APELLIDOS:	PL:
NOMBRE:	DNI:

ESCUELA DE INGENIERÍA INGORMÁTICA

SISTEMAS INTELIGENTES

Examen Final de Teoría. Jueves 11 de enero de 2018.

1.- i) Responde razonadamente a las siguientes cuestiones:

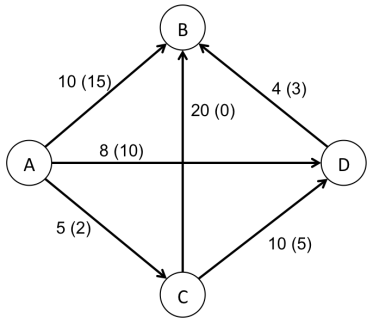
a) **[0,5 puntos]** Dados dos heurísticos admisibles h1 y h2, considérese el heurístico h_{max} definido como $h_{max}(n) = \max(h1(n), h2(n))$ para todo estado n. ¿Es h_{max} admisible?

Los heurísticos h1 y h2 son admisibles, lo que significa que $h1(n) <= h^*(n)$ y $h2(n) <= h^*(n)$ para todo estado n. El heurístico h_{max} se define como $h_{max}(n) = \max(h1(n), h2(n))$. Dado un estado cualquiera n, $h_{max}(n)$ será igual a h1(n) o a h2(n), por lo que $h_{max}(n) <= h^*(n)$. Por lo tanto, **h_{max} es un heurístico admisible**.

b) **[0,5 puntos]** En un paso intermedio de la ejecución de un algoritmo A* que emplea un heurístico admisible h, ABIERTA contiene un nodo n tal que, en ese momento, $f(n) > C^*$. ¿Podemos asegurar que el nodo n no se va a expandir?

No podemos asegurar que el nodo n no se va a expandir. El motivo es que n se podría rectificar en etapas posteriores de la búsqueda, haciendo que g(n) se reduzca y que $f(n) = g(n) + h(n) <= C^*$ por lo que el nodo n se podría expandir, al ser h un heurístico admisible.

ii) Problema (Rutas con peajes). Un conductor necesita realizar un viaje entre una localidad de partida y una localidad de destino. En este contexto, las carreteras, además de una longitud, tienen asociado un peaje (mayor o igual que 0) que es necesario abonar para pasar por ellas. El conductor desea calcular la ruta más corta posible entre las dos localidades, pero sujeto a un presupuesto máximo disponible P para el pago de los peajes. En el digrafo de la figura los nodos representan localidades y los arcos carreteras. Cada arco está etiquetado de la forma *longitud (peaje)*, es decir el peaje se representa entre paréntesis.



a) **[0,75 puntos]** Modela un espacio de búsqueda para el problema: la representación de los estados, la identificación del estado inicial y de los estados objetivo, las reglas de generación de estados sucesores y sus costes.

[En el problema se plantea calcular la ruta más corta dese una ciudad de partida a otra de destino, con la restricción adicional de no sobrepasar un presupuesto máximo para el pago de peajes asociados a las carreteras. De este modo, la información que necesitamos mantener en cada estado es la ciudad en la que se encuentra el conductor y el presupuesto restante para peajes. El resto de información (conexiones entre ciudades, longitud de las carreteras y peajes) es global al problema, y no es necesario almacenarla en cada estado.]

La **representación de los estados** puede ser un par (c, p), donde c es la ciudad en la que se encuentra el conductor y p es el presupuesto restante para pagar peajes. Serían estados válidos aquellos en los que p >= 0. El **estado inicial** vendría dado por (c_{ini}, P), donde c_{ini} es la ciudad de partida y P es el presupuesto máximo inicial; y los **estados objetivo** serían de la forma (c_{dest}, p), donde c_{dest} es la ciudad de destino y p >=0.

La **generación de estados sucesores** se haría del siguiente modo: Para un estado (c, p) se generaría un estado sucesor (c', p') por cada carretera x que vaya de c a otra ciudad c', y p' se calcularía como p' = p - peaje(x). El nuevo estado sería válido siempre que p' >= 0 y el **coste asociado a la regla** sería la longitud de la carretera.

b) **[0,5 puntos]** Define un heurístico admisible no trivial para este problema, justificando su admisibilidad.

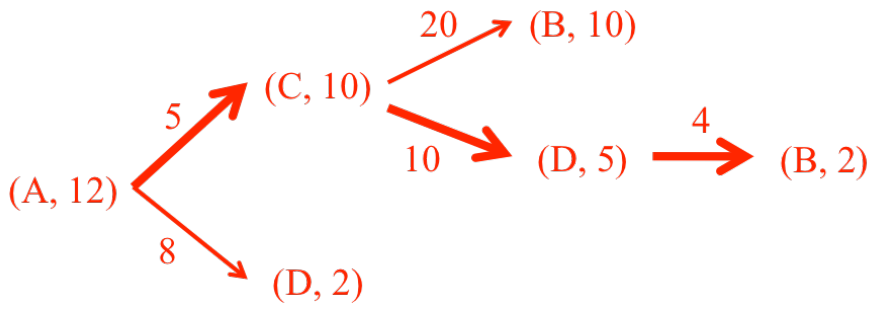
Para definir un heurístico podemos emplear el método de la **relajación del problema**. Si eliminamos la restricción asociada al presupuesto para pagar peajes, el problema relajado sería el de calcular la ruta más corta desde la ciudad del estado a la ciudad de destino (ignorando los peajes). Este problema se puede resolver en tiempo polinomial, por ejemplo mediante el algoritmo de Dijkstra. Así, para un estado cualquiera n = (c, p), h(n) se definiría como:

h(n) = coste de la ruta más corta desde c hasta la ciudad de destino.

Al obtenerse como el coste óptimo de un problema relajado, este heurístico es monótono y por lo tanto admisible.

c) **[0,75 puntos]** Representa el espacio de búsqueda para la instancia representada en la figura, considerando que la localidad de partida es la A y que la de destino es la B, con un presupuesto máximo para peajes de P = 12. Indica una solución óptima y el valor de C*. Indica el valor del heurístico diseñado en el apartado anterior para el estado inicial.

Representación del espacio de búsqueda:



Nótese que el estado inicial es (A,12) y hay dos estados objetivo: (B,10) y (B,2). La **solución óptima** es el camino marcado en el grafo (A-C-D-B), con un coste **C* = 19**. El valor del heurístico definido en el apartado anterior para el estado inicial es **h(ini) = 10** (calculado como el coste de la ruta más corta de A a B en el digrafo del enunciado, ignorando los peajes).

2.- [0,5 puntos] Para cada una de las cinco siguientes afirmaciones sobre algoritmos genéticos, indica si es verdadera o falsa. No es necesario razonar las respuestas. Cada respuesta suma 0,1 si es correcta, resta 0,1 si es incorrecta, y ni suma ni resta si está en blanco.

a) El operador de selección se encarga de elegir qué cromosomas van a formar la siguiente generación del algoritmo genético.
b) Sea un cromosoma (hijo) H generado mediante la aplicación de un operador de cruce a dos cromosomas (padres) P1 y P2. Sería posible que H tuviese peor fitness que P1 y P2.
c) Podemos garantizar que, si dejamos a un algoritmo genético ejecutarse durante suficiente tiempo, acabará encontrando la solución óptima del problema.
d) Habitualmente, la probabilidad de cruce es mayor que la probabilidad de mutación.
e) Sean A y B dos algoritmos genéticos diferentes. Si en un determinado problema, A obtiene una solución mejor que B utilizando un tiempo de ejecución mayor que el utilizado por B, podemos asegurar que A es mejor que B.

Solución:

a) Falso (eso sería el operador de reemplazamiento, el de selección lo que hace es elegir qué cromosomas se van a cruzar entre sí)
b) Verdadero (el hijo puede ser mejor, igual, o peor que los padres, depende de la suerte)
c) Falso (incluso aunque exista algún tipo de mutación, es posible, y de hecho habitual, que la población se estanque y nunca se llegue a la solución óptima le dejemos el tiempo que le dejemos)
d) Verdadero (la de cruce suele ser superior a 0,7, y la de mutación suele ser inferior a 0,3)
e) Falso (para empezar deberíamos probar a ver qué pasa dejándoles el mismo tiempo de ejecución)

3.- a) [1 punto] Representar el siguiente enunciado en lógica de predicados:

"En general los elefantes son grises, pero hay algunos elefantes raros que son de color verde o rosa. Goomy es uno de estos elefantes raros y no es verde."

Y luego responder a la siguiente pregunta, utilizando resolución:

"¿De qué color es Goomy?"

RESPUESTA:

El enunciado en lógica de predicados se puede representar así:

$\forall X(elefante(X) \land \sim raro(X) \Rightarrow color(X,gris))$
$\forall X(elefante(X) \land raro(X) \Rightarrow color(X,rosa) \lor color(X,verde))$
$elefante(goomy) \land raro(goomy) \land \sim color(goomy,verde)$

En forma clausal:

1. $elefante(X) \rightarrow raro(X),color(X,gris)$
2. $elefante(X), raro(X) \rightarrow color(X,rosa),color(X,verde)$
3. $\rightarrow elefante(goomy)$
4. $\rightarrow raro(goomy)$
5. $color(goomy,verde) \rightarrow$

La pregunta:

$\exists X\ (color(goomy,X))$

Negada y en forma clausal:

6. $color(goomy,X) \rightarrow$

Al aplicar resolución tenemos:

7.	$raro(goomy) \rightarrow color(goomy,rosa),color(goomy,verde)$	de 2. y 3. (umg: X/goomy)
8.	$\rightarrow color(goomy,rosa),color(goomy,verde)$	de 7. y 4. (umg: ∅)
9.	$\rightarrow color(goomy,rosa)$	de 5. y 8. (umg: ∅)
10.	$\rightarrow$	de 6. y 9. (umg: X/rosa)

b) [1 punto] La siguiente regla CLIPS modela los movimientos posibles entre dos habitaciones, en el problema de Monkeys&Bannanas, de forma que no es posible moverse a una habitación en la que el Monkey ya estuvo:

```
(defrule move-loc1-loc2
        ?lm <- (location monkey ?loc1)
        (door ?loc1 ?loc2)
        (not (was-location monkey ?loc2))
=>
        (assert (location monkey ?loc2))
        (retract ?lm)
        (assert (was-location monkey ?loc1))
        (printout t "move from " ?loc1 " to " ?loc2 crlf)
)
```

Lo que se pide en esta pregunta es diseñar una nueva regla que se ejecute solamente si la anterior no se puede ejecutar, al no haber habitaciones adyacentes no visitadas, de manera que el Monkey se pueda mover a una habitación adyacente ya visitada. Si se aplica esta regla, debe quedar registrado en la base de hechos que el Monkey visitó la habitación más de una vez.

RESPUESTA:

```
(defrule move-loc1-loc2-1
        (declare (salience -1))
        ?lm <- (location monkey ?loc1)
        (door ?loc1 ?loc2)
=>
        (assert (location monkey ?loc2))
        (retract ?lm)
        (assert (was-location monkey ?loc1))
        (assert (was-location monkey ?loc2 more than once!!))
        (printout t "move from " ?loc1 " to " ?loc2 crlf)
)
```

4.- [3,25 puntos]

4.1. **[0,75 puntos]** Describe el método K-NN (K-vecinos). Además de los pasos fundamentales del algoritmo, debes explicar si es o no necesario normalizar el conjunto de datos y por qué.

La descripción del método debe contener al menos las siguientes características para que se de por válido:

- *En tiempo de entrenamiento el método no hace nada.*
- *En tiempo de test, se calcula la distancia de cada ejemplo a clasificar a todos los ejemplos de entrenamiento, quedándonos con los K más próximos.*
- *Se asigna como clase, la clase mayoritaria,*
- *Se puede utilizar cualquier distancia, no solo la euclidea.*
- *Los K vecinos pueden no tener todos la misma importancia, ya que se pueden utilizar distintas estrategias de ponderación de vecinos.*
- *Funciona bien en conjuntos caracterizados por no muchos atributos.*
- *Es sensible al ruido.*
- *ES NECESARIO normalizar, para evitar que u nos atributos influyan más que otros en el cálculo de las distancia.*

4.2. **[0,75 puntos]** Dado el siguiente conjunto de entrenamiento {(1,+),(2,+),(4,-), (5,-),(6,+)} al que se le ha aplicado SVM usando como kernel $K(x,y)=(xy+1)^2$, obteniendo $\alpha_1=1$, $\alpha_2=3$, $\alpha_3=0$, $\alpha_4=4$ $\alpha_5=0$. ¿Cuantos vectores soporte tiene cada clase? , ¿cuáles son los vectores soporte?¿Es un clasificador lineal? ¿Por qué?

Los vectores soporte son aquellos para los que el valor de α es distinto de 0, esto es, los ejemplos (1,+) y (2,+) para la clas "+" y el ejemplo (4,-) para la clase "-". Es un clasificador no lineal, ya que el kernel no lo es.

4.3. **[0,75 puntos]** Describe el fenómeno del sobreajuste en un árbol de decisión y explica como detectarlo.

El sobreajuste se produce cuando el árbol de decisión construido se ajusta en exceso a los datos de entrenamiento, clasificando mal ejemplos no vistos.
Para detectar si un árbol está sobreajustado hay que probar si al podar, o al realizar una poda más agresiva, el rendimiento del árbol al clasificar ejemplos de test, no vistos, mejora. Si se realiza el mismo procedimiento utilizando como test el conjunto de entrenamiento, si hay sobreajuste, el rendimiento del árbol empeorará cuando se poda.

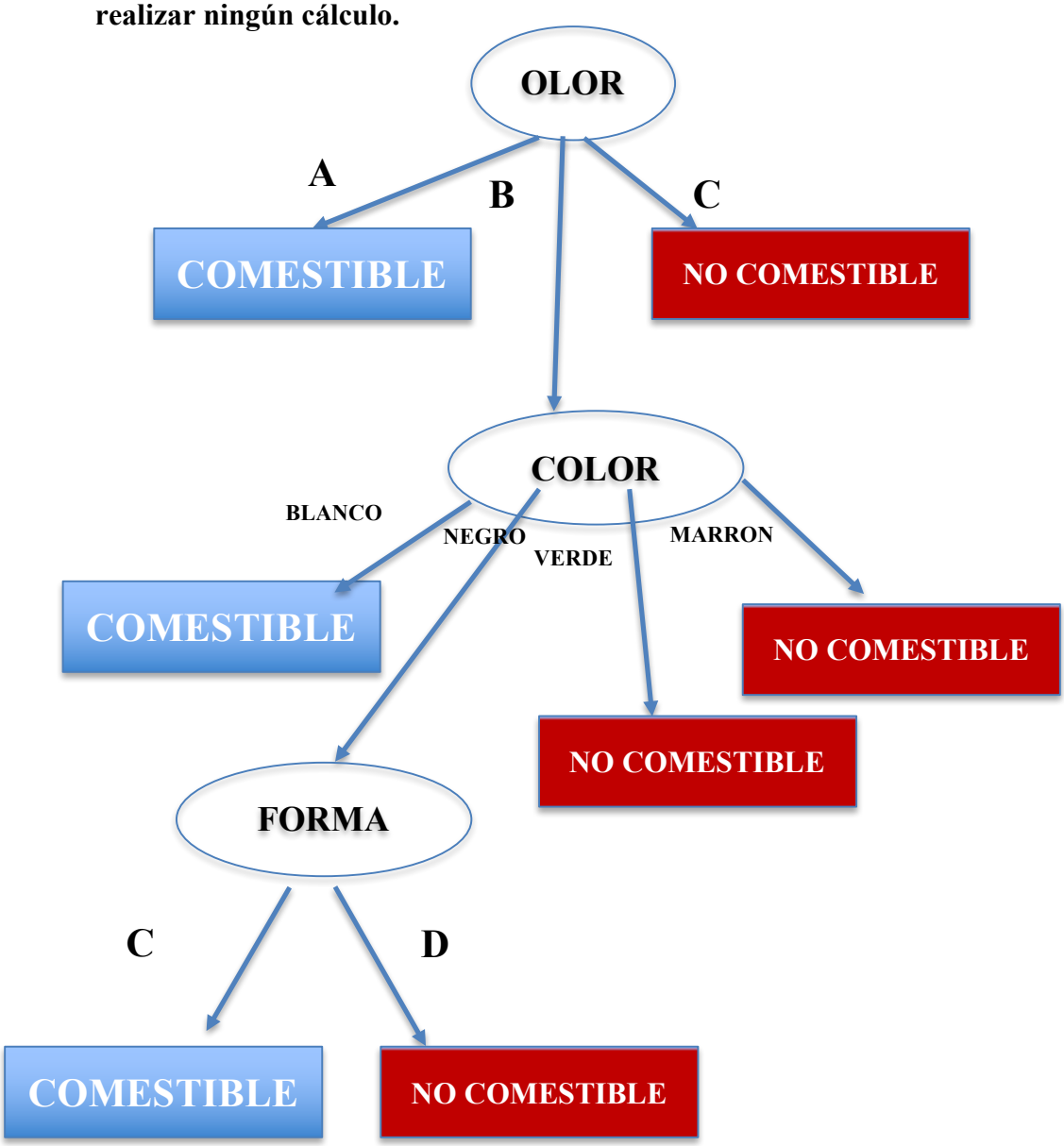
4.4. **[1 punto]** Utilizando el siguiente conjunto de datos para construir un árbol de decisión que ayude a predecir si una seta es comestible o no en función de su forma, color y olor, resuelve las siguientes cuestiones:

Forma	**Color**	**Olor**	**Comestible**
C	Negro	A	SI
D	Negro	A	SI
D	Blanco	A	SI
D	Blanco	B	SI
C	Negro	B	SI
D	Negro	B	NO
D	Verde	B	NO
C	Marrón	B	NO
C	Negro	C	NO
C	Blanco	C	NO
D	Blanco	C	NO

- ¿Qué atributo selecciona ID3 como raíz? ¿Y C4.5? ¿Es el mismo atributo? Explica la razón de que ID· y C4.5 seleccionen (o no) el mismo atributo como raíz como si no lo seleccionan.

La raíz en ambos casos en la variable olor

- Construye el árbol de decisión que se obtiene utilizando ID3. **NOTA: En algunos casos no es necesario realizar todos los cálculos (y en otros no es necesario realizar ningún cálculo). En estos casos es suficiente que expliques por qué no hay que realizar ningún cálculo.**



No es suficiente con dibujar el árbol. Se debe justificar por qué la variable COLOR es la que se debe evaluar cuando OLOR=B y explicar por qué se etiqueta cada hoja.

- ¿Cuál es el error en entrenamiento?

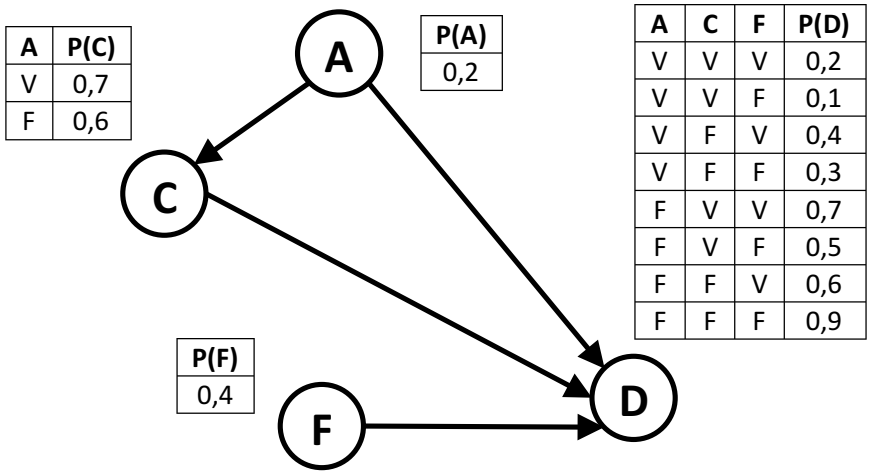
0% puesto que todos los ejemplos de entrenamiento se clasifican bien

Si consideramos el siguiente conjunto de test:

Forma	Color	Olor	Comestible
C	Negro	B	NO
D	Negro	B	NO
C	Blanco	B	SI
D	Negro	B	NO
D	Verde	B	NO
C	Marrón	B	SI

- Calcula el error y la medida F. ¿Es un buen clasificador? ¿Por qué?
- *Se clasifican mal el primer ejemplo y el último, así que el error es 2/6=1/3. El error es del 33%.*
- *Como hay un TP y un FP, la precisión es 0.5. También hay un FN, así que el recall es también 0.5. Así que la F1 es 0.5. Con realizar este cálculo para la clase "Comestible" lo doy por ok.*
- *Para estimar si es un buen clasificador, cabe recordar que en tiempo de entrenamiento, lo que hace el clasificador por defecto es asignar la clase mayoritaria, que EN ENTRENAMIENTO ES "NO COMESTIBLE". Esto significa que cualquier ejemplo de test sería clasificado como "NO COMESTIBLE". De este modo el error en test sería 1/3 también, lo que hace que el árbol obtenido NO sea un buen clasificado, atendiendo al conjunto de test proporcionado.*

5.- Considera la siguiente red bayesiana:



A	P(C)
V	0,7
F	0,6

P(A)
0,2

A	C	F	P(D)
V	V	V	0,2
V	V	F	0,1
V	F	V	0,4
V	F	F	0,3
F	V	V	0,7
F	V	F	0,5
F	F	V	0,6
F	F	F	0,9

P(F)
0,4

a) **[0,7 puntos]** Calcula de forma exacta la siguiente probabilidad condicionada: P(D|¬A, F).

Nota: no es necesario dar el resultado final, sino que puedes dejar las operaciones indicadas, por ejemplo: $\frac{(0{,}5*0{,}3)+0{,}4}{0{,}8*0{,}4}$

Solución:

$$P(D|\neg A,F)=\frac{P(D,\neg A,F)}{P(\neg A,F)}=\frac{P(D,\neg A,F)}{P(D,\neg A,F)+P(\neg D,\neg A,F)}=$$

$$= \frac{P(D, \neg A, F, C) + P(D, \neg A, F, \neg C)}{P(D, \neg A, F, C) + P(D, \neg A, F, \neg C) + P(\neg D, \neg A, F, C) + P(\neg D, \neg A, F, \neg C)} =$$

$$= \frac{(0{,}7 * 0{,}8 * 0{,}4 * 0{,}6) + (0{,}6 * 0{,}8 * 0{,}4 * 0{,}4)}{(0{,}7 * 0{,}8 * 0{,}4 * 0{,}6) + (0{,}6 * 0{,}8 * 0{,}4 * 0{,}4) + (0{,}3 * 0{,}8 * 0{,}4 * 0{,}6) + (0{,}4 * 0{,}8 * 0{,}4 * 0{,}4)} =$$

$$= \frac{0{,}1344 + 0{,}0768}{0{,}1344 + 0.0768 + 0.0576 + 0.0512} = \mathbf{0,66}$$

b) [0,4 puntos] ¿A y F son independientes, si conocemos el valor de D? Debes razonar la respuesta utilizando el criterio de D-separación, o bien la condición de Markov si ésta fuese suficiente.

Solución:

Aplicamos el criterio de D-Separación:

Camino A-D-F: no bloqueado.

Camino A-C-D-F: no bloqueado.

Hay al menos un camino no bloqueado, luego **no son independientes**.

En este caso la condición de Markov no se puede aplicar, debido a que conocemos D.

c) [0,4 puntos] Supón que quieres calcular P(D|¬C,F) utilizando el método de ponderación de la verosimilitud, y que una de las muestras que se generan es la siguiente: {A=F, C=F, D=V, F=V}. ¿Qué peso asignaría el método a esa muestra al generarla?

Solución:

Para generar una muestra, el primer paso es determinar un orden topológico de las variables, por ejemplo: A, C, F, D. La muestra empieza teniendo un peso w=1.

- A: se elige aleatoriamente y en el enunciado nos dice que sale A=F.
- C: será falso obligatoriamente. El nuevo peso será w = 1*0,4 = 0,4 (debido a que la probabilidad de que hubiese salido C=F, sabiendo que A=F, es de 1-0,6=0,4).
- F: será verdadero obligatoriamente. El nuevo peso será w = 0,4*0,4 = 0,16 (debido a que la probabilidad de que hubiese salido F=V es de 0,4).
- D: se elige aleatoriamente y en el enunciado nos dice que sale D=V.

Entonces, la posible muestra final es {A=F, C=F, D=V, F=V} y su peso es w = **0,16**.