APELLIDOS:	PL:
NOMBRE:	DNI:

ESCUELA DE INGENIERÍA INGORMÁTICA

SISTEMAS INTELIGENTES

Examen Final de Teoría. Miércoles 21 de Enero de 2015.

1.- **[3 puntos. Búsqueda]** Se trata de resolver con un algoritmo A* el siguiente problema: dada una matriz cuadrada de dimensión N×N de números no negativos, se trata de calcular un subconjunto de N números tal que la suma sea mínima y de forma que no haya más de un número de una misma fila ni de una misma columna.

Si consideramos la matriz siguiente

$$\begin{pmatrix} 1 & 2 & 3 \\ 5 & 4 & 6 \\ 7 & 9 & 8 \end{pmatrix}$$

la solución óptima viene dada por el subconjunto {1, 4, 8}.

Concretamente se pide lo siguiente:

a) **[1 punto]** Dar una definición precisa del espacio de búsqueda para un valor N. Esto incluye una descripción del estado inicial, la forma de calcular sucesores para cada estado con los costes de los operadores correspondientes y una descripción de los objetivos. Obviamente el espacio debe ser completo y a la vez lo más reducido posible.

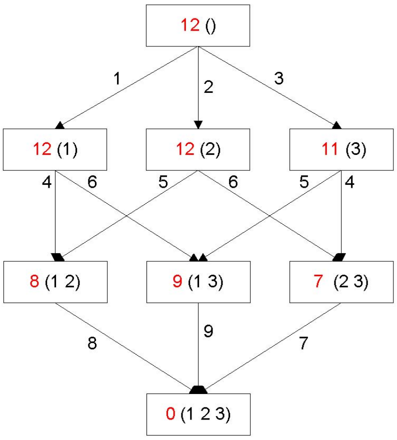
Dibujar el espacio de búsqueda completo para el ejemplo dado por la matriz anterior.

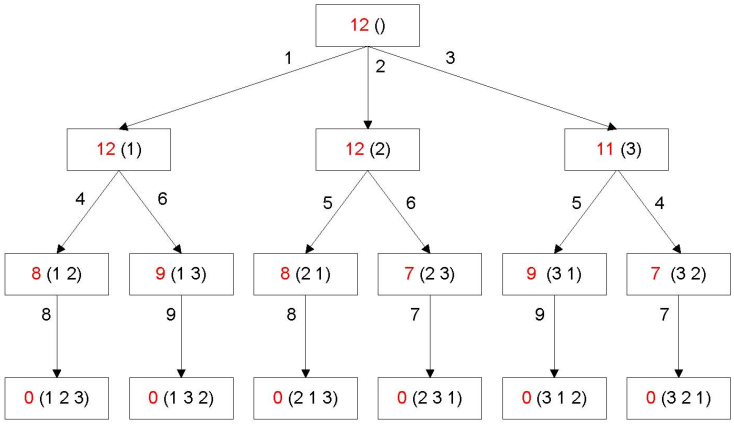
RESP: Un estado se puede definir mediante un subconjunto de elementos de {1..N}. La interpretación de uno de estos estados puede ser la siguiente: si el tamaño del subconjunto es L, entonces para llegar a este estado ya hemos elegido números de las filas 1..L y por lo tanto para llegar a una solución válida tendremos que elegir números de las filas L+1..N tales que no estén en las columnas 1..L y que dos números no estén en la misma fila ni en la misma columna. Así, un estado representa un problema como el original pero restringido a una matriz de tamaño (N-L)×(N-L). Con esta interpretación, el estado inicial vendrá dado por el conjunto vacío y el único estado final vendrá dado por el conjunto total {1,2,..,N}. Dado un estado intermedio n de tamaño L tendrá N-L sucesores, cada uno de los cuales se genera añadiendo al conjunto de n uno de los elementos de {1,..,N} que no están en n. El coste del operador correspondiente es el valor del elemento de la matriz en la posición (L+1,k) si k es el elemento añadido al conjunto.

El espacio definido de esta forma tiene estructura de grafo.

Se puede optar por una definición alternativa en la que el espacio de búsqueda es un árbol. Esta se ilustra con un ejemplo en el apartado b). En este caso los nodos son listas ordenadas de números que indican los índices de las columnas elegidas en cada paso, por ejemplo el estado (2 3) representa que hemos elegido el segundo elemento de la primera fila y el tercero de la segunda fila. Entre las dos opciones, es esperable que la primera sea la mejor ya que resulta en un espacio de búsqueda mucho más reducido.

Las dos figuras siguientes muestran los espacios de búsqueda generados a partir de la matriz anterior de tamaño 3×3:





b) **[1 punto]** Definir un heurístico h_1 "lo mejor posible" para el problema y justificar qué propiedades tiene.

Indicar los valores que toma el heurístico en cada uno de los nodos del espacio pintado en a).

RESP: El problema asociado a cada estado se puede relajar si eliminamos la restricción de que cada número elegido deba estar en una columna diferente (manteniendo que debe estar en columnas distintas a las de los índices que definen el propio estado). De esta forma, el problema relajado se resuelve de forma óptima sin más que elegir el número menor de cada una de las filas L+1..N que no esté en ninguna de las columnas cuyos índices definen el estado. Dado un estado n, $h_1(n)$ lo definimos como el coste óptimo del problema relajado correspondiente al nodo n.

Dado que el heurístico h_1 se obtiene mediante el método de la relajación del problema, es consistente.

Las figuras anteriores muestran estos valores para cada uno de los nodos de los dos espacios de búsqueda (árbol y grafo). Son los números en rojo.

c) **[1 punto]** Indicar los nodos que no se expandirán con el heurístico h anterior y con el heurístico $h_0(n)=0$ para todo nodo n.

RESP: Dado que se trata de un heurístico monótono, la condición que debe cumplir un nodo para que no se expanda es lo contrario de la condición necesaria de expansión, es decir: $f(n) = g(n)+h(n) > C^*$. En este caso $C^* = 13$. Con el heurístico h_1 en el grafo solo hay 4 nodos que no cumplen esta condición, los cuatro que forman parte de la solución óptima. Con el heurístico h_0 solamente la cumplen los 5 nodos solución que no representan soluciones óptimas.

2.- [**2 puntos. Lógica**] Representar el siguiente enunciado en lógica de predicados:
(Formulación 0.7 puntos)

Todos los gatos persiguen a algún ratón.
Algunos ratones son inteligentes.
Los gatos que persiguen a ratones inteligentes no los atrapan.
Cualquier gato que persigue a algún ratón pero no lo atrapa estará hambriento
Tom es un gato y Jerry un ratón y ambos son inteligentes
Tom persigue a Jerry

Luego responder a la pregunta siguiente utilizando refutación:
(Paso a Notación Kowalski y Resolución: 1.3 puntos)

¿Hay algún gato inteligente?

RESP: El enunciado se puede modelar de la siguiente forma:

```java
public static FOLKnowledgeBase createExamenKnowledgeBase(
                        InferenceProcedure infp) {
                FOLKnowledgeBase kb = new FOLKnowledgeBase(DomainFactory.examenDomain
                        infp);
kb.tell("FORALL x (Gato(x) =>(EXISTS y (Raton(y) AND Persigue(x,y))))");
kb.tell("EXISTS x (Raton(x) AND Inteligente(x))");
kb.tell("FORALL x FORALL y (Gato(x) AND Raton(y) AND Inteligente(y) AND
Persigue(x,y)) => NOT Atrapa(x,y))");
// Otra opción para la premisa anterior
//kb.tell("FORALL x FORALL y ((Gato(x) AND Raton(y) AND Persigue(x,y))=>
(Inteligente(y) => NOT Atrapa(x,y)))");
kb.tell("FORALL x ((Gato(x) AND (EXISTS y (Raton(y) AND Persigue(x,y) AND NOT
Atrapa(x,y)))) => Hambriento(x))");
kb.tell("(Gato(Tom) AND Raton(Jerry) AND Inteligente(Tom) AND
Inteligente(Jerry))");
kb.tell("Persigue(Tom, Jerry)");
return kb;
}
```

La pregunta se formaliza así:

```java
String query = "EXISTS x (Gato(x) AND Inteligente(x))";
```

Y el resultado de la refutación, sin utilizar el mecanismo de Green (predicado Answer), es el siguiente. Se llega a la clausula vacía, luego el resultado es afirmativo:

```
-------------------------------------
TFM Resolution, Examen
-------------------------------------
#newClauses: 6
#newClauses: 0Gato Knowledge Base:
FORALL x (Gato(x) => EXISTS y (Raton(y) AND Persigue(x,y)))
EXISTS x (Raton(x) AND Inteligente(x))
FORALL x FORALL y (((Gato(x) AND Raton(y)) AND Inteligente(y)) AND Persigue(x,y))
FORALL x ((Gato(x) AND EXISTS y ((Raton(y) AND Persigue(x,y)) AND
NOT(Atrapa(x,y)))) => Hambriento(x))
(((Gato(Tom) AND Raton(Jerry)) AND Inteligente(Tom)) AND Inteligente(Jerry))
Persigue(Tom,Jerry)

Query: EXISTS x (Gato(x) AND Inteligente(x))
Proof, Answer Bindings: {x=null}
----------------------------------------------------------------------------------
---------------------------------------
|Step | Proof
|Justification                |
----------------------------------------------------------------------------------
---------------------------------------
|1    | EXISTS x (Raton(x) AND Inteligente(x))
|Premise                      |
|2    | FORALL x FORALL y (((Gato(x) AND Raton(y)) AND Inteligente(y)) AND
Persigue(x,y)) |Premise                      |
|3    | [~Gato(v18), ~Inteligente(v18)]
|Goal                         |
|4    | [Inteligente(SC0)]
|Clausified 1                 |
|5    | [Gato(x)]
|Clausified 2                 |
|6    | [Gato(v2)]
|Renaming of 5                |
|7    | [Gato(v11)]
|Renaming of 6                |
|8    | [~Inteligente(c0)]
|Resolution: 3,7 {}, {v18=c0} |
|9    | []
|Resolution: 4,8 {c0=SC0}, {} |
----------------------------------------------------------------------------------
---------------------------------------
```

3.- **[3 puntos. Aprendizaje]**

a) Sea un algoritmo de aprendizaje automático cualquiera. Supongamos que un clasificador que utiliza a este algoritmo se usa para identificar correo spam (Problema A) y para clasificar pacientes como diabéticos o no diabéticos (Problema B). Para probar la bondad del clasificador sobre dichos conjuntos se utiliza:

Problema A: 1000 ejemplos de test, 950 de la clase spam y 50 de la clase no spam.
Problema B: 650 ejemplos de test, 300 ejemplos de la clase diabético, 350 de la clase No_diabético.

a1) [0.5 puntos] Si el porcentaje de acierto obtenido con el algoritmo para el Problema A es del 92%, ¿puede decirse que el rendimiento del algoritmo para el Problema A es bueno? ¿por qué? Y si para el Problema B el porcentaje de acierto es de 70%, puede considerarse que el algoritmo de aprendizaje obtiene buenos resultados para el problema B? ¿por qué?
Para el problema A, si a todos los ejemplos de test le asignamos la clase mayoritaria el porcentaje de acierto sería 950/1000=0.95 (95%). Así que un clasificador que obtuviera un porcentaje del 92% sería MALO porque clasificaría peor que el clasificador por defecto. En cambio, para el problema B, el clasificador por defecto obtendría un porcentaje de acierto del 53.8% así que un clasificador que obtuviera un 70% de aciertos lo mejoraría sensiblemente

a2) [0.25 puntos] ¿Cómo debería ser la matriz de confusión para que la precisión fuese del 50% para el Problema A?
Dado que la precisión se define como TP/(TP+FP), para que la precisión fuera del 50% el número de Falsos Positivos y el de Verdaderos positivos debería ser el mismo.

b) Considera el siguiente conjunto de ejemplos utilizado para aprender a identificar si una seta es venenosa o no

Ejemplo	*EsDura*	*EsOlorosa*	*TieneManchas*	*EsRegular*	*EsVenenosa*
A	0	0	0	0	0
B	0	0	1	0	0
C	1	1	0	1	0
D	1	0	0	1	1
E	0	1	1	0	1
F	0	0	1	1	1
G	0	0	0	1	1
H	1	1	0	0	1
U	1	1	1	1	¿
V	0	1	0	1	¿
W	1	1	0	0	¿

Considerando sólo los ejemplos de la A a la H, responde a las siguientes cuestiones

b1) [0.25 puntos] ¿Qué atributo debería seleccionarse como la raíz del árbol de decisión que genera C4.5? ¿Por qué?

Para saber cual es el atributo más informativo, en este caso debería utilizarse la métrica que usa C4.5 que es Gain Ratio. Gain Ratio se calcula como la Ganancia de Información ponderada por el tamaño de cada clase (trasparencia 51). En este caso particular, la ordenación producida por Gain Ratio y por la Ganancia de Información era la misma así que he considerado correctos todos aquellos que han utilizado CORRECTAMENTE IG en vez de GR

H(esVenenosa) **=-3/8 * log(3/8) - 5/8*log(5/8)**

H(esVenenosa/EsRegular=0)=1 (porque hay igual numero de ejemplos de cada clase)
H(esVenenosa/EsRegular=1)=-1/4*log(1/4)-3/4 log(3/4)=0.8133 (Estos mismos cálculos están en las trasparencia 44)
H(esVenenosa/EsRegular)=1.8133

H(esVenenosa/TieneManchas=0)= -2/5*log(2/5)-3/5*log(3/5)=0.970
H(esVenenosa/TieneManchas=1)= -1/3*log(1/3)-2/3*log(2/3)=0.9183

H(esVenenosa/EsOlorosa=0)= -2/5*log(2/5)-3/5*log(3/5)
H(esVenenosa/EsOlorosa=1)= -1/3*log(1/3)-2/3*log(2/3)

H(esVenenosa/EsDura=0)= -2/5*log(2/5)-3/5*log(3/5)
H(esVenenosa/EsDura=1)= -1/3*log(1/3)-2/3*log(2/3)

H(esVenenosa/EsOlorosa)= H(esVenenosa/EsDura)= H(esVenenosa/EsOlorosa)= 1.8883

Como H(esVenenosa/EsRegular) < H(esVenenosa/EsOlorosa)= H(esVenenosa/EsDura)= H(esVenenosa/EsOlorosa), el atributo que mayor ganancia de información tiene es EsRegular

Ahora habría que calcular SplitInfo de cada Atributo, pero como todos los atributos toman el mismo número de valores distintos (2) no es necesario calcularla. Así que el atributo más informativo es EsRegular y por tanto es la raíz del árbol

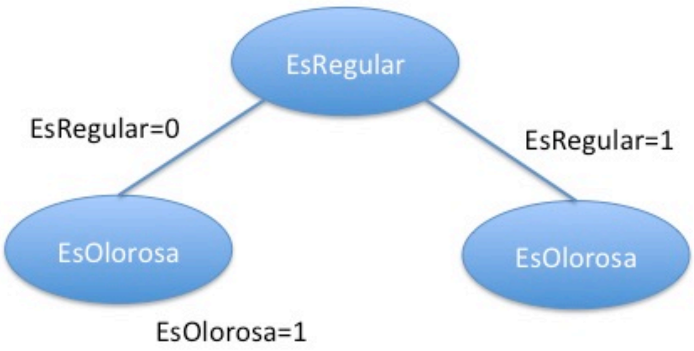
b2) [1 punto] Construye RAZONADAMENTE un árbol del decisión utilizando como métrica Gain Ratio (la que utiliza C4.5). Profundiza en el árbol hasta que en cada nodo todos los ejemplos sean de la misma clase O haya como máximo 2 ejemplos. Para ello puedes utilizar la información contenida en la siguiente tabla, donde cada valor representa GainRatio(Y/X) (Por ejemplo el 0.2368 que aparece en la fila 1 y columna 2 es *GainRatio(EsOlorosa/EsDura=0)*)

X	***Y***			
	EsDura	*EsOlorosa*	*TieneManchas*	*EsRegular*
EsDura=0		0.2368	0.0206	0.4325
EsOlorosa=0	0.2328		0.0206	1
TieneManchas=0	0.0206	0.0206		0.0206
EsRegular=0	0.384	**1**	0	
EsDura=1		0.274	0	0.274
EsOlorosa=1	0.274		0.274	1
TieneManchas=1	0	0.274		0.274
EsRegular=1	0.311	1	0.151	

Ahora ya sabemos que la raíz del árbol es EsRegular, tenemos que decidir cuál es el siguiente nivel del árbol. En la tabla que se adjunta tenemos los valores de Gain Ratio para no tener que calcularlos. De todos ellos solo necesitamos los que hacen referencia a EsRegular

X	***Y***			
	EsDura	*EsOlorosa*	*TieneManchas*	*EsRegular*
EsRegular=0	0.384	**1**	0	
EsRegular=1	0.311	1	0.151	

Para el conjunto de ejemplos que cumplen EsRegular=0, el atributo más informativo es EsOlorosa, lo mismo que para el conjunto de ejemplos que cumplen EsRegular=1, así que el segundo nivel del árbol es



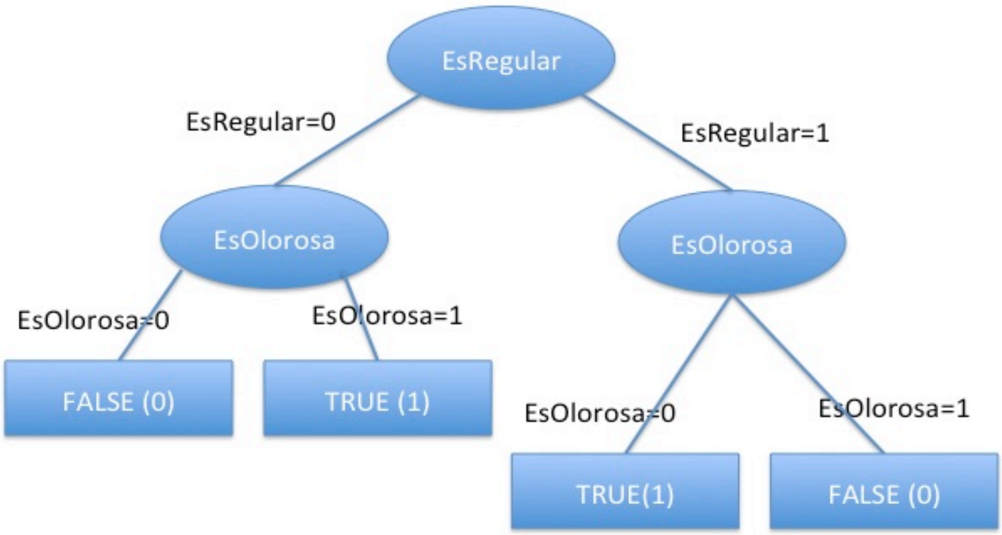
Finalmente, los ejemplos que verifican EsRegular01 son los siguientes

Ejemplo	*EsDura*	*EsOlorosa*	*TieneManchas*	*EsRegular*	*EsVenenosa*
C	1	1	0	1	0
D	1	0	0	1	1
F	0	0	1	1	1
G	0	0	0	1	1

Como puede observarse para estos ejemplos, siempre que EsOlorosa=1, EsVenenosa=0 y siempre que EsOlorosa=0, EsVenenosa=1, así que EsOlorosa determina la clase para este conjunto de ejemplos de forma unívoca. Así pues el siguiente nivel serán hojas

Cuando EsRegular=0 ocurre lo mismo, el valor de EsOlorosa determina unívocamente el valor de la variable objetivo, así que nuevamente sin ningún cálculo concluimos que los nodos del siguiente nivel son hojas

Ejemplo	*EsDura*	*EsOlorosa*	*TieneManchas*	*EsRegular*	*EsVenenosa*
A	0	0	0	0	0
B	0	0	1	0	0
E	0	1	1	0	1
H	1	1	0	0	1



b3) [0.5 puntos] Utilizando el árbol de decisión que has obtenido, clasifica los ejemplos U y W.

Sólo hay que evaluar las variables que aparecen en el árbol para esos dos ejemplos

Ejemplo	*EsOlorosa*	*EsRegular*	*EsVenenosa*
U	1	1	0
W	1	0	1

b4) [0.5 puntos] Aplicando el algoritmo IB3, ¿A qué clase pertenece el ejemplo V?

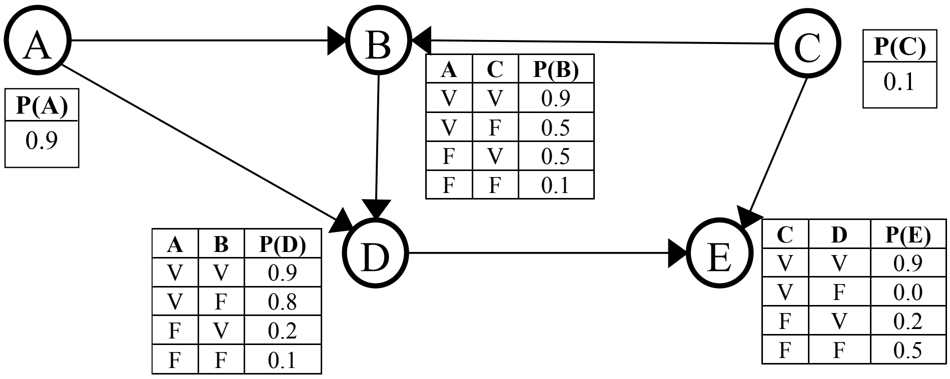
Tenemos que calcular la distancia euclidea del ejemplo V a todos los ejemplos de entrenamiento, seleccionar los tres más próximos, y asignar a V la clase mayoritaria de estos 3 ejemplos más próximos.

D(a,v) = 1
D(b,v) = 1.4142
D(c,v) = 1.4142
D(d,v) = 1.7321
D(e,v) = 1
D(f,v) = 1.7321
D(g,v) = 1.4142
D(h,v) = 1

Así que los más próximos son los ejemplos, A, E y H. El ejemplo A pertenece a la clase EsVenenosa=0 y el E y el H a la clase EsVenenosa=1. Así que la mayoritaria es EsVenenosa=1, que es la que se le asigna al ejemplo V

4.- **[2 puntos. Incertidumbre]**

Considera la siguiente red bayesiana:



P(A)
0.9

A	C	P(B)
V	V	0.9
V	F	0.5
F	V	0.5
F	F	0.1

P(C)
0.1

A	B	P(D)
V	V	0.9
V	F	0.8
F	V	0.2
F	F	0.1

C	D	P(E)
V	V	0.9
V	F	0.0
F	V	0.2
F	F	0.5

a. (0.4 puntos) Calcula P(¬A,¬B,¬C,D,E)
Solución: P(¬A,¬B,¬C,D,E) = 0.1 x 0.9 x 0.9 x 0.1 x 0.2

b. (0.4 puntos) Calcula P(¬B,¬C,D,E)
Solución: P(¬B,¬C,D,E) = P(¬A,¬B,¬C,D,E) + P(A,¬B,¬C,D,E) = (0.1 x 0.9 x 0.9 x 0.1 x 0.2) + (0.9 x 0.5 x 0.9 x 0.8 x 0.2)

c. (0.5 puntos) ¿Cuáles de las siguientes afirmaciones se cumplen, dada la estructura de la red? No es suficiente decir sí o no, debes razonar la respuesta.
   - P(A,C) = P(A)P(C)
     Solución: Esta fórmula nos indicaría que A y C son variables independientes. Con la condición de Markov se puede demostrar muy fácilmente: sabiendo el valor de los padres de A (no tiene), podemos decir que A es independiente de todos sus no-descendientes. C no es un descendiente de A, así que por tanto A y C son independientes.
   - P(D|B) = P(D|B,A)
     Solución: Esta fórmula nos indicaría que D y A son condicionalmente independientes dado B. Esto está claro que no es cierto, porque incluso aunque sepamos el valor de B, A y D son dependientes. Por ejemplo, sabiendo que B es verdadero, la probabilidad de que D sea verdadero es 0,9 si A es verdadero pero es 0,2 si A es falso. Por tanto es claro que es falso.

d. (0.7 puntos) Supón que quieres calcular P(¬B,¬C,E | D), pero no de forma exacta sino de forma aproximada, mediante el algoritmo estándar de muestreo. Explica brevemente cómo tendrías que proceder.
Solución: P(¬B,¬C,E | D) = P(¬B,¬C,E,D) / P(D). Por lo tanto, deberíamos generar un gran número de muestras aleatorias (el número dependerá de la precisión que deseemos). Contaremos cuántas de esas muestras cumplen ¬B∧¬C∧E∧D (llamemos Ns a ese número), y cuántas cumplen D (llamemos Nc a ese número). Entonces, Ns/Nc será una aproximación de P(¬B,¬C,E | D).

Nota: En los apartados a) y b) no es necesario dar el resultado final exacto, es suficiente con dejar indicadas las operaciones necesarias, por ejemplo 0,4 x 0,5 + 0,9 x 0,2 x 0,1 + 0,4 x 0,9.