de los 4 tonos distintos. Una burbuja es una 4-vecina de otra burbuja, si la primera está inmediatamente encima, debajo, a la izquierda o a la derecha de la otra. Un conjunto de burbujas del mismo tono es un conjunto 4-conexo si para todo par de burbujas del conjunto existe una secuencia de burbujas 4-vecinas que las conecta.

Si tocamos una burbuja, entonces esa burbuja y todas las de su conjunto 4-conexo explotan, y las burbujas que hubiera encima caen. Como ejemplo, mostramos un posible estado inicial a la izquierda. La burbuja que va a ser tocada está señalada con una X, y su conjunto 4-conexo está señalado con una línea. El estado sucesor se muestra a la derecha; nótese cómo las burbujas que estaban encima de las que han explotado han caído.




Juego de Burbujas. Antes y después de explotar las burbujas seleccionadas

La tarea es explotar todas las burbujas con el menor número de toques posible.

**Solución**

**Juego de burbujas.** Un estado es una matriz 8x8 con la configuración actual de las burbujas. Cada tono se codifica con un número: 1...4. Las celdas vacías contendrán ceros.

El espacio de estados es el conjunto de todas las matrices 8x8 tales que todas sus celdas contienen números naturales en el intervalo 0...4.

Las acciones disponibles en un cierto estado consisten en tocar una de las burbujas restantes (codificadas como celdas no nulas en la matriz). Necesitaríamos un algoritmo de coloreado de globos (*blob coloring algorithm*) para hallar el conjunto 4-conexo correspondiente a la burbuja tocada. El resultado de la acción es un estado en el que todas las burbujas del conjunto 4-conexo han sido eliminadas, y las burbujas por encima de ellas han caído. El coste de paso siempre es 1.

Para ver si un estado es final comprobamos si todas las celdas de la matriz son cero.

No hay manera de volver a un estado anterior, dado que las burbujas explotadas no pueden ser recuperadas. Así pues, podemos usar búsqueda en árbol o búsqueda en grafo.

Sólo hay un estado final, que es una matriz 8x8 llena de ceros.

Se puede obtener un heurístico consistente calculando para cada tono un vector booleano 8x1 que indique si hay al menos una burbuja de ese tono en cada una de las 8 columnas. A continuación contamos el número de secuencias de valores *true* (1) del vector. Por ejemplo, para el vector (false,true,true,true,false,false,**true,true**) el recuento es 2 porque hay dos secuencias de valores true (hemos señalado la segunda secuencia en **negrita**. El heurístico consistente se obtiene entonces sumando estos recuentos para los 4 tonos. El heurístico es consistente porque en un paso (un toque) sólo podemos eliminar burbujas del mismo tono que estén en columnas adyacentes, y además la caída de las burbujas no puede cambiar ninguna burbuja de una columna a otra. Así pues, el heurístico sólo puede permanecer invariable o bien decrecer en uno en cada paso, que es menor o igual que el coste de paso.

El algoritmo A* sería particularmente adecuado para este problema, dado el heurístico consistente que acabamos de presentar.

**Ejercicio 1.8** Considera un problema de búsqueda en el que todo nodo del árbol de búsqueda tiene exactamente $b$ hijos. Supongamos que sólo hay un estado final, que está a profundidad $d$ (entendemos que el nodo raíz está a profundidad 0).

a) Calcula, en el mejor y peor caso, el número de nodos que serán analizados antes de encontrar el estado final cuando se usa la búsqueda primero en anchura. Haz lo mismo para la búsqueda primero en profundidad.

b) Calcula, en el mejor y peor caso, el máximo número de nodos que podrían estar esperando para ser expandidos cuando se usa la búsqueda primero en anchura. Haz lo mismo para la búsqueda por profundización progresiva.

**Solución**

a) La búsqueda primero en anchura explorará los niveles 0...$d$-1 por completo antes de llegar al estado final. Estos niveles contienen $b^0+b^1+...+b^{d\text{-}1}$ nodos, y todos ellos deben ser explorados. Hay $b^d$ nodos en el nivel $d$, y podría ocurrir que sólo necesitáramos analizar el primero de ellos, si tenemos suficiente suerte. Si tenemos mala suerte, podríamos tener que analizar todos los textitb$^d$ nodos del nivel *d*. Por consiguiente, el número de nodos que analizará la búsqueda primero en anchura estará entre $b^0+b^1+...+b^{d\text{-}1}+1$ nodos (mejor caso) y $b^0+b^1+...+b^d$ nodos (peor caso).

La búsqueda primero en profundidad puede alcanzar la solución a profundidad $d$ analizando únicamente los nodos del camino desde el estado inicial hasta el estado final. Por tanto, en el mejor caso la búsqueda primero en profundidad analiza $d+1$ nodos para llegar al estado final. En el peor caso, podríamos elegir un camino donde no está el estado final, con lo que el número de nodos explorados sería infinito. Es decir, el estado final nunca sería alcanzado.

b) La búsqueda primero en anchura almacena en el conjunto de nodos que se deben explorar todos los nodos del siguiente nivel mientras está expandiendo los nodos del nivel actual. Así pues, cuando acabamos de expandir el último nodo del nivel $d$-1, tenemos $b^d$ nodos en el conjunto de nodos que expandir. Este es el mejor caso, puesto que si el estado final es el último nodo del nivel $d$, entonces tendremos que almacenar en dicho conjunto todos los nodos del nivel $d$+1 excepto los hijos del estado final. Por tanto, en el peor caso tendríamos $b^{d+1}$-$b$ nodos en el conjunto de nodos que expandir.

La búsqueda por profundización progresiva sólo necesita almacenar los hijos sin expandir de los nodos del camino que se está examinando en un momento dado. Si la solución está a profundidad $d$, cuando entramos por primera vez en ese nivel tenemos $b$-1 nodos por expandir en todos los niveles desde el 1 al $d$-1. Así, en ese momento el número de nodos del conjunto de nodos por ser expandidos es $(b\text{-}1)\cdot(d\text{-}1)$. Como los nodos del nivel $d$ nunca serán expandidos, $(b\text{-}1)\cdot(d\text{-}1)$ es tanto el mejor como el peor caso para el número máximo de nodos por expandir.

**Ejercicio 1.9** Da un ejemplo de un problema de búsqueda en el que la búsqueda en árbol nunca pueda entrar en un bucle infinito. Explica tu respuesta.

**Solución**

Los problemas *reparto de pan* y *juego de burbujas* son tales que no hay manera de volver a un estado anterior (véase más arriba para más detalles). Por tanto, la búsqueda en árbol nunca puede generar un nodo con el mismo estado que uno de los ascendientes de ese nodo. ¿Puedes dar otros ejemplos?

**Ejercicio 1.10** Tres misioneros y tres caníbales quieren cruzar un río. Para ello disponen de una barca en la que pueden subir una o dos personas. No debe haber en ningún momento más caníbales que misioneros en ningún lado del río, pues podría resultar peligroso. Se pide representar el problema mediante el formalismo del espacio de estados.

**Solución**

Debemos representar los estados y las acciones que permiten pasar de unos a otros. Para las acciones, hay que definir sus efectos y cuándo son aplicables.

Podemos representar los **estados** mediante una lista de tres elementos (M, C, B), donde:

- M = número de misioneros en la orilla derecha (0, 1, 2, 3)
- C = número de caníbales en la orilla derecha (0, 1, 2, 3)
- B = número de barcas en la orilla derecha (0, 1)

Supongamos una **instancia del problema** en la que inicialmente todos están en la derecha, y que el objetivo es que todos pasen a la izquierda. El problema vendrá definido por:

- Estado inicial: (3,3,1)
- Condición objetivo: (M = 0) $\wedge$ (C = 0)

Existen 10 **acciones** posibles, que podemos agrupar en 5 grupos. Sus **efectos** vienen descritos por las siguientes reglas:

a) Cruza el río un solo misionero:

$$(M, C, 1) \rightarrow (M-1, C, 0)$$
$$(M, C, 0) \rightarrow (M+1, C, 1)$$

b) Cruza el río un solo caníbal:

$$(M, C, 1) \rightarrow (M, C-1, 0)$$
$$(M, C, 0) \rightarrow (M, C+1, 1)$$

c) Cruzan el río dos misioneros:

$$(M, C, 1) \rightarrow (M-2, C, 0)$$
$$(M, C, 0) \rightarrow (M+2, C, 1)$$

d) Cruzan el río dos caníbales:

$$(M, C, 1) \rightarrow (M, C-2, 0)$$
$$(M, C, 0) \rightarrow (M, C+2, 1)$$

e) Cruzan el río un misionero y un caníbal:

$$(M, C, 1) \rightarrow (M-1, C-1, 0)$$
$$(M, C, 0) \rightarrow (M+1, C+1, 1)$$

Estas acciones no son siempre aplicables. Debemos proporcionar una textbfprecondición para cada una de ellas. En este caso, lo más fácil para saber si una regla es aplicable o no es calcular su resultado y comprobar si es un estado **no válido** de acuerdo con la siguiente condición:

(M < 0) $\vee$ (M > 3) $\vee$ //número incorrecto de misioneros

(C < 0) $\vee$ (C > 3) $\vee$ //número incorrecto de caníbales

((M $\neq$ 0) $\vee$ (C > M)) $\vee$ //más caníbales que misioneros en la orilla derecha

((M $\neq$ 3) $\wedge$ (3-C > 3-M)) //más caníbales que misioneros en la orilla izquierda

**Ejercicio 1.11** Un granjero se encuentra a la orilla de un río junto a sus tres pertenencias: un lobo, una cabra, y una inmensa col. El agricultor desea cruzar el río junto con sus pertenencias. Para ello dispone de una barca para cruzar a la otra orilla, pero en la barca solo caben él y una de sus pertenencias.

**Solución**

Representaremos los estados mediante la tupla (barca,lobo,cabra,col), donde cada elemento de la tupla será 1 si el objeto indicado está en la orilla inicial y 0 si esta donde quiere cruzar el agricultor. Se supone que el agricultor está siempre donde está la barca.

El estado inicial será el representado por al tupla (1,1,1,1), mientras que el estado final será el (0,0,0,0)

Las acciones vienen representadas por las siguientes transiciones:

- pasar la cabra DESTINO $(B, L, CA, CO) \rightarrow (0, L, 0, CO)$
- pasar el lobo DESTINO $(B, L, CA, CO) \rightarrow (0, 0, CA, CO)$
- pasar la col DESTINO $(B, L, CA, CO) \rightarrow (0, L, CA, 0)$
- pasar solo a DESTINO $(1, L, CA, CO) \rightarrow (0, L, CA, CO)$
- pasar la cabra ORIGEN $(B, L, CA, CO) \rightarrow (1, L, 1, CO)$
- pasar el lobo ORIGEN $(B, L, CA, CO) \rightarrow (1, 1, CA, CO)$
- pasar la col ORIGEN $(B, L, CA, CO) \rightarrow (1, L, CA, 1)$

- pasar solo a ORIGEN $(0, L, CA, CO) \rightarrow (1, L, CA, CO)$

Las precondiciones serán:

- para pasar cabra a DESTINO B = 1 , CA = 1.
- para pasar lobo a DESTINO B = 1 , L = 1 y (CO = 0 o CA = 0)
- para pasar col a DESTINO B = 1 , CO = 1 y (L = 0 o CA = 0)
- pasar solo a DESTINO (CO $\neq$ CA y L $\neq$ CA)
- para pasar cabra a ORIGEN B = 0 , CA = 0.
- para pasar lobo a ORIGEN B = 0 , L = 0 y (CO = 1 o CA = 1)
- para pasar col a ORIGEN B = 0 , CO = 0 y (L = 1 o CA = 1)
- pasar solo a ORIGEN (CO $\neq$ CA y L $\neq$ CA)

**Ejercicio 1.12** Disponemos de un conjunto de cadenas de caracteres, todas ellas de longitud menor o igual que 3 caracteres. Empezando con la cadena vacía, debemos encontrar una manera de concatenar cadenas del conjunto para formar una cadena objetivo. Por ejemplo, si el conjunto de cadenas es

$$\{ \text{ aa, ba, c, eee, ea, bd, fa, a } \}$$

y la cadena objetivo es eaaabd, entonces una posible solución es

$$\text{ea} \bullet \text{aa} \bullet \text{bd} = \text{eaaabd}$$

Queremos la solución que tenga el mínimo número de operaciones de concatenación, y podemos usar la misma cadena del conjunto todas las veces que sea necesario. Se pide representar el problema mediante el formalismo del espacio de estados.

**Solución**

Un estado es una cadena de caracteres. El espacio de estados es el conjunto de todas las cadenas posibles.

Si el estado actual es $w$, entonces las acciones disponibles son concatenar cada cadena $v$ del conjunto, bien a la izquierda (con lo cual el próximo estado es $w'=vw$) o a la derecha (con lo cual el próximo estado es $w'=wv$), supuesto que la cadena resultante $w'$ sea una subcadena del objetivo $g$. El coste de cada acción es 1, dado que buscamos la solución con el mínimo número de concatenaciones.

No se puede volver a un estado visitado previamente, así que podemos usar búsqueda en árbol para este problema.

Un posible heurístico consistente es:

$$h\left(w\right) = \left\lceil \frac{|g| - |w|}{3} \right\rceil$$

donde $|.|$ denota la longitud de una cadena. Es admisible porque la longitud de la cadena actual no puede aumentar en un paso en más de tres caracteres:

$$|w'|-|w| \leq 3 \Rightarrow -|w| \leq 3-|w'| \Rightarrow |g|-|w| \leq 3+|g|-|w'| \Rightarrow$$

$$\frac{|g| - |w|}{3} \leq 1 + \frac{|g| - |w'|}{3} \Rightarrow$$

$$\left\lceil \frac{|g| - |w|}{3} \right\rceil \leq 1 + \left\lceil \frac{|g| - |w'|}{3} \right\rceil \Rightarrow$$

$$h(w) \leq \mathit{coste}(w,w') + h(w')$$

El algoritmo A* es particularmente adecuado para este problema, dado el heurístico consistente que acabamos de ver.

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| A | | 10 | 15 | | | | |
| B | | | | 5 | 15 | | |
| C | | | | | 5 | 15 | |
| D | | | | | | | 20 |
| E | | | | | | 5 | 5 |
| F | | | | | | | 20 |
| G | | | | | | | |

**Ejercicio 1.13** Las distancias por carretera entre las ciudades de una región vienen dadas en kilómetros en la tabla siguiente:

La siguiente tabla contiene una estimación optimista de la distancia en kilómetros desde la ciudad G a cada una de las demás ciudades:

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| Estimación | 19 | 9 | 9 | 4 | 4 | 3 | 0 |

a) Aplicar el algoritmo A* para encontrar un camino desde la ciudad A a la ciudad G. Indicar para cada iteración el nodo seleccionado para expansión, sus sucesores y los valores correspondientes de g(n), h(n) y f(n).

| Nodo seleccionado | Sucesor | f = g + h | Orden en que se han cerrado |
|---|---|---|---|
| A | B | Valor f de b | ( Orden de b) |
| | C | Valor f de c | ( Orden de c) |

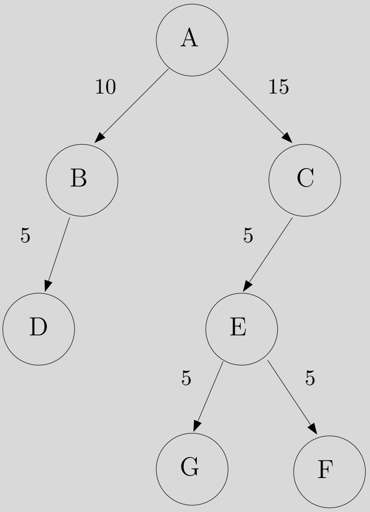
b) Mostrar el árbol de búsqueda resultante, detallando las operaciones realizadas sobre el mismo.

**Solución**

a) La aplicación del algoritmo A* produce la siguiente tabla:

| Nodo expandido | Sucesor f = g + h | Orden en que se han cerrado |
|---|---|---|
| A | B 10+9=19<br>C 15+9=24 | (1)<br>(3) |
| B | D 15+4=19<br>~~E 25+4=29~~<br>A - produce un ciclo | (2)<br>* |
| D | B produce un ciclo<br>~~G 35+0=35~~<br>Nota: aunque hemos encontrado un comino solución hasta G la búsqueda continúa, pues aún no estamos seguros de que sea el de mínimo coste. | <br>* |
| C | A - produce un ciclo<br>E 20+4=24<br>~~F 30+3=33~~<br>Nota: en esta iteración se encuentra un nuevo camino hasta E, con coste 24. Puesto que el camino anterior costaba 29 se tacha y nos quedamos con el nuevo. | <br>(4)<br>* |
| E | C – produce un ciclo<br>F 25+3=28<br>G 25+0=25<br>B - Este camino a B es peor | <br><br>(5) |
| G | Al haber seleccionado el nodo objetivo (G), estamos seguros de haber encontrado el camino de menor coste: A-C-E-G (coste:25) | |

b) El árbol de búsqueda resultante se muestra a continuación:




Árbol de exploración

**Ejercicio 1.14** Un robot "rescatador" se encuentra en una mina en la posición A; en el lugar B se encuentra un minero herido que debe ser recogido inmediatamente y depositado en la salida C. Cada galería de la mina mide 100 metros de longitud (nótese que el dibujo no está a escala).

Además téngase en cuenta que:

- El robot avanza a 10 m/s en horizontal.
- El robot "sube" a 5 m/s y baja a 20 m/s.
- Las aristas en trazo grueso son pasillos obstruidos y en ellos el robot avanza 5 veces más lento que en condiciones normales.

Se pide:

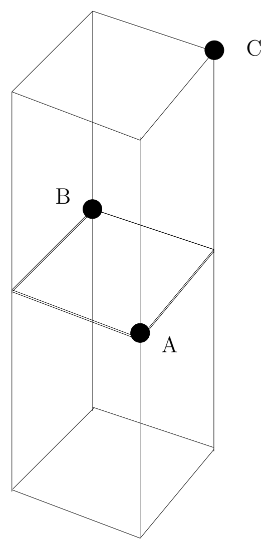
a) Formalizar el problema como búsqueda en un espacio de estados.

b) Dada la siguiente heurística:

$$f(x, y, z) = \frac{\mid x - x_{fin} \mid + \mid y - y_{fin} \mid}{10} + \frac{\mid z - z_{fin} \mid}{20}$$

Donde $x_{fin}$ , $y_{fin}$, $z_{fin}$ son las coordenadas del estado objetivo. Razona si dicha heurística es admisible o no y por qué.

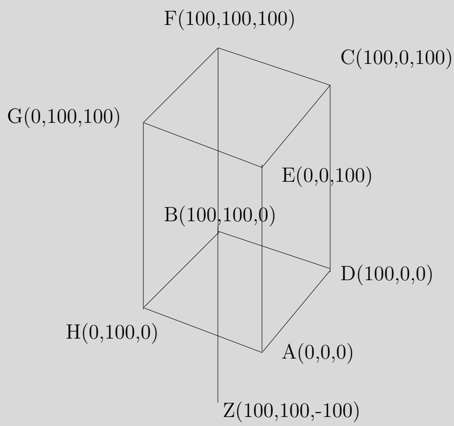
c) Utilizar el algoritmo A* para encontrar un camino óptimo desde B hasta C con la heurística proporcionada indicando en cada paso el nodo seleccionado para expansión y los sucesores del mismo, el grafo y el árbol de exploración.




Robot rescatador

**Solución**

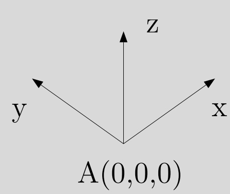
Primero establecemos las coordenadas de cada punto relevante en el problema y le asignamos una etiqueta.




Espacio de estados del problema

a) Formulación del problema: En realidad son 2 problemas.

1. Encontrar un camino óptimo de A a B.
2. Encontrar un camino óptimo de B a C.




Sistema de coordenadas

Reglas de producción:

R1 (avance horizontal): (x,y,z)→(x±100,y,z),(x,y,z)→(x,y±100,z)

R2 (subir): (x,y,z)→(x,y,z+100)

R3 (bajar): (x,y,z)→(x,y,z-100)

Coste

| Precondición | Sin obstáculo | Con obstáculo |
|---|---|---|
| Hay Galería | 10 | 50 |
| Hay Galería | 10 | 50 |
| Hay Galería | 20 | 100 |
| Hay Galería | 5 | 25 |

b) Siendo optimistas, suponemos que siempre hay galería y no hay obstáculos, luego siempre circulamos a máxima velocidad (10 m/sg en horizontal y 20 m/sg en vertical).

¿Satisface la RM? Es decir h($n_i$)-h($n_j$) $\leq$ c($n_i$,$n_j$) $\forall$ $n_i$,$n_j$ / $n_j$ es hijo de $n_i$

Hay dos casos:

1. Me acerco al objetivo en horizontal y en vertical.
2. Me alejo del objetivo en horizontal y en vertical.

Caso 1. Nos acercamos al objetivo en horizontal

$$\frac{|x - x_{fin}| + |y - y_{fin}|}{10} + \frac{|z - z_{fin}|}{20} - \frac{|x' - x_{fin}| + |y - y_{fin}|}{10} + \frac{|z - z_{fin}|}{20} \leq c$$

10 seg $\leq$ c siempre

Caso 2. Nos acercamos en vertical

$$\Delta h = \frac{|z - z_{fin}| - |z' - z_{fin}|}{20} \leq c$$

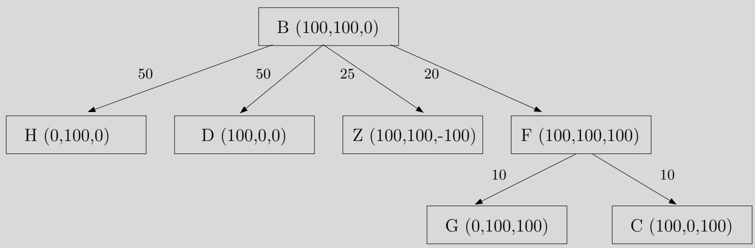
5 seg $\leq$ c siempre

Como se cumple en todos los casos que $h(n_i) - h(n_j) \leq c(n_i, n_j)\ \forall n_i, n_j/n_j$ es hijo de $n_i$, entonces el heurístico propuesto satisface la restricción monótona.

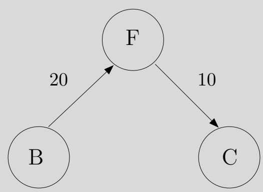
c) A* para encontrar un C óptimo desde B hasta C.

| Nodo expandido | Sucesor f = g + h | Orden en que se han cerrado |
|---|---|---|
| | f(B)0+10+5=15 | |
| B | f(H)=50+25=75<br>f(D)=50+15=65<br>f(Z)=25+20=45<br>f(F)=20+10=30 | <br><br><br>(1) |
| (F) | h(G)= 30+20=50<br>h(C)= 30+0=30 | <br>(2) |
| (C) | | Solución |

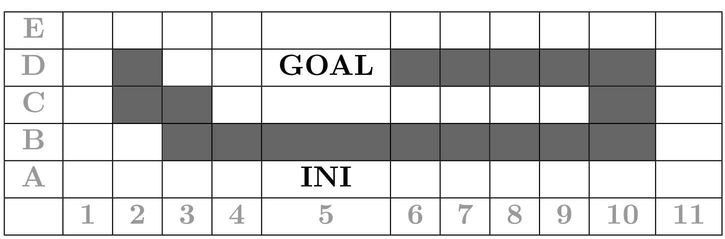
Grafo y árbol de exploración.




Solución final, coste 30 segundos.




**Ejercicio 1.15** Según la siguiente representación:



a) Calcule la heurística Manhattan a cada celda, teniendo en cuenta que la casilla origen es (A5) y la casilla destino (D5) (Rellene el cuadro).

b) Ejecute el algoritmo A* con la heurística Distancia Manhattan y compare con la ejecución manual del algoritmo y la heurística señalada, rellenando la siguiente tabla adjunta como ejemplo.

| Nodo seleccionado | Sucesor | f = g + h | Orden en que se cierran |
|---|---|---|---|

c) Al finalizar el algoritmo establezca la secuencias de acciones de la solución y su coste.

**Solución**

a) Cálculo de la heurística de Manhattan a cada celda, las casillas inicial(A5) y final(D5) se encuentran en negrita.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| E | 5 | 4 | 3 | 2 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| D | 4 | | 2 | 1 | **F** | | | | | | 6 |
| C | 5 | | | 2 | 1 | 2 | 3 | 4 | 5 | | 7 |
| B | **6** | 5 | | | | | | | | | 8 |
| A | 7 | 6 | 5 | 4 | **3** | 4 | 5 | 6 | 7 | 8 | 9 |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |

b) Ejecución del algoritmo A*.

| Nodo seleccionado | Sucesor g + h = f | Orden en que se han cerrado |
|---|---|---|
| A5 | A4 = 1+4=5<br>A6 = 1+4=5 | (1)<br>(2) |
| A4 | A3 = 2+5= 7 | (3) |
| A6 | A7 = 2+5= 7 | (4) |
| A3 | A2 = 3+6= 9 | (5) |
| A7 | A8 = 3+6= 9 | (7) |
| A2 | B2 = 4+5= 9<br>A1 = 4+7=11 | (6)<br>(11) |
| B2 | B1 = 5+6= 11 | (8) |
| A8 | A9 = 4+7= 11 | (12) |
| B1 | C1 = 6+5= 11<br>A1 = 6+7= 13 | (9)<br>No mejora anterior |
| C1 | D1 = 7+4= 11 | (10) |
| D1 | E1 = 8+5= 13 | (13) |
| A1 | B1 y A2 cerrados | (-) |
| A9 | A10 = 5+8= 13 | () |
| E1 | E2 = 9+4= 13 | (14) |
| E2 | E3 = 10+3= 13 | (15) |
| E3 | E4 = 11+2= 13<br>D3 = 11+2= 13 | ()<br>(16) |
| D3 | D4 = 12+1= 13 | (17) |
| D4 | E4 = 13+1= 14<br>C4 = 13+2= 15<br>D5 = 13+0= 13 | ()<br>()<br>(18) |
| D5 | OBJETIVO | COSTE 13 |

c) Solución y secuencia de acciones.

D5-D4-D3-E3-E2-E1-D1-C1-B1-B2-A2-A3-A4-A5

A5(izq)-A4(izq)-A3(izq)-A2(arriba)-B2(izq)-B1(arriba)-C1(arriba)-D1(arriba)-E1(derecha)-E2(derecha)-E3(derecha)-D3(abajo)-D4(derecha)-D5

**Ejercicio 1.16** En el siguiente mapa, las celdas blancas tienen un coste 1 y las celdas marcadas con x tienen un coste 2. Basándonos en el mapa, ejecutaremos algoritmos de búsqueda, teniendo en cuenta que en caso de empate seleccionamos la casilla con menor valor numérico en el eje x, y si persiste el empate seleccionamos el menor valor del eje y. Las acciones disponibles en cada casilla son: Arriba, Abajo, Izquierda y Derecha.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| G | | | | | | | | |
| F | | | | | | | | |
| E | | | | | | | | |
| D | | x | x | x | | GOAL | | |
| C | | | | | | | | |
| B | | | | | | | | |
| A | INI | | | | | | | |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

a) Calcule la heurística Manhattan a cada celda, teniendo en cuenta que la casilla destino es D6 (GOAL)

b) Aplique el algoritmo A* con la heurística Distancia Manhattan previamente calculada, rellenando la siguiente tabla adjunta como ejemplo.

| Nodo Seleccionado | Sucesor g + h = f | Orden en que se cierran |
|---|---|---|

c) Especifique la secuencia de acciones de la solución hallada y su coste.

**Solución**

a) Cálculo de la heurística de Manhattan a cada celda, las casillas inicial I(A1) y final F(D6) se encuentran en negrita.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| G | 8 | 7 | 6 | 5 | 4 | 3 | 4 | 5 |
| F | 7 | 6 | 5 | | | 2 | 3 | 4 |
| E | 6 | 5 | 4 | | 2 | **1** | 2 | 3 |
| D | 5 | 4 | 3 | 2 | **1** | **F** | 1 | 2 |
| C | 6 | 5 | 4 | | 2 | 1 | 2 | 3 |
| B | **7** | 6 | 5 | | | 2 | 3 | 4 |
| A | **I** | 7 | 6 | 5 | 4 | 3 | 4 | 5 |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

b) Ejecución del algoritmo A*.

| Nodo seleccionado | Sucesor g+h = f | Orden en que se han cerrado |
|---|---|---|
| A1 | B1 = 1+7=8<br>A2 = 1+7=8 | (1)<br>(4) |
| B1 | C1 = 2+6=8<br>B2 = 2+6=8 | (2)<br>(5) |
| C1 | D1 = 3+5=8<br>C2 = 3+5=8 | (3)<br>(6) |
| D1 | E1 = 4+6=10<br>D2 = 5+4=9 (gasto 2) | |
| A2 | ~~B2 = 2+6=8~~ Mismo valor<br>A3 = 2+6=8 | ()<br>(8) |
| B2 | ~~C2 = 3+5=8~~ Mismo valor<br>B3 = 3+5=8 | ()<br>(7) |
| C2 | D2 = 5+4= 9 (gasto 2)<br>C3 = 4+4 =8 | ()<br>(9) |
| B3 | ~~C3 = 4+4 = 8~~ Mismo valor<br>~~A3 = 4+ 6=10~~ peor valor | () |
| A3 | A4 = 3+5= 8 | (10) |
| C3 | D3 = 6+3= 9 (gasto 2) | |
| A4 | A5 = 4+4= 8 | (11) |
| A5 | A6 = 5+3= 8 | (12) |
| A6 | A7 = 6+4= 10<br>B6 = 6+2=8 | ()<br>(13) |
| B6 | B7 = 7+3= 10<br>C6 = 7+1 =8 | ()<br>(14) |
| C6 | C5 = 8+2= 10<br>D6 = 8+0 =8<br>C7= 8+2 =10 | ()<br>(15) |
| D6 | OBJETIVO | COSTE 8 |

c) Solución y secuencia de acciones.

Coste=8: D6-C6-B6-A6-A5-A4-A3-A2-A1

A1(izq)-A2(izq)-A3(izq)-A4(izq)-A5(izq)-A6(izq)-B6(arr)-C6(arr)-D6

**Ejercicio 1.17** En una empresa existen 5 empleados y se ha comprobado estadísticamente que el tiempo que tardan en desarrollar una determinada tarea cada uno de ellos es diferente dependiendo de la ciudad en la que se encuentren.

Se necesita de un procedimiento que dada una tabla de tiempos empleados en realizar esa determinada tarea por empleado y ciudad, sea capaz de realizar una asignación de cada empleado por ciudad de forma que cada uno de ellos sólo se dedique a realizar esa tarea en una única ciudad. Además, el tiempo total dedicado a realizar esas tareas en las ciudades deberá ser el menor posible.

Para ello se pide:

a) Formalizar detalladamente el problema para resolverlo utilizando el algoritmo A*

b) Hallar una función heurística sencilla para el problema que satisfaga las condiciones de A* (no se admite $\forall n h(n) = 0$). Demostrar que el heurístico propuesto es admisible.

c) Resolver el problema correspondiente a los datos de la siguiente tabla empleando el algoritmo A* y el heurístico propuesto en el apartado (b). Mostrar el grafo de búsqueda generado en cada iteración por A*, la solución encontrada y su coste.

| | Ciudad 1 | Ciudad 2 | Ciudad 3 | Ciudad 3 |
|---|---|---|---|---|
| E1 | 17 | 11 | 18 | 22 |
| E2 | 43 | 27 | 13 | 9 |
| E3 | 21 | 33 | 7 | 19 |
| E4 | 8 | 28 | 34 | 24 |

(tiempo dedicado por cada empleado/ciudad en realizar esa tarea)

**Solución**

a) Una posible formalización, una lista de asignaciones de empleados a tareas.

$E_{ij}$: El empleado i realiza la tarea en la ciudad j.

$E_{inicial}$: () Lista vacía

$E_{Objetivo}$: $(E_{1a}, E_{2b}, E_{3c}, E_{4d})$ donde a,b,c,d $\in\{1,..,4\}$ y a $\neq$ b $\neq$ c$\neq$ d

Para evitar muchos sucesores asignaremos las tareas a los empleados en orden con estas reglas:

- $E_{ij}$ es aplicable si $E_{i-1k}$ ya se ha aplicado
- El coste para llegar al estado $E_{ij}$=Tabla[i,j]

b) Consideraremos el siguiente heurístico:

$h(Estado) = \sum_{i=filaporasignar} min(fila(i))$, es decir, de todas las filas no asignadas escogemos el mínimo y se suman.

h es claramente admisible pues el coste de las asignaciones será mayor o igual a la suma de los mínimos por fila.

c) Tabla generada por el algoritmo A*.

| Nodo seleccionado | Sucesor g+h = f | Orden en que se han cerrado |
|---|---|---|
| $E_{ini}$ | $E_{11}$ = 17+(9+7+24) = 57<br>$E_{12}$ = 11+(9+7+8) = 35<br>$E_{13}$ = 18 + (9+19+8) = 54<br>$E_{14}$ = 22 + (13+7+8) = 50 | ()<br>(1) |
| $E_{12}$ | $E_{21}$ = 54 + (7+24) = 85<br>$E_{23}$ = 24 + (19+8) = 51<br>$E_{24}$ = 20 + (7+8) = 35 | ()<br>(2) |
| $E_{24}$ | $E_{31}$ = 41 + 34 = 75<br>$E_{33}$ = 27 + 8 = 35 | ()<br>(3) |
| $E_{33}$ | $E_{41}$ = 35 + 0 = 35 | (4) |
| $E_{41}$ | OBJETIVO | COSTE 35 |

Camino Solución: $E_{12}, E_{24}, E_{33}, E_{41}$. Coste= 35

**Ejercicio 1.18** Aplicar el algoritmo A* para hallar un camino que una las ciudades A y G. Las distancias por carretera entre las ciudades de la región vienen dadas en la tabla siguiente (se supone que las carreteras son de doble sentido, y por tanto la matriz es simétrica respecto a la diagonal):

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| A | | 80 | 70 | | | | |
| B | | | | 60 | | 130 | |
| C | | | | 80 | 90 | | |
| D | | | | | | 140 | 65 |
| E | | | | | | | 85 |
| F | | | | | | | |
| G | | | | | | | |

y las distancias aéreas a la ciudad G son:

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| Distancia a G | 130 | 120 | 78 | 60 | 50 | 130 | 0 |

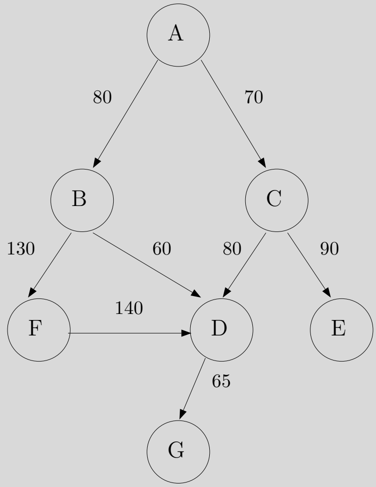
a) Mostrar el árbol de búsqueda resultante y

b) Una tabla donde se detalle en cada paso el nodo seleccionado, así como los valores de g(n), h(n) y f(n) de los nodos sucesores generados.

| Nodo Seleccionado | Sucesor g + h = f | Orden en que se cierran |
|---|---|---|

**Solución**

a) Árbol de búsqueda resultante.




b) Tabla generada.

| Nodo seleccionado | Sucesor g+h = f | Orden en que se han cerrado |
|---|---|---|
| A | B=80+120=200<br>C=70+78=148 | (2)<br>(1) |
| C | D=(70+80)+60=210<br>E=(70+90)+50=210 | (Mejor por B) |
| B | D=(80+60)+60=200<br>F=(80+130)+130=340 | (3) |
| D | F=(80+60+140)+130=410<br>G=(80+60+65)+0=205 | (Mejor por B)<br>(4) |
| G | OBJETIVO | COSTE 205 |

**Ejercicio 1.19** Consideremos una red de carreteras entre varias ciudades. La siguiente tabla muestra las distancias recorridas en kilómetros cuando hay una carretera que conecta dos ciudades directamente:

| | **c0** | **c1** | **c2** | **c3** | **c4** | **c5** | **c6** | **c7** |
|---|---|---|---|---|---|---|---|---|
| **c0** | | | 15 | | | 11 | 13 | |
| **c1** | | | | | 14 | 15 | | |
| **c2** | 15 | | | | 6 | | | |
| **c3** | | | | | 4 | | 21 | |
| **c4** | | 14 | 6 | 4 | | | | |
| **c5** | 11 | 15 | | | | | | 14 |
| **c6** | 13 | | | 21 | | | | 14 |
| **c7** | | | | | | 14 | 14 | |

Se pide aplicar el algoritmo A* para encontrar la ruta más corta desde la ciudad c2 hasta la ciudad c6, usando las siguientes estimaciones heurísticas de coste al objetivo (medidas en kilómetros):

| **n** | **c0** | **c1** | **c2** | **c3** | **c4** | **c5** | **c6** | **c7** |
|---|---|---|---|---|---|---|---|---|
| **h(n)** | 11 | 2 | 12 | 14 | 13 | 8 | 0 | 10 |

Más concretamente, se pide indicar para cada iteración del algoritmo:

a) el nodo seleccionado para expansión, y su valor de f(n),

b) los sucesores generados, indicando si alguno provoca la redirección de punteros o es descartado,

c) los valores de g(n), h(n) y f(n) de dichos sucesores.

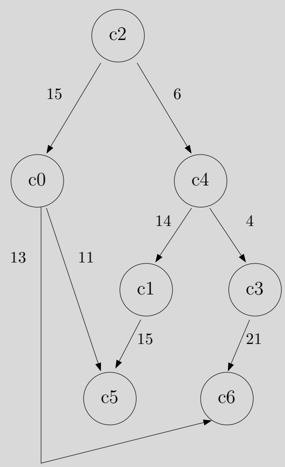
d) Además, deberá proporcionarse el árbol de búsqueda resultante en el momento de finalizar la búsqueda, indicando claramente cuáles están abiertos y cuáles cerrados, y sus valores de f(n).

**Solución**

a) b) y c) La tabla con todos los datos solicitados es la siguiente:

| Nodo seleccionado | Sucesor g+h = f | Orden en que se han cerrado |
|---|---|---|
| c2 | c0 15+11=26<br>c4 6+13=19 | (4)<br>(1) |
| c4 | c1 20+2=22<br>c3 10+14=24 | (2)<br>(3) |
| c1 | c5 35+8=43 | Puntero redirigido a c0 |
| c3 | c6 31+0=31 | Puntero redirigido a c0 |
| c0 | c5 26+8=34<br>c6 28+0=28 | redirigido desde c0<br>(5) redirigido desde c0 |
| c6 | OBJETIVO | COSTE 28 |

d) El árbol resultante de búsqueda cuando se finaliza es el siguiente:




La solución con coste = 28 es [c2, c0, c6]

**Ejercicio 1.20** Consideremos una red de carreteras entre varias ciudades. La siguiente tabla muestra las distancias recorridas en kilómetros cuando hay una carretera que conecta dos ciudades directamente:

| | **c0** | **c1** | **c2** | **c3** | **c4** | **c5** | **c6** | **c7** |
|---|---|---|---|---|---|---|---|---|
| **c0** | | | | 15 | | 4 | | 27 |
| **c1** | | | 8 | | 24 | | | 22 |
| **c2** | | 8 | | | 17 | | 19 | |
| **c3** | 15 | | | | | 16 | 11 | |
| **c4** | | 24 | 17 | | | 8 | | |
| **c5** | 4 | | | 16 | 8 | | | |
| **c6** | | | 19 | 11 | | | | 32 |
| **c7** | 27 | 22 | | | | | 32 | |

Se pide aplicar el algoritmo A* para encontrar la ruta más corta desde la ciudad c2 hasta la ciudad c6, usando las siguientes estimaciones heurísticas de coste al objetivo (medidas en kilómetros):

| **n** | **c0** | **c1** | **c2** | **c3** | **c4** | **c5** | **c6** | **c7** |
|---|---|---|---|---|---|---|---|---|
| **h(n)** | 7 | 17 | 11 | 8 | 0 | 5 | 4 | 23 |

Más concretamente, se pide indicar para cada iteración del algoritmo:

a) el nodo seleccionado para expansión, y su valor de f(n),

b) los sucesores generados, indicando si alguno provoca la redirección de punteros o es descartado,

c) los valores de g(n), h(n) y f(n) de dichos sucesores.

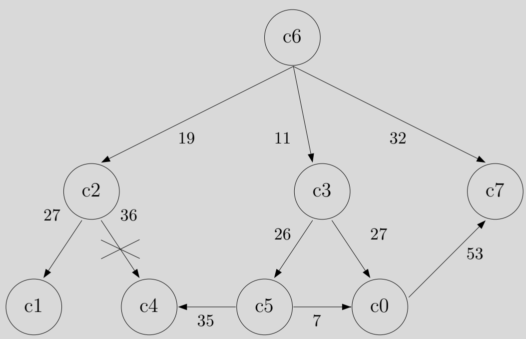
d) Además, deberá proporcionarse el árbol de búsqueda resultante en el momento de finalizar la búsqueda, indicando claramente cuáles están abiertos y cuáles cerrados, y sus valores de f(n).

**Solución**

a) b) y c) La tabla con todos los datos solicitados es la siguiente:

| Nodo seleccionado | Sucesor g+h = f | Orden en que se han cerrado |
|---|---|---|
| c6 | c2 19+11=30<br>c3 11+8=19<br>c7 32+23=55 | (2)<br>(1) |
| c3 | c0 26+7=33<br>c5 27+5=32 | (4)<br>(3) |
| c2 | c1 27+17=44<br>c4 36+0=36 | ()<br>Puntero redirigido a c5 |
| c5 | c0 31+7=38<br>c4 35+0=35 | Descartado (mejor desde c3)<br>(5) redirigido desde c5 |
| c0 | c5 30+5=35<br>c7 53+23=76 | Descartado (mejor desde c3)<br>Descartado (mejor desde c6) |
| c4 | OBJETIVO | COSTE 35 |

d) El árbol resultante de búsqueda cuando se finaliza es el siguiente:




La solución con coste = 35 es [c6, c3, c5, c4]

# Tema 2

# Juegos

Para cada uno de los siguientes juegos, debes proporcionar:

-Una representación adecuada del estado del juego.

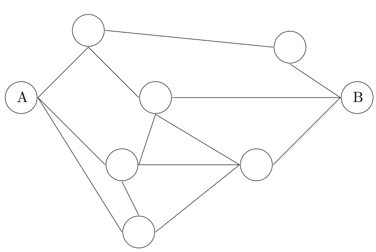
-Condición de terminación.

-Función de utilidad.

-Función de evaluación.

Debes razonar tus respuestas.

**Ejercicio 2.1** *The Shannon switching game.* El juego se desarrolla sobre un grafo finito con dos nodos especiales, A y B. Cada arista del grafo puede ser coloreada o eliminada. Los dos jugadores se llaman Conector y Separador, y alternan jugadas. En su turno, Separador elimina del grafo la arista no coloreada que quiera. Por su parte, en su turno Conector elige una arista no coloreada de las que queden y la colorea. Si Separador consigue que A y B ya no estén conectados en el grafo, gana. Si Conector consigue crear un camino coloreado desde A hasta B, gana.



Ejemplo de estado del juego: Separador ha completado 3 turnos (aristas punteadas), y Conector ha jugado 4 turnos (aristas verdes).

**Solución**

**The Shannon switching game.** El estado del juego es una matriz con N $\times$ N celdas, donde N es el número de nodos del grafo. Cada celda de la matriz puede contener un cero (si no había ninguna arista desde el principio), un uno (si hay una arista sin colorear), un dos (si hay una arista coloreada), o un tres (si había una arista, pero ha sido borrada por Separador).

El test de terminación tiene dos partes. En primer lugar, comprobamos si hay un camino desde A hasta B formado por aristas coloreadas o no coloreadas (marcadas con unos y doses en la matriz, respectivamente); si la respuesta es no, entonces Separador ha ganado. A continuación comprobamos si hay un camino desde A hasta B formado por aristas coloreadas (marcadas con doses en la matriz); si la respuesta es sí, entonces Conector ha ganado. En otro caso, el estado es no terminal.

La función de utilidad devuelve 1 si Separador ha ganado y -1 si Conector ha ganado. No es posible hacer tablas en este juego.

Consideraremos una función de evaluación de esta forma:

$$Eval(estado, turno) = Eval\_Conector(estado) - Eval\_Separador(estado)$$

$Eval\_Conector(estado)$ es el mínimo número de movimientos que Conector necesitaría para ganar el juego si el otro jugador no pudiera mover, y $Eval\_Separador(estado)$ es el mínimo número de movimientos que Separador necesitaría para ganar si el otro jugador no pudiera mover.

El cálculo de Eval_Conector empieza hallando el conjunto de nodos $A\_Conn$ que están conectados con A mediante aristas coloreadas, y el conjunto de nodos $B\_Conn$ que están conectados con B por aristas coloreadas; esto se puede hacer mediante búsqueda primero en anchura, por ejemplo. Entonces Eval_Conector(*estado*) es la mínima distancia sobre aristas no coloreadas entre un nodo de *A_Conn* y un nodo de *B_Conn*, donde cada arista no coloreada tiene coste 1. Las distancias mínimas entre todos los pares de nodos se pueden calcular mediante el algoritmo de Floyd-Warshall. Nótese que Conector gana si Eval_Conector(*estado*)=0.

Por otro lado, Eval_Separador(*estado*) es el flujo máximo del grafo formado por las aristas no borradas (marcadas con unos y doses en la matriz), donde la fuente el A y el sumidero es B. El método de Ford-Fulkerson puede usarse para calcular el flujo máximo de un grafo. Nótese que Separador gana si Eval_Separador(*estado*)=0.

**Ejercicio 2.2** *Dominono.* En un tablero N x N, los jugadores marcan alternativamente X's y O's, como en el tres en raya. Si un jugador hace una marca adyacente a una de sus anteriores marcas (creando un dominó de su símbolo), pierde.

**Solución**

**Dominono.** El estado del juego es una matriz con N $\times$ N celdas. Cada celda contiene un cero (celda vacía), una 'X' o un 'O'.

La condición de terminación consiste en comprobar si hay dos celdas 4-vecinas con 'X' (X pierde) o dos celdas 4-vecinas con 'O' (O pierde).

La función de utilidad devuelve 1 si X ha ganado, -1 si O ha ganado, y 0 si son tablas (la matriz está llena y no hay dominós).

Se puede definir una función de evaluación como sigue:

$$Evaluación(estado, turno) = SpacesX(estado) - SpacesO(estado)$$

donde SpacesX(*estado*) es el número de celdas vacías que no son 4-vecinas de ninguna X, y SpacesO(*estado*) es el número de celdas vacías que no son 4-vecinas de ninguna O. Nótese que X pierde cuando SpacesX(*estado*)=0, mientras que O pierde cuando SpacesO(*estado*)=0.

**Ejercicio 2.3 Domineering.** *Domineering* (también llamado *Stop-Gate* o *Crosscram*) es un juego que se desarrolla en una hoja de papel cuadriculado. Por ejemplo, se puede jugar sobre una matriz 6x6 o sobre un tablero de damas. Los dos jugadores tienen una colección de dominós que colocan sobre el retículo por turnos, cubriendo casillas. Uno de los jugadores, Left, coloca los dominós verticalmente, mientras que el otro, Right, los coloca horizontalmente. El primer jugador que no puede mover pierde.

**Solución**

**Domineering.** El estado del juego es una matriz con N × N celdas. Cada celda puede contener un cero (celda vacía) o una 'X' (celda ocupada).

El test de terminación depende de qué jugador tiene que mover. Si Left tiene que mover, tenemos que comprobar si no hay ningún par de celdas vacías verticalmente adyacentes. Si Right tiene que mover, tenemos que comprobar si no hay ningún par de celdas vacías horizontalmente adyacentes.

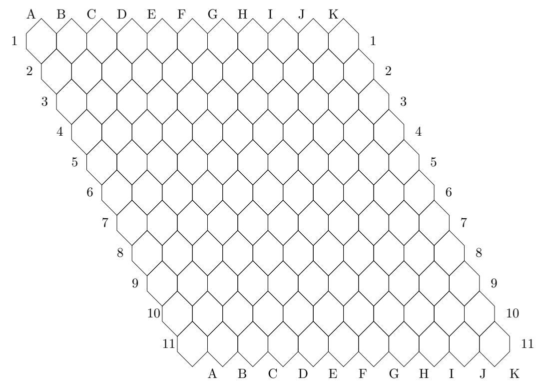
La función de utilidad devuelve 1 si Left ha ganado y -1 si Right ha ganado. No se pueden hacer tablas en este juego.

Se puede plantear una función de evaluación como la siguiente:

$$Evaluación(estado, turno) =$$
$$DoubleSpacesLeft(estado) - DoubleSpacesRight(estado)$$

donde DoubleSpacesLeft(*estado*) es el número de pares de celdas vacías verticalmente adyacentes, y DoubleSpacesRight(*estado*) es el número de pares de celdas vacías horizontalmente adyacentes. Nótes que Left pierde cuando DoubleSpacesLeft(*estado*)=0, mientras que Right pierde cuando DoubleSpacesRight(*estado*)=0.

**Ejercicio 2.4 Chameleon.** Se juega sobre un rombo hecho de hexágonos, con piezas de dos colores. Alice intenta crear una cadena monocromática que enlace las esquinas superior e inferior del tablero, mientras que Bob intenta enlazar las esquinas izquierda y derecha. Los jugadores pueden colocar piezas del color que quieran, una pieza por turno. Si un movimiento da lugar a caminos para los dos jugadores, entonces el jugador que acaba de mover gana.




Juego del camaleón.

**Solución**

**Chameleon.** El estado del juego es una matriz del tamaño del tablero. Para el tablero de ejemplo mostrado en la figura, necesitamos una matriz cuadrada 11x11. Nótese que las esquinas del tablero se corresponden con las esquinas de la matriz, así que tienes que rotar el tablero 45º en sentido opuesto a las agujas del reloj para obtener la configuración de la matriz. En la figura, las letras A, ..., K se corresponden con las filas de la matriz y los números 1, ..., 11 se corresponden con las columnas de la matriz. Las celdas de la matriz pueden contener un cero (celda vacía), un uno (pieza roja) o un dos (pieza azul).

El test de terminación tiene tres partes. En primer lugar, comprobamos si hay un camino desde la esquina superior hasta la esquina inferior formado por piezas del mismo color; si la respuesta es sí, entonces Alice ha ganado. A continuación comprobamos si hay un camino desde la esquina izquierda hasta la esquina derecha formado por piezas del mismo color; si la respuesta es sí, entonces Bob ha ganado. En tercer lugar, si las condiciones anteriores no se verifican y el tablero está lleno, entonces son tablas. En otro caso, el estado no es terminal.

La función de utilidad devuelve 1 si 1 ha ganado, -1 si Bob ha ganado, y 0 si son tablas (el tablero está lleno y nadie ha ganado).

Consideraremos una función de evaluación de esta forma:

$$Evaluación(estado, turno) = Eval\_Bob(estado) - Eval\_Alice(estado)$$

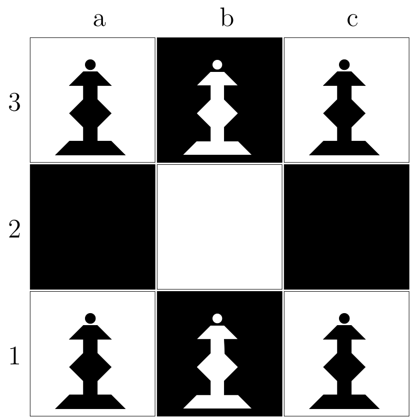
donde Eval_Bob(*estado*) es el mínimo número de movimientos que Bob necesitaría para ganar si el otro jugador no pudiera mover, y Eval_Alice(*estado*) es el mínimo número de movimientos que Alice necesitaría para ganar si el otro jugador no pudiera mover.

El cálculo de Eval_Bob empieza hallando para cada color el conjunto de hexágonos *Left_Conn(color)* que están conectados a la esquina izquierda mediante hexágonos de ese color, y el conjunto de nodos *Right_Conn(color)* que están conectados a la esquina derecha mediante hexágonos de ese color; esto se puede hacer mediante búsqueda primero en anchura, por ejemplo. Definimos una función auxiliar Aux_Bob(*estado,color*) que devuelve la mínima distancia sobre hexágonos vacíos entre un hexágono de *Left_Conn(color)* y un hexágono en *Right_Conn(color)*, donde cada hexágono tiene coste 1. Las mínimas distancias entre cada par de nodos se pueden encontrar mediante el algoritmo de Floyd-Warshall. Entonces Eval_Bob(*estado*) es el mínimo de Aux_Bob(*estado,color*) para todos los posibles valores de *color*. Nótese que Bob gana si Eval_ Bob(*estado*)=0.

El cálculo de Eval_Alice es análogo, pero usa las esquinas superior e inferior. Nótese la similitud con el juego *Shannon's switching game*.

**Ejercicio 2.5 Hexapawn.** Este juego se desarrolla sobre un tablero rectangular de tamaño variable, por ejemplo 3x3 o bien sobre un tablero de ajedrez. Dado un tablero de tamaño NxM, cada jugador empieza con M peones, uno en cada casilla de la fila más cercana al jugador. El objetivo de cada jugador es llevar a uno de sus peones al otro lado del tablero o bien impedir que el otro jugador pueda mover.

Como en el ajedrez, cada peón puede moverse de dos maneras: puede moverse una casilla hacia delante, o puede comer un peón enemigo que esté a una casilla en diagonal por delante de él. Un peón no se puede mover hacia delante si hay un peón en la siguiente casilla. Al contrario que en el ajedrez, los peones no pueden avanzar dos casillas en su primer movimiento. Un jugador pierde si no tiene movimientos legales que hacer o si el contrincante alcanza el final del tablero con un peón.




Hexapawn. Estado inicial para el juego sobre un tablero 3x3.

**Solución**

**Hexapawn.** El estado del juego se almacena como una matriz de tamaño N×M, donde cada celda contiene un cero (no hay peón), uno (peón blanco) o dos (peón negro).

El test de terminación consiste en comprobar si un peón ha llegado al lado contrario del tablero, o bien uno de los jugadores no puede mover ningún peón.

La función de utilidad devuelve 1 si ganan las blancas y -1 si ganan las negras. No hay posibilidad de tablas en este juego.

Consideraremos una función de evaluación como esta:

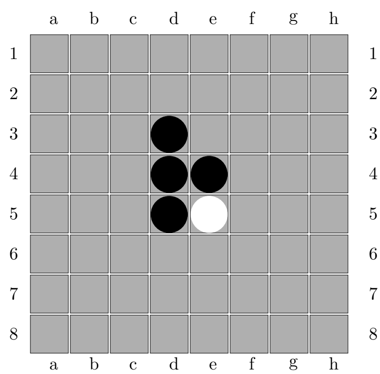
$$\begin{gathered}
Evaluación(estado, turno) = \\
ValorMaterialBlancas(estado) - ValorMaterialNegras(estado) + \\
ValorPosicionalBlancas(estado) - ValorPosicionalNegras(estado)
\end{gathered}$$

El valor material de un estado para un jugador es un múltiplo del número de peones que tiene, por ejemplo 100 puntos por peón. El valor posicional de un estado para un jugador le da bonificaciones por los peones bien posicionados, y penalizaciones por los peones mal posicionados. Por ejemplo, podemos sumar 50 por cada fila que se avanza, y podemos quitar 30 puntos por peón bloqueado (un peón bloqueado es aquel que no puede avanzar). El número concreto de puntos que dar o que quitar dependerá del tamaño del tablero.

**Ejercicio 2.6 Reversi.** Reversi es un juego de mesa para dos jugadores (blancas y negras), que se juega en un tablero de 8x8 casillas. Hay 64 piezas idénticas llamadas discos, que son blancos por un lado y negros por el otro. Los jugadores mueven por turnos colocando un disco sobre el tablero con su color hacia arriba. Durante un movimiento, todos los discos del contrincante que están en una línea recta y están rodeados por el disco que se acaba de poner y otro disco del color del jugador que mueve se cambian al color del jugador que mueve (captura). Después de que se hayan colocado cuatro discos sobre el tablero, solo están permitidas las jugadas que capturan al menos un disco. El objetivo del juego es tener la mayoría de discos mostrando tu color cuando se rellena el último cuadrado en el que se puede jugar.

a) Supongamos que deseamos modelar el problema de jugar al Reversi mediante una búsqueda alfa-beta. Definir una función de evaluación para las hojas del árbol de búsqueda del jugador de las fichas blancas.

b) Consideremos el nodo del árbol del juego asociado al siguiente estado del juego, donde le toca jugar a las blancas. ¿Cuántos hijos tiene? Explica tu respuesta.




Reversi. Posición a evaluar.

**Solución**

**Reversi.**

a) Una función de evaluación ingenua sería como sigue:

$$EvaluaciónIngenua(estado) =$$
$$DiscosBlancos(estado) - DiscosNegros(estado)$$

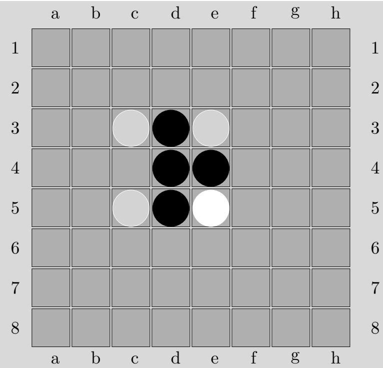
Esto es, el número de discos blancos menos el número de discos negros. Esta función no da buen resultado porque en una sola jugada pueden cambiar de color muchos discos.

Una mejor elección sería la siguiente:

$$Evaluación(estado) = Jugadas(estado) + \alpha \cdot$$
$$JugadaPotenciales(estado) + \beta \cdot DifEsquinas(estado)$$

donde *Jugadas*(*estado*) es el número de jugadas disponibles para el jugador actual, *JugadasPotenciales*(*estado*) es el número de discos del oponente que son adyacentes a una casilla vacía, *DifEsquinas*(*estado*) es la diferencia entre el número de casillas de esquina ocupadas por el jugador actual y el número de casillas de esquina ocupadas por el oponente, y $\alpha$, $\beta$ son constantes positivas que deben determinarse. Los dos primeros términos de la función de evaluación se incluyen porque la movilidad es muy importante en este juego. Si no tienes muchas opciones, es muy probable que te veas forzado a hacer una jugada mala. Finalmente, el tercer término de la función de evaluación modela el hecho de que por lo general es bueno poner un disco en una esquina, ya que los discos de las esquinas no pueden cambiar de color. Una posible mejora sería añadir un cuarto término para modelar la situación en los bordes del tablero.

b) El nodo tiene tres hijos, que se corresponden a las posibles jugadas de las blancas. En la siguiente figura se indican esas tres jugadas con discos translúcidos:



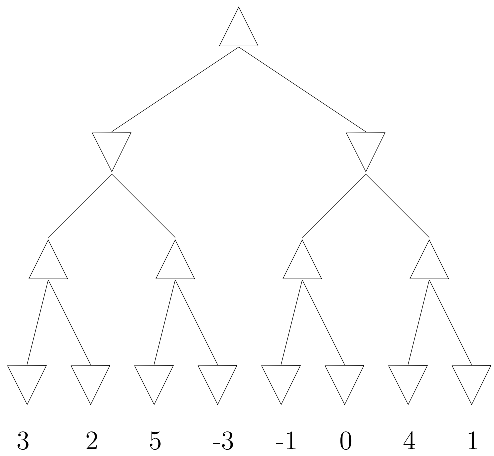

Reversi. Posibles jugadas de las blancas.

**Ejercicio 2.7** Considérese el árbol de juego de la figura, correspondiente a una exploración a profundidad 3 desde la posición actual. Los valores de la función de evaluación estática para cada uno de los nodos hoja son los indicados. Se pide:

a) Aplicar el algoritmo MINIMAX para calcular la valoración de cada nodo, destacando el valor del nodo raíz.

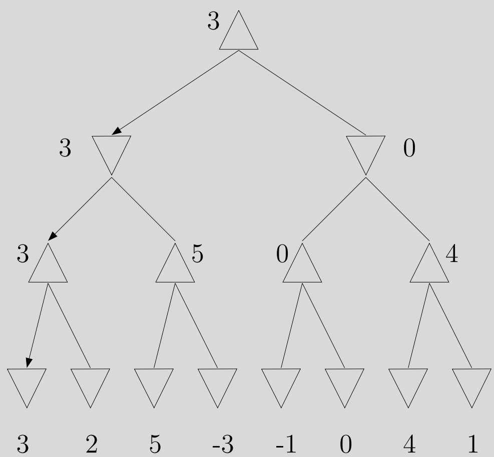
b) Aplicar el algoritmo ALFA-BETA para calcular el valor minimax de la raíz, suponiendo que las ramas del árbol se exploran ordenadamente de izquierda a derecha. Indicar claramente los nodos explorados, los valores de alfa/beta calculados para ellos, las ramas podadas, el valor calculado para la raíz, y la jugada inmediata elegida.

c) Explica si es posible que, en ciertos casos, los algoritmos MINIMAX y ALFA-BETA puedan proporcionar valores distintos para el nodo raíz y, en caso afirmativo, en qué casos.

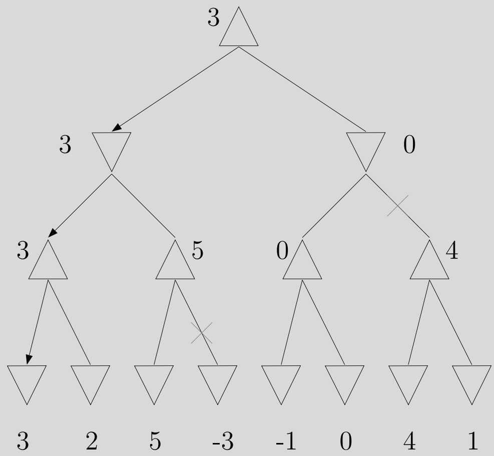



**Solución**

a) Aplicación del algoritmo minimax

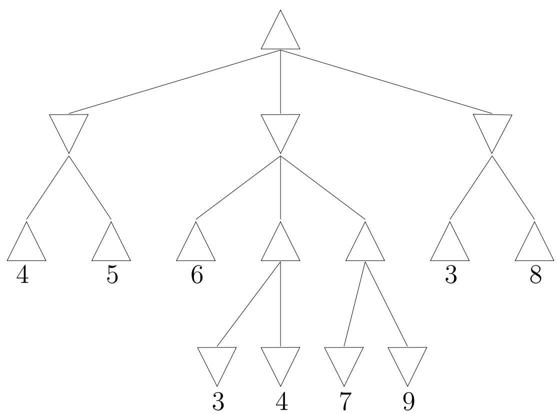



b) Ejecución del algoritmo ALFA-BETA




c) No es posible. Ambos algoritmos devuelven siempre el mismo valor para el nodo raíz.

**Ejercicio 2.8** Consideremos árbol MAX-MIN de la siguiente figura,




Se pide:

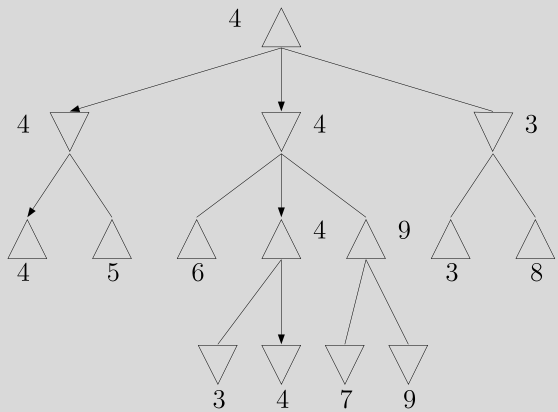
a) Aplicar el algoritmo minimax

b) Aplicar el algoritmo poda alfa-beta, suponiendo que las ramas del árbol se exploran de izquierda a derecha.

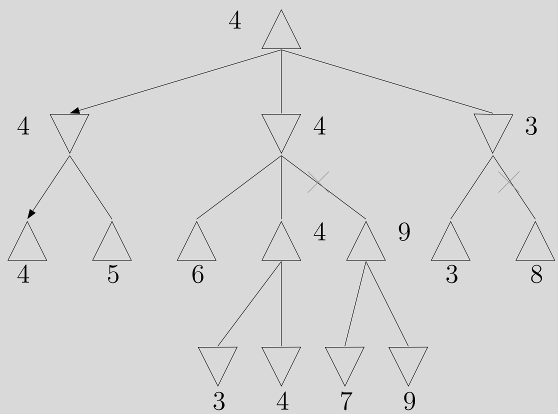
c) Explique en cada caso las acciones resultantes al aplicar ambos algoritmos

**Solución**

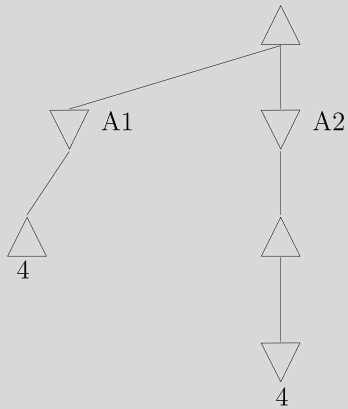
a) Aplicación del algoritmo minimax.




b) Ejecución del algoritmo de la poda alfa-beta de izquierda a derecha.



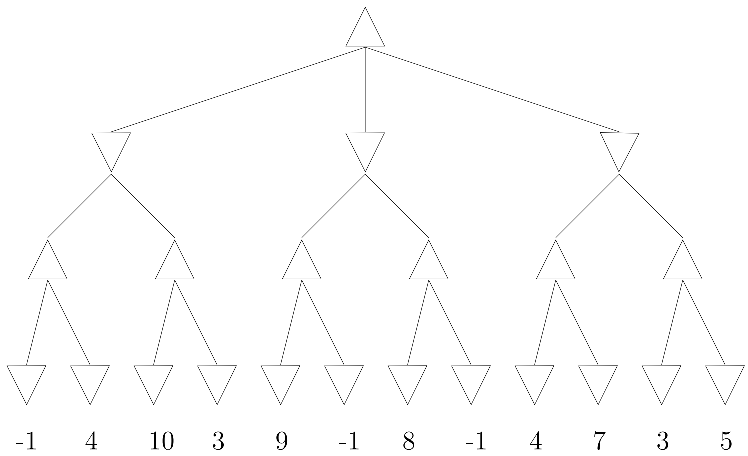
c) En el caso del Minimax según interpretación del algoritmo del libro cabe la posibilidad de un empate, ambos con el valor 4, pero ocasionaría dos movimientos distintos.




Por lo que MAX aplicando el algoritmo MAX-MIN en su primera decisión puede realizar las acciones A1 o A2.

Ocurre lo mismo en el alfa-beta, en el caso de la aplicación web solamente devuelve como solución el nodo A1. La interpretación es que en caso de empate, me quedo con el primer movimiento, MAX estricto.

**Ejercicio 2.9** Dado el siguiente árbol de búsqueda:

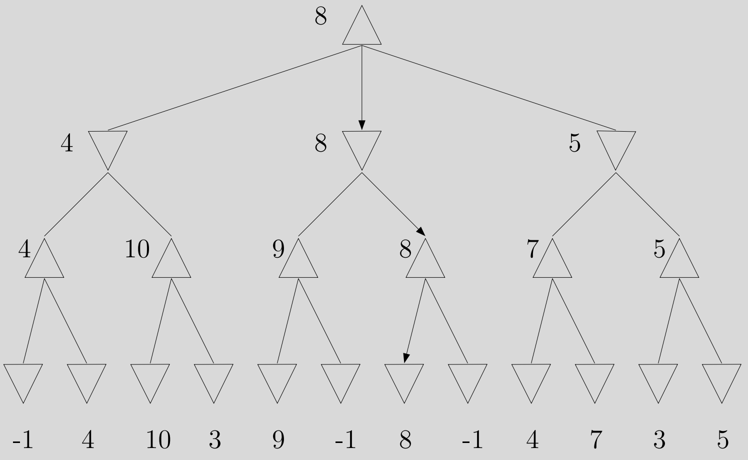



a) Aplicar el algoritmo MIN-MAX

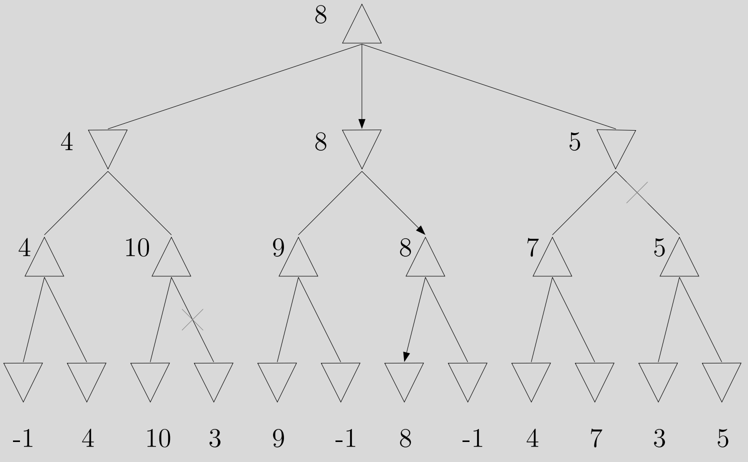
b) Aplica el algoritmo de poda $\alpha - \beta$ generando los nodos hoja de izquierda a derecha y viceversa (de derecha a izquierda) ¿En qué caso se exploran menos nodos?

**Solución**

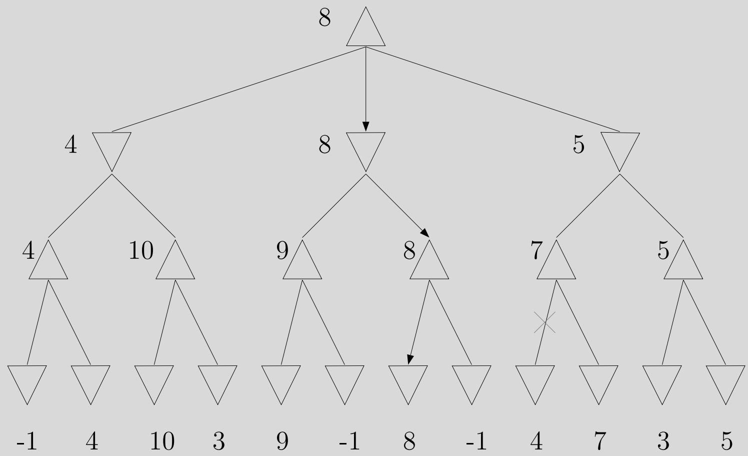
a) Minimax




b) Poda $\alpha - \beta$ de izquierda a derecha

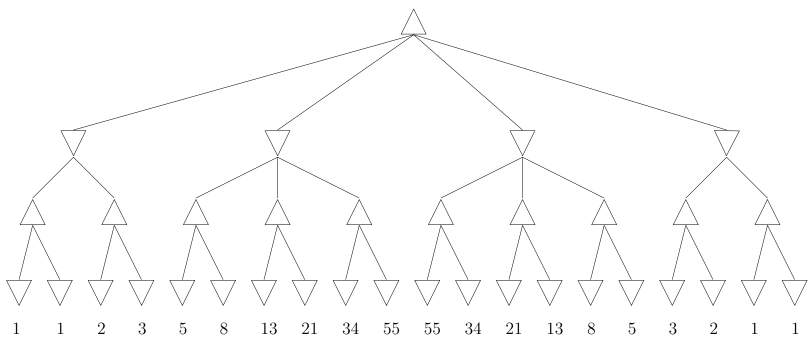



Poda $\alpha - \beta$ de derecha a izquierda




Con lo que el mejor orden de evaluación es de izquierda a derecha

**Ejercicio 2.10** Dado el siguiente árbol de búsqueda:

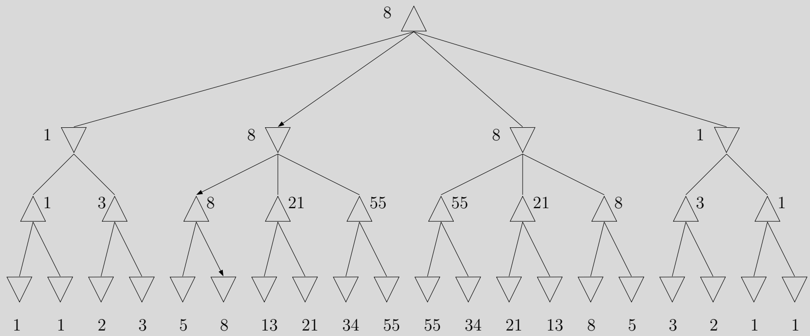



a) Aplicar el algoritmo MIN-MAX

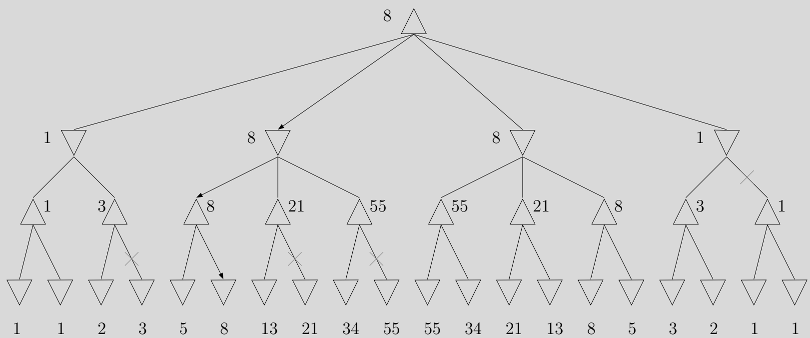
b) Aplica el algoritmo de poda $\alpha - \beta$ generando los nodos hoja de izquierda a derecha y viceversa (de derecha a izquierda) ¿En qué caso se exploran menos nodos?

**Solución**

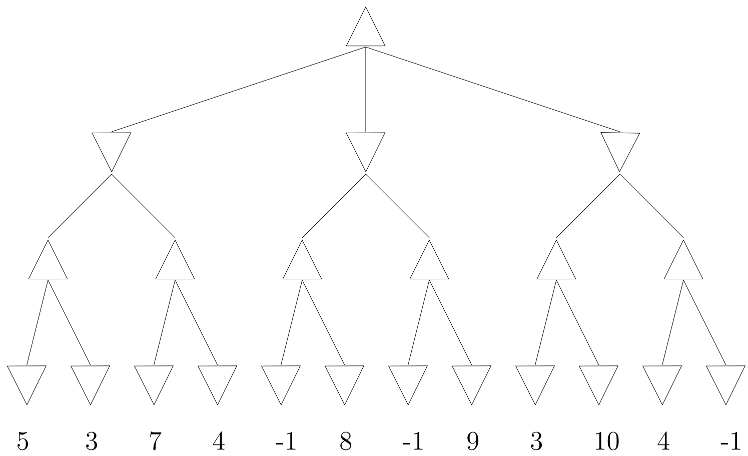
a) Minimax




b) Solución de izquierda a derecha, el otro caso es igual.



**Ejercicio 2.11** Dado el siguiente árbol de búsqueda:

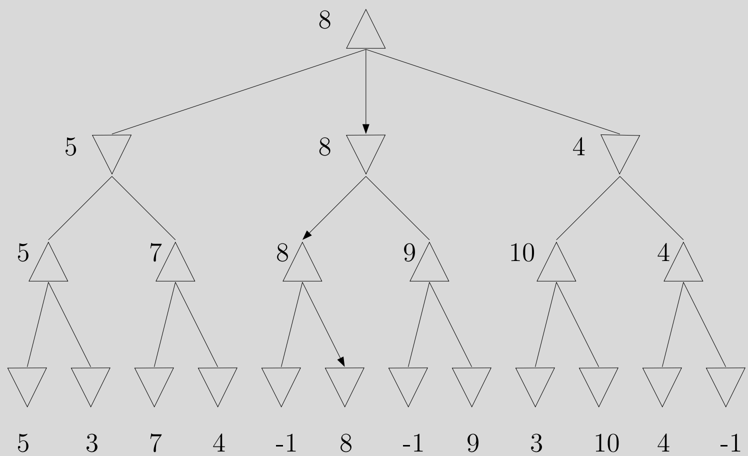



a) Aplicar el algoritmo MIN-MAX

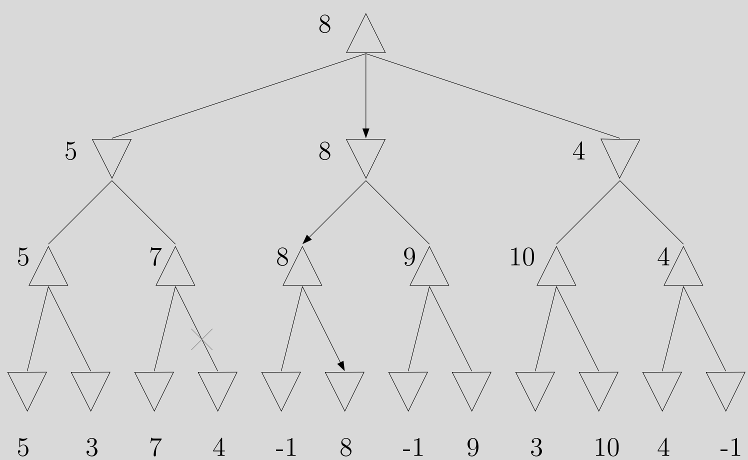
b) Aplica el algoritmo de poda $\alpha - \beta$ generando los nodos hoja de izquierda a derecha y viceversa (de derecha a izquierda) ¿En qué caso se exploran menos nodos?

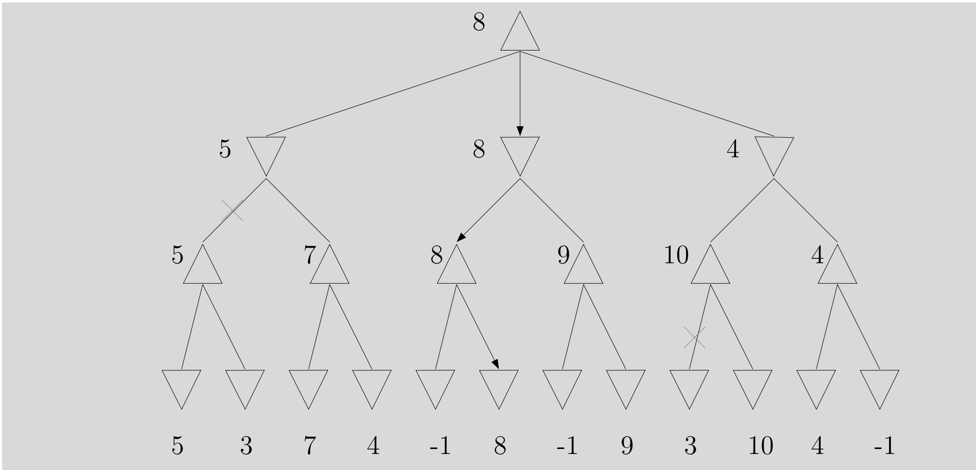
**Solución**

a) Minimax

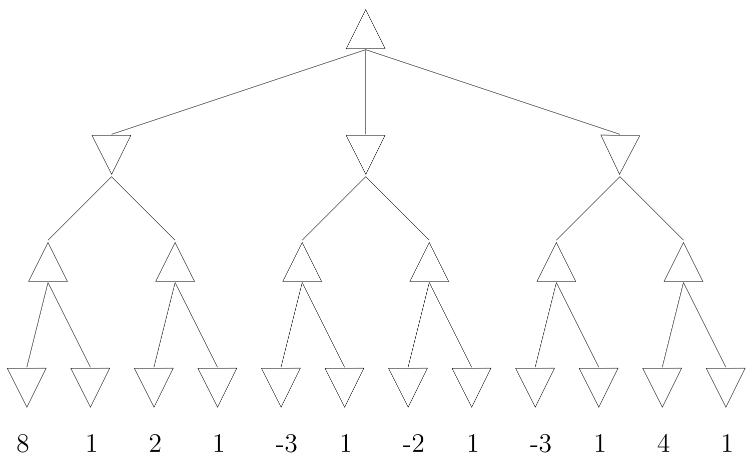



b) Poda $\alpha - \beta$






**Ejercicio 2.12** Dado el siguiente árbol de búsqueda:

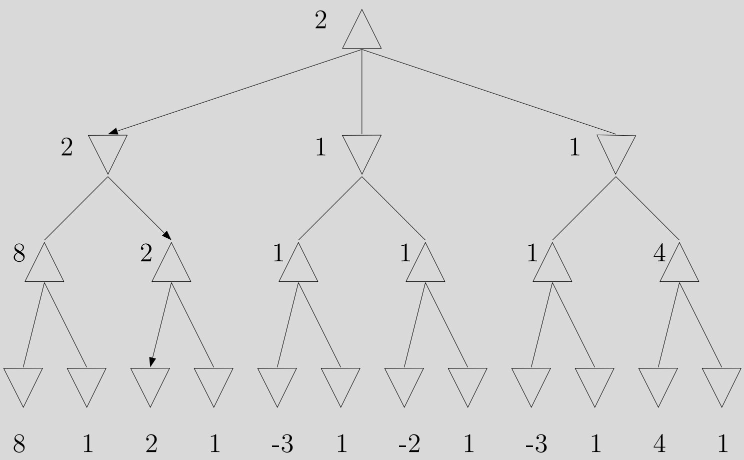



a) Aplicar el algoritmo MIN-MAX

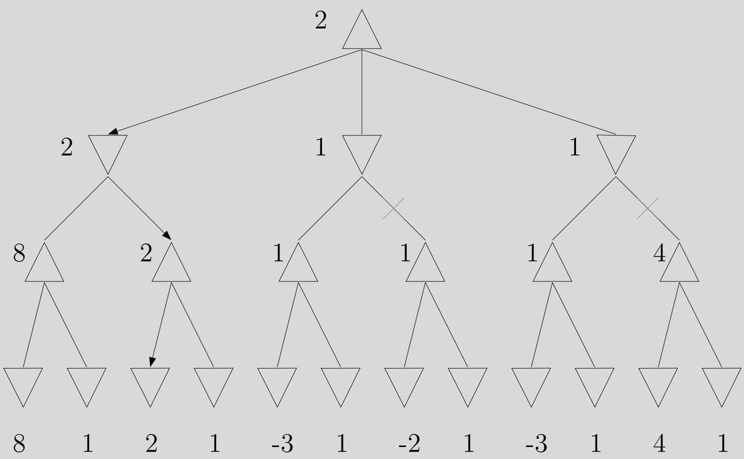
b) Aplica el algoritmo de la poda $\alpha - \beta$ generando los nodos hoja de izquierda a derecha y viceversa (de derecha a izquierda) ¿En qué caso se exploran menos nodos?

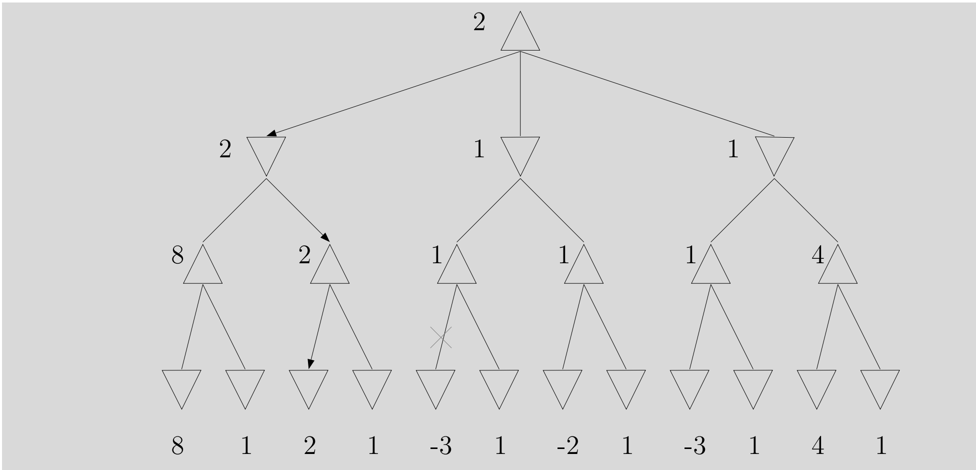
**Solución**

a) Minimax

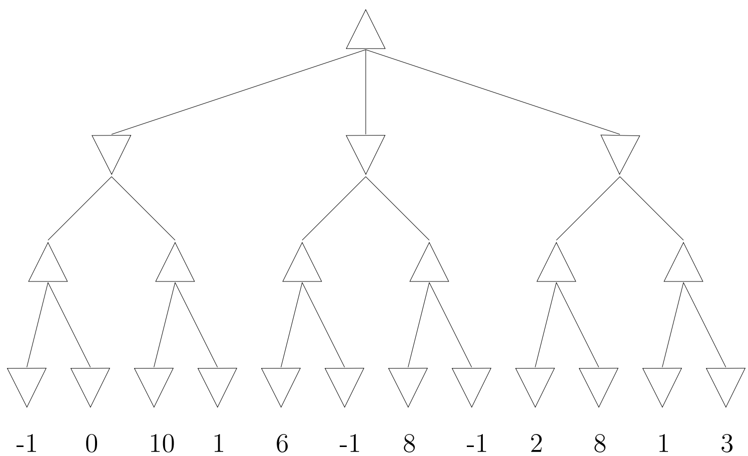



b) Poda $\alpha - \beta$






**Ejercicio 2.13** Dado el siguiente árbol de búsqueda:

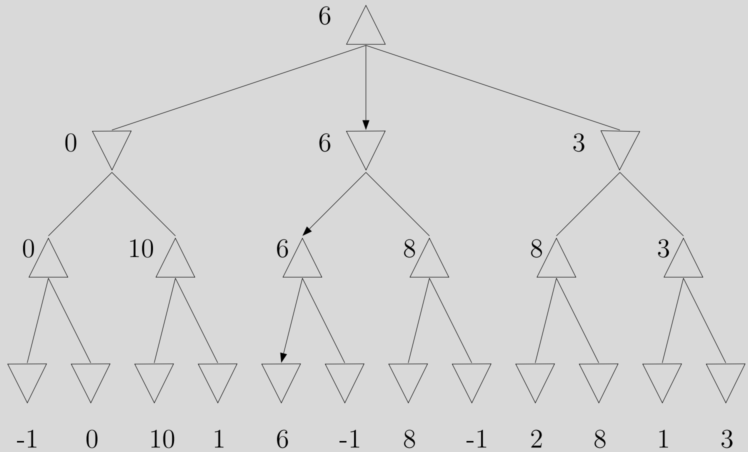



a) Aplicar el algoritmo MIN-MAX

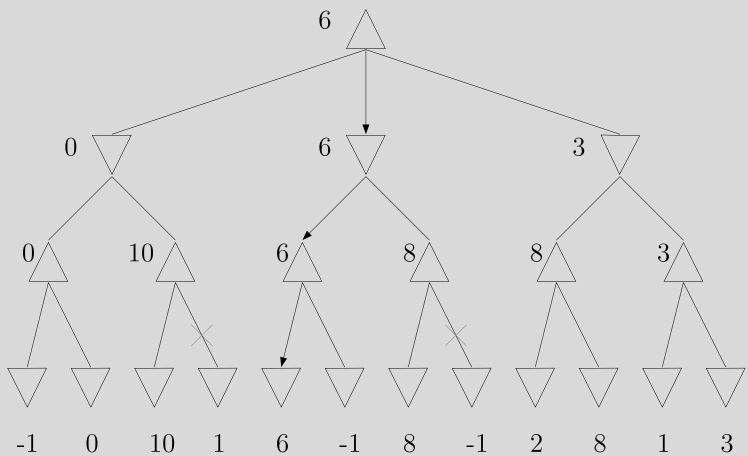
b) Aplica el algoritmo de la poda $\alpha - \beta$ generando los nodos hoja de izquierda a derecha y viceversa (de derecha a izquierda) ¿En qué caso se exploran menos nodos?

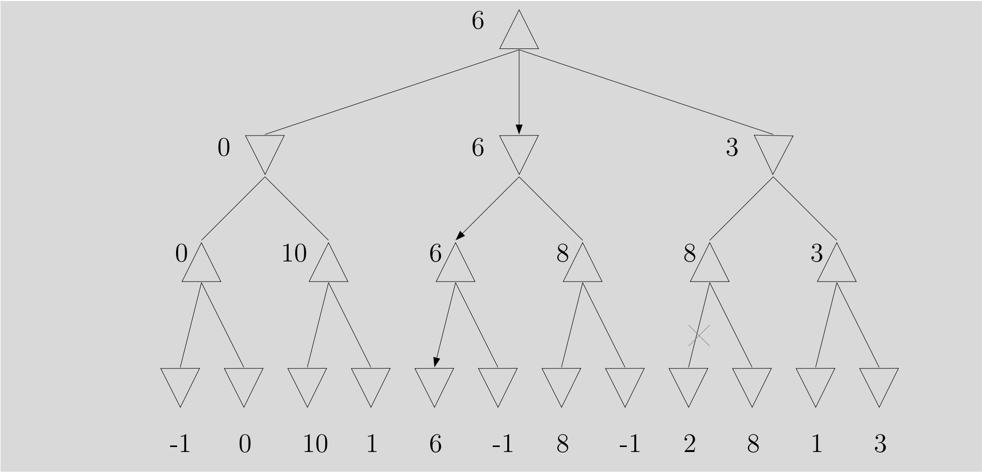
**Solución**

a) Minimax




b) Poda $\alpha - \beta$




6
0
6
3
0
10
6
8
8
3
-1 0 10 1 6 -1 8 -1 2 8 1 3

# Tema 3

# Satisfacción de Restricciones

Modela los siguientes problemas como problemas de satisfacción de restricciones (constraint satisfaction problems, CSPs) dando:

-Variables que utilizar y sus significados.

-Dominio de cada variable.

-Restricciones que deben satisfacerse y sus tipos (unaria, binaria o global).

**Ejercicio 3.1 (Reglas de Golomb).** Este problema tiene varias aplicaciones prácticas, por ejemplo a la ubicación de sensores en radioastronomía y a la cristalografía por rayos X.

Tenemos una regla de longitud L unidades. Podemos colocar marcas a lo largo de la regla en cualquier unidad. Por ejemplo, si L = 8, podemos colocar una marca en cualquiera de las posiciones $0, 1, 2, 3, 4, 5, 6, 7, 8$. Pero no podemos colocar una marca en la posición 1,5.

Queremos colocar M marcas $(m_1, m_2, \ldots, m_M)$ en la regla, de tal manera que todas las $\frac{M(M-1)}{2}$ diferencias mi–mj sean distintas.

**Solución**

**Reglas de Golomb**

*Variables:* $m_1$, $m_2$, ..., $m_M$ , una variable para cada marca.

*Dominios:*

Cada variable tiene el dominio {0, 1, ..., L-1}. Si asignamos, por ejemplo, $m_1$=4, esto quiere decir que colocamos la marca 1 en la posición 4 sobre la regla.

*Restricciones:*

$C_1$. Todas las marcas deben hacerse en posiciones distintas:

$$\textit{AllDiff}\,(m_1,\, m_2,\, ...,\, m_M)$$

$C_2$. Hay una restricción para cada colección de 4 variables $\{m_i, m_j, m_k, m_n\}$ tales que forman dos pares distintos: $(m_i,m_j)\neq(m_k,m_n)$. La restricción es como sigue:

$$|m_i\text{-}m_j|\neq|m_k\text{-}m_n|$$

Todas las restricciones de este problema son globales.

**Ejercicio 3.2 Criptoaritmética.** Expresa el siguiente problema criptoaritmético como un CSP.

SEND + MORE = MONEY

**Solución**

**Problema criptoaritmético**

*Variables:*

$S$, $E$, $N$, $D$, $M$, $O$, $R$, $Y$ (una variable por letra), y $C_{10}$, $C_{100}$, $C_{1000}$, $C_{10000}$ (los dígitos de acarreo).

*Dominios:*

$S, E, N, D, M, O, R, Y \in \{0, \ldots, 9\}$

$C_{10},\ C_{100},\ C_{1000},\ C_{10000} \in \{0,\ 1\}$

*Restricciones:*

C$_1$. $D+E=10 \cdot C_{10}+Y$

C$_2$. $C_{10}+N+R=10 \cdot C_{100}+E$

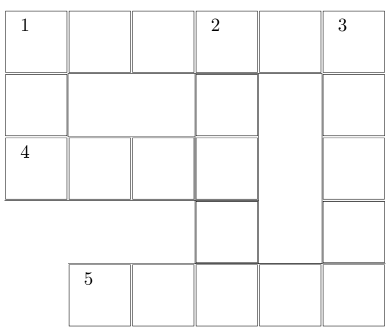
C$_3$. $C_{100}+E+O=10 \cdot C_{1000}+N$

C$_4$. $C_{1000}+S+M=10 \cdot C_{10000}+O$

C$_5$. $C_{10000}=M$

Las restricciones C$_1$ a C$_4$ son globales, mientras que C$_5$ es binaria.

**Ejercicio 3.3 Crucigrama.** Considera el crucigrama mostrado más abajo.




Crucigrama.

Debe resolverse con 6 palabras (1-horizontal, 1-vertical, 2-vertical, 3-vertical, 4- horizontal y 5-horizontal).

**Solución**

**Crucigrama**

*Variables:*

*UnoHorizontal*, *UnoVertical*, *DosVertical*, *TresVertical*, *CuatroHorizontal* y *CincoHorizontal* (una variable para cada palabra a escribir en el crucigrama).

*Dominios:*

Sean $W_3$, $W_4$, $W_5$, $W_6$ los conjuntos de palabras admisibles de 3, 4, 5 y 6 letras, respectivamente. Podemos suponer que estos conjuntos son conocidos (son listas de palabras). Tendremos:

$UnoHorizontal \in W_6$, $UnoVertical \in W_3$, $DosVertical \in W_5$, $TresVertical \in W_5$, $CuatroHorizontal \in W_4$, $CincoHorizontal \in W_5$.

*Restricciones:*

Sea $charAt(x,y)$ una función que devuelve el $y$-ésimo carácter de la cadena $x$. Hay una restricción por cada cuadrado en el que se cruzan dos palabras:

$C_1$. $charAt(UnoHorizontal,1)=charAt(UnoVertical,1)$

$C_2$. $charAt(UnoHorizontal,4)=charAt(DosVertical,1)$

$C_3$. $charAt(UnoHorizontal,6)=charAt(TresVertical,1)$

$C_4$. $charAt(UnoHorizontal,6)=charAt(TresVertical,1)$

$C_5$. $charAt(UnoVertical,3)=charAt(CuatroHorizontal,1)$

$C_6$. $charAt(DosVertical,3)=charAt(CuatroHorizontal,4)$

$C_7$. $charAt(DosVertical,5)=charAt(CincoHorizontal,3)$

$C_8$. $charAt(TresVertical,5)=charAt(CincoHorizontal,5)$

Todas las restricciones de este problema son binarias.

**Ejercicio 3.4 Diseño de un horario.** El director de un instituto tiene que diseñar el horario de una clase. Para simplificar vamos a suponer que el mismo horario se aplica a todos los días. Hay seis horas, que deben rellenarse con estas asignaturas: Lengua, Inglés, Matemáticas, Física, Filosofía y Biología. Hay algunas restricciones:

a) Las horas de Lengua e Inglés no pueden estar adyacentes, para que los alumnos no mezclen ambos idiomas.

b) Las Matemáticas no deben enseñarse a primera ni a última hora, para que los estudiantes estén lo suficientemente atentos.

c) La Física debe ir después de las Matemáticas.

d) La Filosofía sólo se puede asignar a la segunda o a la cuarta hora.

**Solución**

**Diseño de un horario**

*Variables:*

*Lengua*, *Inglés*, *Matemáticas*, *Física*, *Filosofía* y *Biología* (una variable por asignatura).

*Dominios:*

*Lengua*, *Inglés*, *Matemáticas*, *Física*, *Filosofía*, *Biología*$\in\{1, ..., 6\}$, donde los números representan las horas del horario.

*Restricciones:*

Dos asignaturas no se pueden enseñar al mismo tiempo a la misma clase, así que tendremos:

*AllDiff*(*Lengua*, *Inglés*, *Matemáticas*, *Física*, *Filosofía*, *Biología*)

También tendremos las restricciones correspondientes al enunciado del problema:

a) $|Lengua\text{-}Inglés|>1$

b) $Matemáticas\notin\{1,6\}$

c) *Física>Matemáticas*

d) *Filosofía*$\in\{2,4\}$

La restricción *AllDiff* es global; las restricciones a) y c) son binarias, y las restricciones b) y d) son unarias.

**Ejercicio 3.5 Planificación de un proyecto.** Un proyecto de ingeniería se compone de varias tareas. Cada una de ellas tiene su propia duración y sus prerrequisitos, es decir, otras tareas que deben terminarse antes de que la tarea empiece. Esta información viene dada por la siguiente tabla:

| Tarea | Duración(meses) | Prerrequisitos |
|---|---|---|
| A | 3 | - |
| B | 4 | A |
| C | 2 | A |
| D | 5 | B, C |
| E | 2 | D |
| F | 2 | D, E |

Queremos saber si hay una forma de terminar el proyecto en L meses cumpliendo con los requisitos anteriores.

**Solución**

**Planificación de un proyecto**

*Variables:*

$A$, $B$, $C$, $D$, $E$ y $F$; cada una de ellas representa el instante de comienzo de cada tarea.

*Dominios:*

$A$, $B$, $C$, $D$, $E$, $F \in \{0, ..., \mathrm{L}\text{-}1\}$, donde los números representan unidades de tiempo (meses).

*Restricciones:*

Todas las tareas deben finalizar antes de que hayan transcurrido L meses:

$\mathrm{C}_1$. $(A+3 \leq \mathrm{L}) \wedge (B+4 \leq \mathrm{L}) \wedge (C+2 \leq \mathrm{L}) \wedge (D+5 \leq \mathrm{L}) \wedge (E+2 \leq \mathrm{L}) \wedge (F+2 \leq \mathrm{L})$

Los prerrequisitos de cada tarea deben completarse antes de que la tarea comience. Esto produce una restricción por cada tarea, excepto para la tarea A que no tiene prerrequisitos:

$\mathrm{C}_2$. $A+3 \leq B$

$\mathrm{C}_3$. $A+3 \leq C$

$\mathrm{C}_4$. $(B+4 \leq D) \wedge (C+2 \leq D)$

$\mathrm{C}_5$. $D+5 \leq E$

$\mathrm{C}_6$. $(D+5 \leq F) \wedge (E+2 \leq F)$

Las restricciones $\mathrm{C}_1$, $\mathrm{C}_4$ y $\mathrm{C}_6$ son globales, mientras que $\mathrm{C}_2$, $\mathrm{C}_3$ y $\mathrm{C}_5$ son binarias.

**Ejercicio 3.6 Puzzle lógico.** Hay una pastelería famosa donde acude mucha gente a comprar tartas de cumpleaños. Con los siguientes datos, establece un CSP para averiguar el nombre del cliente, el tipo de tarta, el nombre de la persona que recibirá la tarta, y el número de velas.

a) La tarta de manzana es para Jaime, pero no la ha comprado Andrea.

b) Sandra ha reservado la tarta de plátanos, pero no para la muchacha que cumplirá 15 años.

c) Mario va a regalarle una tarta a su padre Manuel.

d) La tarta Battenberg es para Violeta.

e) José no ha encargado la tarta de mantequilla con 45 velas, puesto que a Paula no le gusta la mantequilla.

f) Andrea ha comprado 5 velas menos que Mario.

g) Fernando ha comprado 32 velas, pero no para Patricia, ya que ella va a cumplir 5 años.

h) La oferta especial de la semana es la tarta de ron a mitad de precio.

**Solución**

**Puzzle lógico**

*Variables:*

Usaremos las variables $S_1$, ..., $S_5$ para los nombres de los clientes correspondientes a las cinco tartas. El tipo de la $i$-ésima tarta se representará mediante la variable textit$\mathrm{T}_i$. Los nombres de las personas a las que se les va a regalar las tartas se corresponderán con las variables textit$\mathrm{N}_i$. Por último, los números de años que las personas van a cumplir serán $Y_i$.

*Dominios:*

$S_i \in$ {Andrea, Sandra, Mario, José, Fernando}

$T_i \in$ {Manzana, Plátanos, Battenberg, Mantequilla, Ron}

$N_i \in$ {Jaime, Manuel, Violeta, Paula, Patricia}

$Y_i \in \{5, 15, 32, 40, 45\}$

*Restricciones:*

Las siguientes restricciones *AllDiff* no se especifican explícitamente en el ejercicio, pero deben darse por supuestas considerando la manera en que nos dan los datos:

*AllDiff*($S_1$,..., $S_5$) $\wedge$ *AllDiff*($T_1$,..., $T_5$) $\wedge$ *AllDiff*($N_1$,..., $N_5$) $\wedge$ *AllDiff*($Y_1$,..., $Y_5$)

A continuación escribimos las restricciones que pueden inferirse de los datos del enunciado:

a) [ ∀i∈{1,...,5}, $T_i$=Manzana ⇔ $N_i$=Jaime ] ∧
[ ∀i∈{1,...,5}, $T_i$=Manzana ⇒ $S_i$≠Andrea ]

b) [ ∀i∈{1,...,5}, $T_i$=Plátanos ⇔ $S_i$=Sandra ] ∧
[ ∀i∈{1,...,5}, $T_i$=Plátanos ⇒ $Y_i$≠15 ] ∧
[ ∀i∈{1,...,5}, $Y_i$=15 ⇒ $N_i$∈{Violeta, Paula, Patricia} ]

c) ∀i∈{1,...,5}, $S_i$=Mario ⇔ $N_i$=Manuel

d) ∀i∈{1,...,5}, $T_i$=Battenberg ⇔ $N_i$=Violeta

e) [ ∀i∈{1,...,5}, $S_i$=José ⇒ $T_i$≠Mantequilla ] ∧
[ ∀i∈{1,...,5}, $T_i$=Mantequilla ⇔ $Y_i$=45 ] ∧
[ ∀i∈{1,...,5}, $S_i$=José ⇔ $N_i$=Paula ]

f) ∀i∈{1,...,5} ∀j∈{1,...,5}, ($S_i$=Andrea) ∧ ($S_j$=Mario) ⇒ $Y_i$+5= $Y_j$

g) [ ∀i∈{1,...,5}, $S_i$=Fernando ⇔ $Y_i$=32 ] ∧
[ ∀i∈{1,...,5}, $S_i$=Fernando ⇒ $N_i$≠Patricia ] ∧
[ ∀i∈{1,...,5}, $N_i$=Patricia ⇔ $Y_i$=5 ]

h) De este apartado no se saca ninguna restricción; la frase sólo está para informarnos de que hay una tarta de ron.

Todas las restricciones de este problema son globales.

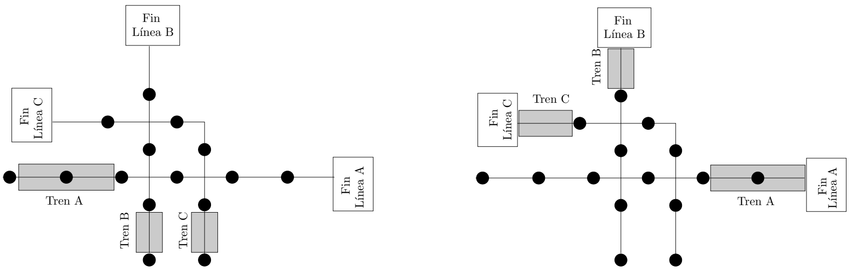
**Ejercicio 3.7 Jefe de estación.** Un jefe de estación debe decidir cuando los trenes A, B y C deben partir. Una vez que los trenes han partido, se moverán un "hueco" en su vía a la hora hasta que llegue al destino.

Cada tren puede salir a las 1, 2 o 3 de la tarde. Existen dos restricciones:

a) Todos los trenes deben salir en horas distintas y

b) Dos trenes no pueden ocupar a la vez los cruces de vías hasta que no pase una hora.

Nótese que el tren A ocupa dos huecos. Además la restricción de colisión es impuesta solamente a la conclusión de cada hora, ya que consideramos en este problema el tiempo como variable discreta.




Jefe de estación. Estado inicial y Estado objetivo.

Describe el problema de satisfacción de restricciones, que deberá indicar al encargado cuándo cada tren debe partir.

**Solución**

**Jefe de estación:**

*Variables y dominios:*

Tendremos una variable por tren, que almacenará su hora de salida: $A$, $B$, $C$ $\in$\{1, 2, 3\}.

*Restricciones:*

1. Cada tren debe salir a una hora distinta:

$$AllDiff(A,B,C) \Leftrightarrow (A \neq B) \wedge (\mathrm{B} \neq \mathrm{C}) \wedge (\mathrm{A} \neq \mathrm{C})$$

2. Restricción de las intersecciones:

$$(A{+}1 \neq B) \wedge (A{+}1 \neq C) \wedge (A{+}2 \neq C) \wedge (B \neq C{+}1)$$

**Ejercicio 3.8 Mini-sudoku.** Mini-Sudoku es una versión del popular juego Sudoku. Usa un tablero con 4x4 regiones de 2x2 celdas cada una.

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 3 | 4 | 2 | 1 |
| 4 | 3 | 1 | 2 |
| 2 | 1 | 4 | 3 |

| | | | |
|---|---|---|---|
| 1 | 2 | | 2 |
| | | | |
| 4 | | | |
| | 4 | | |

Mini-Sudoku válido vs Mini-Sudoku no válido.

Cada celda contiene un dígito del 1 al 4. El objetivo es encontrar una manera de rellenar el puzzle que satisfaga las siguientes tres restricciones:

- Cada dígito solo puede aparecer una vez en cada fila.
- Cada dígito solo puede aparecer una vez en cada columna.
- Cada dígito solo puede aparecer una vez en cada región.

Modela la situación como un problema de satisfacción de restricciones, es decir, especifica las variables, sus dominios y las restricciones.

**Solución**

**Mini-Sudoku**

Formalmente, podemos ver el Mini-Sudoku como un problema de satisfacción de restricciones.

*Variables:*

Sea $S_{ij}$ el valor de la celda $(i,j)$, donde $i$ y $j$ varían dentro de {1, 2, 3, 4}. Nótese que estos son índices sobre el tablero, que no tienen que ver con los dígitos que usamos para rellenar las celdas.

*Dominios:*

Cada variable tiene como dominio de valores {1, 2, 3, 4}.

**Ejercicio 3.9 El puzzle de Einstein.** Cinco casas consecutivas tienen colores diferentes y son habitadas por hombres de diferentes nacionalidades. Cada uno tiene un animal diferente, una bebida preferida y fuma una marca determinada. Además, se sabe que:

1. *El noruego vive en la primera casa*
2. *La casa de al lado del noruego es azul*
3. *El habitante de la tercera casa bebe leche*
4. *El inglés vive en la casa roja*
5. *El habitante de la casa verde bebe café*
6. *El habitante de la casa amarilla fuma Kools*

7. *La casa blanca se encuentra justo después de la verde*
8. *El español tiene un perro*
9. *El ucraniano bebe té*
10. *El japonés fuma Cravens*
11. *El fumador de Old Golds tiene un caracol*
12. *El fumador de Gitanes bebe vino*
13. *Un vecino del fumador de Chesterfields tiene un reno*
14. *Un vecino del fumador de Kools tiene un caballo*

Se trata de averiguar quién tiene una cebra y quién bebe agua. Describe este problema como un problema de satisfacción de restricciones.

**Solución**

**El puzzle de Einstein**

*Variables:*

azul, roja, verde, amarilla, blanca, noruego, ingles, ucraniano, japonés, español, perro, caracol, reno, caballo, cebra, leche, café, té, vino, agua; kools, cravens, golds, gitanes, chesterfields.

*Dominios:*

Todas las variables se instancian en el dominio $\{1, 2, 3, 4, 5\}$, indicando la casa correspondiente.

*Restricciones:*

$R1$: noruego = 1 ;

$R2$: azul = noruego + 1 $\vee$ azul + 1 = noruego;

$R3$: leche = 3;

$R4$: ingles = roja ;

$R5$: verde = café;

$R6$: amarilla = kools;

$R7$: blanca = verde + 1;

$R8$: español = perro;

$R9$: ucraniano = té;

$R10$: japonés = craven;

$R11$: golds = caracol;

$R12$: gitanes = vino;

$R13$: chesterfields = reno + 1 $\vee$ chesterfields + 1 = reno;

$R14$: kools = caballo +1 $\vee$ kools + 1 = caballo;

Y, adicionalmente:

*AllDiff* (azul, roja, verde, amarilla, blanca);

*AllDiff* (noruego, ingles, ucraniano, japonés, español);

*AllDiff* (perro, caracol, reno, caballo, cebra);

*AllDiff* (leche, café, té, vino, agua);

*AllDiff* (kools, cravens, golds, gitanes, chesterfields);

**Ejercicio 3.10 Futoshiki.** Considérese el juego del Futoshiki (desigualdad). Se trata de un tablero de 5 5 en el que deben colocarse los números enteros del 1 al 5. Entre determinadas posiciones de una misma fila pueden aparecer signos de desigualdad ($<$, $>$). Las posiciones están inicialmente vacías, o pueden contener un dígito del 1 al 5 que no se puede cambiar.

La solución al problema debe ser una disposición de números tal que no aparezca ningún número repetido en ninguna fila ni en ninguna columna. Además, si entre dos posiciones consecutivas de una fila aparece un signo de desigualdad, los valores de posiciones deben respetar la desigualdad.

En la figura de la izquierda aparece una instancia del problema de Futoshiki, y en la de la derecha una posible solución al mismo.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $>$ | | | | $>$ | | $>$ | |
| **4** | | | | | | | | **2** |
| | | | | **4** | | | | |
| | | | | | | | $<$ | **4** |
| | $<$ | | $<$ | | | | | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | $>$ | 4 | | 3 | $>$ | 2 | $>$ | 1 |
| **4** | | 3 | | 1 | | 5 | | **2** |
| 2 | | 1 | | **4** | | 3 | | 5 |
| 3 | | 5 | | 2 | | 1 | $<$ | **4** |
| 1 | $<$ | 2 | $<$ | 5 | | 4 | | 3 |

Futoshiki. Estado inicial y Solución.

Se pide formalizar el problema que aparece en la imagen de la izquierda como satisfacción de restricciones.

**Solución**

**Futoshiki**

*Variables:*

Xij - valor de la posición del tablero correspondiente a la fila i, columna j.

*Dominios:*

∀i,j Xij ∈{1,2,3,4,5}

Restricciones:

$$\forall i \quad \underset{j}{Alldiff}\{X_i j\}$$
$$\forall j \quad \underset{i}{Alldiff}\{X_i j\}$$

$X_{2,1} = 4$

$X_{2,5} = 2$

$X_{3,3} = 4$

$X_{4,5} = 4$

$X_{1,1} > X_{1,2}$

$X_{1,3} > X_{1,4}$
$X_{1,4} > X_{1,5}$
$X_{4,4} < X_{4,5}$
$X_{5,1} < X_{5,2}$
$X_{5,2} < X_{5,3}$

**Ejercicio 3.11** Imagina la siguiente situación: una familia de cuatro miembros ha de averiguar como irá cada uno al trabajo o a la escuela, dadas ciertas restricciones. La familia consta de madre, padre, hijo e hija. Cada uno de ellos puede ir en bicicleta o en coche. Además el hijo tiene un patinete con el que puede ir a la escuela. La asignación de los medios de transporte a los miembros de la familia está sujeta a las siguientes restricciones:

a) Solo hay dos bicicletas.

b) El coche solo puede transportar tres personas.

c) El hijo y la hija han de usar el mismo medio de transporte.

d) El hijo y la hija solo pueden ir en coche si al menos uno de los padres va en coche, ya que solo los padres pueden conducir hasta la escuela.

Modela esta situación como un problema de satisfacción de restricciones.

Pista: puedes usar el corchete de Iverson para expresar algunas restricciones:

$$[P] = \begin{cases} 1 & si\ P \text{ es verdadero} \\ 0 & \text{en otro caso} \end{cases}$$

**Solución**

*Variables:*

$X_1$=Modo transporte asignado a la Madre

$X_2$= Modo transporte asignado a la Padre

$X_3$= Modo transporte asignado al Hijo

$X_4$= Modo transporte asignado al Hija

*Dominios:*

$\forall i \in \{1,2,4\}$ $X_i \in$ {Bicicleta,Coche}

$X_3 \in$ {Bicicleta,Coche,Patinete}

*Restricciones:*

1.
$$\sum_{i=1}^{4} [X_i = \text{Bicicleta}] \leq 2$$

2. ($X_1 \neq$Coche) $\vee$ ($X_2 \neq$ Coche) $\vee$ ($X_3 \neq$ Coche) $\vee$ ($X_4 \neq$ Coche)

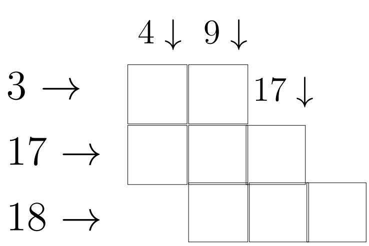
3. $X_3$=$X_4$

4. $\forall i \in \{3,4\}$ $X_i$= Coche $\Rightarrow$ ($X_1$= Coche) $\vee$ ($X_2$= Coche)

**Ejercicio 3.12** Un Kakuro es como un crucigrama pero con números. Cada "palabra" es un conjunto de dígitos que, sumados, dan como resultado el número situado a la izquierda o arriba que serían como las definiciones ordinarias de los crucigramas de palabras.

Las "palabras" sólo pueden usar los números del 1 al 9 y un número dado sólo puede usarse una vez en una palabra.

Dado el siguiente Kakuro se pide:




Kakuro. Estado inicial.

a) Formalizar el problema como un problema de satisfacción de restricciones

b) ¿Se podría resolver por un procedimiento de 2-consistencia en arcos? Razona tu respuesta.

c) Resuélvelo mediante búsqueda con retroceso (backtracking) encontrando al menos una solución al problema.

**Solución**

a) Formalización.

| | | | | |
|---|---|---|---|---|
| | 4↓ | 9↓ | | |
| 3→ | $x_1$ | $x_2$ | 17↓ | |
| 17→ | $x_3$ | $x_4$ | $x_5$ | |
| | 18 → | $x_6$ | $x_7$ | $x_8$ |

Variables: $\mathrm{x}_1$,$\mathrm{x}_2$,$\mathrm{x}_3$,$\mathrm{x}_4$,$\mathrm{x}_5$,$\mathrm{x}_6$,$\mathrm{x}_7$,$\mathrm{x}_8$

Dominio: {1,2,3,4,5,6,7,8,9}

Restricciones:

1. $\mathrm{x}_1 \neq \mathrm{x}_2$, $\mathrm{x}_1 \neq \mathrm{x}_3$, $\mathrm{x}_2 \neq \mathrm{x}_4$, $\mathrm{x}_4 \neq \mathrm{x}_6$, $\mathrm{x}_2 \neq \mathrm{x}_6$, $\mathrm{x}_3 \neq \mathrm{x}_4$, $\mathrm{x}_4 \neq \mathrm{x}_5$, $\mathrm{x}_3 \neq \mathrm{x}_5$, $\mathrm{x}_5 \neq \mathrm{x}_7$, $\mathrm{x}_6 \neq \mathrm{x}_7$, $\mathrm{x}_7 \neq \mathrm{x}_8$, $\mathrm{x}_7 \neq \mathrm{x}_8$
2. $\mathrm{x}_1 + \mathrm{x}_2 = 3$
3. $\mathrm{x}_3 + \mathrm{x}_4 + \mathrm{x}_5 = 17$
4. $\mathrm{x}_6 + \mathrm{x}_7 + \mathrm{x}_8 = 18$
5. $\mathrm{x}_1 + \mathrm{x}_3 = 4$
6. $\mathrm{x}_2 + \mathrm{x}_4 + \mathrm{x}_6 = 9$
7. $\mathrm{x}_5 + \mathrm{x}_7 \ = 17$

b) Consistencia en arcos.Sí se puede resolver por un procedimiento de 2-consistencia en arcos pues todas las restricciones son binarias.

1. $\mathrm{x}_1 \neq \mathrm{x}_2$, $\mathrm{x}_1 \neq \mathrm{x}_3$, $\mathrm{x}_2 \neq \mathrm{x}_4$, $\mathrm{x}_4 \neq \mathrm{x}_6$, $\mathrm{x}_2 \neq \mathrm{x}_6$, $\mathrm{x}_3 \neq \mathrm{x}_4$, $\mathrm{x}_4 \neq \mathrm{x}_5$, $\mathrm{x}_3 \neq \mathrm{x}_5$, $\mathrm{x}_5 \neq \mathrm{x}_7$, $\mathrm{x}_6 \neq \mathrm{x}_7$, $\mathrm{x}_7 \neq \mathrm{x}_8$, $\mathrm{x}_7 \neq \mathrm{x}_8$
2. $\mathrm{x}_1 + \mathrm{x}_2 = 3$

3. $x_3 + x_4 + x_5 = 17$
4. $x_6 + x_7 + x_8 = 18$
5. $x_1 + x_3 = 4$
6. $x_2 + x_4 + x_6 = 9$
7. $x_5 + x_7 = 17$

Empezamos con las más restrictivas.

Según r2 el dominio sería dx1, dx2 ={1,2}

Según r1, r2 y r5:

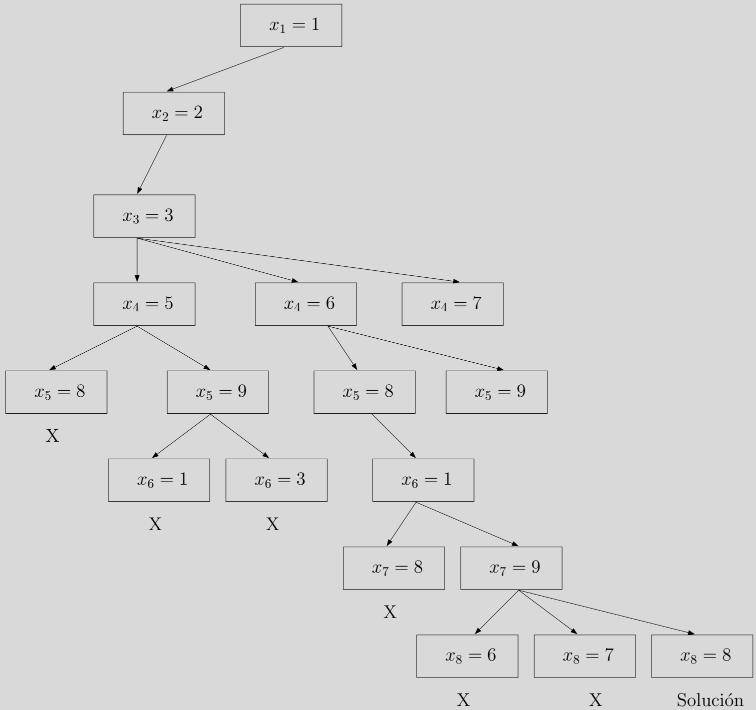
1. $x_1 + x_2 = 3$; $x_1 = 3 - x_2$
2. $x_1 + x_3 = 4$; $3 - x_2 + x_3 = 4$; $x_3 = x_2+1$

dx3 ={2,3}

Según r7 el dominio sería dx5,dx7 ={8,9}

Según r3 y los demás dominios dx4={5,6,7}

c) Solución mediante backtracking: Orden de asignación de variables $x_1..x8$. Orden de asignación de dominios 1..9



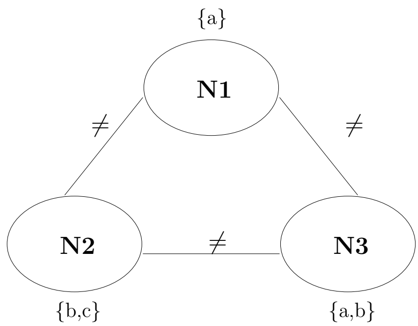

Solución:

| | | | | |
|---|---|---|---|---|
| | 4$\downarrow$ | 9$\downarrow$ | | |
| 3$\rightarrow$ | 1 | 2 | 17$\downarrow$ | |
| 17$\rightarrow$ | 3 | 6 | 8 | |
| | 18 $\rightarrow$ | 1 | 9 | 8 |

**Ejercicio 3.13** Aplique el algoritmo AC-3 al siguiente grafo de restricciones.

a) Rellene la tabla de la izquierda con los pasos del algoritmo Arcos

b) Determine la solución final al finalizar el Algoritmo.




**Solución:**

| Arcos | Revise |
|---|---|
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

**Solución**

La solución final sería: N1= {a} , N2 ={c}, N3={b}.

| **Arcos** | **REVISE** |
|---|---|
| (N1,N2) | FALSE |
| (N1,N3) | FALSE |
| (N2,N1) | FALSE |
| (N2,N3) | FALSE |
| (N3,N1) | TRUE<br>(V3,a) eliminado<br>Añado (N2,N3) |
| (N3,N2) | FALSE |
| (N2,N3) | TRUE<br>(V2,b) eliminado<br>Añado (N1,N2) |
| (N1,N2) | FALSE |

**Ejercicio 3.14** Considere el problema de planificar cinco tareas, T1, T2, T3, T4, T5, cada una de las cuales tarda una hora en completarse. Las tareas deben comenzar a 13:00, 14:00, o 15:00. Las tareas deben ser ejecutadas simultáneamente, sujeto a las restricciones:

- T1 debe comenzar después de T3.
- T3 debe comenzar antes de T4 y después de T5
- T2 no puede ejecutarse al mismo tiempo que T1 o T4
- T4 no puede comenzar a las 14:00.

Se pide:

a) Representar el problema como satisfacción de restricciones (variables, dominios y restricciones) de forma que todas las restricciones sean binarias.

b) Aplicar el algoritmo AC-3 de consistencia en arcos.

c) Aplicar una búsqueda con retroceso para hallar todas las posibles soluciones. Aplicar estrictamente el algoritmo. No se considerarán válidas soluciones obtenidas de otra forma.

**Solución**

a) Formalización

*Variables:*

Cada una de las cinco tareas T1... T5

*Dominios:*

D1 = ... = D5 = {1, 2, 3} (la hora de comienzo de cada tarea, que basta para formular todas las restricciones)

La restricción unaria relativa a T4 la integramos en su dominio:

D1 = D2 = D3 = D5 = {1, 2, 3}

D4 = {1, 3}

*Restricciones binarias.*

T1-T2: T1<>T2

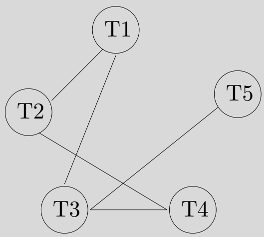
T1-T3: T3 < T1

T2-T4: T2<>T4

T3-T4: T3<T4

T3-T5: T5<T3

b) Consistencia en arcos:




**Restricciones Unarias**

$D_{T4}$={ 1:00, 3:00}

**Restricciones Binarias**

$R_{T1,T2}$:{ (1:00, 2:00), (1:00, 3:00), (2:00, 1:00) (2:00, 3:00), (3:00, 1:00), (3:00, 2:00) }

$R_{T1,T3}$:{ (2:00, 1:00), (3:00, 1:00), (3:00, 2:00)}

$R_{T2,T4}$:{ (1:00, 2:00), (1:00, 3:00), (2:00, 1:00) (2:00, 3:00), (3:00, 1:00), (3:00, 2:00) }

$R_{T3,T4}$:{ (1:00, 2:00), (1:00, 3:00), (2:00, 3:00)}

$R_{T3,T5}$:{ (2:00, 1:00), (3:00, 1:00), (3:00, 2:00)}

Tras la ejecución del algoritmo quedan los dominios:

D1= 3

D2= 1, 2

D3= 2

D4= 3

D5= 1

c) Solución con Búsqueda en árbol: Vemos que las dos combinaciones posibles son soluciones del problema.

**Ejercicio 3.15** Tres parejas (Juan y Juana, Pepe y Pepa, y María y Mario) se deben sentar a una mesa redonda en la que hay seis puestos. El lugar de honor, junto a la ventana, corresponde a Juana. No pueden sentarse en puestos contiguos dos personas del mismo sexo. Tampoco pueden sentarse en puestos contiguos dos personas que sean pareja.

a) Representar el problema como satisfacción de restricciones (variables, dominios y restricciones) de forma que todas las restricciones sean binarias.

b) Aplicar el algoritmo AC-3 de consistencia en arcos.

c) Aplicar una búsqueda con retroceso para hallar todas las posibles soluciones. Aplicar estrictamente el algoritmo. No se considerarán válidas soluciones obtenidas de otra forma.

**Solución**

a) Formalización

*Variables:*

Cada una de las seis posiciones P1... P6. Suponemos que P1 es el lugar de honor.

sex(Jo)=sex(Po)=sex(Mo)=m≠f=sex(Ja)=sex(Pa)=sex(Ma)
par(Jo)=par(Ja)≠par(Po)= par(Pa)≠par(Mo)=par(Ma)≠par(Jo)

*Dominios:*

P1 = ... = P6 = {Jo, Ja, Po, Pa, Ma, Mo} (las seis personas)

La restricción unaria relativa a P1 la integramos en su dominio:

P1 = {Ja}
P2 = ... = P6 = {Jo, Ja, Po, Pa, Ma, Mo}

*Restricciones binarias.*
Si i +1 mod 6 = j mod 6 o i - 1 mod 6 = j mod 6 ⇒ Pi≠Pj & sex(Pi)≠sex(Pj) & par(Pi)≠par(Pj)
En otro caso Pi≠Pj

b) Consistencia en arcos: Quedan los dominios
P1= Ja
P2= Po, Mo
P3= Pa, Ma
P4= Po,Mo, Jo
P5= Pa,Ma
P6=Po, Mo

c) Búsqueda en árbol: Vemos que hay dos soluciones del problema.

| | |
|---|---|
| P1=Ja | P1=Ja |
| P2= Po | P2= Mo |
| P3= Ma | P3= Pa |
| P4= Jo | P4= Jo |
| P5= Pa | P5= Ma |
| P6= Mo | P6= Po |

**Ejercicio 3.16** Pepi, Quique, Rosi y Susi son cuatro hermanos que nacieron en la década de 1980 (es decir, entre los años 1980 y 1989, ambos inclusive). Pepi nació al menos dos años antes que Quique. Quique nació en un año no bisiesto, y Susi en un año bisiesto. Quique nació al menos un año antes que Rosi. Rosi nació exactamente tres años antes que Susi.

a) Representar el problema como satisfacción de restricciones (variables, dominios y restricciones) de forma que todas las restricciones sean unarias o binarias.

b) Aplicar el algoritmo AC-3 de consistencia en arcos.

c) Sobre los dominios simplificados obtenidos en el apartado b, aplicar una búsqueda con retroceso para hallar todas las posibles soluciones. No se considerarán válidas soluciones obtenidas de otra forma.

**Solución**

a) Formalización

*Variables:*
Los años del nacimiento de cada hermano, denotados por P, Q, R y S.

*Dominios:* Hay dos restricciones unarias (año bisiesto/no bisiesto) que consideramos directamente para formular los dominios iniciales.

P = R = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Q = {1, 2, 3, 5, 6, 7, 9}

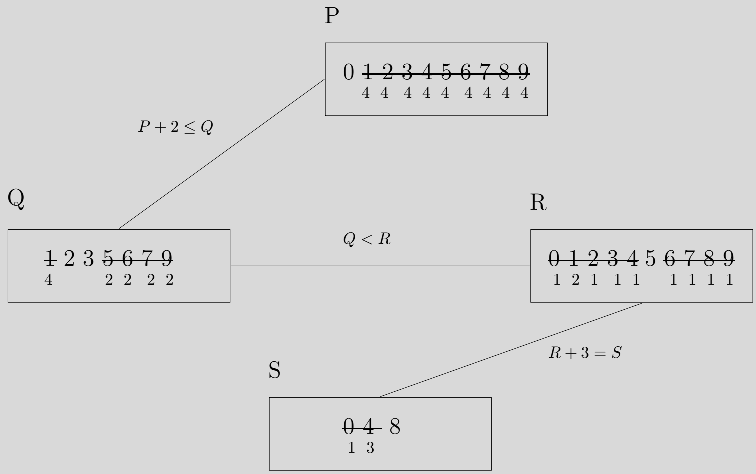
S = {0, 4, 8}

*Restricciones binarias:*

$$R_1\text{: } P+2 \leq Q$$
$$R_2\text{: } Q<R$$
$$R_3\text{: } R+3=S$$

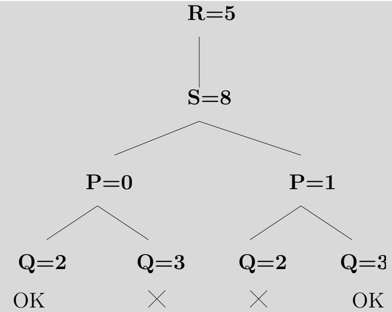
b) Consistencia en arcos:
En el dibujo se han considerado las restricciones en el orden siguiente:

1. R-S en ambos sentidos.
2. Q-R en ambos sentidos.
3. R-S en ambos sentidos.
4. P-Q en ambos sentidos.




Cualquiera que sea el orden, los dominios finales son:
P= {0, 1}
Q= {2, 3}
R= {5}
S= {8}

c) Búsqueda en árbol: Vemos que hay dos soluciones del problema según se ilustra en el gráfico adjunto.




**Nota:** en este problema no se exige la restricción AllDiffs. así que no hay que ponerla explícitamente en la formulación (apartado a). Es obvio que esta restricción se deduce de las desigualdades exigidas, pero preprocesar las restricciones es cosa que no consideramos.

**Ejercicio 3.17** Tres parejas (Ana y Bruno; Carlos y Dolores; y Eloísa y Fernando) se quieren sentar en una mesa redonda cuyos asientos en el orden de las agujas del reloj están numerados 1, 2, 3, 4, 5, 6. Ninguno de ellos quiere sentanse junto a su pareja o expareja (Eloísa y Carlos fueron pareja), pero todos quieren tener a izquierda y derecha a una persona del sexo contrario. Ana quiere sentarse frente a la puerta, en el asiento 1.

a) Representar el problema como satisfacción de restricciones (variables, dominios y restricciones) de forma que todas las restricciones sean unarias o binarias (y AllDiffs).

b) Aplicar el algoritmo AC-3 de consistencia en arcos.

c) Sobre los dominios simplificados obtenidos en el apartado b, aplicar una búsqueda con retroceso para hallar todas las posibles soluciones. Aplicar estrictamente el algoritmo. No se considerarán válidas soluciones obtenidas de otra forma.

**Solución**

a) Formalización

*Variables:*
Los asientos, denotados por A1...A6

*Dominios:*
A1 = {a}
A2...A6 = {a, b, c, d, e, f}
Para plantear el problema más fácilmente consideramos las funciones sexo(X) y los predicados pareja(X,Y), expareja(X,Y).

*Restricciones binarias:*
AllDiffs
A1-A2:
sexo(A1)≠sexo(A2) & pareja(A1,A2)=falso & expareja(A1,A2)=falso
A2-A3:
sexo(A2)≠sexo(A3) & pareja(A2,A3)=falso & expareja(A2,A3)=falso
A3-A4:
sexo(A3)≠sexo(A4) & pareja(A3,A4)=falso & expareja(A3,A4)=falso
A4-A5:
sexo(A4)≠sexo(A5) & pareja(A4,A5)=falso & expareja(A4,A5)=falso
A5-A6:
sexo(A5)≠sexo(A6) & pareja(A5,A6)=falso & expareja(A5,A6)=falso
A6-A1:
sexo(A6)≠sexo(A1) & pareja(A6,A1)=falso & expareja(A6,A1)=falso

b) Consistencia en arcos:
Aplicamos AllDiffs
A1 = {a}
A2...A6 = {b, c, d, e, f}
Ahora, por ejemplo, consideramos la restricción A1-A2.
A2 = {c, f} (son los únicos de distinto sexo que Ana y que no son su pareja)
Consideramos la restricción A2-A3.
A3 = {d} (han de ser mujeres, y Eloísa no está apoyada por Carlos ni por Fernando))
Aplicamos AllDiffs

A1 = {a}
A2 = {c, f}
A3 = {d}
A4...A6 = {b, c, e, f}
Consideramos la restricción A3-A4.
A4 = {b, f } (han de ser varones y Carlos no está apoyado por Dolores)
Consideramos la restricción A4-A5.
A5 = {e} (han de ser mujeres)
Aplicamos AllDiffs
A1 = {a}
A2 = {c, f}
A3 = {d}
A4 = {b, f }
A5 = {e}
A6 = {b, c, f}
Consideramos de nuevo la restricción A4-A5.
A4 = {b} (Fernando no está apoyado por Eloísa)
Aplicamos AllDiffs
A6 = {c, f}
Consideramos la restricción A5-A6.
A6 = {} (ni Carlos ni Fernando están apoyados por Eloísa)
El algoritmo termina, pues un dominio ha quedado vacío. La consideración de las restricciones en cualquier otro orden hubiera producido el mismo resultado.

c) Búsqueda en árbol: No es necesaria. Sabemos que no hay solución.

**Ejercicio 3.18** Resuelva el siguiente sudoku aplicando el algoritmo AC-3.

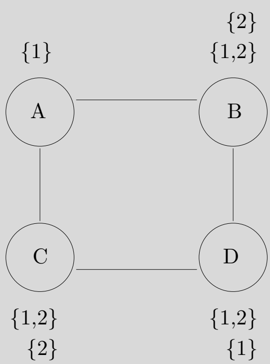
a) Realice el grafo de restricciones teniendo en cuenta los siguientes dominios: A={1}, B={1,2}, C={1,2}, D={1,2}.

b) Rellene la tabla de la izquierda con los pasos del algoritmo

| | | | |
|---|---|---|---|
| 1 | 2 | 4 | 3 |
| 3 | 4 | **A** | **B** |
| 4 | 3 | **C** | **D** |
| 2 | 1 | 3 | 4 |

**Solución**

Grafo de Restricciones




| **Arcos** | **REVISE** |
|---|---|
| A,B | {1} {1,2} FALSE |
| A,C | {1} {1,2} FALSE |
| B,A | {1,2} {1} TRUE,- Elimina 1 B |
| B,D | {2} {1,2} FALSE |
| C,A | {1,2} {1} TRUE – Elimina 1 C |
| C,D | {2} {1,2} FALSE |
| D,B | {1,2} {2} TRUE – Elimina 2 D |
| D,C | {1} {2} FALSE |
| ——- | ——————————- |
| D,B | {1} {2} FALSE |
| D,C | {1} {2} FALSE |
| C,D | {2} {1} FALSE |

La solución sería A={1}, B={2}, C={2}, D={1}.

# Tema 4

# Lógica proposicional

**Ejercicio 4.1** Demuestra que las siguientes fórmulas bien formadas son tautologías:

a) $[P \wedge (P \Rightarrow Q)] \Rightarrow Q$

b) $(P \Rightarrow Q) \Leftrightarrow (\neg Q \Rightarrow \neg P)$

c) $[\neg Q \wedge (P \Rightarrow Q)] \Rightarrow \neg P$

**Solución**

Construimos una tabla de verdad para cada fórmula, y comprobamos que sus valores de verdad son verdaderos sean cuales sean los valores de verdad de las proposiciones simples.

a)

| $P$ | $Q$ | $P \Rightarrow Q$ | $P \wedge (P \Rightarrow Q)$ | $[P \wedge (P \Rightarrow Q)] \Rightarrow Q$ |
|---|---|---|---|---|
| V | V | V | V | V |
| V | F | F | F | V |
| F | V | V | F | V |
| F | F | V | F | V |

b)

| $P$ | $Q$ | $\neg Q$ | $\neg P$ | $P \Rightarrow Q$ | $\neg Q \Rightarrow \neg P$ | $(P \Rightarrow Q) \Leftrightarrow (\neg Q \Rightarrow \neg P)$ |
|---|---|---|---|---|---|---|
| V | V | F | F | V | V | V |
| V | F | V | F | F | F | V |
| F | V | F | V | V | V | V |
| F | F | V | V | V | V | V |

c)

| $P$ | $Q$ | $\neg Q$ | $\neg P$ | $P \Rightarrow Q$ | $\neg Q \wedge (P \Rightarrow Q)$ | $[\neg Q \wedge (P \Rightarrow Q)] \neg P)$ |
|---|---|---|---|---|---|---|
| V | V | F | F | V | F | V |
| V | F | V | F | F | F | V |
| F | V | F | V | V | F | V |
| F | F | V | V | V | V | V |

**Ejercicio 4.2** Formaliza las premisas de los siguientes argumentos como bases de conocimiento y las conclusiones como fórmulas bien formadas de la lógica proposicional. Esto es, debes modelar los argumentos de esta forma: $KB \models \alpha$.

a) El asesino fue o el mayor Brown o el profesor Black. Pero no fue el profesor Black. Así que fue el mayor Brown.

b) Juana o Sandra estaban en la biblioteca. Pero si Juana no estaba, Sandra tampoco. Así que estaban ambas en la biblioteca.

c) Sólo puedes obtener una tarjeta Joven si eres menor de 29 años o estudiante; en otro caso no puedes. Si puedes obtener una tarjeta Joven, puedes obtener entradas de museo con descuento. Pero no eres menor de 29 años. Así que a menos que seas estudiante, no puedes obtener entradas de museo con descuento.

d) Los manifestantes se irán si la universidad detiene los experimentos con animales. Pero esto sólo podría ocurrir por una intervención del gobierno. Por tanto, a menos que el gobierno intervenga, no se irán.

**Solución**

a) **KB:** $BrownAsesino \lor BlackAsesino$
$\neg(BrownAsesino \land BlackAsesino)$
$\neg BlackAsesino$
$\alpha$**:** $BrownAsesino$

b) **KB:** $JuanaBiblioteca \lor SandraBiblioteca$
$\neg JuanaBiblioteca \Rightarrow \neg SandraBiblioteca$
$\alpha$**:** $JuanaBiblioteca \land SandraBiblioteca$

c) **KB:** $PuedeObtenerTarjeta \Leftrightarrow Menor29 \lor Estudiante$
$PuedeObtenerTarjeta \Rightarrow DescuentoEntradas$
$\neg Menor29$
$\alpha$**:** $\neg Estudiante \Rightarrow \neg DescuentoEntradas$

d) **KB:** $PararExperimentos \Rightarrow SeVan$
$PararExperimentos \Rightarrow IntervencionGobierno$
$\alpha$**:** $\neg IntervencionGobierno \Rightarrow \neg SeVan$

**Ejercicio 4.3** Convierte los siguientes conjuntos de fórmulas bien formadas a la forma clausal, y da una traza de la ejecución del algoritmo de resolución sobre la conjunción de las cláusulas obtenidas.

a) S1. $P \Rightarrow Q$
S2. $R \Rightarrow (P \vee Q)$
S3. $R$
S4. $\neg Q$

b) S1. $P \Rightarrow Q$
S2. $S \Rightarrow \neg R$
S3. $\neg P \Rightarrow S$
S4. $P \Rightarrow \neg S$
S5. $R \vee Q$
S6. $\neg(R \wedge Q)$
S7. $\neg Q$

**Solución**

a) **C1.** $\neg P \vee Q$
**C2.** $\neg R \vee P \vee Q$
**C3.** $R$
**C4.** $\neg Q$

---

**C5.** $P \vee QP$(C3,C2)
**C6.** $Q$(C5,C1)
**C7.** $\square$ (C6,C4)

b) **C1.** $\neg P \vee Q$
**C2.** $\neg S \vee \neg R$
**C3.** $P \vee S$
**C4.** $\neg P \vee \neg S$
**C5.** $R \vee Q$

**C6.** $\neg R \vee \neg Q$

**C7.** $\neg Q$

---

**C8.** $S \vee Q$ (C3,C1)

**C9.** $\neg R \vee Q$ (C8,C2)

**C10.** $Q$ (C9,C5)

**C11.** $\square$ (C10,C7)

**Ejercicio 4.4** Escribe el siguiente argumento en el lenguaje de la lógica proposicional. Debes modelarlo de esta forma: $KB \models \alpha$. A continuación establece la validez del argumento mediante el algoritmo de resolución, o bien da un contraejemplo para demostrar que no es válido.

Si Antonio suspende el examen de Inglés, entonces Juan (el profesor de Antonio) estará decepcionado. Si Rafael suspende el examen de Matemáticas, entonces María (la profesora de Rafael) estará decepcionada. Si Juan o María están decepcionados, entonces Sandra (la directora) será informada. Sandra no ha sido notificada por parte de ninguno de los dos profesores. Consecuentemente, Antonio aprobó el examen de Inglés y Rafael aprobó el examen de Matemáticas.

**Solución**

**KB:** $AntonioSuspende \Rightarrow JuanDecepcionado$

$RafaelSuspende \Rightarrow MariaDecepcionada$

$JuanDecepcionado \vee MariaDecepcionada \Rightarrow SandraNotificada$

$\neg SandraNotificada$

$\alpha$**:** $\neg AntonioSuspende \wedge \neg RafaelSuspende$

**C1.** $\neg AntonioSuspende \vee JuanDecepcionado$

**C2.** $\neg RafaelSuspende \vee MariaDecepcionada$

**C3.** $\neg JuanDecepcionado \vee SandraNotificada$

**C4.** $\neg MariaDecepcionada \vee SandraNotificada$

**C5.** $\neg SandraNotificada$

**C6.** $AntonioSuspende \vee RafaelSuspende$

---

**C7.** $\neg MariaDecepcionada$ (C5,C4)

**C8.** $\neg JuanDecepcionado$ (C5,C3)

**C9.** $\neg AntonioSuspende$ (C8,C1)

**C10.** $\neg RafaelSuspende$ (C7,C2)

**C11.** $RafaelSuspende$ (C9,C6)

**C12.** $\square$ (C11,C10)

Si el argumento no hubiera sido válido, podríamos haberlo mostrado dando un contraejemplo, es decir, un modelo en el que la KB es verdadera y la conclusión es falsa.

**Ejercicio 4.5** Tentaizu ("mapa celestial" en japonés) es un puzzle de lógica y álgebra. Se muestra un ejemplo en la figura. Hay que encontrar siete estrellas. Las celdas con números se llaman celdas pista. Las estrellas están ocultas en celdas que no son pistas. Cada celda puede esconder solamente una estrella. Se considera que dos celdas son vecinas si comparten un lado o una esquina; esto es lo que se llama 8-vecindad. Una celda pista con el número $n$ indica que hay $n$ estrellas escondidas en sus celdas vecinas.

| | | | | | |
|---|---|---|---|---|---|
| 1 | | 1 | 2 | $\star$ | |
| | | | | 1 | |
| | | | | | |
| 1 | $\star$ | | 2 | | |
| 2 | | | | | 2 |
| $\star$ | 1 | | | | |

Explica cómo la información proporcionada por las pistas puede expresarse en el lenguaje de la lógica proposicional, y cómo se puede encontrar la solución del puzzle con ayuda de un demostrador de teoremas. Ten en cuenta que los puzzles Tentaizu tienen siempre una única solución.

**Solución**

Sea $S_{ij}$ una proposición que es verdadera si y sólo si hay una estrella escondida en la posición (i,j). En el ejemplo tendremos 36 símbolos de proposición, desde $S_{11}$ hasta $S_{66}$. Para cada celda de pista con número n, tendremos una disyunción de $\binom{m}{n}$ conjunciones de literales, donde m es el número de 8-vecinos de la celda de pista. Estas conjunciones se corresponden con las $\binom{m}{n}$ maneras distintas en que se pueden distribuir n estrellas entre los m 8-vecinos de la celda de pista.

Por ejemplo, en la figura la celda de pista en la posición (5,6) tiene 5 8-vecinos y su número es 2, así que obtenemos una disyunción de $\binom{5}{2} = 10$ conjunciones de literales:

$$(S_{46} \wedge S_{45} \wedge \neg S_{55} \wedge \neg S_{65} \wedge \neg S_{66}) \vee$$
$$\ldots$$
$$(S_{46} \wedge S_{45} \wedge \neg S_{55} \wedge \neg S_{65} \wedge \neg S_{66}) \vee$$

Una vez que convertimos las fórmulas correspondientes a todas las celdas de pista a CNF, podemos comprobar si hay una estrella en la posición (i,j) averiguando si $KB \models S_{ij}$ por medio de un demostrador de teoremas.

**Ejercicio 4.6** Hay una pastelería famosa donde acude mucha gente a comprar tartas de cumpleaños. Modela los siguientes datos en el lenguaje de la lógica proposicional. A continuación di cómo se podría encontrar la solución del puzzle con la ayuda de un demostrador de teoremas.

a) La tarta de manzana es para Jaime, pero no la ha comprado Andrea.

b) Sandra ha reservado la tarta de plátanos, pero no para la muchacha que cumplirá 15 años.

c) Mario va a regalarle una tarta a su padre Manuel.

d) La tarta Battenberg es para Violeta.

e) José no ha encargado la tarta de mantequilla con 45 velas, puesto que a Paula no le gusta la mantequilla.

f) Andrea ha comprado 5 velas menos que Mario.

g) Fernando ha comprado 32 velas, pero no para Patricia, ya que ella va a cumplir 5 años.

h) La oferta especial de la semana es la tarta de ron a mitad de precio.

**Solución**

Usaremos 25 símbolos de proposición para manejar los nombres de los clientes correspondientes a las cinco tartas: $Andrea_i$, $Sandra_i$, $Mario_i$, $Jose_i$ y $Fernando_i$ con $i \in \{1, \ldots, 5\}$. Por ejemplo, $Sandra_4$ indica que Sandra es el nombre de la cliente de la cuarta tarta.

Análogamente, emplearemos 25 símbolos de proposición para gestionar los tipos de las 5 tartas: $Manzana_i$, $Platano_i$, $Battenberg_i$, $Mantequilla_i$, $Ron_i$. Los nombres de las personas a las que se regalan las tartas se manejarán con: $Jaime_i$, $Manuel_i$, $Violeta_i$, $Paula_i$, $Patricia_i$. Finalmente, las edades de las personas a las que se van a regalar las tartas se gestionarán con: $Cinco_i$, $Quince_i$, $TreintaYDos_i$, $Cuarenta_i$, $CuarentaYCinco_i$.

A continuación presentamos las fórmulas de la lógica proposicional que modelan el conocimiento obtenido de los datos proporcionados. Hay una condición que exige que ningún par de tartas tengan el mismo cliente, el mismo tipo, el mismo destinatario ni el mismo número de velas; esto no se afirma explícitamente, pero se puede suponer dada la manera en que nos proporcionan los datos. A fin de expresar que no hay ningún par de tartas que sean del mismo cliente, necesitamos disyunciones de 5 conjunciones de literales:

$$(Andrea_i \wedge \neg Sandra_i \wedge \neg Mario_i \wedge \neg Jose_i \wedge \neg Fernando_i) \vee$$
$$\ldots$$
$$(\neg Andrea_i \wedge \neg Sandra_i \wedge \neg Mario_i \wedge \neg Jose_i \wedge Fernando_i)$$

donde la fórmula anterior se repite para $i \in \{1, \ldots, 5\}$.

Análogamente, tendremos fórmulas para afirmar que no hay ningún par de tartas del mismo tipo para $i \in \{1, \ldots, 5\}$:

$$(Manzana_i \wedge \neg Platano_i \wedge \neg Battenberg_i \wedge \neg Mantequilla_i \wedge \neg Ron_i) \vee$$
$$\ldots$$
$$(\neg Manzana_i \wedge \neg Platano_i \wedge \neg Battenberg_i \wedge \neg Mantequilla_i \wedge Ron_i)$$

Las fórmulas para establecer que no puede haber dos tartas con el mismo destinatario son:

$$(Jaime_i \wedge \neg Manuel_i \wedge \neg Violeta_i \wedge \neg Paula_i \wedge \neg Patricia_i)\vee$$
$$\ldots$$
$$(\neg Jaime_i \wedge \neg Manuel_i \wedge \neg Violeta_i \wedge \neg Paula_i \wedge Patricia_i)\vee$$

Y las fórmulas para decir que no puede haber dos tartas con el mismo número de velas son:

$$(Cinco_i \wedge \neg Quince_i \wedge \neg TreintaYDos_i \wedge \neg Cuarenta_i \wedge \neg CuarentaYCinco_i)\vee$$
$$\ldots$$
$$(\neg Cinco_i \wedge \neg Quince_i \wedge \neg TreintaYDos_i \wedge \neg Cuarenta_i \wedge CuarentaYCinco_i)\vee$$

A continuación escribimos las afirmaciones que pueden inferirse de los datos del problema:

a) La tarta de manzana es para Jaime, pero no la ha comprado Andrea.

$[\forall i \in \{1, \ldots, 5\}, Manzana_i \Leftrightarrow Jaime_i]\wedge$

$[\forall i \in \{1, \ldots, 5\}, Manzana_i \Rightarrow Andrea_i]$

b) Sandra ha reservado la tarta de plátanos, pero no para la muchacha que cumplirá 15 años.

$[\forall i \in \{1, \ldots, 5\}, Platano_i \Leftrightarrow Sandra_i]\wedge$

$[\forall i \in \{1, \ldots, 5\}, Platano_i \Rightarrow \neg Quince_i]\wedge$

$[\forall i \in \{1, \ldots, 5\}, Quince_i \Rightarrow Violeta_i \vee Paula_i \vee Patricia_i]$

c) Mario va a regalarle una tarta a su padre Manuel.

$\forall i \in \{1, \ldots, 5\}, Mario_i \Leftrightarrow Manuel_i$

d) La tarta Battenberg es para Violeta.

$\forall i \in \{1, \ldots, 5\}, Battenberg_i \Leftrightarrow Violeta_i$

e) José no ha encargado la tarta de mantequilla con 45 velas, puesto que a Paula no le gusta la mantequilla.

$[\forall i \in \{1, \ldots, 5\}, Jose_i \Rightarrow \neg Mantequilla_i]\wedge$

$[\forall i \in \{1,\ldots,5\}, Mantequilla_i \Leftrightarrow CuarentaYCinco_i] \wedge$

$[\forall i \in \{1,\ldots,5\}, Jose_i \Leftrightarrow Paula_i]$

f) Andrea ha comprado 5 velas menos que Mario.

$\forall i \in \{1,\ldots,5\} \forall j \in \{1,\ldots,5\}, Andrea_i \wedge Mario_j \Rightarrow Cuarenta_i \wedge CuarentaYCinco_j$

g) Fernando ha comprado 32 velas, pero no para Patricia, ya que ella va a cumplir 5 años.

$[\forall i \in \{1,\ldots,5\}, Fernando_i \Leftrightarrow \neg TreintaYDos_i] \wedge$

$[\forall i \in \{1,\ldots,5\}, Fernando_i \Rightarrow \neg Patricia_i] \wedge$

$[\forall i \in \{1,\ldots,5\}, Patricia_i \Rightarrow Cinco_i]$

h) La oferta especial de la semana es la tarta de ron a mitad de precio.

De aquí no se saca ninguna fórmula; la frase está únicamente para informarnos de que existe una tarta de ron.

Antes de intentar encontrar la solución del puzzle, hay un detalle que debemos considerar. Hay una indeterminación en la asignación de las tartas a los clientes, así que debemos añadir algunos hechos a la KB para obtener una solución única. Por ejemplo, podríamos añadir

$$Andrea_1 \wedge Sandra_2 \wedge Mario_3 \wedge Jose_4 \wedge Fernando_5$$

para que la primera tarta sea la que compra Andrea, la segunda tarta sea la que compra Sandra, y así sucesivamente. Después de hacer esto, para resolver el puzzle preguntamos si KB $\models \alpha$ para el resto de átomos . Por ejemplo, para ver si la cuarta tarta es una tarta de plátanos preguntamos si $KB \models Platano_4$.

**Ejercicio 4.7** Demuestra aplicando el algoritmo de resolución que la siguiente inferencia es correcta:

$\neg Q \wedge (R \Rightarrow Q)$

$\neg R \Rightarrow P$

$\models$

$\neg R$

**Solución**

**C1.** $\neg Q$

**C2.** $\neg R \vee Q$

**C3.** $R \vee P$

**C4.** $R$

---

**C5.** $\neg R$(C1,C2)

**C6.** $\Box$ (C4,C5)

**Ejercicio 4.8** Dada la siguiente inferencia, aplicar la transformación a forma normal conjuntiva y demostrar que dicha inferencia es correcta aplicando el algoritmo de resolución:

$P \Leftrightarrow T$

$(T \Rightarrow \neg S) \Leftrightarrow Q$

$\neg P$

$\models$

$Q$

**Solución**

**C1.** $\neg P \vee T$

**C2.** $P \vee \neg T$

**C3.** $T \vee Q$

**C4.** $S \vee Q$

**C5.** $\neg T \vee \neg S \vee \neg Q$

**C6.** $\neg P$

**C7.** $\neg Q$

---

**C8.** $\neg T$ (C2,C6)

**C9.** $Q$ (C3,C8)

**C10.** $\Box$ (C7,C9)

**Ejercicio 4.9** Dada la siguiente inferencia, aplicar la transformación a forma normal conjuntiva y demostrar que dicha inferencia es correcta aplicando el algoritmo de resolución:

$Q \Rightarrow P$

$(\neg P \Rightarrow \neg S) \Rightarrow Q$

$\neg S$

$\models$

$P$

**Solución**

**C1.** $P \vee \neg Q$

**C2.** $\neg P \vee Q$

**C3.** $S \vee Q$

**C4.** $\neg S$

**C5.** $\neg P$

---

**C6.** $Q$ (C3,C4)

**C7.** $P$ (C1,C6)

**C8.** $\Box$ (C5,C7)

**Ejercicio 4.10** Dada la siguiente inferencia, aplicar la transformación a forma normal conjuntiva y demostrar que dicha inferencia es correcta aplicando el algoritmo de resolución:

$(P \vee Q) \Rightarrow R$

$(P \Rightarrow \neg S) \Rightarrow Q$

$\neg R$

$\models$

$Q$

**Solución**

**C1.** $\neg P \vee R$

**C2.** $\neg Q \vee R$

**C3.** $P \vee Q$

**C4.** $S \vee Q$

**C5.** $\neg R$

**C6.** $\neg Q$

---

**C6.** $\neg P$(C1,C5)

**C7.** $Q$(C3,C7)

**C8.** $\square$ (C6,C8)

**Ejercicio 4.11** Demuestra aplicando el algoritmo de resolución que la siguiente inferencia es correcta:

$P \Rightarrow R$

$(P \Rightarrow \neg S) \Rightarrow Q$

$\neg R$

$\models$

$Q$

**Solución**

**C1.** $\neg P \vee R$

**C2.** $P \vee Q$

**C3.** $S \vee Q$

**C4.** $\neg R$

**C5.** $\neg Q$

---

**C6.** $\neg P$ (C1,C4)

**C7.** $Q$ (C2,C6)

**C8.** $\square$ (C7,C9)

**Ejercicio 4.12** Demuestra aplicando el algoritmo de resolución que la siguiente inferencia es correcta:

$Q \Rightarrow P$

$(\neg P \Rightarrow \neg S) \Rightarrow Q$

$\neg S$

$\models$

$P$

**Solución**

**C1.** $P \vee \neg Q$

**C2.** $\neg P \vee Q$

**C3.** $S \vee Q$

**C4.** $\neg S$

**C5.** $\neg P$

---

**C6.** $Q$ (C3,C4)

**C7.** $P$ (C1,C6)

**C8.** $\square$ (C5,C7)

**Ejercicio 4.13** Demuestra aplicando el algoritmo de resolución que la siguiente inferencia es correcta:

$(P \vee Q) \Rightarrow R$

$(P \Rightarrow \neg S) \Rightarrow Q$

$\neg R$

$\models$

$Q$

**Solución**

**C1.** $\neg P \vee R$

**C2.** $\neg Q \vee R$

**C3.** $P \vee Q$

**C4.** $S \vee Q$

**C5.** $\neg R$

**C6.** $\neg Q$

---

**C7.** $\neg P$ (C1,C5)

**C8.** $Q$ (C3,C7)

**C9.** $\square$ (C6,C8)

**Ejercicio 4.14** Demuestra aplicando el algoritmo de resolución que la siguiente inferencia es correcta:

$P \vee Q \Rightarrow R$

$P \Rightarrow \neg S \Leftrightarrow Q$

$\neg R$

$\models$

$Q$

**Solución**

**C1.** $\neg P \vee R$

**C2.** $\neg Q \vee R$

**C3.** $P \vee Q$

**C4.** $S \vee Q$

**C5.** $\neg P \vee \neg S \vee \neg Q$

**C6.** $\neg R$

**C7.** $\neg Q$

---

**C8.** $\neg P$ (C1,C6)

**C9.** $Q$ (C3,C8)

**C10.** $\square$ (C7,C9)

**Ejercicio 4.15** Demuestra aplicando el algoritmo de resolución que la siguiente inferencia es correcta:

$T \Leftrightarrow (R \vee Q)$

$(\neg R \Rightarrow \neg S) \Rightarrow T$

$\neg S \vee T$

$\models$

$R$

**Solución**

**C1.** $\neg T \vee R \vee Q$

**C2.** $\neg R \vee T$

**C3.** $\neg Q \vee T$

**C4.** $\neg R \vee T$

**C5.** $S \vee T$

**C6.** $\neg S \vee T$

**C7.** $\neg R$

---

**C8.** $\neg R \vee R \vee Q$ (C1,C3)

**C9.** $\neg T \vee T \vee Q$ (C1,C2)

**C10.** $T$ (C5,C6)

**C11.** $R \vee S \vee Q$ (C1,C5)

**C12.** $S \vee Q$ (C7,C11)

No conseguimos la clausula vacía con lo cual no podemos demostrar que sea correcta.

**Ejercicio 4.16** Demuestra aplicando el algoritmo de resolución que la siguiente inferencia es correcta:

$Q \Leftrightarrow P$

$(\neg P \Rightarrow \neg S) \Rightarrow Q$

$\neg S$

$\models$

$P$

**Solución**

**C1.** $P \vee \neg Q$

**C2.** $\neg P \vee Q$

**C3.** $S \vee Q$

**C4.** $\neg S$

**C5.** $\neg P$

---

**C6.** $Q$ (C3,C4)

**C7.** $P$ (C1,C6)

**C8.** $\square$ (C5,C7)

**Ejercicio 4.17** Demuestra aplicando el algoritmo de resolución que la siguiente inferencia es correcta:

$Q \Leftrightarrow P$

$(\neg P \vee \neg S) \Rightarrow Q$

$\neg S$

$\models$

$\neg P$

**Solución**

**C1.** $\neg P \vee Q$

**C2.** $P \vee \neg Q$

**C3.** $S \vee \neg Q$

**C4.** $\neg S$

**C5.** $P$

---

**C6.** $\neg Q$ (C3,C4)

**C7.** $\neg P$ (C1,C6)

**C8.** $\square$ (C5,C7)

**Ejercicio 4.18** Dada la siguiente inferencia, pásala a forma normal conjuntiva y emplea el algoritmo de resolución para determinar si es correcta

$R \Rightarrow P$
$T \Rightarrow Q$
$\neg R \Rightarrow T$
$P \Rightarrow T$
$\models$
$Q$

**Solución**

**C1.** $\neg R \vee P$

**C2.** $\neg T \vee Q$

**C3.** $R \vee T$

**C4.** $\neg P \vee T$

**C5.** $\neg Q$

---

**C6.** $\neg T$ (C2,C5)

**C7.** $R$ (C3,C6)

**C8.** $P$ (C1,C7)

**C9.** $T$ (C4,C8)

**C10.** $\square$ (C6,C9)

**Ejercicio 4.19** Demuestra aplicando el algoritmo de resolución que la siguiente inferencia es correcta:

$Q \Leftrightarrow P$

$(\neg P \vee \neg S) \Rightarrow \neg Q$

$\neg S$

$\models$

$\neg P$

**Solución**

**C1.** $\neg P \vee Q$

**C2.** $P \vee \neg Q$

**C3.** $S \vee \neg Q$

**C4.** $\neg S$

**C5.** $P$

---

**C6.** $\neg Q$ (C3,C4)

**C7.** $\neg P$ (C1,C6)

**C8.** $\square$ (C5,C7)

**Ejercicio 4.20** Demuestra aplicando el algoritmo de resolución si la siguiente inferencia es correcta o no:

$Q \Rightarrow P$

$(\neg P \Rightarrow \neg S) \Rightarrow Q$

$\neg S$

$\models$

$P$

**Solución**

**C1.** $P \vee \neg Q$

**C2.** $\neg P \vee Q$

**C3.** $S \vee Q$

**C4.** $\neg S$

**C5.** $\neg P$

---

**C6.** $Q$ (C3,C4)

**C7.** $P$ (C1,C6)

**C8.** $\square$ (C5,C7)

**Ejercicio 4.21** Aplica el algoritmo de resolución para demostrar si la siguiente inferencia es correcta:

$R \Leftrightarrow Q$

$(\neg Q \vee \neg T) \Rightarrow R$

$\neg T$

$\models$

$Q \wedge R$

**Solución**

**C1.** $\neg R \vee Q$

**C2.** $R \vee \neg Q$

**C3.** $Q \vee R$

**C4.** $T \vee R$

**C5.** $\neg T$

**C6.** $\neg Q \vee \neg R$

---

**C7.** $R$ (C4,C5)

**C8.** $\neg R$ (C1,C6)

**C9.** $\square$ (C7,C8)

**Ejercicio 4.22** Demuestra que la siguiente inferencia es correcta, aplicando el algoritmo de resolución y mediante model checking :

$Q \Rightarrow (P \wedge S)$
$(\neg P \Rightarrow \neg S) \Rightarrow Q$
$\neg S \Leftrightarrow Q$
$\models$
$\neg P$

**Solución**

Aplicando el algoritmo de resolución

**C1.** $\neg Q \vee P$

**C2.** $\neg Q \vee S$

**C3.** $\neg P \vee Q$

**C4.** $S \vee Q$

**C5.** $S \vee Q$

**C6.** $\neg Q \vee \neg S$

**C7.** $P$

---

**C8.** $Q$ (C7, C3)

**C9.** $\neg S$ (C8, C6)

**C10.** $\neg Q$ (C9, C2)

**C11.** $\square$ (C10, C8)

Mediante model checking

| QPS | $Q \Rightarrow (P \wedge S)$ | $(\neg P \Rightarrow \neg S) \Rightarrow Q$ | $\neg S \Leftrightarrow Q$ | $\wedge$ | $\neg P$ |
|---|---|---|---|---|---|
| 000 | 1 | 0 | 0 | 0 | 1 |
| 001 | 1 | 1 | 1 | **1** | **1** |
| 010 | 1 | 0 | 0 | 0 | 0 |
| 011 | 1 | 0 | 1 | 0 | 0 |
| 100 | 0 | 1 | 1 | 0 | 1 |
| 101 | 0 | 1 | 0 | 0 | 1 |
| 110 | 0 | 1 | 1 | 0 | 0 |
| 111 | 1 | 1 | 0 | 0 | 0 |

# Tema 5

# Lógica de primer orden

**Ejercicio 5.1** Traduce estas fórmulas del lenguaje natural al lenguaje de la lógica de primer orden. Usa el predicado Loves, donde Loves(x,y) quiere decir "x ama a y":

a) "Hay alguien que ama a todo el mundo".

b) "Hay alguien que ama a al menos una persona".

c) "Hay alguien que ama a algún otro".

d) "Todos se aman mutuamente".

e) "Hay alguien que es amado por todos".

f) "Hay alguien a quien todos aman".

g) "Todo el mundo tiene a alguien que lo ama".

**Solución**

a) "Hay alguien que ama a todo el mundo".

$\exists x\ \forall y\ Loves(x, y)$

b) "Hay alguien que ama a al menos una persona".

$\exists x\ \exists y\ Loves(x, y)$

c) "Hay alguien que ama a algún otro".

$\exists x\ \exists y\ Loves(x,y) \wedge x \neq y$

d) "Todos se aman mutuamente".

$\forall x\ \forall y\ Loves(x,y)$

e) "Hay alguien que es amado por todos".

$\exists x\ \forall y\ Loves(y,x)$

f) "Hay alguien a quien todos aman".

$\exists x\ \forall y\ Loves(y,x)$

g) "Todo el mundo tiene a alguien que lo ama".

$\forall x\ \exists y\ Loves(y,x)$

**Ejercicio 5.2** Considera la siguiente ontología para las listas:

a) La función Insert(x,y) inserta el elemento x al principio de la lista y. Por ejemplo, la lista (A, B, C) se modela mediante este término: Insert(A, Insert(B, Insert(C, EmptyList))) donde EmptyList es un símbolo de constante que representa a la lista vacía.

b) La función Last(x) devuelve el último elemento de una lista vacía. Devuelve la constante ListError para una lista vacía.

c) Los axiomas de la ontología son los siguientes:

  I La lista vacía no tiene último elemento.

  II El último elemento de una lista con un solo elemento es ese elemento.

  III El último elemento de una lista y es también el último elemento de cualquier lista construida insertando un elemento al principio de y.

Traduce los axiomas de la ontología al lenguaje de la lógica de primer orden, y a continuación explicar cómo encontrarías el último elemento de la lista (A, B, C, D) con la ayuda de un demostrador de teoremas.

**Solución**

Los axiomas de la ontología se expresan en la lógica de primer orden como sigue:

i) $Last(EmptyList) = ListError$

ii) $\forall x Last(Insert(x, EmptyList)) = x$

iii) $\forall x \forall y \; y \neq EmptyList \Rightarrow Last(Insert(x, y)) = Last(y)$

El último elemento de la lista (A, B, C, D) puede hallarse con ayuda de un demostrador de teoremas considerando i) – iii) como la KB, y a continuación preguntando lo siguiente al demostrador:

ASKVARS(KB, x=Last(Insert(A, Insert(B, Insert(C, Insert(D, EmptyList))))))

La salida del demostrador debería ser que la fórmula anterior se infiere de la KB con la lista de ligaduras x/D, que indica que la constante D es el último elemento de la lista.

**Ejercicio 5.3** Los axiomas de Peano, también conocidos como los axiomas de Dedekind–Peano o los postulados de Peano postulates, son un conjunto de axiomas para los números naturales presentados por el matemático italiano del siglo XIX Giuseppe Peano. Han sido empleados prácticamente sin cambios hasta hoy. Tu tarea es traducir un subconjunto de ellos y la definición de la suma al lenguaje de la lógica de primer orden. A continuación explica cómo demostrar que la suma es conmutativa con la ayuda de un demostrador de teoremas. Por último, explica cómo encontrar el resultado de la suma 2+3 con un demostrador de teoremas.

a) 0 es un número natural.

b) Para cualquier número natural $n$, su sucesor $Suc(n)$ también es un número natural.

c) Para cualquier número natural $n$, $Suc(n) = 0$ es falso. Es decir, no hay ningún número natural cuyo sucesor sea 0.

d) Para cualesquiera números naturales $m$ y $n$, si $Suc(m) = Suc(n)$, entonces $m = n$. Es decir, $Suc$ es una inyección.

e) La suma se define mediante las siguientes ecuaciones:
$a + 0 = a$
$a + Suc(b) = Suc(a + b)$

**Solución**

Los axiomas y la definición se traducen al lenguaje de la lógica de primer orden como sigue:

i) Esto quiere decir que necesitamos una constante Zero. No hace falta ninguna fórmula de primer orden.

ii) Esto quiere decir que necesitamos una función Suc. No hace falta ninguna fórmula de primer orden.

iii) $\forall n Suc(n) \neq 0$

iv) $\forall m\ \forall n Suc(m) = Suc(n) \Rightarrow m = n$

v) $[\forall a\ Sum(a, Zero) = a] \wedge [\forall a \forall b Sum(a, Suc(b)) = Suc(Sum(a, b))]$

Para demostrar que la suma es conmutativa con ayuda de un demostrador de teoremas, consideramos iii)-v) como la KB, y a continuación le preguntamos al demostrador si la fórmula de la lógica de primer orden que expresa la conmutatividad de la suma es válida:

$$ASK(KB, \forall a \forall b Sum(a, b) = Sum(b, a))$$

El resultado de la suma 2+3 se puede hallar preguntándole lo que sigue al demostrador:

ASKVARS( KB, x=Sum(Suc(Suc(Zero)), Suc(Suc(Suc(Zero)))) )

La salida del demostrdor debería ser que la fórmula anterior se infiere de la KB con la lista de ligaduras { x / Suc(Suc(Suc(Suc(Suc(Zero))))) }, que indica que 2+3=5.

**Ejercicio 5.4** Un espacio vectorial sobre un cuerpo $F$ es un conjunto $V$ junto con dos operaciones binarias que satisfacen los axiomas que se listan más abajo. Tu primera tarea es traducir estos axiomas al lenguaje de la lógica de primer orden. A continuación explica cómo demostrarías que la siguiente propiedad $(-v) + (-w) = -(v + w)$ se cumple con ayuda de un demostrador de teoremas:

a) Asociatividad de la suma: $u + (v + w) = (u + v) + w$

b) Conmutatividad de la suma: $u + v = v + u$

c) Elemento neutro de la suma: existe un elemento $0 \in V$, llamado el vector nulo, de tal manera que $v + 0 = v$ para cualquier $v \in V$.

d) Elementos inversos respecto a la suma: para todo $v \in V$, existe un elemento $-v \in V$, llamado el inverso aditivo de v, de tal manera que $v + (-v) = 0$.

e) Distributividad del producto escalar respecto a la suma: $a(u + v) = au + av$.

f) Distributividad del producto escalar respecto a la suma en el cuerpo: $(a + b)v = av + bv$.

g) Compatibilidad del producto escalar con el producto en el cuerpo: $a(bv) = (ab)v$.

h) Elemento neutro del producto escalar $1v = v$, donde 1 denota el elemento neutro de la multiplicación en $S$.

**Solución**

Los axiomas se traducen al lenguaje de la lógica de primer orden como sigue:

a) Asociatividad de la suma: $u + (v + w) = (u + v) + w$

$\forall u\, \forall v\, \forall w\, Vector(u) \,\wedge\, Vector(v) \,\wedge\, Vector(w) \Rightarrow$
$Sum(u, Sum(v, w)) = Sum(Sum(u, v), w)$

b) Conmutatividad de la suma: $u + v = v + u$

$\forall u\, \forall v\, Vector(u) \,\wedge\, Vector(v) \Rightarrow Sum(u, v) = Sum(v, u)$

c) Elemento neutro de la suma: existe un elemento $0 \in V$, llamado el vector nulo, de tal manera que $v + 0 = v$ para cualquier $v \in V$.

$\forall v\ Vector(v) \Rightarrow Sum(v, ZeroVector) = v$

d) Elementos inversos respecto a la suma: para todo $v \in V$, existe un elemento $-v \in V$, llamado el inverso aditivo de v, de tal manera que $v + (-v) = 0$.

$\forall v\ Vector(v) \Rightarrow Sum(v, Inverse(v)) = ZeroVector$

e) Distributividad del producto escalar respecto a la suma: $a(u+v) = au + av$.

$\forall v\ Vector(v) \Rightarrow Sum(v, Inverse(v)) = ZeroVector$

f) Distributividad del producto escalar respecto a la suma en el cuerpo: $(a + b)v = av + bv$.

$\forall v\ \forall a\ \forall b\ Vector(v)\ \wedge\ Scalar(a) \Rightarrow$

$Product(a, Sum(u, v)) = Sum(Product(a, u), Product(a, v))$

g) Compatibilidad del producto escalar con el producto en el cuerpo: $a(bv) = (ab)v$.

$\forall v\ \forall a\ \forall b\ Vector(v)\ \wedge\ Scalar(a) \Rightarrow$

$Product(a, Product(b, v)) = Product(FieldProduct(a, b), v)$

h) Elemento neutro del producto escalar $1v = v$, donde 1 denota el elemento neutro de la multiplicación en $S$.

$\forall v\ Vector(v) \Rightarrow Product(ScalarOne, v) = v$

Para demostrar que (-v) + (-w) = -(v+w), consideramos i)-viii) como la KB, y a continuación le preguntamos al demostrador de teoremas si la fórmula de la lógica de primer orden que expresa la propiedad indicada es válida:

$$ASK(KB, \forall v\ \forall w\ Vector(v)\ \wedge\ Vector(w) \Rightarrow$$
$$Sum(Inverse(v), Inverse(w)) = Inverse(Sum(v, w)))$$

**Ejercicio 5.5** Traduce las afirmaciones siguientes a fórmulas de la lógica de primer orden. Se trata de la paradoja del barbero de Bertrand Russell; se da el caso de que ambas afirmaciones no pueden ser ciertas a la vez en ninguna interpretación. ¿Cómo demostrarías esto con ayuda de un demostrador de teoremas?

a) Cualquiera que no se afeite a si mismo debe ser afeitado por el barbero (se supone qu solamente hay un barbero).

b) Aquel a quien el barbero afeite, no se afeita a si mismo.

**Solución**

Las afirmaciones se expresan en la lógica de primer orden como sigue, entendiendo que Shaves(x,y) indica que x afeita a y, y que Barber es el barbero:

i) $\forall x \neg Shaves(x,x) \Rightarrow Shaves(Barber, x)$

ii) $\forall x Shaves(Barber, x) \Rightarrow \neg Shaves(x,x))$

Para demostrar que ambas condiciones no pueden cumplirse al mismo tiempo, usamos una KB vacía y preguntamos al demostrador de teoremas si al negación de la conjunción de ambas condiciones es válida (la respuesta debe ser “sí”):

$$ASK(KB, \neg([\forall x \neg Shaves(x,x) \Rightarrow Shaves(Barber, x)] \wedge$$
$$[\forall x\ Shaves(Barber, x) \Rightarrow \neg Shaves(x,x))]))$$

**Ejercicio 5.6** Traduce al español las siguientes fórmulas de la lógica de primer orden, y determina cuáles de ellas representan afirmaciones verdaderas cuando se interpretan en el conjunto de los números reales, $R$.

a) $\neg \forall x \quad x \neq 0$

b) $\exists x \quad x = x^2 \Rightarrow x < 0$

c) $\forall x \forall y \quad x > y \vee y > x$

d) $\forall x \exists y \quad x > y \Rightarrow x > y^2$

e) $\exists x \forall y \quad x + y = x$

f) $\exists x \forall y \quad x + y = y$

g) $\exists x \forall y \quad x > y \vee -x > y$

h) $\exists x \forall y \quad x > y \vee \neg(x > y)$

i) $\exists x \forall y \quad y > x \Rightarrow y^2 > x$

j) $\exists x \forall y \quad x > y \Rightarrow x > y^2$

k) $\forall x \exists y \forall z \quad xy = yz$

l) $\exists x \forall y \exists z \quad (x + y)z = 1$

m) $\forall x \forall y \quad x > y \Rightarrow \exists z(x > z \wedge z > y)$

n) $\forall x \exists z \forall y \quad x > z \Rightarrow z > y$

**Solución**

a) $\neg \forall x \quad x \neq 0$

No todo número real es distinto de 0. Verdadero.

b) $\exists x \quad x = x^2 \Rightarrow x < 0$

Existe un número real tal que, si ese número es igual a su cuadrado, entonces es negativo. Verdadero: sea cualquier x tal que $x \neg x^2$. Entonces el antecedente es falso, con lo cual la implicación es verdadera.

c) $\forall x \forall y \quad x > y \vee y > x$

Para todo par de números reales x e y, uno de ellos es menor que el otro. Falso: tómese x = y.

d) $\forall x \exists y \quad x > y \Rightarrow x > y^2$

Para todo real x existe un real y tal que si x es mayor que y entonces también es mayor que el cuadrado de y. Verdadero: dado x tómese y = x.

e) $\exists x \forall y \quad x + y = x$

Hay un número real x que puede sumarse a cualquier número real y para obtener de nuevo x. Falso.

f) $\exists x \forall y \quad x + y = y$

Hay un número real x que puede sumarse a cualquier número real y para obtener de nuevo y. Verdadero: tómese x = 0.

g) $\exists x \forall y \quad x > y \vee -x > y$

Hay un número real x tal que, para todo número real y, x es mayor que y o –x es mayor que y. Falso.

h) $\exists x \forall y \quad x > y \vee \neg(x > y)$

Hay un número real x tal que para todo número real y, x es mayor que y o no lo es. Verdadero: ley del tercero excluido.

i) $\exists x \forall y \quad y > x \Rightarrow y^2 > x$

Hay un real x tal que todo real y mayor que x tiene un cuadrado que también es mayor que x. Verdadero: tómese por ejemplo x = 1.

j) $\exists x \forall y \quad x > y \Rightarrow x > y^2$

Hay un real x tal que todo real y menor que x tiene un cuadrado que también es menor que x. Falso.

k) $\forall x \exists y \forall z \quad xy = yz$

Para cualquier número real x, hay algún número real y tal que xy es igual que yz para todos los números reales z. Verdadero: dado x, tómese y = 0.

l) $\exists x \forall y \exists z \quad (x + y)z = 1$

Hay un número real x tal que, dado cualquier número real y, (x + y)z es igual a 1 para algún número real z. Falso: dado x, tómese y = –x.

m) $\forall x \forall y \quad x > y \Rightarrow \exists z(x > z \wedge z > y)$

Entre cualquier par de números reales distintos existe otro número real. Verdadero: R es denso.

n) $\forall x \exists z \forall y \quad x > z \Rightarrow z > y$

Dado cualquier número real x, existe algún número real z tal que si x es mayor que z entonces z será mayor que cualquier número real. Verdadero: dado x, tómese z = x.

**Ejercicio 5.7** Traduce este argumento del lenguaje natural al lenguaje de la lógica de primer orden:

*A todos los perros les gusta la carne.*

*Los perros comen todo lo que les gusta.*

*Stubby es un perro.*

---

*Por consiguiente, Stubby come carne.*

**Solución**

$\forall x\ Dog(x) \Rightarrow Likes(x, Meat)$

$\forall x, y\ Dog(x)\ \wedge\ Likes(x, y) \Rightarrow Eats(x, y)$

$Dog(Stubby)$

$\models$

$Eats(Stubby, Meat)$

**Ejercicio 5.8** Traduce este argumento del lenguaje natural al lenguaje de la lógica de primer orden:

*Uno de los alumnos estudia.*

*Los alumnos estudian y van a la biblioteca.*

*En la biblioteca no hay futbolistas.*

*Juan es futbolista.*

---

*Por consiguiente, Juan no estudia.*

**Solución**

$\exists x\ [Alumno(x)\ \wedge\ Estudia(x)]$

$\forall x\ [Alumno(x) \Rightarrow Estudia(x)\ \wedge\ Biblioteca(x)]$

alternativa: $\forall x\ [Alumno(x)) \Rightarrow Estudia(x)\ \wedge\ Va(x, Biblioteca)]$

$\neg\ \exists x\ [Biblioteca(x)\ \wedge\ Futbolista(x)]$

alternativa: $\forall x\ [Biblioteca(x)) \Rightarrow \neg Futbolista(x)]$

$Futbolista(Juan)$

$\models$

$\neg Estudia(Juan)$

**Ejercicio 5.9** Traduce este argumento del lenguaje natural al lenguaje de la lógica de primer orden:

*Todos los estudiantes aprobaron Historia o Biología.*

*Manuel es un estudiante.*

*Manuel no aprobó Biología.*

---

*Por consiguiente, al menos un estudiante aprobó Historia.*

**Solución**

Solución

**Ejercicio 5.10** Traduce este argumento del lenguaje natural al lenguaje de la lógica de primer orden:

*Si un inspector sospecha que alguien roba, dicha persona es interrogada.*
*Clouseau es un inspector.*
*Lupin no es interrogado.*

---

*Por consiguiente, Clouseau no sospecha que Lupin haya robado.*

**Solución**

$\forall x, y\ Inspector(x)\ \wedge\ Suspects(x, y) \Rightarrow Questioned(y)$
$Inspector(Clouseau)$
$\neg\ Questioned(Lupin)$
$\models$
$\neg\ Suspects(Clouseau, Lupin)$

**Ejercicio 5.11** Traduce este argumento del lenguaje natural al lenguaje de la lógica de primer orden:

*Todos los estudiantes aprobaron Historia o Biología.*
*Manuel es un estudiante.*
*Manuel no aprobó Biología.*

---

*Al menos un estudiante aprobó Historia.*

**Solución**

$\forall x\ (Estudiante(x) \Rightarrow (Aprobo(x, Historia)\ \vee\ Aprobo(x, Biologia)))$
$Estudiante(Manuel)$
$\neg Aprobo(Manuel, Biologia)$
$\models$
$\exists x(Estudiante(x)\ \wedge\ Aprobo(x, Historia))$

**Ejercicio 5.12** Traduce este argumento del lenguaje natural al lenguaje de la lógica de primer orden:

*Si un policía sospecha que alguien roba, esa persona es interrogada.*
*Manolo es un policía.*
*Juan es interrogado.*
*Ana no es interrogado.*

---

*Manolo sospecha que Juan haya robado y no sospecha de Ana.*

**Solución**

$\forall x, y\ Policia(x)\ \wedge\ Sospecha(x, y) \Rightarrow Interrogado(y)$
$Policia(Manolo)$
$Interrogado(Juan)$
$\neg Interrogado(Ana)$
$\models$
$Sospecha(Manolo, Juan)\ \wedge\ \neg\ Sospecha(Manolo, Ana)$

**Ejercicio 5.13** Considérense los predicados de tipo Cocinero/1, Empresario/1 y Restaurante/1, así como el predicado Tiene/2. Se pide formalizar el siguiente argumento expresado en lenguaje natural mediante un lenguaje de predicados de primer orden.

*Chicote es cocinero y tiene el restaurante "El puchero eco-lógico".*
*Hay un empresario que tiene un restaurante y no es cocinero.*
*Todo cocinero empresario tiene un restaurante.*

---

*Por tanto, Chicote es empresario.*

**Solución**

$Cocinero(Chicote)\ \wedge\ Restaurante(EPEL)\ \wedge\ Tiene(Chicote, EPEL)$
$\exists x\ \exists y\ Empresario(x)\ \wedge\ Restaurante(y)\ \wedge\ Tiene(x, y)\ \wedge\ \neg\, Cocinero(x)$
$\forall x\ \exists y\ Cocinero(x)\ \wedge\ Empresario(x) \Rightarrow Restaurante(y)\ \wedge\ Tiene(x, y)$
$\models$
$Empresario(Chicote)$

**Ejercicio 5.14** Traduce este argumento del lenguaje natural al lenguaje de la lógica de primer orden:

*O Poirot es un genio o es un fraude.*
*Si alguien sabe como resolver un caso difícil, entonces es un genio.*
*Poirot sabe como resolver el caso del Orient Express.*

---

*Por consiguiente, si el caso del Orient Express es difícil, entonces Poirot no es un fraude.*

**Solución**

$(Genio(Poirot) \ \vee \ Fraude(Poirot)) \ \wedge \ (\neg \, Genio(Poirot) \vee \neg \, Fraude(Poirot))$

$\forall x, y \; Caso(x) \ \wedge \ Dificil(x) \ \wedge \ Resuelve(y, x) \Rightarrow Genio(y)$

$Caso(OrientExpress) \wedge Resuelve(Poirot, OrientExpress)$

$\models$

$Dificil(OrientExpress) \Rightarrow \neg Fraude(Poirot)$

**Ejercicio 5.15** Formaliza el siguiente argumento expresado en lenguaje natural mediante un lenguaje de predicados de primer orden:

*Cualquiera que sepa latín o griego es sabio.*
*Todos los sabios son inteligentes y trabajadores.*
*Juan no es trabajador.*

---

*Por consiguiente, Juan no sabe latín.*

**Solución**

$\forall x \; [Knows(x, Latin) \vee Knows(x, Greek)] \Rightarrow Wise(x)$

$\forall x \; Wise(x) \Rightarrow Intelligent(x) \wedge HardWorking(x)$

$\neg HardWorking(Juan)$

$\models$

$\neg Knows(Juan, Latin)$

**Ejercicio 5.16** Formaliza el siguiente argumento expresado en lenguaje natural mediante un lenguaje de predicados de primer orden:

*Si alguien es un asesino, entonces Señorita Marple no confía en esa persona.*
*Juan o Eduardo es un asesino.*
*Señorita Marple confía en Juan.*

---

*Por lo tanto, Eduardo es un asesino.*

**Solución**

$\forall x\ Killer(x) \Rightarrow \neg Trust(MissMarple, x)$

$Killer(Jack)\ \vee\ Killer(Edward)$

$Trust(MissMarple, Jack)$

$\models$

$Killer(Edward)$

**Ejercicio 5.17** Formule el siguiente argumento del lenguaje natural a la lógica de primer orden:

*Las personas que vuelan son superhéroes.*
*No todos los superhéroes pueden volar.*
*Ironman y Supergirl son personas que pueden volar.*

---

*Ironman es un superhéroe..*

**Solución**

$\forall x\ Person(x)\ \wedge\ Flies(x) \Rightarrow Superhero(x)$

$\neg\forall x\ Superhero(x) \Rightarrow Flies(x)$

$Person(Ironman) \wedge Person(Supergirl) \wedge Flies(Ironman) \wedge Flies(Supergirl)$

$\models$

$Superhero(Ironman)$

**Ejercicio 5.18** Formaliza el siguiente argumento expresado en lenguaje natural mediante un lenguaje de predicados de primer orden:

*Todos los estudiantes son felices si aprueban una asignatura.*
*Nadie aprueba una asignatura sin hacer los ejercicios de la asignatura.*
*Yo soy estudiante y he hecho los ejercicios de Sistemas Inteligentes.*

---

*Por consiguiente, Yo soy feliz.*

**Solución**

$\forall x\ Student(x) \wedge [\exists y\ Course(y) \wedge Passes(x,y)] \Rightarrow Happy(x)$

$\forall x,y\ Student(x) \wedge Course(y) Passes(x,y) \Rightarrow DoesHomework(x,y)$

$Student(Me)\ \wedge\ Course(IS)\ \wedge\ DoesHomework(Me,IS)$

$\models$

$Happy(Me)$

**Ejercicio 5.19** Formaliza el siguiente argumento expresado en lenguaje natural mediante un lenguaje de predicados de primer orden:

*Si cada ascendiente del ascendiente de un individuo es también ascendiente de ese mismo individuo y ningún individuo es ascendiente de sí mismo, entonces hay alguien sin ascendientes.*

**Solución**

Ascendiente (x,y) significa que x tiene como ascendiente a y

$$[(\forall x\ \forall y\ \forall z\ (Ascendiente(x,y) \wedge\ Ascendiente(y,z)))))\ Ascendiente(x,z))$$
$$\wedge\ \forall x\ \neg Ascendiente(x,x)] \Rightarrow \exists x\ \forall y\ \neg Ascendiente(x,y)$$

# Tema 6

# Planificación

**Ejercicio 6.1** Las torres de Hanoi son un juego matemático. El juego tiene tres palos, y cierta cantidad de discos de distintos tamaños que pueden ensartarse en cualquier palo. El juego empieza con los discos apilados en orden ascendente sobre uno de los palos, con el más pequeño en lo alto, formando una figura cónica. El objetivo es mover toda la pila a otro palo, obedeciendo estas reglas:

i Sólo se puede mover un disco cada vez.

ii Cada movimiento consiste en coger el disco de arriba de uno de los palos y colocarlo en otro palo, quedando encima de los que pudiera haber anteriormente.

iii No se puede colocar un disco encima de otro más pequeño.

Se pide:

a) Escribe el dominio de planificación de las torres de Hanoi. Emplea tres predicados: el predicado unario *Clear* y los predicados binarios *On* y *Smaller*. Basta con una sola acción *Move*.

b) Especifica el siguiente problema de tres discos: *D3*, *D2* y *D1* (listados desde el más grande hasta el más pequeño).

**Solución**

a) Escribe el dominio de planificación de las torres de Hanoi. Emplea tres predicados: el predicado unario *Clear* y los predicados binarios *On* y *Smaller*. Basta con una sola acción *Move*.

**Predicados**:

*Clear(x)* quiere decir que no hay ningún objeto sobre x, donde x puede ser un palo o un disco.

*On(x,y)* quiere decir que x está sobre y. Nótese que x debe ser un disco, mientras que y puede ser un palo o un disco.

*Smaller(x,y)* quiere decir que x es más pequeño que y, es decir, x puede colocarse encima de y. Nótese que x debe ser un disco, mientras que y puede ser un palo o un disco.

**Acciones**:

*Action( **Move**(disc,orig,dest),*
*PRECOND: Clear(disc) ∧ On(disc,orig) ∧ Clear(dest) ∧ Smaller(disc,dest)*
*EFFECT: ¬ Clear(dest) ∧ ¬ On(disc,orig) ∧ Clear(orig) ∧ On(disc,dest) )*

b) Especifica el siguiente problema de tres discos : *D3*, *D2* y *D1* (listados desde el más grande hasta el más pequeño).

**Estado inicial**:

Clear(D1) ∧ Clear(B) ∧ Clear(C) ∧
On(D1,D2) ∧ On(D2,D3) ∧ On(D3,A) ∧
Smaller(D1,D2) ∧ Smaller(D1,D3) ∧ Smaller(D2,D3) ∧
Smaller(D1,A) ∧ Smaller(D1,B) ∧ Smaller(D1,C) ∧
Smaller(D2,A) ∧ Smaller(D2,B) ∧ Smaller(D2,C) ∧
Smaller(D3,A) ∧ Smaller(D3,B) ∧ Smaller(D3,C)

**Estado objetivo**:

Clear(D1) ∧ Clear(A) ∧ Clear(B) ∧
On(D1,D2) ∧ On(D2,D3) ∧ On(D3,C)

**Ejercicio 6.2** Tenemos un robot de reparto de paquetes que puede mover paquetes entre la Escuela de Informática (*Computer Science*), la de Telecomunicaciones (*Telecommunications*) y la Facultad de Medicina (*Medicine*). Supondremos que el robot puede transportar una cantidad ilimitada de carga. Cuando viaja entre la Facultad de Medicina y los otros dos centros, debe salir al exterior. En cambio, puede viajar bajo techo entre ambas escuelas. Si el robot sale al exterior mientras llueve, se mojará a menos que lleve un paraguas. Puede conseguir un paraguas en la Escuela de Informática. Se pide:

a) Escribe el dominio de planificación.

b) Especifica el siguiente problema. Hay cuatro paquetes: *P1* (está en Telecomunicaciones y debe ser entregado en Informática), *P2* (está en Informática y debe entregarse en Telecomunicaciones), *P3* (está en Medicina, y debe enviarse Telecomunicaciones), y *P4* (está en Informática y hay que enviarlo a Medicina). El robot está en Telecomunicaciones, y está lloviendo. Debemos entregar todos los paquetes manteniendo seco el robot.

**Solución**

a) Escribe el dominio de planificación.

**Predicados**:

*At(x,y)* quiere decir que el objeto x está en la ubicación y. Nótese que y debe ser uno de los tres centros, mientras que x puede ser el robot, un paquete o el paraguas.

*On(x,y)* quiere decir que x está siendo acarreado por y. Nótese que y solamente puede ser el robot, mientras que x can be either a package or the umbrella.

*DryPath(x,y)* quiere decir que el camino desde x hasta y está seco.

*Wet(x)* quiere decir que x está mojado.

*Robot(x)* quiere decir que x es un robot.

*Cargo(x)* quiere decir que x es carga, es decir, un paquete o el paraguas.

*Umbrella(x)* quiere decir que x es un paraguas.

**Acciones**:

*Action( **Load**(robot,cargo,location),*
*PRECOND: At(robot,location) ∧ At(cargo,location) ∧ Robot(robot) ∧ Cargo(cargo)*
*EFFECT: ¬ At(cargo,location) ∧ On(cargo,robot) )*

*Action( **UnLoad**(robot,cargo,location),*
*PRECOND: On(cargo,robot) ∧ At(robot,location) ∧ Robot(robot) ∧ Cargo(cargo)*
*EFFECT: At(cargo,location) ∧ ¬On(cargo,robot) )*

*Action( **MoveDry**(robot,umbrella,orig,dest),*
*PRECOND: At(robot,orig) ∧ Robot(robot) ∧ Umbrella(umbrella)*
*∧ DryPath(orig,dest) ∧ On(umbrella,robot)*
*EFFECT: ¬At(robot,orig) ∧ At(robot,dest) )*

*Action( **MoveWet**(robot,umbrella,orig,dest),*
*PRECOND: At(robot,orig) ∧ Robot(robot) ∧ Umbrella(umbrella)*
*∧ ¬ DryPath(orig,dest) ∧ ¬ On(umbrella,robot)*
*EFFECT: ¬At(robot,orig) ∧ At(robot,dest) ∧ Wet(robot) )*

b) Especifica el siguiente problema. Hay cuatro paquetes: *P1* (está en Telecomunicaciones y debe ser entregado en Informática), *P2* (está en Informática y debe entregarse en Telecomunicaciones), *P3* (está en Medicina, y debe enviarse Telecomunicaciones), y *P4* (está en Informática y hay que enviarlo a Medicina). El robot está en Telecomunicaciones, y está lloviendo. Debemos entregar todos los paquetes manteniendo seco el robot.

**Estado inicial**:

At(P1,Telecommunications) ∧ At(P2,ComputerScience) ∧
At(P3,Medicine) ∧ At(P4,ComputerScience) ∧
At(TheUmbrella,ComputerScience) ∧
At(TheRobot,Telecommunications) ∧
Robot(TheRobot) ∧ Umbrella(TheUmbrella) ∧
Cargo(P1) ∧ Cargo(P2) ∧ Cargo(P3) ∧ Cargo(P4) ∧
Cargo(TheUmbrella) ∧

DryPath(Telecommunications,ComputerScience) ∧
DryPath(ComputerScience,Telecommunications)

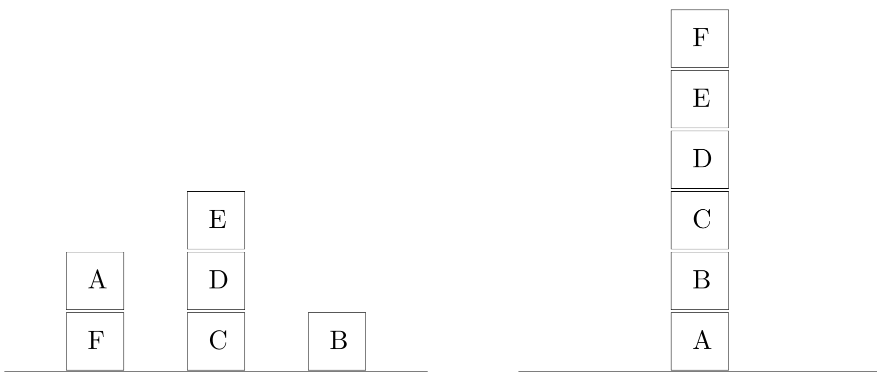
**Estado objetivo**:

At(P1,ComputerScience) ∧ At(P2,Telecommunications) ∧
At(P3,Telecommunications) ∧ At(P4,Medicine) ∧ ¬ Wet(TheRobot)

**Ejercicio 6.3** Se le asigna a un robot la tarea de colocar varios bloques formando una torre. Se pide:

a) Escribe el dominio de planificación .

b) Especifica el siguiente problema .




Estado inicial    Estado objetivo

**Solución**

a) Escribe el dominio de planificación .

**Predicados**:

*On(x,y)* quiere decir que el objeto x está directamente encima del objeto y. Nótese que y puede ser la mesa o un bloque, mientras que x debe ser un bloque.

*Clear(x)* quiere decir que no hay nada encima del bloque x.

*Block(x)* quiere decir que x es un bloque.

*Table(x)* quiere decir que x es una mesa.

**Acciones**:

*Action( **PutOn**(block,orig,dest),*
*PRECOND: Clear(block) ∧ Clear(dest) ∧ Block(dest) ∧ On(block,orig)*
*EFFECT: ¬On(block,orig) ∧ On(block,dest) ∧ Clear(orig) ∧ ¬Clear(dest) )*

*Action( **PutTable**(block,orig,table),*
*PRECOND: Clear(block) ∧ Block(orig) ∧ On(block,orig) ∧ Table(table)*
*EFFECT: ¬On(block,orig) ∧ On(block,table) ∧ Clear(orig) )*

b) Especifica el siguiente problema .

**Estado inicial**:

Block(A) ∧ Block(B) ∧ Block(C) ∧ Block(D) ∧ Block(E) ∧ Block(F)
∧ Table(T) ∧ On(F,T) ∧ On(A,F) ∧
On(C,T) ∧ On(D,C) ∧ On(E,D) ∧
On(B,T) ∧ Clear(A) ∧ Clear(E) ∧ Clear(B)

**Estado objetivo**:

On(F,E) ∧ On(E,D) ∧ On(D,C) ∧
On(C,B) ∧ On(B,A) ∧ On(D,C) ∧
Clear(F)

**Ejercicio 6.4** Un dominó es una pieza rectangular con una línea que divide su cara superior en dos mitades cuadradas. Cada mitad está marcada con cierto número de puntos, o bien está vacía. La tarea consiste en jugar un solitario con una cierta bolsa de dominós. El juego comienza colocando una pieza cualquiera. Dicha pieza es la primera de la línea de juego, una serie de dominós en la que los dominós adyacentes se tocan con valores iguales. El jugador extiende la línea de juego cada vez con un dominó colocado en cualquiera de los dos extremos. El juego termina si todos los dominós disponibles están colocados sobre la mesa, es decir, la línea de juego contiene todos los dominós. Nótese que la bolsa de dominós puede contener algunos repetidos. Se pide:

a) Escribe el dominio de planificación .

b) Especifica el siguiente problema . La bolsa de dominós contiene: 5-0, 5-0, 4-4, 4-6, 6-1, 0-3, 3-4, y 1-0.

**Solución**

a) Escribe el dominio de planificación .

**Predicados**:

*Domino(x,y,z)* quiere decir que hay un dominó x que tiene y puntos en la mitad izquierda y z puntos en la mitad derecha.

*Placed(x,y)* quiere decir que el dominó x está colocado sobre la mesa y.

*Empty(x)* quiere decir que la mesa x está vacía, es decir, no se ha colocado aún ningún dominó sobre ella.

*Leftmost(x,y)* quiere decir que y es el dominó colocado más a la izquierda sobre la mesa x.

*Rightmost(x,y)* quiere decir que y es el dominó colocado más a la derecha sobre la mesa x.

**Acciones**:

*Action( **PlaceFirst**(table,domino,leftside,rightside),*
*PRECOND: Empty(table) ∧ Domino(domino,leftside,rightside)*
*EFFECT: ¬Empty(table) ∧ Placed(domino) ∧ Leftmost(table,domino)*
*∧ Rightmost(table,domino) )*

*Action( **Reverse**(domino,leftside,rightside),*
*PRECOND: ¬Placed(domino) ∧ Domino(domino,leftside,rightside)*
*EFFECT: ¬Domino(domino,leftside,rightside)*
*∧ Domino(domino,rightside,leftside) )*

*Action( **PlaceLeft**(table,leftmost,domino,leftside,rightside,otherside),*
*PRECOND: Leftmost(table,leftmost) ∧ Domino(domino,leftside,rightside)*
*∧ ¬Placed(domino,table) ∧ Domino(leftmost,rightside,otherside)*
*EFFECT: ¬ Leftmost(table,leftmost) ∧ Placed(domino,table)*
*∧ Leftmost(table,domino) )*

*Action( **PlaceRight**(table,rightmost,domino,leftside,rightside,otherside),*
*PRECOND: Rightmost(table,rightmost) ∧ Domino(domino,leftside,rightside)*
*∧ ¬Placed(domino,table) ∧ Domino(rightmost,otherside,leftside)*
*EFFECT: ¬ Rightmost(table,rightmost) ∧ Placed(domino,table)*
*∧ Rightmost(table,domino) )*

b) Especifica el siguiente problema . La bolsa de dominós contiene: 5-0, 5-0, 4-4, 4-6, 6-1, 0-3, 3-4, y 1-0.

**Estado inicial**:
Domino(A,5,0) ∧ Domino(B,5,0) ∧ Domino(C,4,4) ∧
Domino(D,4,6) ∧ Domino(E,6,1) ∧ Domino(F,0,3) ∧
Domino(G,3,4) ∧ Domino(H,1,0) ∧ Empty(Table)

**Estado objetivo**:
Placed(A) ∧ Placed(B) ∧ Placed(C) ∧
Placed(D) ∧ Placed(E) ∧ Placed(F) ∧
Placed(G) ∧ Placed(H)

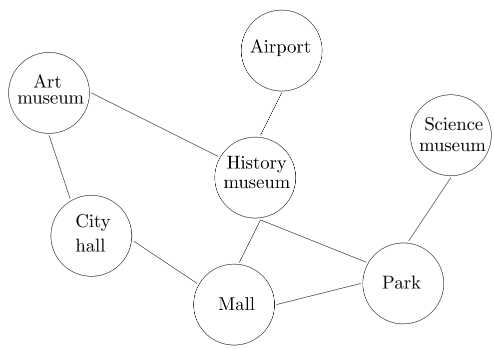
**Ejercicio 6.5** Son las 3PM, y Speedy acaba de llegar al aeropuerto. Quiere visitar los tres museos principales de la ciudad antes de que acabe el día: Arte (Art), Historia (History) y Ciencias (Science). Debido a los atascos de tráfico, las únicas conexiones disponibles son las que se muestran en el grafo de abajo. Cada movimiento desde un lugar a otro adyacente necesita una hora, y los horarios de visita de los museos son los siguientes:

- Museo de Arte: 4PM, 5PM o 6PM.
- Museo de Historia: 4PM, 5PM, 6PM o 7PM.
- Museo de Ciencias: 4PM, 5PM o 6PM.

Esto quiere decir que, por ejemplo, Speedy puede ir del aeropuerto al museo de Historia, llegando allí a las 4PM, que es uno de los momentos en que se puede visitar. A continuación podría ir al centro comercial (Mall), adonde llegaría a las 5PM. Se pide:

a) Escribe el dominio de planificación .

b) Especifica el problema de Speedy .



**Solución**

a) Escribe el dominio de planificación .

**Predicados**:

*Before(x,y)* quiere decir que el tiempo x es el inmediatamente anterior del tiempo y.

*Visited(x)* quiere decir que el museo x ya ha sido visitado.

*Open(x,y)* quiere decir que el museo x está abierto en el tiempo y.

*Now(x)* quiere decir que el tiempo actual es x.

*Location(x)* quiere decir que la ubicación actual del robot es x.

*Path(x,y)* quiere decir que hay un camino desde la ubicación x hasta la ubicación y.

**Acciones**:

*Action( **Move**(now,then,orig,dest),*
*PRECOND: Now(now) ∧ Before(now,then) ∧ Location(orig) ∧ Path(orig,dest) ∧ ¬Open(dest,then)*
*EFFECT: ¬Now(now) ∧ Now(then) ∧ Location(dest) )*

*Action( **Visit**(now,then,orig,dest),*
*PRECOND: Now(now) ∧ Before(now,then) ∧ Location(orig) ∧ Path(orig,dest) ∧ Open(dest,then)*
*EFFECT: ¬Now(now) ∧ Now(then) ∧ Location(dest) ∧ Visited(dest) )*

b) Especifica el problema de Speedy .

**Estado inicial**:

Now(3PM) ∧ Before(3PM,4PM) ∧ Before(4PM,5PM) ∧
Before(5PM,6PM) ∧ Before(6PM,7PM) ∧ Location(Airport) ∧
Path(Airport,History) ∧ Path(History,Airport) ∧
Path(Art,History) ∧ Path(History,Art) ∧
Path(Mall,History) ∧ Path(History,Mall) ∧
Path(Park,History) ∧ Path(History,Park) ∧
Path(Art,City) ∧ Path(City,Art) ∧

Path(Mall,City) ∧ Path(City,Mall) ∧
Path(Mall,Park) ∧ Path(Park,Mall) ∧
Path(Park,Science) ∧ Path(Science,Park) ∧
Open(Art,4PM) ∧ Open(Art,5PM) ∧ Open(Art,6PM) ∧
Open(History,4PM) ∧ Open(History,5PM) ∧
Open(History,6PM) ∧ Open(History,7PM) ∧
Open(Science,4PM) ∧ Open(Science,5PM) ∧ Open(Science,6PM)

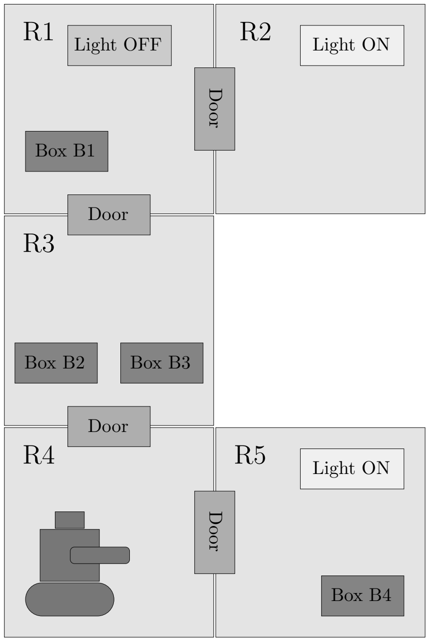
**Estado objetivo**:

Visited(History) ∧ Visited(Art) ∧ Visited(Science)

**Ejercicio 6.6** Shakey es un robot que está diseñado para moverse entre un conjunto de habitaciones conectadas por puertas. Algunas de las habitaciones tienen interruptores de luz, de modo que la luz de la habitación puede encenderse (ON) o apagarse (OFF). Además, hay algunas cajas repartidas por este mundo. Shakey puede empujar una caja de una habitación a otra si hay una puerta que las conecte. Nótese que sólo puede empujar una caja cada vez, y que no puede empujar si está colocado encima de una caja. A fin de accionar el interruptor de luz de una habitación, Shakey debe subirse en una caja que se encuentre en esa habitación. De otra manera no podría alcanzar el interruptor. Se pide:

a) Escribe el dominio de planificación.

b) Especifica el problema de mover todas las cajas a la habitación *R1* y apagar todas las luces, para el siguiente mundo:




N.B.: Shakey fue un robot real que fue desarrollado desde 1966 hasta 1972. En 1970, la revista Life se refirió a él como la "primera persona electrónica" (*the first electronic person*).

**Solución**

a) Escribe el dominio de planificación .

**Predicados**:

*At(x,y)* quiere decir que el objeto x está en la ubicación y. Nótese que y debe ser una habitación, mientras que x puede ser una caja o el robot.

*Door(x,y)* quiere decir que hay una puerta que conecta la habitación x con la habitación y.

*High(x)* quiere decir que el robot x está encima de una caja.

*On(x,y)* quiere decir que x está encima de y. Nótese que y ha de ser una caja, mientras que x ha de ser el robot.

*Switch(x)* quiere decir que hay un interruptor de luz en la habitación x.

*LightOn(x)* quiere decir que la luz de la habitación x está encendida.

*Box(x)* quiere decir que x es una caja.

*Robot(x)* quiere decir que x es un robot.

**Acciones**:

*Action( **Move**(robot,orig,dest),*
*PRECOND: At(robot,orig) ∧ Robot(robot) ∧ ¬High(robot) ∧ Door(orig,dest)*
*EFFECT: ¬At(robot,orig) ∧ At(robot,dest) )*

*Action( **Push**(robot,box,orig,dest),*
*PRECOND: At(robot,orig) ∧ At(box,orig) ∧ Robot(robot) ∧ Box(box) ∧ ¬High(robot) ∧ Door(orig,dest)*
*EFFECT: ¬At(robot,orig) ∧ ¬At(box,orig) ∧ At(robot,dest) ∧ At(box,dest) )*

*Action( **ClimbUp**(robot,box,room),*
*PRECOND: At(robot,room) ∧ At(box,room) ∧ Robot(robot) ∧ Box(box) ∧ ¬High(robot)*
*EFFECT: High(robot) ∧ On(robot,box) )*

*Action( **ClimbDown**(robot,box,room),*
*PRECOND: At(robot,room) ∧ At(box,room) ∧ Robot(robot) ∧ Box(box) ∧ On(robot,box)*
*EFFECT: ¬High(robot) ∧ ¬On(robot,box) )*

*Action( **SwitchOn**(robot,box,room),*
*PRECOND: At(robot,room) ∧ At(box,room) ∧ Robot(robot) ∧ Box(box) ∧ On(robot,box) ∧ Switch(room) ∧ LightOn(room)*
*EFFECT: ¬LightOn(room) )*

*Action( **SwitchOff**(robot,box,room),*
*PRECOND: At(robot,room) ∧ At(box,room) ∧ Robot(robot) ∧ Box(box) ∧ On(robot,box) ∧ Switch(room) ∧ ¬LightOn(room)*
*EFFECT: LightOn(room) )*

b) Especifica el problema de mover todas las cajas a la habitación *R1* y apagar todas las luces, para el siguiente mundo:

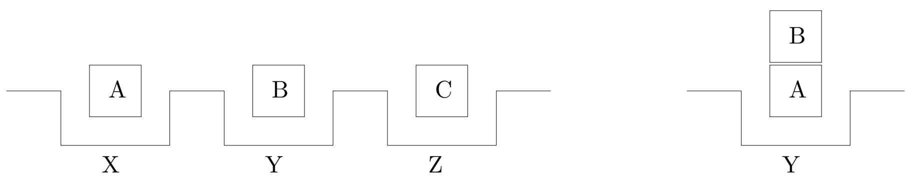
**Estado inicial**:

Robot(Shakey) ∧ Box(B1) ∧ Box(B2) ∧ Box(B3) ∧ Box(B4)
At(Shakey,R4) ∧ At(B1,R1) ∧
At(B2,R3) ∧ At(B3,R3) ∧ At(B4,R5) ∧
Door(R1,R2) ∧ Door(R2,R1) ∧
Door(R1,R3) ∧ Door(R3,R1) ∧
Door(R3,R4) ∧ Door(R4,R3) ∧
Door(R4,R5) ∧ Door(R5,R4) ∧
Switch(R1) ∧ Switch(R2) ∧ Switch(R5) ∧
LightOn(R2) ∧ LightOn(R5)

**Estado objetivo**:

At(B1,R1) ∧ At(B2,R1) ∧ At(B3,R1) ∧ At(B4,R1) ∧
¬LightOn(R1) ∧ ¬LightOn(R2) ∧ ¬LightOn(R5)

**Ejercicio 6.7** El problema del mundo de los bloques simplificado consiste en un robot que tiene la tarea de apilar varios bloques denominados a,b y c de una determinada manera, pero estos bloques sólo pueden situarse en determinadas posiciones fijas llamadas X,Y,Z o bien unos encima de otros. La disposición inicial y final de los bloques es la siguiente:




a) Describir el dominio del mundo de los bloques simplificado especificando las constantes, los predicados y las acciones (pista: sólo es necesaria una acción para resolver óptimamente el problema).

b) Especifica el estado inicial y objetivo descritos anteriormente con tu formulación PDDL del problema.

**Solución**

a) Describir el dominio del mundo de los bloques simplificado especificando las constantes, los predicados y las acciones (pista: sólo es necesaria una acción para resolver óptimamente el problema).

**Constantes**: Bloques: A, B, C Posiciones: X, Y, Z

**Predicados**:
Libre(x): quiere decir que el bloque x o la posición x no tienen a nadie encima.
En(x,y): quiere decir que el bloque x está directamente sobre el bloque y o la posición y.

**Acciones**:

*Action(* ***Mover****(bloq,orig,dest),*
*PRECOND: En(bloq,orig) ∧ Libre(dest) ∧ Libre(orig)*
*EFFECT: ¬ En(bloq,orig) ∧ ¬ Libre(dest) ∧ En(bloq,dest) ∧ Libre(orig))*

b) Especifica el estado inicial y objetivo descritos anteriormente con tu formulación PDDL del problema.

**Estado inicial**:

En(A,X) ∧ En(B,Y) ∧ En(C,Z) ∧ Libre(A) ∧ Libre(B) ∧ Libre(C)

**Estado objetivo**:

En(B,A) ∧ En(A,Y)

**Ejercicio 6.8**

Consideremos el problema de cargar y descargar carga en trenes y transportarla de una estación a otra. Hay tres cargas: C1, C2 y C3. Hay cuatro trenes: T1, T2, T3 y T4. Finalmente, hay cuatro estaciones: S1, S2, S3 y S4. Al principio se tiene que:

a) La carga C1 está en la estación S2, la carga C2 está en la estación S4, y la carga C3 está en la estación S1.

b) El tren T1 está en la estación S2, el tren T2 está en la estación S3, y los trenes T3 y T4 están en la estación S1.

La carga C1 debe enviarse a la estación S4, la carga C2 debe enviarse a la estación S3, y la carga C3 debe enviarse a la estación S2. Por otro lado, las líneas férreas disponibles son las siguientes:

| | S1 | S2 | S3 | S4 |
|---|---|---|---|---|
| S1 | | Sí | Sí | |
| S2 | Sí | | | Sí |
| S3 | Sí | | | |
| S4 | | Sí | | |

Se pide:

a) Definir el dominio de planificación de los trenes de mercancías . Emplea estos predicados: los predicados unarios Train, Station y Cargo; los predicados binarios Railroad, At, e In. Usa estas acciones: Load, Unload y Drive (esta última para llevar un tren de una estación a otra).

b) Especifica el problema indicado anteriormente .

**Solución**

a) Definir el dominio de planificación de los trenes de mercancías . Emplea estos predicados: los predicados unarios Train, Station y Cargo; los predicados binarios Railroad, At, e In. Usa estas acciones: Load, Unload y Drive (esta última para llevar un tren de una estación a otra).

**Predicados**:

At(x,y) quiere decir que el objeto x está en la ubicación y. Nótese que y debe ser una estación, mientras que x puede ser un tren o una carga.

In(x,y) quiere decir que x está dentro de y. Nótese que y solamente puede ser un tren, mientras que x es una carga.

Railroad(x,y) quiere decir que hay una línea férrea desde x hasta y.

Train(x) quiere decir que x es un tren.

Station(x) quiere decir que x es una estación.

Cargo(x) quiere decir que x es carga.

**Acciones**:

*operator **load**(T,C,L)*
*pre: cargo(C), station(L), train(T), at(T,L), at(C,L)*
*post: at(C,L), in(C,T)*

*operator **unload**(T,C,L)*
*pre: cargo(C), train(T), station(L), in(C,T), at(T,L)*
*post: at(C,L), in(C,T)*

*operator **drive**(Train,From,To)*
*pre: train(Train), station(From), station(To), at(Train, From), railroad(From,To)*
*post: at(Train,From), at(Train,To)*

b) Especifica el problema indicado anteriormente .

*start ( train(t1), train(t2), train(t3), train(t4),*
*cargo(c1), cargo(c2), cargo(c3),*
*at(c1,s2), at(c2,s4), at(c3,s1), at(t1,s2), at(t2,s3), at(t3,s1), at(t4,s1),*
*station(s1), station(s2), station(s3), station(s4),*
*railroad(s1,s2), railroad(s1,s3), railroad(s2,s1), railroad(s2,s4), railroad(s3,s1),*
*railroad(s4,s2)*
*)*

*goal(at(c1,s4), at(c2,s3), at(c3,s2))*

La solución al problema sería:

*drive(t1,s2,s4) drive(t3,s1,s2) load(t1,c2,s4) load(t3,c1,s2)*
*drive(t3,s2,s1) drive(t1,s4,s2) drive(t1,s2,s1) drive(t3,s1,s2)*
*load(t4,c3,s1) drive(t3,s2,s4) drive(t1,s1,s3) drive(t4,s1,s2)*
*unload(t1,c2,s3) unload(t3,c1,s4) unload(t4,c3,s2)*

INFORMACIÓN: Plan length: 15

**Ejercicio 6.9** Consideremos el siguiente mapa, donde el agente parte de la posición inicial A2 y debe llegar a la posición destino D2. Las casillas sombreadas en gris, son muros que el agente no puede sobre pasar.

| | | | | |
|---|---|---|---|---|
| D | | GOAL | | |
| C | | | | |
| B | | | | |
| A | | INI | | |
| | 1 | 2 | 3 | 4 |

El estado inicial del problema es definido mediante los siguientes predicados binarios:

- at (coordenadaX,coordenadaY). Agente está en coordenadaX, coordenadaY.
- inc (coordendaOLD,coordenadaNEW). Incrementamos una coordenada X o Y.
- dec (coordendaOLD,coordenadaNEW). Decrementamos una coordenada X o Y.
- camino(coordenadaX,coordenadaY). Es transitable la casilla con coordenadaX, coordenadaY.

```
start (
at(c_2,c_a),
inc (ca,cb),inc(cb,cc),inc(cc,cd),
dec (cb,ca),dec(cc,cb),dec(cd,cc),
inc (c1,c2),inc (c2,c3),inc (c3,c4),
dec (c2,c1),dec (c3,c2),dec (c4,c3),
camino(c1,ca),camino (c2,ca),camino (c3,ca),camino (c4,ca),
camino (c1,cb),camino (c2,cb), camino (c1,cc),
camino (c1,cd),camino (c2,cd),camino (c3,cd),camino (c4,cd)
)
```

y el estado final:

```
goal(
at(c_2,c_d)
)
```

Se pide:

a) Definir las acciones utilizando los predicados binarios definidos en el problema.

- moveUp (PX,PY,BY)
- moveDown (PX,PY,BY)
- moveLeft (PX,PY,BX)
- moveRight (PX,PY,BX)

b) Establezca la solución del problema.

**Solución**

a) Definir las acciones utilizando los predicados binarios definidos en el problema.

- moveUp (PX,PY,BY)

  *operator* ***moveUp*** *(PX,PY,BY)*
  *pre: camino(PX,BY), dec (BY,PY), at(PX,PY)*
  *post: at(PX,PY), at(PX,BY)*

- moveDown (PX,PY,BY)

  *operator* ***moveDown*** *(PX,PY,BY)*
  *pre: camino(PX,BY),inc (BY,PY), at(PX,PY)*
  *post: at(PX,PY), at(PX,BY)*

- moveLeft (PX,PY,BX)

  *operator* ***moveLeft*** *(PX,PY,BX)*
  *pre: camino(BX,PY),dec (PX,BX), at(PX,PY)*
  *post: at(PX,PY), at(BX,PY)*

- moveRight (PX,PY,BX)

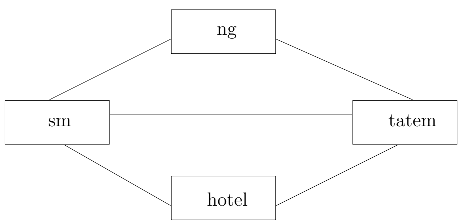
  *operator* ***moveRight*** *(PX,PY,BX)*
  *pre: camino(BX,PY), inc (PX,BX), at(PX,PY)*
  *post: at(PX,PY), at(BX,PY)*

b) Establezca la solución del problema.

*moveLeft(c_2,c_a,c_1)*
*moveUp(c_1,c_a,c_b)*
*moveUp(c_1,c_b,c_c)*
*moveUp(c_1,c_c,c_d)*
*moveRight(c_1,c_d,c_2)*

**Ejercicio 6.10** Planificación de itinerarios. Alice, se encuentra en Londres, acaba de llegar al hotel, son las 13 horas y tiene media jornada para hacer una visita a determinados lugares interesantes de la ciudad. Para completar su día, debe planificar y tener en cuenta tanto los horarios de visita como las conexiones disponibles entre lugares a visitar. Para desplazarse de un lugar a otro adyacente necesita una hora y los horarios de visita son los que se detallan a continuación:

- National Gallery (ng): 17 a 19 horas (incl.)
- Science Museum (sm): 14 a 17 horas (incl.)
- Tate Modern (tatem): 15 a 16 horas (incl.)




Utilice los siguientes predicados para resolver los apartados:

- before(X,Y) – Tiempo X es el inmediatamente anterior del tiempo Y.
- visited(X) – Museo X ya ha sido visitado.
- open(X,Y) – Museo X está abierto en el tiempo Y.
- now(X) – Tiempo actual es X.
- location(X) – Ubicación actual de Alice es X.
- path(X,Y) – Camino desde la ubicación X hasta la ubicación Y.

Se pide:

a) Formula el estado inicial en STRIPS (Alice está en el hotel a las 13h).

b) Formula el estado objetivo en STRIPS, (Alice visite los tres museos).

c) Formula las acciones:

- Move(now,then,orig,dest), Alice se mueve desde orig hasta dest en el periodo de tiempo now hasta then
- Visit(now,then,X), Alice visita museo X en el periodo de tiempo now hasta then

d) Muestre la secuencia de acciones necesarias para planificar la visita de Alice.

**Solución**

a) Formula el estado inicial en STRIPS (Alice está en el hotel a las 13h).

*start (*
*now(h13PM), location(hotel),*
*before(h13PM,h14PM), before(h14PM,h15PM),*
*before(h15PM,h16PM), before(h16PM,h17PM),*
*before(h17PM,h18PM), before(h18PM,h19PM),*
*path(hotel,tatem), path(ng,tatem),*
*path(hotel,sm), path(sm,hotel),*
*path(sm, tatem), path(ng, tatem),*
*path(tatem,ng), path(ng, tatem),*
*path(sm, ng), path(ng, sm),*
*open(tatem,h15PM), open(tatem,h16PM),*
*open(ng,h17PM),open(ng,h18PM),*
*open(ng,h19PM),open(sm,h14PM),*
*open(sm,h15PM),open(sm,h16PM), open(sm,h17PM)*
*)*

b) Formula el estado objetivo en STRIPS, (Alice visite los tres museos).

*goal( visited(sm),visited(ng),visited(tatem) )*

c) Formula las acciones:

- Move(now,then,orig,dest), Alice se mueve desde orig hasta dest en el periodo de tiempo now hasta then

  *operator **move**(Now,Then,Orig,Dest)*
  *pre: location(Orig), path(Orig, Dest),now(Now), before(Now, Then)*
  *post: location(Orig),location(Dest), now(Now),now(Then)*

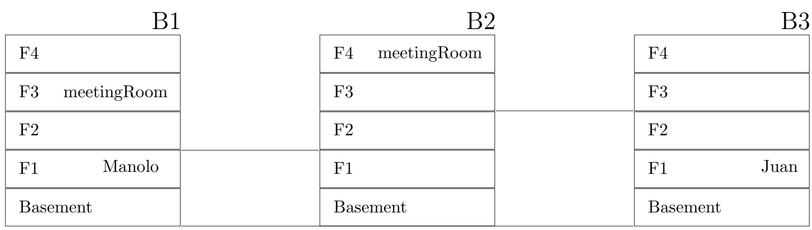
- Visit(now,then,X), Alice visita museo X en el periodo de tiempo now hasta then

  *operator **visit**(Now,Then,X)*
  *pre: now(Now), before(Now, Then), location(X), open(X, Now)*
  *post: now(Now),now(Then), visited(X)*

d) Muestre la secuencia de acciones necesarias para planificar la visita de Alice.

*move(h13PM,h14PM,hotel,sm)*
*visit(h14PM,h15PM,sm)*
*move(h15PM,h16PM,sm,tatem)*
*visit(h16PM,h17PM,tatem)*
*move(h17PM,h18PM,tatem,ng)*
*visit(h18PM,h19PM,ng)*

**Ejercicio 6.11** Manolo y Juan tienen que reunirse para escribir un examen. Tal y como muestra la siguiente figura, Manolo está en el primer edificio de la primera planta y Juan en el tercer edificio de la tercera planta.




La facultad tiene dos salas de reuniones, una de ellas en el edificio 1 en la tercera planta y la otra en el edificio 2 en la cuarta planta. Manolo y Juan pueden subir y bajar de plantas en cada edificio por las escaleras y pasar de un edificio a otro por las pasarelas. Los edificios están conectados por los bajos y además tienen una pasarela del edificio 1 al edificio 2 por la segunda planta y del edificio 2 al edificio 3 por la tercera planta. Se pide:

a) Formula el enunciado como un problema de planificación con las siguientes acciones:

*stairUp(P,X,Y,B)*: Persona P sube las escaleras de la planta X a la planta Y en el edificio B.

*stairDown(P,X,Y,B)*: Persona P baja las escaleras de la planta X a la planta Y en el edificio B.

*cross(P,Pz,Bx,By)*: Persona P cruza del piso z del edificio x al edificio y.

*meet(P1,P2,X,B)*: Personas P1 y P2 se reúnen en la sala de reuniones de la planta X del edificio B.

*lift(P,X,Y,B)*: Persona P se monta en el ascensor del piso X al piso Y en el edificio B.

Utilizaremos los siguientes predicados:

*at(P,X,B)*: Persona P está en el piso X en el edificio B.

*building (B)*: P es un edificio.

*lowerLevel(X,Y)*: X es un nivel inferior al nivel Y.

*access(Pz,Bx,By)*: Existe una pasarela del piso pz del edificio Bx al edificio By.

*meetingRoom (X,B)*: Hay una sala de reuniones en el piso X del edificio B.

*meeting(P1,P2)*: La persona P1 y P2 están reunidas.

*hurry(P)*: La persona P tiene prisa.

b) Establecer el estado inicial y objetivo con los predicados mencionados teniendo en cuenta que ni Manolo ni Juan tienen prisa.

c) Determinar la secuencia de acciones como solución al problema planteado en el apartado b.

d) Determine las modificaciones necesarias al estado inicial y establezca la solución al problema teniendo en cuenta que Manolo tiene prisa.

**Solución**

a) operator stairUp(P,X,Y,B)
pre: at(P,X,B), building (B), lowerLevel(X,Y)
post: at(P,X,B), at(P,Y,B)

operator stairDown(P,X,Y,B)
pre: at(P,X,B), building (B), lowerLevel(Y,X)
post: at(P,X,B), at(P,Y,B)

operator cross(P,Pz,Bx,By)
pre: at(P,Pz,Bx), access(Pz,Bx,By)
post: at(P,Pz,Bx), at(P,Pz,By)

operator meet(P1,P2,X,B)
pre: at(P1,X,B), at(P2,X,B), meetingRoom (X,B)
post: meeting(P1,P2)

operator lift(P,X,Y,B)
pre: at(P,X,B), building (B), hurry(P)
post: at(P,X,B), at(P,Y,B)

b) start(

lowerLevel(basement,f1),lowerLevel(f1,f2),lowerLevel(f2,f3),
lowerLevel(f3,f4), at(manolo, f1,b1), at(juan, f3, b3),
building(b1),building(b2),building(b3),
access(f2, b1, b2), access(f2, b2, b1), access(f3, b3, b2), access(f3, b2, b3),
access(basement, b3, b2),access(basement, b2, b3), access(basement, b1, b2),
access(basement, b2, b1), meetingRoom(f3, b1), meetingRoom(f4, b2),
)

goal(
meeting(manolo,juan)
)

c) cross(juan,f3,b3,b2)
stairDown(juan,f3,f2,b2)
cross(juan,f2,b2,b1)
stairUp(manolo,f1,f2,b1)
stairUp(manolo,f2,f3,b1)
stairUp(juan,f2,f3,b1)
meet(manolo,juan,f3,b1)

INFORMACIÓN: Plan length: 7

d) Añadimos el predicado *hurry(manolo)* en el estado final.

Y la solución obtenida sería:

lift(manolo,f1,f2,b1)
cross(manolo,f2,b1,b2)
cross(juan,f3,b3,b2)
stairUp(juan,f3,f4,b2)
lift(manolo,f2,f4,b2)
meet(manolo,juan,f4,b2)

INFORMACIÓN: Plan length: 6

# Tema 7

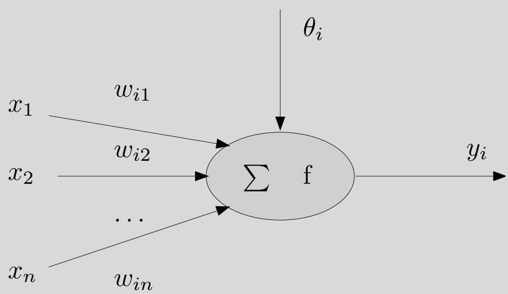
# Redes Neuronales

**Ejercicio 7.1** Diseñar un perceptrón simple para clasificar a los alumnos que pueden acceder a una beca de investigación. Los alumnos aptos de una beca de investigación deberán tener una nota media superior a 1,5 de las 40 asignaturas cursadas por los alumnos. Las calificaciones de las asignaturas se puntúan con 0 (suspenso), 1 (aprobado), 2 (notable), 3 (sobresaliente) y 4 (matrícula de honor). Determinar el número de neuronas, así como los pesos sinápticos y umbrales de la red neuronal propuesta.

**Solución**

Necesitamos un perceptrón simple con 40 entradas (n=40) y una única neurona de salida, donde los pesos sinápticos tendrán un valor de 1/40 y el umbral de 1,5. La función de activación es la función signo, con lo que:

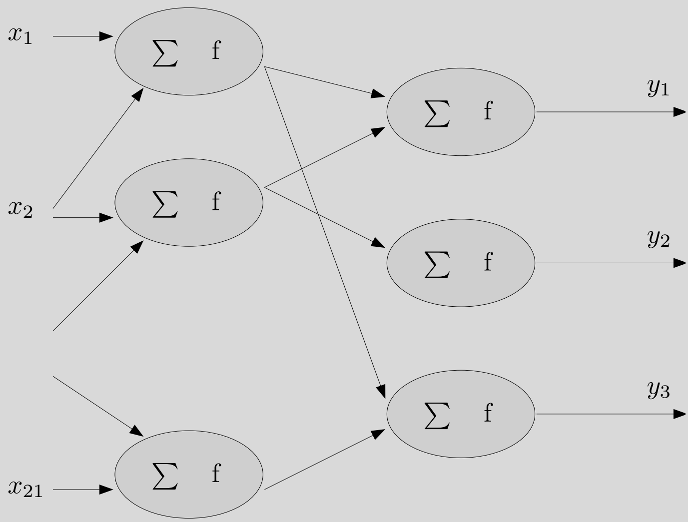
$$y = sgn \sum_{i=1}^{40} 1/40\ x_i\ 1{,}5$$




**Ejercicio 7.2** Se desea predecir los niveles de colesterol de alta densidad (HDL), baja densidad (LDL) y muy baja densidad (VLDL o triglicéridos endógenos), a partir de las medidas de 21 componentes espectrales de 200 personas. Los valores máximos de los tipos de colesterol son los siguientes: 324 mg/dl (HDL), 260 mg/dl (LDL) y 102 mg/dl (VLDL). El valor mínimo para los tres tipos de colesterol es cero. Diseñar la red neuronal propuesta indicando el numero de neuronas (de entrada, ocultas y de salida) y funciones de activación.

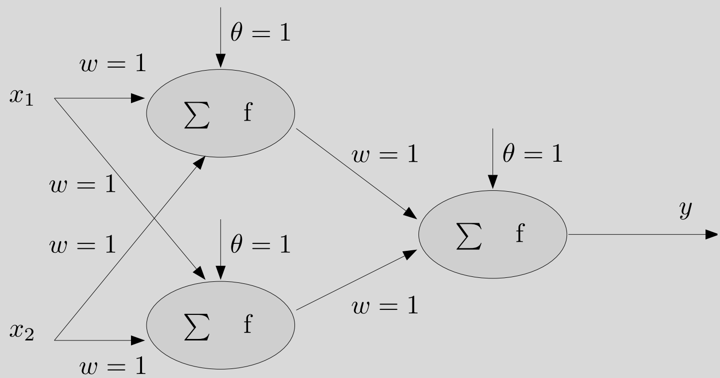
**Solución**

Perceptrón multicapa con 21 entradas, una capa oculta y 3 neuronas de salida (una por cada tipo de colesterol). Las funciones de activación de las neuronas de salida serán de tipo logística y los datos serán normalizados, con lo que los valores de HDL, LDL y VLDL deberán ser divididos por sus respectivos valores máximos, ya que los mínimos son cero.




**Ejercicio 7.3** Implementar la función lógica XOR con un perceptrón de neuronas bipolares, función de activación signo y una capa oculta. Determinar el número de neuronas de cada capa, los pesos sinápticos y umbrales de la red.

**Solución**




**Ejercicio 7.4** Se desea realizar una clasificación de portales web en 8 categorías diferentes. Para ello se han valorado 20 aspectos generales obtenidos de 1000 portales. Diseñar la arquitectura de una red neuronal para realizar dicha clasificación.

**Solución**

Se necesita un perceptrón multicapa con 20 entradas, al menos una capa oculta y 3 neuronas de salida con función de activación paso. Cada una de las 8 categorías será codificada con las 3 salidas binarias. También se podrían utilizar 8 neuronas de salida (una para cada categoría), con función de activación logística.

**Ejercicio 7.5** Diseñar una red neuronal para la conducción automática de un vehículo. Para el entrenamiento se dispone de 5000 fotogramas de 320x240 píxeles, con los correspondientes valores de ángulo de giro del volante y la variación de velocidad ejecutados por un piloto experto. Detallar las características de la red neuronal propuesta.

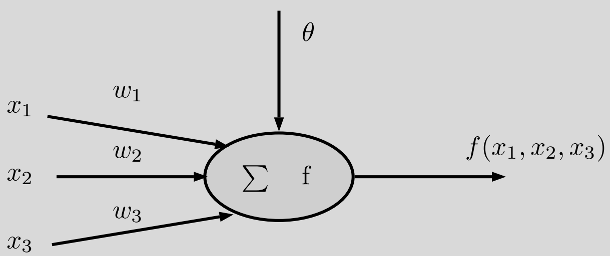
**Solución**

La red tendrá N entradas (una por cada bit), los pesos sinápticos serán unitarios (no hay ponderación de los valores de entrada), el umbral será N/2 y la función de activación del tipo paso, con lo que:

$$y = sgn \sum_{i=1}^{N} x_i \; N/2$$

**Ejercicio 7.6** Determinar los pesos sinápticos, umbral y función de activación de un perceptrón simple que indique si un número binario tiene la mayoría de sus bits activados o no.

**Solución**

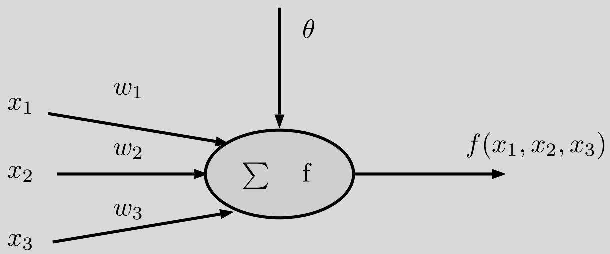



donde la función de activación f es la función escalón y $w_1$, $w_2$, $w_3 = 1$; $\theta = 1{,}5$.

**Ejercicio 7.7** Diseña un perceptrón simple que calcule la siguiente función lógica:

| x1 | x2 | x3 | F(x1,x2,x3) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

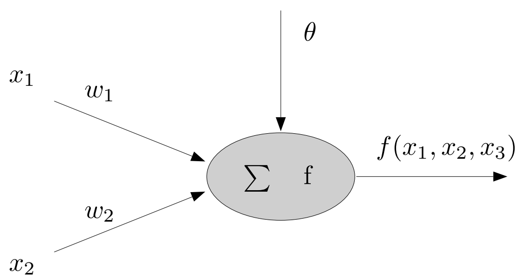
**Solución**




donde la función de activación f es la función escalón y $w_1$, $w_2$, $w_3 = 1$; $\theta$ = 1,5.

**Ejercicio 7.8** Dados los pesos sinápticos y el umbral de un perceptrón simple con la arquitectura que se muestra más abajo, debe determinarse la función booleana que implementa, teniendo en cuenta que la función de activación es la función signo. Los pesos sinápticos y el umbral son:

| | |
|---|---|
| Entrada 1 | 2.58 |
| Entrada 2 | -2.31 |
| Sesgo | 3.18 |



**Solución**

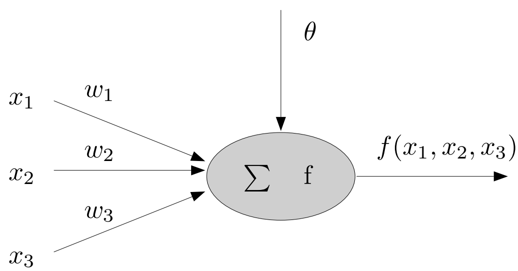
g = potencial sináptico de la neurona de salida

f = resultado de la neurona de salida

| **Entrada 1** | **Entrada 2** | **g** | **f** |
|---|---|---|---|
| -1 | -1 | -3.45 | -1 |
| -1 | 1 | -8.07 | -1 |
| 1 | -1 | 1.71 | 1 |
| 1 | 1 | -2.91 | -1 |

**Ejercicio 7.9** Dados los pesos sinápticos y el umbral de un perceptrón simple con la arquitectura que se muestra más abajo, debe determinarse la función booleana que implementa, teniendo en cuenta que la función de activación es la función escalón. Los pesos sinápticos y el umbral son:

| | |
|---|---|
| Entrada 1 | -2.38 |
| Entrada 2 | -3.15 |
| Entrada 3 | 3.66 |
| Sesgo | -1.84 |



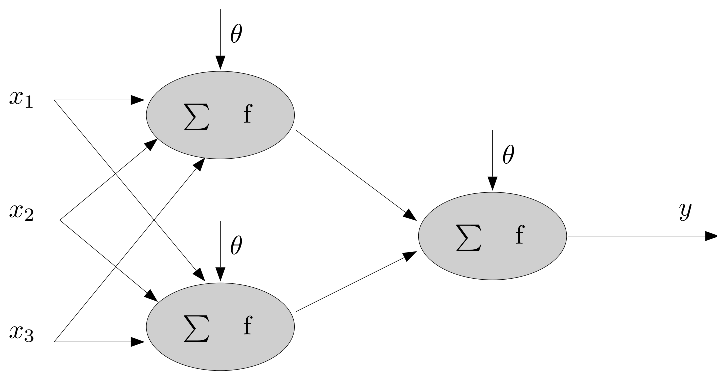
**Solución**

g = potencial sináptico de la neurona de salida

f = resultado de la neurona de salida

| Entrada1 | Entrada2 | Entrada3 | g | f |
|---|---|---|---|---|
| 0 | 0 | 0 | 1.84 | 1 |
| 0 | 0 | 1 | 5.50 | 1 |
| 0 | 1 | 0 | -1.31 | 0 |
| 0 | 1 | 1 | 2.35 | 1 |
| 1 | 0 | 0 | -0.54 | 0 |
| 1 | 0 | 1 | 3.12 | 1 |
| 1 | 1 | 0 | -3.69 | 0 |
| 1 | 1 | 1 | -0.03 | 0 |

**Ejercicio 7.10** Dados los pesos sinápticos y umbrales de un perceptrón multicapa con la arquitectura que se muestra más abajo, debe determinarse cuál es la función booleana que implementa,




a) Teniendo en cuenta que la función de activación es la función escalón. Además, los pesos sinápticos y los umbrales de la capa de entrada a la capa oculta son:

| | **Neurona oculta 1** | **Neurona oculta 2** |
|---|---|---|
| **Entrada 1** | 0.51 | 3.93 |
| **Entrada 2** | 2.08 | 3.96 |
| **Entrada 3** | -2.09 | -3.74 |
| **Sesgo** | 0.11 | -2.93 |

Mientras que los pesos sinápticos y el umbral de la capa oculta a la capa de salida son:

| **Neurona oculta 1** | **Neurona oculta 2** | **Sesgo** |
|---|---|---|
| -4.49 | -0.59 | -4.70 |

b) Teniendo en cuenta que la función de activación es la función signo. Además, los pesos sinápticos y los umbrales de la capa de entrada a la capa oculta son:

| | **Neurona oculta 1** | **Neurona oculta 2** |
|---|---|---|
| **Entrada 1** | -0.83 | -3.53 |
| **Entrada 2** | 2.20 | -4.08 |
| **Entrada 3** | -5.00 | -3.14 |
| **Sesgo** | -1.98 | -1.54 |

Mientras que los pesos sinápticos y el umbral de la capa oculta a la capa de salida son:

| **Neurona oculta 1** | **Neurona oculta 2** | **Sesgo** |
|---|---|---|
| -1.03 | 0.39 | -0.81 |

**Solución**

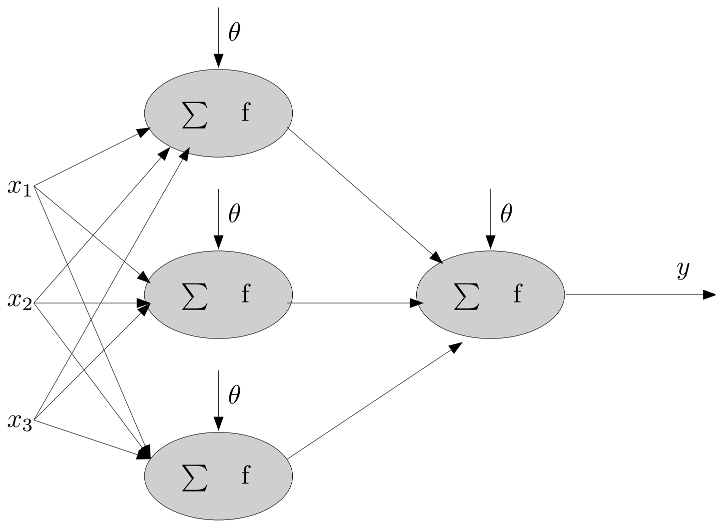
a)

| Entrada | h1 | h2 | z1 | z2 | g | f |
|---|---|---|---|---|---|---|
| 0 0 0 | -0.11 | 2.93 | 0 | 1 | 4.11 | 1 |
| 0 0 1 | -2.20 | -0.81 | 0 | 0 | 4.70 | 1 |
| 0 1 0 | 1.97 | 6.89 | 1 | 1 | -0.38 | 0 |
| 0 1 1 | -0.12 | 3.15 | 0 | 1 | 4.11 | 1 |
| 1 0 0 | 0.40 | 6.86 | 1 | 1 | -0.38 | 0 |
| 1 0 1 | -1.69 | 3.12 | 0 | 1 | 4.11 | 1 |
| 1 1 0 | 2.48 | 10.82 | 1 | 1 | -0.38 | 0 |
| 1 1 1 | 0.39 | 7.08 | 1 | 1 | -0.38 | 0 |

b)

| Entrada | h1 | h2 | z1 | z2 | g | f |
|---|---|---|---|---|---|---|
| -1 -1 -1 | 5.61 | 12.29 | 1 | 1 | 0.17 | 1 |
| -1 -1 1 | -4.39 | 6.01 | -1 | 1 | 2.23 | 1 |
| -1 1 -1 | 10.01 | 4.13 | 1 | 1 | 0.17 | 1 |
| -1 1 1 | 0.01 | -2.15 | 1 | -1 | -0.61 | -1 |
| 1 -1 -1 | 3.95 | 5.23 | 1 | 1 | 0.17 | 1 |
| 1 -1 1 | -6.05 | -1.05 | -1 | -1 | 1.45 | 1 |
| 1 1 -1 | 8.35 | -2.93 | 1 | -1 | -0.61 | -1 |
| 1 1 1 | -1.65 | -9.21 | -1 | -1 | 1.45 | 1 |

**Ejercicio 7.11** Dados los pesos sinápticos y umbrales de un perceptrón multicapa con la arquitectura que se muestra más abajo, debe determinarse cuál es la función booleana que implementa,




a) Teniendo en cuenta que la función de activación es la función signo. Además, los pesos sinápticos y los umbrales de la capa de entrada a la capa oculta son:

| | **Neurona oculta 1** | **Neurona oculta 2** | **Neurona oculta 3** |
|---|---|---|---|
| **Entrada 1** | 6.93 | -0.92 | 1.35 |
| **Entrada 2** | 1.32 | 3.95 | 4.23 |
| **Entrada 3** | 6.21 | 3.30 | 2.38 |
| **Sesgo** | -1.58 | 2.19 | 5.36 |

Mientras que los pesos sinápticos y el umbral de la capa oculta a la capa de salida son:

| **Neurona oculta 1** | **Neurona oculta 2** | **Neurona oculta 2** | **Sesgo** |
|---|---|---|---|
| 3.18 | 3.79 | 4.45 | 7.90 |

b) Teniendo en cuenta que la función de activación es la función signo. Además, los pesos sinápticos y los umbrales de la capa de entrada a la capa oculta son:

| | **Neurona oculta 1** | **Neurona oculta 2** | **Neurona oculta 3** |
|---|---|---|---|
| **Entrada 1** | 3.73 | -2.67 | 0.23 |
| **Entrada 2** | 4.69 | -4.89 | -0.22 |
| **Entrada 3** | 3.69 | -0.70 | 0.55 |
| **Sesgo** | 0.31 | -0.98 | 0.43 |

Mientras que los pesos sinápticos y el umbral de la capa oculta a la capa de salida son:

| **Neurona oculta 1** | **Neurona oculta 2** | **Neurona oculta 2** | **Sesgo** |
|---|---|---|---|
| 2.61 | 2.12 | 1.20 | -0.74 |

**Solución**

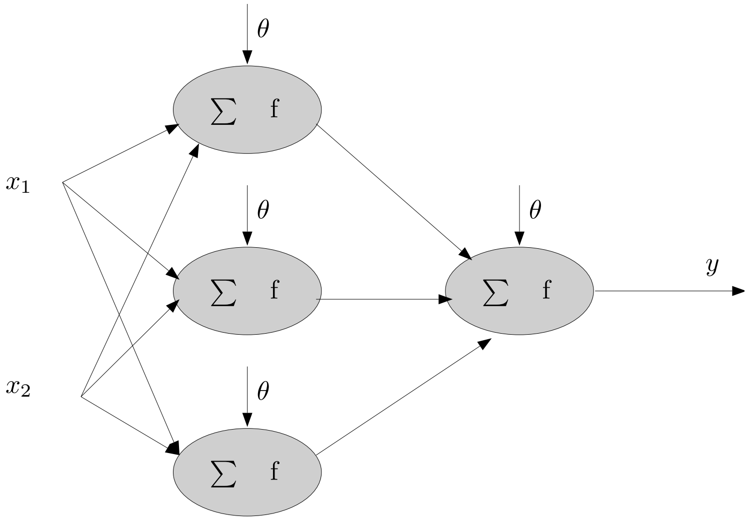
a)

| **Entrada** | **h1** | **h2** | **h3** | **z1** | **z2** | **z3** | **g** | **f** |
|---|---|---|---|---|---|---|---|---|
| 0 0 0 | 1.58 | -2.19 | -5.36 | 1 | 0 | 0 | -4.72 | 0 |
| 0 0 1 | 7.79 | 1.11 | -2.98 | 1 | 1 | 0 | -0.93 | 0 |
| 0 1 0 | 2.90 | 1.76 | -1.13 | 1 | 1 | 0 | -0.93 | 0 |
| 0 1 1 | 9.11 | 5.06 | 1.25 | 1 | 1 | 1 | 3.52 | 1 |
| 1 0 0 | 8.51 | -3.11 | -4.01 | 1 | 0 | 0 | -4.72 | 0 |
| 1 0 1 | 14.72 | 0.19 | -1.63 | 1 | 1 | 0 | -0.93 | 0 |
| 1 1 0 | 9.83 | 0.84 | 0.22 | 1 | 1 | 1 | 3.52 | 1 |
| 1 1 1 | 16.04 | 4.14 | 2.60 | 1 | 1 | 1 | 3.52 | 1 |

b)

| **Entrada** | **h1** | **h2** | **h3** | **z1** | **z2** | **z3** | **g** | **f** |
|---|---|---|---|---|---|---|---|---|
| -1 -1 -1 | -12.42 | 9.24 | -0.99 | -1 | 1 | -1 | -0.95 | -1 |
| -1 -1 1 | -5.04 | 7.84 | 0.11 | -1 | 1 | 1 | 1.45 | 1 |
| -1 1 -1 | -3.04 | -0.54 | -1.43 | -1 | -1 | -1 | -5.19 | -1 |
| -1 1 1 | 4.34 | -1.94 | -0.33 | 1 | -1 | -1 | 0.03 | 1 |
| 1 -1 -1 | -4.96 | 3.90 | -0.53 | -1 | 1 | -1 | -0.95 | -1 |
| 1 -1 1 | 2.42 | 2.50 | 0.57 | 1 | 1 | 1 | 6.67 | 1 |
| 1 1 -1 | 4.42 | -5.88 | -0.97 | 1 | -1 | -1 | 0.03 | 1 |
| 1 1 1 | 11.80 | -7.28 | 0.13 | 1 | -1 | 1 | 2.43 | 1 |

**Ejercicio 7.12** Dados los pesos sinápticos y umbrales de un perceptrón multicapa con la arquitectura que se muestra más abajo, debe determinarse cuál es la función booleana que implementa,




Teniendo en cuenta que la función de activación es la función signo. Además, los pesos sinápticos y los umbrales de la capa de entrada a la capa oculta son:

| | **Neurona oculta 1** | **Neurona oculta 2** | **Neurona oculta 3** |
|---|---|---|---|
| **Entrada 1** | -1.48 | -2.67 | 2.77 |
| **Entrada 2** | 1.40 | -1.55 | -1.54 |
| **Sesgo** | 3.88 | 0.70 | 3.27 |

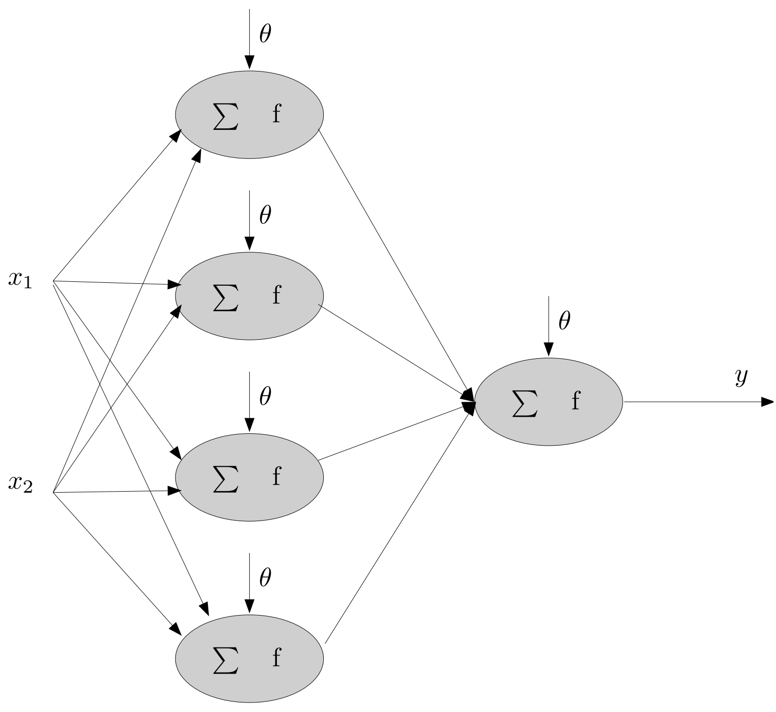
Mientras que los pesos sinápticos y el umbral de la capa oculta a la capa de salida son:

| **Neurona oculta 1** | **Neurona oculta 2** | **Neurona oculta 2** | **Sesgo** |
|---|---|---|---|
| 0.57 | -4.83 | 0.32 | 3.55 |

**Solución**

| **Entrada** | **h1** | **h2** | **h3** | **z1** | **z2** | **z3** | **g** | **f** |
|---|---|---|---|---|---|---|---|---|
| -1 -1 | -3.80 | 3.52 | -4.50 | -1 | 1 | -1 | -9.27 | -1 |
| -1 1 | -1.00 | 0.42 | -7.58 | -1 | 1 | -1 | -9.27 | -1 |
| 1 -1 | -6.76 | -1.82 | 1.04 | -1 | -1 | 1 | 1.03 | 1 |
| 1 1 | -3.96 | -4.92 | -2.04 | -1 | -1 | -1 | 0.39 | 1 |

**Ejercicio 7.13** Dados los pesos sinápticos y umbrales de un perceptrón multicapa con la arquitectura que se muestra más abajo, debe determinarse cuál es la función booleana que implementa,




Teniendo en cuenta que la función de activación es la función signo. Además, los pesos sinápticos y los umbrales de la capa de entrada a la capa oculta son:

| | **Neurona oculta 1** | **Neurona oculta 2** | **Neurona oculta 3** | **Neurona oculta 4** |
|---|---|---|---|---|
| **Entrada 1** | 6.35 | 1.61 | 1.94 | 5.10 |
| **Entrada 2** | -0.95 | 1.59 | 2.09 | 7.61 |
| **Sesgo** | 5.45 | 4.09 | 3.10 | 2.57 |

Mientras que los pesos sinápticos y el umbral de la capa oculta a la capa de salida son:

| **Neurona oculta 1** | **Neurona oculta 2** | **Neurona oculta 3** | **Neurona oculta 4** | **Sesgo** |
|---|---|---|---|---|
| 2.28 | -0.87 | 0.18 | 7.57 | 7.43 |

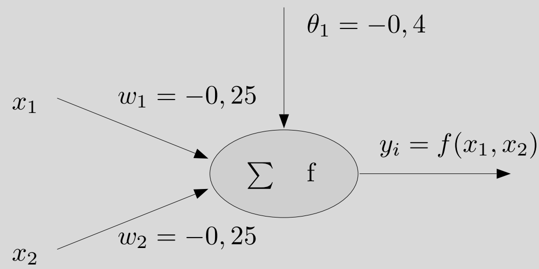
**Solución**

| **Entrada** | **h1** | **h2** | **h3** | **h4** | **z1** | **z2** | **z3** | **z4** | **g** | **f** |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 0 | -5.45 | -4.09 | -3.10 | -2.57 | 0 | 0 | 0 | 0 | -7.43 | 0 |
| 0 1 | -6.40 | -2.50 | -1.01 | 5.04 | 0 | 0 | 0 | 1 | 0.14 | 1 |
| 1 0 | 0.90 | -2.48 | -1.16 | 2.53 | 1 | 0 | 0 | 1 | 2.42 | 1 |
| 1 1 | -0.05 | -0.89 | 0.93 | 10.14 | 0 | 0 | 1 | 1 | 0.32 | 1 |

**Ejercicio 7.14** Diseña un perceptrón simple que calcule la función lógica NAND:

| $x_1$ | $x_2$ | $F(x_1, x_2)$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**Solución**

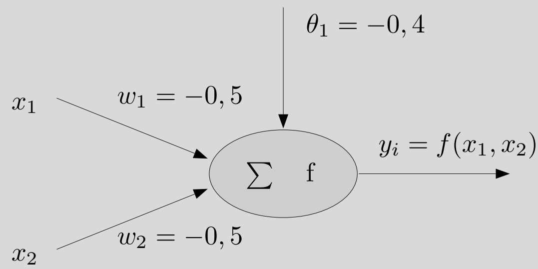



donde la función de activación f es la función signo.

**Ejercicio 7.15** Diseña un perceptrón simple que calcule la siguiente función lógica:

| $x_1$ | $x_2$ | $x_3$ | $F(x_1, x_2, x_3)$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

**Solución**

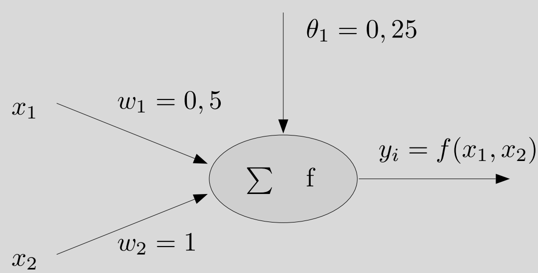



donde la función de activación f es la función signo.

**Ejercicio 7.16** Diseña un perceptrón simple que calcule la siguiente función lógica:

| $x_1$ | $x_2$ | $F(x_1, x_2)$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**Solución**

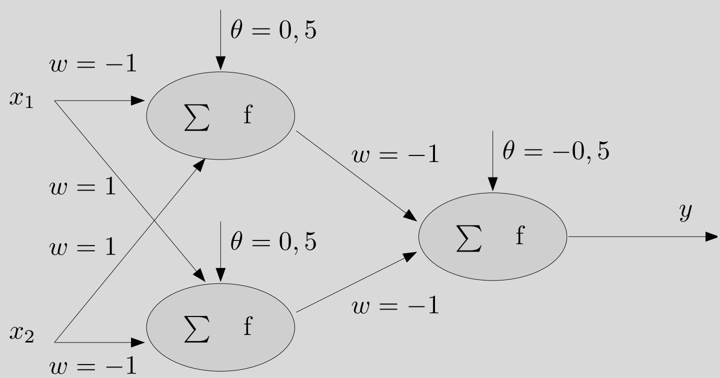



donde la función de activación f es la función signo.

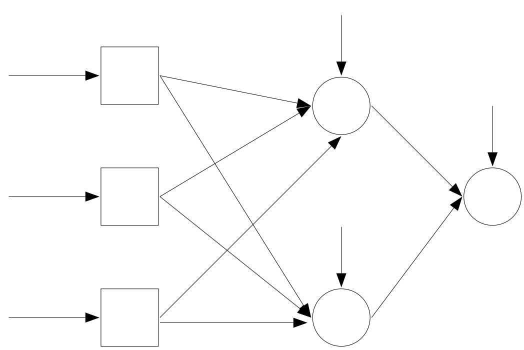
**Ejercicio 7.17** Diseña un red neuronal que calcule la siguiente función lógica:

| $x_1$ | $x_2$ | $F(x_1, x_2)$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**Solución**




**Ejercicio 7.18** Dados los pesos sinápticos y los sesgos de un perceptrón multicapa con la arquitectura dada en la figura y la función de activación signo, halla la función booleana que implementa.



Pesos sinápticos y sesgos de la capa de entrada a la capa oculta:

| | Neurona oculta 1 | Neurona oculta 2 |
|---|---|---|
| Entrada 1 | 4,60 | -1,39 |
| Entrada 2 | 2,00 | 2,40 |
| Entrada 3 | 5,00 | 4,96 |
| Sesgo | -2,80 | -1,84 |

Pesos sinápticos y sesgos de la capa oculta a la capa de salida:

| Neurona oculta 1 | Neurona oculta 2 | Sesgo |
|---|---|---|
| -3.63 | -1.16 | -1.79 |

**Solución**

$h_i$ = synaptic potential of the i-th hidden neuron

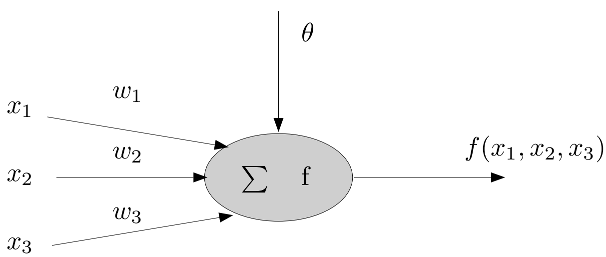
$z_i$ = result of the i-th hidden neuron

g = synaptic potential of the output neuron

f = result of the output neuron

| **Entrada 1** | **Entrada 2** | **Entrada 3** | **h1** | **h2** | **z1** | **z2** | **g** | **f** |
|---|---|---|---|---|---|---|---|---|
| -1 | -1 | -1 | -8.80 | -4.13 | -1 | -1 | 6.58 | 1 |
| -1 | -1 | 1 | 1.20 | 5.79 | 1 | 1 | -3.00 | -1 |
| -1 | 1 | -1 | -4.80 | 0.67 | -1 | 1 | 4.26 | 1 |
| -1 | 1 | 1 | 5.20 | 10.59 | 1 | 1 | -3.00 | -1 |
| 1 | -1 | -1 | 0.40 | -6.91 | 1 | -1 | -0.68 | -1 |
| 1 | -1 | 1 | 10.40 | 3.01 | 1 | 1 | -3.00 | -1 |
| 1 | 1 | -1 | 4.40 | -2.11 | 1 | -1 | -0.68 | -1 |
| 1 | 1 | 1 | 14.40 | 7.81 | 1 | 1 | -3.00 | -1 |

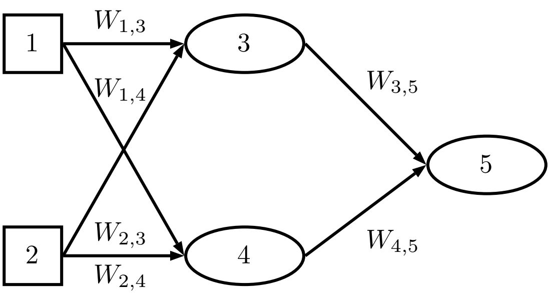
**Ejercicio 7.19** Dado el siguiente perceptrón simple, halla la función booleana que implementa, donde f es la función paso, $w_1$, $w_2$, $w_3 = -1$; $\theta = -1,5$.



**Solución**

| **x1** | **x2** | **x3** | **f(x1,x2,x3)** |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

**Ejercicio 7.20** Sea la red neuronal de la figura:




La función de activación de todas las unidades es la función escalón, a(v) = 0 si $v < 0$, a(v)=1 en otro caso.

Los pesos son: $w_{13} = 1, w_{23} = 1, w_{14} = -1, w_{24} = -1, w_{35} = 1, w_{45} = 1$

Los sesgos son: $b_3 = 1,5, b_4 = -0,5, b_5 = 0,5$

a) Determinar qué función booleana calcula la red.

b) Supongamos una red con los mismos parámetros que la anterior, salvo b5 (que puede tomar cualquier valor). Determinar el intervalo de valores de b5 para los cuales se calcula la misma función booleana determinada en a)

**Solución**

a) Determinar qué función booleana calcula la red.

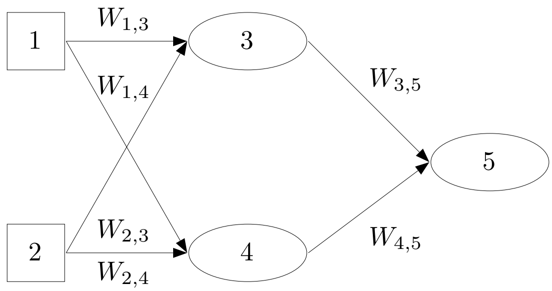
| **x1** | **x2** | **v3** | **a3** | **v4** | **a4** | **v5** | **a5** |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0,5 | 1 | -1,5 | 0 | 0,5 | 1 |
| 1 | 0 | -0,5 | 0 | -0,5 | 0 | -0,5 | 0 |
| 0 | 1 | -0,5 | 0 | -0,5 | 0 | -0,5 | 0 |
| 0 | 0 | -1,5 | 0 | 0,5 | 1 | 0,5 | 1 |

b) Supongamos una red con los mismos parámetros que la anterior, salvo b5 (que puede tomar cualquier valor). Determinar el intervalo de valores de b5 para los cuales se calcula la misma función booleana determinada en a)

| **x1** | **x2** | **v5** | **a3** | **v4** | **a4** | **v5** |
|---|---|---|---|---|---|---|
| 1 | 1 | 0,5 | 1 | -1,5 | 0 | -b5 + 1 |
| 1 | 0 | -0,5 | 0 | -0,5 | 0 | -b5 + 0 |
| 0 | 1 | -0,5 | 0 | -0,5 | 0 | -b5 + 0 |
| 0 | 0 | -1,5 | 0 | 0,5 | 1 | -b5 + 1 |

Para que la activación a5 sea la misma que antes debe cumplirse -b5 + 1 $>$ 0 y b5 + 0 $<$ 0 sea 0 $<$ b5 $<$ 1

**Ejercicio 7.21** Sea la red neuronal de la figura.




La función de activación de todas las unidades es la función escalón, a(v) = 0 si $v < 0$, a(v)=1 en otro caso.

Los pesos son todos iguales a 2. Los sesgos son: $b_3 = 0,5 b_4 = 1, b_5 = 3,5$

a) Determinar qué función booleana calcula la red.

b) Diseñar una red sin capa oculta que calcule la misma función booleana.

**Solución**

a) Determinar qué función booleana calcula la red.

| **x1** | **x2** | **v3** | **a3** | **v4** | **a4** | **v5** | **a5** |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 3,5 | 1 | 3 | 1 | 0,5 | 1 |
| 1 | 0 | 1,5 | 1 | 1 | 1 | 0,5 | 1 |
| 0 | 1 | 1,5 | 1 | 1 | 1 | 0,5 | 1 |
| 0 | 0 | -0,5 | 0 | -1 | 0 | -3,5 | 0 |

b) Diseñar una red sin capa oculta que calcule la misma función booleana.

Basta observar por ejemplo que la unidad 3 tiene la misma salida que la a5.

**Ejercicio 7.22** Dado los siguientes pesos sinápticos y sesgo de un perceptron multicapa con la siguiente estructura, debes encontrar la función booleana que implementa. Función Activación: Paso

Pesos sinápticos y sesgo de la capa entrada a la capa oculta:

| | Neurona Oculta 1 | Neurona Oculta 2 | Neurona Oculta 3 |
|---|---|---|---|
| Entrada 1 | 3.39 | -4.61 | -3.06 |
| Entrada 2 | 3.21 | -4.34 | -3.03 |
| Entrada 3 | 1.42 | -2.23 | 3.49 |
| Sesgo | 1.67 | -1.84 | 3.42 |

Pesos sinápticos y sesgo de la capa oculta a la capa salida:

| Neurona Oculta 1 | Neurona Oculta 2 | Neurona Oculta 3 | Sesgo |
|---|---|---|---|
| -1.53 | 2.40 | -0.07 | -0.27 |

**Solución**

$h_i$ = potencial sináptico de la i-ésima neurona oculta
$z_i$ = resultado de la i-ésima neurona oculta
g = potencial sináptico de la neurona de salida
f = resultado de la neurona de salida

| **Ent1** | **Ent2** | **Ent3** | **h1** | **h2** | **h3** | **z1** | **z2** | **z3** | **g** | **f** |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | -1.67 | 1.84 | -3.42 | 0 | 1 | 0 | 2.67 | 1 |
| 0 | 0 | 1 | -0.25 | -0.39 | 0.07 | 0 | 0 | 1 | 0.20 | 1 |
| 0 | 1 | 0 | 1.54 | -2.50 | -6.45 | 1 | 0 | 0 | -1.26 | 0 |
| 0 | 1 | 1 | 2.96 | -4.73 | -2.96 | 1 | 0 | 0 | -1.26 | 0 |
| 1 | 0 | 0 | 1.72 | -2.77 | -6.48 | 1 | 0 | 0 | -1.26 | 0 |
| 1 | 0 | 1 | 3.14 | -5.00 | -2.99 | 1 | 0 | 0 | -1.26 | 0 |
| 1 | 1 | 0 | 4.93 | -7.11 | -9.51 | 1 | 0 | 0 | -1.26 | 0 |
| 1 | 1 | 1 | 6.35 | -9.34 | -6.02 | 1 | 0 | 0 | -1.26 | 0 |

**Ejercicio 7.23** Dado los siguientes pesos sinápticos y sesgo de un perceptron multicapa con la siguiente estructura, debes encontrar la función booleana que implementa. Función Activación: Paso

Pesos sinápticos y sesgo de la capa entrada a la capa oculta:

| | Neurona Oculta 1 | Neurona Oculta 2 | Neurona Oculta 3 |
|---|---|---|---|
| Entrada 1 | -0.45 | -3.10 | 3.15 |
| Entrada 2 | -0.60 | 1.15 | -2.99 |
| Entrada 3 | 2.35 | -2.06 | 0.43 |
| Sesgo | -0.35 | -0.72 | 1.44 |

Pesos sinápticos y sesgo de la capa oculta a la capa salida:

| Neurona Oculta 1 | Neurona Oculta 2 | Neurona Oculta 3 | Sesgo |
|---|---|---|---|
| -3.46 | -3.29 | 3.33 | -3.76 |

**Solución**

$h_i$ = potencial sináptico de la i-ésima neurona oculta

$z_i$ = resultado de la i-ésima neurona oculta

g = potencial sináptico de la neurona de salida

f = resultado de la neurona de salida

| Ent1 | Ent2 | Ent3 | h1 | h2 | h3 | z1 | z2 | z3 | g | f |
|---|---|---|---|---|---|---|---|---|---|---|
| -1 | -1 | -1 | -1,85 | 4.73 | -2.03 | -1 | 1 | -1 | 0.60 | 1 |
| -1 | -1 | 1 | 2,85 | 0.61 | -1.17 | 1 | 1 | -1 | -6.32 | -1 |
| -1 | 1 | -1 | -3,05 | 7.03 | -8.01 | -1 | 1 | -1 | 0.60 | 1 |
| -1 | 1 | 1 | 1,65 | 2.91 | -7.15 | 1 | 1 | -1 | -6.32 | -1 |
| 1 | -1 | -1 | -0,95 | -1.47 | 4.27 | -1 | -1 | 1 | 13.84 | 1 |
| 1 | -1 | 1 | 3,75 | -5.59 | 5.13 | 1 | -1 | 1 | 6.92 | 1 |
| 1 | 1 | -1 | -2,15 | 0.83 | -1.71 | -1 | 1 | -1 | 0.60 | 1 |
| 1 | 1 | 1 | 2,55 | -3.29 | -0.85 | 1 | -1 | -1 | 0.26 | 1 |

# Tema 8

# Clasificación

**Ejercicio 8.1** Dado el siguiente conjunto de datos de entrenamiento, se pide usar el clasificador redes bayesianas con $m$ constante de estimación $m = 2$ para el test de los patrones correspondientes a las siguientes personas: una persona que tiene tos y fiebre, una persona que tiene la nariz congestionada y fiebre, y una persona que tiene la nariz congestionada y piel enrojecida.

| Muestra | $x_1$ (nariz congestionada) | $x_2$ (tos) | $x_3$ (piel enrojecida) | $x_4$ (fiebre) | Clase |
|---|---|---|---|---|---|
| 1 | Sí | Sí | Sí | No | + (enfermo) |
| 2 | Sí | Sí | No | No | + (enfermo) |
| 3 | No | No | Sí | Sí | + (enfermo) |
| 4 | Sí | No | No | No | - (sano) |
| 5 | No | No | No | No | - (sano) |
| 6 | No | Sí | Sí | No | - (sano) |

**Solución**

En este ejercicio $C = 2$, $V = \{Enfermo, Sano\}$, y $D = 4$. Por tanto, para cada $y \in V$ tenemos:

$$P(y|\mathbf{x}) = \frac{P(y)P(\mathbf{x}|y)}{P(Enfermo)P(\mathbf{x}|Enfermo) + P(Sano)P(\mathbf{x}|Sano)}$$

$$P(\mathbf{x}|y) = \prod_{d=1}^{4} P(x_d|y)$$

De las frecuencias de clase en el conjunto de entrenamiento se obtienen las probabilidades de clase a priori:

$$P(Enfermo) = \frac{3}{6} = \frac{1}{2}$$

$$P(Sano) = \frac{3}{6} = \frac{1}{2}$$

Por otro lado, ya que todas las componentes de entrada tienen solo dos posibles valores, podemos fijar unas probabilidades a priori para la $m$-estimación a $p = 0{,}5$, que quiere decir que:

$$P(x_d|y) = \frac{n' + 1}{n + 2}$$

Ahora calculamos las probabilidades a posteriori:

$P(x_1 = Sí|Enfermo) = \frac{2+1}{3+2} = \frac{3}{5}$ (consideramos las muestras 1, 2 y 3)

$P(x_1 = Sí|Sano) = \frac{1+1}{3+2} = \frac{2}{5}$ (consideramos las muestras 4, 5 y 6)

$P(x_2 = Sí|Enfermo) = \frac{1+1}{3+2} = \frac{2}{5}$ (consideramos las muestras 1, 2 y 3)

$P(x_2 = Sí|Sano) = \frac{1+1}{3+2} = \frac{2}{5}$ (consideramos las muestras 4, 5 y 6)

$P(x_3 = Sí|Enfermo) = \frac{2+1}{3+2} = \frac{3}{5}$ (consideramos las muestras 1, 2 y 3)

$P(x_3 = Sí|Sano) = \frac{1+1}{3+2} = \frac{2}{5}$ (consideramos las muestras 4, 5 y 6)

$P(x_4 = Sí|Enfermo) = \frac{1+1}{3+2} = \frac{2}{5}$ (consideramos las muestras 1, 2 y 3)

$P(x_4 = Sí|Sano) = \frac{0+1}{3+2} = \frac{1}{5}$ (consideramos las muestras 4, 5 y 6)

Por último, calculamos las probabilidades para cada persona que está enferma:

a) Una persona que tiene tos y fiebre: $\mathbf{x} = (No, Sí, No, Sí)$

$P(\mathbf{x}|Enfermo) = P(x_1 = No|Enfermo)P(x_2 = Sí|Enfermo)P(x_3 = No|Enfermo)P(x_4 = Sí|Enfermo) =$
$\frac{2}{5} \cdot \frac{2}{5} \cdot \frac{2}{5} \cdot \frac{2}{5} = \frac{16}{625}$
$P(\mathbf{x}|Sano) = P(x_1 = No|Sano)P(x_2 = Sí|Sano)P(x_3 = No|Sano)P(x_4 = Sí|Sano) =$
$\frac{3}{5} \cdot \frac{2}{5} \cdot \frac{3}{5} \cdot \frac{1}{5} = \frac{18}{625}$
$P(Enfermo|\mathbf{x}) = \frac{P(Enfermo)P(\mathbf{x}|Enfermo)}{P(Enfermo)P(\mathbf{x}|Enfermo) + P(Sano)P(\mathbf{x}|Sano)} =$
$\frac{\frac{1}{2} \cdot \frac{16}{625}}{\frac{1}{2} \cdot \frac{16}{625} + \frac{1}{2} \cdot \frac{18}{625}} = \frac{16}{16+18} = \frac{8}{17} \approx 0{,}4706$
$P(Sano|\mathbf{x}) = 1 - P(Enfermo|\mathbf{x}) = \frac{9}{17} \approx 0{,}5294$
Por tanto, se predice que esta persona está sana.

b) Una persona que tiene la nariz congestionada y fiebre: $\mathbf{x} = (Sí, No, No, Sí)$
$P(\mathbf{x}|Enfermo) = P(x_1 = Sí|Enfermo)P(x_2 = No|Enfermo)P(x_3 = No|Enfermo)P(x_4 = Sí|Enfermo) =$
$\frac{3}{5} \cdot \frac{3}{5} \cdot \frac{2}{5} \cdot \frac{2}{5} = \frac{36}{625}$
$P(\mathbf{x}|Sano) = P(x_1 = Sí|Sano)P(x_2 = No|Sano)P(x_3 = No|Sano)P(x_4 = Sí|Sano) =$
$\frac{2}{5} \cdot \frac{3}{5} \cdot \frac{3}{5} \cdot \frac{1}{5} = \frac{18}{625}$
$P(Enfermo|\mathbf{x}) = \frac{P(Enfermo)P(\mathbf{x}|Enfermo)}{P(Enfermo)P(\mathbf{x}|Enfermo) + P(Sano)P(\mathbf{x}|Sano)} =$
$\frac{\frac{1}{2} \cdot \frac{36}{625}}{\frac{1}{2} \cdot \frac{36}{625} + \frac{1}{2} \cdot \frac{18}{625}} = \frac{36}{36+18} = \frac{2}{3} \approx 0{,}6667$
$P(Sano|\mathbf{x}) = 1 - P(Enfermo|\mathbf{x}) = \frac{1}{3} \approx 0{,}3333$
Por tanto, se predice que esta persona está enferma.

c) Una persona que tiene la nariz congestionada y piel enrojecida:
$\mathbf{x} = (Sí, No, Sí, No)$
$P(\mathbf{x}|Enfermo) = P(x_1 = Sí|Enfermo)P(x_2 = No|Enfermo)P(x_3 = Sí|Enfermo)P(x_4 = No|Enfermo) =$

$\frac{3}{5} \cdot \frac{3}{5} \cdot \frac{3}{5} \cdot \frac{3}{5} = \frac{81}{625}$

$P(\mathbf{x}|Sano) = P(x_1 = Sí|Sano)P(x_2 = No|Sano)P(x_3 = Sí|Sano)P(x_4 = No|Sano) =$

$\frac{2}{5} \cdot \frac{3}{5} \cdot \frac{2}{5} \cdot \frac{4}{5} = \frac{48}{625}$

$$P(Enfermo|\mathbf{x}) = \frac{P(Enfermo)P(\mathbf{x}|Enfermo)}{P(Enfermo)P(\mathbf{x}|Enfermo) + P(Sano)P(\mathbf{x}|Sano)} =$$

$$\frac{\frac{1}{2} \cdot \frac{81}{625}}{\frac{1}{2} \cdot \frac{81}{625} + \frac{1}{2} \cdot \frac{48}{625}} = \frac{81}{81+48} = \frac{27}{43} \approx 0{,}6279$$

$P(Sano|\mathbf{x}) = 1 - P(Enfermo|\mathbf{x}) = \frac{16}{43} \approx 0{,}3721$

Por tanto, se predice que esta persona está enferma.

**Ejercicio 8.2** Dado el siguiente conjunto de datos de entrenamiento, se pide usar el clasificador de redes bayesianas con $m$ constante de estimación $m = 4$ para el test de los patrones correspondientes a las siguientes personas: una persona que tiene fiebre alta, y una persona que sufre vómitos y temblores.

| Muestra | $x_1$ (fiebre) | $x_2$ (vómitos) | $x_3$ (diarrea) | $x_4$ (temblores) | Clase |
|---|---|---|---|---|---|
| 1 | no | no | no | no | sano |
| 2 | media | no | no | no | gripe |
| 3 | alta | no | no | sí | gripe |
| 4 | alta | sí | sí | no | salmonela |
| 5 | media | no | sí | no | salmonela |
| 6 | no | sí | sí | no | inflamación intestinal |
| 7 | media | sí | sí | no | inflamación intestinal |

**Solución**

En este ejercicio $C = 4$, $V = \{Sano, Gripe, Salm, Inte\}$, y $D = 4$. Por tanto, para cada $y \in V$ tenemos:

$$P(y|\mathbf{x}) = \frac{P(y)P(\mathbf{x}|y)}{\sum\limits_{v \in V} P(v)P(\mathbf{x}|v)}$$

$$P(\mathbf{x}|y) = \prod_{d=1}^{4} P(x_d|y)$$

De las frecuencias de clase en el conjunto de entrenamiento se obtienen las probabilidades de clase a priori:

$$P(Sano) = \frac{1}{7}$$

$$P(Gripe) = \frac{2}{7} = P(Salm) = P(Inte)$$

Por otro lado:

$$P(x_1|y) = \frac{n' + 4 \cdot \frac{1}{3}}{n+4} = \frac{n' + \frac{4}{3}}{n+4}$$

$$P(x_2|y) = \frac{n' + 4 \cdot \frac{1}{2}}{n+4} = \frac{n'+2}{n+4}$$

$$P(x_3|y) = \frac{n' + 4 \cdot \frac{1}{2}}{n+4} = \frac{n'+2}{n+4}$$

$$P(x_4|y) = \frac{n' + 4 \cdot \frac{1}{2}}{n+4} = \frac{n'+2}{n+4}$$

Ahora calculamos las probabilidades de clase para cada dato de test:

a) Una persona que tiene fiebre alta: $\mathbf{x} = (Alta, No, No, No)$

$P(\mathbf{x}|Sano) = P(x_1 = Alta|Sano)P(x_2 = No|Sano)P(x_3 = No|Sano)$
$P(x_4 = No|Sano) = \frac{4}{15} \cdot \frac{3}{5} \cdot \frac{3}{5} \cdot \frac{3}{5} = \frac{36}{625}$

$P(\mathbf{x}|Gripe) = P(x_1 = Alta|Gripe)P(x_2 = No|Gripe)P(x_3 = No|Gripe)$
$P(x_4 = No|Gripe) = \frac{7}{18} \cdot \frac{4}{6} \cdot \frac{4}{6} \cdot \frac{3}{6} = \frac{336}{3888} = \frac{7}{81}$

$P(\mathbf{x}|Salm) = P(x_1 = Alta|Salm)P(x_2 = No|Salm)P(x_3 = No|Salm)$
$P(x_4 = No|Salm) = \frac{7}{18} \cdot \frac{3}{6} \cdot \frac{2}{6} \cdot \frac{4}{6} = \frac{168}{3888} = \frac{7}{162}$

$P(\mathbf{x}|Inte) = P(x_1 = Alta|Inte)P(x_2 = No|Inte)P(x_3 = No|Inte)$
$P(x_4 = No|Inte) = \frac{4}{18} \cdot \frac{2}{6} \cdot \frac{2}{6} \cdot \frac{4}{6} = \frac{64}{3888} = \frac{2}{243}$

$$P(Sano|\mathbf{x}) = \frac{P(Sano)P(\mathbf{x}|Sano)}{\sum_{v\in V} P(v)P(\mathbf{x}|v)} =$$

$$\frac{\frac{1}{7} \cdot \frac{36}{625}}{\frac{1}{7} \cdot \frac{36}{625} + \frac{2}{7} \cdot \frac{7}{81} + \frac{2}{7} \cdot \frac{7}{162} + \frac{2}{7} \cdot \frac{2}{243}} = \frac{\frac{36}{4375}}{\frac{1205}{25306}} = \frac{849}{4913} \approx 0{,}1728$$

$$P(Gripe|\mathbf{x}) = \frac{P(Gripe)P(\mathbf{x}|Gripe)}{\sum_{v\in V} P(v)P(\mathbf{x}|v)} =$$

$$\frac{\frac{2}{7} \cdot \frac{7}{81}}{\frac{1}{7} \cdot \frac{36}{625} + \frac{2}{7} \cdot \frac{7}{81} + \frac{2}{7} \cdot \frac{7}{162} + \frac{2}{7} \cdot \frac{2}{243}} = \frac{\frac{2}{81}}{\frac{1205}{25306}} = \frac{937}{1807} \approx 0{,}5185$$

$$P(Salm|\mathbf{x}) = \frac{P(Salm)P(\mathbf{x}|Salm)}{\sum_{v\in V} P(v)P(\mathbf{x}|v)} =$$

$$\frac{\frac{2}{7} \cdot \frac{7}{162}}{\frac{1}{7} \cdot \frac{36}{625} + \frac{2}{7} \cdot \frac{7}{81} + \frac{2}{7} \cdot \frac{7}{162} + \frac{2}{7} \cdot \frac{2}{243}} = \frac{\frac{1}{81}}{\frac{1205}{25306}} = \frac{937}{3614} \approx 0{,}2593$$

$$P(Inte|\mathbf{x}) = \frac{P(Inte)P(\mathbf{x}|Inte)}{\sum_{v\in V} P(v)P(\mathbf{x}|v)} =$$

$$\frac{\frac{2}{7} \cdot \frac{2}{243}}{\frac{1}{7} \cdot \frac{36}{625} + \frac{2}{7} \cdot \frac{7}{81} + \frac{2}{7} \cdot \frac{7}{162} + \frac{2}{7} \cdot \frac{2}{243}} = \frac{\frac{4}{1701}}{\frac{1205}{25306}} = \frac{313}{6338} \approx 0{,}0494$$

Por tanto, se predice que esta persona sufre gripe.

b) Una persona que sufre vómitos y temblores: $\mathbf{x} = (No, Sí, No, Sí)$
$P(\mathbf{x}|Sano) = P(x_1 = No|Sano)P(x_2 = Sí|Sano)P(x_3 = No|Sano)$
$P(x_4 = Sí|Sano) = \frac{7}{15} \cdot \frac{2}{5} \cdot \frac{3}{5} \cdot \frac{2}{5} = \frac{84}{625}$
$P(\mathbf{x}|Gripe) = P(x_1 = No|Gripe)P(x_2 = Sí|Gripe)P(x_3 = No|Gripe)$
$P(x_4 = Sí|Gripe) = \frac{4}{18} \cdot \frac{2}{6} \cdot \frac{4}{6} \cdot \frac{3}{6} = \frac{96}{3888} = \frac{2}{81}$

$P(\mathbf{x}|Salm) = P(x_1 = No|Salm)P(x_2 = Sí|Salm)P(x_3 = No|Salm)$
$P(x_4 = Sí|Salm) = \frac{4}{18} \cdot \frac{3}{6} \cdot \frac{2}{6} \cdot \frac{2}{6} = \frac{48}{3888} = \frac{1}{81}$
$P(\mathbf{x}|Inte) = P(x_1 = No|Inte)P(x_2 = Sí|Inte)P(x_3 = No|Inte)$
$P(x_4 = Sí|Inte) = \frac{4}{18} \cdot \frac{4}{6} \cdot \frac{2}{6} \cdot \frac{2}{6} = \frac{64}{3888} = \frac{2}{243}$

$$P(Sano|\mathbf{x}) = \frac{P(Sano)P(\mathbf{x}|Sano)}{\sum_{v \in V} P(v)P(\mathbf{x}|v)} =$$

$$\frac{\frac{1}{7} \cdot \frac{84}{625}}{\frac{1}{7} \cdot \frac{84}{625} + \frac{2}{7} \cdot \frac{2}{81} + \frac{2}{7} \cdot \frac{1}{81} + \frac{2}{7} \cdot \frac{2}{243}} = \frac{527}{882} \approx 0{,}5975$$

$$P(Gripe|\mathbf{x}) = \frac{P(Gripe)P(\mathbf{x}|Gripe)}{\sum_{v \in V} P(v)P(\mathbf{x}|v)} =$$

$$\frac{\frac{2}{7} \cdot \frac{2}{81}}{\frac{1}{7} \cdot \frac{84}{625} + \frac{2}{7} \cdot \frac{2}{81} + \frac{2}{7} \cdot \frac{1}{81} + \frac{2}{7} \cdot \frac{2}{243}} = \frac{537}{2446} \approx 0{,}2195$$

$$P(Salm|\mathbf{x}) = \frac{P(Salm)P(\mathbf{x}|Salm)}{\sum_{v \in V} P(v)P(\mathbf{x}|v)} =$$

$$\frac{\frac{2}{7} \cdot \frac{1}{81}}{\frac{1}{7} \cdot \frac{84}{625} + \frac{2}{7} \cdot \frac{2}{81} + \frac{2}{7} \cdot \frac{1}{81} + \frac{2}{7} \cdot \frac{2}{243}} = \frac{446}{4063} \approx 0{,}1098$$

$$P(Inte|\mathbf{x}) = \frac{P(Inte)P(\mathbf{x}|Inte)}{\sum_{v \in V} P(v)P(\mathbf{x}|v)} =$$

$$\frac{\frac{2}{7} \cdot \frac{2}{243}}{\frac{1}{7} \cdot \frac{84}{625} + \frac{2}{7} \cdot \frac{2}{81} + \frac{2}{7} \cdot \frac{1}{81} + \frac{2}{7} \cdot \frac{2}{243}} = \frac{179}{2446} \approx 0{,}0732$$

Por tanto, se predice que esta persona está sana.

**Ejercicio 8.3** Dado el siguiente conjunto de datos de entrenamiento, se pide usar el clasificador de redes bayesianas con $m$ constante de estimación $m = 5$ para el test de los patrones correspondientes a un estudiante con ingreso medio y calificación crediticia razonable.

| Muestra | $x_1$ (ingreso) | $x_2$ (estudiante) | $x_3$ (calificación crediticia) | Clase (compra un ordenador) |
|---|---|---|---|---|
| 1 | Alto | No | Razonable | No |
| 2 | Alto | No | Excelente | No |
| 3 | Alto | No | Razonable | Sí |
| 4 | Medio | No | Razonable | Sí |
| 5 | Bajo | Sí | Razonable | Sí |
| 6 | Bajo | Sí | Mala | No |
| 7 | Bajo | Sí | Excelente | Sí |
| 8 | Medio | No | Mala | No |
| 9 | Bajo | Sí | Razonable | Sí |
| 10 | Medio | Sí | Mala | Sí |
| 11 | Medio | Sí | Excelente | Sí |
| 12 | Medio | No | Excelente | Sí |
| 13 | Alto | Sí | Razonable | Sí |
| 14 | Medio | No | Excelente | No |

**Solución**

En este ejercicio $C = 2$, $V = \{Compra, NoCompra\}$, y $D = 3$. Por tanto, para cada $y \in V$ tenemos:

$$P(y|\mathbf{x}) = \frac{P(y)P(\mathbf{x}|y)}{\sum\limits_{v \in V} P(v)P(\mathbf{x}|v)} =$$

$$\frac{P(y)P(\mathbf{x}|y)}{P(Compra)P(\mathbf{x}|Compra) + P(NoCompra)P(\mathbf{x}|NoCompra)}$$

$$P(\mathbf{x}|y) = \prod_{d=1}^{3} P(x_d|y)$$

De las frecuencias de clase en el conjunto de entrenamiento se obtienen las probabilidades de clase a priori:

$$P(Compra) = \frac{9}{14}$$

$$P(NoCompra) = \frac{5}{14}$$

Por otro lado:

$$P(x_1|y) = \frac{n' + \frac{5}{3}}{n + 5}$$

$$P(x_2|y) = \frac{n' + \frac{5}{2}}{n + 5}$$

$$P(x_3|y) = \frac{n' + \frac{5}{3}}{n + 5}$$

Ahora calculamos las probabilidades de clase para el dato de test:
$\mathbf{x} = (Medio, Sí, Razonable)$

$$P(\mathbf{x}|Compra) = P(x_1 = Medio|Compra)P(x_2 = Sí|Compra)$$

$$P(x_3 = Razonable|Compra) = \frac{17}{42} \cdot \frac{17}{28} \cdot \frac{20}{42} = \frac{728}{6221}$$

$$P(\mathbf{x}|NoCompra) = P(x_1 = Medio|NoCompra)P(x_2 = Sí|NoCompra)$$

$$P(x_3 = Razonable|NoCompra) = \frac{11}{30} \cdot \frac{7}{20} \cdot \frac{8}{20} = \frac{77}{1500}$$

$$P(Compra|\mathbf{x}) = \frac{P(Compra)P(\mathbf{x}|Compra)}{P(Compra)P(\mathbf{x}|Compra) + P(NoCompra)P(\mathbf{x}|NoCompra)} =$$

$$\frac{\frac{9}{14} \cdot \frac{728}{6221}}{\frac{9}{14} \cdot \frac{728}{6221} + \frac{5}{14} \cdot \frac{77}{1500}} = \frac{3135}{3899} \approx 0{,}8041$$

$$P(NoCompra|\mathbf{x}) = \frac{P(NoCompra)P(\mathbf{x}|NoCompra)}{P(Compra)P(\mathbf{x}|Compra) + P(NoComp)P(\mathbf{x}|NoComp)} =$$

$$\frac{\frac{5}{14} \cdot \frac{77}{1500}}{\frac{9}{14} \cdot \frac{728}{6221} + \frac{5}{14} \cdot \frac{77}{1500}} = \frac{764}{3899} \approx 0{,}1959$$

Por tanto, se predice que esta persona puede comprar el ordenador.

**Ejercicio 8.4**

Consideremos una tarea de diagnóstico médico. Sabemos que sobre un 10.6 % del total de la población tiene diabetes. Hay un test de laboratorio que produce un resultado positivo correcto en el 99 % de los casos en que la enfermedad está presente, y un resultado negativo correcto en el 94 % de los casos donde la enfermedad no está presente.

a) Suponer que observamos un paciente con el que el test devuelve positivo como resultado. Calcular la probabilidad de que él/ella sufra diabetes.

b) Un segundo test (se asume que es independiente del primero) es llevado a cabo, y devuelve un resultado negativo. Recalcular la probabilidad de que él/ella sufra diabetes.

**Solución**

En este ejercicio $C = 2$, $V = \{Diabet, NoDiabet\}$. Por tanto, para cada $y \in V$ tenemos:

$$P(y|\mathbf{x}) = \frac{P(y)P(\mathbf{x}|y)}{P(Diabet)P(\mathbf{x}|Diabet) + P(NoDiabet)P(\mathbf{x}|NoDiabet)}$$

Las probabilidades de clase a priori son:

$P(Diabet) = 10{,}6$

$P(NoDiabet) = 89{,}4$

a) El dato observado es $\mathbf{x} = (Positivo)$. Luego se tiene:

$P(\mathbf{x}|Diabet) = 0{,}99$

$P(\mathbf{x}|NoDiabet) = 0{,}06$

Ahora podemos calcular la probabilidad de sufrir diabetes:

$$P(Diabet|\mathbf{x}) = \frac{P(Diabet)P(\mathbf{x}|Diabet)}{P(Diabet)P(\mathbf{x}|Diabet) + P(NoDiabet)P(\mathbf{x}|NoDiabet)} =$$

$$\frac{10{,}6 \cdot 0{,}99}{10{,}6 \cdot 0{,}99 + 89{,}4 \cdot 0{,}06} \approx 0{,}6617$$

b) Después del segundo test el dato observado es $\mathbf{x} = (Positivo, Negativo)$. Luego se tiene:

$P(\mathbf{x}|Diabet) = 0{,}99 \cdot 0{,}01 = 0{,}0099$

$P(\mathbf{x}|NoDiabet) = 0{,}06 \cdot 0{,}94 = 0{,}0564$

Ahora podemos recalcular la probabilidad de sufrir diabetes:

$$P(Diabet|\mathbf{x}) = \frac{P(Diabet)P(\mathbf{x}|Diabet)}{P(Diabet)P(\mathbf{x}|Diabet) + P(NoDiabet)P(\mathbf{x}|NoDiabet)} =$$

$$\frac{10{,}6 \cdot 0{,}0099}{10{,}6 \cdot 0{,}0099 + 89{,}4 \cdot 0{,}0564} \approx 0{,}0204$$

**Ejercicio 8.5** Calcular las siguientes medidas de rendimiento dada la siguiente matriz de confusión:

a) Exactitud (Accuracy).

b) Verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP), falsos negativos (FN).

c) Precisión (precision), tasa de falsos positivos (fallout), exhaustividad (recall), F-medida (F-measure).

| | | Actual class | |
|---|---|---|---|
| | | Positive | Negative |
| Predicted class | Positive | 182 | 31 |
| | Negative | 12 | 245 |

**Solución**

a) $Accuracy = \dfrac{n_{11} + n_{22}}{n_{11} + n_{12} + n_{21} + n_{22}} = \dfrac{182 + 245}{182 + 31 + 12 + 245} \approx 0{,}9085$

b) $TP = n_{11} = 182$

$TN = n_{22} = 245$

$FP = n_{12} = 31$

$FN = n_{21} = 12$

c) $Precision = \dfrac{182}{182+31} \approx 0{,}8545$

$Fallout = \dfrac{31}{31+245} \approx 0{,}1123$

$Recall = \dfrac{182}{182+12} \approx 0{,}9381$

$F-measure = 2 \cdot \dfrac{0{,}8545 \cdot 0{,}9381}{0{,}8545+0{,}9381} = 0{,}8944$

**Ejercicio 8.6** Dado el siguiente conjunto de datos de entrenamiento, se pide usar el clasificador de redes bayesianas con $m$ constante de estimación $m = 3$ para predecir el valor de la variable JugarTenis para el siguiente patrón de test: (Nublado, Fresco, Normal, Débil).

| Día | Pronóstico | Temperatura | Humedad | Viento | JugarTenis |
|---|---|---|---|---|---|
| D1 | Soleado | Caluroso | Alto | Débil | No |
| D2 | Soleado | Caluroso | Alto | Fuerte | No |
| D3 | Nublado | Caluroso | Alto | Débil | Sí |
| D4 | Lluvia | Suave | Alto | Débil | Sí |
| D5 | Lluvia | Frío | Normal | Débil | Sí |
| D6 | Lluvia | Frío | Normal | Fuerte | No |
| D7 | Nublado | Frío | Normal | Fuerte | Sí |
| D8 | Soleado | Suave | Alto | Débil | No |
| D9 | Soleado | Frío | Normal | Débil | Sí |
| D10 | Lluvia | Suave | Normal | Débil | Sí |
| D11 | Soleado | Suave | Normal | Fuerte | Sí |
| D12 | Nublado | Suave | Alto | Fuerte | Sí |
| D13 | Nublado | Caluroso | Normal | Débil | Sí |
| D14 | Lluvia | Suave | Alto | Fuerte | No |

**Solución**

En este ejercicio $C = 2$, $V = \{Sí, No\}$, y $D = 4$. Por tanto, para cada $y \in V$ tenemos:

$$P(y|\mathbf{x}) = \frac{P(y)P(\mathbf{x}|y)}{P(Sí)P(\mathbf{x}|Sí) + P(No)P(\mathbf{x}|No)}$$

$$P(\mathbf{x}|y) = \prod_{d=1}^{4} P(x_d|y)$$

De las frecuencias de clase en el conjunto de entrenamiento se obtienen las probabilidades de clase a priori:

$$P(Sí) = \frac{9}{14}$$

$$P(No) = \frac{5}{14}$$

Por otro lado:

$$P(x_1|y) = \frac{n' + 1}{n + 3}$$

$$P(x_2|y) = \frac{n' + 1}{n + 3}$$

$$P(x_3|y) = \frac{n' + \frac{3}{2}}{n + 3}$$

$$P(x_4|y) = \frac{n' + \frac{3}{2}}{n + 3}$$

Ahora calculamos las probabilidades a posteriori:

$$P(x_1 = Nublado|Sí) = \frac{4 + 1}{9 + 3} = \frac{5}{12}$$

$$P(x_1 = Nublado|No) = \frac{0 + 1}{5 + 3} = \frac{1}{8}$$

$$P(x_2 = Fresco|Sí) = \frac{3 + 1}{9 + 3} = \frac{1}{3}$$

$$P(x_2 = Fresco|No) = \frac{1 + 1}{5 + 3} = \frac{1}{4}$$

$$P(x_3 = Normal|Sí) = \frac{6 + \frac{3}{2}}{9 + 3} = \frac{3}{8}$$

$$P(x_3 = Normal|No) = \frac{1 + \frac{3}{2}}{5 + 3} = \frac{5}{16}$$

$P(x_4 = \textit{Débil}|\textit{Sí}) = \dfrac{6+\dfrac{3}{2}}{9+3} = \dfrac{5}{8}$

$P(x_4 = \textit{Débil}|No) = \dfrac{2+\dfrac{3}{2}}{5+3} = \dfrac{7}{16}$

Por último, calculamos las probabilidades de que la variable *JugarTenis* tenga valor *Sí* o *No*:

$P(\mathbf{x}|\textit{Sí}) = P(x_1 = Nublado|\textit{Sí})P(x_2 = Fresco|\textit{Sí})P(x_3 = Normal|\textit{Sí})$
$P(x_4 = \textit{Débil}|\textit{Sí}) = \dfrac{5}{12} \cdot \dfrac{1}{3} \cdot \dfrac{5}{8} \cdot \dfrac{5}{8} = \dfrac{125}{2304}$

$P(\mathbf{x}|No) = P(x_1 = Nublado|No)P(x_2 = Fresco|No)P(x_3 = Normal|No)$
$P(x_4 = \textit{Débil}|No) = \dfrac{1}{8} \cdot \dfrac{1}{4} \cdot \dfrac{5}{16} \cdot \dfrac{7}{16} = \dfrac{35}{8192}$

$P(\textit{Sí}|\mathbf{x}) = \dfrac{P(\textit{Sí})P(\mathbf{x}|\textit{Sí})}{P(\textit{Sí})P(\mathbf{x}|\textit{Sí}) + P(No)P(\mathbf{x}|No)} =$

$\dfrac{\dfrac{9}{14} \cdot \dfrac{125}{2304}}{\dfrac{9}{14} \cdot \dfrac{125}{2304} + \dfrac{5}{14} \cdot \dfrac{35}{8192}} = \approx 0{,}9580$

$P(No|\mathbf{x}) = 1 - P(\textit{Sí}|\mathbf{x}) \approx 0{,}0419$

Por tanto, *JugarTenis* se predice ser *Sí*.

# Tema 9

# Árboles de decisión

**Ejercicio 9.1**

Queremos clasificar si dado un globo está hinchado en base a cuatro atributos: el color del globo, el tamaño del globo, la 'acción' que la persona que sostiene el globo está desarrollando, y la edad de la persona que sostiene el globo. Dado el siguiente conjunto de entrenamiento, usa el algoritmo ID3 para obtener un árbol de decisión. ¿Cuál es la clase predicha para la muestra de test (Morado, Grande, Bajar, Niño)?

| Muestra | Color | Tamaño | Acción | Edad | ¿Hinchado? |
|---|---|---|---|---|---|
| 1 | Amarillo | Pequeño | Estirar | Adulto | Sí |
| 2 | Amarillo | Pequeño | Estirar | Niño | Sí |
| 3 | Amarillo | Pequeño | Bajar | Adulto | Sí |
| 4 | Amarillo | Pequeño | Bajar | Niño | No |
| 5 | Amarillo | Pequeño | Bajar | Niño | No |
| 6 | Amarillo | Grande | Estirar | Adulto | Sí |
| 7 | Amarillo | Grande | Estirar | Niño | Sí |
| 8 | Amarillo | Grande | Bajar | Adulto | Sí |
| 9 | Amarillo | Grande | Bajar | Niño | No |
| 10 | Amarillo | Grande | Bajar | Niño | No |
| 11 | Morado | Pequeño | Estirar | Adulto | Sí |
| 12 | Morado | Pequeño | Estirar | Niño | Sí |
| 13 | Morado | Pequeño | Bajar | Adulto | Sí |
| 14 | Morado | Pequeño | Bajar | Niño | No |
| 15 | Morado | Pequeño | Bajar | Niño | No |
| 16 | Morado | Grande | Estirar | Adulto | Sí |
| 17 | Morado | Grande | Estirar | Niño | Sí |
| 18 | Morado | Grande | Bajar | Adulto | Sí |
| 19 | Morado | Grande | Bajar | Niño | No |
| 20 | Morado | Grande | Bajar | Niño | No |

**Solución**

La entropía inicial es $H(\mathbf{x}_1, ..., \mathbf{x}_{20}) = 0{,}971$. Calculamos la ganancia de información para dividir cada atributo en el nodo raíz:

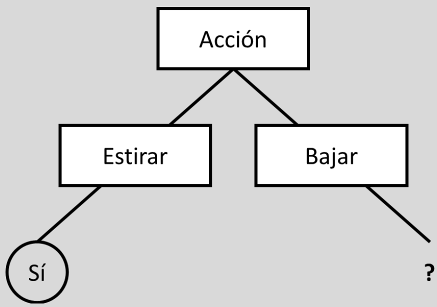
$$IG(\mathbf{x}_1, ..., \mathbf{x}_{20}, Color) = 0{,}971 - \frac{10}{20} \cdot 0{,}971 - \frac{10}{20} \cdot 0{,}971 = 0$$

$$IG(\mathbf{x}_1, ..., \mathbf{x}_{20}, \textit{Tamaño}) = 0{,}971 - \frac{10}{20} \cdot 0{,}971 - \frac{10}{20} \cdot 0{,}971 = 0$$

$$IG(\mathbf{x}_1, ..., \mathbf{x}_{20}, \textit{Acción}) = 0{,}971 - \frac{8}{20} \cdot 0 - \frac{12}{20} \cdot 0{,}918 = 0{,}42$$

$$IG(\mathbf{x}_1, ..., \mathbf{x}_{20}, Edad) = 0{,}971 - \frac{8}{20} \cdot 0 - \frac{12}{20} \cdot 0{,}918 = 0{,}42$$

Por lo tanto, podemos elegir *Acción* o *Edad* para dividir en el nodo raíz. Si elegimos *Acción* obtenemos:




Ahora debemos elegir un atributo para dividir en el nodo $\textit{Acción} = Bajar$, donde $S_{Bajar} = \{\mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_8, \mathbf{x}_9, \mathbf{x}_{10}, \mathbf{x}_{13}, \mathbf{x}_{14}, \mathbf{x}_{15}, \mathbf{x}_{18}, \mathbf{x}_{19}, \mathbf{x}_{20}\}$

| Muestra | Color | Tamaño | Edad | ¿Hinchado? |
|---|---|---|---|---|
| 3 | Amarillo | Pequeño | Adulto | Sí |
| 4 | Amarillo | Pequeño | Niño | No |
| 5 | Amarillo | Pequeño | Niño | No |
| 8 | Amarillo | Grande | Adulto | Sí |
| 9 | Amarillo | Grande | Niño | No |
| 10 | Amarillo | Grande | Niño | No |
| 13 | Morado | Pequeño | Adulto | Sí |
| 14 | Morado | Pequeño | Niño | No |
| 15 | Morado | Pequeño | Niño | No |
| 18 | Morado | Grande | Adulto | Sí |
| 19 | Morado | Grande | Niño | No |
| 20 | Morado | Grande | Niño | No |

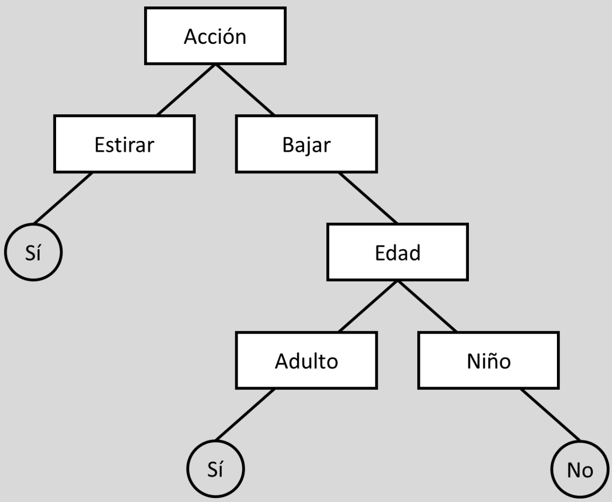
La entropía inicial es $H(S_{Bajar}) = 0{,}918$.

$$IG(S_{Bajar}, Color) = 0{,}918 - \frac{6}{12} \cdot 0{,}918 - \frac{6}{12} \cdot 0{,}918 = 0$$

$$IG(S_{Bajar}, \mathit{Tamaño}) = 0{,}918 - \frac{6}{12} \cdot 0{,}918 - \frac{6}{12} \cdot 0{,}918 = 0$$

$$IG(S_{Bajar}, Edad) = 0{,}918 - \frac{4}{12} \cdot 0 - \frac{8}{12} \cdot 0 = 0{,}918$$

Por lo tanto, debemos elegir *Edad* para dividir en el nodo *Acción* = *Bajar*, y de esta manera se consigue el árbol final:



La clase predicha para la muestra de test *(Morado, Grande, Bajar, Niño)* es *No* (no hinchado), ya que $Acción = Bajar$ y $Edad = Niño$.

**Ejercicio 9.2** Consideremos el siguiente conjunto de datos de entrenamiento para una tarea de evaluación del riesgo de crédito (Riesgo es el atributo objetivo). Usa el algoritmo ID3 para obtener un árbol de decisión. ¿Cuál es la clase predicha para la muestra de test (Malo, Bajo, Adecuado, Superior a 35K)?

| Muestra | Riesgo | Historial | Deuda | Aval | Sueldo (euros) |
|---|---|---|---|---|---|
| 1 | Alto | Malo | Alto | Ninguno | 0-15K |
| 2 | Alto | Desconocido | Alto | Ninguno | 15-35K |
| 3 | Moderado | Desconocido | Bajo | Ninguno | 15-35K |
| 4 | Alto | Desconocido | Bajo | Ninguno | 0-15K |
| 5 | Bajo | Desconocido | Bajo | Ninguno | $> 35$K |
| 6 | Bajo | Desconocido | Bajo | Adecuado | $> 35$K |
| 7 | Alto | Malo | Bajo | Ninguno | 0-15K |
| 8 | Moderado | Malo | Bajo | Adecuado | $> 35$K |
| 9 | Bajo | Bueno | Bajo | Ninguno | $> 35$K |
| 10 | Bajo | Bueno | Alto | Adecuado | $> 35$K |
| 11 | Alto | Bueno | Alto | Ninguno | 0-15K |
| 12 | Moderado | Bueno | Alto | Ninguno | 15-35K |
| 13 | Bajo | Bueno | Alto | Ninguno | $> 35$K |
| 14 | Alto | Malo | Alto | Ninguno | 15-35K |

**Solución**

La entropía inicial es $H(\mathbf{x}_1, ..., \mathbf{x}_{14}) = 1{,}531$. Calculamos la ganancia de información para dividir cada atributo en el nodo raíz:

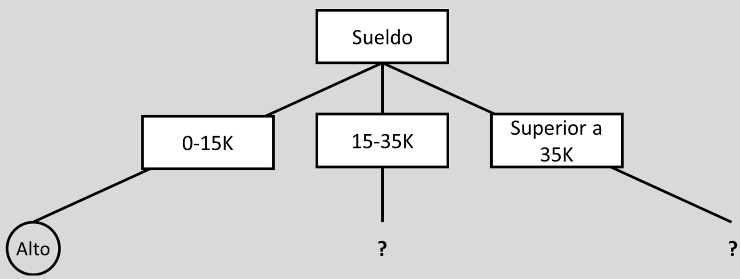
$$IG(\mathbf{x}_1, ..., \mathbf{x}_{14}, Historial) = 1{,}531 - \frac{4}{14} \cdot 0{,}811 - \frac{5}{14} \cdot 1{,}371 - \frac{5}{14} \cdot 1{,}522 = 0{,}266$$

$$IG(\mathbf{x}_1, ..., \mathbf{x}_{14}, Deuda) = 1{,}531 - \frac{7}{14} \cdot 1{,}379 - \frac{7}{14} \cdot 1{,}557 = 0{,}063$$

$$IG(\mathbf{x}_1, ..., \mathbf{x}_{14}, Aval) = 1{,}531 - \frac{3}{14} \cdot 0{,}918 - \frac{11}{14} \cdot 1{,}435 = 0{,}207$$

$$IG(\mathbf{x}_1, ..., \mathbf{x}_{14}, Sueldo) = 1{,}531 - \frac{4}{14} \cdot 0 - \frac{4}{14} \cdot 1 - \frac{6}{14} \cdot 0{,}650 = 0{,}967$$

Por lo tanto, debemos elegir *Sueldo* para dividir en el nodo raíz:



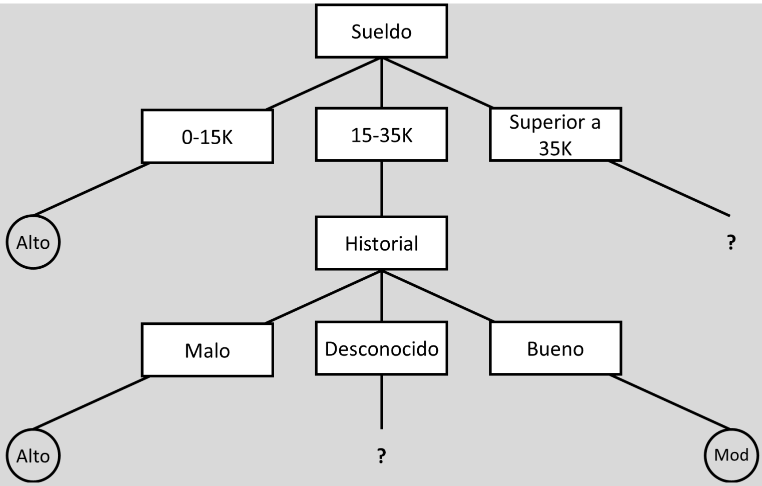

Ahora debemos elegir un atributo para dividir en el nodo *Sueldo = 15-35K*, donde $S_{15\text{-}35K} = \{\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_{12}, \mathbf{x}_{14}\}$. La entropía inicial es $H(S_{15\text{-}35K}) = 1$.

$$IG(S_{15\text{-}35K}, Historial) = 1 - \frac{1}{4} \cdot 0 - \frac{1}{4} \cdot 0 - \frac{2}{4} \cdot 1 = 0{,}5$$

$$IG(S_{15\text{-}35K}, Deuda) = 1 - \frac{3}{4} \cdot 0{,}918 - \frac{1}{4} \cdot 0 = 0{,}311$$

$$IG(S_{15\text{-}35K}, Aval) = 1 - \frac{4}{4} \cdot 1 = 0$$

Por lo tanto, debemos elegir *Historial* para dividir en el nodo *Sueldo = 15-35K*:

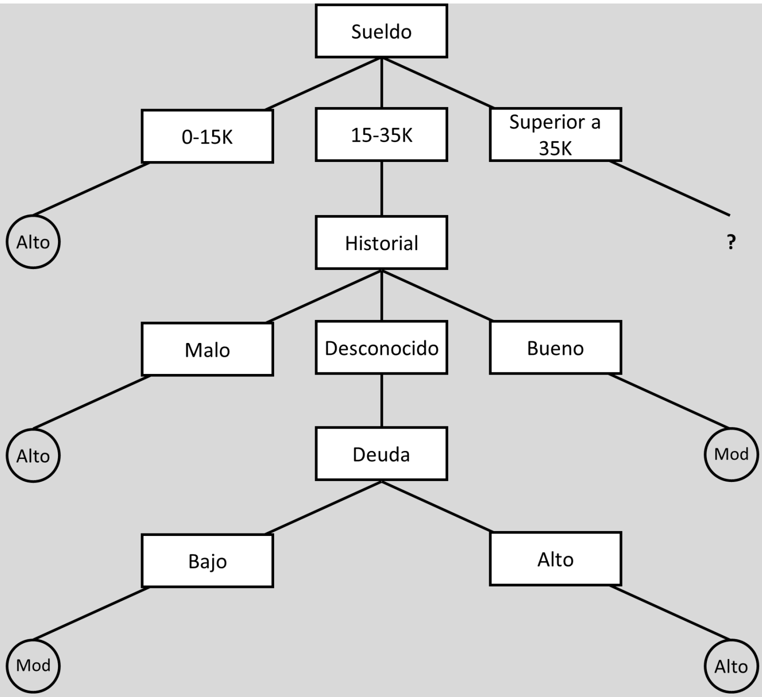



Ahora debemos elegir un atributo para dividir en el nodo $Historial = Desconocido$, donde $S_{Desconocido} = \{\mathbf{x}_2, \mathbf{x}_3\}$. La entropía inicial es $H(S_{Desconocido}) = 1$.

$IG(S_{Desconocido}, Deuda) = 1 - \dfrac{1}{2} \cdot 0 - \dfrac{1}{2} \cdot 0 = 1$

$IG(S_{Desconocido}, Aval) = 1 - \dfrac{2}{2} \cdot 1 = 0$

Por lo tanto, debemos elegir $Deuda$ para dividir en el nodo $Historial = Desconocido$:



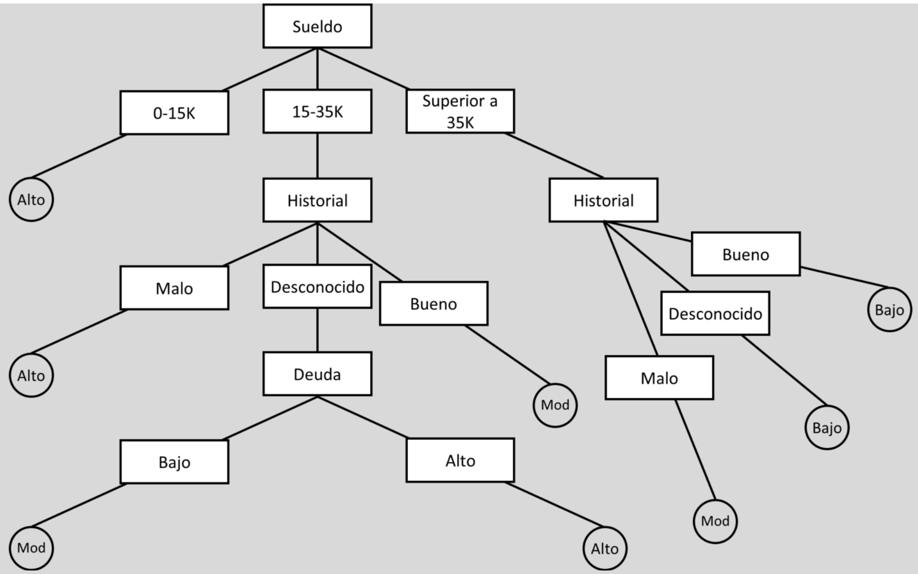

Ahora debemos elegir un atributo para dividir en el nodo *Sueldo* $=$ *Superior a 35K*, donde $S_{Superior\ a\ 35K} = \{\mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_8, \mathbf{x}_9, \mathbf{x}_{10}, \mathbf{x}_{13}\}$. La entropía inicial es $H(S_{Superior\ a\ 35K}) = 0{,}65$.

$$IG(S_{Superior\ a\ 35K}, Historial) = 0{,}65 - \frac{2}{6} \cdot 0 - \frac{3}{6} \cdot 0 - \frac{1}{6} \cdot 0 = 0{,}65$$

$$IG(S_{Superior\ a\ 35K}, Deuda) = 0{,}65 - \frac{4}{6} \cdot 0{,}811 - \frac{2}{6} \cdot 0 = 0{,}109$$

$$IG(S_{Superior\ a\ 35K}, Aval) = 0{,}65 - \frac{3}{6} \cdot 0 - \frac{3}{6} \cdot 0{,}918 = 0{,}191$$

Por lo tanto, debemos elegir *Historial* para dividir en el nodo *Sueldo* $=$ *Superior a 35K*. El árbol final queda:




La clase predicha para la muestra de test *(Malo, Bajo, Adecuado, Superior a 35K)* es *Moderado.*

**Ejercicio 9.3** Consideremos el siguiente conjunto de datos de entrenamiento para predecir el sexo de una persona (Sexo es el atributo objetivo). Usa el algoritmo ID3 para obtener un árbol de decisión. ¿Cuál es la clase predicha para la muestra de test (Largo, Alto, Joven)?

| Persona | Pelo | Peso | Edad | Sexo |
|---|---|---|---|---|
| 1 | Corto | Alto | Joven | Masculino |
| 2 | Largo | Bajo | Joven | Femenino |
| 3 | Corto | Bajo | Joven | Masculino |
| 4 | Largo | Bajo | Joven | Femenino |
| 5 | Corto | Bajo | Joven | Femenino |
| 6 | Corto | Alto | Anciano | Masculino |
| 7 | Largo | Bajo | Anciano | Femenino |
| 8 | Largo | Alto | Joven | Masculino |
| 9 | Largo | Alto | Anciano | Masculino |

**Solución**

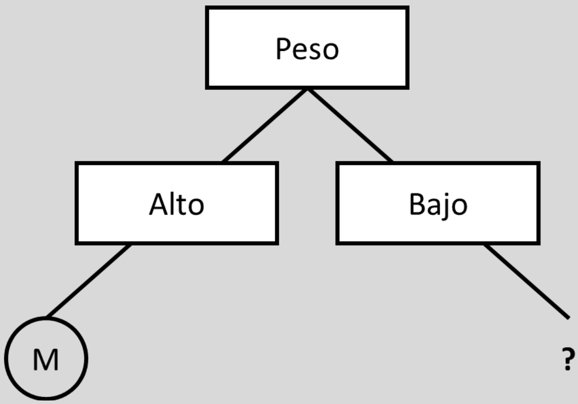
La entropía inicial es $H(\mathbf{x}_1, ..., \mathbf{x}_9) = 0{,}991$. Calculamos la ganancia de información para dividir cada atributo en el nodo raíz:

$IG(\mathbf{x}_1, ..., \mathbf{x}_9, Pelo) = 0{,}991 - \frac{5}{9} \cdot 0{,}971 - \frac{4}{9} \cdot 0{,}811 = 0{,}091$

$IG(\mathbf{x}_1, ..., \mathbf{x}_9, Peso) = 0{,}991 - \frac{4}{9} \cdot 0 - \frac{5}{9} \cdot 0{,}722 = 0{,}590$

$IG(\mathbf{x}_1, ..., \mathbf{x}_9, Edad) = 0{,}991 - \frac{3}{9} \cdot 0{,}918 - \frac{6}{9} \cdot 1 = 0{,}018$

Por lo tanto, debemos elegir *Peso* para dividir en el nodo raíz:

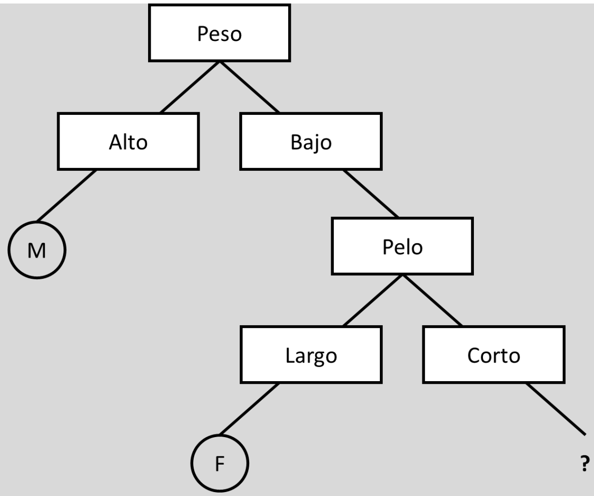



Ahora debemos elegir un atributo para dividir en el nodo $Peso = Bajo$, donde $S_{Bajo} = \{\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_7\}$. La entropía inicial es $H(S_{Bajo}) = 0{,}722$.

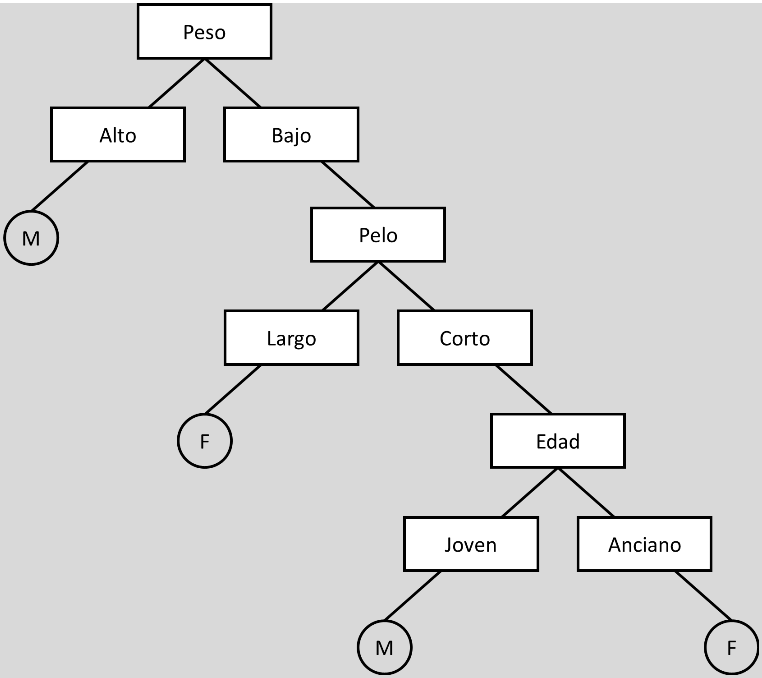
$IG(S_{Bajo}, Pelo) = 0{,}722 - \frac{3}{5} \cdot 0 - \frac{2}{5} \cdot 1 = 0{,}322$

$IG(S_{Bajo}, Edad) = 0{,}722 - \frac{1}{5} \cdot 0 - \frac{4}{5} \cdot 0{,}811 = 0{,}073$

Por lo tanto, debemos elegir *Pelo* para dividir en el nodo $Peso = Bajo$:




Ahora el único atributo disponible para dividir en el nodo $Pelo = Largo$ es $Edad$, donde $S_{Largo} = \{\mathbf{x}_3, \mathbf{x}_5\}$. Pero encontramos que: 1) no hay muestras para $Edad = Anciano$ ; 2) no hay un valor objetivo más frecuente (hay una muestra de cada clase). Una posible solución para romper este empate es la siguiente:




La clase predicha para la muestra de test *(Largo, Alto, Joven)* es *Masculino.*

**Ejercicio 9.4** Consideremos el siguiente conjunto de datos de entrenamiento para predecir si una persona se quema por el sol (Resultado es el atributo objetivo). Usa el algoritmo ID3 para obtener un árbol de decisión. ¿Cuál es la clase predicha para la muestra de test (Rojo, Alto, Medio, Sí)?

| Persona | Pelo | Altura | Peso | Loción | Resultado |
|---|---|---|---|---|---|
| 1 | Rubio | Medio | Ligero | No | Sí |
| 2 | Rubio | Alto | Medio | Sí | No |
| 3 | Castaño | Bajo | Medio | Sí | No |
| 4 | Rubio | Bajo | Medio | No | Sí |
| 5 | Pelirrojo | Medio | Pesado | No | Sí |
| 6 | Castaño | Alto | Pesado | No | No |
| 7 | Castaño | Medio | Pesado | No | No |
| 8 | Rubio | Bajo | Ligero | Sí | No |

**Solución**

La entropía inicial es $H(\mathbf{x}_1, ..., \mathbf{x}_8) = 0{,}954$. Calculamos la ganancia de información para dividir cada atributo en el nodo raíz:

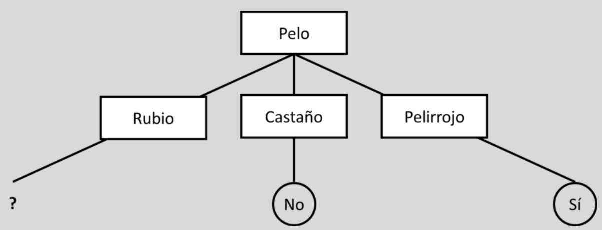
$$IG(\mathbf{x}_1, ..., \mathbf{x}_8, Pelo) = 0{,}954 - \frac{4}{8} \cdot 1 - \frac{3}{8} \cdot 0 - \frac{1}{8} \cdot 0 = 0{,}454$$

$$IG(\mathbf{x}_1, ..., \mathbf{x}_8, Altura) = 0{,}954 - \frac{3}{8} \cdot 0{,}918 - \frac{3}{8} \cdot 0{,}918 - \frac{2}{8} \cdot 0 = 0{,}265$$

$$IG(\mathbf{x}_1, ..., \mathbf{x}_8, Peso) = 0{,}954 - \frac{3}{8} \cdot 0{,}918 - \frac{3}{8} \cdot 0{,}918 - \frac{2}{8} \cdot 1 = 0{,}015$$

$$IG(\mathbf{x}_1, ..., \mathbf{x}_8, Loción) = 0{,}954 - \frac{5}{8} \cdot 0{,}971 - \frac{3}{8} \cdot 0 = 0{,}347$$

Por lo tanto, debemos elegir *Pelo* para dividir en el nodo raíz:



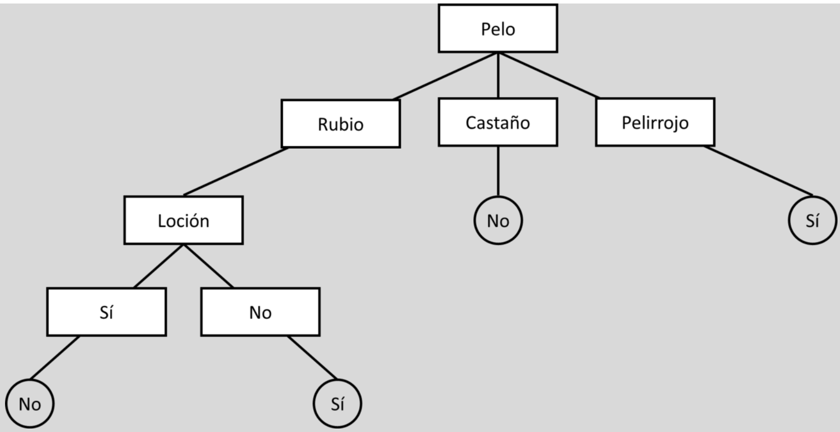

Ahora debemos elegir un atributo para dividir en el nodo $Pelo = Rubio$, donde $S_{Rubio} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_4, \mathbf{x}_8\}$. La entropía inicial es $H(S_{Rubio}) = 1$.

$$IG(S_{Rubio}, Altura) = 1 - \frac{1}{4} \cdot 0 - \frac{2}{4} \cdot 1 - \frac{1}{4} \cdot 0 = 0{,}5$$

$$IG(S_{Rubio}, Peso) = 1 - \frac{2}{4} \cdot 1 - \frac{2}{4} \cdot 1 = 0$$

$$IG(S_{Rubio}, Loción) = 1 - \frac{2}{4} \cdot 0 - \frac{2}{4} \cdot 0 = 1$$

Por lo tanto, debemos elegir *Loción* para dividir en el nodo $Pelo = Rubio$. El árbol final queda:




La clase predicha para la muestra de test *(Rojo, Alto, Medio, Sí)* es *Sí* (dicha persona se quema por el sol).

**Ejercicio 9.5** Ya que el algoritmo ID3 es un procedimiento de búsqueda local sobre el espacio de los árboles de decisión, es posible que no encuentre un árbol óptimo en términos de tamaño y rendimiento sobre el conjunto de entrenamiento. Proporciona un ejemplo de conjunto de datos para los que ID3 no devuelve el árbol más pequeño que clasifica correctamente todos los datos de entrenamiento.

**Solución**

Un posible conjunto de datos es el que se muestra a continuación. El árbol de decisión más pequeño tiene profundidad dos, y se logra derivando primero en A1 y después en A2. Sin embargo, ID3 elige A3, y produce un árbol de decisión de profundidad tres. Los detalles se dejan al lector.

| Muestra | A1 | A2 | A3 | Clase |
|---|---|---|---|---|
| 1 | V | V | V | + |
| 2 | V | V | V | + |
| 3 | V | F | F | - |
| 4 | V | F | V | - |
| 5 | F | F | V | + |
| 6 | F | F | F | + |
| 7 | F | V | F | - |
| 8 | F | V | V | - |

**Ejercicio 9.6** Proporciona un ejemplo de conjunto de datos para los que ID3 no devuelve un árbol de decisión que clasifica correctamente todos los datos de entrenamiento.

**Solución**

Un posible conjunto de datos es el que se muestra a continuación. Hay que tener en cuenta que las muestras 1 y 2 tienen los mismos valores para el vector de entrada (A1, A2, A3), mientras que los valores objetivo son diferentes.

| Muestra | A1 | A2 | A3 | Clase |
|---|---|---|---|---|
| 1 | V | F | V | + |
| 2 | V | F | V | - |
| 3 | V | F | F | - |
| 4 | V | V | V | - |
| 5 | F | F | V | + |
| 6 | F | F | F | + |
| 7 | F | V | F | - |
| 8 | F | V | V | - |

# Tema 10

# Agrupamiento

**Ejercicio 10.1** Sean $p_1 = (0, 0)$, $p_2 = (1, 1)$, $p_3 = (1, 0)$ y $p_4 = (2, 0{,}5)$. Considera las siguientes tres particiones:

| Partición | Cluster 1 | Cluster 2 |
|---|---|---|
| 1 | $p_1, p_2$ | $p_3, p_4$ |
| 2 | $p_1, p_4$ | $p_2, p_3$ |
| 3 | $p_1, p_2, p_3$ | $p_4$ |

a) Encuentra el agrupamiento que minimiza la suma de la distancia euclídea al cuadrado.

b) Encuentra el agrupamiento que minimiza la suma de la distancia de Manhattan.

c) Encuentra el agrupamiento que minimiza la suma de la distancia de Chebychev.

**Solución**

Primero tenemos que calcular el centroide de cada grupo. El centroide de un grupo es la media de los puntos que pertenecen a dicho grupo. Por tanto:

- Partición 1, centroide del grupo 1 ($c_{11}$) es: $c_{11} = \dfrac{(p_1 + p_2)}{2} = (0{,}5, 0{,}5)$

- Partición 1, centroide del grupo 2 ($c_{12}$) es: $c_{12} = \frac{(p_3 + p_4)}{2} = (1{,}5, 0{,}25)$
- Partición 2, centroide del grupo 1 ($c_{21}$) es: $c_{21} = \frac{(p_1 + p_4)}{2} = (1, 0{,}25)$
- Partición 2, centroide del grupo 2 ($c_{22}$) es: $c_{22} = \frac{(p_2 + p_3)}{2} = (1, 0{,}5)$
- Partición 3, centroide del grupo 1 ($c_{31}$) es: $c_{31} = \frac{(p_1 + p_2 + p_3)}{3} = (\frac{2}{3}, \frac{1}{3})$
- Partición 3, centroide del grupo 2 ($c_{32}$) es: $c_{32} = p_4 = (2, 0{,}5)$

El segundo paso es calcular la suma de las distancias de cada punto al centroide de su grupo. Por tanto:

- Partición 1: $d(c_{11}, p_1) + d(c_{11}, p_2) + d(c_{12}, p_3) + d(c_{12}, p_4)$
- Partición 2: $d(c_{21}, p_1) + d(c_{22}, p_2) + d(c_{22}, p_3) + d(c_{21}, p_4)$
- Partición 3: $d(c_{31}, p_1) + d(c_{31}, p_2) + d(c_{31}, p_3) + d(c_{32}, p_4)$

Las definiciones de las distancias propuestas entre dos puntos $q_1(x_1, y_1)$ y $q_2(x_2, y_2)$ son:

- Euclídea al cuadrado: $d(q_1, q_2) = (x_1 - x_2)^2 + (y_1 - y_2)^2$
- Manhattan: $d(q_1, q_2) = |x_1 - x_2| + |y_1 - y_2|$
- Chebychev: $d(q_1, q_2) = max\{|x_1 - x_2|, |y_1 - y_2|\}$

Por tanto, se tiene:

- Distancia euclídea al cuadrado

  Partición 1:

  $d(c_{11}, p_1) + d(c_{11}, p_2) + d(c_{12}, p_3) + d(c_{12}, p_4) =$

  $0{,}5 + 0{,}5 + 0{,}3125 + 0{,}3125 = 1{,}625$

Partición 2:

$d(c_{21},p_1)+d(c_{22},p_2)+d(c_{22},p_3)+d(c_{21},p_4)=$

$1{,}0625+1{,}0625+0{,}25+0{,}25=2{,}625$

Partición 3:

$d(c_{31},p_1)+d(c_{31},p_2)+d(c_{31},p_3)+d(c_{32},p_4)=$

$0{,}5445+0{,}5645+0{,}2245+0=1{,}3335$

- Distancia Manhattan

  Partición 1:

  $d(c_{11},p_1)+d(c_{11},p_2)+d(c_{12},p_3)+d(c_{12},p_4)=$

  $1+1+0{,}75+0{,}75=3{,}5$

  Partición 2:

  $d(c_{21},p_1)+d(c_{22},p_2)+d(c_{22},p_3)+d(c_{21},p_4)=$

  $1{,}25+1{,}25+0{,}5+0{,}5=3{,}5$

  Partición 3:

  $d(c_{31},p_1)+d(c_{31},p_2)+d(c_{31},p_3)+d(c_{32},p_4)=$

  $0{,}99+1{,}01+0{,}67+0=2{,}67$

- Distancia Chebychev

  Partición 1:

  $d(c_{11},p_1)+d(c_{11},p_2)+d(c_{12},p_3)+d(c_{12},p_4)=$

  $max\{|0{,}5-0|,|0{,}5-0|\}+max\{|0{,}5-1|,|0{,}5-1|\}+max\{|1{,}5-1|,|0{,}25-0|\}+max\{|1{,}5-2|,|0{,}25-0{,}5|\}=$

  $max\{0{,}5,0{,}5\}+max\{0{,}5,0{,}5\}+max\{0{,}5,0{,}25\}+max\{0{,}5,0{,}25\}=$

  $0{,}5+0{,}5+0{,}5+0{,}5=2$

  Partición 2:

  $d(c_{21},p_1)+d(c_{22},p_2)+d(c_{22},p_3)+d(c_{21},p_4)=$

  $1+1+0{,}5+0{,}5=3$

  Partición 3:

  $d(c_{31},p_1)+d(c_{31},p_2)+d(c_{31},p_3)+d(c_{32},p_4)=$

$$\frac{2}{3} + \frac{2}{3} + \frac{1}{3} + 0 = \frac{5}{3} = 1{,}67$$

De esta forma, se puede completar la siguiente tabla:

| | Errores | | |
|---|---|---|---|
| Partición | Dist. euclídea | Dist. Manhattan | Dist. Chebychev |
| 1 | 1.625 | 3.5 | 2 |
| 2 | 2.625 | 3.5 | 3 |
| 3 | **1.3335** | **2.67** | **1.67** |

En los tres casos, la Partición 3 es la mejor.

Obsérvese que en dicha partición, $p_3$ es el centroide o prototipo del grupo 1, es decir, la mejor muestra representativa que minimiza el error del grupo.

**Ejercicio 10.2** Calcular los siguientes índices de rendimiento para los datos del ejercicio anterior e indicar el mejor agrupamiento en cada caso.

a) Índice de Davies-Bouldin (DB)

b) Índice de Dunn (D)

**Solución**

La versión simplificada de las ecuaciones (2 grupos en este caso) es la siguiente:

$$DB = \frac{\sigma_1 + \sigma_2}{d(\mu_1, \mu_2)}$$

$$D = \frac{d(\mu_1, \mu_2)}{max(\Delta_1, \Delta_2)}$$

Por tanto, los datos pedidos son:

| Partición | DB | D |
|---|---|---|
| 1 | 1.2283 | 0.5 |
| 2 | 6.1231 | 0.1213 |
| 3 | **0.4867** | **0.6518** |

Los mejores resultados se encuentran resaltados en **negrita**.

**Ejercicio 10.3** Aplicar el algoritmo k-means (se asumen 2 grupos) a los datos del ejercicio 1, utilizando los siguientes puntos de inicio:

a) Sean $m_1 = (0,0)$ y $m_2 = (1,1)$

b) Sean $m_1 = (1,1)$ y $m_2 = (2,2)$

Compara la salida del algoritmo k-means con el agrupamiento propuesto en el ejercicio 1.

**Solución**

a) Centroides iniciales $m_1 = (0,0)$ y $m_2 = (1,1)$

Calculamos las distancias. Obsérvese que se ha usado la suma de las distancias euclídeas al cuadrado:

$d(p_1, m_1) = 0$ $d(p_2, m_1) = 2$ $d(p_3, m_1) = 1$ $d(p_4, m_1) = 4{,}25$
$d(p_1, m_2) = 2$ $d(p_2, m_2) = 0$ $d(p_3, m_2) = 1$ $d(p_4, m_2) = 1{,}25$

Y ahora asignamos cada muestra al centroide más cercano:

| Muestra | $p_1$ | $p_2$ | $p_3$ | $p_4$ |
|---|---|---|---|---|
| Centroide más cercano | $m_1$ | $m_2$ | $m_1$ | $m_2$ |

Se actualizan los centroides, por lo que queda $m_1 = (0{,}5, 0)$ y $m_2 = (1{,}5, 0{,}75)$.

Y el algoritmo para porque no hay más cambios.

| Grupo 1 ($m_1$) | Grupo 2 ($m_2$) |
|---|---|
| $p_1, p_3$ | $p_2, p_4$ |

b) Centroides iniciales $m_1 = (1,1)$ y $m_2 = (2,2)$

Calculamos las distancias.

$d(p_1, m_1) = 2$ $d(p_2, m_1) = 0$ $d(p_3, m_1) = 1$ $d(p_4, m_1) = 1{,}25$
$d(p_1, m_2) = 8$ $d(p_2, m_2) = 2$ $d(p_3, m_2) = 5$ $d(p_4, m_2) = 2{,}25$

Y ahora asignamos cada muestra al centroide más cercano:

| Muestra | $p_1$ | $p_2$ | $p_3$ | $p_4$ |
|---|---|---|---|---|
| Centroide más cercano | $m_1$ | $m_1$ | $m_1$ | $m_1$ |

Obsérvese que hay un solo grupo, mientras que el segundo está muy lejos del resto de muestras. Ésta es una de las desventajas cuando la selección de los centroides iniciales es mala. Sin embargo, hay varias alternativas para continuar el algoritmo. Por ejemplo, podemos crear un nuevo grupo que consista en la muestra más alejada de su centroide. En este caso, un nuevo grupo con $p_1$ podría ser creado para continuar con el algoritmo. Se puede comprobar que la solución final es el mismo agrupamiento que en la sección previa.

**Ejercicio 10.4** Calcular los siguientes índices de rendimiento para los datos del ejercicio anterior y comparar los resultados con los obtenidos del ejercicio 2.

a) Índice de Davies-Bouldin (DB)

b) Índice de Dunn (D)

**Solución**

$$DB = \frac{\sigma_1 + \sigma_2}{d(\mu_1, \mu_2)} = \frac{0{,}5 + 0{,}5590}{1{,}25} = 0{,}8472$$

$$D = \frac{d(\mu_1, \mu_2)}{max(\Delta_1, \Delta_2)} = \frac{1{,}25}{2{,}0616} = 0{,}6063$$

**Ejercicio 10.5** Dada la siguiente tabla, aplica el algoritmo k-means (se asumen 2 grupos) con los siguientes puntos de inicio:

a) Sean $m_1 = (10, 10, 10)$ y $m_2 = (100, 100, 100)$

b) Sean $m_1 = (20, 20, 100)$ y $m_2 = (100, 100, 100)$

| | Características del núcleo de las células | | |
|---|---|---|---|
| Muestra | Radio (r) | Textura (t) | Perímetro (p) |
| 1 | 17.99 | 10.38 | 122.8 |
| 2 | 20.57 | 17.77 | 132.9 |
| 3 | 13.54 | 14.36 | 87.46 |
| 4 | 13.08 | 15.71 | 85.63 |
| 5 | 9.5 | 12.44 | 60.34 |
| 6 | 15.34 | 14.26 | 102.5 |
| 7 | 21.16 | 23.04 | 137.2 |
| 8 | 16.65 | 21.38 | 110 |
| 9 | 17.14 | 16.4 | 116 |
| 10 | 14.58 | 21.53 | 97.41 |

**Solución**

a) Centroides iniciales $m_1 = (10, 10, 10)$ y $m_2 = (100, 100, 100)$

| Dist. | Muest. | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Centr. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $m_1$ | **12787** | 15276 | **6031** | **5761** | **2540** | **8602** | 16474 | **10173** | **11327** | **7794** |
| $m_2$ | 15277 | **14153** | 14966 | 14866 | 17429 | 14524 | **13522** | 13228 | 14110 | 13460 |

El centroide más cercano está en **negrita**.

Los centroides actualizados son $m_1 = (14{,}7275, 15{,}8075, 97{,}7675)$ y $m_2 = (20{,}8650, 20{,}4050, 135{,}0500)$

| Dist. | Muest. | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Centr. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $m_1$ | **258** | **11** | 2355 | 2524 | 5774 | 1127 | **11** | 646 | 392 | 1457 |
| $m_2$ | 666 | 1272 | **109** | **150** | **1439** | **25** | 1648 | **184** | **338** | **32** |

Los centroides actualizados son $m_1 = (19{,}9067, 17{,}0633, 130{,}9667)$ y $m_2 = (14{,}2614, 16{,}5829, 94{,}1914)$

| Dist. | Muest. | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Centr. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $m_1$ | **115** | **4** | 1940 | 2103 | 5117 | 839 | **76** | 468 | **232** | 1174 |
| $m_2$ | 870 | 1539 | **50** | **75** | **1185** | **75** | 1939 | **278** | 483 | **34** |

Los centroides actualizados son $m_1 = (19{,}2150, 16{,}8975, 127{,}2250)$ y $m_2 = (13{,}7817, 16{,}6133, 90{,}5567)$

| Dist. | Muest. | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Centr. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $m_1$ | **63** | **34** | 1619 | 1769 | 4587 | 633 | **141** | **323** | **130** | 931 |
| $m_2$ | 1096 | 1840 | **14** | **25** | **948** | **150** | 2271 | 408 | 658 | **71** |

Los centroides actualizados son $m_1 = (18{,}7020, 17{,}7940, 123{,}7800)$ y $m_2 = (13{,}2080, 15{,}6600, 86{,}6680)$

El algoritmo para porque no se han producido cambios.

b) Centroides iniciales $m_1 = (20, 20, 100)$ y $m_2 = (100, 100, 100)$

| Dist. | Muest. | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Centr. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $m_1$ | **616** | **1087** | **230** | **272** | **1740** | **60** | **1394** | **113** | **277** | **38** |
| $m_2$ | 15277 | 14153 | 14966 | 14866 | 17429 | 14524 | 13522 | 13228 | 14110 | 13460 |

Obsérvese que solo hay un grupo ya que el segundo centroide está muy lejos del resto de muestras. Siguiendo la misma técnica del ejercicio 3, se puede comprobar que el resultado final es el mismo agrupamiento que en la sección previa.

**Ejercicio 10.6** Calcular los siguientes índices de rendimiento para los datos del ejercicio anterior.

a) Índice de Davies-Bouldin (DB)

b) Índice de Dunn (D)

**Solución**

a) $DB = \dfrac{(\sigma_1 + \sigma_2)}{(d(\mu_1, \mu_2))} = \dfrac{(10{,}7763 + 11{,}5482)}{37{,}5771} = 0{,}5941$

b) $D = \dfrac{d(\mu_1, \mu_2)}{max(\Delta_1, \Delta_2)} = \dfrac{37{,}5771}{78{,}4587} = 18{,}2276$

**Ejercicio 10.7** Dados los puntos $A = (1, 2)$, $B = (3, 4)$, $C = (1, 6)$, $D = (5, 2)$, $E = (7, 2)$, $F = (6, 3)$, $G = (8, 7)$ y $H = (10, 6)$, se pide realizar un agrupamiento en 3 grupos aplicando el algoritmo k-means con la distancia de Manhattan, usando los siguientes puntos de inicio: $\mu_1 = A$, $\mu_2 = B$ y $\mu_3 = D$.

**Solución**

Primero se calcula la distancia de cada punto del conjunto de datos a los centroides (en este caso son los puntos de inicio), y posteriormente cada punto se asigna al grupo con el centroide más cercano.

| Puntos | Dist. a $\mu_1$ | Dist. a $\mu_2$ | Dist. a $\mu_3$ | Grupo |
|---|---|---|---|---|
| A(1,2) | 0 | 4 | 4 | 1 |
| B(3,4) | 4 | 0 | 4 | 2 |
| C(1,6) | 4 | 4 | 8 | 1 |
| D(5,2) | 4 | 4 | 0 | 3 |
| E(7,2) | 6 | 6 | 2 | 3 |
| F(6,3) | 6 | 4 | 2 | 3 |
| G(8,7) | 12 | 8 | 8 | 2 |
| H(10,6) | 13 | 9 | 9 | 2 |

C y G tienen dos posibilidades de asignación a grupo. En este caso hemos seleccionado el $\mu_i$ con el menor $i$.

Ahora calculamos los nuevos centroides:

$\mu_1 = \frac{1}{2}(A + C) = \frac{1}{2}((1, 2) + (1, 6)) = \frac{1}{2}(2, 8) = (1, 4)$

$\mu_2 = \frac{1}{3}(B + G + H) = \frac{1}{3}((3, 4) + (8, 7) + (10, 6)) = \frac{1}{3}(21, 17) = (7, \frac{17}{3})$

$\mu_3 = \frac{1}{3}(D + E + F) = \frac{1}{3}((5,2) + (7,2) + (6,3)) = \frac{1}{3}(18,7) = (6, \frac{7}{3})$

Ahora se vuelve a calcularla distancia de cada punto del conjunto de datos a los nuevos centroides, y posteriormente cada punto se asigna al grupo con el centroide más cercano.

| Puntos | Dist. a $\mu_1$ | Dist. a $\mu_2$ | Dist. a $\mu_3$ | Grupo |
|---|---|---|---|---|
| A(1,2) | 2 | 9.66 | 5.33 | 1 |
| B(3,4) | 2 | 5.66 | 4.66 | 1 |
| C(1,6) | 2 | 6.33 | 8.66 | 1 |
| D(5,2) | 6 | 5.66 | 1.33 | 3 |
| E(7,2) | 8 | 3.66 | 1.33 | 3 |
| F(6,3) | 6 | 3.66 | 0.66 | 3 |
| G(8,7) | 10 | 2.33 | 6.66 | 2 |
| H(10,6) | 11 | 3.33 | 7.66 | 2 |

Calculamos los nuevos centroides:

$\mu_1 = \frac{1}{3}(A + B + C) = \frac{1}{3}((1,2) + (3,4) + (1,6)) = \frac{1}{3}(5,12) = (\frac{5}{3}, 4)$

$\mu_2 = \frac{1}{2}(G + H) = \frac{1}{2}((8,7) + (10,6)) = \frac{1}{2}(18,13) = (9, \frac{13}{2})$

$\mu_3 = \frac{1}{3}(D + E + F) = \frac{1}{3}((5,2) + (7,2) + (6,3)) = \frac{1}{3}(18,7) = (6\frac{7}{3})$

Ejecutamos una nueva iteración del algoritmo:

| Puntos | Dist. a $\mu_1$ | Dist. a $\mu_2$ | Dist. a $\mu_3$ | Grupo |
|---|---|---|---|---|
| A(1,2) | 2.66 | 12.5 | 5.33 | 1 |
| B(3,4) | 1.33 | 8.5 | 4.66 | 1 |
| C(1,6) | 2.66 | 8.5 | 8.66 | 1 |
| D(5,2) | 5.33 | 8.5 | 1.33 | 3 |
| E(7,2) | 7.33 | 6.5 | 1.33 | 3 |
| F(6,3) | 5.33 | 6.5 | 0.66 | 3 |
| G(8,7) | 9.33 | 1.5 | 6.66 | 2 |
| H(10,6) | 10.33 | 1.5 | 7.66 | 2 |

La asignación a grupos es la misma que en la iteración anterior, por lo que el algoritmo para su ejecución.

**Ejercicio 10.8** Dados los puntos $x_1 = (2, 10)$, $x_2 = (2, 5)$, $x_3 = (8, 4)$, $x_4 = (5, 8)$, $x_5 = (7, 5)$, $x_6 = (6, 4)$, $x_7 = (1, 2)$ y $x_8 = (4, 9)$, se pide realizar un agrupamiento en 3 grupos aplicando el algoritmo k-means con la distancia de Manhattan, usando los siguientes puntos de inicio: $x_1$, $x_4$ y $x_7$.

**Solución**

Se recuerda que la distancia de Manhattan entre los puntos $x$ e $y$ se calcula de la siguiente manera:

$d(x, y) = |x_1 - y_1| + |x_2 - y_2|$

Se considera que $\mu_1 = x_1$, $\mu_2 = x_4$ y $\mu_3 = x_7$.

Primero se calcula la distancia de cada punto del conjunto de datos a los centroides (en este caso son los puntos de inicio), y posteriormente cada punto se asigna al grupo con el centroide más cercano.

| Puntos | Dist. a $\mu_1$ | Dist. a $\mu_2$ | Dist. a $\mu_3$ | Grupo |
|---|---|---|---|---|
| $x_1 = (2, 10)$ | 0 | 5 | 9 | 1 |
| $x_2 = (2, 5)$ | 5 | 6 | 4 | 3 |
| $x_3 = (8, 4)$ | 12 | 7 | 9 | 2 |
| $x_4 = (5, 8)$ | 5 | 0 | 10 | 2 |
| $x_5 = (7, 5)$ | 10 | 5 | 9 | 2 |
| $x_6 = (6, 4)$ | 10 | 5 | 7 | 2 |
| $x_7 = (1, 2)$ | 9 | 10 | 0 | 3 |
| $x_8 = (4, 9)$ | 3 | 2 | 10 | 2 |

Ahora calculamos los nuevos centroides:

$\mu_1 = x_1 = (2, 10)$

$\mu_2 = \frac{1}{5}(x_3+x_4+x_5+x_6+x_8) = \frac{1}{5}((8, 4)+(5, 8)+(7, 5)+(6, 4)+(4, 9)) = \frac{1}{5}(30, 30) = (6, 6)$

$\mu_3 = \frac{1}{2}(x_2 + x_8) = \frac{1}{2}((2, 5) + (1, 2)) = \frac{1}{2}(3, 7) = (\frac{3}{2}, \frac{7}{2})$

Una vez hecho esto se vuelve a calcularla distancia de cada punto del conjunto de datos a los nuevos centroides, y posteriormente cada punto se asigna al grupo con el centroide más cercano.

| Puntos | Dist. a $\mu_1$ | Dist. a $\mu_2$ | Dist. a $\mu_3$ | Grupo |
|---|---|---|---|---|
| $x_1 = (2, 10)$ | 0 | 8 | 7 | 1 |
| $x_2 = (2, 5)$ | 5 | 5 | 2 | 3 |
| $x_3 = (8, 4)$ | 12 | 4 | 7 | 2 |
| $x_4 = (5, 8)$ | 5 | 3 | 8 | 2 |
| $x_5 = (7, 5)$ | 10 | 2 | 7 | 2 |
| $x_6 = (6, 4)$ | 10 | 2 | 5 | 2 |
| $x_7 = (1, 2)$ | 9 | 9 | 2 | 3 |
| $x_8 = (4, 9)$ | 3 | 5 | 8 | 1 |

Ahora se tendría que realizar una nueva iteración del algoritmo, calculando nuevamente la distancia de cada punto del conjunto de datos a los nuevos centroides y asignando posteriormente cada punto al grupo con el centroide más cercano. El algoritmo finaliza cuando la asignación a grupos de una iteración es la misma que la de la iteración anterior.

# Tema 11

# Modelos no paramétricos

**Ejercicio 11.1** El clima mundial se estudia mediante imágenes obtenidas por satélites. Supongamos que un satélite toma una imagen de tamaño 180x360 píxeles cada día, y que cada píxel tiene cuatro componentes que son números reales en el intervalo [0,1]: el primero es la proporción de nubes, el segundo es la concentración de monóxido de carbono, el tercero es la concentración de vapor de agua, y el último es la temperatura en superficie.

Por otro lado, imaginemos que algunos de los sensores de los píxeles están rotos, de manera que hay algunos píxeles que no tienen datos válidos. ¿Cómo estimarías estos valores ausentes?

**Solución**

Podemos usar regresión no paramétrica para este problema. En general, dado un punto de consulta $\mathbf{x}_q$ en un espacio de entrada $\mathbf{R}^n$, la tarea es estimar $y_q = h(\mathbf{x}_q)$ a partir de un conjunto de ejemplos $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ de tamaño $N$. En nuestro caso:

- Cada punto de consulta es de la forma $\mathbf{x}_q = (i, j)$, es decir, las coordenadas de un píxel con valores ausentes. Nótese que $i \in 1, 2, ..., 180$, y $j \in 1, 2, ..., 360$.
- La dimensión del espacio de entrada es $n = 2$, ya que un píxel tiene dos coordenadas.

- Hay cuatro funciones que estimar $h1$, $h2$, $h3$ y $h4$, que corresponden a la proporción de nubes, la concentración de monóxido de carbono, la concentración de vapor de agua, y la temperatura en superficie, respectivamente. Nótese que cada función se estima por separado.
- El tamaño del conjunto de ejemplos es $N - 180 \cdot 360 - NumBroken$, donde $NumBroken$ es el número de píxeles defectuosos en la imagen de $180 \times 360$ píxeles.
- Para cada función que estimar $h1$, $h2$, $h3$ y $h4$, el conjunto de ejemplos está formado por ternas $(i_k, j_k, v_k)$, donde $(i_k, j_k)$ son las coordenadas del píxel válido $k$-ésimo, es decir, no defectuoso; y $v_k \in [0, 1]$ es el valor del $k$-ésimo píxel válido para la componente que estemos considerando (proporción de nubes, concentración de monóxido de carbono, concentración de vapor de agua o temperatura en superficie, respectivamente).

**Ejercicio 11.2** Consideremos el conjunto de datos de flores Iris, también llamado conjunto de datos Iris de Fisher, presentado por Sir Ronald Fisher (1936). Dicho conjunto consiste en 50 muestras de cada una de las tres especies de Iris (Iris setosa, Iris versicolor and Iris virginica). De cada muestra se obtuvieron cuatro rasgos: la longitud y la anchura de los sépalos y los pétalos, en centímetros.

Debemos diseñar un sistema que, dada una muestra no vista anterioremente con sus cuatro rasgos, la clasifique en una de las tres especies.

**Solución**

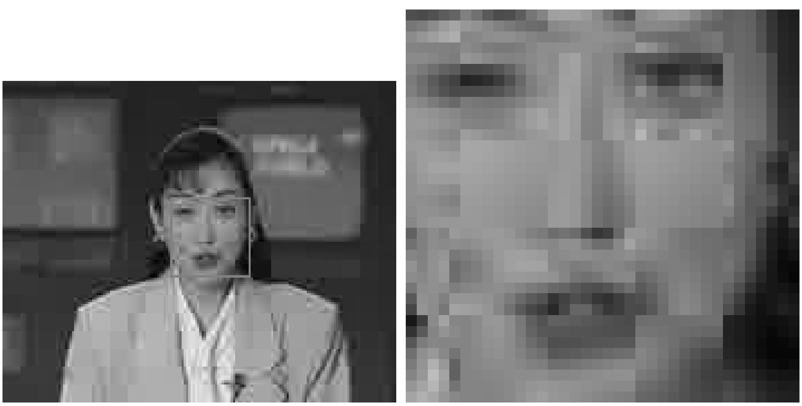
Se puede emplear un modelo de vecinos mas próximos para este problema de clasificación. En general, dado un punto de consulta $\mathbf{x}_q$ en un espacio de entrada $\mathbf{R}^n$, la tarea es estimar $y_q = h(\mathbf{x}_q)$ a partir de un conjunto de ejemplos $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ de tamaño $N$. En nuestro caso:

- Cada punto de consulta es de la forma $\mathbf{x}_q = (LengthSepals, WidthSepals, LengthPetals, WidthPetals)$, o sea, las características de una flor no vista anteriormente. Nótese que $\mathbf{x}_q \in \mathbf{R}^4$.
- El rango de la función $h$ a estimar es $\{Setosa, Versicolor, Virginica\}$, esto es, $h(\mathbf{x}_q) \in \{Setosa, Versicolor, Virginica\}$.
- La dimensión del espacio de entrada es $n = 4$, dado que hay cuatro rasgos.

- El tamaño del conjunto de ejemplos (también llamado conjunto de entrenamiento) es $N = 3 \cdot 50 = 150$, puesto que disponemos de 50 muestras de cada clase.
- El conjunto de ejemplos está formado por tuplas $(LengthSepals_k, WidthSepals_k, LengthPetals_k, WidthPetals_k, Class_k)$, donde $k \in \{1, 2, \ldots, 150\}$ es el índice de la flor de muestra. Nótese que $Class_k \in \{Setosa, Versicolor, Virginica\}$ es la clase a la que pertenece la $k$-ésima flor de muestra.
- A fin de calcular $y_q = h(\mathbf{x}_q)$ para una flor no vista anteriormente, realizamos una elección por mayoría simple entre los $k$ vecinos más próximos, es decir, la clase de salida es el valor de $h(\mathbf{x})$ que aparece más veces en $NN(k, \mathbf{x}_q)$.
- Debemos elegir el tamaño del entorno $k$ que maximice el número de aciertos de clasificación. Se puede usar otro conjunto de ejemplos para este propósito, que no tenga ninguna muestra en común con el conjunto de entrenamiento. Dicho conjunto se llama conjunto de validación.

**Ejercicio 11.3** La restauración de imágenes y vídeo trata con datos visuales que han sido corrompidos por algún proceso (véanse las figuras de abajo). Por ejemplo, podríamos tener datos de vídeo de mala calidad, donde la corrupción consiste en defectos de compresión introducidos por el algoritmo de compresión de vídeo. Por otra parte, el ruido gaussiano introducido por las imperfecciones de los sensores de las cámaras fotográficas produce imágenes ruidosas (granuladas).

Dada una imagen de entrada o un fotograma de vídeo que necesitan ser restaurados, ¿cómo lo harías?



Fotograma de un vídeo de mala calidad (izquierda) y detalle (derecha).

Fotografía sin ruido (izquierda) y fotografía con ruido (derecha).

**Solución**

Podemos usar regresión no paramétrica para este problema. En general, dado un punto de consulta $\mathbf{x}_q$ en un espacio de entrada $\mathbf{R}^n$, la tarea es estimar $y_q = h(\mathbf{x}_q)$ a partir de un conjunto de ejemplos $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ de tamaño $N$. En nuestro caso:

- Cada punto de consulta es de la forma $\mathbf{x}_q = (i, j)$, es decir, las coordenadas de un píxel de una foto o fotograma corrompido. Nótese que $i \in \{1, 2, ..., Q\}$, y $j \in \{1, 2, ..., R\}$, donde el tamaño de la imagen corrompida es $Q \times R$.
- La dimensión del espacio de entrada es $n = 2$, puesto que cada píxel tiene dos coordenadas.
- Hay tres funciones que estimar $h1$, $h2$ y $h3$, que corresponden a los canales rojo, verde y azul de la imagen, respectivamente. Nótese que cada función se estima por separado.
- El tamaño del conjunto de ejemplos es $N = Q \cdot R$.
- Para cada función que estimar $h1$, $h2$ y $h3$, el conjunto de ejemplos está formado por ternas $(i_k, j_k, v_k)$, donde $(i_k, j_k)$ son las coordenadas del $k$-ésimo píxel; y $v_k \in [0, 1]$ es el valor del $k$-ésimo píxel para el canal de color que estamos considerando (rojo, verde o azul, respectivamente).

- La anchura del núcleo $k$ controla la estimación. Si $k$ es alto, entonces la imagen de salida será muy suave, pero los detalles pequeños se perderán. Por otro lado, si $k$ es bajo, entonces la mayor parte de los defectos seguirán siendo visibles, pero no se perderán los detalles pequeños de la imagen.

**Ejercicio 11.4** Dada una imagen y un ángulo de rotación $\theta$ en radianes, $\theta \in [0, 2\pi]$, ¿cómo obtendrías la imagen rotada?

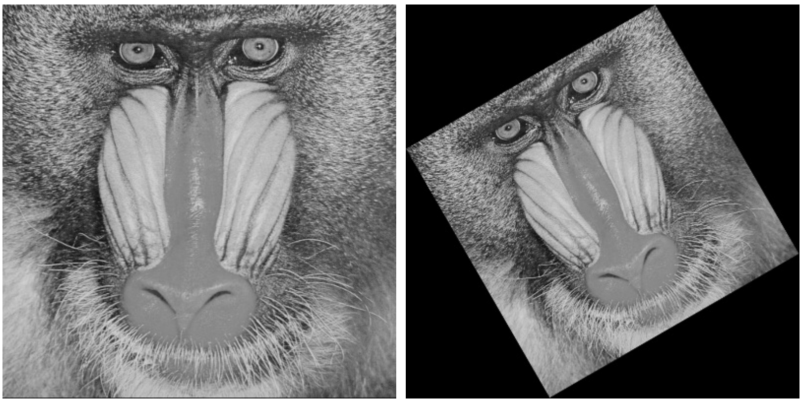


Imagen original (izquierda) e imagen rotada para $\theta = \pi/6$ radianes (derecha).

**Solución**

Podemos usar regresión no paramétrica para este problema. En general, dado un punto de consulta $\mathbf{x}_q$ en un espacio de entrada $\mathbf{R}^n$, la tarea es estimar $y_q = h(\mathbf{x}_q)$ a partir de un conjunto de ejemplos $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ de tamaño $N$. En nuestro caso:

- Cada punto de ejemplo es de la forma $\mathbf{x}_k - (i, j)$, esto es, las coordenadas de un píxel de la imagen de entrada. Para que las expresiones matemáticas sean más sencillas, consideraremos un sistema de coordenadas para los píxeles donde el origen de coordenadas $(0, 0)$ es el centro de la imagen de entrada. Para una imagen de entrada de tamaño $(Q + 1) \times (R + 1)$, esto implica que $i \in [-Q/2, Q/2]$, y $j \in [-R/2, R/2]$. Nótese que el centro de la imagen rotada es el mismo que el de la imagen de entrada, es decir, sus coordenadas de píxel son también $(0, 0)$.
- La dimensión del espacio de entrada es $n = 2$, dado que cada píxel tiene dos coordenadas.
- Hay tres funciones que estimar $h1$, $h2$ y $h3$, que corresponden a los canales rojo, verde y azul de la imagen, respectivamente. Nótese que cada función se estima por separado.
- El tamaño del conjunto de ejemplos es $N = (Q + 1) \cdot (R + 1)$.
- Para cada función que estimar $h1$, $h2$ y $h3$, el conjunto de ejemplos está formado por ternas $(i_k, j_k, v_k)$, donde $(i_k, j_k)$ son las coordenadas del $k$-ésimo píxel de la imagen de entrada; y $v_k \in [0, 1]$ es el valor del $k$-ésimo píxel de la imagen de entrada para el canal de color que estemos considerando (rojo, verde o azul, respectivamente).
- Los puntos de consulta $\mathbf{x}_q$ deben elegirse de tal manera que se obtengan los píxeles de la imagen rotada. El punto de consulta que debe usarse para el píxel $(i', j')$ de la imagen rotada es:

  $$\mathbf{x}_q = \begin{pmatrix} cos(-\theta) & -sen(-\theta) \\ sen(-\theta) & cos(-\theta) \end{pmatrix} \begin{pmatrix} i' \\ j' \end{pmatrix}$$

que corresponde a una rotación de ángulo $-\theta$ alrededor del origen de coordenadas. Nótese que es $-\theta$ y no $\theta$ , porque necesitamos rotar los píxeles de salida un ángulo $-\theta$ para expresarlos en el sistema de coordenadas de la imagen de entrada. Si un punto de consulta se sale del rango válido, es decir $\mathbf{x}_q \notin [-Q/2, Q/2] \times [-R/2, R/2]$, entonces ponemos la salida a cero (negro), como se ve en la figura que ilustra el enunciado del ejercicio. Dejamos como ejercicio que halles el tamaño de la imagen rotada, teniendo en cuenta que las esquinas de la imagen de entrada deben estar sobre el borde de la imagen rotada, como se muestra en la imagen mencionada.

**Ejercicio 11.5** Hemos recogido datos sobre incendios en una región. Tenemos pares $(d_i, a_i)$, donde $d_i \in [0, 2\pi]$ es la dirección media del viento en radianes y $a_i$ es el área quemada diaria para el i-ésimo día registrado. Queremos diseñar un sistema que, dada la previsión de la dirección del viento, de como salida una estimación del área quemada. Nótese que $d_i$ es una variable direccional, con lo que $d_i = 0$ tiene el mismo significado que $d_i = 2\pi$.

**Solución**

Podemos usar regresión no paramétrica para este problema. En general, dado un punto de consulta $\mathbf{x}_q$ en un espacio de entrada $\mathbf{R}^n$, la tarea es estimar $y_q = h(\mathbf{x}_q)$ a partir de un conjunto de ejemplos $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ de tamaño $N$. En nuestro caso:

- Cada punto de consulta es de la forma $\mathbf{x}_q = (d_q)$, esto es, una dirección del viento. Nótese que $d_q \in [0, 2\pi]$. Téngase también en cuenta que dq puede coincidir o no con alguna de las direcciones del viento registradas.
- La dimensión del espacio de entrada es $n = 1$, ya que la dirección del viento viene dada por un único número real.
- La función que hay que estimar es el área quemada diaria, $y_q = a_q = h(\mathbf{x}_q) = h(d_q)$.
- El tamaño del conjunto de ejemplos $N$ es el número de días registrados.

- El conjunto de ejemplo está formado por los pares $(d_i, a_i)$.
- Tenemos que usar una definición de distancia para la regresión que tenga en cuenta que los ángulos dan la vuelta en $d_i = 2\pi$:

  $Distance(\mathbf{x}_j, \mathbf{x}_q) = Distance(d_j, d_q) = min(|d_j - d_q|, 2\pi - |d_j - d_q|)$

**Ejercicio 11.6** Consideremos la distancia de Levenshtein entre dos cadenas, que se define como el mínimo número de modificaciones necesarias para transformar una cadena en la otra, siendo las modificaciones admitidas la inserción, eliminación o sustitución de un solo carácter.

Por ejemplo, la distancia de Levenshtein entre 'kitten' y 'sitting' es 3, dado que las siguientes tres modificaciones convierten una en la otra, y no hay manera de conseguirlo con menos de tres modificaciones:

a) kitten $\Rightarrow$ sitten (sustitución de 's' por 'k')

b) sitten $\Rightarrow$ sittin (sustitución de 'i' por 'e')

c) sittin $\Rightarrow$ sitting (inserción de 'g' al final).

Hemos obtenido un gran número de cadenas de texto que corresponden a correos electrónicos no deseados, y otras cadenas de texto que corresponden a correos electrónicos legítimos. Queremos usar este conjunto de datos para diseñar un detector de correo no deseado basado en la distancia de Levenshtein.

**Solución**

Se puede emplear un modelo de vecinos más próximos para este problema de clasificación. En general, dado un punto de consulta $\mathbf{x}_q$ en un espacio de entrada, la tarea es estimar $y_q = h(\mathbf{x}_q)$ a partir de un conjunto de ejemplos $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ de tamaño $N$. En nuestro caso:

- Cada punto de consulta es una cadena de texto $\mathbf{x}_q \in \Sigma^*$ de un correo electrónico que queremos clasificar, donde $\Sigma^*$ es el conjunto de todas las cadenas sobre un alfabeto $\Sigma$.
- El rango de la función $h$ que debe estimarse es $\{Spam, Legitimate\}$, o sea, $h(\mathbf{x}_q) \in \{Spam, Legitimate\}$.

- No hay dimensión del espacio de entrada, ya que $\Sigma^*$ carece de dimensionalidad.
- El tamaño del conjunto de ejemplos $N$ es el número de textos de muestra que hemos recogido.
- El conjunto de ejemplos está formado por tuplas $(\mathbf{x}_i, y_i)$, donde $\mathbf{x}_i$ es la $i$-ésima cadena de texto recogida e $y_i$ es la clase a la que pertenece.
- A fin de calcular $y_q = h(\mathbf{x}_q)$ para una cadena de texto no vista anteriormente, realizamos una elección por mayoría simple entre los $k$ vecinos más próximos, es decir, la clase de salida es el valor de $h(\mathbf{x})$ que aparece más veces en $NN(k, \mathbf{x}_q)$. El conjunto de los vecinos más próximos $NN(k, \mathbf{x}_q)$ se calcula con respecto a la distancia de Levenshtein.
- Debemos elegir el tamaño del entorno $k$ que maximice el número de aciertos de clasificación. Se puede usar otro conjunto de ejemplos para este propósito, que no tenga ninguna muestra en común con el conjunto de entrenamiento. Dicho conjunto se llama conjunto de validación.

**Ejercicio 11.7** Estamos siguiendo la trayectoria de un coche desde dos helicópteros, cuyos dispositivos de medición cometen errores. Tenemos $N_1$ datos de seguimiento $(a_i, b_i, c_i)$ del primer helicóptero, donde $a_i$ es la posición medida en la dirección Norte-Sur, $b_i$ es la posición medida en la dirección Este-Oeste, y $c_i$ es el instante de tiempo de la medición. Por otro lado, tenemos $N_2$ datos $(d_j, e_j, f_j)$ del segundo helicóptero, donde $d_j$ es la posición medida en la dirección Norte-Sur, $e_j$ es la posición medida en la dirección Este-Oeste, y $f_j$ es el instante de tiempo de la medición. ¿Cómo estimarías la posición del coche en cualquier instante de tiempo mediante un modelo no paramétrico?

**Solución**

Podemos usar regresión no paramétrica para este problema. En general, dado un punto de consulta $\mathbf{x}_q$ en un espacio de entrada $\mathbf{R}^n$, la tarea es estimar $y_q = h(\mathbf{x}_q)$a partir de un conjunto de ejemplos $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_M, y_N)\}$ de tamaño $N$. En nuestro caso:

- Cada punto de consulta es de la forma $\mathbf{x}_q = t$, es decir, un instante de tiempo que podría o no estar entre los correspondientes a las mediciones de los helicópteros.
- La dimensión del espacio de entrada es $n = 1$, ya que el instante de tiempo viene dado por un solo número real.
- Tenemos dos funciones que estimar $h1$ y $h2$, que corresponden a la posición en la dirección Norte-Sur y la posición en la dirección Este-Oeste, respectivamente. Nótese que cada función se estima por separado.
- El tamaño del conjunto de ejemplos es $N = N1 + N2$, esto es, los conjuntos de datos de ambos helicópteros son fusionados.
- Para la primera función que estimar $h1$, el conjunto de ejemplos es el siguiente:

  $\{(c_i, a_i)|i \in \{1, ..., N_1\}\} \cup \{(f_j, d_j)|j \in \{1, ..., N_2\}\}$
- Para la segundo función que estimar $h2$, el conjunto de ejemplos es el siguiente:

  $\{(c_i, b_i)|i \in \{1, ..., N_1\}\} \cup \{(f_j, e_j)|j \in \{1, ..., N_2\}\}$

**Ejercicio 11.8** Dado el siguiente conjunto de datos de entrenamiento, se pide usar el clasificador del vecino más cercano con número de vecinos más cercanos $k = 3$ y la medida de la distancia de Hamming para el test de los patrones correspondientes a las siguientes personas: una persona que tiene tos, una persona que tiene la nariz congestionada que tiene tos y fiebre, y una persona que tiene fiebre.

| Muestra | $x_1$ (nariz congestionada) | $x_2$ (tos) | $x_3$ (piel enrojecida) | $x_4$ (fiebre) | Clase |
|---|---|---|---|---|---|
| 1 | Sí | Sí | Sí | No | + (enfermo) |
| 2 | Sí | Sí | No | No | + (enfermo) |
| 3 | No | No | Sí | Sí | + (enfermo) |
| 4 | Sí | No | No | No | - (sano) |
| 5 | No | No | No | No | - (sano) |
| 6 | No | Sí | Sí | No | - (sano) |

**Solución**

Las tres muestras de test a ser clasificadas son las siguientes:

Muestra de test 1 = (No, Sí, No, No)

Muestra de test 2 = (Sí, Sí, No, Sí)

Muestra de test 3 = (No, No, No, Sí)

Ahora se calculan las distancias de Hamming de los datos de entrenamiento a los datos de test. Se recuerda que la distancia de Hamming se define como el número de componentes que difieren:

| Muestra | Distancia 1 | Distancia 2 | Distancia 3 |
|---|---|---|---|
| 1 | 2 | 2 | 4 |
| 2 | 1 | 1 | 3 |
| 3 | 3 | 3 | 1 |
| 4 | 2 | 2 | 2 |
| 5 | 1 | 3 | 1 |
| 6 | 1 | 3 | 3 |
| 7 | 2 | 4 | 2 |

Por tanto, si se quiere calcular la distancia de la muestra 1 a la muestra de test 1 se hace:

Hamming(((Sí, No, Sí, No), (Sí, Sí, No, No)) = 0 + 1 + 1 + 0 = 2

Los $k$ vecinos más cercanos a la muestra de test 1 son las muestras de entrenamiento 2, 5 y 6; cuyas etiquetas son positivo, negativo, y negativo, respectivamente. Por tanto, la muestra de test 1 es clasificada como negativa.

Los $k$ vecinos más cercanos a la muestra de test 2 son las muestras de entrenamiento 1, 2 y 4; cuyas etiquetas son positivo, positivo, y negativo, respectivamente. Por tanto, la muestra de test 2 es clasificada como positiva.

Los $k$ vecinos más cercanos a la muestra de test 3 son las muestras de entrenamiento 3, 4, 5 y 7 (hay un empate entre las muestras de entrenamiento 4 y 7); cuyas etiquetas son negativo, positivo, positivo, y positivo, respectivamente. Por tanto, la muestra de test 3 es clasificada como positiva.

**Ejercicio 11.9** Dado el siguiente conjunto de datos de entrenamiento, se pide usar el clasificador del vecino más cercano con número de vecinos más cercanos $k = 3$ y la medida de la distancia de Hamming para el test de los patrones correspondientes a las siguientes personas: una persona que tiene fiebre alta, y una persona que sufre vómitos y temblores.

| Muestra | $x_1$ (fiebre) | $x_2$ (vómitos) | $x_3$ (diarrea) | $x_4$ (temblores) | Clase |
|---|---|---|---|---|---|
| 1 | no | no | no | no | sano |
| 2 | media | no | no | no | gripe |
| 3 | alta | no | no | sí | gripe |
| 4 | alta | sí | sí | no | salmonela |
| 5 | media | no | sí | no | salmonela |
| 6 | no | sí | sí | no | inflamación intestinal |
| 7 | media | sí | sí | no | inflamación intestinal |

**Solución**

Las dos muestras de test a ser clasificadas son las siguientes:

Muestra de test 1 = (Alta, No, No, No)

Muestra de test 2 = (No, Sí, No, Sí)

Ahora se calculan las distancias de Hamming de los datos de entrenamiento a los datos de test. Se recuerda que la distancia de Hamming se define como el número de componentes que difieren:

| Muestra | Distancia a test 1 | Distancia a test 2 |
|---|---|---|
| 1 | 1 | 2 |
| 2 | 1 | 3 |
| 3 | 1 | 2 |
| 4 | 2 | 3 |
| 5 | 2 | 4 |
| 6 | 3 | 2 |
| 7 | 3 | 3 |

Los $k$ vecinos más cercanos a la muestra de test 1 son las muestras de entrenamiento 1, 2 y 3; cuyas etiquetas son sano, gripe, y gripe, respectivamente. Por tanto, la muestra de test 1 es clasificada como gripe.

Los $k$ vecinos más cercanos a la muestra de test 2 son las muestras de entrenamiento 1, 3 y 6; cuyas etiquetas son sano, gripe, e inflamación del intestino, respectivamente. Por tanto, hay un triple empate entre las tres clases, por lo que la muestra de test 2 no puede ser clasificada concluyentemente.

**Ejercicio 11.10** Dado el siguiente conjunto de datos de entrenamiento, se pide usar el clasificador del vecino más cercano con número de vecinos más cercanos $k = 5$ y la medida de la distancia de Hamming para el test de los patrones correspondientes a un estudiante con ingreso medio y calificación crediticia razonable.

| Muestra | $x_1$ (ingreso) | $x_2$ (estudiante) | $x_3$ (calificación crediticia) | Clase (compra un ordenador) |
|---|---|---|---|---|
| 1 | Alto | No | Razonable | No |
| 2 | Alto | No | Excelente | No |
| 3 | Alto | No | Razonable | Sí |
| 4 | Medio | No | Razonable | Sí |
| 5 | Bajo | Sí | Razonable | Sí |
| 6 | Bajo | Sí | Mala | No |
| 7 | Bajo | Sí | Excelente | Sí |
| 8 | Medio | No | Mala | No |
| 9 | Bajo | Sí | Razonable | Sí |
| 10 | Medio | Sí | Mala | Sí |
| 11 | Medio | Sí | Excelente | Sí |
| 12 | Medio | No | Excelente | Sí |
| 13 | Alto | Sí | Razonable | Sí |
| 14 | Medio | No | Excelente | No |

**Solución**

La muestra de test a ser clasificada es la siguiente:

Muestra de test 1 = (Medio, Sí, Razonable)

Ahora se calculan las distancias de Hamming de los datos de entrenamiento a los datos de test. Se recuerda que la distancia de Hamming se define como el número de componentes que difieren:

| Muestra | Distancia a la muestra de test 1 |
|---|---|
| 1 | 2 |
| 2 | 3 |
| 3 | 2 |
| 4 | 1 |
| 5 | 1 |
| 6 | 2 |
| 7 | 2 |
| 8 | 2 |
| 9 | 1 |
| 10 | 1 |
| 11 | 1 |
| 12 | 2 |
| 13 | 2 |
| 14 | 2 |

Por ejemplo, la distancia entre la muestra de test y la muestra de entrenamiento número 1 es la siguiente: el ingreso es diferente, el estudiante es diferente y la calificación crediticia es la misma, por tanto:

Hamming(Muestra de test, Muestra de entrenamiento 1) = 1 + 1 + 0 = 2

En el caso de la muestra de entrenamiento 2 y la muestra de test, todos los valores de los atributos son diferentes, por tanto:

Hamming(Muestra de test, Muestra de entrenamiento 2) = 1 + 1 + 1 = 3

Los $k$ vecinos más cercanos a la muestra de test son las muestras de entrenamiento 4, 5, 9, 10 y 11; cuyas etiquetas son Sí, Sí, Sí, Sí, y Sí, respectivamente. Por tanto, la muestra de test es clasificada como Sí (compra un ordenador).

**Ejercicio 11.11** Dado el siguiente conjunto de datos de entrenamiento, se pide usar el clasificador del vecino más cercano con número de vecinos más cercanos $k = 2$ y la medida de la distancia de Manhattan para el test de los patrones (215,45,12), (170,162,43).

| Muestra | $x_1$ (rojo) | $x_2$ (verde) | $x_3$ (azul) | Clase |
|---|---|---|---|---|
| 1 | 241 | 45 | 24 | Manzana |
| 2 | 22 | 180 | 16 | Hoja |
| 3 | 192 | 171 | 8 | Tronco |
| 4 | 201 | 192 | 23 | Tronco |
| 5 | 199 | 203 | 42 | Tronco |
| 6 | 41 | 241 | 32 | Hoja |
| 7 | 239 | 31 | 22 | Manzana |
| 8 | 250 | 41 | 27 | Manzana |

**Solución**

La distancia de Manhattan a las muestras de test son las siguientes:

| Muestra | Dist. a la muestra de test 1 | Dist. a la muestra de test 2 |
|---|---|---|
| 1 | 38 | 207 |
| 2 | 332 | 193 |
| 3 | 153 | 66 |
| 4 | 172 | 81 |
| 5 | 204 | 71 |
| 6 | 390 | 219 |
| 7 | 48 | 221 |
| 8 | 54 | 217 |

Los $k$ vecinos más cercanos a la muestra de test 1 son las muestras de entrenamiento 1 y 7; cuyas etiquetas son Manzana, y Manzana, respectivamente. Por tanto, la muestra de test 1 es clasificada como Manzana.

Los $k$ vecinos más cercanos a la muestra de test 2 son las muestras de entrenamiento 3 y 5; cuyas etiquetas son Tronco, y Tronco, respectivamente. Por tanto, la muestra de test 2 es clasificada como Tronco.

**Ejercicio 11.12** Dado el siguiente conjunto de datos de entrenamiento, se pide usar el clasificador del vecino más cercano con número de vecinos más cercanos $k = 2$ y la medida de la distancia euclídea para el test de los patrones (215,45,12), (170,162,43).

| Muestra | $x_1$ (rojo) | $x_2$ (verde) | $x_3$ (azul) | Clase |
|---|---|---|---|---|
| 1 | 241 | 45 | 24 | Manzana |
| 2 | 22 | 180 | 16 | Hoja |
| 3 | 192 | 171 | 8 | Tronco |
| 4 | 201 | 192 | 23 | Tronco |
| 5 | 199 | 203 | 42 | Tronco |
| 6 | 41 | 241 | 32 | Hoja |
| 7 | 239 | 31 | 22 | Manzana |
| 8 | 250 | 41 | 27 | Manzana |

**Solución**

La distancia euclídea a las muestras de test son las siguientes:

| Muestra | Dist. a la muestra de test 1 | Dist. a la muestra de test 2 |
|---|---|---|
| 1 | 820 | 19091 |
| 2 | 55490 | 22957 |
| 3 | 16421 | 1790 |
| 4 | 21926 | 2261 |
| 5 | 26120 | 2523 |
| 6 | 69092 | 23003 |
| 7 | 872 | 22363 |
| 8 | 1466 | 21297 |

Los $k$ vecinos más cercanos a la muestra de test 1 son las muestras de entrenamiento 1 y 7; cuyas etiquetas son Manzana, y Manzana, respectivamente. Por tanto, la muestra de test 1 es clasificada como Manzana.

Los $k$ vecinos más cercanos a la muestra de test 2 son las muestras de entrenamiento 3 y 4; cuyas etiquetas son Tronco, y Tronco, respectivamente. Por tanto, la muestra de test 2 es clasificada como Tronco.

# Tema 12

# Problemas de Decisión

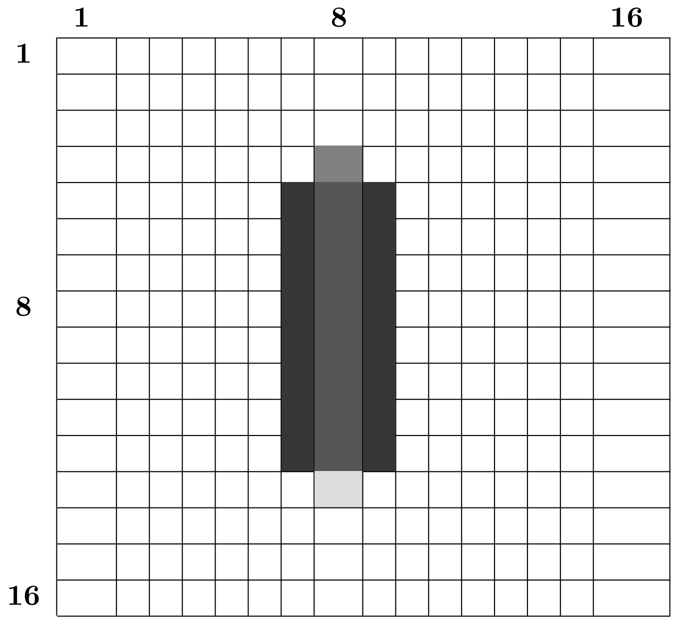
**Ejercicio 12.1** El profesor Van Helsing quiere llegar al ataúd del conde Drácula para matarlo. Debe hacerlo lo antes posible, ya que Drácula está a punto de despertarse. La posición del ataúd de Drácula se muestra en rojo, mientras que el punto de partida de Van Helsing se muestra en verde. Las acciones son *Up*, *Down*, *Left* y *Right*.

Los cuadrados negros son muros, que no se pueden atravesar. Hay un corredor (dibujado en azul) que lleva directamente al ataúd. Van Helsing necesita 3 minutos para cruzar cada cuadrado del corredor, pero si intenta ir hacia arriba dentro de él hay una probabilidad del 15 % de que tenga que retirarse en dirección contraria debido a los murciélagos. Por otra parte, necesita 2 minutos para cruzar un cuadrado normal (en blanco), y es seguro que su acción logrará el resultado pretendido. Se pide:

a) Definir el entorno de este problema de decisión.

b) Definir el modelo de transición y las recompensas de estado.

c) Definir dos posibles políticas para este problema: una que lo guíe a través del corredor, y otra que lo evite.




**Solución**

a) Definir el entorno de este problema de decisión.

**Entorno**:

Hay 16·16=256 estados, que podemos identificar mediante pares (i,j), donde i es el índice de la fila y j es el índice de la columna. Solamente hay un estado objetivo: (5,8), que es el ataúd de Drácula.

b) Definir el modelo de transición y las recompensas de estado.

**Modelo de transición**

Sea s=(i,j) un estado, y s'=(i',j') el estado que se pretende con la acción a. Entonces el modelo de transición viene dado por las siguientes probabilidades de transición:

P( s' | s,a )=1 si s se corresponde con un cuadrado blanco.
P( s' | s,Up )=0.85 si s se corresponde con un cuadrado azul.
P((i+1,j) | s,Up )=0.15 si s se corresponde con un cuadrado azul.

Nótese que algunas probabilidades de transición no se han definido, como por ejemplo P( s' | s,Down ) para un cuadrado azul. Esto se debe a que el ejercicio no las especifica.

**Recompensas de estado**:

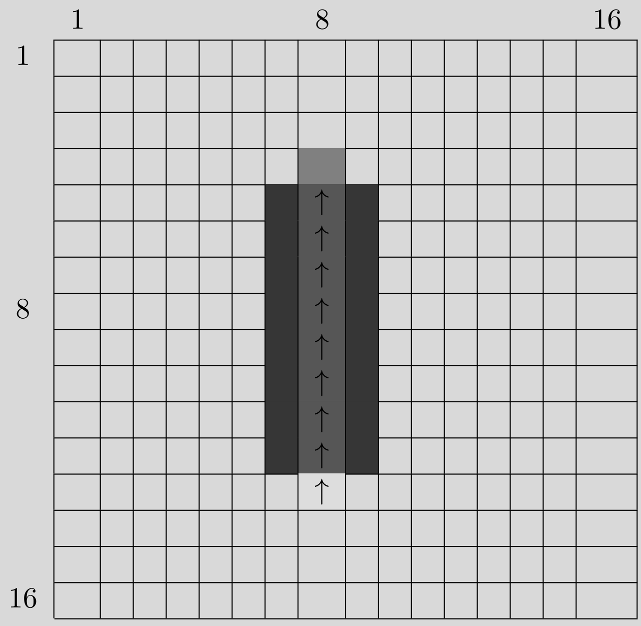
Sea s=(i,j) un estado. Entonces la recompensa de estado es:
R(s)=–2 si s se corresponde con un cuadrado blanco.
R(s)=–3 si s se corresponde con un cuadrado azul.
R(s)=0 en otro caso.

c) Definir dos posibles políticas para este problema: una que lo guíe a través del corredor, y otra que lo evite.

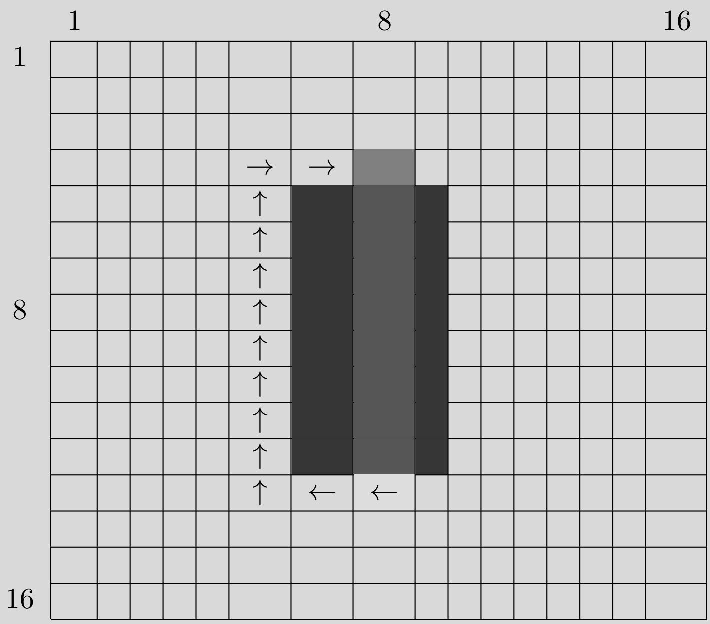
**Posibles políticas**:

La política que va a través del corredor viene dada por:
$\forall s \pi(s)$=Up

Esto se puede representar como sigue (las acciones recomendadas fuera del corredor son irrelevantes para esta política)



Más abajo se da una posible política que evita el corredor. Nótese que no hay posibilidad de que nos alejemos del camino que pretendemos seguir, así que las acciones recomendadas fuera de dicho camino son irrelevantes.



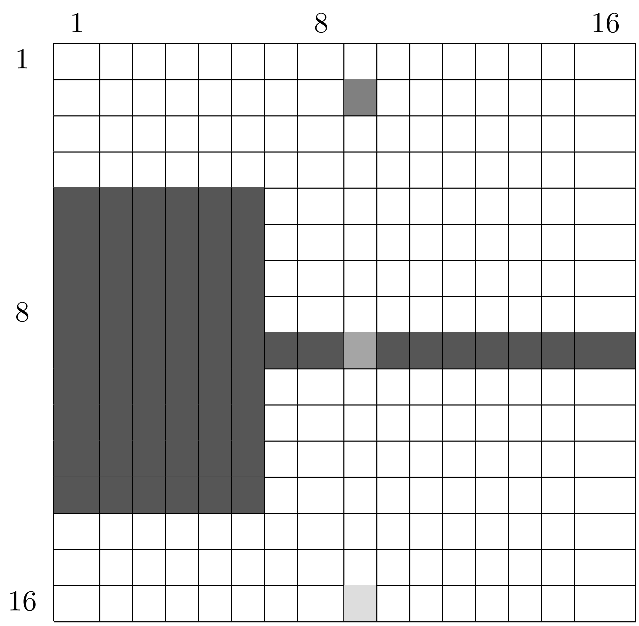
**Ejercicio 12.2** Un mercader (marcado en verde claro) quiere llevar sus mercancías a la ciudad (marcada en rojo). Hay un río que no se puede vadear marcado en azul. Su objetivo es llegar a la ciudad perdiendo el menor dinero posible. Cada día puede elegir una de cuatro acciones: Up, Down, Left y Right.

Cada día pierde 1 unidad de oro debido al consumo de vituallas. Además, si cruza el puente (marcado en rosa) debe pagar 100 unidades de oro al sheriff de Nottingham. Si cruza por el bosque (marcado en verde oscuro), tiene que pagar 2 unidades de oro a Robin Hood cada día, y hay una probabilidad del 60 % de que tenga que huir en una dirección distinta a la que pretendía (un 20 % de probabilidad en cada una de las 3 direcciones alternativas). Nótese que podría tener que huir varias veces durante su estancia en el bosque. Los cuadrados normales (marcados en blanco) no presentan ningún peligro: nadie le quitará dinero, y logrará el resultado pretendido sea cual sea su acción. Se pide:

a) Definir el entorno de este problema de decisión.

b) Definir el modelo de transición y las recompensas de estado.

c) Definir dos posibles políticas para este problema: una que vaya a través del bosque, y otra que cruce el puente.



**Solución**

a) Definir el entorno de este problema de decisión.

**Entorno**:

Hay 16·16=256 estados, que podemos identificar mediante pares (i,j), donde i es el índice de la fila y j es el índice de la columna. Solamente hay un estado objetivo: (2,9), que es la ciudad.

b) Definir el modelo de transición y las recompensas de estado.

**Modelo de transición**:

Sea s=(i,j) un estado, y s'=(i',j') el estado que se pretende con la acción a. Además, sea s"=(i",j") alguno de los estados no pretendidos para la acción a. Entonces el modelo de transición viene dado por estas probabilidades de transición:

P( s' | s,a )=1 si s se corresponde con un cuadrado blanco o rosa.
P( s' | s,a )=0,40 si s se corresponde con un cuadrado verde oscuro.
P(s" | s,a )=0,20 si s se corresponde con un cuadrado verde oscuro. Nótese que si s sólo tiene tres vecinos (porque s es adyacente a un borde del bosque), entonces esta probabilidad sube a 0,30.

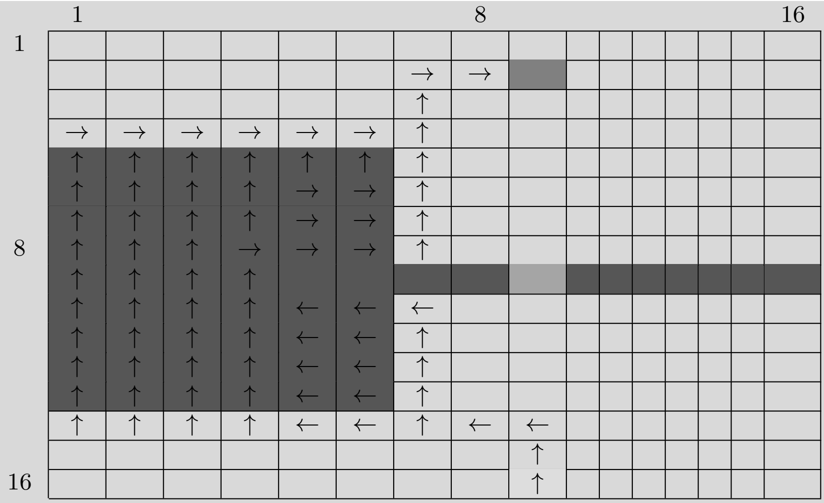
**Recompensas de estado**:

Sea s=(i,j) un estado. Entonces la recompensa de estado es:
R(s)=–1 si s se corresponde con un cuadrado blanco.
R(s)=–3 si s se corresponde con un cuadrado verde oscuro.
R(s)=–101 si s se corresponde con el cuadrado rosa, es decir, s=(9,9).
R(s)=0 en otro caso.

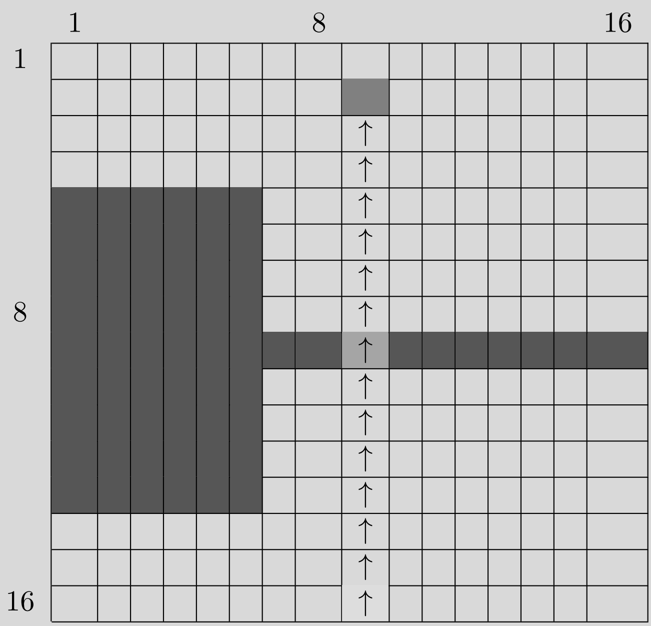
c) Definir dos posibles políticas para este problema: una que vaya a través del bosque, y otra que cruce el puente.

**Posibles políticas**:

Más abajo se da una posible política que va a través del bosque. Nótese que no hay necesidad de especificar la acción recomendada para todos los estados, ya que la mayoría de los estados nunca serán visitados bajo esta política:




A continuación se da una posible política que cruza el puente. En este caso, la probabilidad de que nos desvíen de nuestro camino es cero, así que las direcciones recomendadas fuera del camino recomendado son irrelevantes:



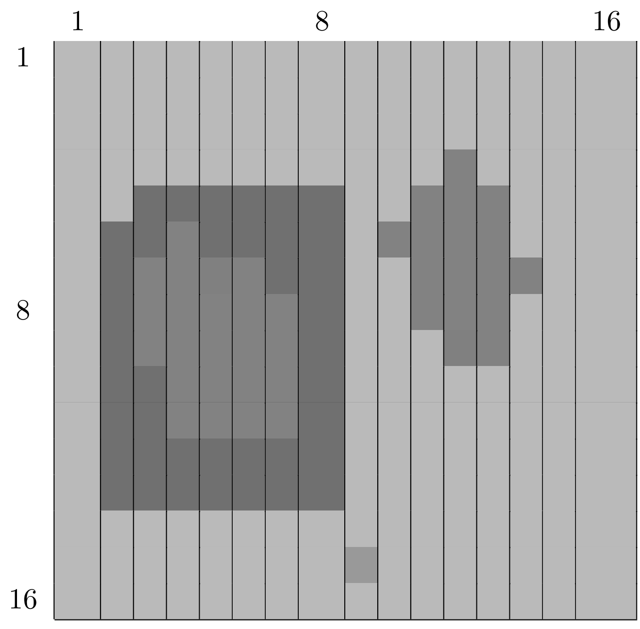
**Ejercicio 12.3** El capitán de un barco pirata (mostrado en gris) quiere llegar a uno cualquiera de los dos puertos piratas (mostrados en rojo) para vender el botín. Si va al puerto de la izquierda, ganará 1000 piezas de oro. Por otro lado, si va al puerto de la derecha, ganará 500 piezas de oro. Cada día de navegación supone un coste de 20 piezas de oro debido al consumo de vituallas y al mantenimiento del barco.

Los cuadrados marrones son tierra, así que no pueden atravesarse. Los cuadrados azules claros no presentan peligro: la acción pretendida siempre es ejecutada. Sin embargo, en los cuadrados azules oscuros hay una probabilidad del 60 % de que no haya viento y el barco no se mueva. Se pide:

a) Definir el entorno de este problema de decisión.

b) Definir el modelo de transición y las recompensas de estado.

c) Definir dos posibles políticas para este problema: una que vaya al puerto de la izquierda, y otra que vaya al puerto de la derecha.



**Solución**

a) Definir el entorno de este problema de decisión.

**Entorno**:

Hay 16·16=256 estados, que podemos identificar mediante pares (i,j), donde i es el índice de la fila y j es el índice de la columna. Hay dos estados objetivo: (7,5), que es el puerto de la izquierda; y (9,12), que es el puerto de la derecha.

b) Definir el modelo de transición y las recompensas de estado.

**Modelo de transición**:

Sea s=(i,j) un estado, y s'=(i',j') el estado que se pretende con la acción a. Entonces el modelo de transición viene dado por estas probabilidades de transición:
P( s' | s,a )=1 si s se corresponde con un cuadrado azul claro.
P( s' | s,a )=0,40 si s se corresponde con un cuadrado azul oscuro.
P( s | s,a )=0,60 si s se corresponde con un cuadrado azul oscuro.

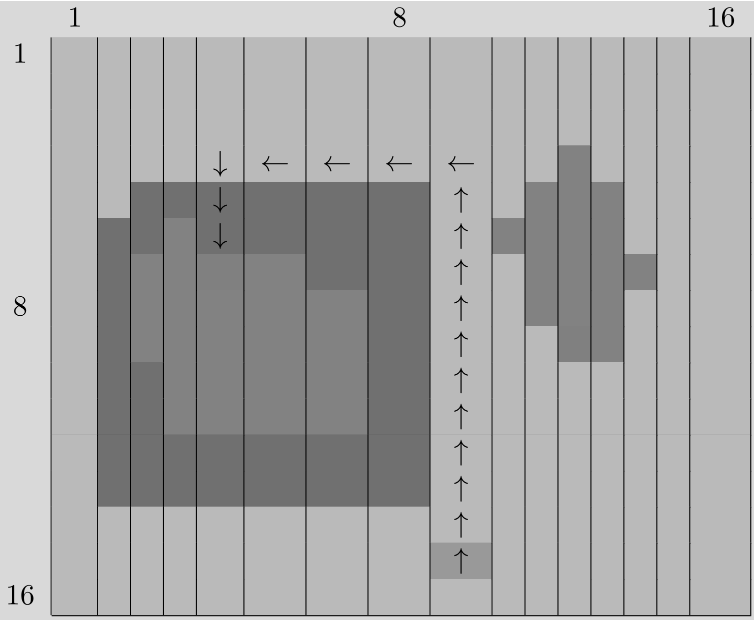
**Recompensas de estado**:

Sea s=(i,j) un estado. Entonces la recompensa de estado es:
R(s)=–20 si s se corresponde con un cuadrado azul claro o azul oscuro.
R(s)=1000 si s=(7,5), que es el puerto de la izquierda.
R(s)=500 si s=(9,12), que es el puerto de la derecha.
R(s)=0 en otro caso.

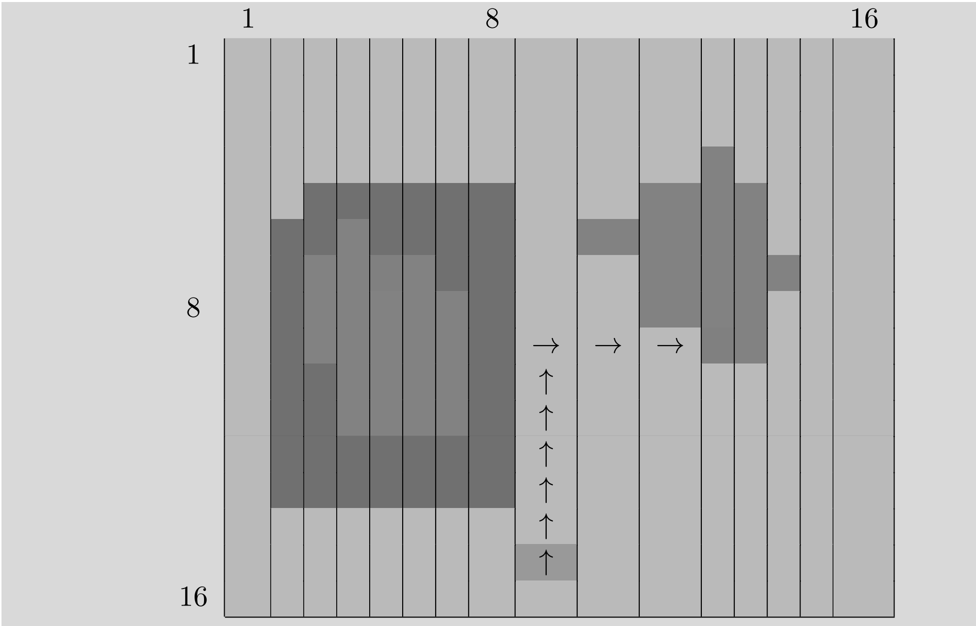
c) Definir dos posibles políticas para este problema: una que vaya al puerto de la izquierda, y otra que vaya al puerto de la derecha.

**Posibles políticas**:

Más abajo se da una posible política que va a través del bosque. Nótese que no hay necesidad de especificar la acción recomendada para todos los estados, ya que la mayoría de los estados nunca serán visitados bajo esta política:




A continuación se da una posible política que cruza el puente. En este caso, la probabilidad de que nos desvíen de nuestro camino es cero, así que las direcciones recomendadas fuera del camino recomendado son irrelevantes:




**Ejercicio 12.4** Un jugador tiene actualmente $s$ euros. Su objetivo es alcanzar una fortuna de $N$ euros. Puede apostar cualquier cantidad $a \in 1, 2, \ldots, min(s, N-s)$. Gana la cantidad que ha apostado con probabilidad $p$ y pierde con probabilidad $1-p$. El proceso continúa hasta que consigue su objetivo o bien pierde todo su dinero. Se pide:

a) Definir el entorno de este problema de decisión.

b) Definir el modelo de transición y las recompensas de estado.

c) Definir dos posibles políticas para este problema: una conservadora que siempre apueste 1 euro, y una agresiva que siempre apueste la máxima cantidad para conseguir el objetivo.

**Solución**

a) Definir el entorno de este problema de decisión.

**Entorno**:

Hay N+1 estados que corresponden con las posibles cantidades de dinero que tiene el jugador: 0, 1, ... , N. Hay dos estados objetivo: 0 y N. Nótese que 0 es un estado objetivo porque el proceso termina si se llega a ese estado, aunque no sea deseable para el jugador

b) Definir el modelo de transición y las recompensas de estado.

**Modelo de transición**:

Sea $s \in 0, 1, \ldots, N$ un estado y una acción $a \in 1, 2, \ldots, min(s, N - s)$. Entonces el modelo de transición viene dado por las siguientes probabilidades de transición:

P( s | s,a )=1 si $s \in \{0, N\}$.
P( s+a | s,a )=p si $s \notin \{0, N\}$.
P( s–a | s,a )=1–p si $s \notin \{0, N\}$.

**Recompensas de estado**:

Sea $s \in 0, 1, \ldots, N$ un estado. El objetivo del jugador es llegar a tener N euros, pero no obtiene ninguna recompensa inmediata en los estados intermedios. Así pues, las recompensas son no nulas solamente para los estados objetivo:

R(0)=–1.
R(N)=1.
R(s)=0 si $s \notin \{0, N\}$.

c) Definir dos posibles políticas para este problema: una conservadora que siempre apueste 1 euro, y una agresiva que siempre apueste la máxima cantidad para conseguir el objetivo.

**Posibles políticas**:

La política conservadora viene dada por:

$\forall s \notin \{0, N\}, \pi(s) = 1$

La política agresiva viene dada por:

$\forall s \notin \{0, N\}, \pi(s) - min(s, N - s)$

**Ejercicio 12.5** Una persona quiere vender su casa. Recibe una oferta cada semana. Supongamos que las ofertas son independientes y tienen un valor de $j$ euros con probabilidad $p_j$, para $j \in 0, 1, \ldots, N$. Supondremos también que una oferta que no es aceptada inmediatamente puede aceptarse en alguna semana posterior. Mientras la casa siga sin venderse hay unos costes de mantenimiento c cada semana. Se pide:

a) Definir el entorno de este problema de decisión.

b) Definir el modelo de transición y las recompensas de estado.

c) Definir una política razonable para vender la casa.

**Solución**

a) Definir el entorno de este problema de decisión.

**Entorno**:

Hay N+1 estados que corresponden a las máximas ofertas hasta el momento (sin que hayamos vendido la casa todavía): $NotSold_0$, $NotSold_1$, ..., $NotSold_N$. También hay N+1 estados objetivo que se corresponden con los posibles precios finales: $Sold_0$, $Sold_1$, ..., $Sold_N$.

b) Definir el modelo de transición y las recompensas de estado.

**Modelo de transición**:

Los estados objetivo $Sold_0$, $Sold_1$, ..., $Sold_N$ no tienen ninguna acción disponible. Por otra parte, sean $s = NotSold_i$ un estado no objetivo, y a una acción, $a \in \{Sell, DontSell\}$. Las probabilidades de transición son las siguientes:

P( Soldi | NotSoldi, Sell ) = 1.
P(NotSoldi | NotSoldi, DontSell) = $1 - p_{i+1} - p_{i+2} - \ldots - p_N$.
P(NotSoldj | NotSoldi, DontSell) = $p_j$ si j>i.
P(NotSoldj | NotSoldi, DontSell) = 0 si j<i, puesto que la nueva oferta es peor que la máxima oferta hasta el momento.

**Recompensas de estado**:

Las recompensas para los estados objetivo son los precios de venta de la casa. Las recompensas para los demás estados son negativas, y corresponden al coste de mantenimiento semanal:

R($Sold_i$) = i.
R($NotSold_i$) = –c.

c) Definir una política razonable para vender la casa.

**Posible política**:

Si estamos en el estado $NotSold_i$, el incremento esperado en la oferta para la siguiente semana viene dado por:

$$\text{E[OfferIncrease]}= \sum_{j=i+1}^{N} (j-i)p_j$$

Como el coste de esperar otra semana es c, podríamos decidir vender la casa si el coste de esperar otra semana es mayor que el incremento esperado en la oferta:

$$\pi(NotSold_i) = \begin{cases} Sell & \text{sii E[OfferIncrease]} \leq c \\ DontSell & \text{sii E[OfferIncrease]} > c \end{cases}$$

**Ejercicio 12.6** Un coche puede estar en uno de los estados $0, 1, \ldots, N$, donde 0 representa a un coche nuevo con muy poco mantenimiento, y $N$ representa un coche viejo, muy costoso de mantener. Cada año tenemos que decidir si seguimos con el coche viejo con un coste de mantenimiento $c_i$, donde $i \in 0, 1, \ldots, N$ es el estado actual del coche. Nótese que $c_i < c_{i+1}$, es decir, los costes de mantenimiento suben a medida que el estado del coche empeora. Si vendemos el coche, pagamos un precio $k$ por un coche nuevo (en el estado 0) y nos dan $t_i$ por el viejo. Nótese que $t_i > t_{i+1}$, esto es, el precio de venta del coche disminuye a medida que empeora su estado. La probabilidad de que un coche en el estado $i$ esté en el estado $j$ el próximo año es $p_{ij}$, donde $p_{ij} = 0$ para $j < i$, esto es, el estado de un coche usado no puede mejorar.

Se pide:

a) Definir el entorno de este problema de decisión.

b) Definir el modelo de transición y las recompensas de estado.

c) Definir una política razonable para cambiar el coche.

**Solución**

a) Definir el entorno de este problema de decisión.

**Entorno**:

Hay N+1 estados que corresponden a los estados del coche viejo mientras se está utilizando: $NotReplaced_0$, $NotReplaced_1$, ..., $NotReplaced_N$. Hay también N+1 estados que corresponden a los posibles estados del coche viejo en el momento de su reemplazo: $Replaced_0$, $Replaced_1$, ..., $Replaced_N$. No hay estados objetivo, dado que el proceso de reemplazo continúa indefinidamente.

b) Definir el modelo de transición y las recompensas de estado.

**Modelo de transición**:

Los estados $NotReplaced_0$, $NotReplaced_1$, ..., $NotReplaced_N$ solamente tienen una acción disponible UseNew, que quiere decir que empezamos a usar el coche nuevo. Por el contrario, en los estados $s = NotReplaced_i$ tenemos que elegir entre dos acciones disponibles $a \in \{Replace, DontReplace\}$. Las probabilidades de transición son como sigue:

P($NotReplaced_0$ | $Replaced_i$, UseNew ) = 1.
P($NotReplaced_j$ | $Replaced_i$, UseNew ) = 0 si j>0.
P( $Replaced_i$ | $NotReplaced_i$, Replace ) = 1.
P( $NotReplaced_j$ | $NotReplaced_i$, DontReplace ) = $p_{ij}$ si j $\geq$ i.
P( $NotReplaced_j$ | $NotReplaced_i$, DontReplace ) = 0 si j<i.
El resto de probabilidades de transición son cero.

**Recompensas de estado**:

La recompensa para un estado $Replaced_i$ es la diferencia entre el precio de venta del coche viejo y el precio de compra del coche nuevo. Las recompensas para los estados $NotReplaced_i$ son los costes de mantenimiento:

R($Replaced_i$) = $t_i$–k  0.
R($NotReplaced_i$) = $-c_i$  0.

c) Definir una política razonable para cambiar el coche.

**Posible política**:

La acción recomendada para los estados $Replaced_i$ es siempre Use-New ya que no hay ninguna otra acción disponible:

$$\pi(Replaced_i) = UseNew$$

Si estamos en el estado $NotReplaced_i$, el coste de mantenimiento esperado para el próximo año viene dado por:

$$E[MaintenanceCost] = \sum_{j=i}^{N} c_j p_{ij}$$

Como el coste de cambiar el coche es k–ti, podríamos decidir cambiar el coche cuando el coste de mantenimiento si seguimos con él es mayor que el coste de comprar uno nuevo:

$$\pi(NotReplaced_i) = \begin{cases} Replace & \text{sii } E[MantenanceCost] \geq k - t_i \\ DontReplace & \text{sii } E[MantenanceCost] < k - t_i \end{cases}$$

Nótese que esta política no será óptima en la mayoría de los casos, ya que no tiene en cuenta el coste de mantenimiento acumulado de mantener un coche viejo durante varios años.

**Ejercicio 12.7** Halla la acción más adecuada para la celda (3,2) del mundo 4x3:

| | | | | |
|---|---|---|---|---|
| 1 | | | 0,8 | 1 |
| 2 | | | **0,4** | -1 |
| 3 | | | 0,2 | |
| y / x | 1 | 2 | 3 | 4 |

Ten en cuenta lo siguiente:

- Las acciones en cada estado son Izquierda, Derecha, Arriba y Abajo.
- Cada acción logra el efecto pretendido con probabilidad 0.8, pero el resto de las veces la acción mueve al agente en una dirección perpendicular a la pretendida, con una probabilidad 0.1 en cada dirección.

- Si el agente choca con un muro, permanece en el mismo cuadrado.

Izquierda: ____________________
Derecha: ____________________
Arriba: ____________________
Abajo: ____________________

Marque Acción Más Adecuada: ☐ Izquierda ☐ Derecha ☐ Arriba ☐ Abajo

**Solución**

Izquierda: 0,4 x 0,8 + 0,8 x 0,1 + 0,2 x 0,1 = 0,42
Derecha: -1 x 0,8 + 0,8 x 0,1 + 0,2 x 0,1 = -0,7
Arriba: 0,8 x 0,8 + 0,4 x 0,1 + -1 x 0,1 = 0,58
Abajo: 0,2 x 0,8 + 0,4 x 0,1 + -1 x 0,1 = 0,1

Marque Acción Más Adecuada: ☐ Izquierda ☐ Derecha ■ Arriba ☐ Abajo

**Ejercicio 12.8** Halla la acción más adecuada para la celda (3,2) del mundo 4x3:.

| | | | | |
|---|---|---|---|---|
| 1 | | | 0,8 | 1 |
| 2 | | ■ | **0,4** | -1 |
| 3 | | | 0,2 | |
| y / x | 1 | 2 | 3 | 4 |

Ten en cuenta lo siguiente:

- Las acciones en cada estado son Izquierda, Derecha, Arriba y Abajo.
- Cada acción logra el efecto pretendido con probabilidad 0.6. El resto de las veces la acción mueve al agente en una dirección perpendicular a la pretendida, con una probabilidad 0.1 en cada dirección y 0.2 en la dirección contraria.
- Si el agente choca con un muro, permanece en el mismo cuadrado.

Izquierda: ______________________________
Derecha: ______________________________
Arriba: ______________________________
Abajo: ______________________________

Marque Acción Más Adecuada: ☐ Izquierda ☐ Derecha ☐ Arriba ☐ Abajo

**Solución**

Izquierda: 0,4 x 0,6 + 0,8 x 0,1 + 0,2 x 0,1 + -1 x 0,2 = 0,14
Derecha: -1 x 0,6 + 0,8 x 0,1 + 0,2 x 0,1 + 0,4 x 0,2 = -0,42
Arriba: 0,8 x 0,6 + 0,4 x 0,1 + -1 x 0,1 + 0,2 x 0,2 = 0,46
Abajo: 0,2 x 0,6 + 0,4 x 0,1 + -1 x 0,1 + 0,8 x 0,2 = 0,22

Marque Acción Más Adecuada: ☐ Izquierda ☐ Derecha ■ Arriba ☐ Abajo

**Ejercicio 12.9** Halla la acción más adecuada para la celda (3,1) del mundo 4x3:

| | | | | |
|---|---|---|---|---|
| 1 | | | 0,8 | 1 |
| 2 | | | **0,4** | -1 |
| 3 | | | 0,2 | |
| y / x | 1 | 2 | 3 | 4 |

Ten en cuenta lo siguiente:

- Las acciones en cada estado son Izquierda, Derecha, Arriba y Abajo.
- Cada acción logra el efecto pretendido con probabilidad 0'7. El resto de las veces la acción mueve al agente en una dirección contraria a la pretendida, con una probabilidad 0'3.
- Si el agente choca con un muro, permanece en el mismo cuadrado

Izquierda: ________________________________________
Derecha: ________________________________________
Arriba: ________________________________________
Abajo: ________________________________________

Marque Acción Más Adecuada: ☐ Izquierda ☐ Derecha ☐ Arriba ☐ Abajo

**Solución**

Izquierda: 0,4 x 0,7 + -1 x 0,3 = -0,02
Derecha: -1 x 0,7 + 0,4 x 0,3 = -0,58
Arriba: 0,8 x 0,7 + 0,2 x 0,3 = 0,62
Abajo: 0,2 x 0,7 + 0,8 x 0,3 = 0,38

Marque Acción Más Adecuada: ☐ Izquierda ☐ Derecha ■ Arriba ☐ Abajo

**Ejercicio 12.10** Halla la acción más adecuada para la celda (3,2) del mundo 4x3:

| | | | | |
|---|---|---|---|---|
| 1 | | | 0,8 | 1 |
| 2 | | | **0,5** | -1 |
| 3 | | | 0,2 | |
| y / x | 1 | 2 | 3 | 4 |

Ten en cuenta lo siguiente:

- Las acciones en cada estado son Izquierda, Derecha, Arriba y Abajo.
- Cada acción logra el efecto pretendido con probabilidad 0'5. El resto de las veces la acción mueve al agente en una dirección perpendicular a la pretendida, con una probabilidad 0'2 en cada dirección y 0'1 en la dirección contraria a la pretendida
- Si el agente choca con un muro, permanece en el mismo cuadrado

Izquierda: ________________________________
Derecha: ________________________________
Arriba: ________________________________
Abajo: ________________________________

Marque Acción Más Adecuada: ☐ Izquierda ☐ Derecha ☐ Arriba ☐ Abajo

**Solución**

Izquierda: 0,5x 0,5 + 0,8 x 0,2 + 0,2 x 0,2 + -1x 0,1 = 0,35
Derecha: -1 x 0,5 + 0,2 x 0,2 + 0,8 x 0,2 + 0,5 x 0,1 = -0,25
Arriba: 0,8 x 0,5 + -1 x 0,2 + 0,5 x 0,2 + 0,2 x 0,1 = 0,32
Abajo: 0,2 x 0,5 + -1 x 0,2 + 0,5 x 0,2 + 0,8 x 0,1 = 0,08

Marque Acción Más Adecuada: ■ Izquierda ☐ Derecha ☐ Arriba ☐ Abajo

**Ejercicio 12.11** Halla la acción más adecuada para la celda (3,2) del mundo 4x3:

| | | | | |
|---|---|---|---|---|
| 1 | | | 0,8 | 1 |
| 2 | | ■ | **0,5** | -0,1 |
| 3 | | | 0,2 | |
| y / x | 1 | 2 | 3 | 4 |

Ten en cuenta lo siguiente:

- Las acciones en cada estado son Izquierda, Derecha, Arriba y Abajo.
- Cada acción logra el efecto pretendido con probabilidad 0'5. El resto de las veces la acción mueve al agente en una dirección perpendicular a la pretendida, con una probabilidad 0'2 en cada dirección y 0'1 en la dirección contraria a la pretendida.
- Si el agente choca con un muro, permanece en el mismo cuadrado

Izquierda: ______________________
Derecha: ______________________
Arriba: ______________________
Abajo: ______________________

Marque Acción Más Adecuada: ☐ Izquierda ☐ Derecha ☐ Arriba ☐ Abajo

**Solución**

Izquierda: 0,5x 0,5 + 0,8 x 0,2 + 0,2 x 0,2 + -0,1x 0,1 = 0,44
Derecha: -0,1 x 0,5 + 0,2 x 0,2 + 0,8 x 0,2 + 0,5 x 0,1 = 0,2
Arriba: 0,8 x 0,5 + -0,1 x 0,2 + 0,5 x 0,2 + 0,2 x 0,1 = 0,5
Abajo: 0,2 x 0,5 + -0,1 x 0,2 + 0,5 x 0,2 + 0,8 x 0,1 = 0,26

Marque Acción Más Adecuada: ☐ Izquierda ☐ Derecha ■ Arriba ☐ Abajo

**Ejercicio 12.12** Basándonos en la siguiente representación (u=utilidades, r = recompensa) y el modelo de transición de la matriz adjunta.

| | | |
|---|---|---|
| | u=4 | |
| u=3 | r = -0,02 | u= -1 |
| | u=1 | |

| | **Arriba** | **Abajo** | **Izquierda** | **Derecha** |
|---|---|---|---|---|
| **Arriba** | 0,8 | 0 | 0,1 | 0,1 |
| **Abajo** | 0 | 0,8 | 0,1 | 0,1 |
| **Izquierda** | 0,1 | 0,1 | 0,8 | 0 |
| **Derecha** | 0,1 | 0,1 | 0 | 0,8 |

a) Elegir la mejor acción de las cuatro disponibles (Arriba, Abajo, Izquierda, Derecha).

b) ¿Estimar la utilidad esperada de la casilla central.

**Solución**

a) Elegir la mejor acción de las cuatro disponibles (Arriba, Abajo, Izquierda, Derecha).

= recompensa + max 0,8*3+ 0,1*4 + 0,1*1 (izq), 0,8*4+ 0,1*3 + 0,1*-1 (arb), 0,8*1+ 0,1*3 + 0,1*-1 (aba), 0,8*(-1)+0,1*4 + 0,1*1 (dch)

= -0,02+max2,9(izq), 3,4 (arb),1 (aba), -0,3 (dch)

La mejor acción sería **Arriba**.

b) ¿Estimar la utilidad esperada de la casilla central.

= -0,02+3,4 = 3,38

**Ejercicio 12.13** Halla la acción más adecuada para la celda (3,1) del mundo 4x3:

| | | | | |
|---|---|---|---|---|
| 1 | | | 0,8 | 1 |
| 2 | | | 0,4 | -1 |
| 3 | | | | |
| y / x | 1 | 2 | 3 | 4 |

Ten en cuenta lo siguiente:

- Las acciones en cada estado son Izquierda, Derecha, Arriba y Abajo.
- Cada acción logra el efecto pretendido con probabilidad 0'8, pero el resto de las veces la acción mueve al agente en una dirección perpendicular a la pretendida, con una probabilidad 0'1 en cada dirección.
- Si el agente choca con un muro, permanece en el mismo cuadrado

Izquierda: ____________________
Derecha: ____________________
Arriba: ____________________
Abajo: ____________________

Marque Acción Más Adecuada: ☐ Izquierda ☐ Derecha ☐ Arriba ☐ Abajo

**Solución**

Izquierda: 0,8 x 0,8 + 0,8 x 0,1 + 0,4 x 0,1 = 0,76
Derecha: 1 x 0,8 + 0,4 x 0,1 + 0,8 x 0,1 = 0,92
Arriba: 0,8 x 0,8 + 0,8 x 0,1 + 1 x 0,1 = 0,82
Abajo: 0,4 x 0,8 + 0,8 x 0,1 + 1 x 0,1 = 0,5

Marque Acción Más Adecuada: □ Izquierda ■ Derecha □ Arriba □ Abajo

umaeditorial