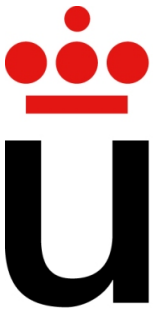


Universidad
Rey Juan Carlos

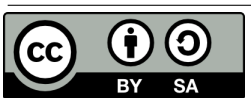
ESCUELA TÉCNICA SUPERIOR
DE INGENIERÍA INFORMÁTICA

Ejercicios

INTELIGENCIA ARTIFICIAL

CURSO 2022-2023

Autores: **Sascha Ossowski**
Holger Billhardt
Alberto Fernández-Gil



Grupo de Inteligencia Artificial
Universidad Rey Juan Carlos

Hoja de Problemas Tema 2
Búsqueda no-informada

Universidad Rey Juan Carlos

Ejercicio 1:

1.1. El enfoque de los Agentes Inteligentes concibe el objetivo de la Inteligencia Artificial como el intento de construir sistemas

(a) que "actúen" como las personas (tipo Eliza).

(b) que "actúen" de forma eficiente (racional) en su entorno.

(c) que "piensen" como las personas (tipo General Problem Solver, GPS).

(d) Ninguna de las respuestas es correcta.

1.2. ¿Para cual(es) de las siguientes tareas pueden construirse agentes basados en algoritmos de búsqueda en espacios de estados (p.e. la búsqueda de coste uniforme)?

(a) Encontrar una solución al problema de las Torres de Hanoi.

(b) Gestionar el tráfico rodado en una red de autopistas urbanas.

(c) Conducir un taxi en las calles de Barcelona.

(d) Jugar a las cuatro en raya contra un jugador humano.

1.3. ¿Cuáles de las siguientes afirmaciones acerca de los algoritmos de búsqueda no informados es (son) cierta(s)?

(a) Los algoritmos de búsqueda no informados requieren de información heurística para que sean óptimos.

(b) La búsqueda en amplitud es óptima y completa siempre y cuando el coste de los operadores sea constante.

(c) La búsqueda en profundidad es óptima y completa siempre que el coste de los operadores sea constante.

(d) Tanto la complejidad en tiempo como la complejidad en espacio de la búsqueda en amplitud se pueden expresar en función del número de nodos expandidos

1.4. Contemple el problema de búsqueda de la figura 1. ¿Cuál es el factor de ramificación del árbol de búsqueda generado por los métodos de búsqueda en el espacio de estados, si no se filtran estados repetidos?

(a) 0.5

(b) 2

(c) 4

(d) 6

1.5. Contemple el problema de búsqueda de la figura 1. En el árbol generado por la búsqueda en amplitud, si no se filtran estados repetidos, ¿cuántos nodos hay a nivel de profundidad k (suponiendo que la raíz tiene nivel 0, es decir $k \geq 0$)

(a) $k/4$

(b) 4^k

(c) $4*k^2$

(d) k^4

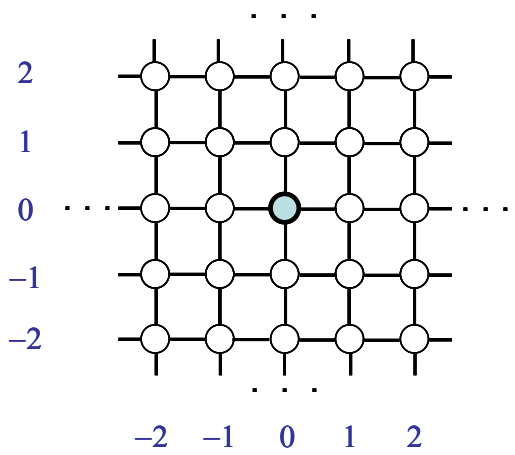


Figura 1. *Red 2D regular e infinito.*

- *Estado inicial (0,0).*
- *Estado meta (x,y); x,y ∈Z.*
- *Movimiento entre estados directamente conectados a coste 1*

Ejercicio 2:

En una mesa se encuentran dos jarras, una con una capacidad de 3 litros (llamada *Tres*), y la otra con una capacidad de 4 litros (llamada *Cuatro*). Inicialmente, *Tres* y *Cuatro* están vacías. Cualquiera de ellas puede llenarse con el agua de un grifo *G*. Asimismo, el contenido tanto de *Tres* como de *Cuatro* puede vaciarse en una pila *P*. Es posible echar todo el agua de una jarra a la otra. No se dispone de dispositivos de medición adicionales. Se trata de encontrar una secuencia de operadores que deje exactamente dos litros de agua en *Cuatro*.

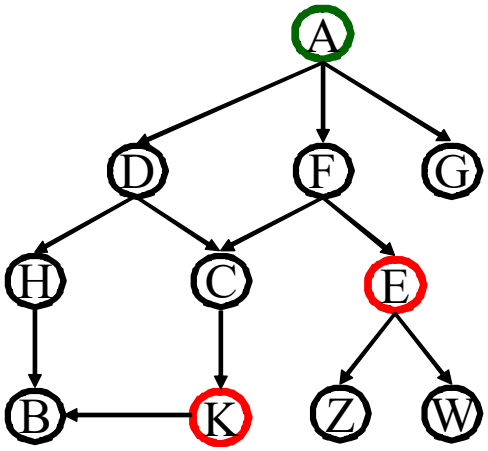
a) Modele este problema como un problema de búsqueda. Con tal fin, defina el estado inicial, el conjunto de estados meta, los operadores (especificando sus precondiciones y postcondiciones), así como el coste de cada operador.

b) Caracterice el conocimiento a priori del agente de resolución del problema correspondiente? Facilite ejemplos de los resultados de la función expandir.

c) Encuentre una solución al problema.

Ejercicio 3:

El grafo que se muestra al lado determina un problema de búsqueda. Cada nodo representa un estado; los arcos modelan la aplicación de operadores. Suponga que *A* es el estado inicial y que *K* y *E* son estados meta



a) Desarrolle el árbol de búsqueda que genera la búsqueda en amplitud, *sin* filtrar estados repetidos. ¿Cuál de los nodos meta se encuentra primero?

b) Indique el orden en que se expanden los nodos

c) Ponga el estado de la lista abierta en cada paso del algoritmo

Ejercicio 4:

a) ¿Cómo se podría instanciar el algoritmo de búsqueda genérico (de la transparencia 19) para implementar una búsqueda en profundidad?

b) Para el grafo del ejercicio 3, desarrolle el árbol de búsqueda que genera la búsqueda en profundidad. Indique el orden en que se expanden los nodos ¿Cuál de los nodos meta se encuentra primero?

c) Haga un análisis de complejidad de dicho algoritmo, similar al de la transparencia 27, asumiendo un límite de profundidad d^* fijado a priori.

Ejercicio 5:

Considere el problema de las jarras del ejercicio 2. Simule la estrategia de búsqueda en amplitud para este problema, asumiendo que se filtran *todos los estados repetidos*. Indique el orden en que se expanden los nodos, y ponga el estado de la lista abierta en cada paso del algoritmo.

Ejercicio 6:

Considere el siguiente problema:

> *Un hombre se encuentra en la orilla izquierda de un río junto con un lobo, una oveja y una col. Quiere cruzar el río llevando consigo el lobo, la oveja y la col. En la barca sólo hay dos plazas, una de las cuales debe ir ocupada por el hombre. Cada uno de los restantes pasajeros (lobo, oveja, col) ocupa una plaza, de tal modo que sólo uno puede acompañar al hombre en cada viaje. Además, no puede dejar solos en una orilla al lobo con la oveja, ni a la oveja con la col (ni los tres), porque la primera se comería a la segunda en cada paso*

Suponga que se modela el problema como un espacio de estados, donde cada estado se describe como un par de conjuntos, indicando quien(es) se encuentran en cada orilla del río (c = col; o = oveja; l = lobo; h = hombre). Por tanto, el estado inicial del problema sería ($\{c,o,l,h\},\{\}$), y el estado meta ($\{\},\{c,o,l,h\}$).

a) Simule la estrategia de *búsqueda en amplitud* para este problema, asumiendo que se filtran *todos* los estados repetidos. Expanda los sucesores de los nodos siguiendo las preferencias del hombre, las cuales se ordenan (de mayor a menor) como sigue: viajar con la oveja, viajar con la col, viajar sólo, viajar con el lobo. Dibuje el árbol de búsqueda correspondiente e indique el orden en el que se exploran los nodos.

b) Simule ahora la estrategia de *búsqueda en profundidad* para este problema. Dibuje el árbol de búsqueda correspondiente e indique el orden en el que se exploran los nodos, asumiendo que se filtran *todos* los estados repetidos y siguiendo las preferencias del apartado anterior.

c) Suponga ahora que no se filtra ningún estado repetido. ¿La búsqueda en amplitud y la búsqueda en profundidad seguirían siendo completas? Razone brevemente su respuesta.

Ejercicio 7:

Suponga que en la red de carreteras de Rumania presentada en clase (transparencia 30) nuestra agente se encuentra en Caiva (C) y desea trasladarse a Fagaras (F)

a) Desarrolle el árbol de búsqueda que genera la *búsqueda de coste uniforme*, indicando los valores de g para cada nodo. Suponga que se filtran *ciclos simples*

b) Indique el orden en que se expanden los nodos

c) Ponga el estado de la lista abierta en cada paso del algoritmo

Hoja de Problemas
Tema 3
Búsqueda Heurística

1. Contesta a las siguientes preguntas:
   (a) ¿Cuál es el objetivo de una función heurística aplicada a la búsqueda en el espacio de estados?
   (b) ¿Cuál es la definición de heurística consistente?
   (c) ¿Cuál es la definición de heurística optimista?
   (d) ¿Una función heurística consistente es también optimista?
   (e) ¿Qué condiciones garantizan que el algoritmo A^* sea óptimo?

2. Dado el siguiente problema de búsqueda:

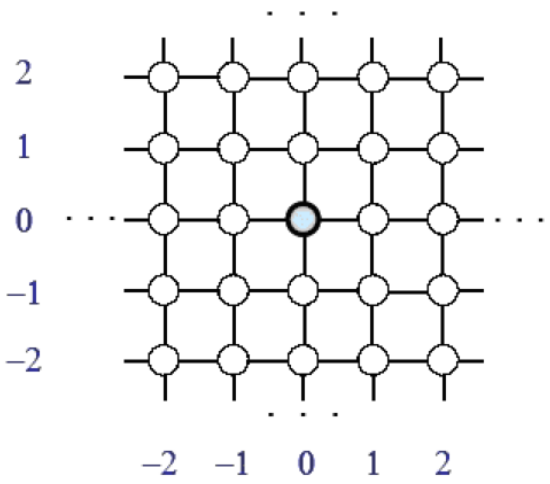
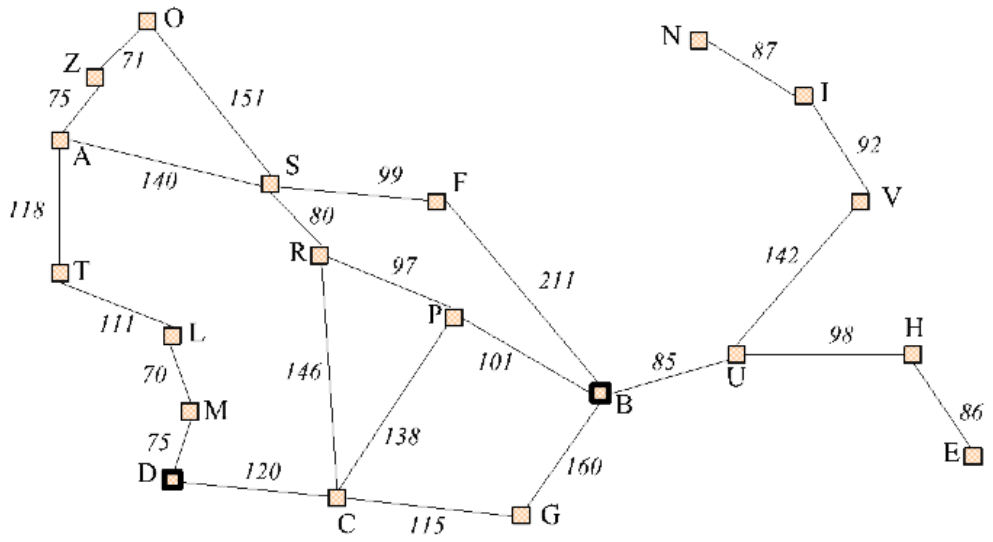


Figura 1: Red 2D regular e infinita, Estado inicial $(0,0)$, Estado final: $(x,y) \quad x,y \in Z$, Movimiento entre estados directamente conectados a coste 1

Por un genérico nodo (u,v), ¿cuáles de las siguientes funciones heurísticas son optimistas?

(a) $h^*((u,v)) = |u-x| + |v-y|$
(b) $h^*((u,v)) = |u-x| * |v-y|$
(c) $h^*((u,v)) = 2 * min(|u-x|, |v-y|)$
(d) $h^*((u,v)) = |u| + |x| + |v| + |y|$
(e) $h^*((u,v)) = \sqrt{(|u-x|^2 + |v-y|^2)}$

3. El grafo que se muestra a continuación representa un esquema de un mapa de carreteras. Los nodos están etiquetados con el nombre de ciudades, y los arcos con la distancia por carretera entre dichas ciudades. Inicialmente nuestro agente se encuentra en la ciudad D, y procura trasladarse por carretera a la ciudad B por el camino más corto. Con tal fin, puede hacer uso de su conocimiento de la región, y en particular de la distancia aérea entre ciudades. La función h^* que se muestra al lado indica la distancia aérea entre cualquier ciudad y la ciudad de destino B.



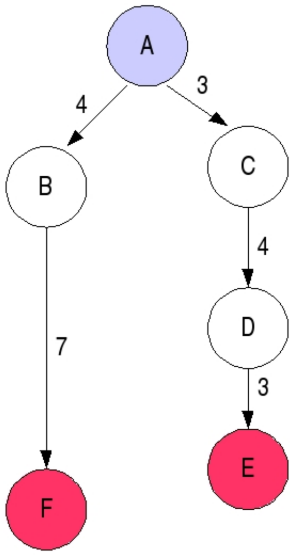
	h^*
A	366
B	0
C	160
D	242
E	161
F	178
G	77
H	151
I	226
L	244
M	241
N	234
O	380
P	98
R	193
S	253
T	329
U	80
V	199
Z	374

Suponga que el programa de agente se basa en el algoritmo A^* para resolver este problema.

(a) Desarrolle el árbol de búsqueda que genera el algoritmo A^*, asumiendo que se filtran ciclos simples (p.e.: $D - M - D$). Indique el orden en el que se expanden los nodos, así como el valor de g, h^*, y f^* de cada nodo del árbol de búsqueda.

(b) Muestre también el estado de la lista abierta en cada paso del algoritmo.

4. El grafo que se muestra en la figura describe un problema de búsqueda. Suponga que A es el estado inicial y que F y E son estados meta. Los arcos están etiquetados con el coste real de los operadores (hasta allí, el problema coincide con el de la transparencia 12 del Tema 3)



   (a) Asigne los valores de la función heurística h^*, de modo que resulte ser *optimista y no consistente*

   (b) Desarrolle el árbol de búsqueda que genera el algoritmo A^*. ¿Es el algoritmo A^* óptimo en este caso?

5. Considere el problema de los bloques cuyo estado inicial y estado meta se muestran en la siguiente figura:

   Desarrolle el árbol de búsqueda que expande el algoritmo A^*, utilizando la siguiente heurística:

$$h^*(n) = \textit{número de bloques descolocados}$$

   Con tal fin, considere que un bloque está descolocado si *por debajo* no tiene el elemento correcto (bien el bloque deseado o bien la mesa). Filtre los ciclos simples, indique el orden de expansión de los estados y muestre en cada paso los valores de f^*, g y h^*. Suponga que el coste de cada operador es 1.

6. Las recientes lluvias han provocado daños en la infraestructura de un municipio que deben ser reparados con urgencia. Concretamente, hay N obras por realizar y se ha pedido presupuesto a M empresas constructoras para cada una de las obras. El coste de encargar cada obra a cada empresa viene dado por una tabla como la siguiente, donde $C_{i,j}$ indica el coste de encargar a la empresa E_i la obra O_j

	Obra O_1	Obra O_2	...	Obra O_N
$EmpresaE_1$	$C_{1,1}$	$C_{1,2}$		$C_{1,N}$
$EmpresaE_2$	$C_{2,1}$	$C_{2,2}$		$C_{2,N}$
...				
$EmpresaE_M$	$C_{M,1}$	$C_{M,2}$		$C_{M,N}$

El Ayuntamiento ha decidido asignar *una sola obra por empresa*. El problema consiste en decidir qué obra se asignará a cada empresa, de modo que se minimice el coste total. Los técnicos deciden utilizar el algoritmo A^* para resolver el problema.

(a) Defina una representación "eficiente" del problema, especificando el conjunto de posibles estados, estado inicial, estados finales, así como operador(es) y su coste.

(b) Defina una "buena" función heurística h^* optimista para el problema general. ¿Es su función h^* también consistente?

(c) Considere el siguiente caso particular (los costes se expresan en millones de Euros)

	Obra O_1	Obra O_2	Obra O_3	Obra O_4
$EmpresaE_1$	2	3	2	4
$EmpresaE_2$	5	5	4	5
$EmpresaE_3$	6	5	4	3
$EmpresaE_4$	10	8	6	6

Desarrolle el árbol de búsqueda que genera el algoritmo A^* (puede suponer el "mejor caso"). Indique el orden en el que se expanden los nodos, los valores de g, h^* y f^* para cada nodo del árbol de búsqueda, y la evolución de la lista abierta.

Grupo de Inteligencia Artificial
Universidad Rey Juan Carlos

Hoja de Problemas Tema 4
Búsqueda heurística avanzada

Universidad Rey Juan Carlos

Ejercicio 1:

1.1 El que el algoritmo A^* utilice una función heurística h^* consistente
- (a) es condición necesaria para que $h^*(n) > g(n)$ en todos los nodos n del árbol de búsqueda.
- (b) es condición suficiente para que el valor de f^* crezca de forma monótona en todos los caminos del árbol de búsqueda.
- (c) es condición necesaria para que el algoritmo A^* sea óptimo (ó admisible).
- (d) es condición suficiente para que el algoritmo A^* sea óptimo (ó admisible).

1.2 ¿Cuáles de las siguientes afirmaciones acerca del *algoritmo* A^* son verdaderas y cuáles son falsas?
- (a) Si h^* es optimista, entonces el valor f^* crece de forma débilmente monótona en todos los caminos del árbol de búsqueda.
- (b) Si h^* es optimista, entonces también es consistente.
- (c) Si h^* es optimista, entonces el algoritmo A^* es óptimo.
- (d) Si h^* no es optimista, entonces tampoco es consistente.

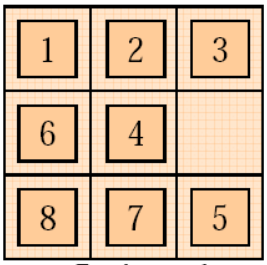
1.3 Si se aplica el algoritmo A^* con una función heurística h_1^*, tal que $h_1^*(n)=0$ para todos los nodos n, entonces
- (a) la función h_1^* es optimista.
- (b) todas las funciones heurísticas optimistas h_2^* son más informadas que h_1^*.
- (c) el algoritmo A^* expande los mismos nodos que la búsqueda de coste uniforme.
- (d) no existe ninguna función heurística optimistas h_2^* con la que el algoritmo A^* expande menos nodos que con h_1^*.

1.4 Suponga que para un problema a resolver con el algoritmo A^* se dispone de varias funciones heurísticas optimistas h_1^*, ..., h_l^*, todas ellas de fácil evaluación. ¿Cuáles de las siguientes afirmaciones son correctas y cuáles son falsas?
- (a) Si hay una función h_k^* que es más informada que h_i^*, entonces se puede prescindir de h_i^* en el proceso de búsqueda.
- (b) Si para un nodo n se cumple que $h_i^*(n) < h_k^*(n)$, entonces se puede prescindir de h_i^* en todo el proceso de búsqueda.
- (c) Para cada nodo n es conveniente elegir la función h_i^* de máximo valor.
- (d) Si h_i^* *no* es consistente, entonces se puede prescindir de h_i^* en el proceso de búsqueda.

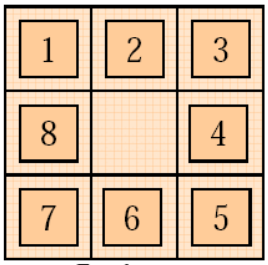
1.5 ¿Cuáles de las siguientes afirmaciones acerca de la búsqueda en línea son ciertas y cuáles son falsas?
- (a) Se intercala una fase de búsqueda (elección de acciones) con una fase de acción/percepción.
- (b) La búsqueda en línea siempre es óptima y completa.
- (c) Es conveniente aplicar la búsqueda con horizonte cuando el espacio de búsqueda es demasiado grande para realizar una *única* búsqueda A^*.
- (d) Una búsqueda en línea es más eficiente cuanto mayor sea su índice competitivo (coste del camino real entre coste del camino óptimo).

Ejercicio 2:

Considere el 8-puzzle cuyo estado inicial y estado meta se muestran en la siguiente figura:



Estado inicial



Estado meta

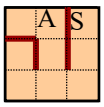
Desarrolle el árbol de búsqueda que expande el algoritmo A^* utilizando las siguientes heurísticas. Evite ciclos generales, indique el orden de expansión de los estados y muestre en cada paso los valores de h^*, g, y f^*. Cuando puede elegir entre varios nodos para ser expandidos, asuma el "peor caso".

a) $h_a^*(n)$ = número de piezas descolocados en n respecto al estado meta

b) $h_c^*(n)$ = suma de las distancias de Manhattan de las piezas en n respecto al estado meta

Ejercicio 3:

Considere el problema el laberinto que se presenta en la siguiente figura.

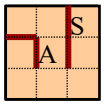


El agente A tiene el objetivo de encontrar la salida S. Las únicas acciones de las que el agente dispone son los movimientos (derecha, arriba, abajo, e izquierda) del cuadrado en el que se encuentra el agente en un momento dado a un cuadrado adyacente. Sin embargo, cada una de estas acciones sólo es posible si en la dirección correspondiente no existe una barrera ni se saldría del tablero. Cada acción tiene un **coste de una unidad**. De antemano, el agente **conoce el mapa** del laberinto **y la posición de la salida**, pero **no las posiciones de las barreras**. Además el agente es capaz de identificar en cada momento su propia posición en el mapa.

a) Define una función heurística h^* para este problema. ¿Es su función h^* optimista y/o consistente? ¡Justifique brevemente su propuesta!

b) Suponiendo la posición inicial del agente que se indica en la figura arriba, aplique la *búsqueda A** con su función heurística h^* a este problema. Desarrolle el árbol de búsqueda suponiendo que **no se evitan estados repetidos**. Indique el orden en el que se expanden los nodos, así como los valores de g, h^* y f^* para cada nodo del árbol de búsqueda. **(Si al expandir un nodo hay que elegir aleatoriamente entre varios, expanda preferiblemente primero un nodo que está más cerca de un nodo meta.)**

c) Suponga el siguiente estado inicial:



Suponga que el agente **no tiene ninguna información heurística** inicial respecto a las distancias de las distintas posiciones en el tablero hacía la salida. Aplique la *búsqueda A** con el aprendizaje de la función heurística h^* a este problema. Desarrolle el árbol de búsqueda suponiendo que **no se evitan estados repetidos**. Indique el orden en el que se expanden los nodos, así como los valores de g, h^* y f^* para cada nodo del árbol de búsqueda. Además, anote los valores de la función heurística h* de cada nodo en una tabla **de tal modo que se aprecian los cambios de estos valores** a lo largo de la ejecución del algoritmo. **(Si al expandir un nodo hay que elegir aleatoriamente entre varios, expanda preferiblemente primero el nodo que está más cerca de un nodo meta.)**

Ejercicio 4:

En juego de los "cuadrados latinos" se parte de un tablero 3 × 3 vacío. En cada posición vamos colocamos números del 1 al 9, ninguno de los cuales puede repetirse. El objetivo es tener el tablero completo, es decir, un número en cada posición del mismo, y es necesario que el valor de la suma de las filas, columnas y diagonales sea siempre el mismo valor: 15. Un ejemplo en el cual tenemos el tablero completo, se han utilizado todos los números pero no se consigue el objetivo indicado está en la siguiente figura. En este ejemplo sólo una diagonal y la última fila cumplen que la suma de sus números es 15.

1	9	2
7	5	6
8	4	3

a) Defina una representación eficiente para el juego de los cuadrados latinos 3×3, especificando el conjunto de estados, el estado inicial, y las operaciones permitidos en cada estado.

b) Considere el estado inicial que se muestra a continuación donde tenemos seis posiciones ocupadas, tres libres, y sólo una diagonal cumple que la suma de sus números es 15. Cada paso tiene coste uno.

2		4
	5	3
6	1	

Asimismo, considere la siguiente función heurística definida para cualquier estado *n* del tablero:

$h^*(n)$ = el número de filas + el número de columnas + el número de diagonales en el estado *n* que no cumplen que la suma de sus números es 15.

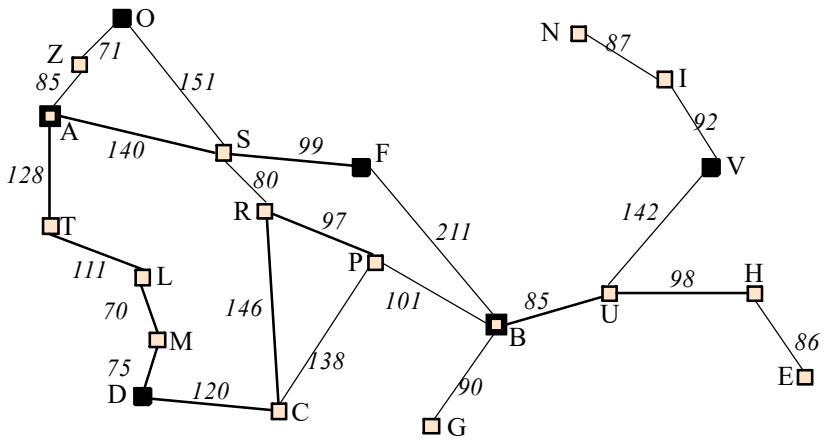
Por ejemplo, el valor de h^* de la configuración de la primera figura es 6, mientras que el valor de h^* para la configuración de la segunda figura es 7.

Desarrolle el árbol de búsqueda que genera el algoritmo A^* para este problema. Indique el orden en el que se expanden los nodos, los valores de g, h^* y f^* para cada nodo del árbol de búsqueda, y la evolución de la lista *abierta*.

c) ¿La función h^* es optimista y/o consistente? ¿El algoritmo A^* encuentra siempre la solución de menor coste? Razone brevemente sus respuestas.

Ejercicio 5:

El grafo que se muestra a continuación representa un esquema de un mapa de carreteras. Los nodos están etiquetados con el nombre de ciudades, y los arcos con la distancia por carretera entre dichas ciudades. La función h^* que se muestra al lado indica la distancia aérea entre cualquier ciudad y la ciudad de *B*. Suponga que nuestro agente dispone de un coche eléctrico con una novedosa batería, que le permite recorrer 320km sin recargar. Sólo puede recargar en ciudades con estación de recarga pública, los cuales se indican en el mapa por los cuadrados rellenos (ciudades *D*, *F*, *O*, y *V*). Inicialmente el agente se encuentra en la ciudad *A* con la batería completamente cargada, y desea trasladarse a *B* por la ruta más corta posible (e.d. sin quedarse tirado por el camino con la batería vacía). Conoce la red de carreteras, la posición de las estaciones de recarga, y las características de su vehículo, por lo que en cada momento sabe a qué ciudades vecinas puede ir dado el estado de carga de su batería. El agente decide aplicar la búsqueda A^*para resolver el problema.



	h^*
A	366
B	0
C	160
D	242
E	161
F	178
G	77
H	151
I	226
L	244
M	241
N	234
O	380
P	98
R	193
S	253
T	329
U	80
V	199
Z	374

a) Represente el problema como espacio de estados, definiendo el conjunto de estados *S*, el estado inicial s_0, los estados meta (a través de la función *meta?*), el coste $c(s_i,s_j)$ para ir de una ciudad s_i a la ciudad vecina s_j, así como la función *expandir* (para dichas definiciones, puede suponer que exista un predicado binario *adyacente* que indica que dos ciudades son vecinas, un predicado unario *carga* que indica si hay una estación de recarga en una ciudad, y una función binaria *d* que proporciona la distancia entre dos ciudades vecinas de acuerdo con el mapa).

b) Desarrolle el árbol de búsqueda que genera el algoritmo A^* (no se filtran ciclos de ningún tipo). Indique el orden en el que se expanden los nodos, los valores de *g*, h^*, y f^* de cada nodo del árbol de búsqueda, así como el estado de la lista *abierta* en cada paso del algoritmo.

c) ¿Para la representación del apartado (a), la función heurística h^* (distancia área entre ciudades) sería optimista? Justifique brevemente su opinión.

1. Contesta a las siguientes preguntas:

   - A cuáles de los siguientes juegos puede aplicarse la búsqueda Minimax?
     (a) Parchís
     (b) Juego de las Damas
     (c) Pacman (o Comecocos)
     (d) Tres-en-Raya
   - Cuáles de las siguientes afirmaciones acerca del algoritmo Minimax son ciertas y cuáles son falsas?
     (a) El algoritmo Minimax realiza una exploración primero en anchura del árbol de juego.
     (b) El algoritmo Minimax maximiza la utilidad mínima que puede conseguir el jugador *max*
     (c) Puede haber estrategias que funcionan mejor que Minimax si el contrincante *no es óptimo*
     (d) Puede haber estrategias que funcionan mejor que Minimax si el contrincante *es óptimo*
     (e) Con la poda alfa-beta se eliminan nodos que nunca serán alcanzados
   - La incorporación de la poda $\alpha - \beta$ en el algoritmo Minimax...
     (a) ...en algunos casos, puede empeorar la calidad de la solución (i.e. es posible que se elija una jugada peor que el Minimax simple).
     (b) ...puede mejorar la velocidad con la que se encuentra una solución (i.e. se elige una jugada más rápidamente que el Minimax simple).
     (c) ...suele mejorar la calidad de la solución (i.e. se elige una mejor jugada que el Minimax simple).
     (d) ...produce siempre la misma solución (i.e. se elige la misma jugada que el Minimax simple).
   - Cuáles de las siguientes afirmaciones acerca del algoritmo ExpectMinimax son ciertas y cuáles son falsas?
     (a) Los nodos azar representan una mala jugada del jugador Min.
     (b) Los nodos azar representan una buena jugada del jugador Min.
     (c) Los nodos azar representan un evento aleatorio.
     (d) Si un nodo azar tiene k sucesores equiprobables entonces, al evaluar el árbol de juego, se pueden sumar las evaluaciones de los sucesores del nodo azar y dividir dicha suma por k.

2. Considera el siguiente juego: Dos contrincantes (*min* y *max*) manejan una ficha en un tablero como en figura 1. *Min* y *max* mueven respectivamente las fichas B y A. Los dos jugadores mueven por turno, empezando por *Max*. Cada jugador debe mover su ficha a un espacio vacío adyacente en *una u otra dirección*. Si el adversario ocupa un espacio adyacente, entonces un jugador puede saltar sobre el adversario al siguiente espacio vacío, si existe. Por ejemplo, si A está sobre 3 y B está sobre 2, entonces *max* puede mover A hacia atrás en el espacio 1. El juego termina cuando un jugador alcanza el extremo opuesto del tablero. Si *max* alcanza el espacio 5 primero, el valor de utilidad para *max* será $+\infty$, si *min* alcanza el espacio 1 primero el valor de utilidad de *max* será $-\infty$.

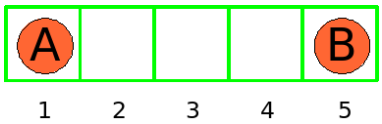


Figura 1: Tablero

   (a) Defina una función de evaluación $e(s)$ para el generico estado s, donde s es una hoja del árbol, y no necesariamente un estado donde uno de los jugadores gana.

   (b) Aplica el algoritmo Minimax con suspensión hasta el nivel 6, empezando con el tablero de figura 1 (nótese que empezando con el nodo raíz con etiqueta *max* de nivel 0, las hojas de nivel 6 también tendrán etiqueta *max*). Especifica los valores calculados para cada nodo y determina la mejor jugada para max.

3. Considera el árbol en figura 2 de un juego de dos personas, donde los cuadrados son nodos *max* y los círculos son nodos *min*.

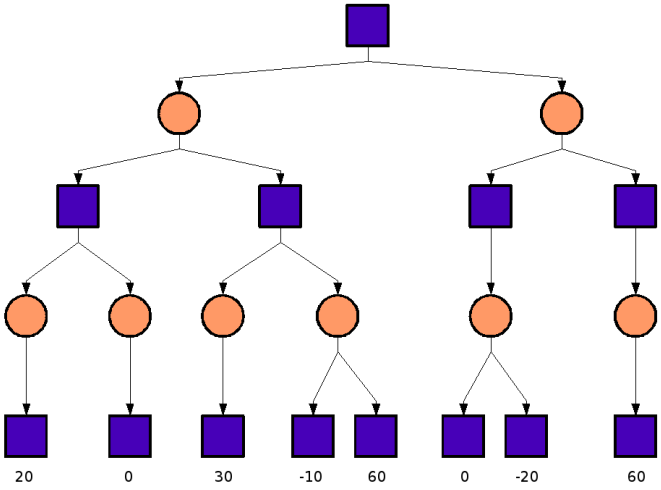


Figura 2: Árbol de juego

(a) Aplique el algoritmo minimax con poda alfa-beta, propaga los valores de evaluación hasta el nodo raíz, marca la mejor jugada para max, y marca todos los subárboles que se podan.

4. Considera el árbol en figura 3 de un juego de dos personas, donde los círculos son nodos *max* y los cuadrados son nodos *min*.

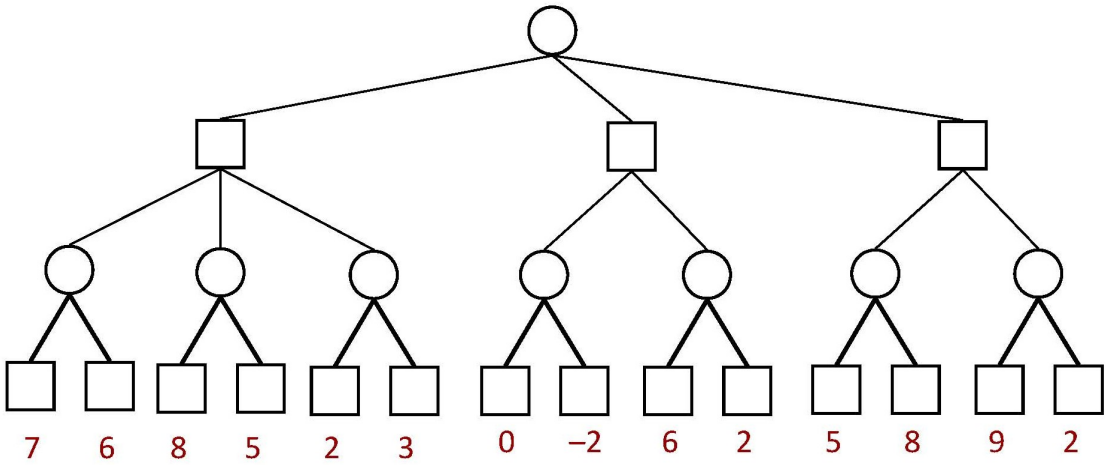


Figura 3: Árbol de juego

(a) Aplique el algoritmo minimax con poda alfa-beta, propaga los valores de evaluación hasta el nodo raíz, marca la mejor jugada para max, y marca todos los subárboles que se podan.

5. Dos conductores, el agente y su contrario, se plantean competir sobre un circuito de ciudades con las siguientes reglas:

   - El recorrido del circuito se hace por tramos, partiendo de la ciudad marcada como Salida (ver figura 4).
   - Los jugadores alternativamente eligen el tramo a recorrer entre aquellos que parten de la ciudad en la que se encuentran. Una vez elegido el tramo, ambos conductores lo recorren y se apunta los kilómetros del tramo el conductor que lo eligió, sin importar quién llega antes a la ciudad de destino. Es decir, si el agente empieza el recorrido y decide ir a B, el agente se apuntará 30 kilómetros. A continuación el contrario debe elegir, partiendo de B, entre ir a C o a D.
   - Un conductor no puede ir a una ciudad en la que haya estado antes, por lo que la competición acaba cuando el conductor al que le toca moverse no puede ir a ninguna ciudad no visitada anteriormente.
   - Gana el conductor que haya acumulado más kilómetros con los tramos recorridos.

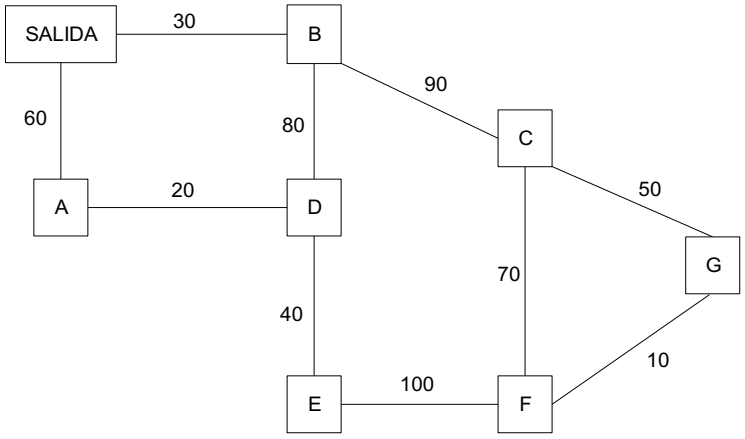


Figura 4: Circuito

(a) Defina una representación eficiente para los estados del juego.

(b) Desarrolle el árbol de búsqueda Minimax hasta ply 4 (dos jugadas del agente y del contrario, respectivamente; empieza el agente). Genere los sucesores de un nodo en orden alfabético, es decir, el primer sucesor del nodo Salida sería A y el segundo B.

(c) Defina una función de evaluación adecuada para los nodos hoja, y propague sus valores a través del árbol. Qué jugada haría el agente?

(d) Qué partes del árbol del apartado (c) no se expandirían si se aplicara la poda ?

6. Considere el siguiente árbol de juego. Evalúe el árbol utilizando el algoritmo $ExpectMinimax$. Las probabilidades de los diferentes nodos son 0,5 para cada acción en los nodos de azar del nivel 3 y los que se indican en el árbol para los nodos de nivel 1.

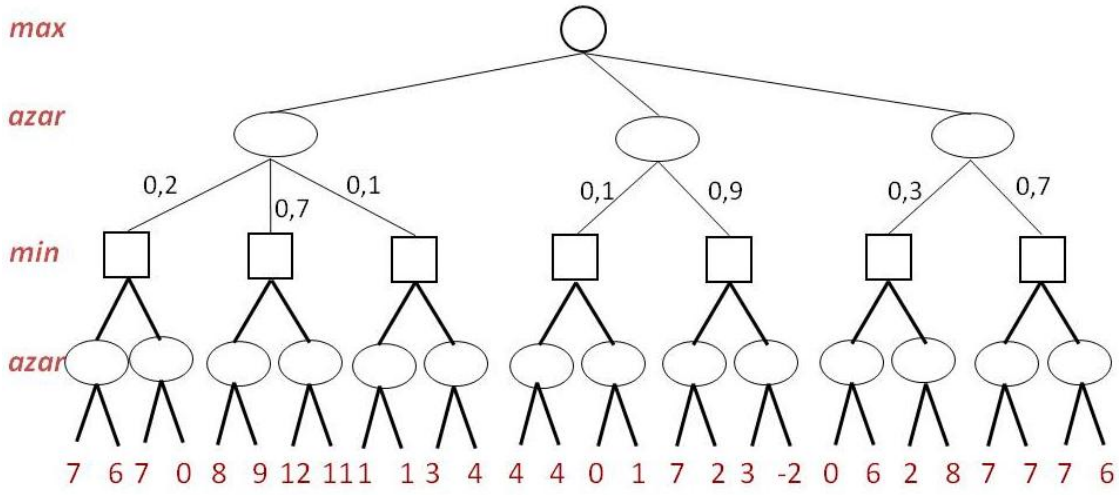


Figura 5: Árbol $ExpectMinimax$

7. Considere un juego para 3 personas (de suma no-nula), en el que no está permitido formar alianzas. Los jugadores se llaman J_1, J_2 y J_3. A diferencia de los juegos bipersonales de suma nula, la función de evaluación en este juego devuelve una tripleta de valores (x_1, x_2, x_3) para cada hoja del árbol de juego que evalúa, tal que x_i es el valor del nodo (estado) para el jugador J_i.

   (a) En el árbol de juego que se muestra a continuación, propague las tripletas de valores desde las hojas hasta la raíz del árbol, anotando en cada nodo interior la tripleta que corresponda (en la figura, en cada nivel del árbol se ve a qué jugador le toca en cada jugada). La mejor jugada del jugador sp1 es la de la izquierda o de la derecha? Justifique brevemente su elección.

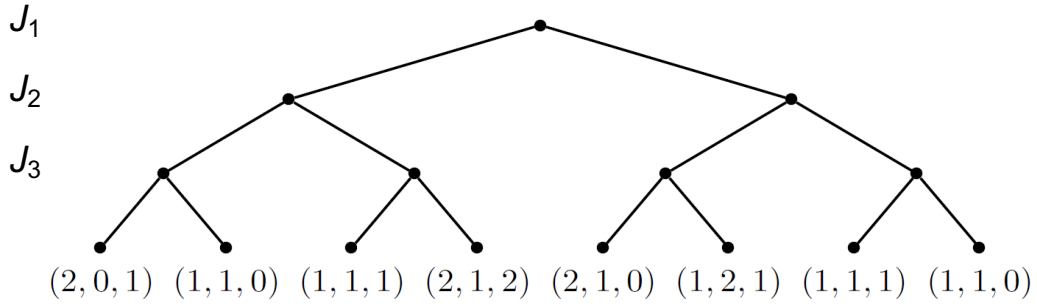


Figura 6: Árbol de juego con 3 jugadores

   (b) Asuma ahora para el ejercicio del apartado anterior que el valor del tercer nodo hoja de la izquierda es (1,0,2) (en vez de (1,1,1)). Qué problema se produce a la hora de propagar las tripletas hacia arriba? Cómo se puede mejorar el procedimiento de propagación para poder abarcar estos casos? Justifique brevemente su opinión

Grupo de Inteligencia Artificial Universidad Rey Juan Carlos

Hoja de Problemas Tema 6
Búsqueda con estados estructurados: Restricciones

Universidad Rey Juan Carlos

Ejercicio 1:

1.1. Si se resuelve un problema de satisfacción de restricciones mediante búsqueda con asignaciones *parciales*, entonces

(a) Un *estado* siempre asigna un valor a todas las variables

(b) Un *operador* elige un valor para una variable *no asignada*

(c) Un *estado meta* es una asignación cualquiera que cumple todas las restricciones

(d) El coste estimado de cada asignación siempre es estrictamente menor que el coste real

(e) Se suele aplicar la heurística de conflictos mínimos

1.2. ¿Cuáles de las siguientes afirmaciones acerca de los problemas de satisfacción de restricciones (CSP) son verdaderas y cuáles son falsas?

(a) Sea $X=\{x_1, x_2, \ldots, x_n\}$ el conjunto de variables de un CSP y $D_1, D_2, \ldots, D_n$ sus respectivos dominios. Un predicado $R(x_i, x_j) \subseteq D_i \times D_j$ es una restricción binaria entre las variables x_i y x_j

(b) El problema del *thrashing* en el algoritmo de *chronological backtracking* se refiere a que evalúa algunas restricciones tantas veces que produce resultados incorrectos.

(c) Sea $X=\{A,B\}$ el conjunto de variables de un CSP, $D(A)=\{1,2,4,5,6,12\}$ y $D(B)=\{blue, green, orange\}$ sus dominios, y $R(A,B)$: $A=|B|$ una restricción binaria sobre las variables. El CSP es arco consistente.

(d) Si al ejecutar el algoritmo de arco consistencia un dominio se queda con un único valor, el CSP es inconsistente, es decir, no tiene solución.

1.3. ¿Cuáles de las siguientes afirmaciones acerca de los problemas de satisfacción de restricciones es (son) verdadera(s)?

(a) El algoritmo de satisfacción de restricciones con salto atrás (*backjumping*) realiza una búsqueda en amplitud en el espacio de asignaciones parciales de valores a variables.

(b) El algoritmo MAC (*Maintaining Arc Consistency*) combina el algoritmo de vuelta atrás cronológica (*cronological backtracking*) y el algoritmo de arco consistencia.

(c) El algoritmo MAC (*Maintaining Arc Consistency*) combina el algoritmo de comprobación hacia delante (*forward checking*) y el algoritmo de arco consistencia.

(d) El algoritmo MAC (*Maintaining Arc Consistency*) intercala la satisfacción y la propagación de restricciones.

(e) El algoritmo de comprobación hacia delante (*forward checking*) realiza una búsqueda A^* en el espacio de asignaciones totales de valores a variables.

Grupo de Inteligencia Artificial Universidad Rey Juan Carlos

Hoja de Problemas Tema 6
Búsqueda con estados estructurados: Restricciones

Universidad Rey Juan Carlos

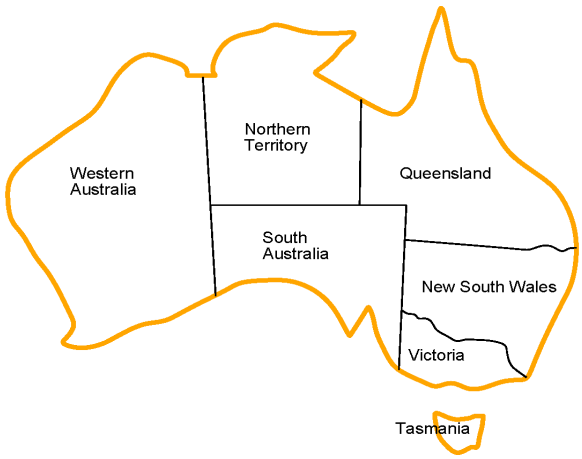
Ejercicio 2:

Se trata de colorar el mapa de los estados de Australia con tres colores (rojo, verde, y amarillo) de tal modo que ningún par de estados adyacentes tengan el mismo color.

a) Modele el problema como CSP binario, indicando las variables, dominios, y restricciones.

b) Represente el CSP en un grafo

c) Dé una solución al CSP



Ejercicio 3:

Un problema *Soduko* se refiere a un tablero de 9x9 casillas, algunas de las cuales contienen inicialmente dígitos de entre 1 y 9. El tablero se divide asimismo en 9 subcuadrículas de 3x3. Se trata de rellenar las restantes casillas con dígitos de entre 1 y 9 tal que

- los dígitos de todas las líneas *horizontales* son diferentes entre ellos, y
- los dígitos de todas las líneas *verticales* son diferentes entre ellos, y
- los dígitos dentro de cada *subcuadrícula* son diferentes entre ellos.

La figura de al lado muestra un ejemplo del estado inicial de un problema *Sodoku*.

	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

Modele un problema Soduko como CSP (no necesariamente binario), indicando las variables, dominios, y restricciones.

Ejercicio 4:

Considere el siguiente CSP con seis variables booleanas (dominio $\{0,1\}$) y cuatro restricciones ternarias:

$$R_1(x_1,x_2,x_6): x_1+x_2+x_6 = 1$$
$$R_2(x_1,x_3,x_4): x_1-x_3+x_4 = 1$$
$$R_3(x_4,x_5,x_6): x_4+x_5-x_6 \geq 1$$
$$R_4(x_2,x_5,x_6): x_2+x_5-x_6 = 0$$

a) Represente el CSP como hipergrafo (cada arco puede conectar a más de 2 nodos)

b) La *codificación de variables ocultas* permite convertir un CSP con restricciones de orden superior en un CSP equivalente con restricciones binarias. Para cada restricción de orden k (k>2)

se introduce una nueva variable cuyo dominio se corresponde con el predicado que define la restricción (e.d., un conjunto de *k*-tuplas). Asimismo, se añaden restricciones binarias entre cada una de las variables que estaban involucradas en la restricción *k*-área original y la nueva variable, indicando que el valor de la primera ha de ser igual al valor del elemento correspondiente en la *k*-tupla que represente el valor de la nueva variable (escribimos $v[i]$ para referirnos al *i*-ésimo elemento de la *k*-tupla *v*).

Aplique dicho procedimiento al hipergrafo del apartado a)

Ejercicio 5:

Aplique el algoritmo MAC al problema de las 4 reinas.

Ejercicio 6:

Dadas las variables **A, B, C, D**, sean

$$D_A = [1,50],\ D_B = [20,40],\ D_C = [1,30],\ D_D = [1,30]$$

los respectivos dominios (todos los D_x son subconjuntos de los números naturales). Las variables están relacionadas por las siguientes restricciones:

R1: **A <= B**

R2: **B+7 < C**

R3: **D <> B**

R4: **D > A**

R5: **D+10 < C**

Se pide:

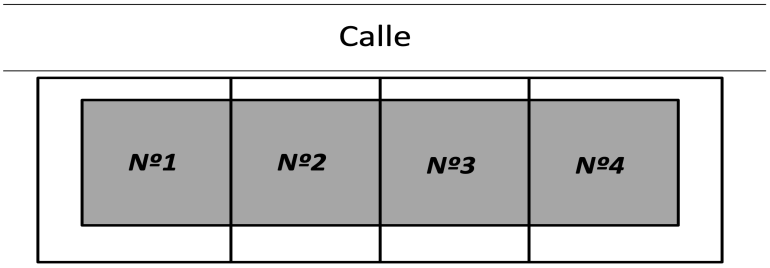
a) Dibujar el grafo del CSP correspondiente;

b) Aplicar el algoritmo de **arco consistencia**, detallando cómo se modifican los dominios cada vez que el algoritmo evalúa una restricción, en una tabla como la que se indica abajo. Las restricciones tienen que ser evaluadas en ambas direcciones, y en el orden dado (e.d. R1-R2-R3-R4-R5-R1-…).

Evaluación	D_A	D_B	D_C	D_D
	[1,50]	[20,40]	[1,30]	[1,30]
R1	[1,40]	[20,40]	[1,30]	[1,30]
…	…	…	…	…

Ejercicio 7:

En una calle viven 4 familias A, B, C y D en casas adosadas cuyos números son 1, 2, 3 y 4. (ver figura)



Familia D vive en una casa con menor número que B. Familia B vive en una casa con mayor número que A. Hay al menos una otra casa entre las casas de B y de C. Familia C vive en una casa que no hace esquina. Obviamente, ninguna familia comparte casa con otra.

a) Represente el problema como CSP, identificando las variables, los dominios de cada variable, y las restricciones entre ellas.

b) Encuentre una solución al problema usando como algoritmo de resolución el *chronological backtracking con forward checking*, evidenciando cómo se modifican los dominios después de asignar cada variable. Suponga que las variables y valores se exploran de menor a mayor. ¿Qué solución se encuentra?

c) Aplique el algoritmo de arco-consistencia a los dominios del apartado (a), indicando en cada paso del algoritmo la restricción tratada (e.d. las variables involucradas) y los valores eliminados (p.e. "Trato restricción $R_{X1,X2}$: Elimino valor v_7 del dominio D_{X1}") ¿Cómo quedan los dominios al final?

Ejercicio 8:

Un representante comercial tiene que visitar 6 clientes (Repsol, Telefónica, ACS, Endesa, Iberdrola, Unión Fenosa) en un día laborable (de 9 a 19). No se puede visitar dos clientes en la misma hora. Además hay que visitar los clientes ACS y Unión Fenosa antes de Endesa. Repsol está fuera de Madrid, mientras que todos los demás están en Plaza de Castilla. Por lo tanto, para ir de Repsol a cualquier otro cliente se tarda dos horas, mientras que para ir de un cliente a otro en Plaza de Castilla no se tarda nada. Florentino Pérez de ACS solo puede recibir el representante de 15 a 17 (luego se va al Bernabeu a ver la Champions).

a) Modelice el problema como CSP, identificando el conjunto de variables, el dominio de cada variable, y el conjunto de restricciones. Dibuje el grafo de restricciones.

b) Aplique el algoritmo de búsqueda con backtracking para encontrar una solución. Evaluar las variables y los valores a asignar en el orden dado (es decir Repsol, Telefónica, …, Unión Fenosa; y 9, 10, 11, …, 19)

Ejercicio 1:

Representar en $\mathcal{ALC}$ el siguiente conocimiento:
TBox:

1. Las mujeres son las personas femeninas
2. Un hombre es una persona que no es una mujer (o los hombres son personas que no son mujeres), y viceversa
3. Los padres son los hombres que crean alguna persona
4. Las madres son las mujeres que crean alguna persona
5. Los padres y las madres son los progenitores
6. Las abuelas son madres de algún progenitor, y viceversa
7. Una esposa es una mujer que tiene un cónyuge (persona), y viceversa
8. Una "madre sin hijas" es una madre cuyas creaciones (p.e. sus hijos) no son mujeres

ABox:

9. Pedro y Rocío son los padres de Juan (Pedro y Juan son hombres, Rocío es mujer)
10. Alex es un hombre o mujer que crea algo

Ejercicio 2:

Representar como lógica de primer orden la base de conocimiento del ejercicio 1.

Ejercicio 3:

Sea la siguiente base de conocimiento en lógica de descripciones $\mathcal{ALC}$.

C = {Gato, Perro, Animal}
R = {tiene}
$\triangle$ = {miko, pipo, silvester, juan, ana}
BC = $<\mathcal{T},\mathcal{A}>$ con:
$\mathcal{T}$ = { Gato $\sqsubseteq$ Animal,
Perro $\sqsubseteq$ Animal,
Gato $\sqsubseteq \neg$Perro }
$\mathcal{A}$ = {Perro(miko),
Perro(pipo),
Gato(silvester),
tiene(juan,miko),
tiene(ana,pipo),
tiene(ana,silvester)}

Indicar los conjuntos de instancias pertenecientes a los siguientes conceptos:

1. Animal
2. Perro $\sqcup$ Gato
3. Perro $\sqcap$ Gato
4. $\exists$tiene.Gato
5. $\exists$tiene.Perro
6. $\forall$tiene.Perro
7. $\forall$tiene.Perro $\sqcap \exists$tiene.$\top$

Ejercicio 4:

Construir las jerarquías de subsunción (clasificación) de la siguiente base de conocimiento (T-Box)

Empresa ⊑ ¬Persona,
EmpresaPrivada ⊑ Empresa,
SociedadAnónima ⊑ Empresa,
Empleado ≡ Persona ⊓ ∃trabajaPara.Empresa,
∃trabajaPara.⊤ ⊑ Persona,
Empresario ≡ Persona ⊓ ∃posee.Empresa,
∃posee.⊤ ⊑ Persona

Ejercicio 5:

Construir las jerarquías de subsunción (clasificación) de la base de conocimiento del ejercicio 1

Ejercicio 6:

Dada la siguiente base de conocimiento en lógica de descripciones $\mathcal{ALC}$

C = {Empresa, Persona, EmpresaPrivada, SociedadAnónima, Empleado, Empresario}
R = {trabajaPara, posee}
$\mathcal{T}$ = { Empresa ⊑ ¬Persona,
EmpresaPrivada ⊑ Empresa,
SociedadAnónima ⊑ Empresa,
Empleado ≡ Persona ⊓ ∃trabajaPara.Empresa,
∃trabajaPara.⊤ ⊑ Persona,
Empresario ≡ Persona ⊓ ∃posee.Empresa,
∃posee.⊤ ⊑ Persona
}
$\mathcal{A}$ = { Persona(Ana),
Empresa (IBM),
(Empresa ⊓ SociedadAnónima)(Telefónica),
Persona(Luis)
(Persona ⊓ Empleado)(Marta),
trabajaPara(Ana,IBM),
trabajaPara(Luis,Telefónica),
Posee(Marta,Telefónica),
Posee(Ana,Telefónica) }

a) Indicar los conjuntos de instancias pertenecientes a los siguientes conceptos, suponiendo el dominio △ = {Ana, IBM, Telefónica, Luis, Marta}
   1. Empleado
   2. Persona
   3. ∃trabajaPara.Empresa
   4. ∀trabajaPara.Empresa
   5. Persona ⊓ ∀trabajaPara.Empresa
   6. ∃posee.EmpresaPrivada
   7. ∃posee.SociedadAnonima
b) Traducir la base de conocimiento a lógica de primer orden

Ejercicio 7:

Representar el siguiente conocimiento en lógica de descripciones $\mathcal{ALC}$ y en lógica de primer orden

a) Todos los hombres son personas
b) Un empleado es una persona que trabaja para una empresa
c) Los profesores son empleados que imparten algún curso
d) Los perros comen huesos (entre otras cosas)
e) Un conductor de autobús es una persona que conduce autobuses
f) Pedro es un conductor de autobuses que trabaja para la empresa EMT, conduce el autobús A27 y al menos imparte un curso

Ejercicio 8:

1) Siendo A, B y C nombres de conceptos y r nombre de rol ¿Cuáles de las siguientes afirmaciones son ciertas respecto a lógica de descripciones $\mathcal{ALC}$?
   (a) $A \sqcap \forall r.B \sqcap \neg \bot$ no es un concepto bien formado
   (b) $A \sqcap B \sqcap \bot$ es insatisfacible
   (c) $A \sqsubseteq A \sqcap B$
   (d) $\forall r.C \sqcap \exists r.\bot \equiv \forall r.C \sqcap \exists r.C$

2) Sea Δ={a,b,c,d} un dominio de individuos, C={a,b} y r={<a,a>,<a,b>,<a,c>,<a,d>, <b,d>} ¿Cuál de los siguientes es el conjunto de individuos del concepto $\forall r.(C \sqcap \neg C)$ en lógica de descripciones $\mathcal{ALC}$?
   (a) {}
   (b) {a}
   (c) {a, b}
   (d) {c, d}

3) Siendo C un concepto y r un nombre de rol ¿Cuáles de las siguientes afirmaciones son ciertas respecto a lógica de descripciones $\mathcal{ALC}$?
   (a) $\forall r.C \sqsubseteq \exists r.C$
   (b) $\exists r.C \sqsubseteq \exists r.\top$
   (c) $\exists r.C \sqcap \forall r.C$ es insatisfacible
   (d) $\exists r.C \sqcap \forall r.\neg C$ es satisfacible

Ejercicio 1:

1) ¿Cuál de las siguientes afirmaciones es cierta?
   (a) RDFS (RDF Schema) es un lenguaje de consulta
   (b) En RDFS no se puede especificar el dominio de las propiedades
   (c) SPARQL es un lenguaje para escribir ontologías
   (d) En SPARQL se pueden realizar comparaciones numéricas

2) ¿Cuál de las siguientes afirmaciones es cierta?
   (a) En RDF no se puede expresar la relación estándar *subclase de*
   (b) En RDFS no se puede expresar la relación estándar *subclase de*
   (c) En OWL no se puede expresar la relación estándar *subclase de*
   (d) En OWL sí se puede expresar la relación estándar *subclase de*

3) ¿Cuál de las siguientes afirmaciones es cierta?
   (a) SPARQL es un lenguaje para consultar datos RDF
   (b) En RDF se puede definir una clase $A \equiv C \sqcup D$
   (c) En RDFS se puede definir una clase $A \equiv C \sqcup D$
   (d) En OWL se puede definir una clase $A \equiv C \sqcup D$

Ejercicio 2:

Representar utilizando RDF Schema el siguiente conocimiento:

a) La capital de España es Madrid.
b) Picasso pintó el Guernica.

Puede utilizar un grafo o la notación Turtle. Defina las clases y propiedades que considere oportunas. Todas las URIs creadas tendrán como base *http://ejercicio.com/*. Si lo desea puede utilizar los siguientes prefijos y/o definir otros si es necesario.

@prefix p2: <http://ejercicio.com/>
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>

Ejercicio 3:

En este ejercicio se va a utilizar una herramienta (*Protégé, https://protege.stanford.edu/*) para construir ontologías y para consolidar algunos conceptos de las Lógicas de Descripciones.

NOTA: No es necesario saber utilizar la herramienta *Protege* pero es interesante en este ejercicio analizar las inferencias que se realizan.

Utilizando la herramienta *Protege 5.5* realizar los siguientes pasos:

1. Crear un proyecto nuevo (File → New, aunque al abrir el programa ya crea uno nuevo)

2. Crear la clase *Animal* y las subclases *Tigre*, *Vaca* y *Oveja*
   - *Tigre* $\sqsubseteq$ *Animal*
   - *Vaca* $\sqsubseteq$ *Animal*
   - *Oveja* $\sqsubseteq$ *Animal*

   [Pestaña "Entities" o "Classes", se pueden habilitar estas pestañas en el menú Window – Tabs]

3. Crear la clase *ProductoAnimal* y sus subclases *Leche* y *Huevos*, es decir:
   - *Leche* $\sqsubseteq$ *ProductoAnimal*
   - *Huevos* $\sqsubseteq$ *ProductoAnimal*

4. Definir la propiedad "*come*" con dominio *Animal* y rango *Animal* $\sqcup$ *Planta* $\sqcup$ *ProductoAnimal*
   - Pestaña "Object properties" (en Entities o activándola en el menú Window – Tabs)
   - Tanto en dominio como rango, por defecto se usa la intersección
   - Al añadir un dominio o rango, la pestaña "Class hierarchy" permite seleccionar una clase (p.e. *Animal*). Si se quiere añadir una expresión se hace en "Class expression editor". En este caso, el rango sería: *Animal or Planta or ProductoAnimal*
   - Otra opción de modelado sería crear una clase *Comida* con esas tres subclases

5. Crear instancia (pestaña "Individuals by class") de Tigre (p.e. t1) y Vaca (p.e. v1) e indicar que *t1 come v1* (en "Object property assertions")

6. Expresar: algo que come animales es carnívoro y viceversa (los carnívoros comen animales):
   - *Carnívoro* $\equiv \exists come.Animal$
   - Crear clase *Carnívoro* y añadir una expresión "Equivalent To". La expresión se puede hacer introducir de dos formas:
     a) "Object restriction creator": seleccionar la propiedad *come*, el "Restriction type" = Some (existential) y "Restriction filler" = *Animal*
     b) "Class expresión editor": *come some Animal*

7. Expresar: un vegetariano es algo que no come animales, y viceversa
   - *Vegetariano* $\equiv \forall come.\neg Animal$ (o $\neg(\exists come.Animal)$)
   - La expresión es: *come only (not Animal)*

8. Expresar: Los tigres comen vacas
    - $Tigre \sqsubseteq \exists come.Vaca$

9. INFERENCIA 1: Jerarquía de clases inferida (pestaña "Class hierarchy (inferred)"). Observar qué se deduce y explicar.
    - Activar el razonador si no lo está, menú "Reasoner" y "Start reasoner". A partir de entonces, si se realizan cambios en la ontología hay que pulsar "Synchronize reasoner" para que actualice las inferencias.
    - Si no está activa, en alguna pestaña (p.e. "Classes"), incluir la ventana que aparece en menú "Window – Views – Ontology views – Classification results"

10. INFERENCIA 2: instancias inferidas. Observar en "Classification results" y explicar.

11. Expresar: La vaca es animal vegetariano
    - $Vaca \sqsubseteq Animal \sqcap Vegetariano$ (o simplemente añadir $Vaca \sqsubseteq Vegetariano$)

12. Expresar: Las vacas locas son vacas que comen ovejas
    - $VacaLoca \sqsubseteq Vaca \sqcap \exists come.Oveja$

13. INFERENCIA 3: Sincronizar el razonador. Observar qué se deduce y explicar.

14. INFERENCIA 4: Quitar *Vegetariano* de *Vaca* (11). Sincronizar el razonador. Observar qué se deduce y explicar.

15. Volver a dejar $Vaca \sqsubseteq Animal \sqcap Vegetariano$ y modificar *Vegetariano* (7):
    - $Vegetariano \equiv \forall come.Planta$ (en vez de $\neg Animal$)

16. INFERENCIA 5: Sincronizar el razonador. Observar qué se deduce y explicar.

17. Añadir que *Animal* es disjunto de *Planta*.
    - $Animal \sqsubseteq \neg Planta$
    - Esto se puede hacer añadiendo el axioma o añadiendo la clase en "disjoints"

18. INFERENCIA 6: Sincronizar el razonador. Observar qué se deduce y explicar.

Ejercicio 4:

1. Introducir en *Protege* el siguiente conocimiento (hoja de ejercicios del tema Lógica de Descripciones):

   Mujer ≡ Persona ⊓ Femenino
   Hombre ≡ Persona ⊓ ¬Mujer
   Padre ≡ Hombre ⊓ ∃crea.Persona
   Madre ≡ Mujer ⊓ ∃crea.Persona
   Progenitor ≡ Padre ⊔ Madre
   Abuela ≡ Madre ⊓ ∃crea.Progenitor
   Esposa ≡ Mujer ⊓ ∃tieneConyuge.Hombre
   MadreSinHijas ≡ Madre ⊓ ∀crea.(¬Mujer)

2. Clasificar la taxonomía (jerarquía de clases inferida). Observar el resultado.

3. Introducir algunas instancias, por ejemplo:
   Pedro y Rocío son los padres de Juan (Pedro y Juan son hombres, Rocío es mujer)
   Alex es un hombre o mujer que crea algo (sugerencia: crear una clase equivalente a "hombre o mujer que crea algo")

4. Obtener instancias inferidas

Ejercicio 5:

Realizar las siguientes consultas SPARQL a la dbpedia (http://dbpedia.org/snorql o http://dbpedia.org/sparql):

1. Capitales de Europa, ordenadas (http://dbpedia.org/class/yago/WikicatCapitalsInEurope)

2. Capitales de Europa y su nombre, ordenadas (http://xmlns.com/foaf/0.1/name)

3. Capitales de Europa que no tienen nombre, ordenadas

4. Capitales de Europa, su nombre y su seudónimo, ordenadas (http://xmlns.com/foaf/0.1/nick)

5. Capitales de Europa y su nombre (si lo tienen), ordenadas

6. Capitales de Europa, su nombre (si lo tienen) y su seudónimo (si lo tienen), ordenadas

7. Capitales de Europa y, si tienen ambos, su nombre y su seudónimo, ordenadas

8. Capitales europeas y su nombre, cuyo nombre contiene una "e"

9. Capitales europeas y población de más de 2.000.000 habitantes, ordenados por nº habitantes (de mayor a menor)
   (http://dbpedia.org/ontology/populationTotal)

10. Capitales europeas con más de 1000 km^2 (en m^2)
    (http://dbpedia.org/ontology/areaTotal)

11. Capitales europeas o asiáticas con más de 1000 km^2 (en m^2)
    (http://dbpedia.org/class/yago/WikicatCapitalsInAsia)

12. Capitales europeas con más de 1000 km^2 (en m^2) y capitales de Asia con más de 10 millones de habitantes

13. Capitales europeas y población de más de 2.000.000 habitantes, de países de menos de 500.000 km^2 (en m^2)
    (http://dbpedia.org/ontology/country)
    (http://dbpedia.org/ontology/area)

14. Actores de cine que están casados (propiedad *dbo:spouse*) entre ellos, han participado (propiedad *dbo:starring*) en la misma película, y tal que uno de ellos ha nacido (*dbo:birthPlace*) en un país (instancia de *dbo:Country*) cuya capital (*dbo:capital*) tiene más de 2 millones de habitantes (*dbo:populationTotal*), y el otro ha nacido en una ciudad que tiene un equipo en la NBA (instancia de *yago:WikicatNationalBasketballAssociationTeams*). La localización del equipo se expresa con la propiedad *dbo:location*. El prefijo dbo se define:

    ```
    PREFIX dbo: <http://dbpedia.org/ontology/>
    ```

Ejercicio 1:

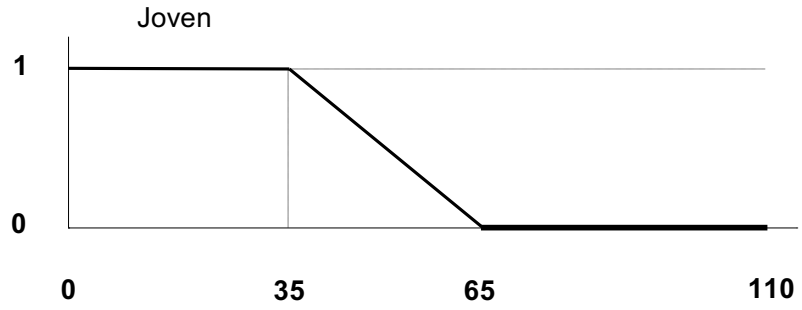
Demostrar que la t-conorma dual del Producto (Prod(x,y) = x*y) es Prod*(x,y) = x + y – x*y. Si es necesario, tomar como negación N(x) = 1 – x.

Ejercicio 2:

Demostrar que la t-conorma dual del Mínimo(x,y) es el Máximo(x,y). Si es necesario, tomar como negación N(x) = 1 – x.

Ejercicio 3:

Dado el conjunto borroso *Joven* cuya función de pertenencia está representada en la siguiente figura:



Si Juan tiene 50 años, calcular los grados de verdad de

a) Juan es joven
b) Juan es viejo
c) Juan no es viejo
d) Juan no es muy joven ni moderadamente viejo
e) Juan es moderadamente joven o es viejo
f) Pintar las gráficas de los conjuntos borrosos *viejo* y *no joven*

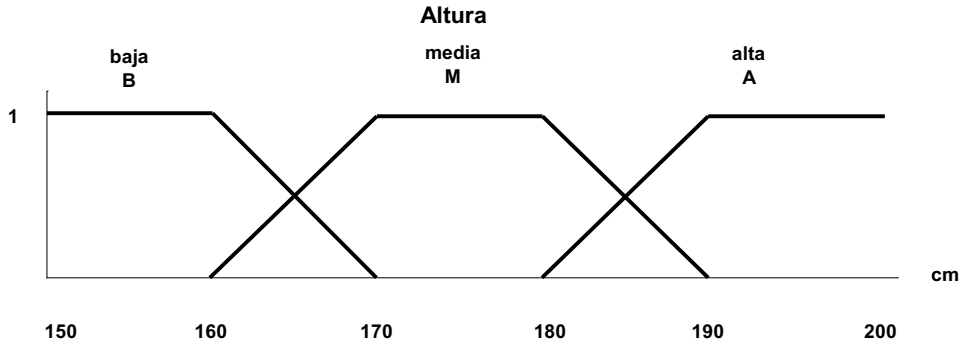
Realizar los cálculos para las tres t-normas y t-conormas estudiadas en la teoría.

Ejercicio 4:

Si Juan mide 185 cm y Pedro mide 168 cm, calcular los grados de verdad de:

a) Juan no es muy alto o Pedro es moderadamente bajo
b) Juan no es alto o Pedro es de media altura

Realizar los cálculos para las tres t-normas y t-conormas estudiadas en la teoría.



Ejercicio 5:

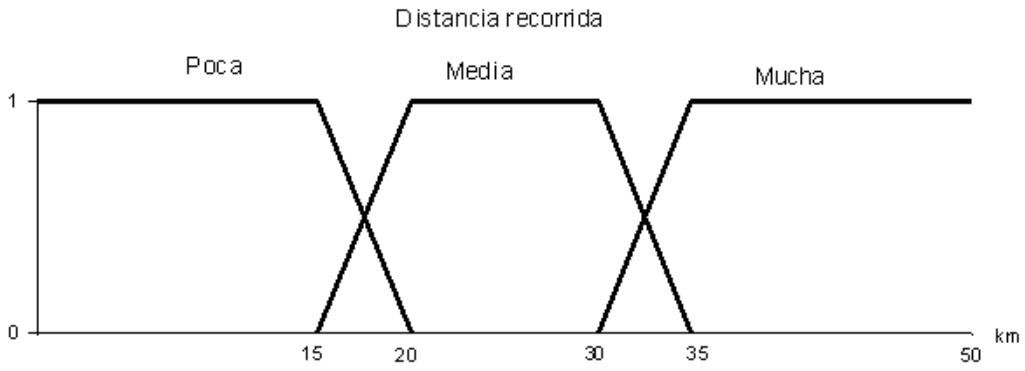
X, *Y* son variables que toman valores en [0,10], ligadas por la regla "Si *X* es Grande entonces *Y* es Pequeña", con $\mu_{Grande}(x) = x/10$.

a) Obtener el grado de verdad de la regla usando la implicación de Reichenbach:
$J(x,y) = 1 - x + x*y$
b) Rellenar la siguiente tabla aplicando el resultado obtenido en el apartado anterior

x	y	G(x) → P(y)
0	0	
0	10	
10	0	
10	10	
9	1	
9	8	
1	8	

Ejercicio 6:

Se representa la variable *distancia media diaria recorrida en el Camino de Santiago* mediante los conjuntos borrosos mostrados en la siguiente figura:



Supóngase que José ha recorrido una media de 31 km al día y Ana 16 km.

Obtener el grado de verdad de las siguientes expresiones:

a) Si José recorre mucha distancia Ana recorre poca
b) Si Ana recorre muy poca distancia José no recorre mucha

En caso de que sea necesario utilizar las siguientes funciones:

- T-norma = T(x,y) = x*y
- T-conorma = S(x,y) = x + y – x*y
- Implicación = J(x,y) = 1 – x + x*y

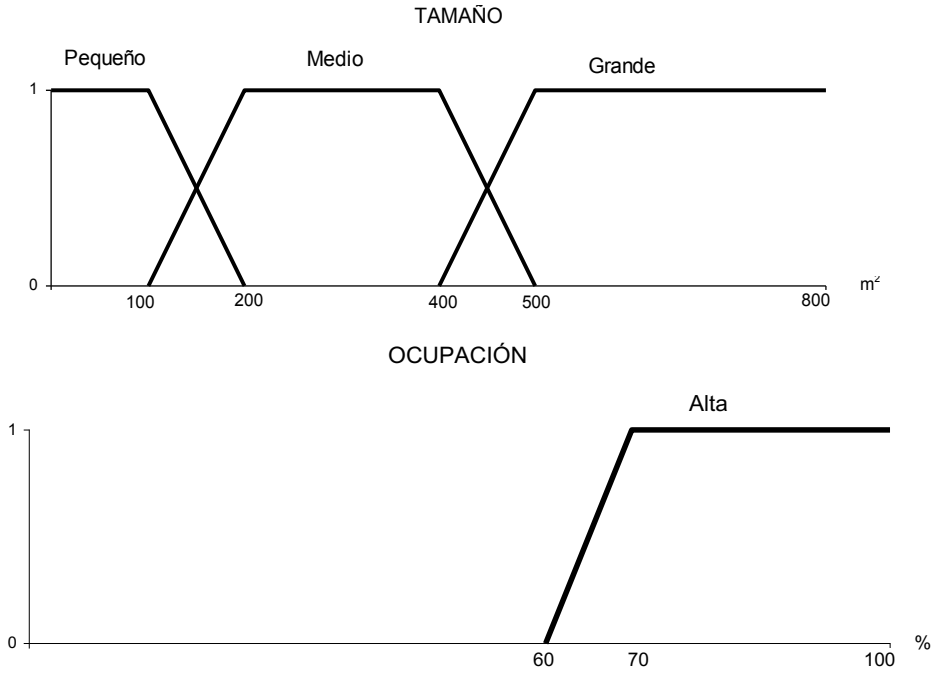
Ejercicio 7:

X, *Y* son variables que toman valores en [0,10], ligadas por la regla "Si *X* es Grande entonces *Y* es Pequeña", con $\mu_{Grande}(x) = x/10$.
Se observa que *X* es muy grande*.

Calcular Y* utilizando la t-norma e implicación de Mamdani (t-norma = implicación= Mín(x,y)).

Ejercicio 8:

Sean las variables lingüísticas representadas en las siguientes figuras:



Obtener el grado de verdad de las siguientes expresiones en lógica borrosa:

a) 170 m^2 es un tamaño pequeño pero 66% es una ocupación muy alta
b) 170 m^2 no es un tamaño medio o 35% es una ocupación baja

En caso de que sea necesario utilizar las funciones de Lukasiewicz:

- T-norma = W(x,y) = Max(0, x+y–1)
- T-conorma = W*(x,y) = Min(1, x+y)
- Implicación = J(x,y) = Min(1,1 – x + y)

Ejercicio 9:

1) ¿Cuáles de las siguientes afirmaciones son verdaderas?
   (a) Las lógicas borrosas son lógicas multivaluadas
   (b) Las lógicas borrosas son extensiones de la lógica clásica
   (c) Para toda t-norma T, $\forall x,y\ T(x,y) \leq$ Mínimo(x,y)
   (d) La unión de conjuntos borrosos se realiza mediante una t-norma

2) ¿Cuáles de las siguientes afirmaciones son verdaderas?
   (a) Para toda t-norma T y t-conorma S, $\forall x,y \in [0,1]\ T(x,y) \geq S(x,y)$
   (b) Para toda t-conorma S, $\forall x,y \in [0,1]\ S(x,y) \leq$ Mín(x,y)
   (c) Para toda t-norma T, $\forall x,y \in [0,1]$ Mín(x,y) $\geq T(x,y)$
   (d) Para toda t-norma T y t-conorma S, $\forall x,y \in [0,1]\ S(x,y) \geq T(x,y)$

3) ¿Cuáles de las siguientes afirmaciones son verdaderas respecto a la lógica borrosa?
   (a) $\forall x\ \mu_{P \cap Q}(x) \leq \mu_P(x)$
   (b) $\forall x\ \mu_{P \cap Q}(x) \geq \mu_P(x)$
   (c) $\forall x\ \mu_{P \cup Q}(x) \leq \mu_P(x)$
   (d) $\forall x\ \mu_{P \cup Q}(x) \geq \mu_P(x)$

	(a)	(b)	(c)	(d)
1				
2				
3				

Ejercicio 1:

Construya el árbol de decisión de acuerdo al algoritmo ID3 para los siguientes datos donde se concluye dar o no un crédito respecto a un cliente bancario de acuerdo a ciertos atributos (nivel de ingresos, tipo de contrato, etc.).

Cliente	**Moroso**	**Antigüedad**	**Ingresos**	**Trabajo Fijo**	Conceder
1	si	> 5	$600 - 1200$	si	no
2	no	< 1	$600 - 1200$	si	si
3	si	$1 - 5$	> 1200	si	no
4	no	> 5	> 1200	no	si
5	no	< 1	> 1200	si	si
6	si	$1 - 5$	$600 - 1200$	si	no
7	no	$1 - 5$	> 1200	si	si
8	no	< 1	< 600	si	no
9	no	> 5	$600 - 1200$	no	no
10	si	$1 - 5$	< 600	no	no

Ejercicio 2:

Construya el árbol de decisión de acuerdo al algoritmo ID3 para los siguientes datos donde se concluye si jugar o no al tenis de acuerdo a las condiciones del tiempo.

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Ejercicio 3:

En una tienda de electrodomésticos quieren evaluar la calidad de los productos que venden. Para ello han hecho una entrevista a los clientes de la tienda cuyos resultados se detallan en la siguiente tabla:

Cliente	**Marca**	**Material**	**Calidad**
1	UFESA	Fibra	Media
2	LG	Plástico	Baja
3	Siemens	Metal	Alta
4	LG	Metal	Alta
5	UFESA	Plástico	Baja
6	Siemens	Fibra	Media
7	Siemens	Metal	Media
8	UFESA	Metal	Media

Usando estos ejemplos, desarrolle un árbol de decisión que permite predicir la calidad de los electrodomésticos. Utiliza el algoritmo presentado en clase, indicando en cada paso la expresión y el valor de la ganancia de cada atributo.

Se puede emplear un "cálculo aproximado" basado en las siguientes igualdades:

$\log_2 1 = 0$
$\log_2 1/2 = -1$
$\log_2 1/3 = -1{,}6$
$\log_2 2/3 = -0{,}6$
$\log_2 1/4 = -2$
$\log_2 3/4 = -0{,}4$
$\log_2 1/5 = -2{,}3$
$\log_2 2/5 = -1{,}3$
$\log_2 3/5 = -0{,}7$
$\log_2 4/5 = -0{,}3$
$\log_2 1/6 = -2{,}6$
$\log_2 5/6 = -0{,}25$
$\log_2 1/7 = -2{,}8$
$\log_2 2/7 = -1{,}8$
$\log_2 3/7 = -1{,}2$
$\log_2 4/7 = -0{,}8$
$\log_2 5/7 = -0{,}5$
$\log_2 6/7 = -0{,}2$
$\log_2 1/8 = -3$
$\log_2 3/8 = -1{,}4$
$\log_2 5/8 = -0{,}7$
$\log_2 7/8 = -0{,}2$

<u>NOTA:</u> A pesar de que no se suele usar la función de activación por umbral en la práctica, en los ejercicios empleamos en todas las neuronas una función de activación por umbral (para simplificar los cálculos). Además, para realizar la actualización de los pesos, se emplea como substituto de la derivada de la función de activación por umbral la función g'(x)=1.

Ejercicio 1:

Diseñe una neurona percetrón con dos entradas binarias y una función de activación por umbral que calcule la función lógica de la coimplicación, es decir, que calcule $x_1 \leftrightarrow x_2$. Determina un conjunto de pesos adecuados para que la neurona calcule esta función (No es necesario aplicar ningún algoritmo concreto.)

Ejercicio 2:

Diseñe una neurona percetrón con tres entradas binarias y una función de activación por umbral cuya salida de activa, si y sólo sí dos o más de las entradas tienen un valor 0. Determina un conjunto de pesos adecuados para que la neurona calcule esta función (No es necesario aplicar ningún algoritmo concreto.)

Ejercicio 3:

Un banco quiere clasificar los clientes potenciales en fiables o no fiables. El banco tiene un dataset de clientes antiguos, con los siguientes atributos:

- Estado civil: {casado/a, soltero/a, divorciado/a}
- Género: {varón, mujer}
- Edad: {[18-30], [31-50], [51-65], [65+] }
- Ingresos: {[10K-25K], [26K-50K], [51K-65K], [66K-100K], [100K+]}

a) Diseñe una red neuronal con neuronas perceptrón simples de activación umbral y de una única capa que se podría entrenar para predecir si un cliente es fiable o no.

b) Inicializa los pesos de la red, inventa un caso de entrenamiento y reproduce cómo el algoritmo de aprendizaje realizaría modificaciones en la red con este ejemplo.

Ejercicio 4:

Una empresa de videojuegos está desarrollando un FPSG (first person shooting game). Para implementar los personajes artificiales del juego, el jefe de proyecto, ex estudiante del curso de IA en la URJC, ha pensado que podría ser interesante e innovador utilizar una red neuronal. Dicha red tendía que implementar el algoritmo de control de los personajes artificiales, usando los siguientes inputs:

- Salud: de 0 (débil) hasta 2 (fuerte)
- Tiene cuchillo: si o no
- Tiene arma: si o no
- Enemigos: número de enemigos en el campo visual

Las acciones que el personaje puede ejecutar son:

- Escapar
- Andar
- Atacar
- Esconderse

a) Diseñe una red neuronal (de una única capa) de perceptrones simples con (función de activación umbral y entradas exclusivamente binarias) que se podría entrenar para implementar el algoritmo de control de los personajes artificiales.

b) Considera el siguiente vector de input:

x=[Salud=2, Tiene cuchillo=no, Tiene arma=si, Enemigos=2]

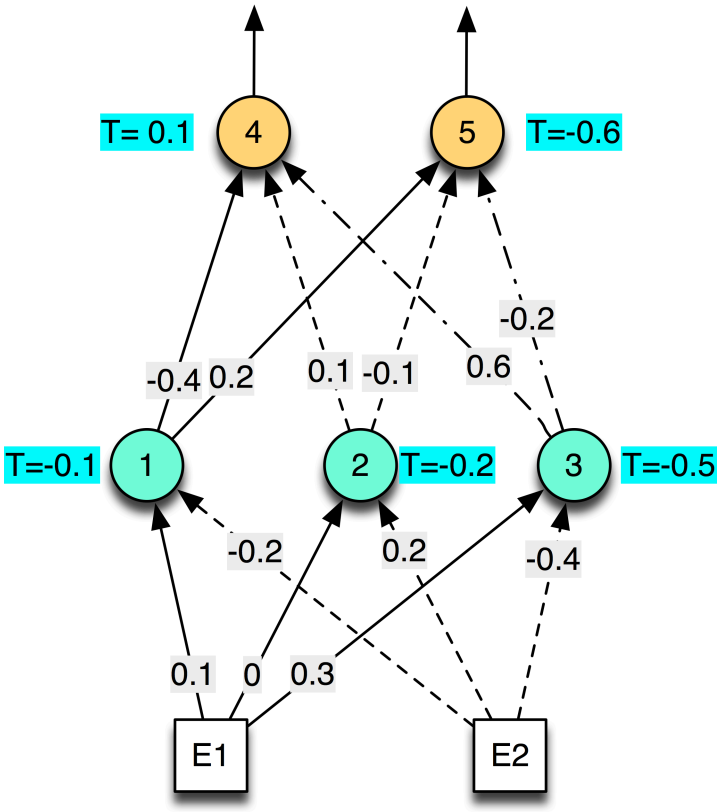
y la salida deseada:

y=Atacar.

Inicializa los pesos y los sesgos de las neuronas de tu red de manera aleatoria en el rango [-0.5,0.5]. Usando el elemento de entrenamiento definido arriba indica como cambiarían los pesos. Por simplicidad, y dado que las neuronas tienen una función de activación por umbral, emplea g'(x)=1 como sustituto de la derivada de la función de activación.

Ejercicio 5:

Sea la red neuronal presentada en la figura. Supón que la función de activación de las neuronas 1,2,3,4 y 5 sea la **función umbral** que devuelve 1 si la suma pesada de las entradas es mayor que 0. Además, en la retropropagación de los errores emplea **g'(x)=1** como sustituto de la derivada de la función de activación. T denota el sesgo de cada neurona.



En este caso, las entradas de la red neuronal no tienen valores binarias. Para neuronas con una función de activación por umbral, la salida, sin embargo, si será binaria.

Dado el siguiente elemento del conjunto de entrenamiento:

(x,y) = (x=[E1=0.6, E2=0.1], y=[4=0, 5=1])

donde el valor la entrada 1 es 0.6, el valor de la entrada 2 es 0.1, la salida de la neurona 4 es 0 y la salida de la neurona 5 es 1.

Sea 0.1 la constante de aprendizaje aplica el algoritmo de retropropagación para actualizar los pesos de la red con este elemento del conjunto de entrenamiento e indicando como cambian los pesos.

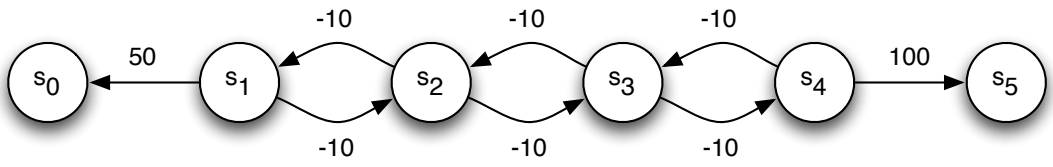
1. Considera el siguiente escenario. Una persona se encuentra delante de una máquina tragaperras y decide si juega o no. Si decide jugar, debe pagar un euro como precio del juego. La máquina tiene 3 botones: A, B y COBRAR. El juego consiste en sumar números a un contador que inicialmente está en 0 y conseguir una suma de 3. Para aumentar el contador, el jugador puede presionar los botones A o B. Al presionar el botón A, la máquina siempre suma 2 al contador. Al presionar el botón B la máquina suma aleatoriamente 1 o 2 al contador (cada uno de los valores con una probabilidad de 0,5).

   El jugador puede aumentar el contador (mediante los botones A y B), mientras el contador tiene un valor menor a 3. Si el valor es mayor o igual a 3, los dos botones A y B ya no funcionan y el juego termina. Si al terminar el valor del contador es igual a 3, se activa el botón COBRAR y, al presionar este botón, la máquina devuelve 2 euros de ganancia. Si el valor del contador es mayor a 3, el botón COBRAR no se activa y el juego termina sin ganancias para el jugador.

   (a) Modeliza este problema como un proceso de decisión de Markov: Especifica los diferentes estados (incluye también el estado en el que el jugador decide si juega o no juega con la máquina), las acciones con sus recompensas y respectivas transiciones. NOTA: Puedes usar un grafo o un conjunto de tablas para especificar el modelo.

   (b) Suponiendo que la constante de descuento γ tenga el valor 1, determina los valores de la política óptima (π^*) así como los valores de las funciones valor-estado (V^*) y valor-estado-acción (Q^*) para todos los estados y pares estado-acción y con respecto a la política óptima. NOTA: para determinar estos valores NO es necesario aplicar el algoritmo visto en clase. ¿Es razonable jugar a este juego?

2. Considera el problema en la siguiente figura.



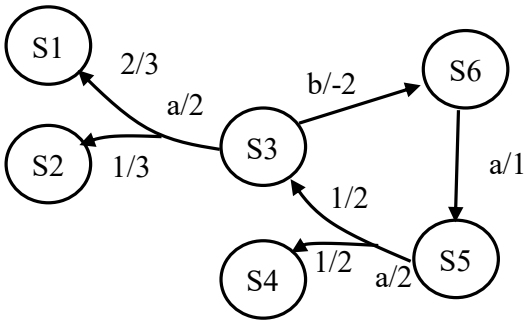
   Si hay una flecha entre dos estados s_i y s_j significa que se puede ejecutar un acción que hace transitar el agente, de manera determinista, del estado s_i al estado s_j, recibiendo la correspondiente *recompensa*. Los estados s_0 y s_5 son estados terminales.

   (a) Calcule los valores $V^*(s)$ para todos los estados, suponiendo que $\gamma = 1$. Calcule también los valores de $Q^*(s, a)$ para todas las parejas estado-acción, y la política óptima en este entorno.

   (b) Calcule los valores $V^*(s)$ para todos los estados, suponiendo que $\gamma = 0{,}8$. Calcule también los valores de $Q^*(s, a)$ para todas las parejas estado-acción, y la política

óptima en este caso. ¿Qué diferencia hay respecto al caso anterior? Argumente su respuesta.

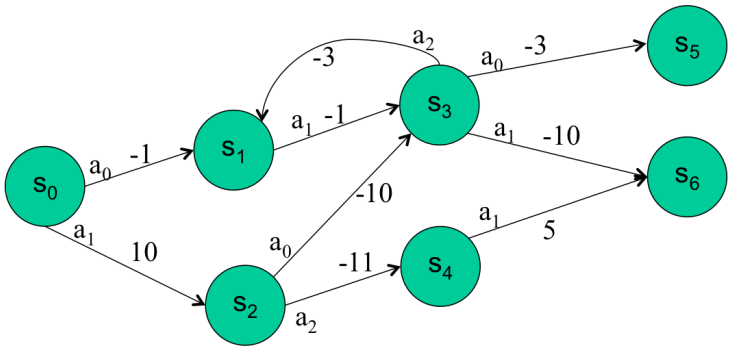
3. Considera el siguiente grafo correspondiente a un problema modelado como Proceso de decisión de Markov. Los nodos representan los grafos y las flechas las acciones (asociados con sus nombres, recompensas y probabilidades de transición).



Calcule los valores de la función Q* (función estado-acción-valor) para las todas las acciones de este problema. Nota: NO es necesario aplicar un algoritmo para calcular los valores (aunque también es una opción válida).

4. El siguiente grafo representa un MDP determinista. Aplicando el algoritmo de iteración de valores para la resolución de MDPs, presenta en una tabla la evolución de los valores de Q^* y determina los valores finales de V^* y π^*. Realiza el algoritmo haste que se hayan obtenido los valores finales de Q^*. Los parámetros a utilizar son los siguientes:

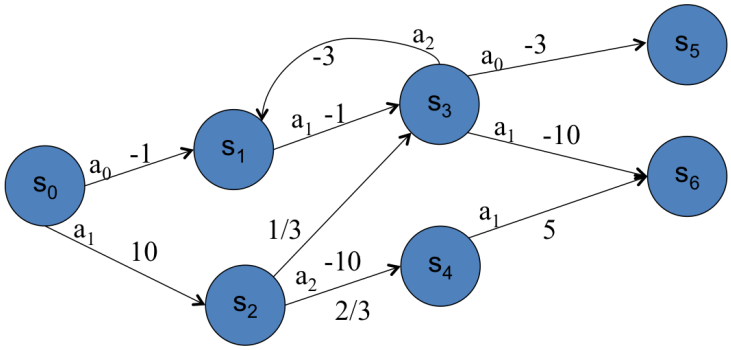
   - valores de Q^* inicializados a 0.
   - Orden de ejecución en cada iteración: estados de s_0 a s_6 y para cada estado, acciones de a_0 a a_2.
   - $\gamma = 1$



5. El siguiente grafo representa un MDP NO determinista. Aplicando el algoritmo de iteración de valores para la resolución de MDPs, presenta en una tabla la evolución de los valores de Q^* y determina los valores finales de V^* y π^*. Realiza el algoritmo haste que se hayan obtenido los valores finales de Q^*. Los parámetros a utilizar son los siguientes:

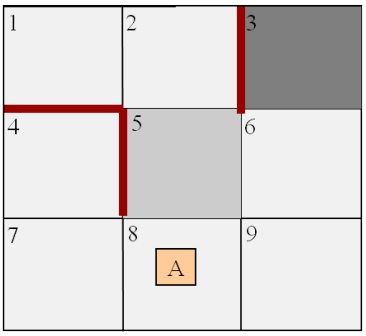
- valores de Q^* inicializados a 0.
- Orden de ejecución en cada iteración: estados de s_0 a s_6 y para cada estado, acciones de a_0 a a_2.
- $\gamma = 1$

NOTA: en comparación con el grafo del ejercicio anterior, en este caso existe una transición no determinista (a_2 en s_2).



1. Cuáles de las siguientes afirmaciones acerca del algoritmo Q-learning son ciertas
   (a) Para garantizar la convergencia de los valores Q a los valores Q^* óptimos, el entorno tiene que ser determinista.
   (b) **Para garantizar la convergencia de los valores Q a los valores Q^* óptimos, el entorno tiene que ser estacionario.**
   (c) Los algoritmos de aprendizaje por refuerzo aprenden la pólitica óptima $\pi^*(s)$ usando un conjunto de parejas $< s, \pi^*(s) >$
   (d) **Los algoritmos de aprendizaje por refuerzo en su versión básica necesitan almacenar un valor $Q(s, a)$ para toda pareja estado-acción** $< s, a >$

2. Considera el problema logístico presentado en la siguiente figura:



   El objetivo del agente (A) es moverse lo más rápido posible a la casilla más oscura (casilla 3). Para ello debe usar una política que le permite encontrar las acciones óptimas para este fin. El agente se puede mover de casilla a casilla (pero no en diagonal) y no puede atravesar las barreras (lineas gordas). La casilla 5 tiene una peculiaridad: por ella el agente se pude desplazar mucho más rápido.

   Para aprender la política óptima del robot se ha decidido usar el algoritmo Q learning. Para ello, se ha asignado las siguientes recompensas a las acciones del agente:

   - +50 a cualquier movimiento que le hace entrar a la casilla 3;
   - -1 a cualquier movimiento que le hace entrar en la casilla 5;
   - -2 a cualquier otro movimiento.

   Se supone que una vez llegado a la casilla 3, el agente no realiza ninguna acción más.

   (a) ¿Que metodo de actualización de la función Q se debe emplear en este problema y por qué?
   (b) Simula el funcionamiento del algoritmo Q learning para este problema pasa o paso teniendo en cuenta lo siguiente:
      - Los valores de la función Q se inicializan todos al valor 0;

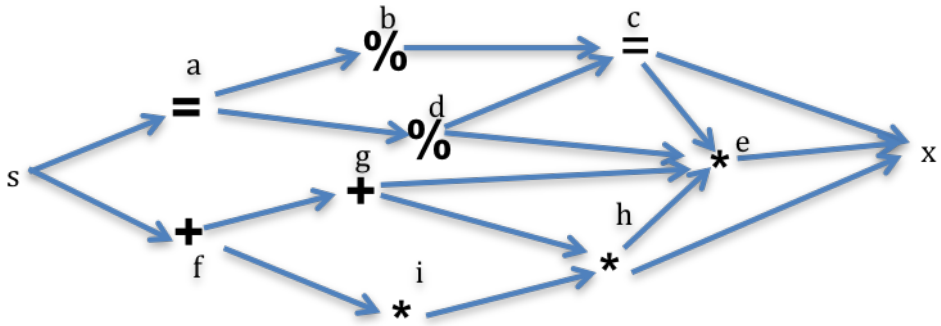
- El agente emplea una política greedy. Es decir, en cada estado elige siempre la acción con el mayor valor Q.
- el factor de descuento (gamma) es 1.
- Si varias acciones tienen en un estado el mismo mejor valor Q, el agente elige la accion según el siguiente órden: izquierda (L), arriba (U) derecha (R), abajo (D)

Representa el camino que el agente recorre y los cambios en los valores de la función Q.

(c) Repite el mismo problema anterior 2 veces, pero con los valores de la función Q ya aprendidos.

(d) Calcula los valores de las recompensas accumuladas (V^*)para cada uno de los estados (casillas) si el agente siguiese la politica óptima.

3. En un videojuego, un jugador tiene que recorrer una especie de laberinto desde un punto de salida (s) a un punto de destino (x) tal y como se presenta la siguiente figura:



El jugador puede desplazarse de un punto de cruze a otro por los caminos existentes (flechas). El jugador sólo puede irse hacía adelante (hacía la derecha) y no puede volver por el camino por el que haya venido. Cuando el jugador entra en un nuevo punto de cruze ocurren los siguientes eventos según los símbolos que marcan el cruze:

- +: el jugador gana 2 monedas
- =: el jugador gana 9 monedas
- *: el jugador gana 1 moneda
- %: l jugador pierde una vida

En la salida (x) el jugador no gana monedas ni pierde ninguna vida y se termina el juego.

Para identificar los estados del problema (puntos de cruze) se emplean las letras minúsculas de la figura. Las acciones se identifican por a_1 y a_2, de tal forma que en cada estado el camino más hacia arriba corresponde a a_1 y el otro (si existiese) a a_2. Es decir, tomar el camino de"d" a "c" corresponde a realizar la acción a_1 en el estado

"d", mientras el camino de "d" a "e" corresponde a realizar la acción a_2 en el estado "d".

Se supone que **el jugador no conoce este escenario a priori** y su objetivo es aprender el camino más lucrativo en sucesivos instancias del juego. El camino ma?s lucrativo es aquél en el que el jugador obtiene el mayor número de monedas y pierde menos vidas. Para establecer una relación entre vidas y monedas, el jugador estima que cada vida equivale a 10 monedas.

Apliqué el algoritmo de Q-learning a este problema con los siguientes parámetros:

- Los valores de la función Q se inicializan todos a 0
- Emplea el siguiente factor de descuento: $\gamma = 1$
- El jugador elige sus acciones con una política greedy (ávara)
- Si en un momento dado no tiene ningún criterio mejor, prefiere siempre las acciones (caminos) más arriba.

(a) Ejecuta el algoritmo Q-learning para una instancia del problema, indicando las acciones que el jugador realiza (los puntos por los que pasa) y la evolución de los valores de la función Q.

(b) Dado los valores de la función Q aprendidos, ejecuta el algoritmo de nuevo para una segunda y una tercera instancia del juego. Nuevamente indica las acciones que el jugador realiza y la evolución de los valores de la función Q.

(c) Indica si el jugador aprendería el camino óptimo si se ejecutase el algoritmo de forma iterativa en nuevas instancias del juego. Argumenta, por qué encuentra (no encuentra) el camino óptimo.

4. Considera el siguiente problema con un único estado. Hay una tragaperra con tres posibles botones (A, B y C), cada uno de los cuales devuelve una *recompensa* de 4, 5 y 3 respectivamente. Queremos aprender a seleccionar el botón que más *recompensa* genera, utilizando Q-learning con $\gamma = 0$ (ya que es un problema con un único estado) para entornos deterministas. Supongamos que al principio los valores $Q(a)$ son todos 0.

   (a) Compara la suma de las *recompensas* obtenidas así como los valores de $Q(a)$ aprendidos después de 5 interacciones con la tragaperra, usando la política de selección *greedy*, ϵ-*greedy* con $\epsilon = 0{,}01$ y ϵ-*greedy* con $\epsilon = 0{,}1$. Si la selección de la acción a ejecutar se basa en el valor $Q(a)$ y hay más de una acción con el mismo valor de $Q(a)$, se elije en order alfabético. Para simular la aleatoriedad de las políticas ϵ-*greedy* utilice la siguiente idea: Antes de elejir una acción, el agente calcula un número aleatorio entre 0 y 1. Si este número es mayor que $1 - \epsilon$ elige la acción con el mayor valor Q. En caso contrario elige una de las otras acciones (la primera por órden alfabética). Para simular este procedimiento, utilice la siguiente secuencia de números aleatorios $\{0{,}3; 0{,}95; 0{,}4; 0{,}999; 0{,}2\}$.

   (b) ¿Que pasaría si se inicializasen los valores $Q(a)$ todos a 10, y se aplicase la política *greedy*?

5. Aplica el algoritmo de aprendizaje Q-learning al problema presentado en el siguiente grafo. Presenta la evolución de los valores de Q en una tabla. Repite 3 episodios: En los dos primeros episodios el agente elige siempre la acción de mayor valor Q (y la acción b en caso de empate) En el tercer episodio, ocurre lo mismo, salvo en el estado inicial s_0, en el que el agente elige la acción b para explorar. Respecto a la acción b en s_2, supón que la primera vez que el agente ejecuta esta acción, se transita al estado s_3, la segunda vez al estado s_4, la tercera de nuevo a s_3 y así sucesivamente. Utiliza los siguientes parámetros:

   - los valores de $Q(s, a)$ son inicializados en 0
   - el estado inicial en los 3 episiodios es el estado s_0
   - $\gamma = 1$ y $\alpha = 0,1$