CUESTIONES DE TEORÍA

Razona brevemente cómo puede afectar el problema de las ramas infinitas a cada técnica de búsqueda ciega (2 frases por técnica de búsqueda ciega).

Indica para cada técnica la característica, ventaja y desventaja más característica.

Responde posteriormente a la pregunta, de forma razonada y breve, ¿depende del problema que se está tratando, por ejemplo, 8-puzzle o camino más corto en un grafo? (2 frases).

Búsqueda	Característica Implementación	Ventaja más importante	Desventajas más importante	Ramas infinitas
Anchura	Frontera es una cola FIFO	Método completo y complejo. Es óptimo	Gran complejidad espacio-temporal y consume mucha memoria	No le afectan por ser completo y complejo.
Profundidad	Frontera es una pila	Complejidad espacial reducida y permite ahorrar memoria	No es óptimo ni complejo, con gran complejidad temporal	Se puede perder por ramas infinitas
Profundidad limitada	Límite de profundidad a partir del cual no genera más hijos	No se pierde por ramas infinitas	No es completo ni soluciona el problema del límite	No se pierde por ramas infinitas
Profundidad con retroceso	Genera sólo un hijo por nodo e iteración	Ahorra más memoria que en profundidad	Se puede perder por ramas infinitas	Se puede perder por ramas infinitas
Profundidad iterativa	Aplica una búsqueda en profundidad limitada incrementando el límite si no se encuentra la solución	Óptima	Tarda mucho	
Bidireccional	Aplica 2 búsquedas, desde estado inicial a estado final	Tarda menos	Necesita conocer el estado final	No le afectan si una búsqueda es en anchura
Coste uniforme	Utiliza los pesos de los arcos	Óptima considerando los	Consume mucha memoria	No le afectan por ser óptima y

		pesos de los arcos		completo

Si depende, pues hay problemas como el de las 8 reinas que no tienen ramas infinitas.

Ejercicio test

¿Qué objetivo tiene la heurística en A*?

- Estima cuál es el coste del estado solución.
- Informa cuáles son los estados que se deben explorar.
- Orienta a la búsqueda hacia donde cree que está el estado solución.
- Indica exactamente cuánto falta para encontrar el estado solución

El principio de resolución de la Lógica de Predicados se aplica…

- Sobre dos cláusulas que tienen al menos un predicado afirmado en una y negado en otra.
- Sobre dos cláusulas que tienen uno y sólo un predicado afirmado en una y negado en otra.
- Sobre dos cláusulas que tienen un predicado común con el mismo signo y los mismos argumentos.
- Sobre un conjunto de cláusulas que tienen un predicado común y está negado en alguna de ellas.

¿En cuál de las siguientes situaciones son dos variables de una red bayesiana independientes?

- Cuando el único camino que las conecta no tiene nodos sumidero y ninguno conocido.
- Cuando uno de los caminos que las conecta está cortado por conocer o no conocer alguna de sus variables.
- Cuando el camino que las conecta sólo tiene nodos sumidero no conocidos.
- Cuando no hay ningún camino que las conecte.

¿Qué ocurre en un grafo de Schank si el que realiza la acción es el mismo que se enlaza con la primitiva con el enlace de objeto?

- El que realiza la acción está ejecutando algo sobre sí mismo.
- El que realiza la acción está solo.
- El que realiza la acción no puede ser el mismo que el objeto de la primitiva.
- El que realiza la acción está obteniendo algún beneficio de la acción que ejecuta.

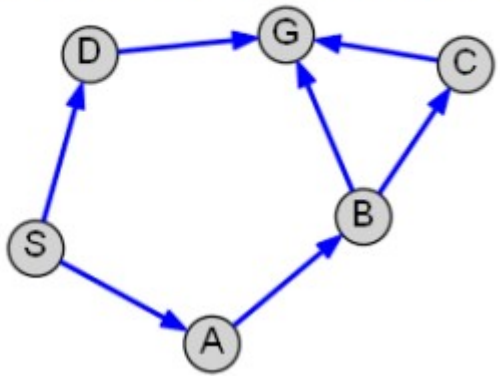
¿Qué significa que un sistema basado en reglas utilice el axioma del mundo cerrado?

- Que no puede disparar reglas con hipótesis negadas en su antecedente.
- Que considera que lo que está fuera del mundo es desconocido.
- Que considera que es falso todo aquello que ni conoce ni puede deducir.
- Que no puede deducir la falsedad de aquello que no esté afirmado.

¿Por qué la complejidad espacial de la búsqueda en profundidad es menor que la de la búsqueda en anchura?

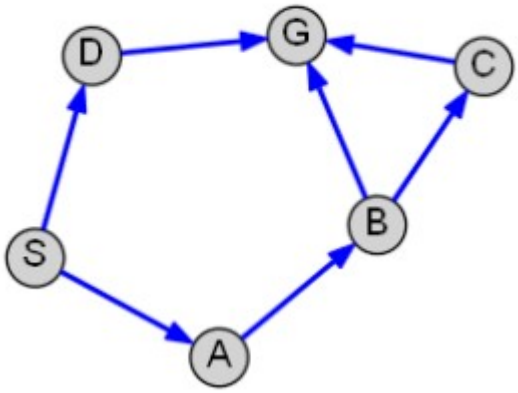
- Porque las ramas exploradas que no llevan a la solución no se introducen en memoria.
- Porque sólo es necesario almacenar la rama que llega a la solución.
- Porque su complejidad no es exponencial.
- Porque se borran estados de memoria cuando la búsqueda examina nodos menos profundos.

- ¿Qué solución se obtiene al realizar una búsqueda en profundidad (DFS) sobre el siguiente grafo? S es nuestro nodo inicial, G representa el objetivo de nuestro problema de búsqueda y los hijos de un nodo se visitan en orden alfabético.



- S→ A→ B→ G
- S→ A → B→ C→ G
- S→ D→ G

¿Qué solución se obtiene al realizar una búsqueda en anchura (BFS) sobre el siguiente grafo? S es nuestro nodo inicial, G representa el objetivo de nuestro problema de búsqueda y los hijos de un nodo se visitan en orden alfabético.



- S→ A→ B→ G
- S→ A→ B→ C→ G
- S→ D→ G

En una búsqueda en profundidad sobre un árbol de profundidad m y factor de ramificación b, ¿cuántos nodos puede haber en la frontera?

- Como mucho b*m
- Hasta bm

En una búsqueda en profundidad sobre un árbol de profundidad m y factor de ramificación b, ¿cuántos nodos puede que tengamos que explorar?

- No más de b*m
- Hasta b^m

En una búsqueda en anchura sobre un árbol de ramificación b, ¿cuántos nodos puede haber en la frontera?

- Como mucho b*m
- Hasta b^s, donde s es la profundidad a la que se encuentra la solución más cercana a la raíz del árbol.

- Hasta b^m, donde m es la profundidad del árbol.

En una búsqueda en anchura sobre un árbol de ramificación b, ¿cuántos nodos puede que tengamos que explorar?

- Como mucho b*m
- Hasta b^s, donde s es la profundidad a la que se encuentra la solución más cercana a la raíz del árbol.
- Hasta b^m, donde m es la profundidad del árbol.

¿Es completa una búsqueda en profundidad sobre un árbol? En otras palabras, ¿devuelve siempre una solución en caso de que ésta exista?

- Verdadero
- Falso

¿Es completa una búsqueda en anchura sobre un árbol? En otras palabras, ¿devuelve siempre una solución en caso de que ésta exista?

- Verdadero
- Falso

¿Es óptima la solución encontrada al realizar una búsqueda en profundidad sobre un árbol? Es decir, de las soluciones que puede tener un problema, ¿devuelve siempre la solución más cercana al nodo inicial de la búsqueda?

- Verdadero
- Falso

¿Es óptima la solución encontrada al realizar una búsqueda en anchura sobre un árbol? Es decir, de las soluciones que puede tener un problema, ¿devuelve siempre la solución más cercana al nodo inicial de la búsqueda?

- Verdadero
- Falso

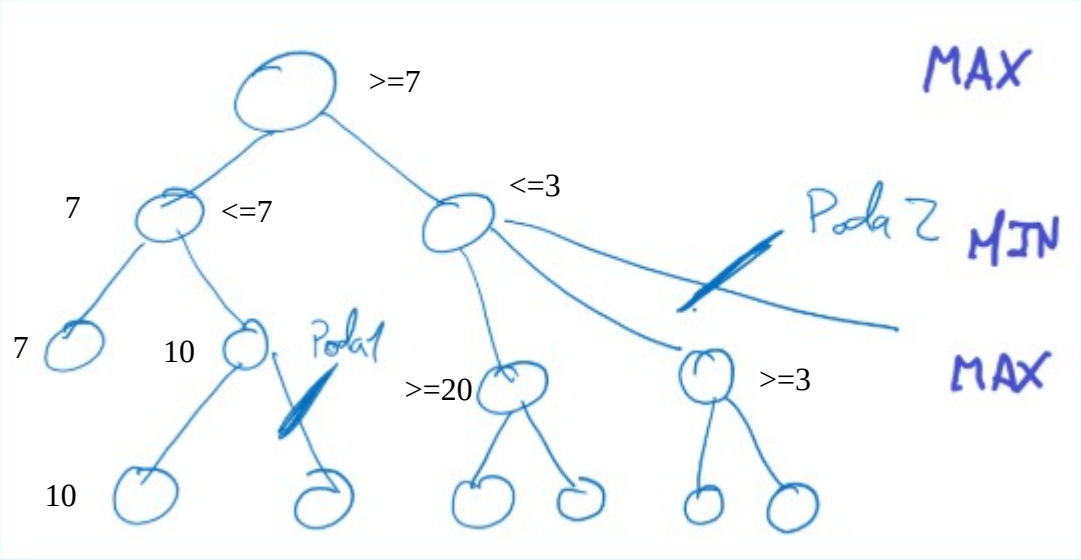
Debes diseñar un sistema basado en marcos para representar la información que maneja un sistema. Enumera las principales cosas que debes tener en cuenta para que tu diseño sea correcto. No escribas más de 15 frases simples. Por ejemplo, un detalle a tener en cuenta que puede ser o no correcto sería "Las clases se dibujan con rectángulos y las instancias con elipses"

- Las clases se representan con rectángulo y las instancias con elipses.
- Las clases tienen un nombre y unas propiedades, restricciones, métodos, etc.
- Las instancias pueden tener valores concretos en las propiedades.
- Las clases representan conceptos abstractos y las instancias conceptos concretos.
- Las clases se pueden asociar con enlaces de subclases, que se deben poder leer “ES-UN”, y especifica herencia de propiedades, restricciones, etc.
- Las instancias están asociadas a una clase con un enlace de instancia, que se debe poder leer “ES-UN”.
- Las propiedades pueden ser otras clases, y entonces sale un enlace de propiedad desde la propiedad a la clase.
- Cuando una propiedad de una instancia es otra instancia, también sale un enlace de propiedad que une la primera con la segunda.

Explica con no más de 15 frases, cómo funciona una neurona artificial y qué significa su entrenamiento. No escribas fórmulas, explica las ideas.

- Una neurona artificial es un elemento computacional que recibe unas entradas.
- Calcula la suma ponderada de las entradas y se activa si dicha suma supera un umbral.
- Su entrenamiento consiste en ajustar los pesos de la suma ponderada para acertar sobre un conjunto de datos de entrenamiento.
- Divide el espacio de entrada en dos por un plano. A un lado del plano se activa y al otro no.

Asigna valores a los nodos hoja del siguiente árbol para que, al ejecutar el algoritmo de poda alfa-beta, se ejecuten las podas indicadas y sólo las indicadas. Comenta además por qué ocurre cada poda, con una frase para cada una de ellas.



Poda 1: No existe un valor menor o igual a 7, y mayor o igual a 10.

Poda 2: No existe un valor menor o igual a 7, y menor o igual a 3.

Explica el funcionamiento de la construcción automática ID3

Se recibe un conjunto de datos descritos por una serie de acracterísticas.

Una de las características es la que se desea predecir.

En cada paso se busca la característica que más entropía reduce del conjunto a clasificar.

Se divide por esa característica y se repite mientras haya conjuntos con elementos de diferente característica objetivo.

Explica en qué es y en qué se basa el aprendizaje no supervisado. ¿ En qué consiste el aprendizaje K-means?

El aprendizaje no supervisado es el que ocurre cuando el conjunto de entrenamiento no tiene características objetivo.

Se basa en agrupar elementos parecidos entre sí y diferenciarlos de los que son más diferentes.

K-medios empieza seleccionando algunos elementos como patrones ideales.

1. Después asigna las palabras al patrón ideal más aprecido.
2. Calcula el centroide de los grupos generados y los considera los nuevos patrones ideales.

3. Repite desde el paso 1 hasta que no cambian los grupos.

EJERCICIOS DE EXAMEN

Ejercicio Sowa-Schank

1. **Marcos evaluó el trabajo de Pedro con un 6. (Sowa)**

[EVALUAR]
➔ (AGT) → [PERSONA: #MARCOS]
➔ (PASADO)
➔ (OBJ) → [TRABAJO: #]
➔ (POS) → [PERSONA: #PEDRO] → (VALOR) → [NOTA: #6]

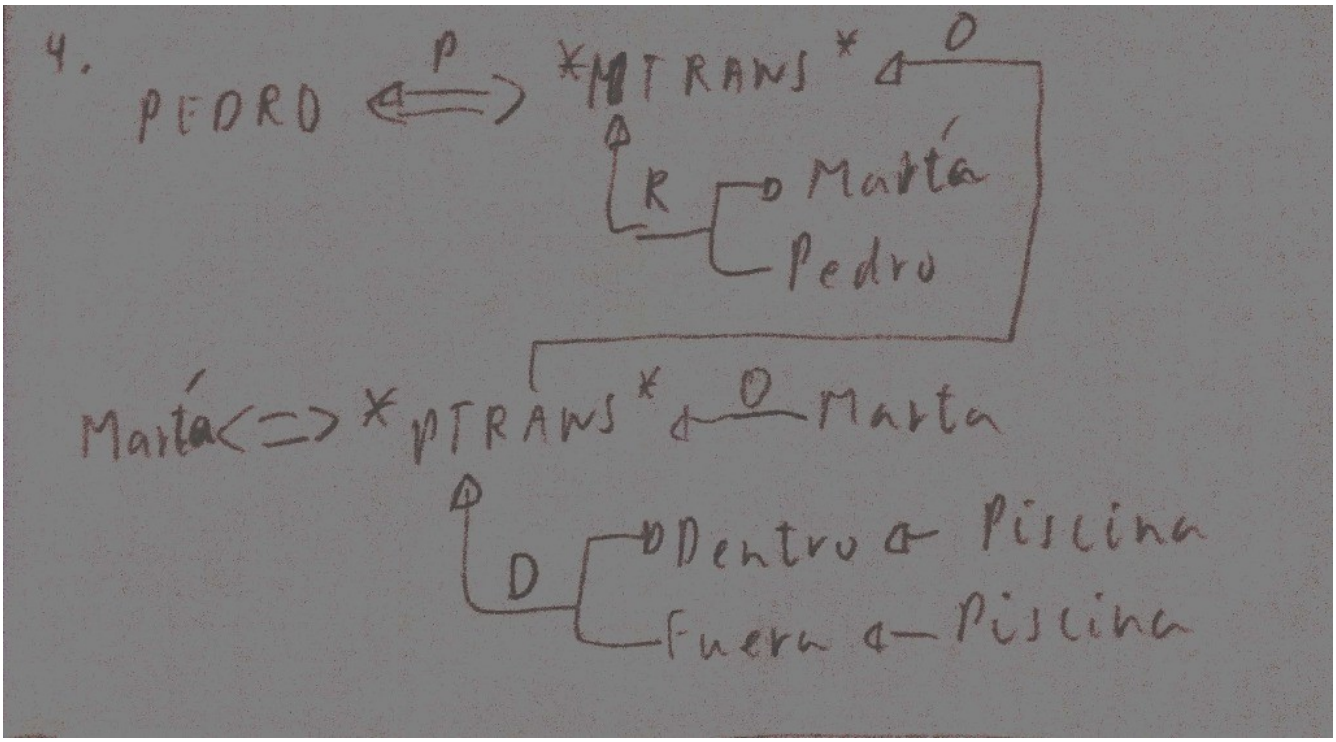
2. **Para aprobar, hay que pensar. (Sowa)**

[CONDICION]
(SI) → [PROPOSICION: [PENSAR]
➔ (AGT) → [PERSONA: {X}]]
(ENTONCES) → [PROPOSICION: [APROBAR]
➔ (AGT) → [PERSONA: {X}]]

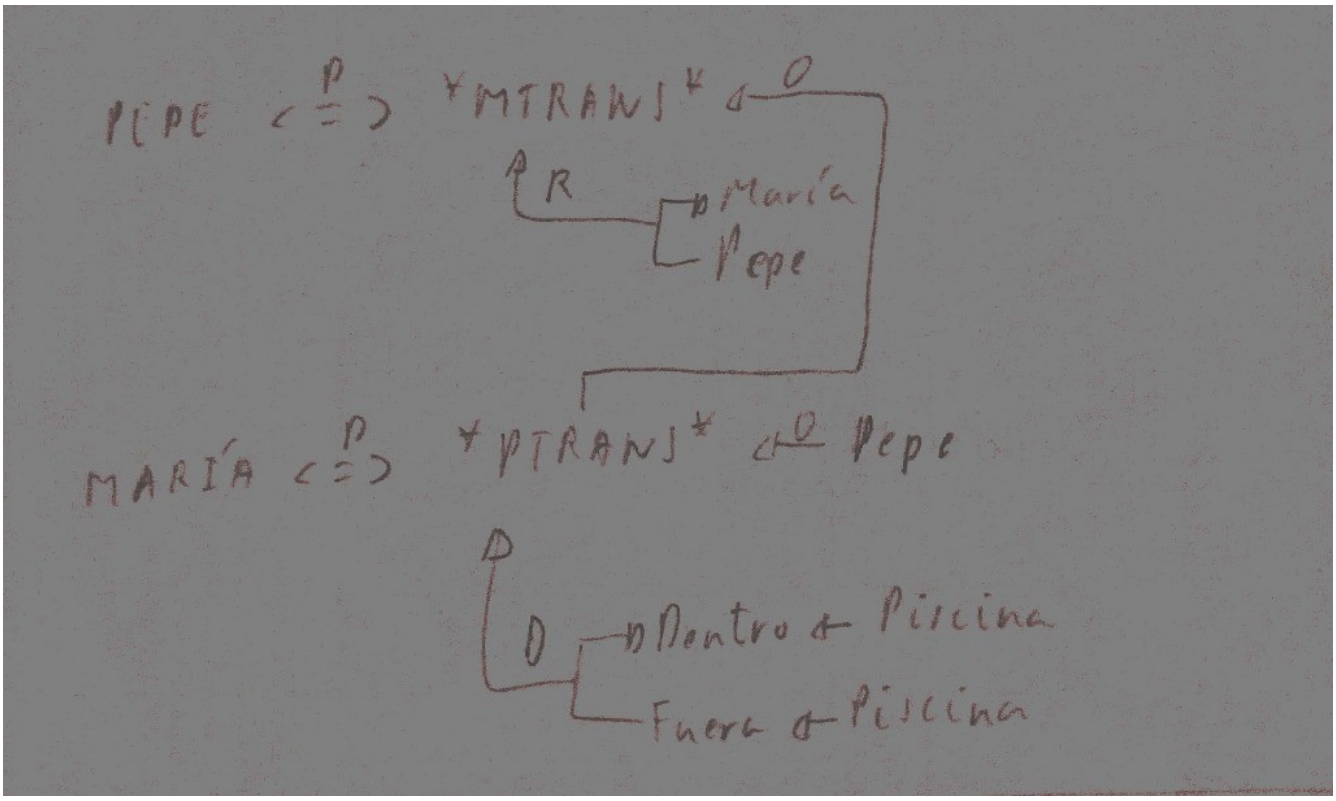
3. **Marta se pregunta si su programa es correcto. (Sowa)**

➔ [PREGUNTAR]
➔ (AGT) → [PERSONA: #MARTA]
➔ (RCP) → [PERSONA: #MARTA]
➔ (OBJ) → [PROGRAMA: #]
➔ (POS) → [PERSONA: #MARTA] → (ATR) → [CORRECTO]

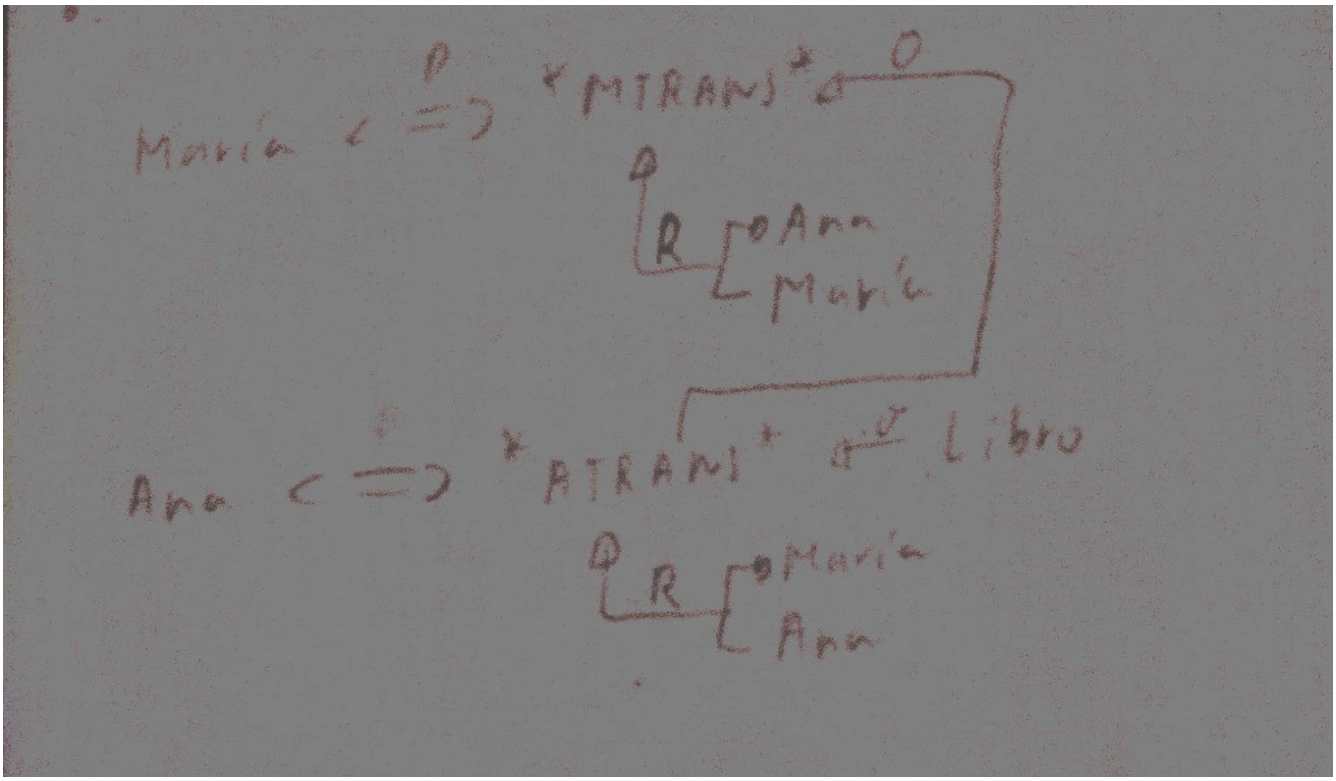
4. **Pedro sugirió a Maria tirar a Marta a la piscina (Schank)**



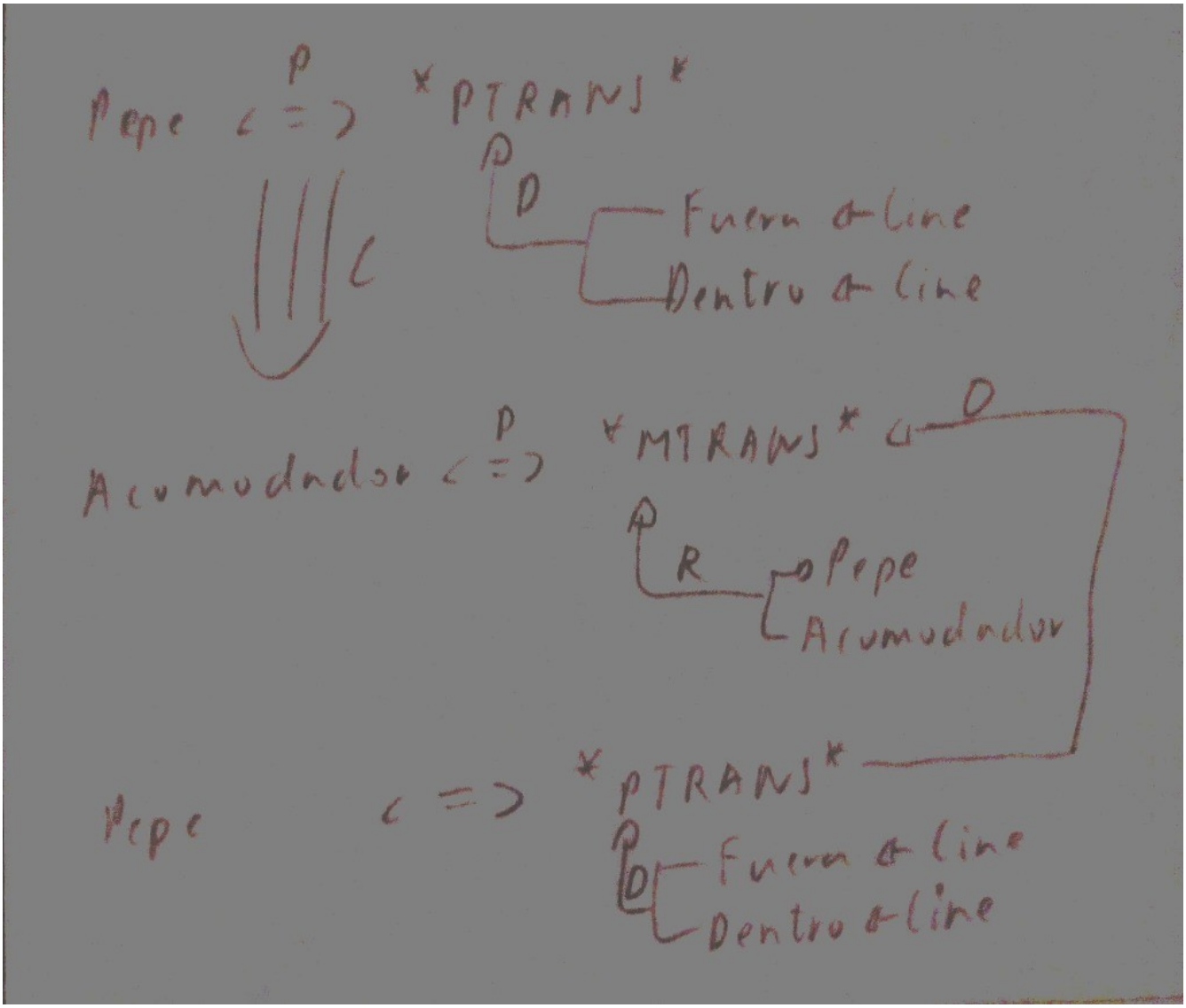
5. **Pepe le pidió a María que le acercase a la piscina (Schank)**



6. **María le pidió prestado el libro a Ana (Schank)**



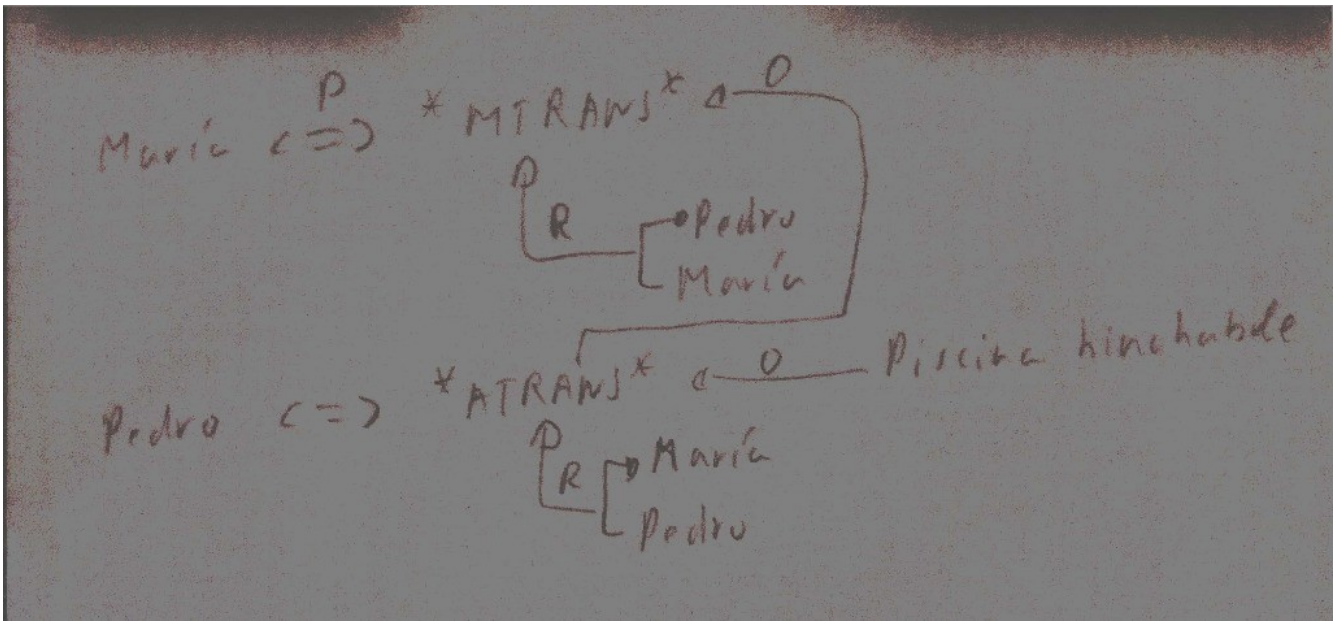
7. **Pepe salió del cine porque el acomodador se lo pidió (Schank).**



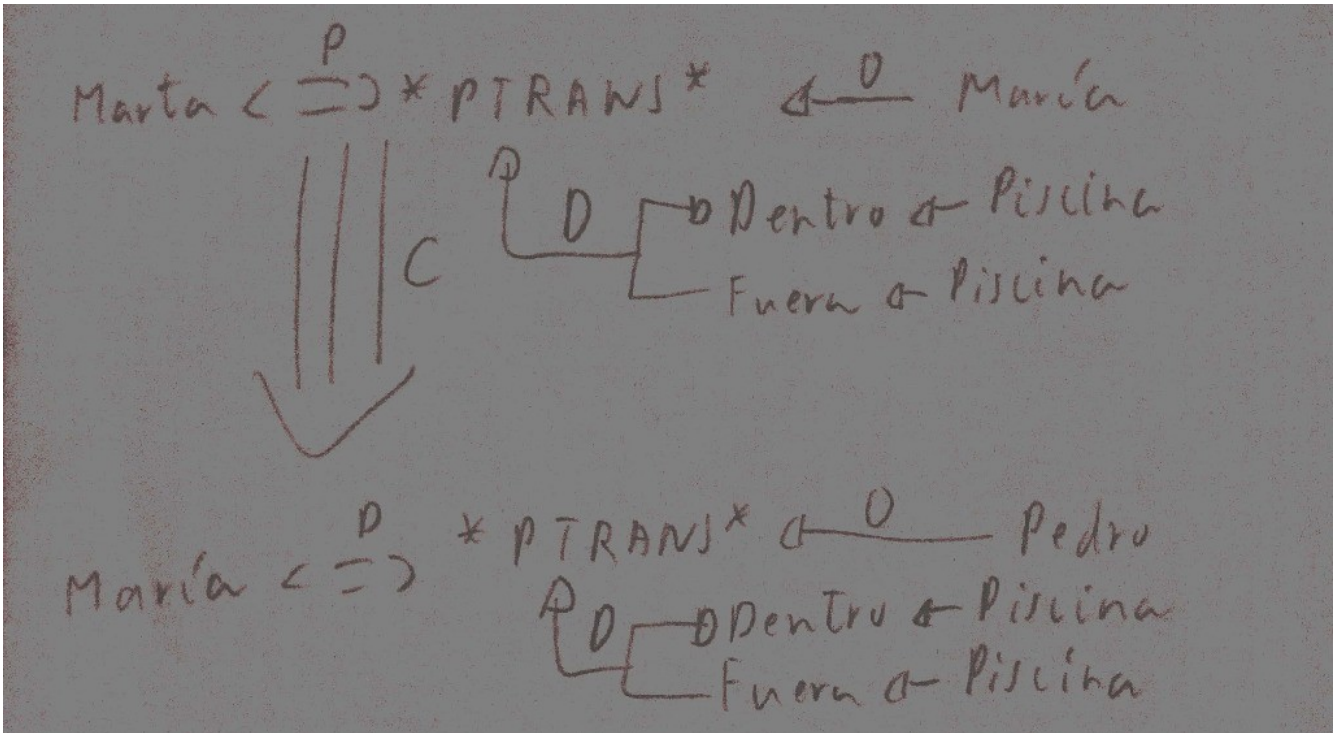
8. **Ana deduce que Marcos está intentando manipularle con memes con medias verdades (Sowa).**

➔ [DEDUCIR]
- ➔ (AGT) → [PERSONA: #ANA]
- ➔ (OBJ) → [PROPOSICION: [INTENTAR MANIPULAR]
  - ➔ (AGT) → [PERSONA: #MARCOS]
  - ➔ (RCP) → [PERSONA: #ANA]
  - ➔ (INSTR) → [MEMES: {*}] → (ATR) → [MEDIAS VERDADES]

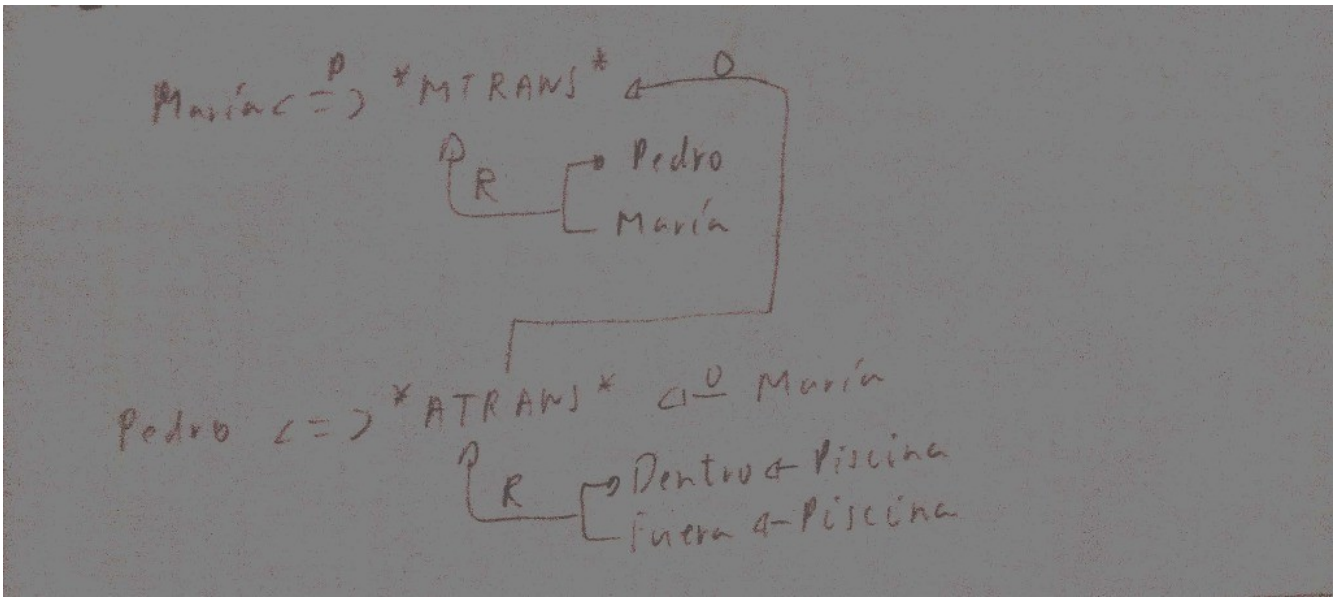
9. **María le pidió a Pedro la piscina hinchable (Schank).**



10. **Marta lanzó a Maria a la piscina porque María había lanzado a Pedro a la pisicina. (Schank)**



11. María le pidió a Pedro que la lanzara a la piscina. (SCHANK)



12. Marta decidió leer 2 periódicos, pues el suyo estaba sesgado. (Sowa)

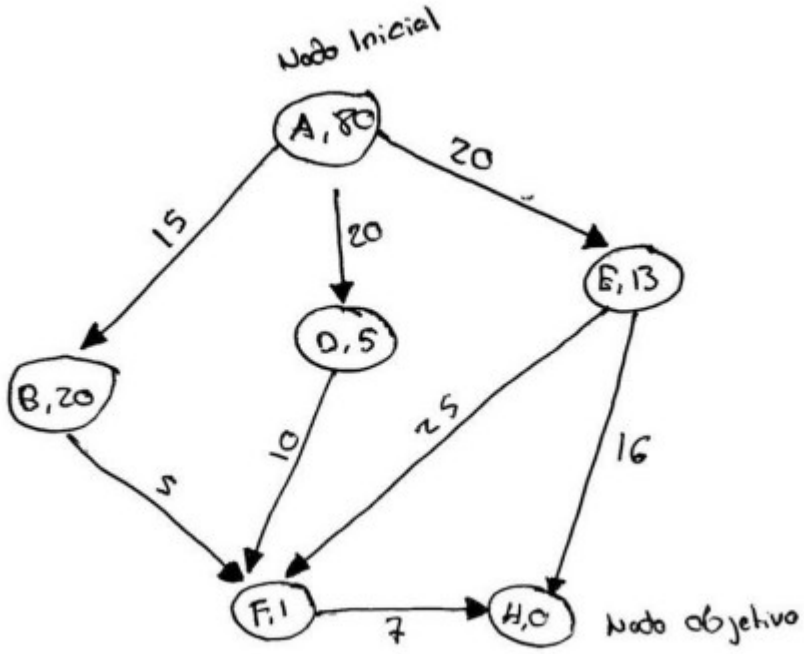
[DECIDIR]
- ➔ (AGT) → [PERSONA: #MARTA]
- ➔ (PASADO)
- ➔ (OBJ) → [PROPOSICION: [LEER]
  - ➔ (AGT)→ [PERSONA: #MARTA]
  - ➔ (OBJ) → [PERIODICO: {*}@2]]
- ➔ (CAUSA) → [PERIODICO: #]
  - ➔ (POS) → [PERSONA: #MARTA] → (ATR) → [SESGADO]

13. Pedro observa un par de noticias antiguas en un periódico, omitiendo las noticias actuales. (Sowa)

[OBSERVAR]
- ➔ (AGT) → [PERSONA: #PEDRO]
- ➔ (OBJ) → [PROPOSICION: [PAR]
  - ➔ (AGT) → [PERIODICO: #X]
  - ➔ (OBJ) → [NOTICIAS: {*} → (ATR) → [ANTIGUAS]]
- ➔ [PROPOSICION: [OMITIR]
  - ➔ (AGT) → [PERIODICO: #X]
  - ➔ (OBJ) → [NOTICIAS: {*} → (ATR) → [ACTUALES]

Ejercicios de Búsqueda

1. **Aplicación A***



a) **Antes de aplicar A*, ¿podrías asegurar si se obtendrá o no la solución óptima al aplicarlo?**

El método de búsqueda A* siempre asegura que si existe una solución la encuentra, dando como resultado el camino más óptimo.

b) **Aplica A***

Para cada iteración, representamos la frontera y el camino. Cada iteración debe indicar la siguiente información:

(Nombre, coste, heurística, f, mejorPadre)

Iteración	Frontera	Camino
0	[(A,0,80,80,-)]	[]
1	[(B,15,20,35,A),**(D,20,5,25,A)**, (E,20,13,33,A)]	[(A,0,80,80,-)]
2	[(B,15,20,35,A),(E,20,13,33,A), **(F,30,1,31,D)**]	[(A,0,80,80,-),(D,20,5,25,A)]
3	[(B,15,20,35,A),**(E,20,13,33,A)**, (H,37,0,37,F)]	[(A,0,80,80,-),(D,20,5,25,A), (F,30,1,31,D)]
4	[**(B,15,20,35,A)**,(H,36,0,36,E)]	[(A,0,80,80,-),(D,20,5,25,A),

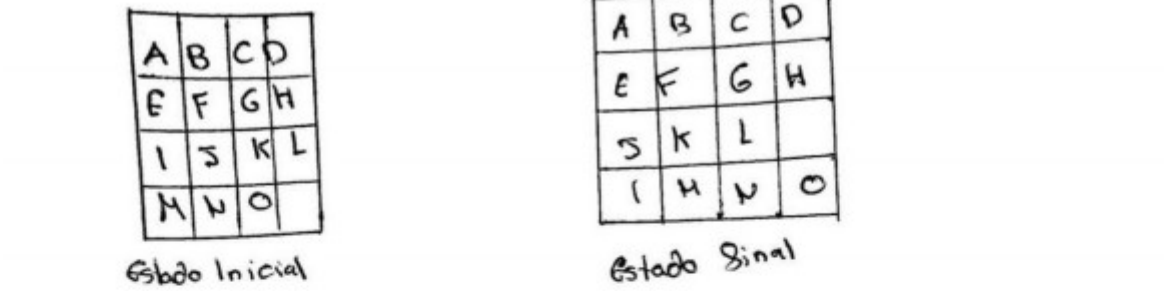
		(F,30,1,31,D), (E,20,13,33,A)]
5	[(H,27,0,27,F)]	[(A,0,80,80,-),(D,20,5,25,A), (F,20,1,21,B),(E,20,13,33,A), (B,15,20,35,A)]
6	[]	[(A,0,80,80,-),(D,20,5,25,A), (F,20,1,21,B),(E,20,13,33,A), (B,15,20,35,A),(H,27,0,27,F)]

Camino: H→ F→ B→ A

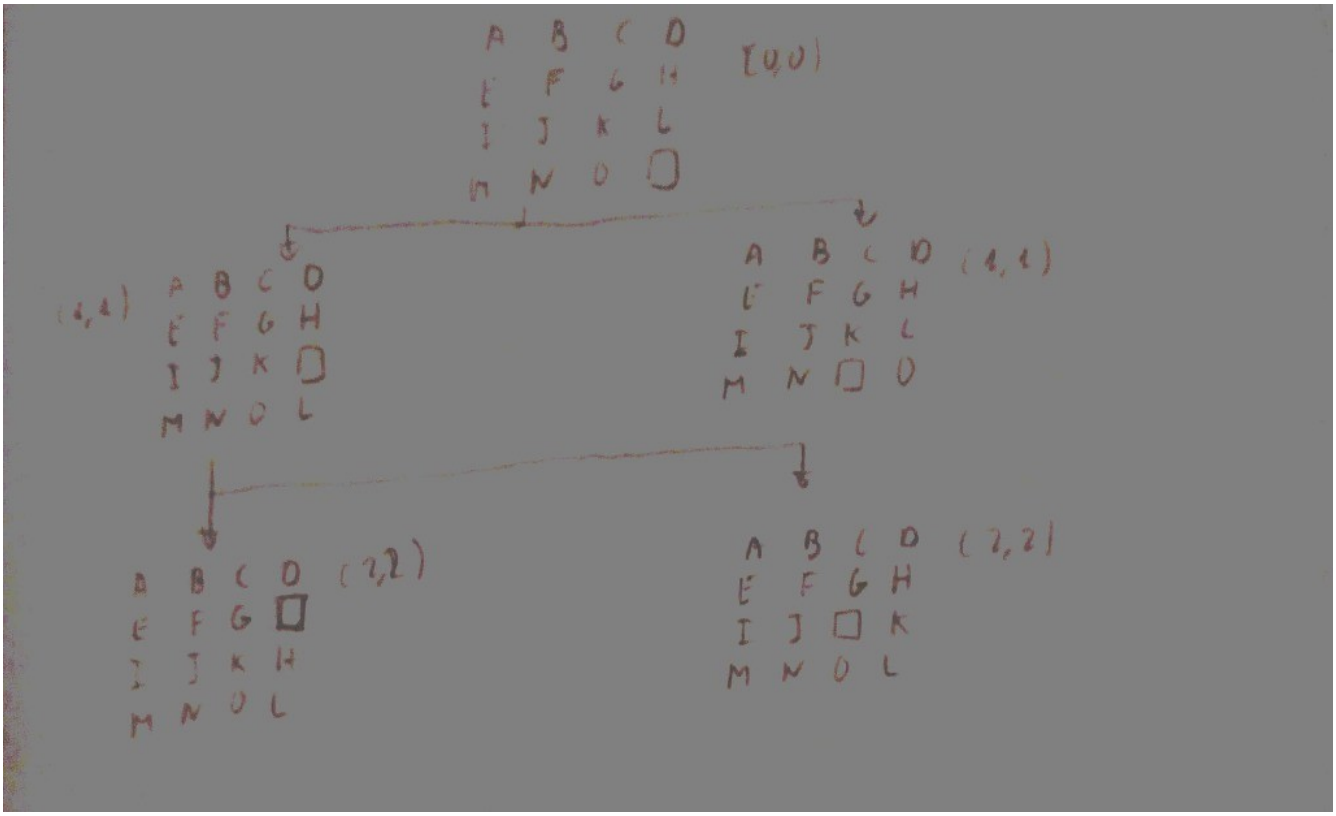
2. **Puzzle 15**
    a) **Aplica 6 iteraciones de la búsqueda en profundidad.**
    b) **Aplica 5 iteraciones de la búsqueda bidireccional en anchura**
    c) **Aplica 10 iteraciones de la búsqueda A***

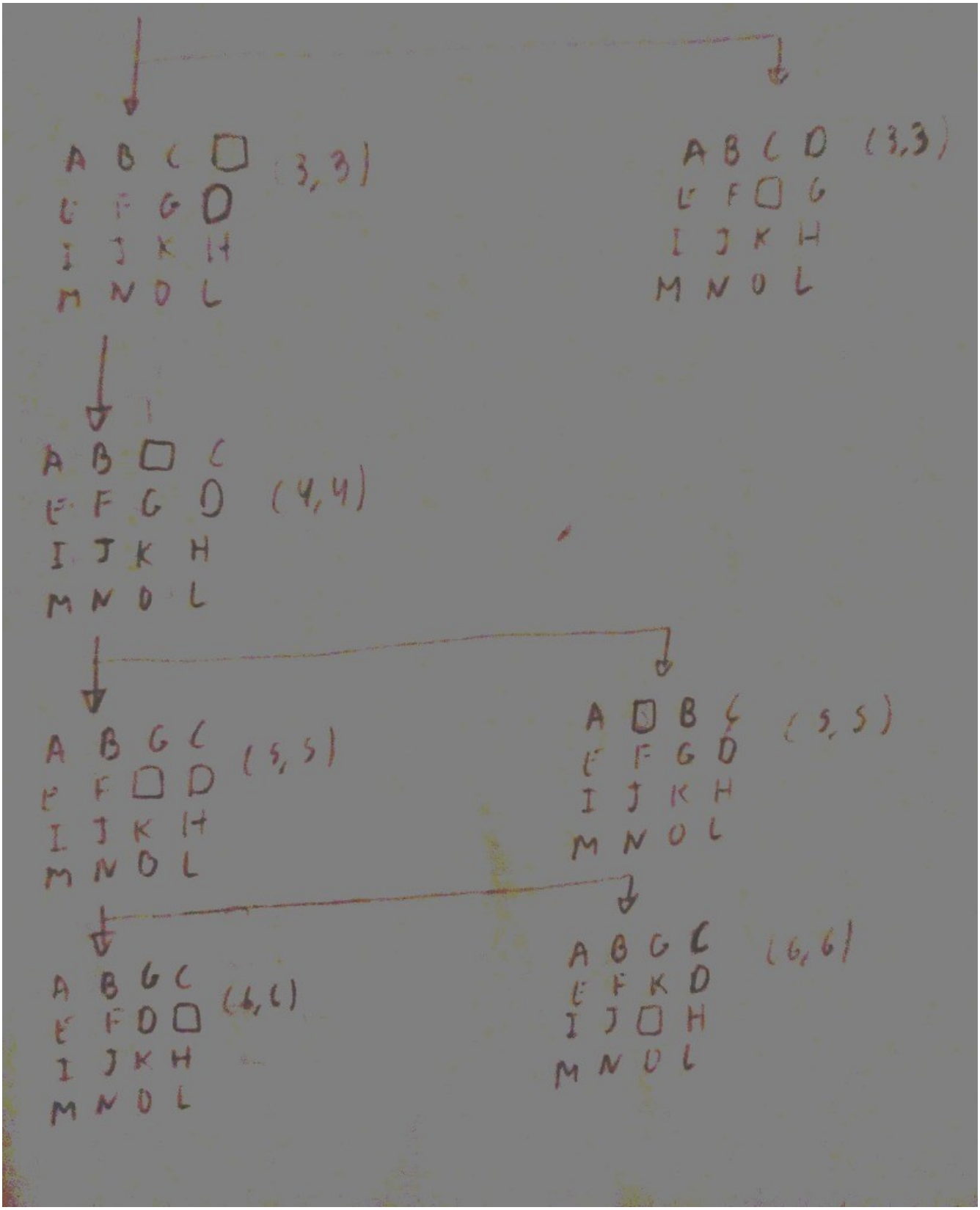
Reglas de producción:

1. **Arriba.**
2. **Derecha**
3. **Abajo**
4. **Izquierda**

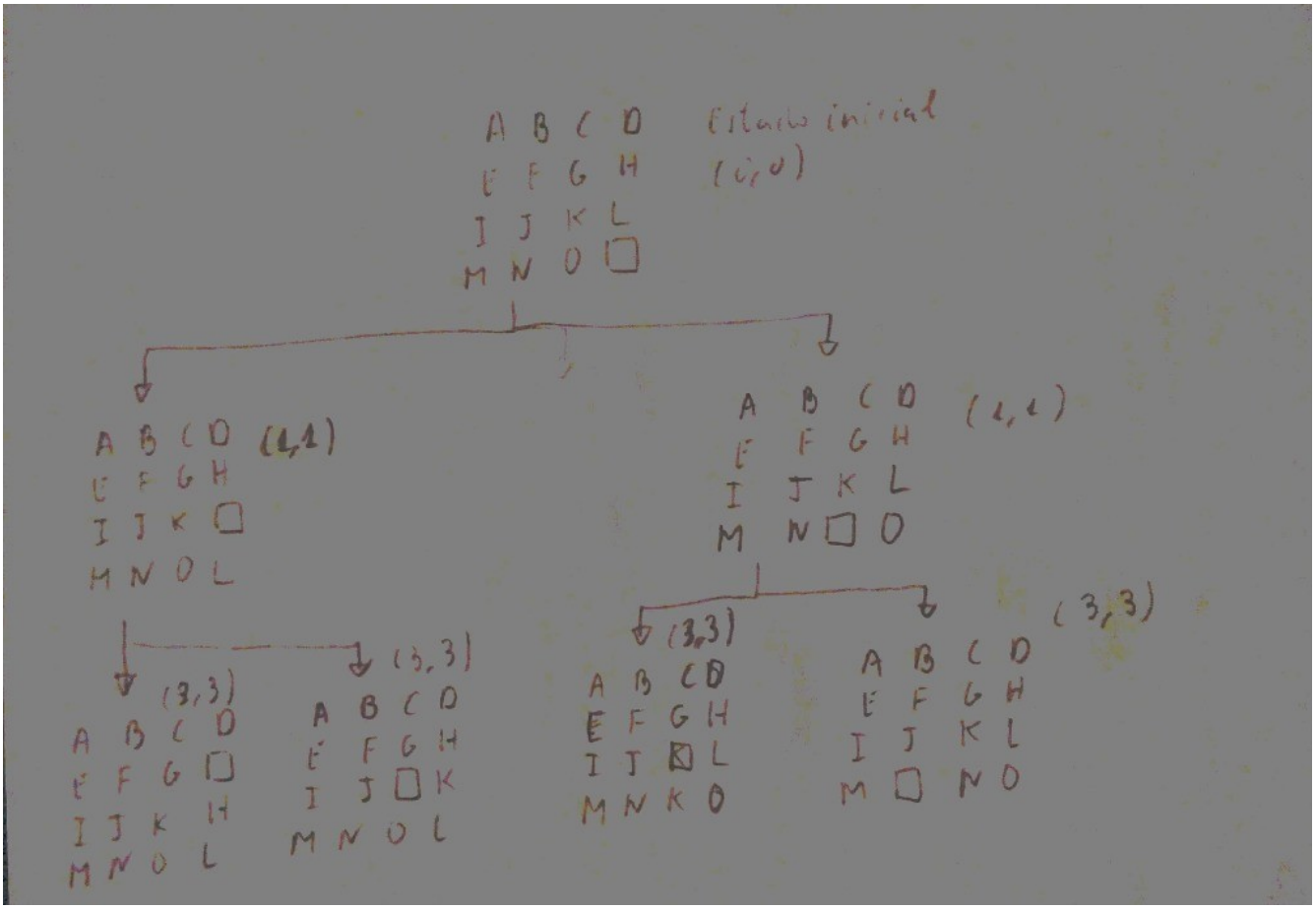


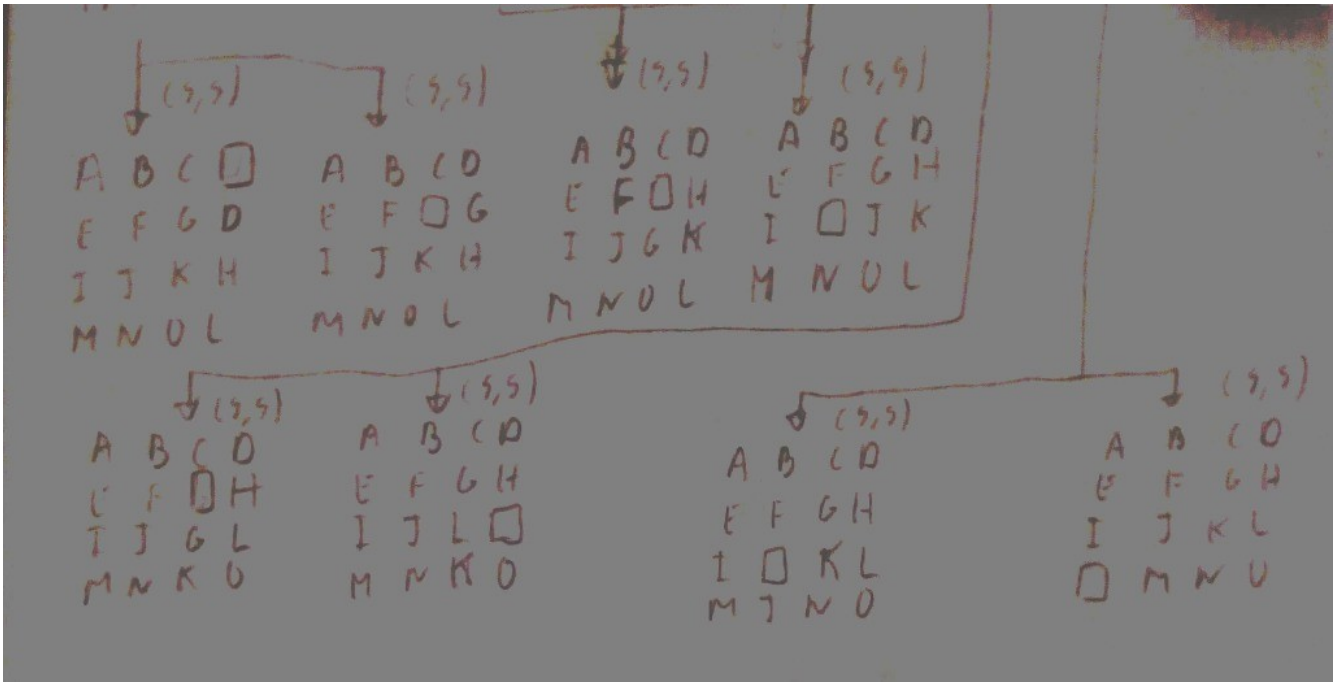
Apartado a

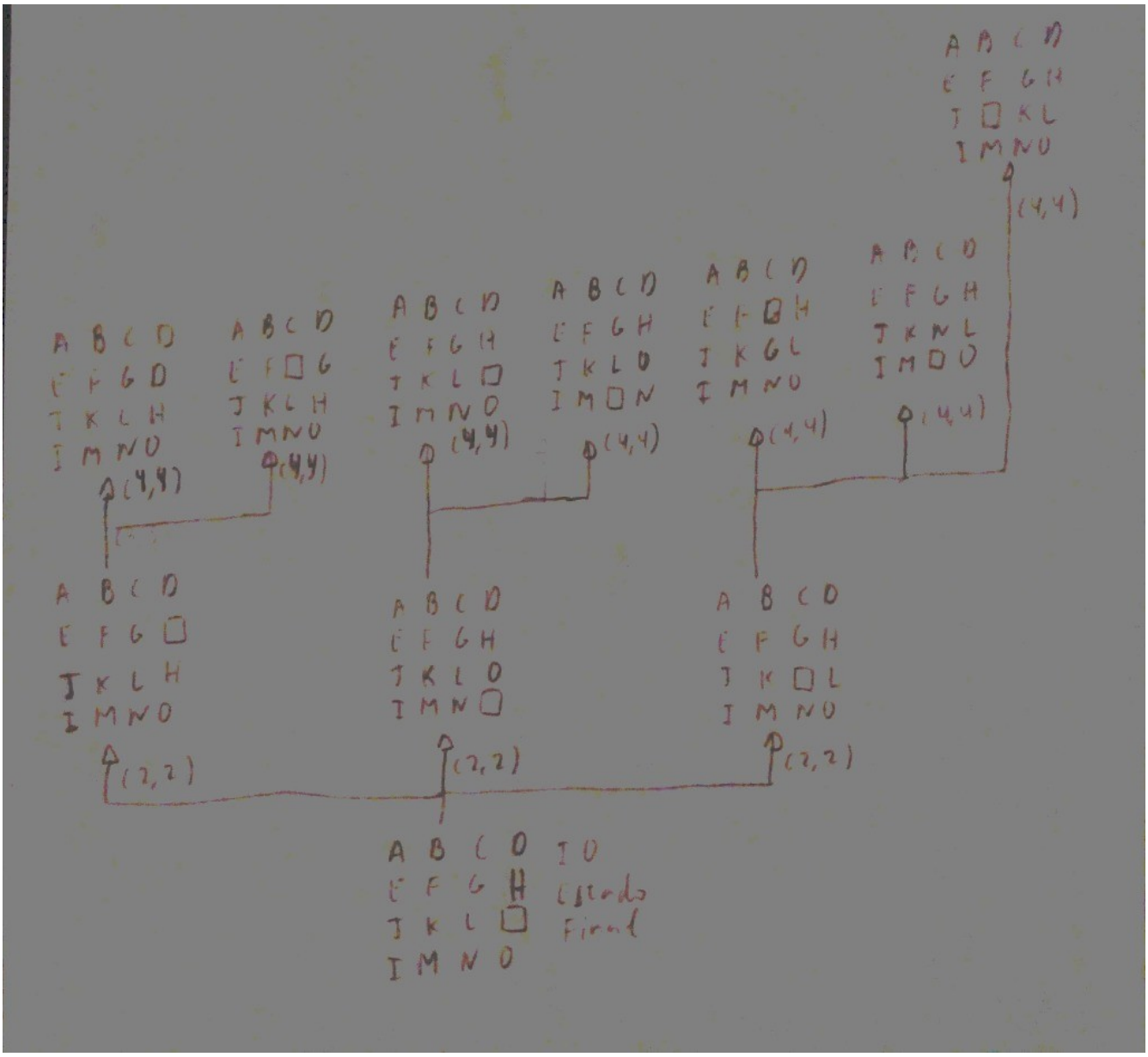

A B C D
E F G H
I J K L
M N O
(0,0)
(1,0)
A B C D
E F G H
I J K
M N O L
(1,1)
A B C D
E F G H
I J K L
M N O
(2,0)
A B C D
E F G
I J K H
M N O L
(2,1)
A B C D
E F G H
I J K L
M N O


A B C □
E F G D
I J K H
M N O L
(3,3)
A B C D
E F □ G
I J K H
M N O L
(3,3)
A B □ C
E F G D
I J K H
M N O L
(4,4)
A B G C
E F □ D
I J K H
M N O L
(5,5)
A □ B G
E F G D
I J K H
M N O L
(5,5)
A B G C
E F D □
I J K H
M N O L
(6,6)
A B G C
E F K D
I J □ H
M N O L
(6,6)

Apartado b

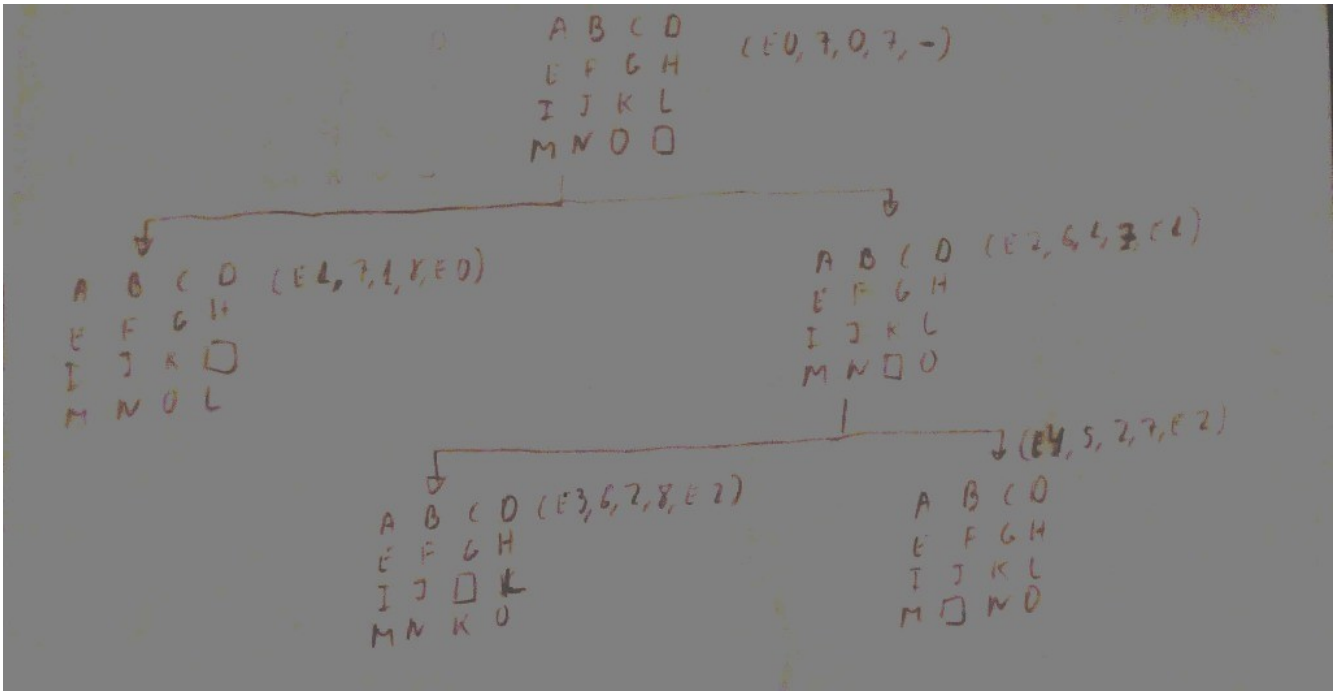


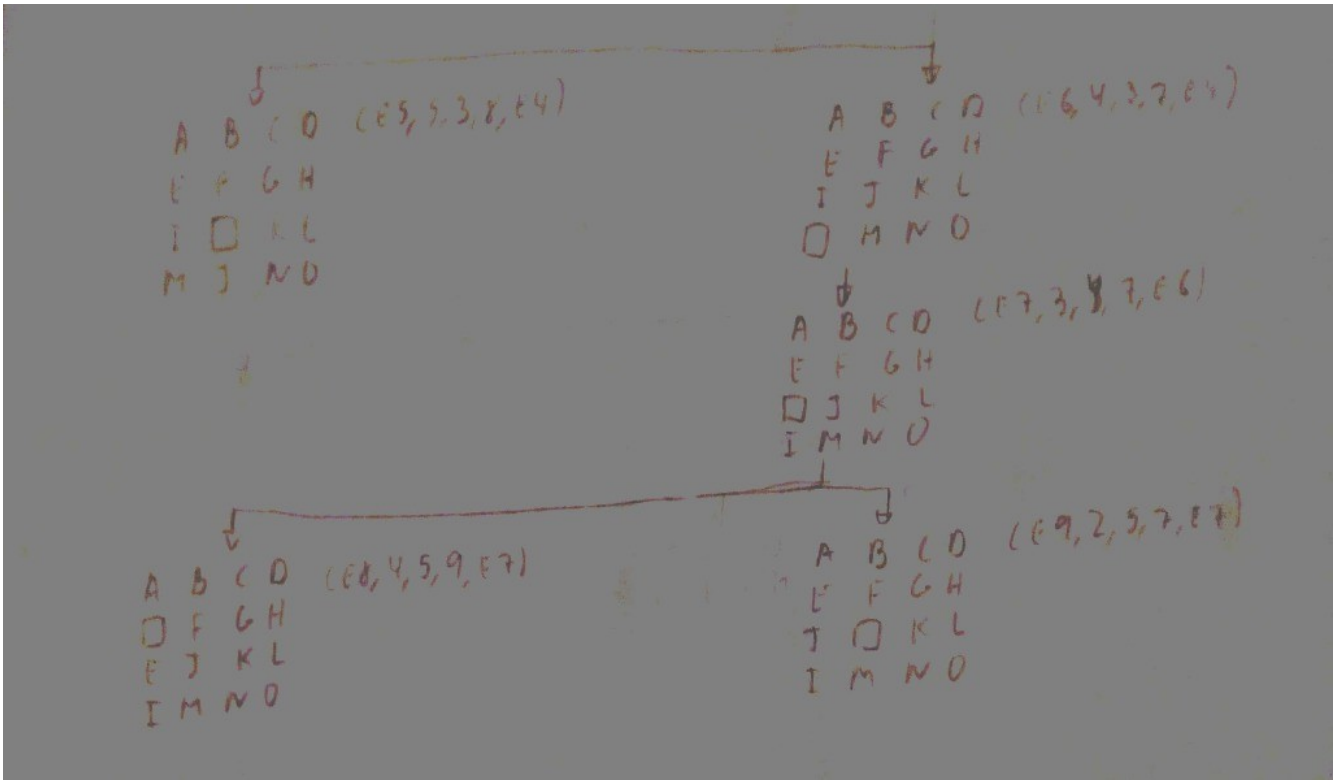



A B C D
E F G H
I J K L
I M N O
(4,4)
A B C D
E F G H
I J K L
I M N O
(4,4)
A B C D
E F G H
I J K L
I M N O
(4,4)
A B C D
E F G H
I J K L
I M N O
(4,4)
A B C D
E F G H
I J K L
I M N O
(4,4)
A B C D
E F G H
I J K L
I M N O
(2,2)
A B C D
E F G H
I J K L
I M N O
(2,2)
A B C D
E F G H
I J K L
I M N O
(2,2)
A B C D
E F G H
I J K L
I M N O
Estado Final

Apartado c

Cada iteración dará la siguiente información: (Estado,Coste, Heurística, f, mejorPadre)





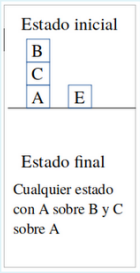
3.

En el mundo de los bloques se tiene una mesa y unos bloques que se encuentran bien encima de la mesa o bien encima de otro bloque formando pilas de bloques (ver imagen abajo). Sólo hay dos tipos de operaciones:

1. Coger un bloque que no tenga nada encima y ponerlo encima de otro cualquiera (no es necesario que sea contiguo)
2. Coger un bloque que no contenga nada encima y ponerlo sobre la mesa a la derecha de todas las pilas de bloques (no es necesario que dicho bloque esté en la pila más a la derecha)

Entre las posibles operaciones a aplicar en un estado, tiene prioridad el coger bloques de izquierda a derecha y situarlos en pilas de bloques, o la mesa, también de izquierda a derecha.

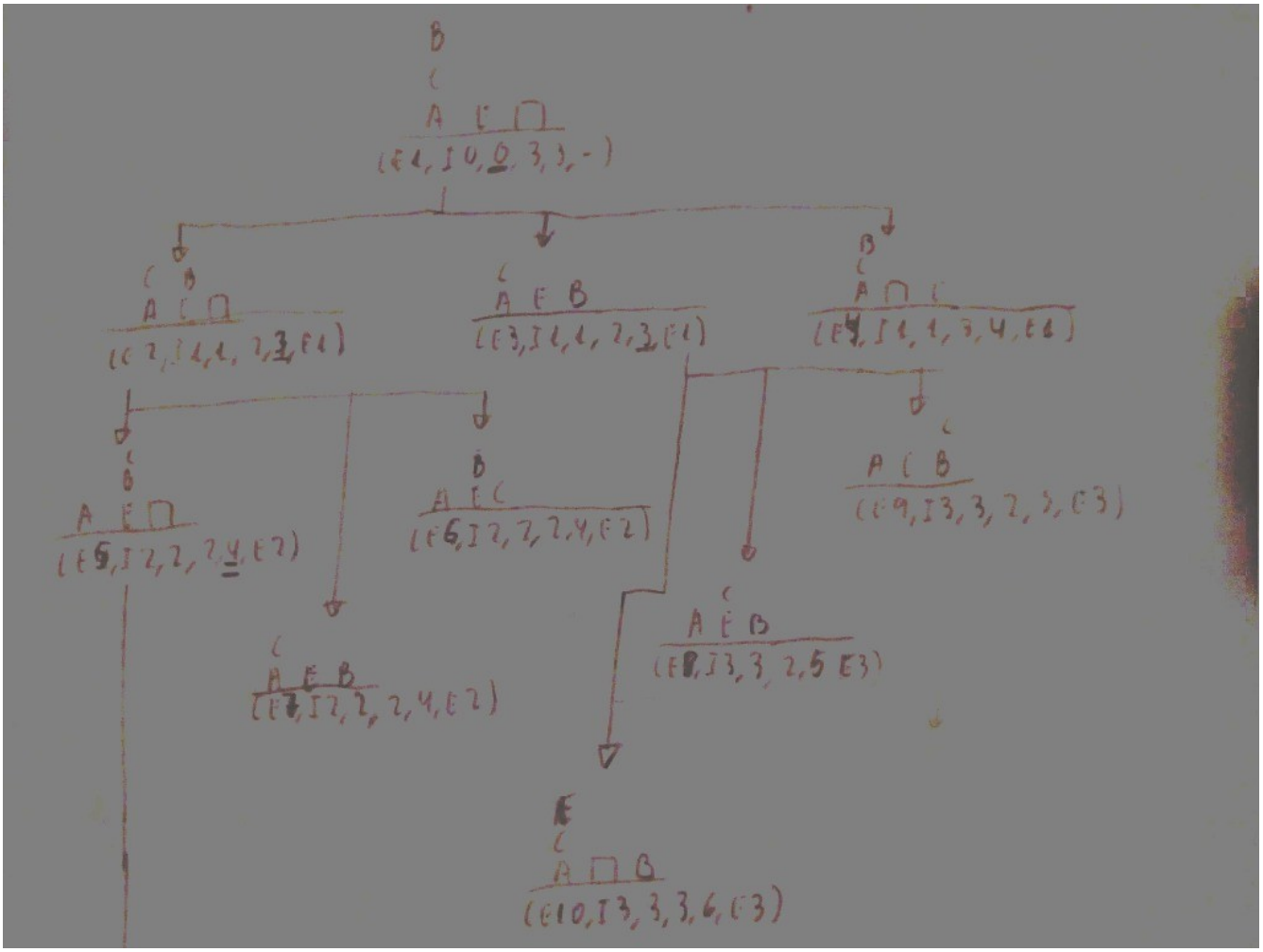
Sean los siguientes estados inicial y objetivo.

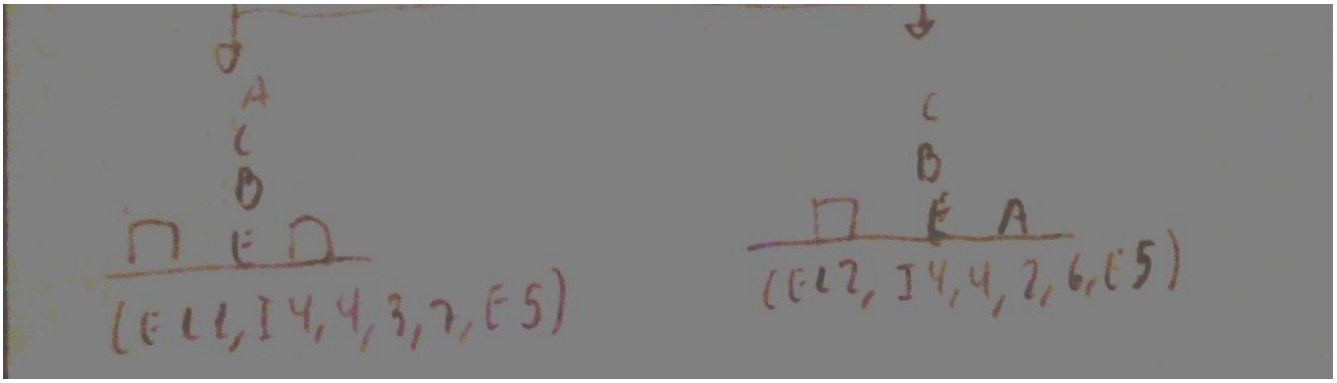


Para la evaluación de si se ha alcanzado el estado objetivo o no, sólo se comprueba si los bloques están encima de aquello en lo que deben estar. El orden horizontal (izquierda a derecha) de las pilas de bloques es irrelevante.

Asumiendo que el estado inicial está generado en la iteración 0, aplica 4 iteraciones de la búsqueda A*, dibujando el árbol de búsqueda y usando como función heurística la suma, para cada bloque mal colocado (no encima de donde tiene que estar), de uno más el número de bloques que tiene encima apilados. Por ejemplo, en el estado inicial anterior, ni A, ni C, están sobre lo que tienen que estar, por lo que se suma 3 + 2 = 5. Notas:

- En caso de empate, selecciona el estado más a la izquierda en el árbol de búsqueda.
- Cerrar un nodo sin hijos cuesta una iteración.
- NO SE DEBEN GENERAR ESTADOS SI NO ES APLICANDO EL MÉTODO DE BÚSQUEDA
- No se deben generar estados repetidos, es decir, estados en los que todos los bloques estén sobre los mismos otros bloques o la mesa.
- En el árbol de búsqueda, se debe indicar junto a cada estado, y en este orden: un nombre inventado para el estado (E1,E2,E3....), la iteración en la que el estado se genera, el coste del estado, el valor heurístico, la suma de los dos valores anteriores y el nombre del estado mejor padre.

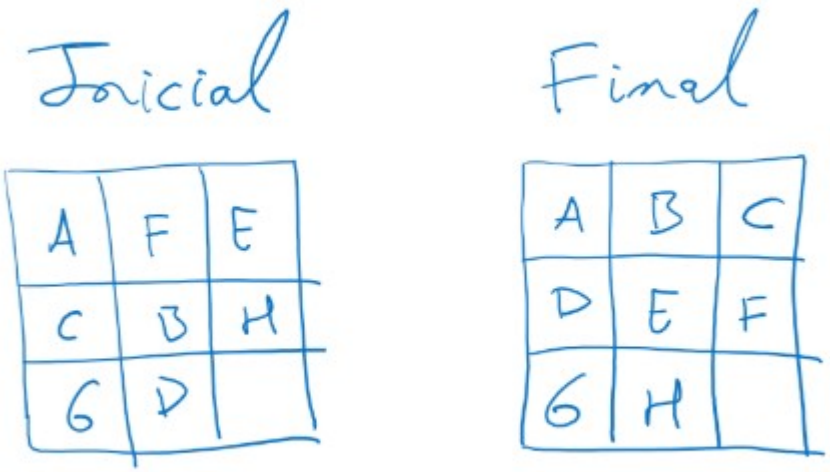

B
A E
(E4, I0, 0, 3, 3, -)
C B
A C
(E7, I4, 1, 2, 3, E1)
C
A E B
(E3, I1, 1, 2, 3, E1)
B
A C
(E4, I1, 1, 3, 4, E1)
B
A E
(E5, I2, 2, 2, 4, E2)
B
A C
(E6, I2, 2, 2, 4, E2)
C
A C B
(E9, I3, 3, 2, 3, E3)
C
A E B
(E7, I2, 2, 2, 4, E2)
C
A E B
(E8, I3, 3, 2, 5 E3)
E
A B
(E10, I3, 3, 3, 6, E3)


A
B
E
(E11, I4, 4, 3, 7, E5)
C
B
E A
(E12, I4, 4, 2, 6, E5)

4. **En el 8-puzzle se tiene un tablero de 3x3 casillas en las que hay colocadas 8 fichas (siempre hay una casilla vacía o hueco). Sólo hay un único movimiento posible que es mover una de las fichas adyacentes a la casilla vacía a dicha casilla, dejando la casilla que ocupaba anteriormente vacía.**

   Entre las posibles operaciones a aplicar en un estado, se debe dar prioridad a las operaciones que sitúen al hueco 1. arriba, 2. abajo, 3. izquierda, 4 derecha, de la casilla que ocupa al inicio de la operación.

   Sean los siguientes estados inicial y objetivo

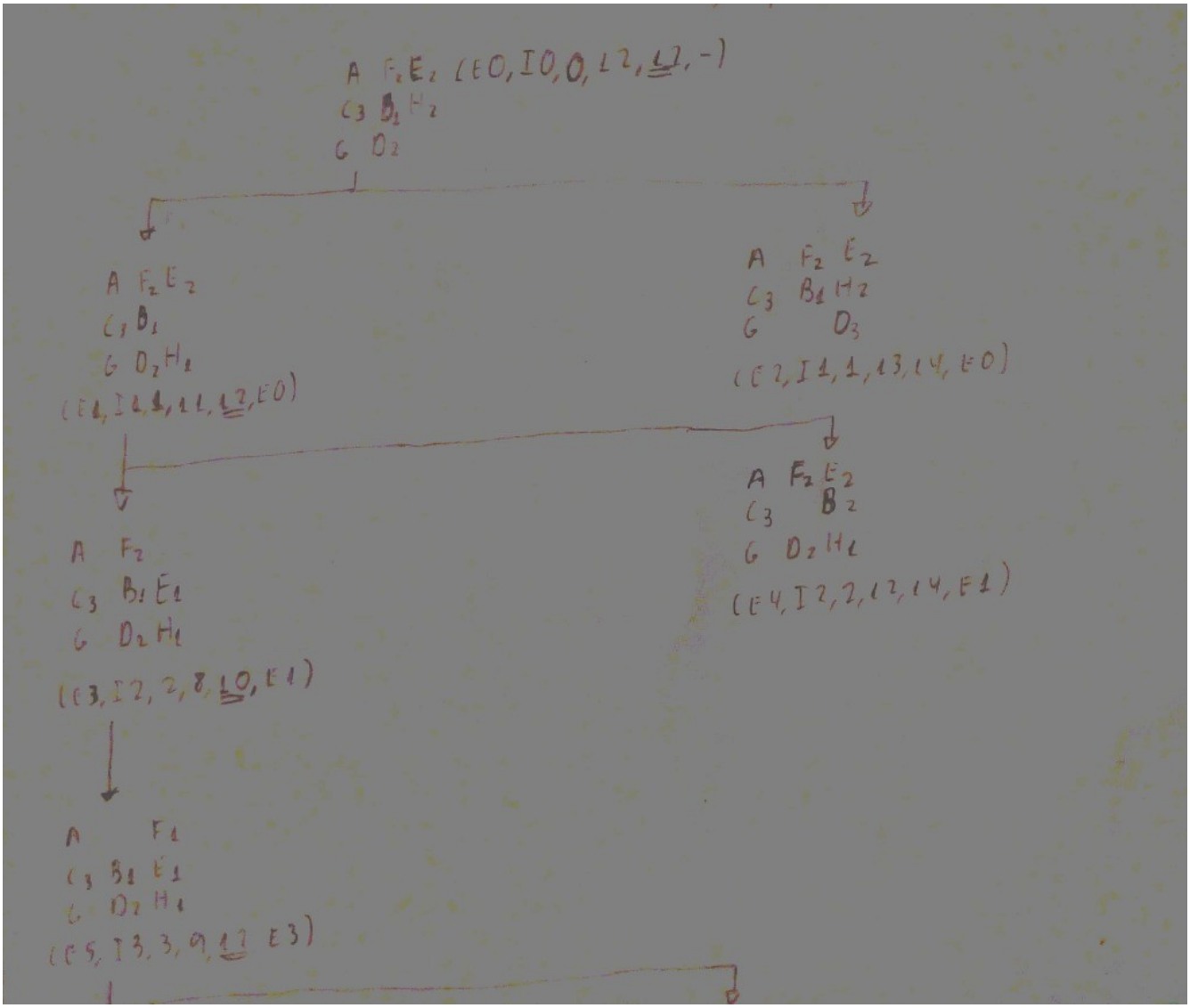


   Asumiendo que el estado inicial está generado en la iteración 0, aplica 4 iteraciones de la búsqueda A*, dibujando el árbol de búsqueda y usando como función heurística la suma de las distancias de manhattan de cada ficha mal colocada hasta su casilla objetivo (la distancia de manhattan es la suma de las distancias horizontal y vertical entre la ficha y la posición objetivo).

   En caso de empate, selecciona el estado más a la izquierda en el árbol de búsqueda.

   - Cerrar un nodo sin hijos cuesta una iteración.
   - NO SE DEBEN GENERAR ESTADOS SI NO ES APLICANDO EL MÉTODO DE BÚSQUEDA
   - No se deben generar estados repetidos, es decir, estados en los que todos los bloques estén sobre los mismos otros bloques o la mesa.

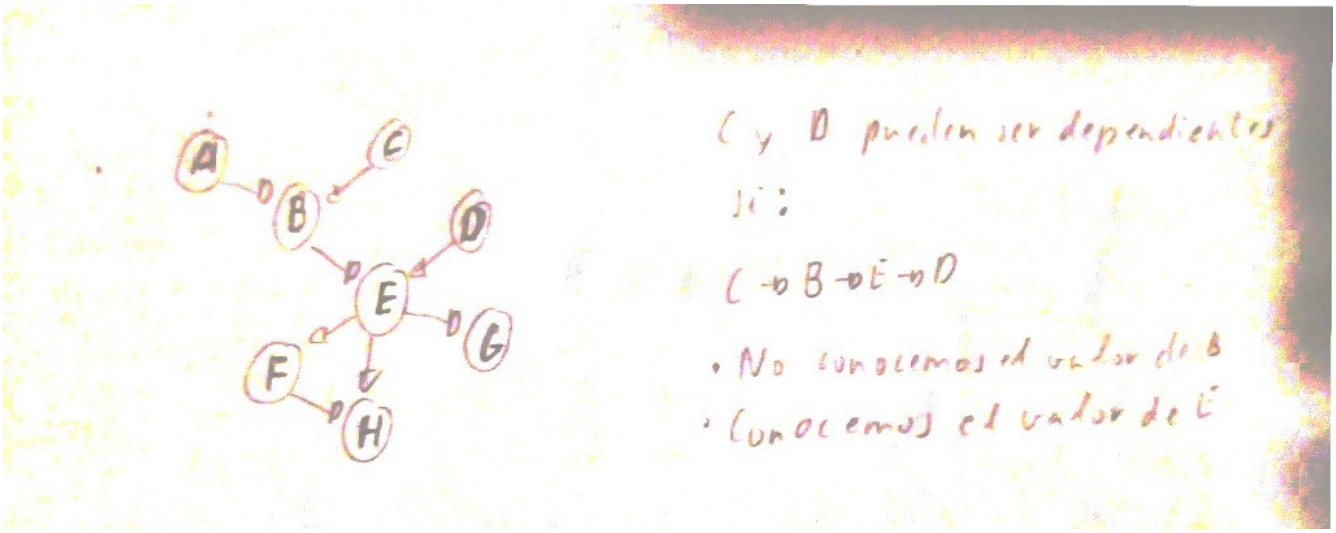
- En el árbol de búsqueda, se debe indicar junto a cada estado, y en este orden: un nombre inventado para el estado (E1,E2,E3....), la iteración en la que el estado se genera, el coste del estado, el valor heurístico, la suma de los dos valores anteriores y el nombre del estado mejor padre.



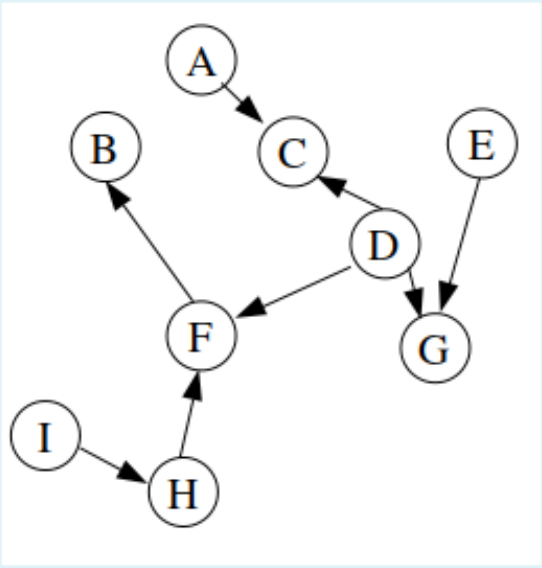
Redes bayesianas

Dibuja la red bayesiana asociada a las siguientes distribuciones de probabilidad conocidas, e indica en qué situaciones pueden C y D ser **dependientes**:

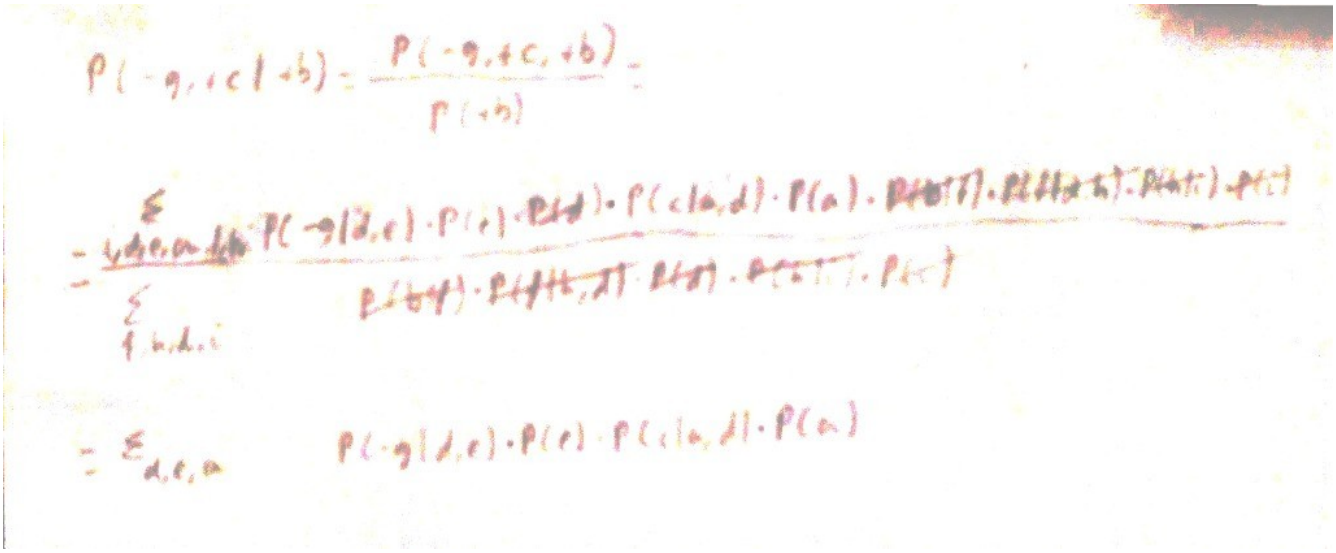
- P(A)
- P(B|A,C)
- P(C)
- P(D)
- P(E|B,D)
- P(F|E)
- P(G|E)
- P(H|E,F)



Para el siguiente grafo de una red bayesiana, responda a las siguientes preguntas.



1. Calcule la probabilidad P(-g,+c|+b)
2. Indica en qué condiciones E y A son independientes
3. Si B y C son conocidos, ¿qué nodos son independientes de H?



E y A son independendientes si se cumple lo siguiente:

O bien C no se conoce, o G no se conoce, o D se conoce.

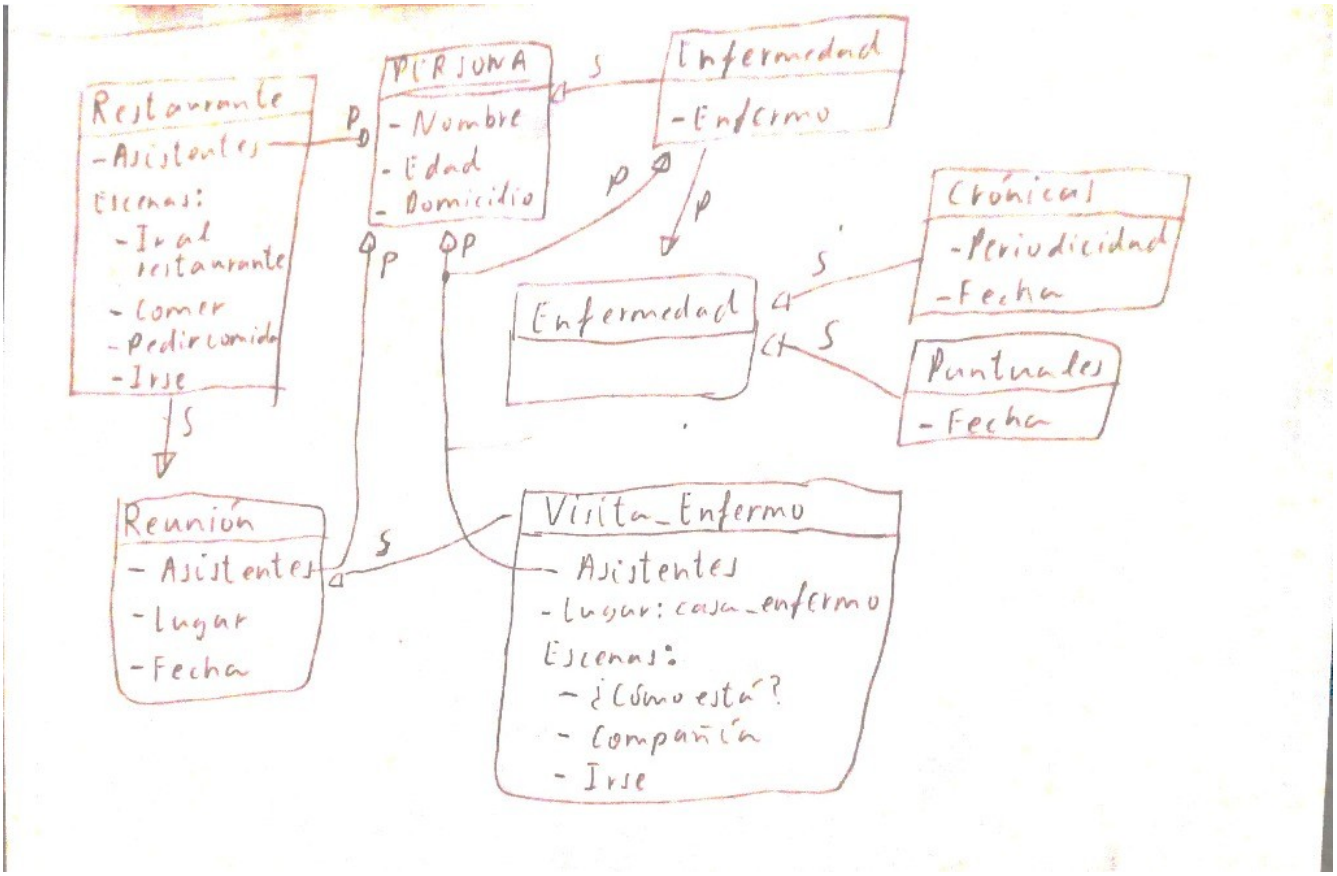
Si B y C son conocidos:

Sólo el nodo E es independiente de H

Ejercicio Marcos

1. **Representa con marcos y/o guiones la siguiente información:**

   Toda persona tiene un nombre, una edad y un domicilio. Un enfermo es una persona que padece una enfermedad. Las enfermedades pueden ser puntuales o croónicas. Las enfermedades puntuales ocurren en una fecha. Las crónicas además tienen una periodicidad. Todo tipo de reunión tiene unos asistentes, un lugar de reunión y una fecha. Cuando se visita a un enfermo crónico, que es un tipo de reunión, unos de los asistentes van a la casa del enfermo, que es otro asistente, se le pregunta como está, se hace compañía y se va. Cuando se va a un restaurante, que es un tipo de reunión, los asistentes van al restaurante, piden comida, comen y se van.



Ejercicio Lógica

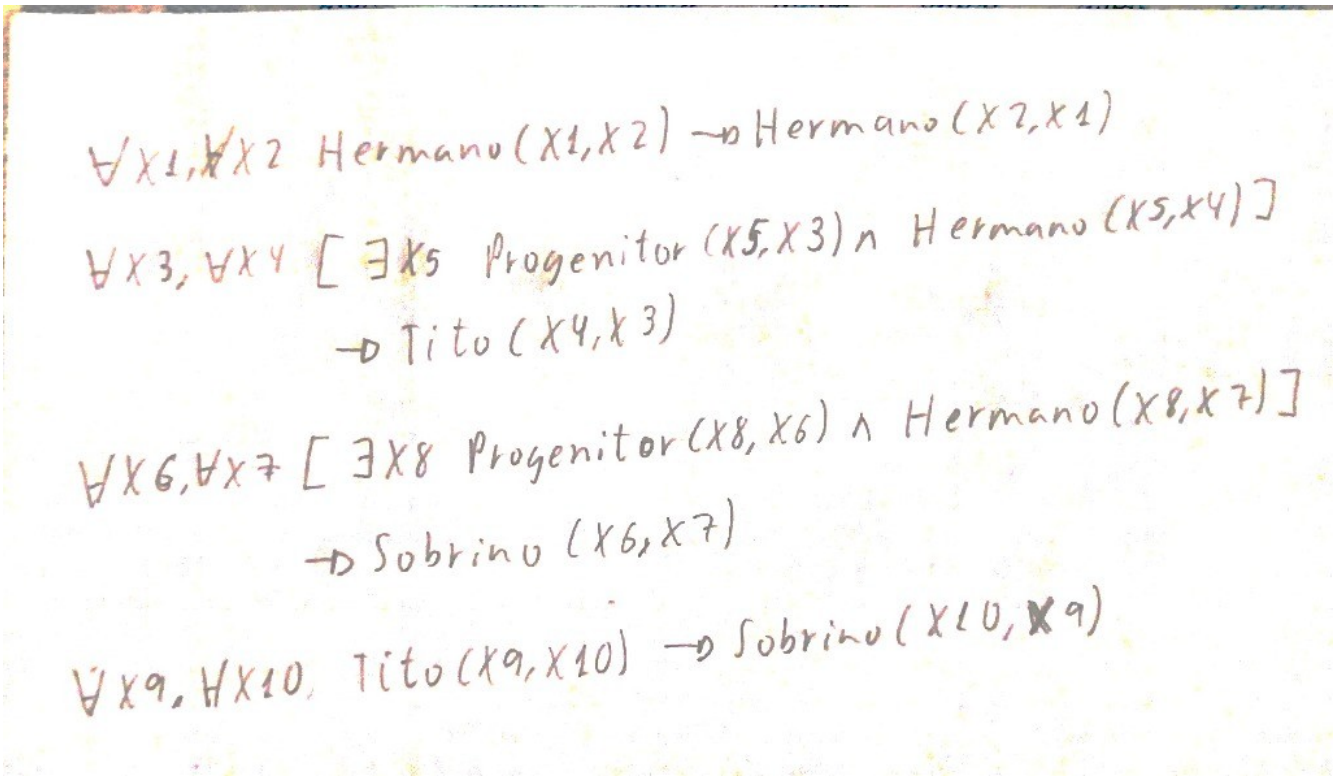
Sean las siguientes hipótesis que debes representar con lógica de predicados:

- Hermano es una relación conmutativa, es decir, "para todo x1, x2 si x1 es hermano de x2, entones x2 es hermano de x1"
- Tito es el hermano de mi progenitor, es decir, "para todo x3, x4, si existe x5 que es progenitor de x3 y hermano de x4, entonces x4 es tito de x3"
- Sobrino es el el descendiente de mi hermano, es decir, "para todo x6, x7, si existe x8 que es progenitor de x6 y hermano de x7, entonces x6 es sobrino de x7

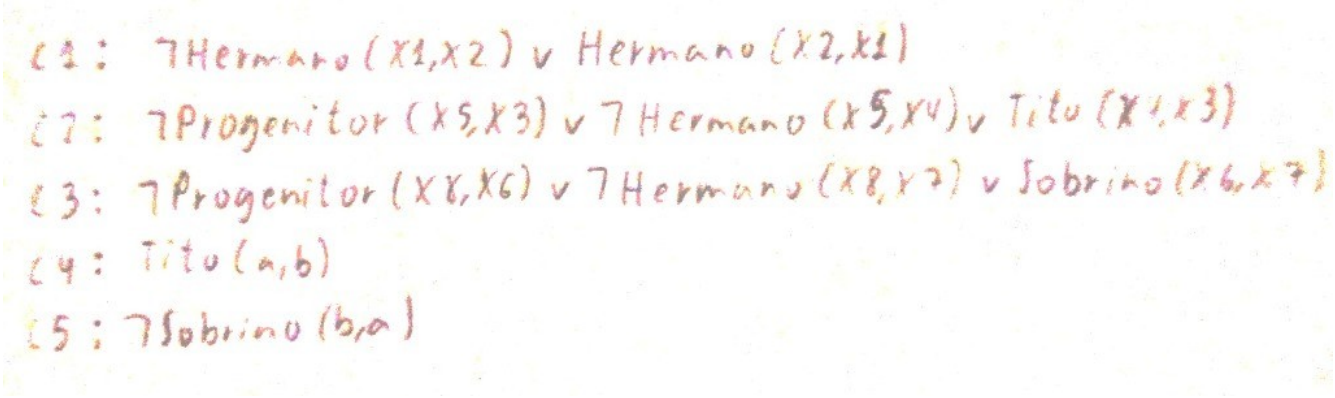
Demuestra, a través de los pasos indicados a continuación, que tito y sobrino son relaciones inverdas, es decir, "para todo x9, x10, si x9 es tito de x10, entonces x10 es sobrino de x9. Para ello, diferencia tres secciones en una imagen de una cara de un folio (aproximadamente) y rellena las secciones con:

1. Sección 1: La representación en lógica de predicados de las hipótesis y de la conclusión.
2. Sección 2: El resultado y sólo el resultado de haber pasado a forma normal conjuntiva las hipótesis y la negación de la conclusión a demostrar.
3. Sección 3: el proceso de resolución indicadndo, paso a paso, qué cláusulas unificas para aplicar el principio de resolución, junto con las sustituciones de variables que realices, y el resultado de cada aplicación del principio de resolución.

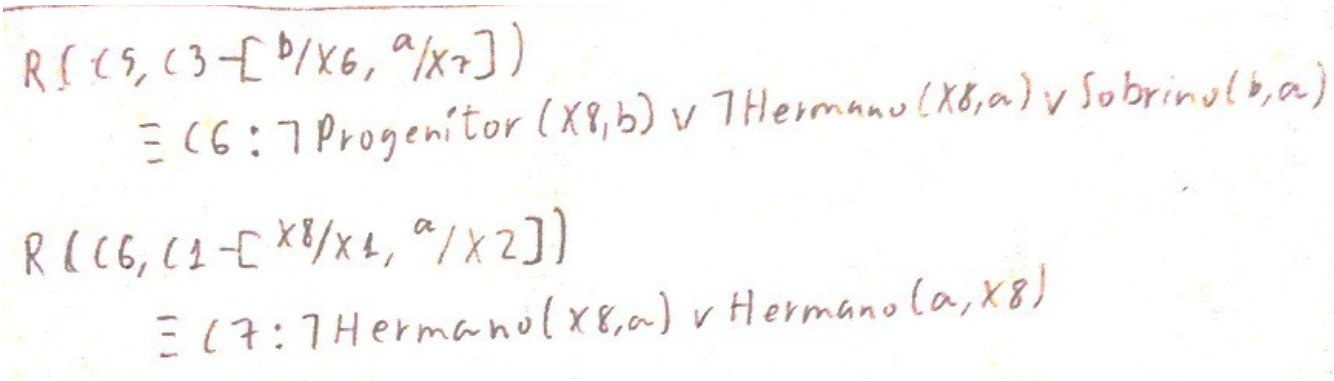
SECCIÓN 1



SECCIÓN 2



SECCIÓN 3



Como no hemos llegado a una contradicción, no podemos demostrar la conclusión a partir de las hipótesis dadas.

Sean las siguientes hipótesis que deberéis representar con lógica de predicados:

- El hermano del progenitor de una persona es su tito. Para simplificar, representa que "para todo par x1, x3, si existe x2 que es progenitor de x1 y x3 es hermano de x2, entonces x3 es tito de x1"
- Descendiente y progenitor son relaciones inversas. Es decir, para todo par x5, x6, si x5 es descendiente de x6, entonces x6 es progenitor de x5, y viceversa.
- Juan es hermano de Aurora
- Anabel es descendiente de Aurora

Demuestra, a través de los pasos indicados a continuación, que Juan es tito de Anabel.

Para ello, diferencia tres secciones en una imagen de una cara de un folio (aproximadamente), y rellena las secciones con:

1. Sección 1: La representación en lógica de predicados de las hipótesis y de la conclusión.
2. Sección 2: El resultado (sólo el resultado) de haber pasado a forma normal conjuntiva las hipótesis y la negación de la conclusión a demostrar.
3. Sección 3: El proceso de resolución indicando, paso a paso, qué cláusulas unificáis para aplicar el principio de resolución, junto con las sustituciones de variables que realices, y el resultado de cada aplicación del principio de resolución.

Sección 1

F1: $\forall x1 \forall x3 [\exists x2 \text{ Progenitor}(x2, x3) \wedge \text{Hermano}(x3, x2)] \rightarrow \text{Tito}(x3, x1)$

F2: $\forall x5 \forall x6 \text{ Descendiente}(x5, x6) \leftrightarrow \text{Progenitor}(x6, x5)$

F3: Hermano (Juan, Aurora)

F4: Descendiente (Anabel, Aurora)

F5: Tito (Juan, Anabel)

Sección 2

C1: $\neg$ Progenitor $(x2, x3) \vee \neg$ Hermano $(x3, x2) \vee$ Tito $(x3, x1)$

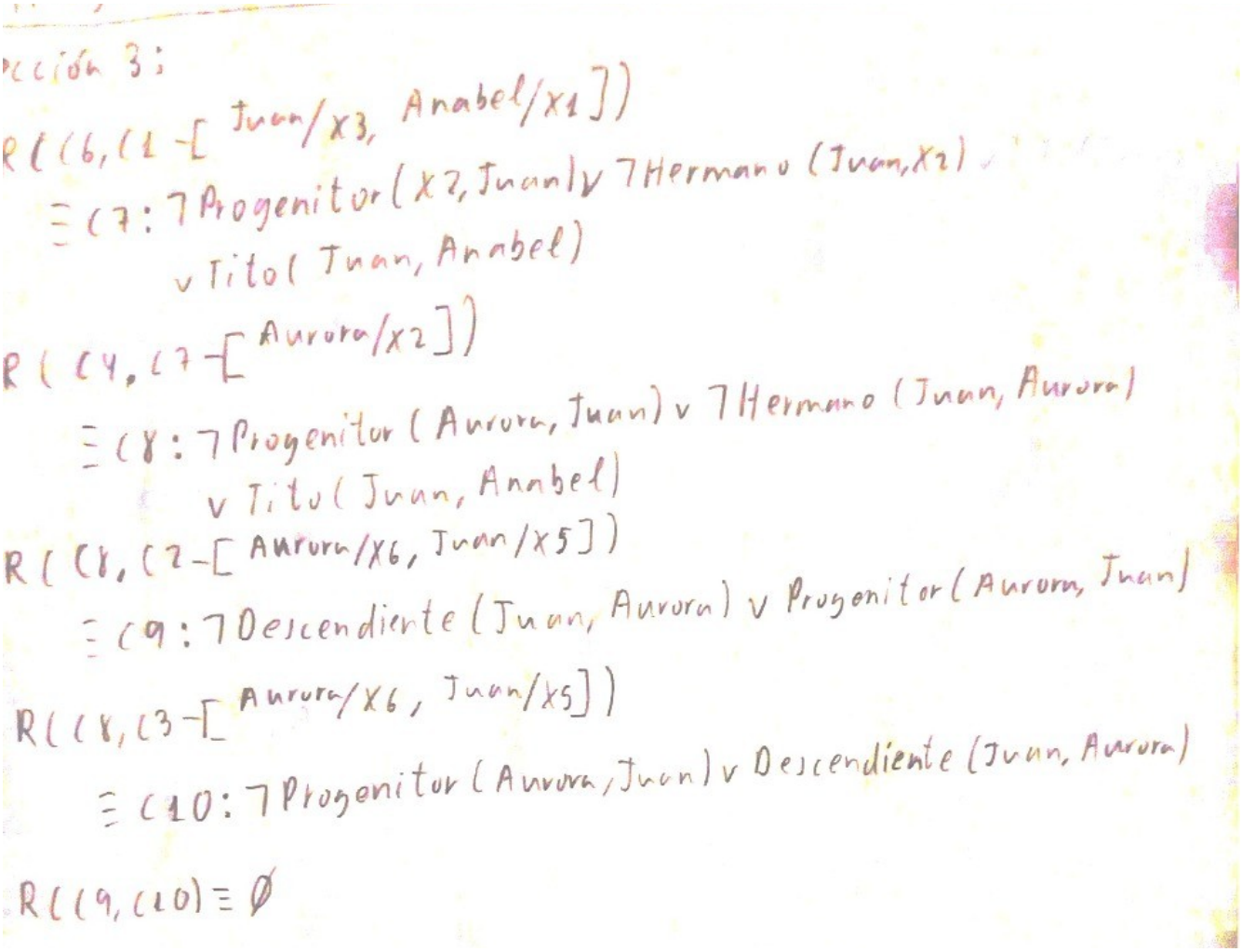
C2: $\neg$ Descendiente $(x5, x6) \vee$ Progenitor $(x6, x5)$

C3: $\neg$ Progenitor $(x6, x5) \vee$ Descendiente $(x5, x6)$

C4: Hermano (Juan, Aurora)

C5: Descendiente (Anabel, Aurora)

C6: $\neg$ Tito (Juan, Anabel)



Como hemos llegado a una contradicción, hemos demostrado que Juan es tito de Anabel.

Ejercicio Prácticas

1. Define una plantilla vector con un campo nombre, y un campo valores, restringido a enteros y con cardinalidad entre 0 y 100. Por ejemplo:
   (vector (nombre v1) (valores 1 2 3 4 5 6 7))
   (vector (nombre v2) (valores 1 2 4 6 8))
   (vector (nombre v3) (valores 2 2 2 2 2 3 2))
   (vector (nombre v4) (valores 1))

```
(deftemplate vector (slot nombre)(multislot valores (type INTEGER)(cardinality 0 100)))

    (deffacts vectores
        (vector (nombre v1) (valores 1 2 3 4 5 6 7))
        (vector (nombre v2) (valores 1 2 4 6 8))
        (vector (nombre v3) (valores 2 2 2 2 2 3 2))
        (vector (nombre v4) (valores 1)))
```

2. Crea un conjunto de reglas tal que dado hechos vectores como los siguientes:

```
(vector (nombre v1) (valores 1 2 3 4 5 6 7))
(vector (nombre v2) (valores 1 2 4 6 8))
(vector (nombre v3) (valores 2 2 2 2 2 3 2))
(vector (nombre v4) (valores 1))
```

   Muestre por pantalla el nombre de aquellos vectores que tienen alguno de sus elementos repetido al menos tres veces.

```
    (defrule repetido

    (vector (nombre ?n) (valores $?))

    (exists (vector (nombre ?n) (valores $? ?y $? ?y $? ?y $?)))

    =>

    (printout t ?n " tiene valores repetidos al menos 3 veces" crlf))
```

3. Crea las reglas necesarias que permitan guardar en un contador único la suma de los valores de cada vector

```
(defrule SumaContador

    ?f1 <- (vector (nombre ?x) (valores $?antes ?y $?despues))

    ?f2 <- (suma ?x ?contador)

    =>

    (retract ?f1)

    (retract ?f2)
```

```
        (assert (vector (nombre ?x)(valores $?antes $?despues)))
        (assert (suma ?x (+ ?contador ?y))))
(defrule crearContadorSuma
        (vector (nombre ?x))
        (not (suma ?x ?))
        =>
        (assert (suma ?x 0)))
```

4. Crea una regla que indique aquellos vectores tales que un valor sea obtenido por la suma de otros 2 valores.

```
        (deffacts vectores
                (vector (nombre v1) (valores 1 2 3 4 5 6 7))
                (vector (nombre v2) (valores 1 2 4 6 8))
                (vector (nombre v3) (valores 2 2 2 2 2 3 2))
                (vector (nombre v4) (valores 1))
                (eliminar 4))
(defrule seleccionarVector
        (vector (nombre ?n)(valores $?antes ?a ?b ?c $?despues))
        (test (= ?c (+ ?a ?b)))
        =>
        (printout t ?n " : " ?c " = " ?a " + " ?b crlf))
```

5. Crea un conjunto de reglas tal que dado el hecho `(eliminar <valor>)`, elimine de todos los vectores las apariciones del valor indicado. Por ejemplo, si tuviésemos los hechos

```
(vector (nombre v1) (valores 1 2 3 4 5 6 7))
(vector (nombre v2) (valores 1 2 4 6 8))
(vector (nombre v3) (valores 2 2 2 2 2 3 2))
(vector (nombre v4) (valores 1))
(eliminar 2)
```

Entonces la base de hechos resultante tras ejecutar el programa sería: `(vector (nombre v1) (valores 1 3 4 5 6 7))`

```
(vector (nombre v2) (valores 1 4 6 8))
(vector (nombre v3) (valores 3))
(vector (nombre v4) (valores 1))
(eliminar 2)
```

(deffacts vectores

(vector (nombre v1) (valores 1 2 3 4 5 6 7))

(vector (nombre v2) (valores 1 2 4 6 8))

(vector (nombre v3) (valores 2 2 2 2 2 3 2))

(vector (nombre v4) (valores 1))

(eliminar 4))

(defrule eliminarValor

(eliminar ?x)

?h <- (vector (nombre ?n)(valores $?cabeza ?x $?cola))

=>

(modify ?h (valores $?cabeza $?cola)))

6. Crea un conjunto de reglas que obtenga el elemento de en medio de vectores con un número impar de elementos, o la media de los dos elementos de en medio de vectores con un número par de elementos. Por ejemplo, para los vectores:

```
(vector (nombre v1) (valores 1 2 3 4 5 6 7))
(vector (nombre v2) (valores 1 2 4 6 8))
(vector (nombre v3) (valores 2 2 2 2 3 2))
(vector (nombre v4) (valores 1)
```

Debe imprimir:

```
"v1 tiene un número impar de valores con 4 en medio"
"v2 tiene un número impar de valores con 4 en medio"
"v3 tiene un número par de valores con 2 como la media de sus dos valores centrales"
"v4 tiene un número impar de valores con 1 en medio"
```

(defrule MediaImpar

?f <- (vector (nombre ?n)(valores $?antes ?x $?despues))

```
        (test (=(length $?antes)(length $?despues)))
        =>
        (printout t ?n " es un vector impar con el valor " ?x " en medio" crlf))


(defrule MediaPar
        ?f <- (vector (nombre ?n)(valores $?antes ?x ?y $?despues))
        (test (= (length $?antes)(length $?despues)))
        =>
        (printout t ?n " es un vector par con media " (/(+ ?x ?y)2) crlf))
```

Ejercicio 2

1. Define una plantilla calificaciones con los campos nombre, curso, asignatura, y el multicampo notas. Por ejemplo:
    1. (calificaciones (nombre “Garcia,Carlos”)(curso “20/21”)(asignatura SIN)(notas 4 5 8 9))

```
(deftemplate calificaciones (slot nombre)(slot curso) (slot asignatura)(multislot notas))

(deffacts Notas
        (calificaciones (nombre "Garcia,Carlos")(curso "20/21")(asignatura SIN)(notas 4 5))
        (calificaciones (nombre "Lorenzo,Jaime")(curso "19/20")(asignatura PAS)(notas 4 6))
        (calificaciones (nombre "Jorge")(curso "20/21")(asignatura ED)(notas 5 1))
)
```

2. Define una regla que cree un hecho con la última calificación botenida.

```
(defrule ultima-nota
        ?h<-(calificaciones (nombre ?n)(curso ?c)(asignatura ?a)(notas $? ?ultima))
        =>
        (assert (ultimaCalificacion ?n ?c ?a ?ultima))
)
```

3. Define una regla que cree un hecho que muestre una mejoría en las notas.

```
(defrule MejoraCalificaciones
        (calificaciones (nombre ?n)(curso ?c)(asignatura ?a)(notas $?))
        (forall (calificaciones (nombre ?n)(curso ?c)(asignatura ?a)(notas $? ?x ?y $?)) (test (<= ?x ?y)))
        =>
        (assert (mejoraCalificacion ?n ?c ?a)))
```

4. Define una regla que calcule la media de las notas e indique si el alumno está suspenso o aprobado

```
(defrule aprobado
        (calificaciones (nombre ?n)(curso ?c)(asignatura ?a)(notas $?antes ?x ?y $?despues))
        (exists (test (>= ( / (+ ?x ?y) 2) 5)))
        =>
        (printout t ?n " esta aprobado con una nota media de " (/(+ ?x ?y)2) crlf)
)

(defrule suspenso
        (calificaciones (nombre ?n)(curso ?c)(asignatura ?a)(notas $?antes ?x ?y $?despues))
        (exists (test (< ( / (+ ?x ?y) 2) 5)))
        =>
        (printout t ?n " esta suspenso con una nota media de " (/(+ ?x ?y)2) crlf)
)
```