octubre 2025

BACKTRACKING

Podas

Propagación

Heurísticas

Joaquín - Steven - Elías - Alexander - Paula

HOY

Resumen

Kakuro Square

Acertijo

12:20 **13:30**

MOTIVACIÓN

Backtracking es como la vida: avanzas, **la cagas**,

retrocedes y vuelves a empezar.

Eso es aprender

BACKTRACKING





Backtracking es **igual o más rápido** que la fuerza bruta

BACKTRACKING

Problemas de decisión

Búsqueda de una solución factible

Problemas de optimización

Búsqueda de la mejor solución a un problema. Ejemplo:

Problemas de enumeración

Búsqueda de todas las soluciones posibles. Ejemplo:

BACKTRACKING

- **X**: Variables
- **D**: Dominios
  - Dx: dominio de **X**
- **R**: Restricción
  - c/u para un subconjunto de **X**

isSolvable(X,D,R):

```
if X es vacío, return True
x ← una variable de X
for v E D_x:
    if x = v viola R, continue
    x ← v
    if isSolvable(X - {x}, D, R):
        return True
    x ← vacío
return False
```

BACKTRACKING CON MEJORAS

Poda

Permite reducir el espacio de búsqueda, **se podan ramas** del decision tree que no conducen a una solución válida. Se consideran criterios como factibilidad, dominio y consistencia

Propagación

Cuando a una variable se le asigna un valor, se puede **propagar esta información** para luego poder reducir el dominio de valores de otras variables.

Heurísticas

Es una aproximación al **mejor criterio** para abordar un problema

PODAS

Permite reducir el espacio de búsqueda, **se podan ramas** del decision tree que no conducen a una solución válida. Se consideran criterios como factibilidad, dominio y consistencia. En particular lo que hacemos acá es incluir nuevas restricciones dada una asignación de x

```
isSolvable(X,D,R):
    if X es vacío, return True
    x ← una variable de X
    for v E D_x:
        if x = v viola R, continue
        x ← v
        if isSolvable(X - {x}, D, R):
            return True
        x ← vacío
    return False
```

PROPAGACIÓN

Cuando a una variable se le asigna un valor, se puede propagar esta información para luego poder reducir el dominio de valores de otras variables.

```
isSolvable(X,D,R):
    if X es vacío, return True
    x ← una variable de X
    for v E D_x:
        if x = v viola R, continue
        x ← v Propagar
        if isSolvable(X - {x}, D, R):
            return True
        x ← vacío Propagar
    return False
```

HEURÍSTICAS

Es una aproximación al **mejor criterio** para abordar un problema

```
isSolvable(X,D,R):
    if X es vacío, return True
    x ← una variable de X La mejor
    for v E D_x de mejor a peor:
        if x = v viola R, continue
        x ← v
        if isSolvable(X - {x}, D, R):
            return True
        x ← vacío
    return False
```

BACKTRACKING

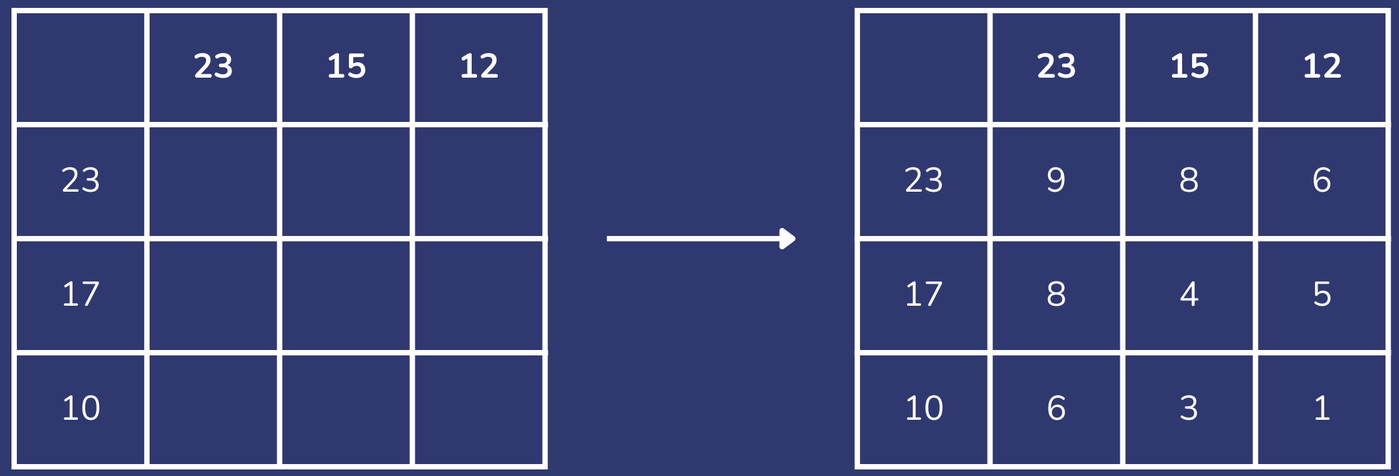
→ EJERCICIO

Un tablero de *kakuro square* es una grilla de NxM donde cada posición tiene una celda que puede ser rellenada por un número natural distinto de 0. Además, cada fila y columna tiene un número natural que indica el valor que debe tener la suma de los números correspondientes a la fila o columna. Por ejemplo:

	23	**15**	**12**
23			
17			
10			



	23	**15**	**12**
23	9	8	6
17	8	4	5
10	6	3	1

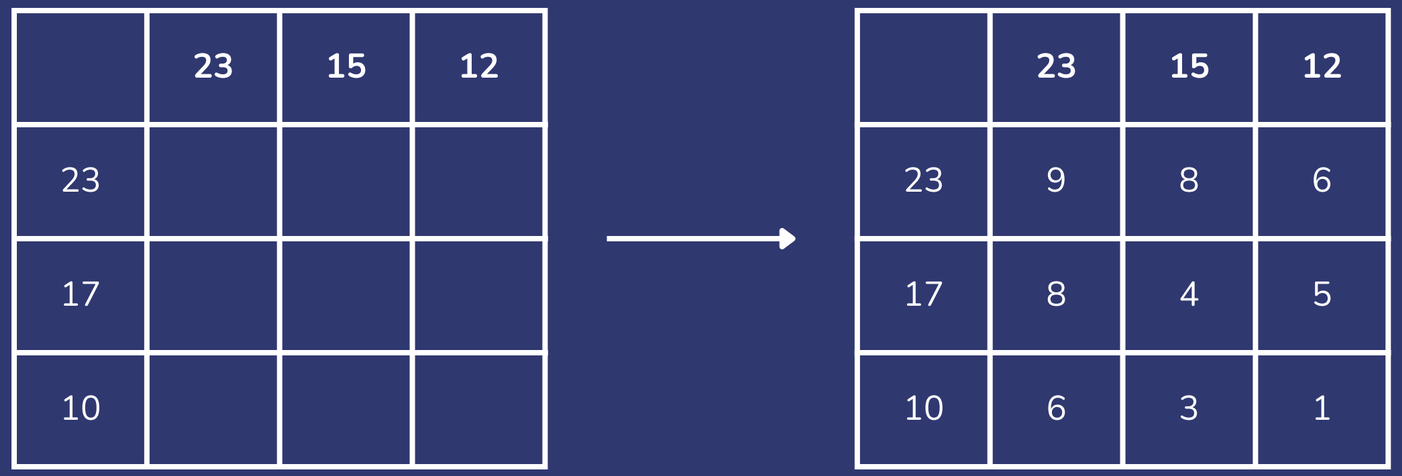


El problema de rellenar la grilla cumpliendo con las restricciones se puede resolver utilizando la estrategia de *backtracking*.

a) Identifica las variables del problema y sus respectivos dominios.

b) Explica con tus palabras cómo se podría revisar en tiempo O(1) si la asignación de una variable rompe una restricción.

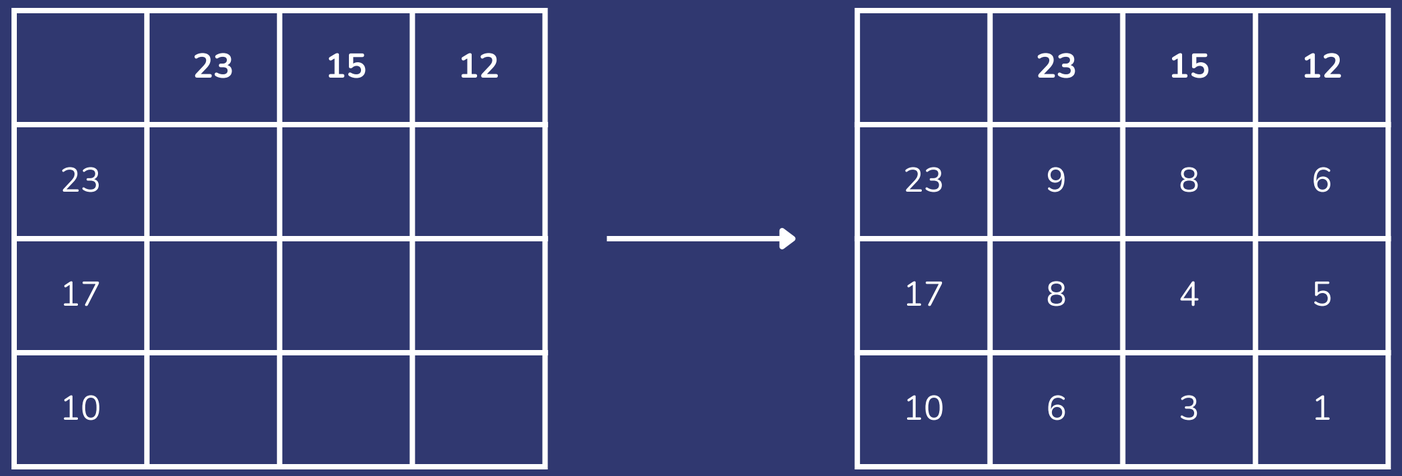
c) Explica con tus palabras una poda o una heurística que se pueda aplicar para resolver este problema más eficientemente.



	23	15	12
23			
17			
10			

	23	15	12
23	9	8	6
17	8	4	5
10	6	3	1

a) Identifica las variables del problema y sus respectivos dominios.



a) Identifica las variables del problema y sus respectivos dominios.

Para resolver con backtracking este problema se debe asignar a cada celda un número, por lo que **las variables son las celdas**.

En teoría cada celda puede tener cualquier número natural, pero ya que no puede haber un dominio infinito para resolverlo con backtracking es necesario acotar los dominios. **Un dominio acotado correcto puede ser los números del 1 al MAX(restriction)**. Ya que se sabe que nunca un número puede ser mayor al valor de su restricción.

	23	15	12
23			
17			
10			

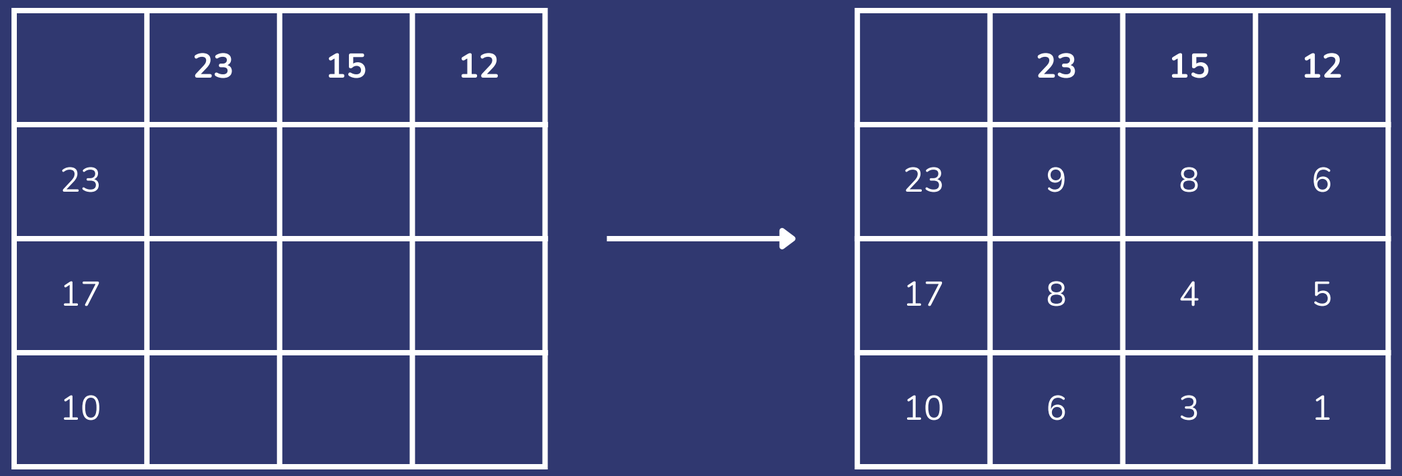


	23	15	12
23	9	8	6
17	8	4	5
10	6	3	1

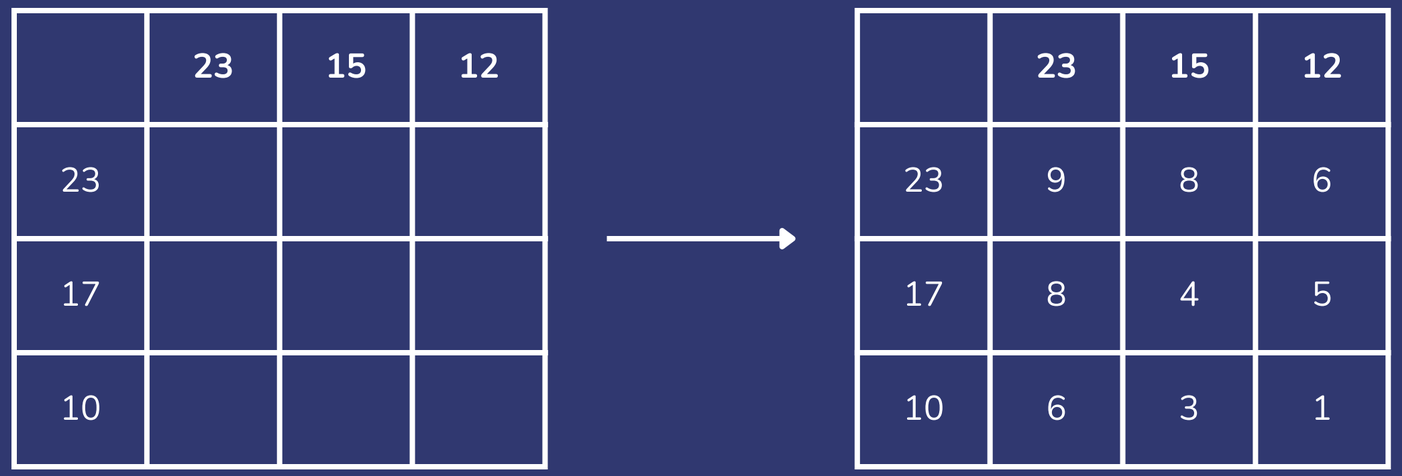
a) Identifica las variables del problema y sus respectivos dominios.

las variables son las celdas.

Un dominio acotado correcto puede ser los números del 1 al MAX(restriction).



b) Explica con tus palabras cómo se podría revisar en tiempo O(1) si la asignación de una variable rompe una restricción.



b) Explica con tus palabras cómo se podría revisar en tiempo O(1) si la asignación de una variable rompe una restricción.

Se puede mantener un **contador** por cada restricción del problema que cuente la **suma actual de cada fila o columna**. De esta manera se inmediatamente se rompe la restricción cuando el contador supera la restricción.

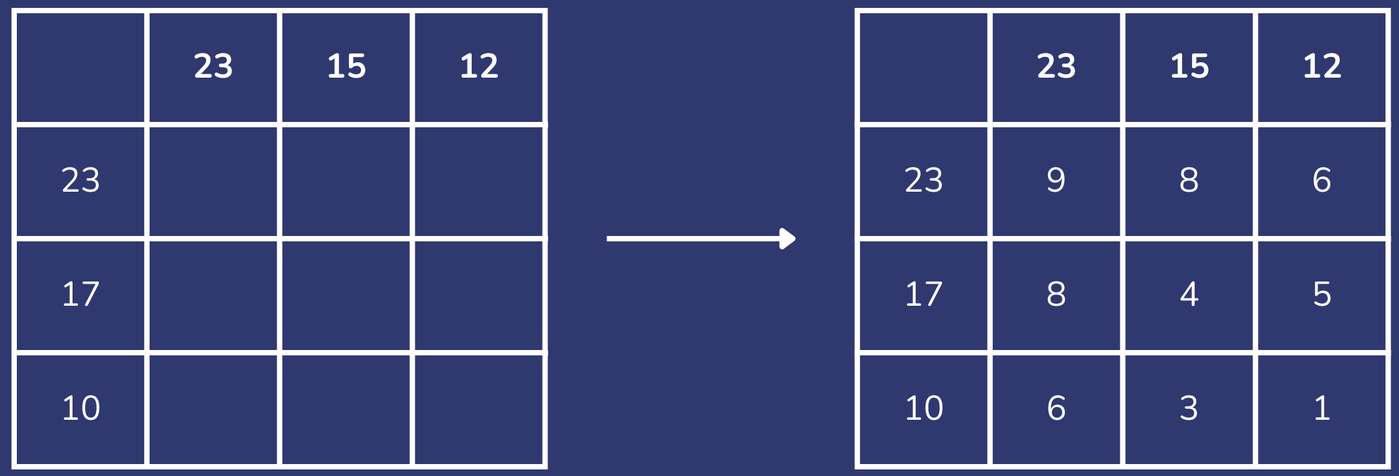
	23	**15**	**12**
23			
17			
10			



	23	**15**	**12**
23	9	8	6
17	8	4	5
10	6	3	1

b) Explica con tus palabras cómo se podría revisar en tiempo O(1) si la asignación de una variable rompe una restricción.

contador **suma** **actual de cada fila o columna**.



c) Explica con tus palabras una poda o una heurística que se pueda aplicar para resolver este problema más eficientemente.

	23	**15**	**12**
23			
17			
10			



	23	**15**	**12**
23	9	8	6
17	8	4	5
10	6	3	1

c) Explica con tus palabras una poda o una heurística que se pueda aplicar para resolver este problema más eficientemente.

Poda por valores disponibles: Si en una celda no quedan valores válidos disponibles (todos violan las restricciones de suma o exceden los rangos permitidos), podamos inmediatamente esa rama sin seguir explorando. Esto permite detectar tempranamente estados sin solución y evitar explorar subárboles completos del espacio de búsqueda que están destinados a fallar.

	23	**15**	**12**
23			
17			
10			



	23	**15**	**12**
23	9	8	6
17	8	4	5
10	6	3	1

c) Explica con tus palabras una poda o una heurística que se pueda aplicar para resolver este problema más eficientemente.

Heurística de Forward Checking: Después de asignar un valor a una celda, propagamos inmediatamente las restricciones actualizando los dominios de las celdas restantes en la misma fila o columna. Eliminamos del dominio de estas celdas todos aquellos valores que harían que la suma parcial sobrepase el valor objetivo de la restricción. Esto permite detectar inconsistencias anticipadamente antes de realizar más asignaciones, reduciendo significativamente el espacio de búsqueda al identificar callejones sin salida más temprano en el proceso.

Considere el siguiente acertijo lógico: En cinco casas alineadas de izquierda a derecha, cada una de un color diferente, viven 5 personas de diferentes nacionalidades, cada una de las cuales prefiere un deporte diferente, una bebida diferente y una mascota diferente. Además, están dados los siguientes hechos:

- La persona inglesa vive en la casa roja.
- La persona española tiene el perro.
- La persona argentina vive en la primera casa de la izquierda.
- Se prefiere el fútbol en la casa amarilla.
- La persona que prefiere el basketball vive en la casa junto a la persona que tiene el gato.
- La persona argentina vive junto a la casa azul.
- La persona que prefiere el volleyball tiene el conejo.
- La persona que prefiere el tenis bebe jugo de naranja.
- La persona chilena bebe té.
- La persona japonesa prefiere el judo.
- Se prefiere el fútbol en la casa junto a la casa donde se tiene el caballo.
- Se bebe café en la casa verde.
- La casa verde está inmediatamente a la derecha (a su derecha) de la casa blanca.
- Se bebe leche en la casa del medio.

¿Dónde vive la cebra y en qué casa se bebe agua?

¿Dónde vive la cebra y en qué casa se bebe agua?

En este contexto se le pide:

a) Modele este problema como un CSP (Problema de Satisfacción de Restricciones), para esto defina cuáles son las variables, dominios y restricciones del problema.



¿Dónde vive la cebra y en qué casa se bebe agua?

En este contexto se le pide:

a) Modele este problema como un CSP (Problema de Satisfacción de Restricciones), para esto defina cuáles son las variables, dominios y restricciones del problema.

Variables:

- Color[i], 1 ≤ i ≤ 5: color de la casa i.
- Nacionalidad[i], 1 ≤ i ≤ 5: nacionalidad de la persona que vive en la casa i.
- Deporte[i], 1 ≤ i ≤ 5: deporte preferido por la persona que vive en la casa i.
- Bebida[i], 1 ≤ i ≤ 5: bebida preferida por la persona que vive en la casa i.
- Mascota[i], 1 ≤ i ≤ 5: mascota preferida por la persona que vive en la casa i.

Dominios:

- Color = {rojo, amarillo, azul, verde, blanco}.
- Nacionalidad = {chilena, peruana, inglesa, española, argentina}.
- Deporte = {fútbol, tenis, vóleibol, básquetbol, judo}.
- Bebida = {té, jugo de naranja, café, leche, agua}.
- Mascota = {perro, gato, conejo, caballo, cebra}.

Restricciones:

Las restricciones están dadas por los hechos definidos para el problema. Por ejemplo:

“La persona inglesa vive en la casa roja”.

¿Dónde vive la cebra y en qué casa se bebe agua?

En este contexto se le pide:

b) **Proponga el pseudocódigo de la función solveCSP() para resolver el acertijo**, indicando los parámetros adecuados para realizar la llamada inicial y obtener la solución al acertijo. Asuma que dispone de una función verificaCSP() adecuada para verificar las restricciones del problema.

b) pseudocódigo de la función solveCSP()

input: mascota buscada, bebida buscada

output: id de la casa donde está la mascota, id de la casa donde está la bebida

solveCSP(mascota, bebida):

```
0   for i = 0, 24 Casa[i] ← null
1   if Acertijo(Casa, 0):          // resuelve el acertijo, si tiene solución es true
2       for i = 0, 24              // recorre todas las variables
3           if Casa[i] = mascota
4               m ← (i + 1) / 5    // el ID de la casa correspondiente
5           if Casa[i] = bebida
6               b ← (i + 1) / 4    // el ID de la casa correspondiente
7       return b, m                // retorna los ID de la casa de la bebida y la mascota
8   return -1, -1                  // no había solución
```

b) pseudocódigo de la función Acertijo(Casa, i)

input: Arreglo Casa[0...24], // todas las variables de forma consecutiva en el arreglo Casa índice: $0 \leq i \leq 25$

output: true si hay solución, en Casa están los valores de cada variable

Acertijo(Casa, i):

```
1  if i = 25: return true
2  for v = Dominio(Casa[i])
3      if verificaCSP(Casa, i, v):  // asume que tiene acceso a los hechos (restricciones)
4          Casa[i] ← v
5          if Acertijo(Casa, i + 1):
6              return true
7          Casa[i] ← null
8  return false
```

¿Dónde vive la cebra y en qué casa se bebe agua?

En este contexto se le pide:

c) Explique cómo aplicaría en su solución podas y propagación, de modo de beneficiar el desempeño de su solución al utilizar los hechos conocidos del acertijo.

Las podas se pueden aplicar en la línea 3 de Acertijo(), por ejemplo, dado que sabemos que “La persona argentina vive en la primera casa de la izquierda” podemos agregar una restricción que rechace cualquier valor de dominio distinto para la nacionalidad de la persona de la Casa[1]. De igual forma, es posible aplicar propagación: al asignar un valor de una variable en la línea 4 de Acertijo() retiramos ese valor de los dominios de las variables asociadas. Por ejemplo, al asignar tenis como deporte, retiramos tenis de los dominios de las demás variables por asignar.

¿Dónde vive la cebra y en qué casa se bebe agua?

En este contexto se le pide:

d) Adicionalmente a los hechos indicados en el acertijo, se sabe que habitualmente: Los argentinos prefieren el fútbol y el café, los japoneses el judo y los gatos, los chilenos el fútbol y el té, los españoles los perros y el café, los ingleses el té y el tenis.**¿De qué forma utilizaría esta información para mejorar el desempeño de su solución?**

¿Dónde vive la cebra y en qué casa se bebe agua?

En este contexto se le pide:

d) Adicionalmente a los hechos indicados en el acertijo, se sabe que habitualmente: Los argentinos prefieren el fútbol y el café, los japoneses el judo y los gatos, los chilenos el fútbol y el té, los españoles los perros y el café, los ingleses el té y el tenis.**¿De qué forma utilizaría esta información para mejorar el desempeño de su solución?**

Este caso corresponde a la aplicación de heurísticas en Backtracking. En este caso, como las variables se eligen secuencialmente, lo que podemos hacer en la línea 2 de Acertijo() es escoger el valor del dominio de v a evaluar según las asignaciones ya definidas en Casa.

octubre 2025



BACKTRACKING

Podas

Propagación

Heurísticas

Joaquín - Steven - Elías - Alexander - Paula