Tema 1:

- **Aspectos de la inteligencia:**

La memoria, el pensamiento abstracto y el razonamiento, el lenguaje y la comunicación, el aprendizaje, la resolución del problema, la creatividad.

- **Tipos de inteligencia según Howard Gardner:**

Inteligencia Lingüística: la que tienen los **escritores**, los poetas, los buenos redactores.

Inteligencia Lógica-matemática: la que utilizamos para resolver problemas de lógica y matemáticas. Es la inteligencia que tienen los **científicos**. Se corresponde con el modo de pensamiento del hemisferio lógico.

Inteligencia Espacial: consiste en formar un modelo mental del mundo en tres dimensiones, es la inteligencia que tienen los **marineros**, los **ingenieros**, los cirujanos, los escultores, los arquitectos, o los decoradores.

Inteligencia Musical: es naturalmente la de los **cantantes**, compositores, músicos, bailarines.

Inteligencia Corporal-kinestésica: o la capacidad de utilizar el propio cuerpo para realizar actividades o resolver problemas. Es la inteligencia de los deportistas, los artesanos, los cirujanos y los **bailarines**.

Inteligencia Intrapersonal: es la que nos permite entendernos a **nosotros mismos**. No está asociada a ninguna actividad concreta.

Inteligencia Interpersonal: la que nos permite entender **a los demás**, y la solemos encontrar en los buenos vendedores, políticos, profesores o terapeutas.

Inteligencia Emocional: es formada por la inteligencia **intrapersonal y la interpersonal** y juntas determinan nuestra capacidad de dirigir nuestra propia vida de manera satisfactoria.

Inteligencia Naturalista: la que utilizamos cuando observamos y estudiamos la **naturaleza**. Es la que demuestran los biólogos o los herbolarios.

Inteligencia Cibernética: la que desarrollan las personas estudiando y aprovechando la ciencia que se ocupa de los **sistemas de control y telecomunicaciones**.

- **Historia:**

El encuentro conocido como la **conferencia de Dartmouth**, duró dos meses y se llevó a cabo con tal éxito que se considera esta conferencia como la **introductora del término Inteligencia Artificial** y con él una nueva área científica de conocimiento.

La **Etapa de Expansión** se produjo en los **años 80.**

La época donde suceden años **de crítica y madurez** ocurrió en los difíciles **años 70**.

Fue John McCarthy quién definió **LISP** en los laboratorios de IA del MIT en 1958 siendo una gran contribución al campo de la IA.

- **Hofstadter:**

Postulo que la inteligencia es la habilidad de **responder flexiblemente** a diferentes situaciones, saber **aprovechar circunstancias fortuitas**, dar **sentido a mensajes ambiguos** o contradictorios, encontrar **similitudes entre situaciones diferentes** y **generar nuevos conceptos** e ideas innovadoras. En ningún momento comenta sobre la rapidez. Este tipo dice que no hay diferentes tipos de inteligencias, el buen hijo de puta discriminando.

- **Definición de Inteligencia Artificial:**

"El arte de construir máquinas capaces de hacer cosas que requerirían inteligencia si las hicieran los seres humanos". (Minsky, 1986)

El estudio de los cálculos que hacen posible percibir, razonar y actuar". (Winston, 1992)

- **Tipos de inteligencias artificiales:**

La IA fuerte pretende construir máquinas **que sean conscientes (como una persona)** de lo que hacen, que piensen como los humanos, de tal manera **que "La sala china" no pueda demostrar** que no es consciente de lo que hace.

La IA débil pretende construir máquinas que **no sean conscientes** de lo que hacen, sino que actúen como los humanos en ciertas tareas, o sea máquinas que **pasen la prueba de Turing**. Se puede **comportar de manera inteligente**, pero **no por ello** de la misma manera que un ser **humano.**

El efecto Flynn es la **subida continua** de las puntuaciones de **Cociente Intelectual** a lo largo del tiempo, explica que hay aumento en el coeficiente intelectual, pero nos asegura que **no es simétrico**.

Entre las explicaciones que se han dado a este fenómeno podemos encontrar una mejor nutrición y a las influencias médicas.

- **Deep Blue:**

Deep Blue fue una **supercomputadora** desarrollada por IBM en 1997 para **jugar al ajedrez**. Fue la primera que venció a un campeón del mundo vigente, Gary Kaspárov. La definición de inteligencia es **subjetiv**a y **no podemos decir** que el hecho de que la **IA ha ganado** el juego de ajedrez **le hace más inteligente** que tu **oponente**.

- **Alan Turing:**

La prueba de Alan Turing se ha desarrollado **para detectar** si es posible **distinguir entre una máquina y un ser humano**. A estos efectos, un ser humano habla simultáneamente con una máquina y otro ser humano, sin tener contacto visual con ninguno de los dos. Si no es posible distinguir entre hombre y máquina, es de **suponer** que la **máquina** tiene la **misma capacidad intelectual que el ser humano**. Durante la prueba, el ser humano no pregunta cosas especiales anteriormente definidas, por lo que, en este caso, lo que se quiere detectar no es una Inteligencia Artificial débil, sino que una Inteligencia Artificial fuerte.

- **Searle:**

La sala china de Searle es un contraejemplo a la prueba de Turing. La sala china propone que, si Searle es encerrado en una sala con la única misión de traducir unos textos en chino, sin entender el idioma y con la única ayuda de un manual no se puede decir que el manual, que Searle y/o que la sala entienda el lenguaje chino, por tanto, un computador que traduce signos chinos no se puede decir que entienda chino **ni que use la "inteligencia" para ello**. Maneja **información sintáctica,** respuestas en base a una serie de reglas predefinidas sin conciencia alguna de la propia acción.

- **Conclusión:**

Para Alan Turing si una maquina se comporta de manera inteligente se puede afirmar que dicha maquina posee inteligencia **(IA débil),** pero **para Searle,** aunque una maquina posea un comportamiento inteligente no se podrá decir que lo es, en tanto en cuanto, dicha maquina no posea conciencia de lo que está haciendo **(IA fuerte).**

- **Áreas de Aplicación:**

Problemas de percepción: visión y habla, Planificación, estrategias inteligentes, Robótica,
Predicción financiera, Aprendizaje, Minería de datos, Juegos, Mundos virtuales, Internet, Sistemas expertos.

- **Futuro de la IA:**

Orientado a abordar aquellas tareas que, ya sea por lo **incomodo, peligroso o complicado**, conviene apoyarlas o delegarlas en sistemas inteligentes artificiales.

Tema 2:

- **Sistemas de producción:**

El proceso de búsqueda se puede realizar explorando **un árbol (árbol de búsqueda) o en general un grafo** (eliminando repeticiones de estados)

Propuesto por **POST** en 1943.

BH (Base de Hechos). Conjunto de representaciones de uno o más estados por los que atraviesa el problema. Constituye la estructura de datos global

RP (Reglas de Producción). Conjunto de operadores para la transformación de los estados del problema, es decir, de la base de hechos. Cada regla tiene dos partes: **Precondiciones // Postcondiciones**

EC (Estrategia de control). Determina el conjunto de reglas aplicables mediante un proceso de pattern-matching y resuelve conflictos entre varias reglas a aplicar mediante el filtrado.

1. Determinar el conjunto de reglas aplicables y aplicar el filtrado.
2. Aplicar la regla seleccionada. La selección depende de la información de control.
3. Repetir hasta que se den las condiciones de terminación.

- **Caracterización del problema:**

Identificando si se puede **descomponer** el problema ante el que nos encontramos, viendo **si se pueden ignorar** o al menos deshacer pasos erróneos hacia la solución, descubriendo si el **universo es predecible**, comprobando si la **bondad** de una solución es relativa o absoluta y si la solución **es un estado o un camino**. Además, ver el papel que desempeña **el conocimiento** a la hora de encontrar la solución.

- **Ignorables** -> Demostración de teoremas.
- **Recuperables** -> Podemos retroceder.
- **Irrecuperables** -> No se puede retroceder.

- **Problemas clásicos de búsqueda:**

- **Jarra de agua:**

Jarra 1	Jarra 2	Regla a aplicar
0	0	2
0	3	9
3	0	2
3	3	7
4	2	5
0	2	9
2	0	[illegible]

- **8-Puzzles:**

La mejor heurística a la hora de encontrar solución al problema es, **la suma de las distancias** de las piezas a sus posiciones en el objetivo utilizando la **distancia Manhattan**.

- **Otros problemas:**
  - El problema del viajante de comercio
  - El dilema del granjero
  - El dilema de los misioneros
  - Las torres de Hanoi
  - **El problema del puente (Explicación en las transparencias)**

- **Estrategias de búsqueda básica:**

- Ciclo de **control básico** dentro de una estrategia de control:
  - **E1** Exploración de la frontera.
  - **E2** Cálculo de reglas aplicables.
  - **E3** Resolución de conflictos.
  - **E4** Aplicación de regla y memorización de estado.

- **Tipos de Estrategias:**

Las estrategias para considerar las podemos subdividir en:

Irrevocables: Presenta la característica de que **no se permite la vuelta atrás**. Mantenemos una frontera unitaria.

Tentativas: La búsqueda es multi o mono camino. Se mantienen **estados de vuelta atrás** por si el estado actual no llega a buen fin.

En todo momento se debe producir un avance y este debe ser dirigido.
Este avance debe ser metódico.

- **Estrategia Irrevocable:**

- **Disponemos del suficiente conocimiento local.**
- **Tiene problemas de mesetas, máximos locales y crestas.**
- **Algoritmos voraces. (no permiten la vuelta atrás)**
- Las equivocaciones sólo alargan la búsqueda.
- Debemos especificar una función de evaluación f() que nos proporcione un mínimo (máximo) en el estado final.

- **Estrategia Tentativa:**

Ej: Una estrategia heurística para encontrar el camino a la meta en un laberinto, solo con esos datos.

- **No informadas:**

"Son ciegas en el sentido de que el orden en el cual la búsqueda progresa no depende de la naturaleza de la solución que buscamos"
La estrategia de búsqueda ciega es más eficiente en pequeños problemas.

  - **Búsqueda en profundidad:** variación del **backtrackin**.
  - **Búsqueda en anchura:** mayor prioridad a los nodos de menor profundidad.
  - **Coste uniforme:** serán similar al de anchura cuando el coste de aplicación de cada regla sea unitario.

¿Solución óptima **en grafos finitos**? En la búsqueda en **profundidad.**
Puede la búsqueda de coste uniforme **se vuelva infinita**, **cuando el coste del nodo expandido sea cero y conduzca de nuevo al mismo estado**.

- **Informadas:**

Al contrario de las "ciegas" las informadas sí que van a disponer de información de lo **prometedor que es un nodo** para llegar desde él a la solución.

Algoritmo de búsqueda A*.

Los conceptos básicos de la búsqueda heurística son: **Completitud, admisibilidad, dominación y optimalidad.**

Un algoritmo A1 es dominante sobre A2 si cada nodo expandido por A1 es también expandido por A2.

- **Búsqueda A*:**

A*: f*(n) = g*(n) + h*(n)

- **g*(n) = c(s, n)**
  - Coste del camino de **coste mínimo** desde el nodo inicial s al nodo n.
  - Estimada por **g(n).**

- **h*(n)**
  - Coste del camino de **coste mínimo** de todos los caminos desde el nodo n a cualquier estado solución tj.
  - Estimada por **h(n).**

- **f*(n)**
  - Coste del camino de coste mínimo desde el nodo inicial hasta un nodo solución condicionado a pasar por n.
  - Estimada por **f(n).**

- **C***
  - Coste del camino de coste mínimo desde el nodo inicial a un nodo solución.

F*(n)=C* en cada nodo del camino óptimo.

Una **función heurística** h(n) se dice que es a**dmisible** (garantiza la obtención de un **camino de coste mínimo hasta un objetivo**) cuando se cumple:

$$h(n) \leq h^*(n) \quad \forall n$$

Cuanto más correctamente estimemos h(n) menos nodos de búsqueda generaremos.

Problema: si nuestra función heurística nos devuelve un valor superior a h*, para algún nodo, **no se puede garantizar que encontremos la solución óptima**.

A la hora de hallar la heurística para un problema, debemos tener en cuenta:
- Que tenga unas buenas restricciones, puesto que se podría quedar fuera la solución óptima.
- Acercar nuestra heurística lo máximo posible a **la heurística óptima** de ese problema para no perder la solución óptima.

- **Problemas de camino mínimo:**

Si una función heurística no varía (es constante o nula), no modifica la diferencia entre los valores de f(n) de cualquier nodo de búsqueda, por tanto, se buscará en los mismos que si no usamos una heurística.
Para implementar el algoritmo A* y obtener el camino óptimo utilizaremos la distancia de **Manhattan, ya que no podemos movernos en diagonal**. (Sólo puede avanzar en 4 direcciones)

La **distancia de Manhattan (h(n))** se utiliza para calcular el camino más corto sin movimientos diagonales, mientras que **para el 8-con tenemos diagonales**, por lo que podemos obtener un camino más corto en diagonal que si nos desplazamos sin diagonales.

Para un problema de camino más corto en el que nos podemos mover en las **8 direcciones** se **utilizará el cálculo de la diagonal.**

Heurística óptima: h1((x, y)) = |m-x| + |n-y|

Coste actual óptimo: h2((x,y)) = |x| + |y|

Heurística admisible: h3((x,y)) = sqrt((m-x)^2 + (n-y)^2)

- **Inconvenientes de mantener la admisibilidad:**

El mantenimiento de la admisibilidad fuerza al algoritmo a **consumir mucho tiempo** en discriminar caminos cuyos costes no varían muy significativamente.

No es práctico para problemas de mayor envergadura.

En general el nivel de información de las heurísticas permite encontrar antes la solución, pero tiene la desventaja de requerir un mayor coste computacional para su cálculo.

- **Relajación de la restricción de optimalidad:**

Al relajar podemos alcanzar una **solución en menor tiempo** y en algunos casos **obtener la solución óptima.**

- Técnica de ajuste de pesos:

El objetivo de la técnica de ajustes de pesos es definir **una función f() ponderada, fw(), como alternativa a la utilizada en A*.**

- Técnica de la admisibilidad- Ɛ:

Objetivo: **Aumentar la velocidad de búsqueda a costa de obtener una solución subóptima.**

A*Ɛ opera de forma idéntica al algoritmo A* **salvo que selecciona aquel nodo de Lista Focal con menor valor de Hf(n).**

- Algoritmo de estimación de coste de búsqueda A:

Lista focal es una **sublista de ListaFrontera** que contiene solo nodos con f(n) menor al mejor valor de los f(n) de listaFrontera por un factor.

Lista Frontera en el algoritmo A*sirve para añadir y coger de ella los nodos a los que podemos acceder desde el nodo actual.

- Comparación de algoritmos:

El algoritmo de ponderación dinámica es más sencillo, pero únicamente es aplicable a problemas donde se conoce la **profundidad.**

Utilizando la técnica de relajación de la restricción de optimalidad "Algoritmo de ponderación dinámica" **en los últimos niveles no utiliza ninguna búsqueda ni en anchura ni en profundidad.**

Tema 3:

- **Juegos como problemas de búsqueda:**

- **Imposible generar** todo el **árbol de búsqueda**.
- **Estado (N): configuración del juego** en un **momento** dado.
- **Árbol de juego. Cada arista** de ese árbol indica un **posible movimiento**.
- En cada nivel se **van alternando los jugadores**.
- **Factor de ramificación (B):** número de **posibles movimientos** que se pueden **realizar**.
- **Wolfgang Kempelen** crea el "ajedrecista mecánico" en el año **1760**.
- El **inicio** de la **jugada** queda **definido** por el **análisis** del árbol.

- **Estrategia exhaustiva: MiniMax:**

Genera todos los nodos del árbol hasta **la profundidad deseada.**

Evalúa cada nodo hoja.

Asigna un **valor al nodo raíz**, siempre **debe tener valor**.

- Si la decisión la toma el jugador **MIN**, asociar a ese **nodo el mínimo** de los valores de sus hijos, y **el máximo** en caso de **MAX**.

El **método MiniMax** funciona teniendo en cuenta el **mejor movimiento** para ti suponiendo que el **contrincante** realiza **el peor** para ti.

Se nos podría **presentar un problema** aplicando **la estrategia MiniMax**, cuando necesitamos una respuesta en muy poco tiempo.

- **Juegos multijugador:**

Minimax extendido a **juegos** de **más** de **dos jugadores**.

Se sustituye el **valor** de un **nodo** por un **vector de valores** (tantos valores como jugadores).

- **Estrategia de poda: α-β:**

La poda alfa beta es una técnica de búsqueda que **reduce el número de nodos evaluados en un árbol de juego** por el Minimax.

α es el **valor** de la **mejor opción hasta el momento a lo largo del camino para MAX**, esto implicará por lo tanto la elección del **valor más alto**, **β** es el **valor** de la **mejor opción hasta el momento a lo** largo **del camino** para **MIN**, esto implicará por lo tanto la elección del **valor más bajo**.

El **valor MiniMax** de un nodo estará siempre acotado **α <= V(N) <= β.**

Al principio inicializamos α = -∞ y β = ∞ al no tener ninguna evidencia.

Esta búsqueda **alfa-beta va actualizando** el valor de los parámetros **según se recorre el árbol.** El método **realizará** la **poda de las ramas restantes** cuando el **valor actual** que se está **examinando** sea **peor** que el **valor actual** de **α** o **β** para MAX o MIN, respectivamente.

α = max[α, V (Nk, α, β)]	**β = min[β, V (Nk,α, β)]**

- **<u>Uso de movimientos de libro:</u>**

Imposible seleccionar un movimiento consultando la configuración actual del juego en un catálogo y extrayendo el movimiento correcto.

Razonable para algunas partes de ciertos juegos.

- En **ajedrez**, tanto la secuencia de apertura como los finales están muy estudiados.

El rendimiento del programa puede mejorarse si se le proporciona una **lista de movimientos** (**movimientos de libro**).

Se usa el libro en las aperturas y los finales combinado con el procedimiento MiniMax para la parte central de la partida. (Ciertas partes muy estudiadas)

- **<u>Espera del reposo:</u>**

Consiste en **explorar nodos hasta que su valor no cambie de manera drástica después de explorar un nivel más**, es decir, el valor del nodo se **estabilice** de un nivel al siguiente.

Busca evitar el efecto horizonte.

- **<u>Técnica de bajada progresiva:</u>**

Restricciones de tiempo: algoritmos presentados anteriormente no adecuados.

Recorrer **nodos por nivele**s.

Al **llegar** la **petición de jugada**, **devolver** la **solución** del **último nivel** que se haya **completado**.

- **<u>Poda heurística:</u>**

Objetivo: **reducir B desarrollando únicamente los mejores movimientos de cada nivel**.

g(N): Función adicional de evaluación.

- De **bajo coste**.

- **Versión simplificada de f(N).**

Factor de ramificación: **Factor (Nodo) = Factor (Padre (Nodo)) – Rango (Nodo)**

- **Continuación heurística:**

Intento de evitar el efecto horizonte:
Provocado por la **limitación en profundidad**: solo se **puede tener conocimiento** hasta la **profundidad seleccionada.**

Selecciona un **subconjunto** de **nodos terminales** para desarrollar **búsquedas más profundas**.

Tema 4:

- **Problemas de satisfacción de restricciones (CSP):**

Conjunto de **variables** definidas sobre **dominios** finitos y conjunto de **restricciones** definidas sobre subconjuntos de dichas variables.

Solución al problema: **la relación n-aria que satisface todas las restricciones del problema**.

Dependiendo de los **requerimientos** del problema **hay que encontrar todas las soluciones o sólo una**.

Visualizarlo como un grafo de restricciones puede **usarse para simplificar el proceso** de solución.

- **Redes de restricciones:**

Un **CSP** se puede **representar** como un **grafo**.

- **CSP binario:**

Aquel en el que todas las **restricciones** tienen a los **sumo dos variables** respectivamente.

Todo problema n-ario se puede formular como un problema binario.

Ejemplos de CSP binarios:
- Asignación de tareas para un robot.
- Coloreado de mapas.
- Generación de crucigramas.
- N-reinas.

- **Coloreado de mapas:**

Todo esto es incorrecto:

Se tiene que resolver siempre utilizando el algoritmo AC3, ya que backtracking no nos garantiza que vaya a encontrar una solución.

Se necesita el mismo número de colores que número de territorios fronterizos con el área que más territorios fronterizos tenga.

- **Generación de crucigramas.**

Características**: CSP Binario, discreto (dominios grandes).**

- **N-reinas.**

Dominios: **columnas posibles** {1, 2, ..., **n**}.

Características: **Dominios discretos y restricciones binarias.**

- **Criptoaritmética:**

Características: **Dominios discretos y restricciones múltiples**.

- **Restricciones temporales:**

Características: Dominios continuos y restricciones binarias.

- **Árbol de interpretaciones:**

Partimos de un **nodo raíz** que **supervisa** el **proceso**.

La **solución** se **construye** de forma **incremental** de tal forma que **cada hoja** es una **interpretación**.

Cada nivel corresponde a una **asignación** de **valor para** una **característica** de **datos**. El **orden** de **descenso** viene especificado por **a**.

Cada nodo identifica una **posibilidad** de **asignación**.

- **Métodos de resolución:**

- Inferencia:

Consistencia de caminos.

- Búsqueda:

Generación y test: generar de forma sistemática y exhaustiva cada una de las posibles asignaciones a las variables y comprobar si satisfacen todas las restricciones. Hay que explorar el espacio definido por el producto cartesiano de los dominios de las variables.

Backtracking: se trata de construir la solución de forma gradual, instanciando variables en el orden definido por la permutación dada

Backjumping: parecido al BT pero el retroceso no se hace a la variable instanciada anteriormente sino a la variable más profunda que está en conflicto con la variable actual.

Eficiencia en **problemas** de **gran tamaño**:

Backjumping > Backtracking > Generación y test

Diferencias:

Cuando encontramos un espacio de dominios vacío, **Backtracking** sólo puede volver al **nodo anterior**, es decir, subir un nivel mientras que **Backjumping** puede **saltar** al **nodo** en **conflicto**.

- Algoritmos híbridos:

Forward Checking.

Maintaining Arc Consistency.

Heurísticas.

- **<u>Generación y test:</u>**

Se basa en **expandir una a una todas** las **posibilidades** del **problema**.

Busca la **solución** mediante una **expansión** del **árbol** en **anchura**.

Es muy **poco eficiente**.

- **<u>Backtracking:</u>**

Si **no se puede extender**: backtracking

- **cronológico: se elimina la última decisión.**
- **no cronológico: se elimina una decisión anterior.**

Backtraking **siempre** da la **solución óptima** siempre que el **algoritmo** con el **que trabajemos devuelva** todas las **posibles soluciones**.

Estrategia:
Se trata de construir la **solución** de forma **gradual**, instanciando variables en el orden definido por la permutación dada

- Trashing e inconsistencia de nodo:
Relacionado con las **restricciones unarias**. Sucede cuando un dominio contiene un valor que no satisface una restricción unaria.

- Inconsistencia de arista:
Relacionado con las **restricciones binarias**. Sucede cuando existe una restricción binaria entre dos variables de tal forma que para un determinado valor de la primera variable no existe ninguna asignación posible para la segunda.

- Dependencia de la ordenación:
El orden de selección de las variables es un factor crítico. Se han desarrollado diversas heurísticas de selección de variable y de valor.

- **Forward checking:**

Los **valores** de las **variables futuras que son inconsistentes con** la **asignación actual** son **temporalmente eliminados** de sus **dominios**.

Si el **dominio** de una **variable futura** se queda **vacío**, la instanciación de **la variable actual** se **deshace y** se **prueba** con un **nuevo valor**.

Forward Checking **no soluciona** la **dependencia de la ordenación** del Backtraking.

- **Propagación de Restricciones:**

Transformar el **problema** en **otro** más **sencillo** sin **inconsistencias** de arco.

Propiedad de consistencia de arista:

Una arista dirigida $c(e_p) = <V_i, V_j>$ es consistente si y sólo si para todo valor asignable a V_i existe al menos un valor en V_j que **satisface la restricción asociada a la arista.**

Obtendremos: **Una, ninguna o varias soluciones**.

Un **CSP** puede **transformarse** en una **red consistente mediante** un algoritmo sencillo (**AC3**) que **examina** las **aristas**, **eliminando** los **valores** que **causan inconsistencia del dominio** de cada variable.

- **Algoritmo AC3:**

Cuando un **dominio** queda **vacío** se queda **inconsistente y sin solución**.

Si el grafo es **consistente puede** tener **una solución** o **más**.

Tras la **utilización** del **algoritmo AC3** hemos **logrado eliminar** todos los **valores** que **causan inconsistencia** del **dominio** de **cada variable, usaremos backtracking** únicamente **cuando** la **eliminación** de **inconsistencias de lugar** a **más** de una **solución**.

El algoritmo AC3 de búsqueda por CSP, **la variable Q**: contiene todas las **restricciones binarias del problema** en **ambos sentidos**.