Tema 2. Resolución de problemas con búsqueda

- Representación de problemas y espacio de estados
- Búsqueda de soluciones
  - Búsqueda ciega
  - Búsqueda heurística
  - Búsqueda local
  - *Planificación*
- Aplicaciones

Simon's ant & system complexity

Where is the complexity? Just because a system's environment is complex does not mean that the systems operating within it must be complex as well.

Material desarrollado por distintos autores del departamento de ISIA

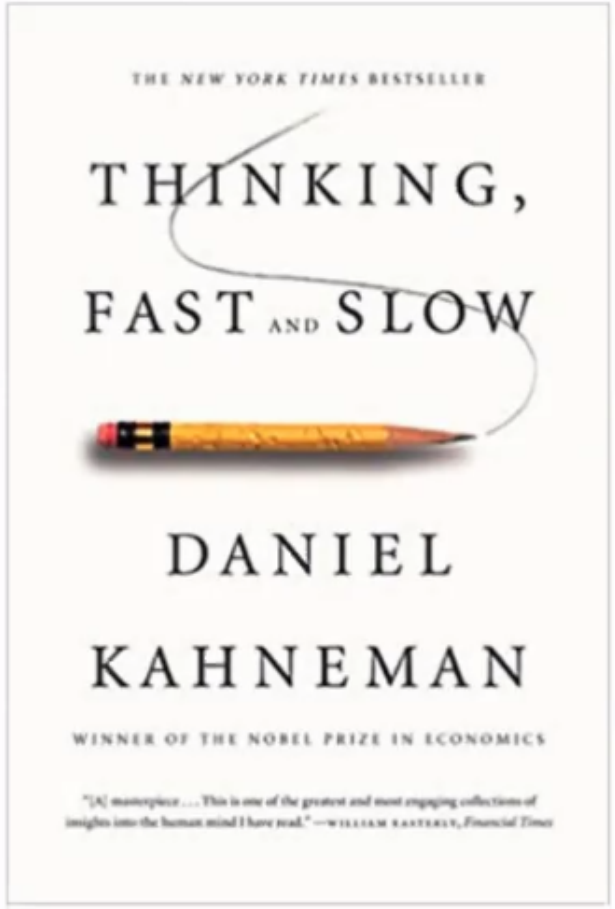
Resolución de problemas con búsqueda

- Las estrategias de control que guían la búsqueda constituyen una parte importante de la IA clásica: algoritmos de planificación, sistemas de reglas,..
- Ir de un punto a otro, resolver un crucigrama, solitario cartas, jugar al ajedrez, diagnosticar una enfermedad, resolver una derivada o una integral, ..
- **Problema de planificación**
  - Estado inicial del mundo
  - Pasos (acciones) que nos llevan a un estado solución
  - Ejecución secuencial de las acciones
- Se dispone de varias acciones legales → no tiene sentido elegir la primera aplicable sino realizar un proceso de búsqueda para "pensar" cual es la mejor y luego aplicarla.
- La planificación es el proceso explícito de deliberación que elige y organiza las acciones anticipando su salida (su comportamiento)
  - El objetivo es conseguir ciertos objetivos preestablecidos
  - *AI planning*: estudio computacional de este proceso de deliberación
- ¿Los humanos planificamos cuando resolvemos problemas?
  - Simulación mental de acciones/consecuencias/ ..
  - ¿Los agentes inteligentes deben planificar?

Planificación/actuación en la resolución de problemas (humano)

- Muchas veces actuamos sin planificación explícita
  - Propósito inmediato
  - Comportamientos muy automatizados
  - Cuando puedo cambiar el orden sin problema
- Actuar después de planificar
  - Cuando la situación es nueva
  - Tareas complejas
  - Mucho riesgo/coste
  - Cuando colaboramos con otros

System 1 and System 2



Daniel Kahneman's identified two modes of thinking in humans:

- **System 1**: operates automatically and quickly, with little or no effort and no sense of voluntary control:
  - Drive a car on an empty road, recognize a friend's face, complete "2 + 2 = ...", etc.

- **System 2**: allocates attention to the effortful mental activities that demand it, including complex (compositional) computations:
  - Check the validity of a complex logical argument, Count the occurrences of the letter a in a page of text, Search memory to identify a surprising sound, perform a complex mental computation, etc.

Google Research

Planificación

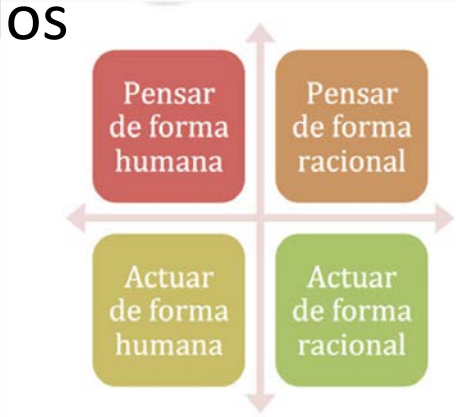
- Se llama planificación al proceso de búsqueda y articulación de una secuencia de acciones que permitan alcanzar un objetivo.
- Aplicaciones
  - Robótica, robot móviles y vehículos autónomos
  - Simulación
  - Logística
  - Cadenas de montaje
  - Gestión de crisis (evacuaciones, incendios,..)

A menudo nos bastan planes aceptables, aunque no sean óptimos (**satisfacción vs. optimalidad**).

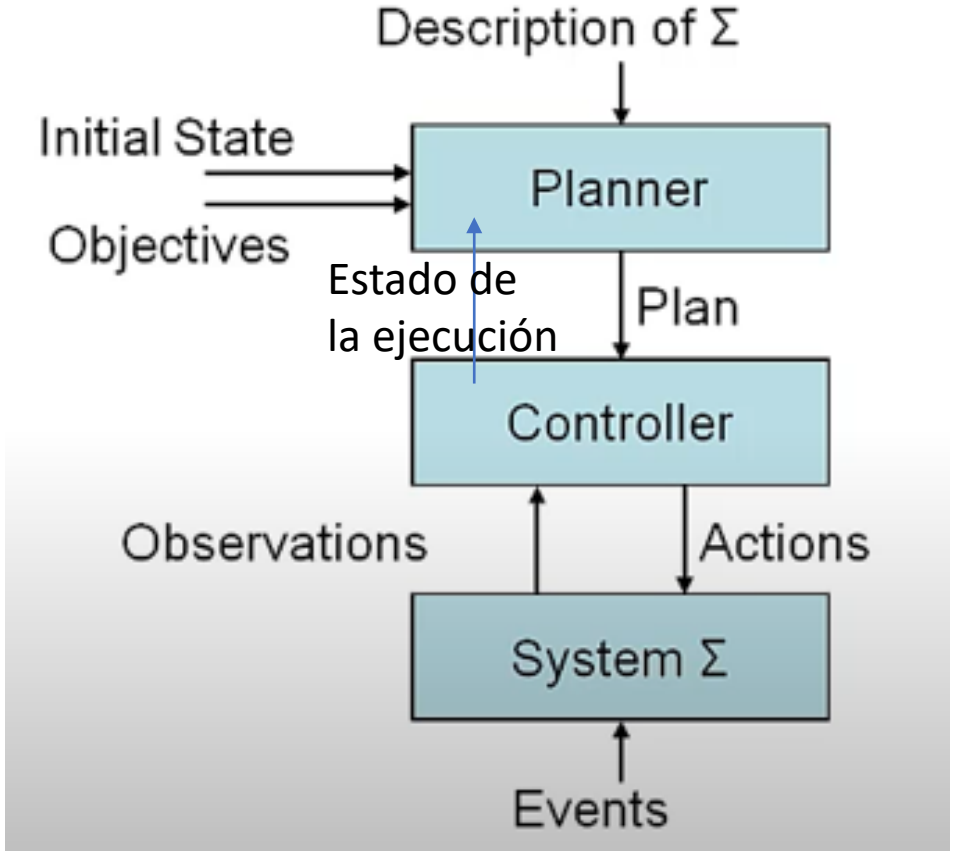
- **Planificación clásica**: Tiene en cuenta entornos completamente observables, deterministas, finitos, estáticos y discretos.
- Se amplia el estudio a problemas de planificación complejos
  - Aspectos dinámicos, temporalidad, dependencias entre objetivos, complejidad, ..

Planificación (pensar vs actuar)

- Planificador
  - IN: modelo conceptual del mundo (representación del problema), estado inicial, estado objetivo
  - OUT: Genera el plan
- Controlador
  - IN: plan, estado real (observado)
  - OUT: acción a ejecutar
- Sistema de transición de estados
  - Evolución de la ejecución de acciones y eventos

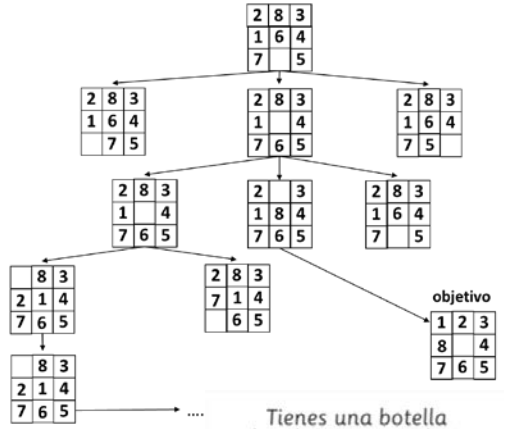


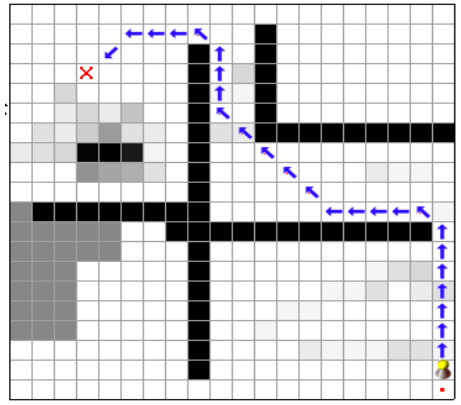
Cuatro enfoques distintos de la Inteligencia Artificial

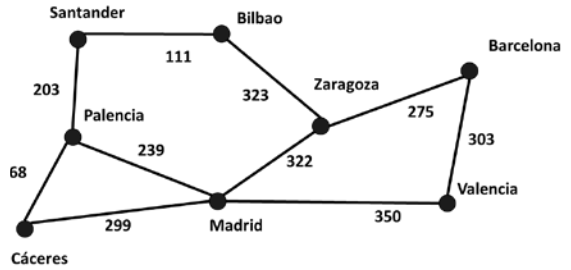


Tipos de problemas

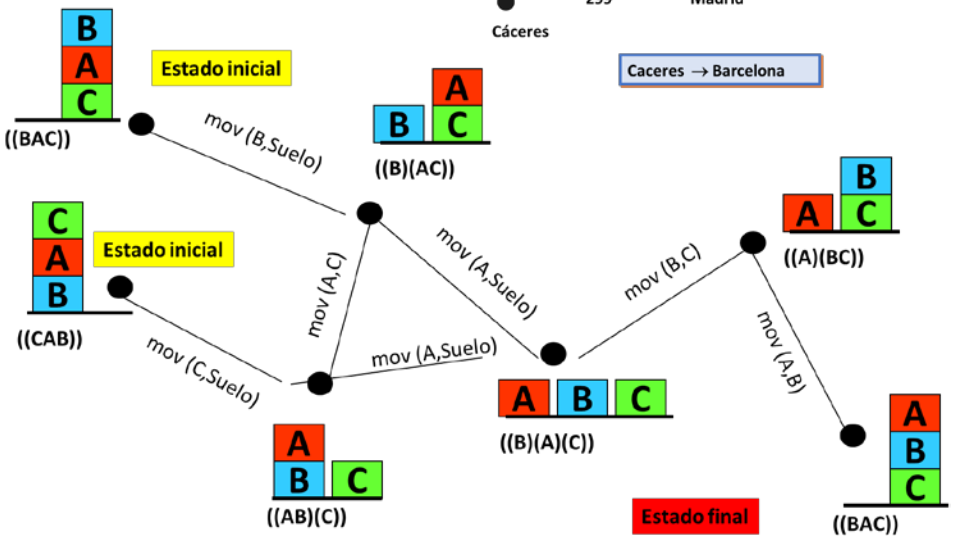
- Problemas de juguete
  - Descripciones concisas / exactas
  - Se usan para ilustrar las técnicas de resolución
  - Permiten comparar rendimiento







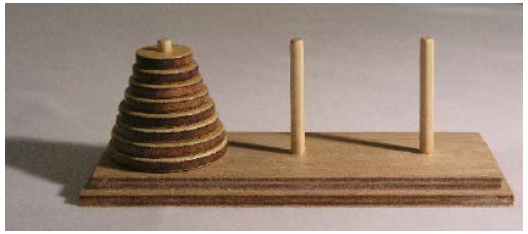
- Problemas reales
  - No hay una descripción única y consensuada
  - La representación afecta a la resolución
  - Lo importante es una solución



Cubo de rubik

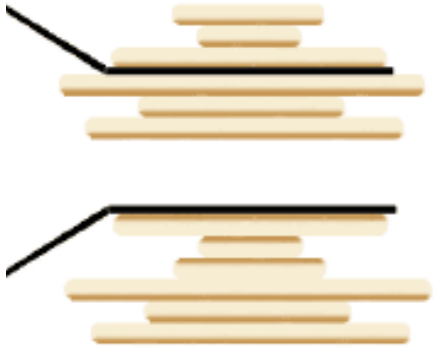
- Los humanos no resuelven el cubo de forma óptima (hacen más pasos)
- El cubo ha supuesto un reto desde que se inventó, ha sido objeto de estudio por parte de la inteligencia artificial, intentando buscar la solución óptima y creando heurísticas, bases de datos de patrones y otras técnicas que ayudaran a buscar la solución.
- En julio de 2010 Tomas Rokicki, Herbert Kociemba, Morley Davidson, y John Dethridge probaron que el número máximo de movimientos que hay que realizar para resolver cualquier estado del cubo de rubik es 20 → podas, simetrías, repeticiones,..
- Se puede resolver el cubo de rubik de manera sub-óptima ganando mucho tiempo de búsqueda
- Se suele abordar el problema por etapas (subobjetivos)
  - Dividir el desarrollo del cubo en etapas.
  - Cada etapa tiene un objetivo concreto (fácil de verificar)
  - Etapas en secuencia

Otros puzles

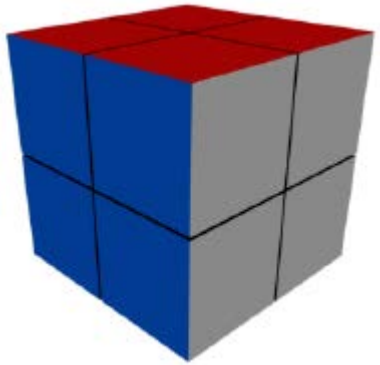


1883

1874

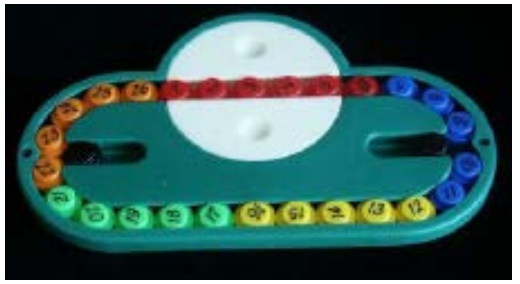


1975



Rush hour
1979

Topspin 20 1989



Topspin 26

1981

Sokoban

M12

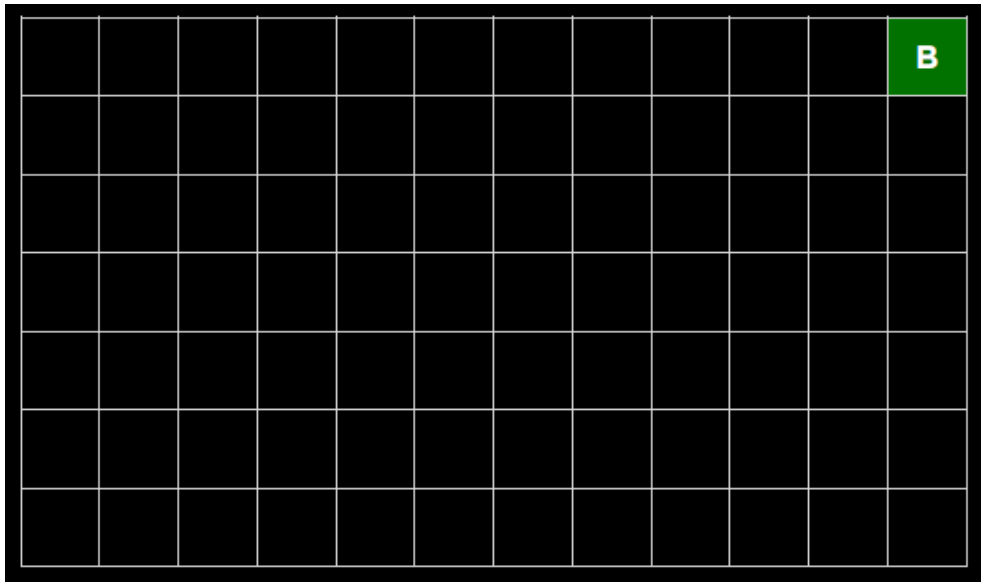
Tipos de problemas

El tipo de problema determina la búsqueda que se puede aplicar y el conocimiento que necesitamos para resolverlo

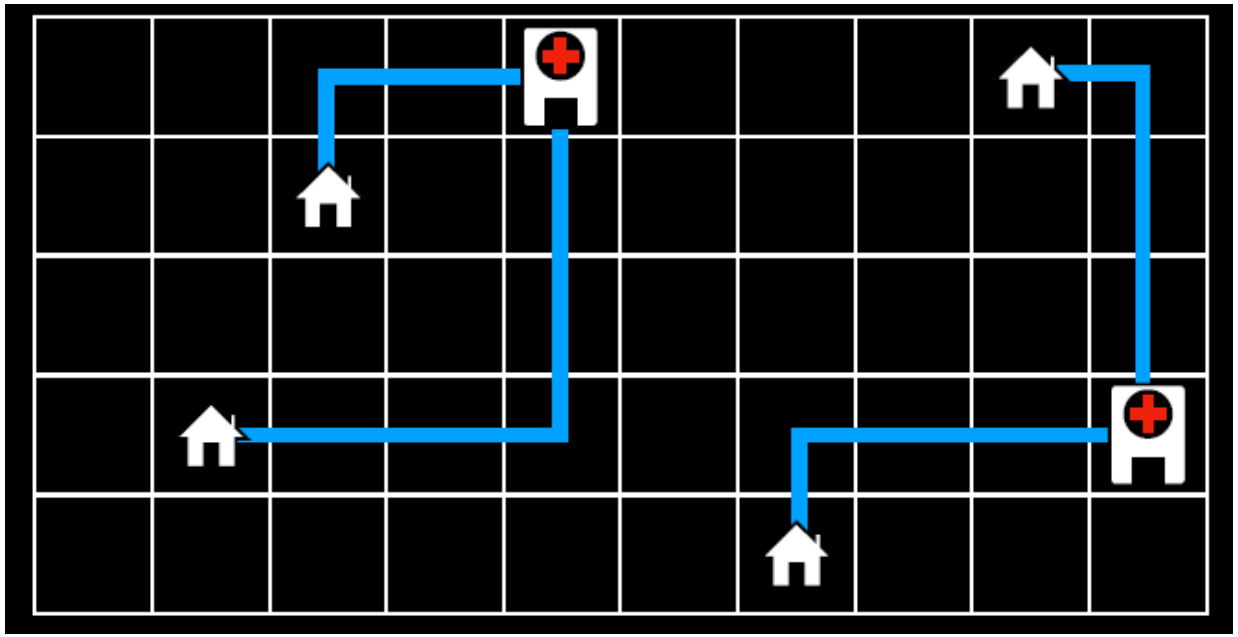
- Problemas de búsqueda clásica
  - El agente de búsqueda devuelve una secuencia de operadores (8 puzle)
- Problema de búsqueda con adversario
  - Los agentes de búsqueda devuelven un único operador (ajedrez)
- Problemas de optimización (problemas de configuración)
  - No importa el camino sino la solución
  - Búsqueda local / CSP
- Problemas lógicos (conocimiento) que usan inferencias lógicas y razonamiento para llegar a conclusiones → sistemas basados en reglas
- Problemas con incertidumbre → conocimiento probabilístico

Tipos de problemas (ejemplo)

La solución es el camino, el estado GOAL lo sabemos



La solución es la configuración de casas y hospitales de forma que se minimice la distancia de la casas al hospital más cercano



Tipos de problemas

- Ignorables → si se puede alcanzar solución desde E también se puede desde A(E)
  - Al ser el problema ignorable podemos usar estrategias irrevocables (sobre todo si el coste en memoria es alto, estados complejos)
  - Ignorables monótonos (se añaden hechos sin modificar los existentes) → lógica clásica monótona
  - Reversibles (con vuelta atrás)
- No ignorables (irrecuperables)

Representación de problemas

- Para que una máquina sea capaz de resolver un problema necesitamos:
  - Una **representación formal** del problema
  - Un **algoritmo** que resuelva el problema
- Espacio de estados: modelo conceptual que describe los elementos necesarios
  - Modelo de transición de estados
- Un ejemplo característico son los juegos de tablero
  - Las configuraciones del mundo (estados) corresponden con las configuraciones del tablero
  - Las transformaciones válidas corresponden a las reglas del juego → restricciones
- Conocimiento que hay que representar: reglas del juego, restricciones, elementos del problema, representación, estado final, calidad de los estados, heurísticas,.....
  - En esta primera parte vemos representaciones “libres” (estados simples)
  - Hay lenguajes específicos (STRIPS, PDDL) para modelar dominios → algoritmos de planning que permiten representaciones estructuradas para los estados.

Representación de problemas

Según el paradigma del **espacio de estados** un problema se representa mediante:

- **Estado inicial** *Vamos a ver problemas de estado único*
- **Operadores**: Acciones que transforman un estado en otro
- **Comprobación de objetivo:** Dado un estado permite determinar si es o no un objetivo
- **Función de coste de camino:** Suma de los **costes de los operadores** aplicados para llegar al estado objetivo desde el estado inicial

Espacio de Estados = grafo dirigido de todos los estados alcanzables desde el estado inicial aplicando operadores válidos..

El espacio de estados NO se construye entero ..

La jungla de cristal

- Descripción
  - 2 garrafas vacías con capacidades de 4 y 3 litros, respectivamente
  - El objetivo es que la garrafa de 4 litros ha de contener exactamente 2 litros
  - Tenemos un grifo para rellenarlas, un lavabo para vaciarlas, y posibilidad de trasvasar líquido de una garrafa a la otra, hasta que la 1ª se vacíe o la 2ª se llene

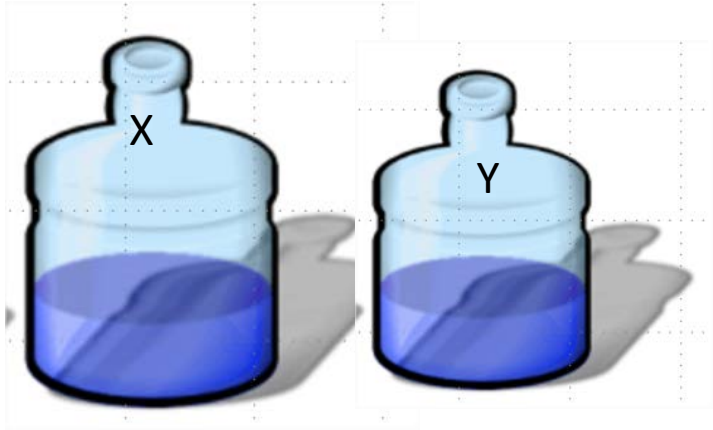
(X, Y)
Inicial: (0,0)
Objetivo: 2 y lo que sea (2, Y)

Operadores:
llena-4: llenar (del grifo) la garrafa de 4 litros
…

La jungla de cristal 3 https://www.dailymotion.com/video/xbcehl

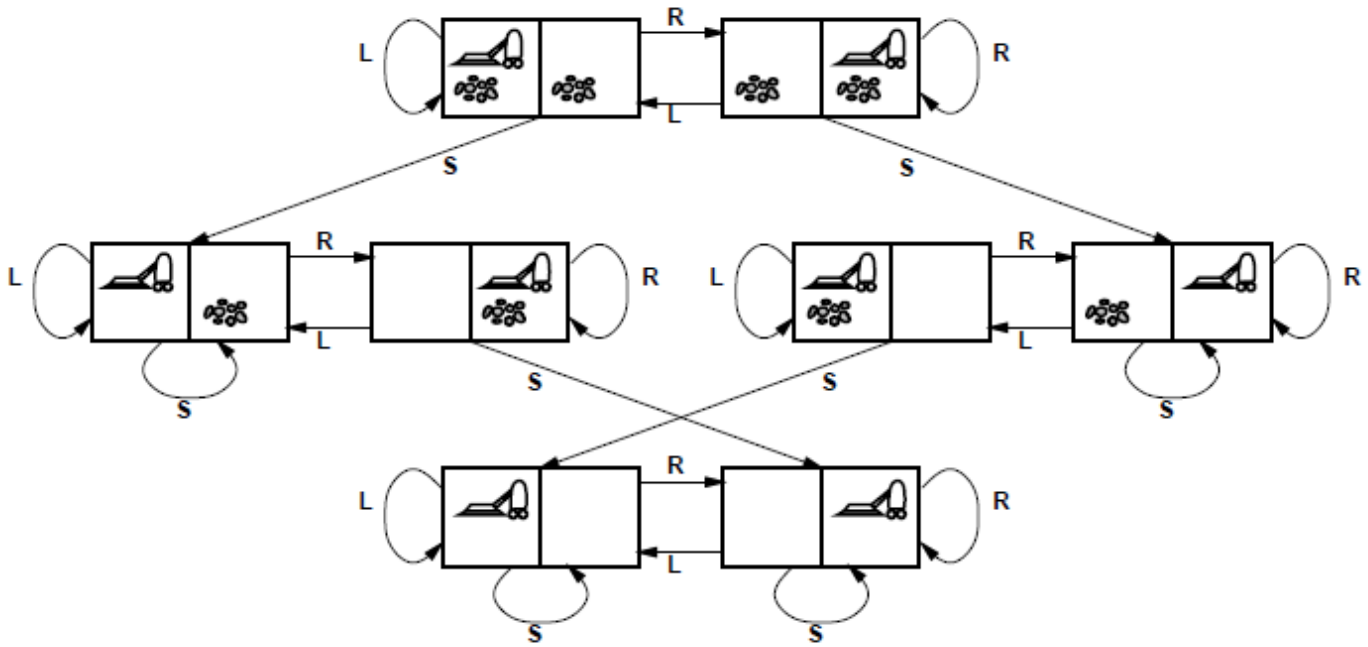
Formalizar el problema

- Especificación de operadores
  - *llena-4 (X, Y) → (4, Y)*
    - Precondición: *{ X < 4 }*
      - *Estamos asumiendo X natural ≤ 4, así que se aplicará si X = 0, 1, 2 o 3*
    - Acción: *construir el estado (4, Y)*
      - *Estamos asumiendo que llenamos la garrafa por completo*
  - *vacía-4 (X, Y) → (0, Y)*
    - Precondición: *{ X > 0 }*
    - Acción: *construir el estado (0, Y)*
      - *Estamos asumiendo que vaciamos la garrafa por completo*
  - *vierte-4 (X, Y) → (X', Y')*
    - Precondición: *{ X > 0 ∧ Y < 3 }*
    - Acción: *construir el estado (X', Y') donde*
      - *Y' = mínimo {3, X+Y}*
      - *X' = X – (Y' – Y)*



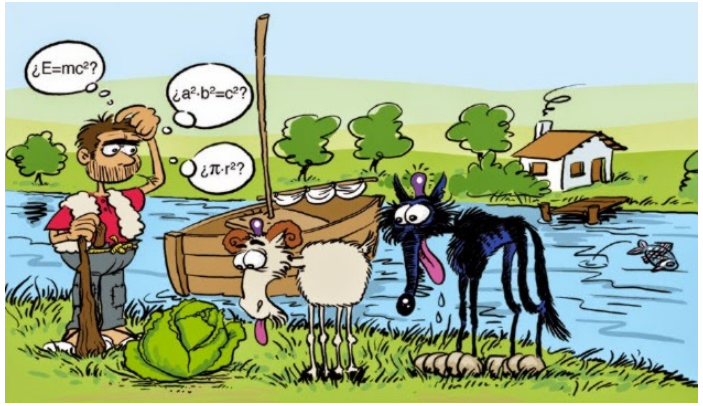
```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
    static: seq, an action sequence, initially empty
            state, some description of the current world state
            goal, a goal, initially null
            problem, a problem formulation

    state ← UPDATE-STATE(state, percept)
    if seq is empty then
        goal ← FORMULATE-GOAL(state)
        problem ← FORMULATE-PROBLEM(state, goal)
        seq ← SEARCH(problem)
    action ← RECOMMENDATION(seq, state)
    seq ← REMAINDER(seq, state)
    return action
```

Example: vacuum world state space graph



states??: integer dirt and robot locations (ignore dirt amounts etc.)
actions??: $Left$, $Right$, $Suck$, $NoOp$
goal test??: no dirt
path cost??: 1 per action (0 for $NoOp$)

Ejemplos simples: representación de problemas



http://www.novelgames.com/en/wolf/popup.php

Situación inicial:

Granjero	Lobo	Cabra	Col
Izq	Izq	Izq	Izq

Objetivo:

Granjero	Lobo	Cabra	Col
Der	Der	Der	Der

El lobo, la cabra y la col
Objetivo: todos han cruzado
Acciones: tener en cuenta las restricciones.

- 4 elementos: granjero, lobo, cabra y col
- 2 posiciones: derecha o izquierda
- Representación estado:

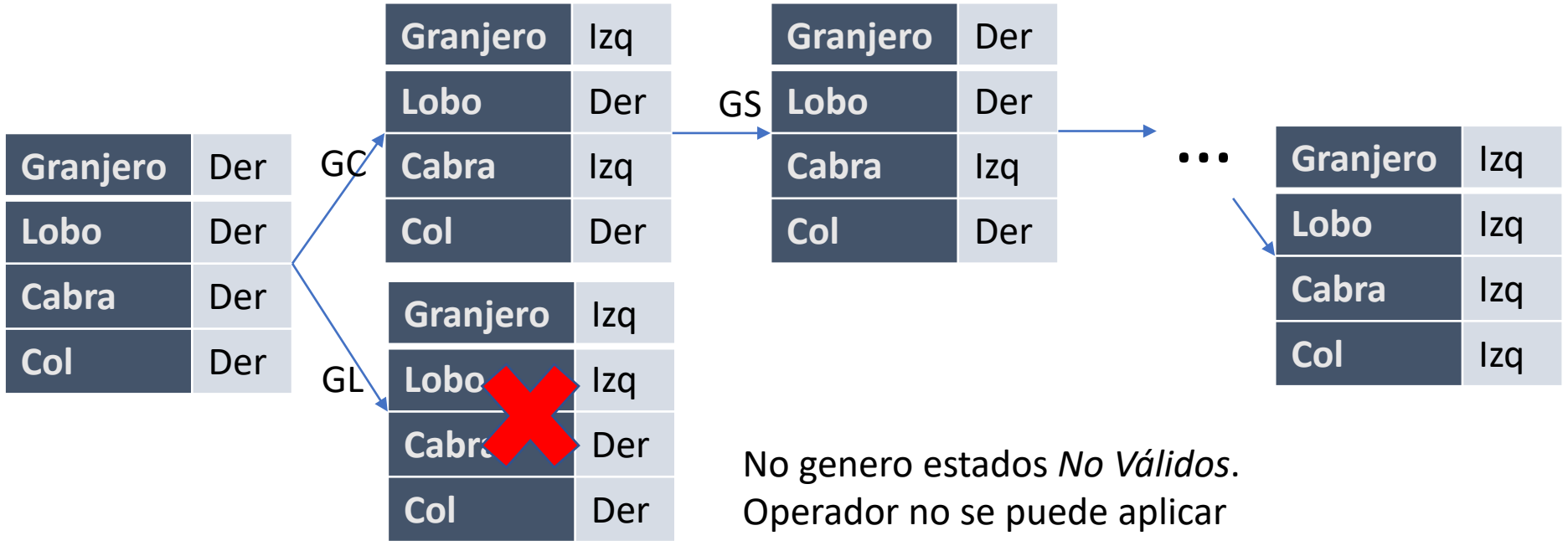
Granjero	Lobo	Cabra	Col
Der / Izq	Der / Izq	Der / Izq	Der / Izq

- Movimientos posibles: ¿cuántos?
  - Barca de izquierda a derecha con granjero solo
  - Barca de derecha a izquierda con granjero solo

  Barca con granjero solo (GS)
  Barca con granjero y con lobo (GL)
  Barca con granjero y con cabra (GC)
  Barca con granjero y con col (Gcol)

Búsqueda en el espacio de estados

- Buscar cómo llegar a un **estado final** a partir de un estado inicial
  - Dos estados están conectados si hay una operación que se pueda llevar a cabo para transformar un estado en el otro
  - La búsqueda corresponde a la resolución de problemas usando el método **genera y prueba**: *generadores* y *testers* "inteligentes"



Características de una buena representación

- **Representación de estados y transformaciones entre estados**
- ¿Cómo sabemos si una representación de conocimiento es buena?
  - ¿nos permite resolver el problema? ¿es sencillo comprobar estado objetivo?
  - Representar las restricciones naturales del problema
  - Transparente, concisa, completa, rápida y fácil de computar.
- Es muy importante trabajar al nivel de abstracción correcto.
  - Representar todo lo importante y nada que no sea importante.
- **La representación va ligada al problema concreto que queramos resolver pero debe ser reutilizable para variaciones del mismo problema**
- En el estado sólo incluimos la información variable
- La información fija del problema es una estructura/variables/constantes externas

Ejemplo. Colocar monedas

- Tenemos una mesa cuadrada **con n*n casillas** y una pila de monedas con **m monedas** iguales, siendo m tan grande como sea necesario. Empezamos a jugar con un amigo, colocando una moneda **en una casilla cualquiera**.
  Cada uno en su turno toma una moneda de la pila y la pone en la mesa según las reglas siguientes:
  - Cada moneda ocupa el espacio de una casilla de la mesa
  - **No se puede mover** ninguna de las monedas de la mesa
  - La moneda no puede salir del borde de la mesa
  - La moneda no puede estar encima de otra (ni total ni parcialmente)
  - **Pierde el jugador al que no le quede ninguna casilla disponible**
- Define el problema según el paradigma del espacio de estados, estableciendo una representación adecuada para los estados, el estado inicial y el estado final y precondiciones y postcondiciones de los operadores.

Representación

- Matriz n x n
  - Espacio de estados enorme 2^{n^2}
  - Cada estado ocupa n^2
  - Factor de ramificación enorme: n^2 operadores

No tiene sentido.. Pensar en lo que es imprescindible para resolver el problema

- Un estado se representa mediante **el número de casillas vacías** (X)
  Cada estado ocupa poco (un número) X= [0,n^2]
  El espacio de estados tiene n^2 estados posibles.
- Estado inicial: X= n^2
- Estado final: X=0
- Operadores: **Un único operador**

colocarMoneda (estado(x))
Precondición: **x > 0** (ojo, sin esta precondición el espacio de estados sería infinito)
Poscondición: estado(x) → estado (x-1)

El puzle deslizante (de varios tamaños)

- https://es.wikipedia.org/wiki/Juego_del_15

Juego del 15 resuelto

El juego del 15 o taken es un juego de deslizamiento de piezas que presentan un determinado orden inicial dentro de una cajita cuadrada. Tiene su origen en Estados Unidos, en el siglo XIX.

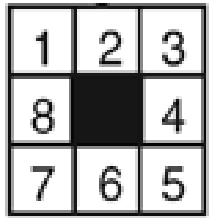
El taken es una cajita formada por 16 casillas de las cuales sólo quince están ocupadas.

Ejemplo de estado inicial: todas las fichas están colocadas en orden numérico, excepto la 14 y la 15, que tienen sus posiciones intercambiadas.

El juego consiste en maniobrar todas las fichas para corregir el error que hay en la fila inferior de la cajita, de manera que todas las fichas queden en orden consecutivo.

Representación del problema

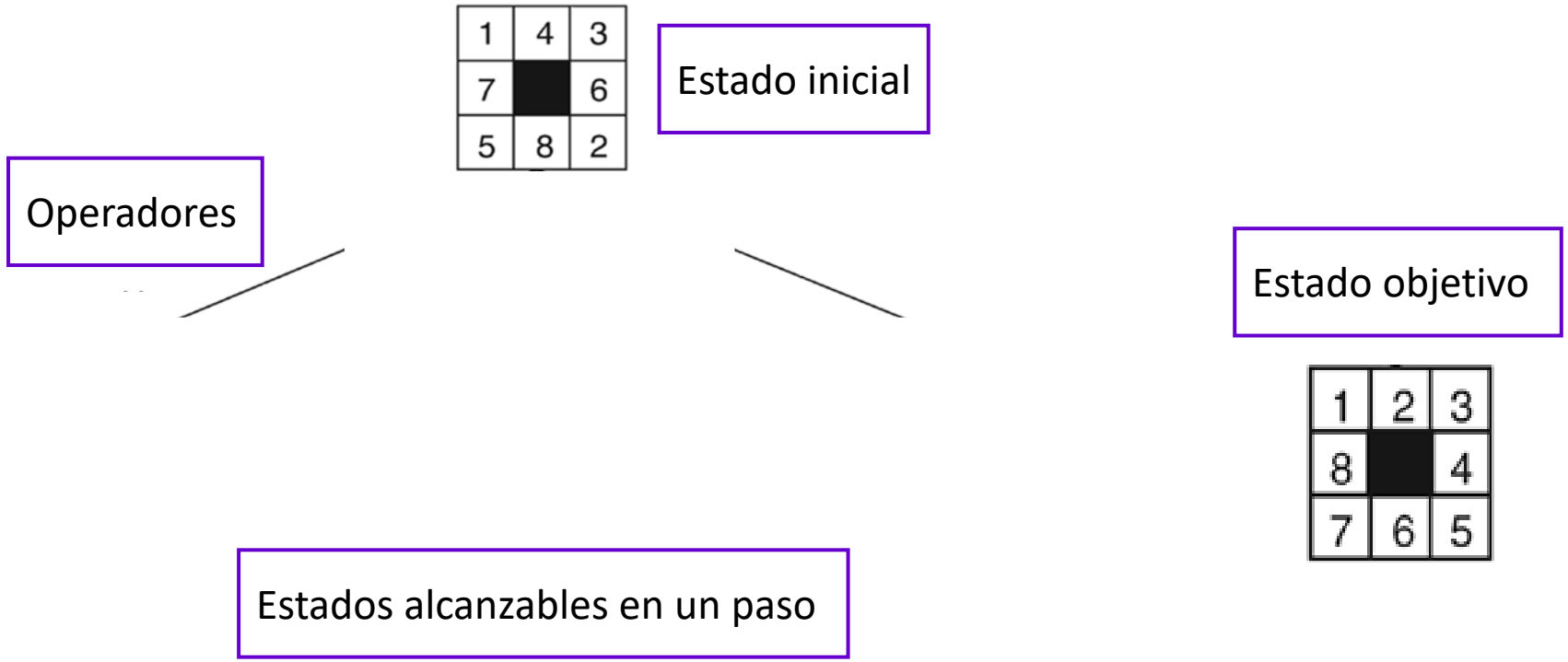
- Se representa un problema mediante abstracción
  - ➢ Eliminar detalles irrelevantes en la representación
- **Una representación puede simplificar o complicar la resolución del problema**



- **Ej: Representación del problema del 8-puzzle**
  - Números del 1-7 y un hueco.
  - El estado no debe incluir el material o el color del tablero (irrelevante)
  - Estados: localización de cada ficha (y hueco) en cada una de las 9 casillas
  - Estado inicial: puede ser cualquiera
  - Coste de operadores: cada operador tiene coste 1
  - Coste del camino: suma de aplicar operadores = número de pasos

Definición de **espacio de estados**

- **Grafo dirigido:**
  - Vértices = estados
  - Arcos = acciones, transformaciones entre estados
- Ejemplo: 8-puzzle



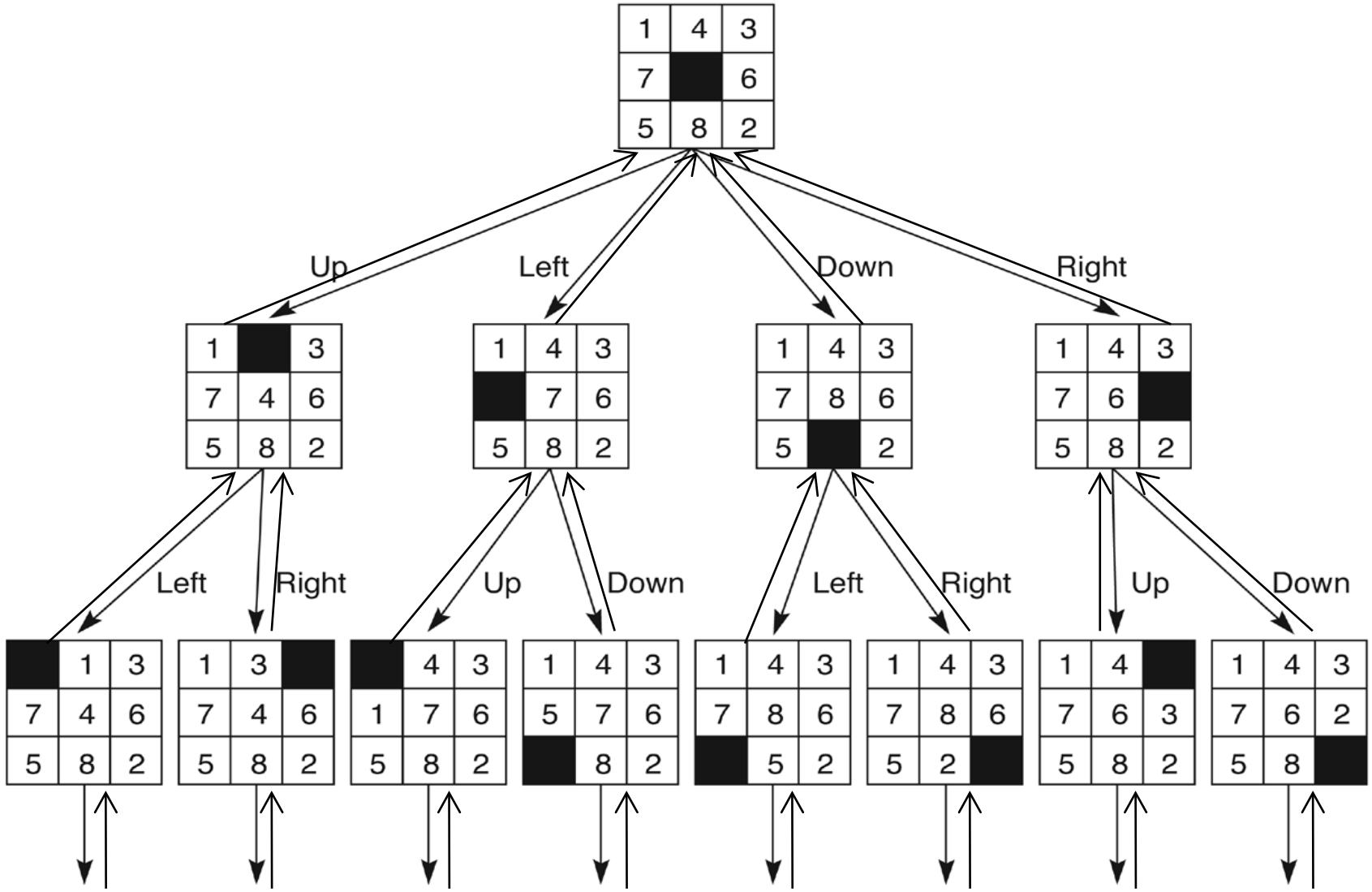
Representación de los operadores

- Estados y operadores "cooperan" para la resolución del problema → Su representación debe ser compatible
- Los operadores\acciones deben ser
  - Tan **generales** como sea posible
  - **Deterministas** (aplicados al mismo estado dan siempre el mismo resultado)
- Ejemplo del Puzzle: modos de definir operadores:
  1. Un operador por número (ficha) y movimiento: 32 (8*4) operadores

     Muevo el 4 arriba, muevo el 5 abajo

  2. Mejor opción definir 4 operadores que mueven **el hueco** arriba, abajo, derecha o izquierda

8-puzzle. Espacio de estados



```
C:\Datos\Clases\IA\[redacted]search>python eightpuzzle.py
A random puzzle:
-------------
| 2 | 7 | 5 |
-------------
| 1 | 4 |   |
-------------
| 3 | 6 | 8 |
-------------
BFS found a path of 11 moves: ['left', 'up', 'left', 'down', 'down', 'right', 'up', 'right', 'up', 'left', 'left']
After 1 move: left
-------------
| 2 | 7 | 5 |
-------------
| 1 |   | 4 |
-------------
| 3 | 6 | 8 |
-------------
Press return for the next state...
After 2 moves: up
-------------
| 2 |   | 5 |
-------------
| 1 | 7 | 4 |
-------------
| 3 | 6 | 8 |
-------------
Press return for the next state...
After 3 moves: left
-------------
|   | 2 | 5 |
-------------
| 1 | 7 | 4 |
-------------
| 3 | 6 | 8 |
-------------
Press return for the next state...
After 4 moves: down
-------------
| 1 | 2 | 5 |
-------------
|   | 7 | 4 |
-------------
| 3 | 6 | 8 |
-------------
Press return for the next state...
After 5 moves: down
-------------
| 1 | 2 | 5 |
-------------
| 3 | 7 | 4 |
-------------
|   | 6 | 8 |
-------------
Press return for the next state...
```

Niveles de representación de estados

- Ejemplo: Representación de los estados del 8-puzzle
  - Descripción:
    - Estados: localización de cada ficha y del hueco en cada una de las 9 casillas
  - Implementación: Muchas opciones
    - matriz 3*3,
    - vector de longitud 9,
    - conjunto de hechos {(superior_izda = 3), (superior_centro = 8), ...}

Conceptualización

Formalización

Qué

Descripción (especificación) de estados y operadores a nivel abstracto con diagramas o texto (pseudocódigo)

Cómo

Implementación de la Representación:

Define estados mediante una estructura de datos/conocimiento

Define operadores en un lenguaje formal

Implementación de estados y operadores en un formalismo o lenguaje

En el caso de un puzle de tamaño **3×3 (8-puzle)** el número de posibles estados es de 9!=362.880 (realmente por las simetrías son 9!/2 = 181.440)

Este tamaño permite, con la capacidad de los ordenadores actuales, hacer una búsqueda exhaustiva para encontrar el camino desde cualquier estado al estado final.

Este método encontrará solución aunque, posiblemente, de una forma completamente in

- Si jugamos con el puzle de tamaño **4×4 (15-puzle)** el número de posibles estados es de $16! \approx 2 \times 10^{13}$, **demasiado grande para una búsqueda exhaustiva**.
  Necesitamos estrategias de búsqueda que nos permitan alcanzar las soluciones de forma más eficiente y usando menos recursos (en tiempo y en espacio almacenado).

No todos los estados iniciales tienen solución.

- Sam Loyd en 1878 “llevo al mundo a la locura” por un premio de $1.000 a quien fuera capaz de resolver el **famoso puzzle de 15**, el cual no tenía solución debido que se necesitaba un número impar de permutaciones y, **como veremos en la práctica**, **sólo un número de movimientos par** tiene solución.
- **El número de posibles estados iniciales es n!, siendo n el número de fichas**. Por tanto, en nuestro caso, tendremos más de 130.000 millones de posibles estados iniciales. Sin embargo, sólo la mitad de esas combinaciones tiene solución.

Algunos datos del ajedrez

Leontxo García, libro *Ajedrez y ciencia, pasiones mezcladas*,

Leyenda del rey *Sheram*, de la India, que quiso premiar al inventor del ajedrez con cualquier cosa que pidiera. El inventor le pidió un grano de trigo por la primera casilla del tablero, dos por la segunda, cuatro por la tercera, ocho por la cuarta, y así, sucesivamente, **doblando el número cada vez, hasta llegar a la casilla 64**.
El rey pensó que era un regalo muy simple, pues él era muy rico.
Hasta que hicieron los cálculos del grano de trigo que debía desembolsar su Majestad: en la casilla 64 **habría 9.223.372.036.854.775.808** granos de trigo, que sumados a los del resto del tablero, quedan en 18.446.744.073.709.551.615. Es decir, **más de 18 trillones de granos de trigo**.
Los consejeros de la corte estimaron que sería necesario acumular la cosecha de trigo en todo el mundo **durante 2.000 años para poder pagar la deuda**.

¿Cuántos barcos de 100.000 toneladas falta para transportar todo ese trigo? Pues nada menos que 3.689.348 barcos.
¿Y cuánto espacio ocuparían esos cargueros en el mar si los pusiéramos en fila, uno detrás de otro? Darían 17 vueltas al planeta.

Los misioneros y los caníbales

- 3 misioneros y 3 caníbales en la orilla de un río junto con 1 bote
- El objetivo es que pasen todos a la otra orilla
- Hay tres restricciones
  - Deben cruzar usando el bote
  - En el bote sólo pueden ir 1 o 2 personas
  - En ninguna de las orillas puede haber más caníbales que misioneros

Los misioneros y los caníbales. Representación Estados. Opción 3

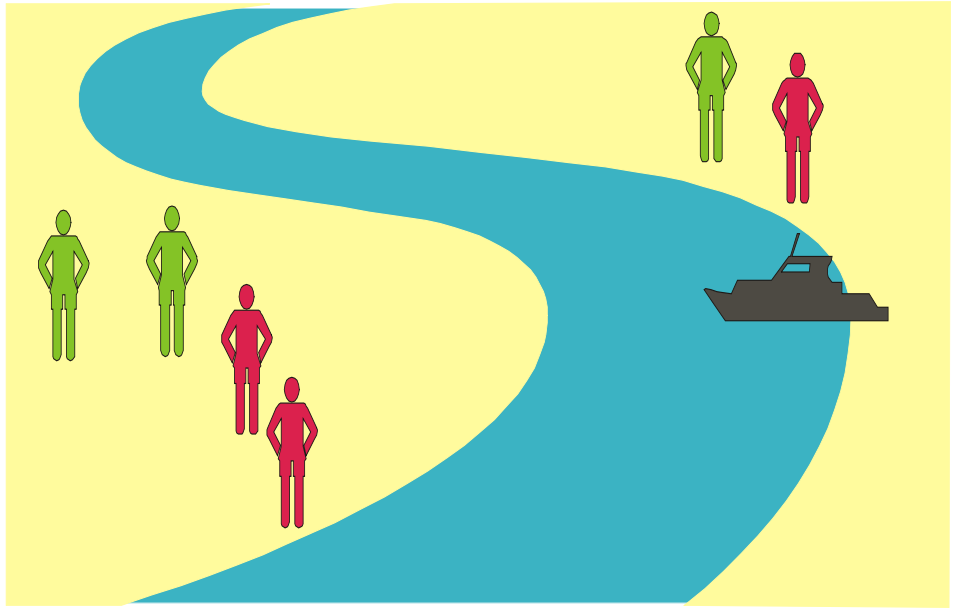
- Estado: Nº de misioneros, caníbales y bote en la **orilla de partida**
- Acordamos que la orilla de partida sea el margen **izquierdo** del río
- Estado = *(NM, NC, B)*
  - *NM* es el número de misioneros en la orilla izquierda *(0, 1, 2 ó 3)*
  - *NC* es el número de caníbales en la orilla izquierda *(0, 1, 2 ó 3)*
  - *B* indica si el bote está en la orilla izquierda *(0 = NO ó 1 = SÍ)*
- El sitio donde está el bote es fundamental para los viajes → Determina si son o no aplicables ciertos operadores

(2, 2, 0) ≠ (2, 2, 1)

2M **no** es aplicable

2M **sí** es aplicable

Los misioneros y los caníbales. Representación Estados

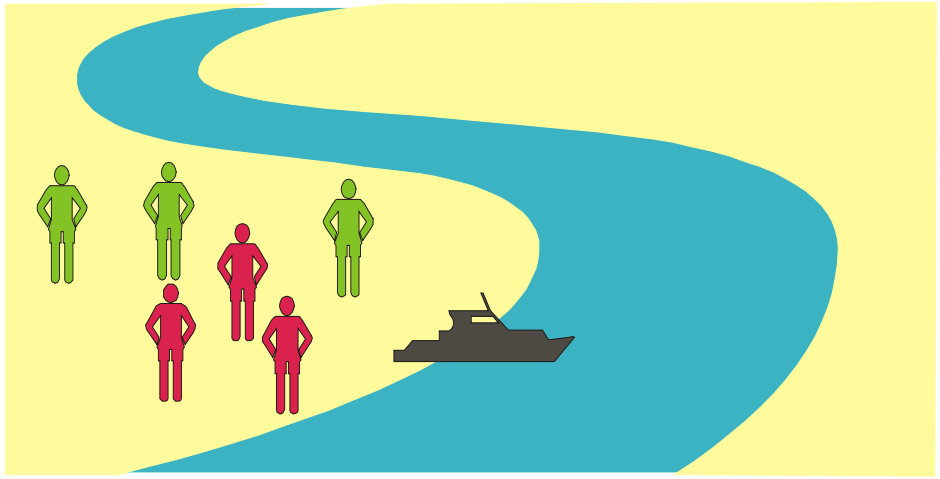


(der, der, izq, der, der, izq, izq)

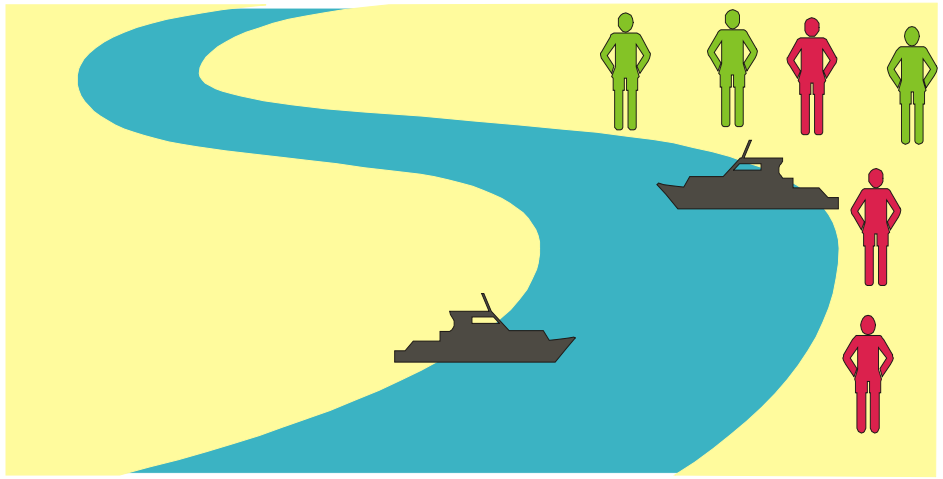
(2, 2, 1, 1, der)

(2, 2, 0)

Los misioneros y los caníbales. Estado inicial/objetivo



Estado inicial (3, 3, 1)



Estado objetivo (0, 0, 0)

(0, 0, 1) no es posible

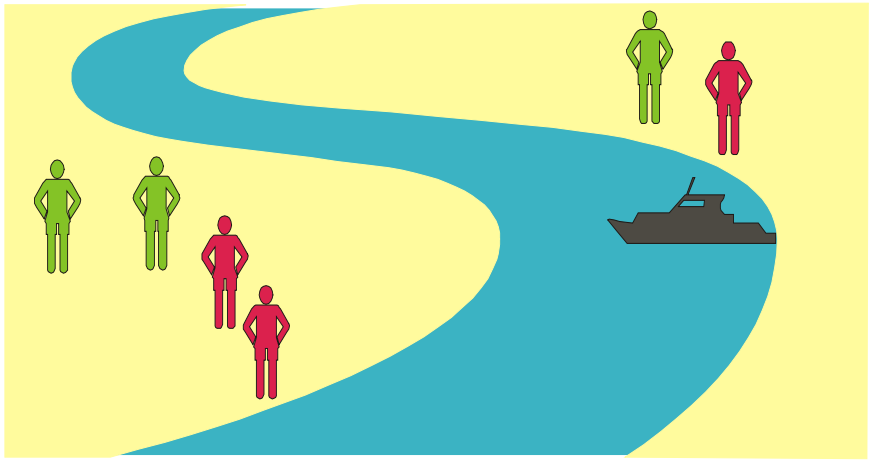
Las restricciones del problema es conocimiento que ponemos en la formalización del problema.

La definición de qué estados son peligrosos y no cumplen las restricciones del problema es también conocimiento del problema que usamos en su resolución.

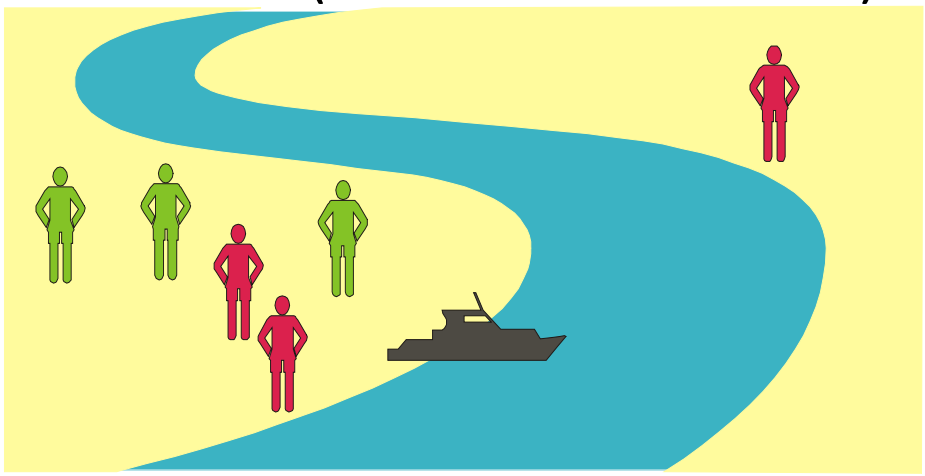
Una representación de conocimiento adecuada → posibilita la resolución de los problemas

Los misioneros y los caníbales. Estados intermedios

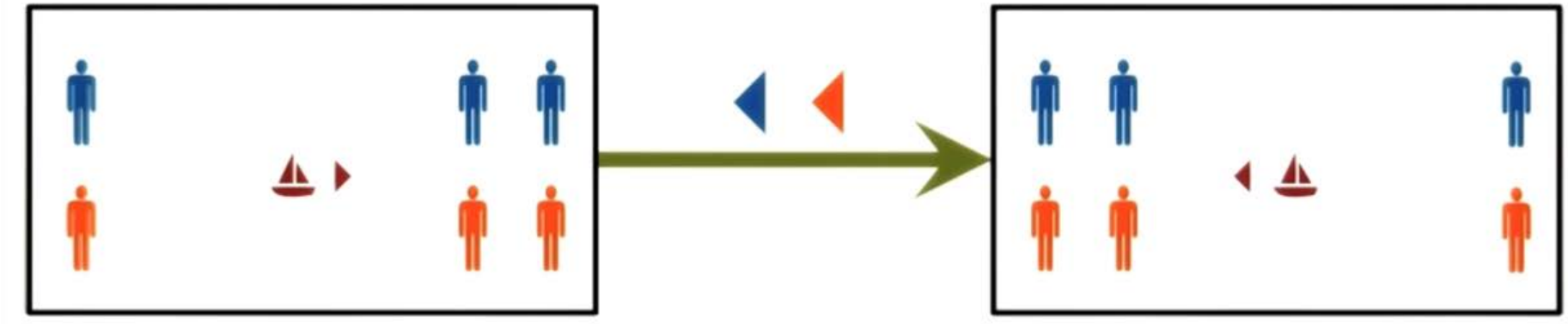
Los estados intermedios deben de ser no peligrosos
Volver a una situación anterior no tiene sentido (no meterse en ciclos)



(2, 2, 0)



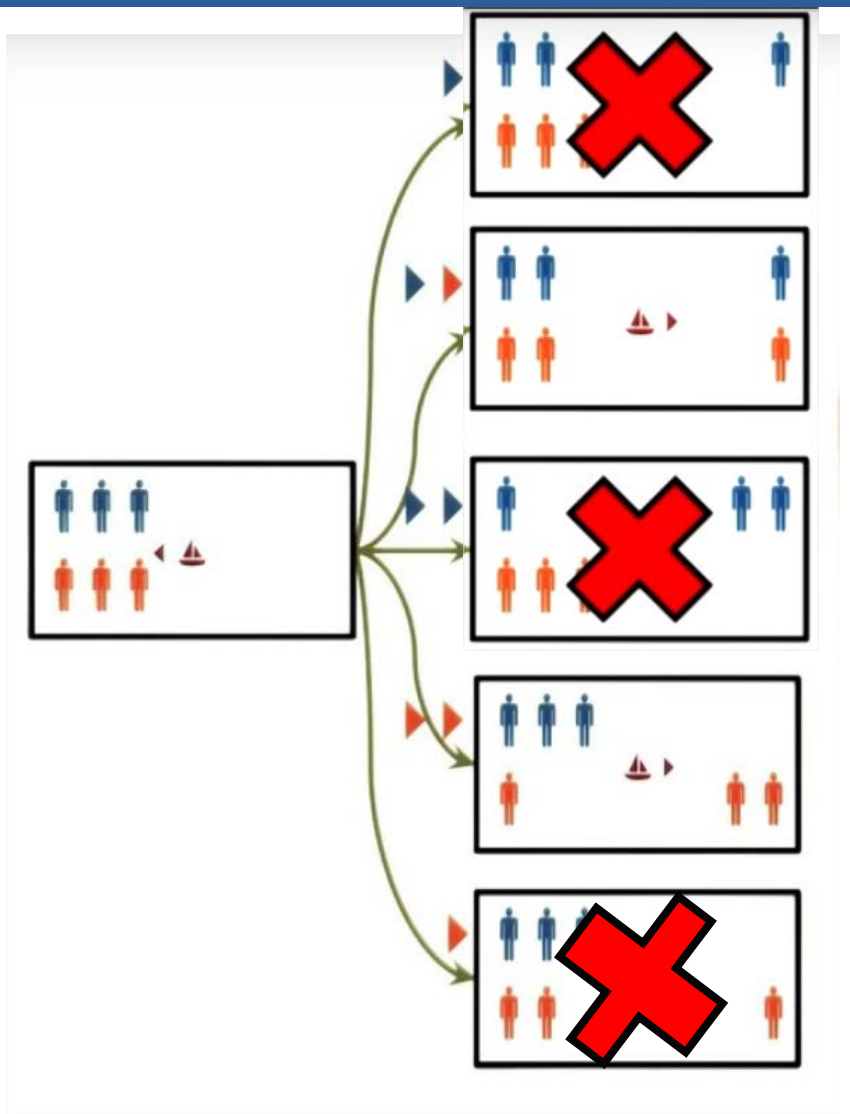
(3, 2, 1)



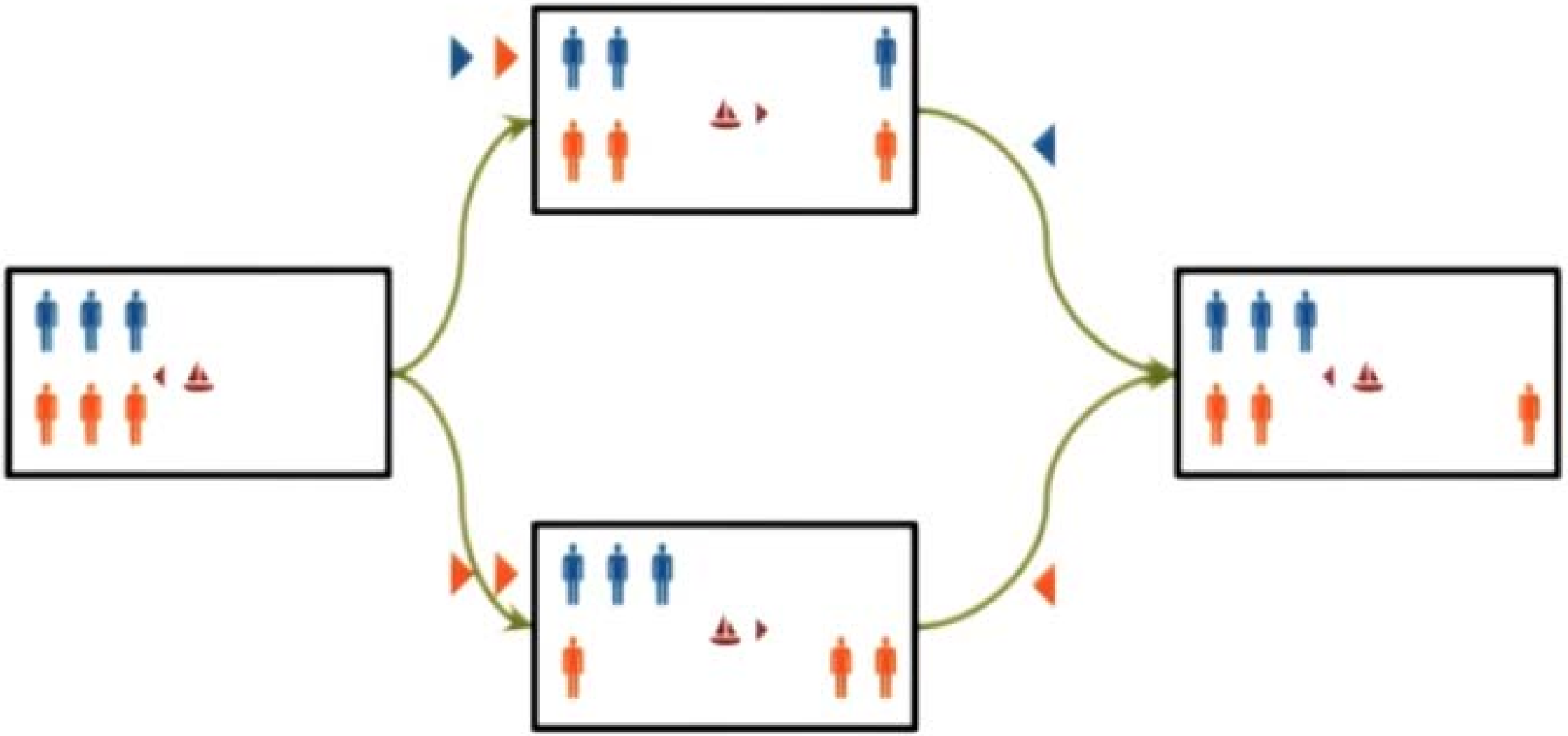
(1,1,0)

(2,2,1)

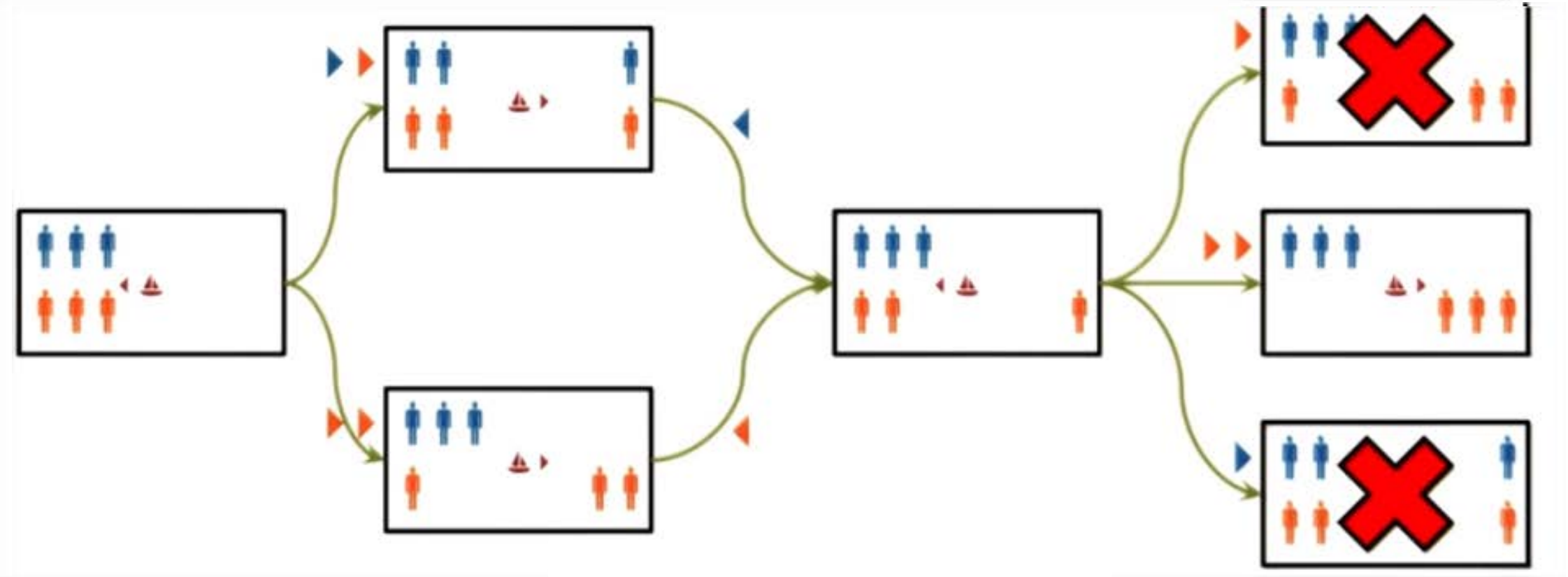
Caníbales: naranjas.

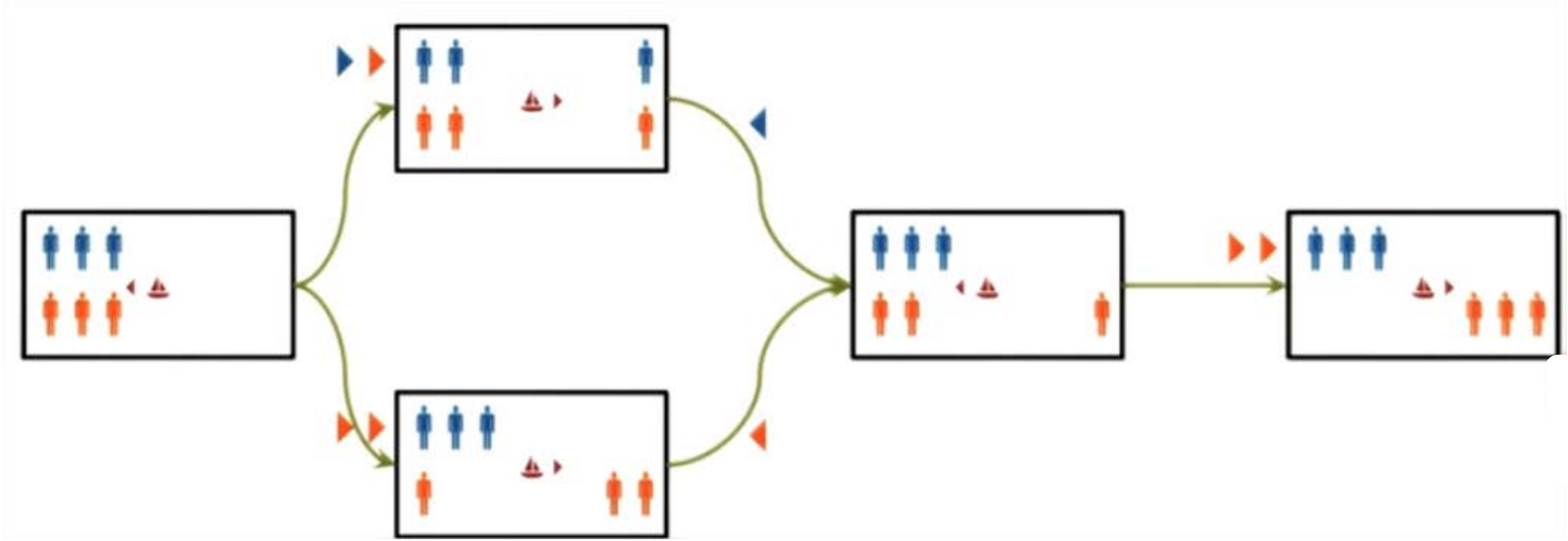


Genera y prueba debe filtrar movimientos ilegales y los no efectivos



Solución





Los misioneros y los caníbales. Formalización de operadores

- 5 operadores posibles:
  - Cruza 1 misionero: *cruzaM*
  - Cruzan 2 misioneros: *cruzaMM*
  - Cruza 1 caníbal: *cruzaC*
  - Cruzan 2 caníbales: *cruzaCC*
  - Cruza 1 misionero y 1 caníbal: *cruzaMC*
- Asumimos coste operador = 1

- Especificación de operadores:
  - El sitio donde está el bote es fundamental. Por ejemplo, no podría cruzar ningún misionero en los estados *(0, _, 1)* y *(3, _, 0)*
  - *cruzaM (NM, NC, B)*
    - Precondiciones:
      $\{ (NM > 0 \wedge B = 1) \vee (NM < 3 \wedge B = 0) \}$
    - Postcondiciones:
      - $(NM, NC, 1) \rightarrow (NM-1, NC, 0)$
      - $(NM, NC, 0) \rightarrow (NM+1, NC, 1)$

Falta comprobar condición de peligro

- Función de coste de camino = número de veces que se cruza el río

Los misioneros y los caníbales. Situaciones de peligro

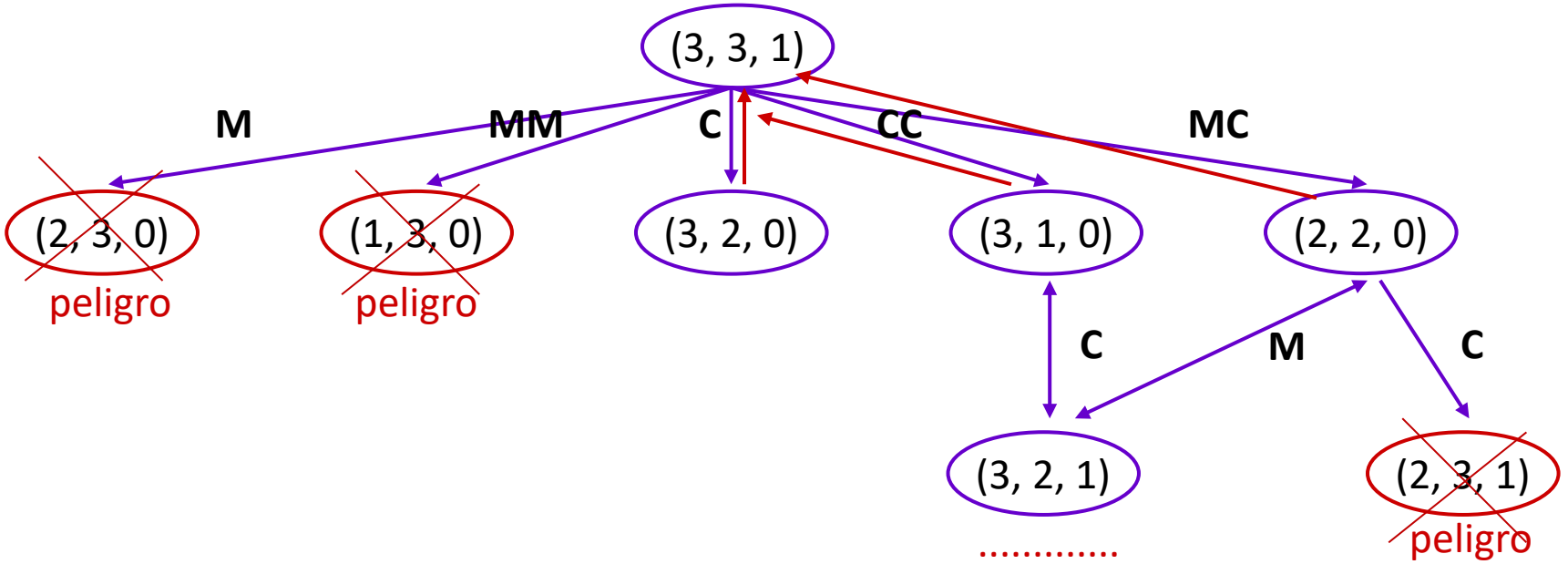
- En el bote no hay peligro (por viajar un máximo de 2 personas en él) → Si el máximo fuese 3 o 4, sí habría peligro *(y sería otro problema)*
- Hay que comprobar la condición de peligrosidad en las orillas (en estados)
  - ***(3, 3, 1)*** y *(2, 2, 0)* no son estados peligrosos
  - *(1, 2, 0)* y *(2, 3, 0)* son ejemplos de estados peligrosos en la orilla izquierda.
- Condición de peligrosidad
  - *Opción 1:* ***NM < NC*** ***➔ claramente no funciona***
    - En *(2, 1, 0)* ¿no hay peligro? → ¡En la orilla derecha hay 1 misionero con 2 caníbales!
    - En *(0, 2, 1)* ¿hay peligro? → ¡Si no hay misioneros no hay peligro!
    - **Sólo se mira la seguridad en la orilla izquierda pero no en la orilla derecha**
  - *Opción 2:* $(NM < NC \wedge NM \neq 0) \vee (NM > NC \wedge NM \neq 3)$

0 < NM < 3 y NM <> NC

Los misioneros y los caníbales. Tamaño del espacio de estados

- 4*4*2 = **32 estados posibles** *(NM, NC, B)*
  - 4 estados son inalcanzables
    - Obvios: (0, 0, 1) y (3, 3, 0)
    - No tan obvios: (3, 0, 1) y (0, 3, 0)
  - 28 estados posibles
- De los 28 estados posibles:
  - 12 estados de peligro
    - (1, 2, _), (2, 3, _), (1, 3, _), (2, 1, _), (2, 0, _), (1, 0, _)
  - Se generan los 16 estados legales o seguros
- **El espacio de estados estaría compuesto por 16 estados**
- A veces la condición de peligrosidad (o alguna otra restricción sobre el nuevo estado) se codifica dentro del operador como poscondición
  - **Genera y prueba**
  - No generar los estados de peligro

Los misioneros y los caníbales. Espacio de estados



Ejercicio: Desarrollar el resto del espacio de estados identificando los estados en peligro (si se identifica un estado de peligro no se sigue por ahí, no se cumplen las condiciones del problema)

Los misioneros y los caníbales. Ejemplo de representación formal

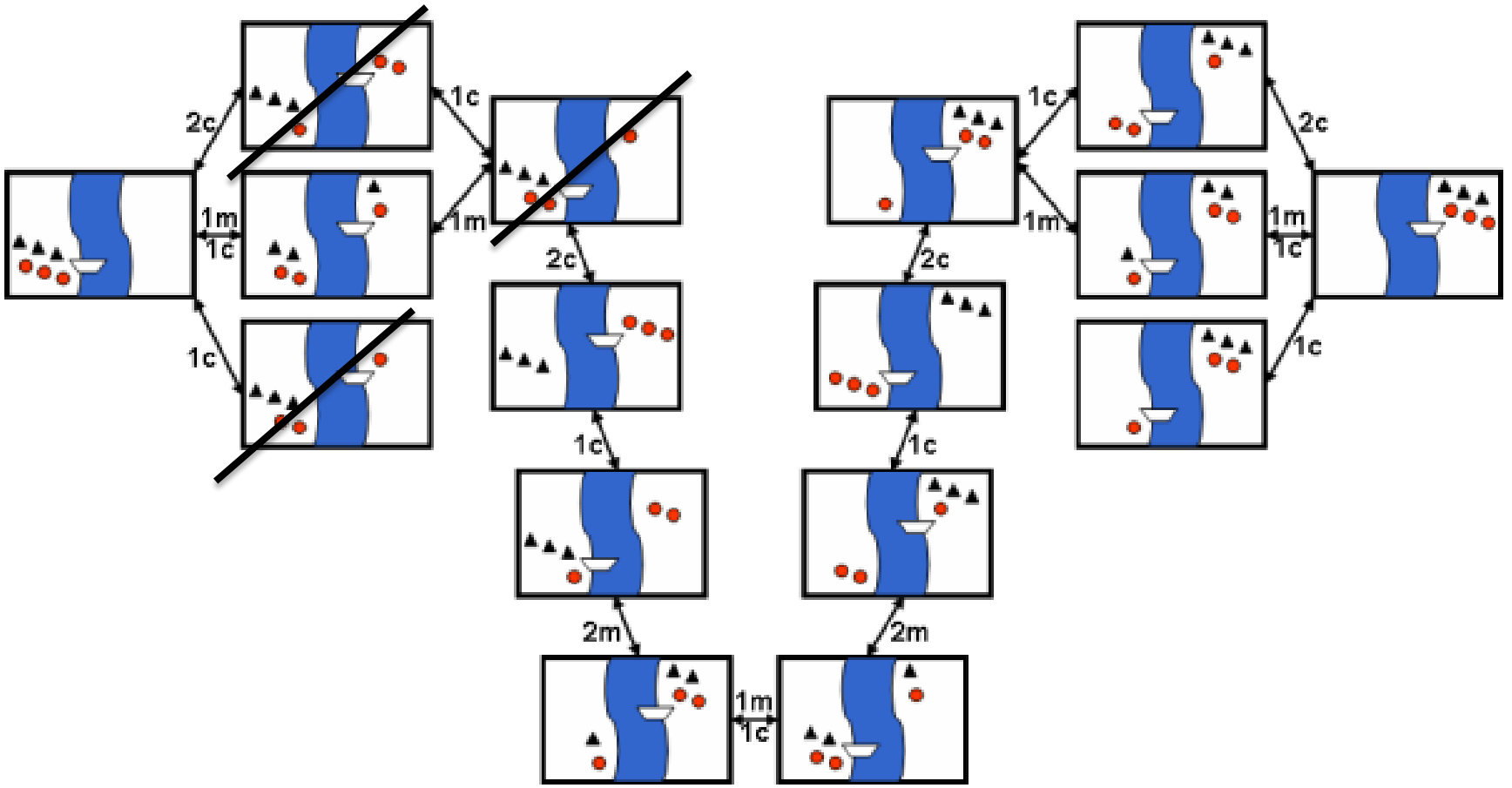
- Estado inicial: **(3, 3, 1)**
- Estado objetivo: **(0, 0, 0)**
- Restricciones: estados no válidos (condición de peligro **(NM, NC, B)**)
  - **$(NM < NC \wedge NM \neq 0) \vee (NM > NC \wedge NM \neq 3)$**
- Operadores: **cruzaM, cruzaMM, cruzaC, cruzaCC, cruzaMC**
  - Especificación **cruzaM (NM, NC, B)**
    - Precondiciones
      - **$(B = 1 \wedge NM > 0) \vee (B = 0 \wedge NM < 3)$**
    - Postcondiciones
      - **$(NM, NC, 1) \rightarrow (NM - 1, NC, 0)$**
      - **$(NM, NC, 0) \rightarrow (NM + 1, NC, 1)$**
      - **Estado destino no peligroso**
    - Coste del operador: 1
  - …
- Coste de la solución = número de operadores aplicados

Los misioneros y los caníbales. Conclusiones

- La representación de los estados afecta a
  - la facilidad/dificultad de la especificación de operadores
  - la complejidad para resolver el problema
- *(M1, M2, M3, C1, C2, C3, B)*
  - *cruzaM1 (M1, M2, M3, C1, C2, C3, B)*
    - Precondiciones: *{ M1 = B }*
    - Postcondiciones:
      - *si B = izquierda entonces M1 := derecha* $\wedge$ *B := derecha*
      - *si B = derecha entonces M1 := izquierda* $\wedge$ *B := izquierda*
      - *estado destino no peligroso*
  - Habría que especificar 21 operadores
    - 3 para cruzar a un misionero, 3 para cruzar a un caníbal
    - 3 para cruzar a dos misioneros, 3 para cruzar a dos caníbales
    - 9 para cruzar a un caníbal y a un misionero
- *(NM_OI, NC_OI, NM_OD, NC_OD, B)* → La información redundante supone hacer cambios en más componentes

Otra versión. Nueva situación de peligro.

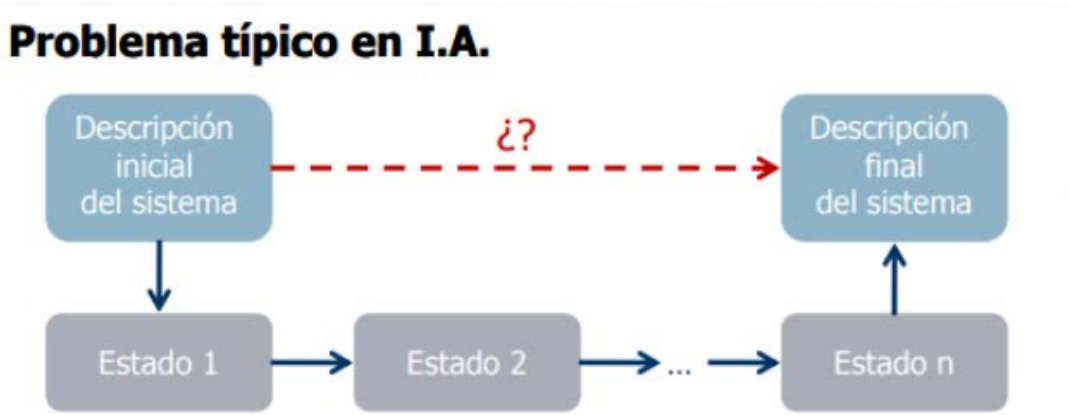
- Tres misioneros se perdieron explorando una jungla. Separados de sus compañeros, sin alimento y sin radio, sólo sabían que para llegar a su destino debían ir siempre hacia adelante. Los tres misioneros se detuvieron frente a un río que les bloqueaba el paso, preguntándose que podían hacer. De repente, aparecieron tres caníbales llevando un bote, pues también ellos querían cruzar el río. Ya anteriormente se habían encontrado grupos de misioneros y caníbales, y cada uno respetaba a los otros, pero sin confiar entre ellos. **Los caníbales se comían a los misioneros cuando les superaban en número, y los misioneros bautizaban a los caníbales en situaciones similares.** Todos querían cruzar el río, pero el bote no podía llevar más de dos personas a la vez y los misioneros no querían que los caníbales les aventajaran en número. ¿Cómo puede resolverse el problema, sin que en ningún momento hayan más caníbales que misioneros en cualquier orilla del río?


2c
1c
1m
1c
1c
1m
2c
1c
2m
1m
1c
1c
1m
2c
2c
1c
2m
1m
1c
2c
1c

Ejercicio: test chino

1 - Todo el mundo tiene que cruzar el río utilizando la balsa.
2 - La balsa sólo tiene capacidad para dos personas.
3 - Los únicos que saben manejar la balsa son la madre, el padre y el policía. Sin uno de ellos la balsa no se mueve.
4 - El padre no puede permanecer con ninguna de sus hijas sin que esté presente la madre.
5 - La madre no puede permanecer con ninguno de sus hijos sin que esté presente el padre.
6 - El ladrón roba a la familia si no está el policía: no puede permanecer con ningún miembro de la familia sin la presencia del policía (pero sí puede estar solo)

Ejercicios para practicar



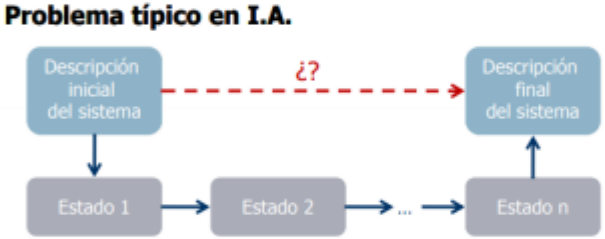
Aunque en algunos de los problemas se proponen cuestiones particulares, para todos los problemas se pide resolver las siguientes cuestiones generales:

a) Proponer una representación de los posibles estados y describir el espacio completo de estados. ¿Qué tamaño tiene este espacio?
b) Indica un estado inicial
c) Indica la función de comprobación de objetivo.
d) Define las acciones y función de transición válida para resolver el problema como una búsqueda en el espacio de estados.
e) Indica cual es el factor de ramificación teórico (máximo o medio) del árbol de búsqueda.
f) Define una función de coste.
g) Define una función heurística h' que sirva para medir la bondad de un estado respecto del objetivo buscado. Analiza las propiedades de admisibilidad y consistencia de h'.

Colección genérica de problemas de representación para el Tema 2

Se incluye una colección de problemas de sencillos que han sido recopilados de distintas fuentes y que se usan en los cursos de IA clásica para ejemplificar las técnicas de representación y resolución de problemas más sencillas. Algunos de estos problemas puede que los conozcáis de las asignaturas de algoritmia y/o de las explicaciones de clase y/o de exámenes de otros años. Algunos pueden proponerse para resolverlos en el laboratorio.

Cada problema se podría resolver con distintos métodos y se pueden plantear distintas alternativas para la representación y resolución del problema. Los problemas de los que se ocupa la I.A. son NP-difíciles aunque algunos problemas en el tamaño presentado en esta colección no lo son. El objetivo es que sirvan de práctica de la representación y resolución de problemas del Tema 2 de la asignatura IA 1.



Aunque en algunos de los problemas se proponen cuestiones particulares, para todos los problemas se pide resolver las siguientes cuestiones generales:

a) Proponer una representación de los posibles estados y describir el espacio completo de estados. ¿Qué tamaño tiene este espacio?
b) Indica un estado inicial
c) Indica la función de comprobación de objetivo.
d) Define las acciones y función de transición válida para resolver el problema como una búsqueda en el espacio de estados.
e) Indica cual es el factor de ramificación teórico (máximo o medio) del árbol de búsqueda.
f) Define una función de coste.
g) Define una función heurística h' que sirva para medir la bondad de un estado respecto del objetivo buscado. Analiza las propiedades de admisibilidad y consistencia de h'.

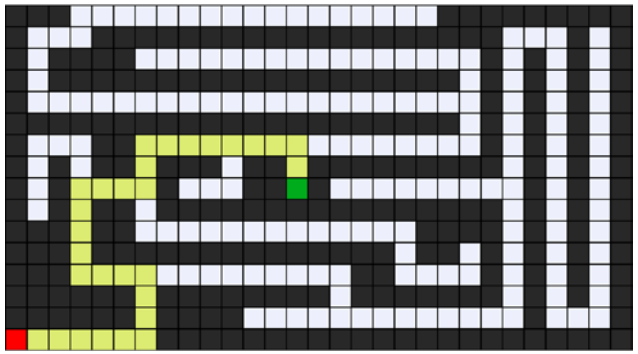
¿Por qué estudiar búsqueda y no otros temas más de moda?

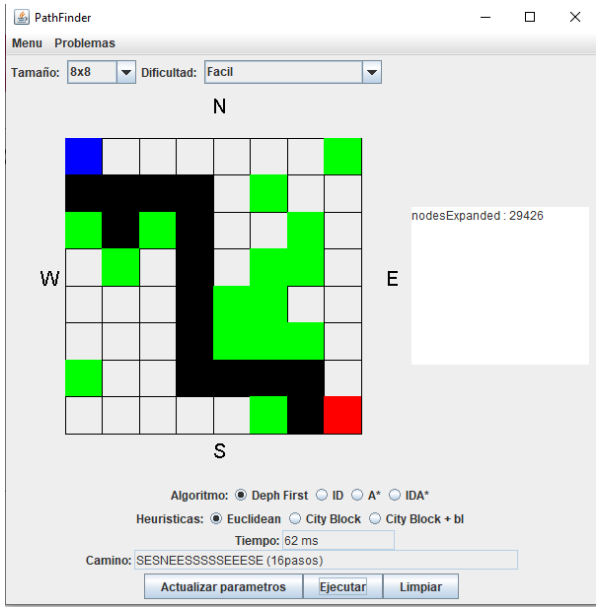
- Aplicaciones para encontrar rutas: viajes, redes de teléfonos…
- Planificación de los movimientos de un robot
- Videojuegos
- Muchos problemas no se pueden resolver de otra forma
- Es un área de investigación viva

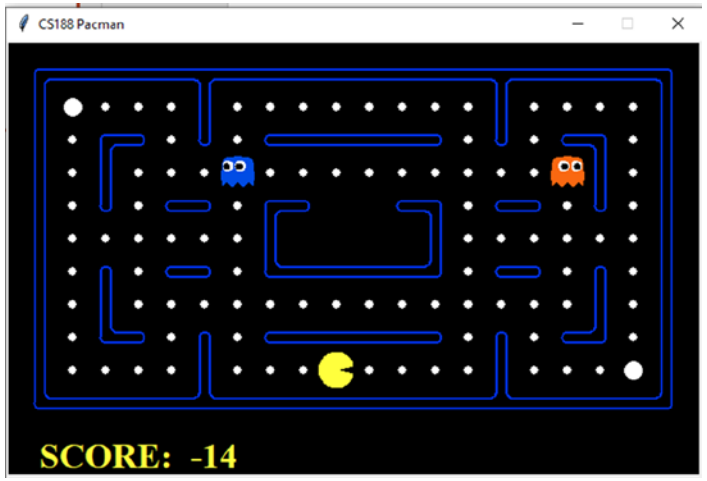
- Proporciona un marco de referencia
  - Muchos problemas de IA se pueden conceptualizar como búsqueda aunque ser resuelvan en la práctica por otro método
- Explora ideas que se pueden exportar a otros campos

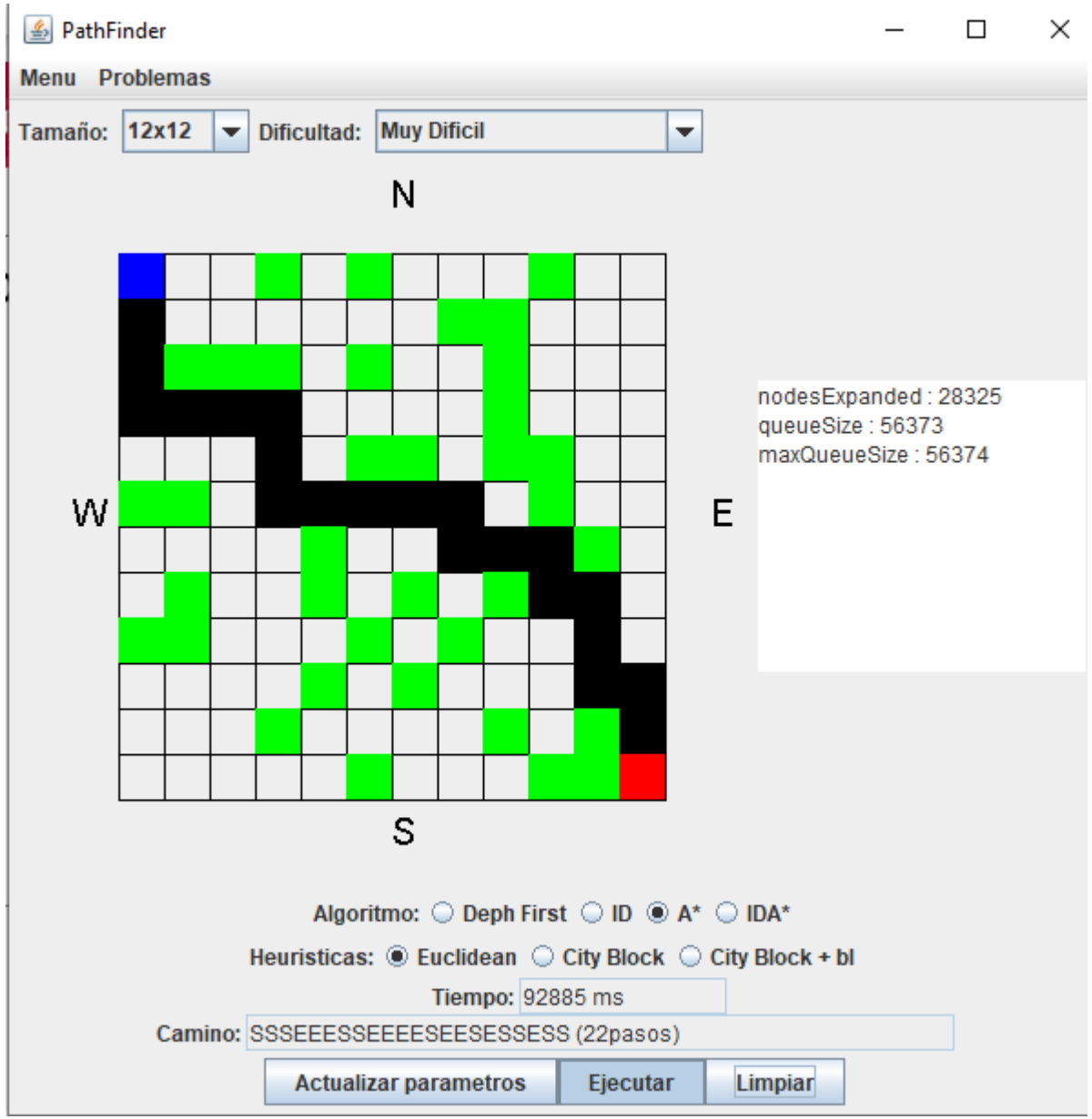
Ejercicios tipo laberintos

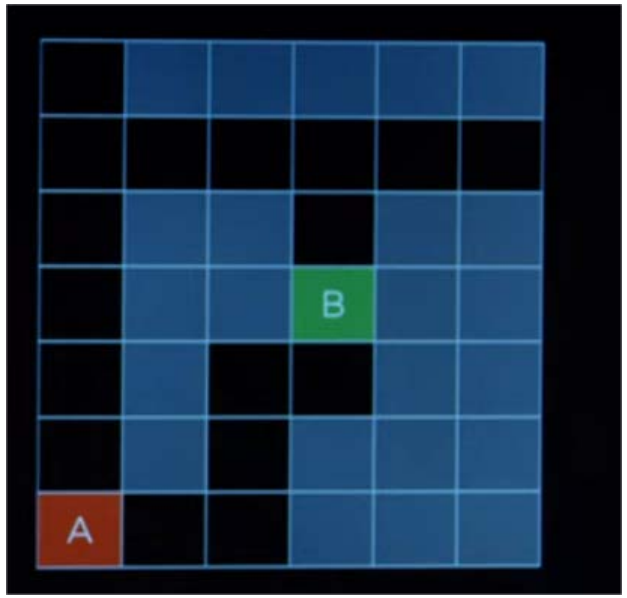
- Ejemplos pathfinding
  - En este problema los estados son simples (x,y)
  - Las acciones son movimientos: arriba, abajo, izquierda, derecha



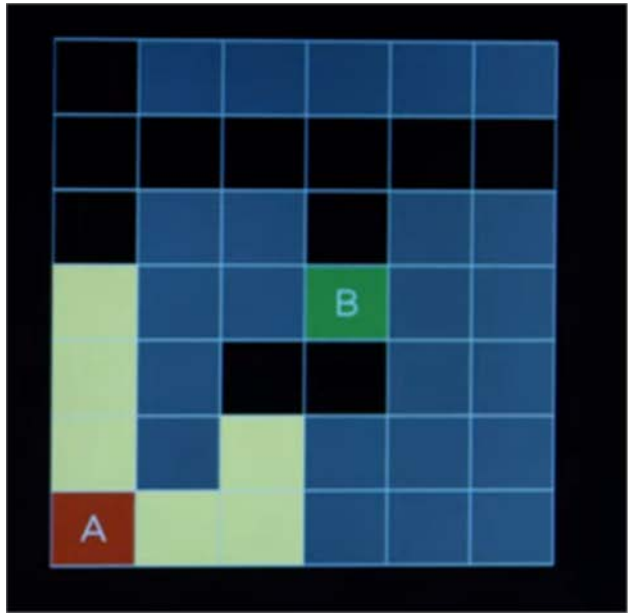


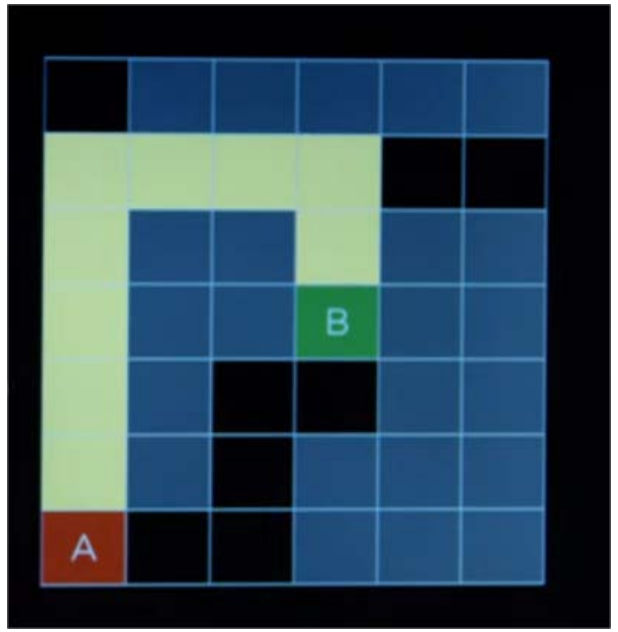


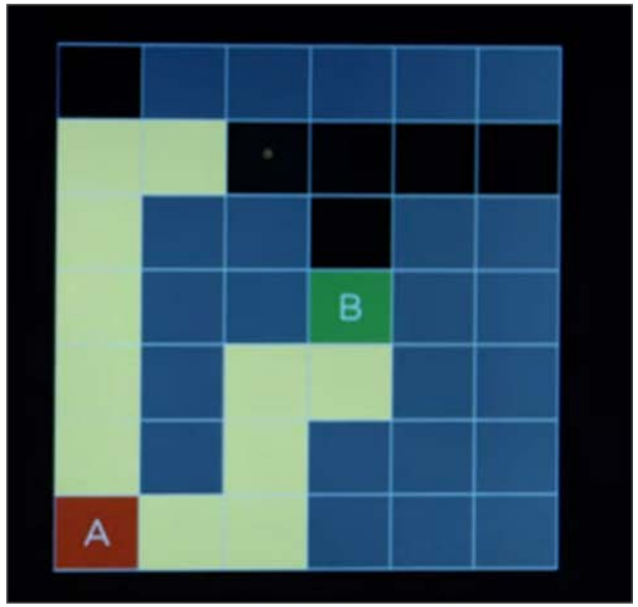

PathFinder
Menu
Problemas
Tamaño:
12x12
Dificultad:
Muy Dificil
N
W
E
S
nodesExpanded : 28325
queueSize : 56373
maxQueueSize : 56374
Algoritmo:
Deph First
ID
A*
IDA*
Heuristicas:
Euclidean
City Block
City Block + bl
Tiempo:
92885 ms
Camino:
SSSEEESSEEEESEESESSESS (22pasos)
Actualizar parametros
Ejecutar
Limpiar



Breadth-First Search







¿Dónde se toman las decisiones?
¿Cual sería la intuición humana?

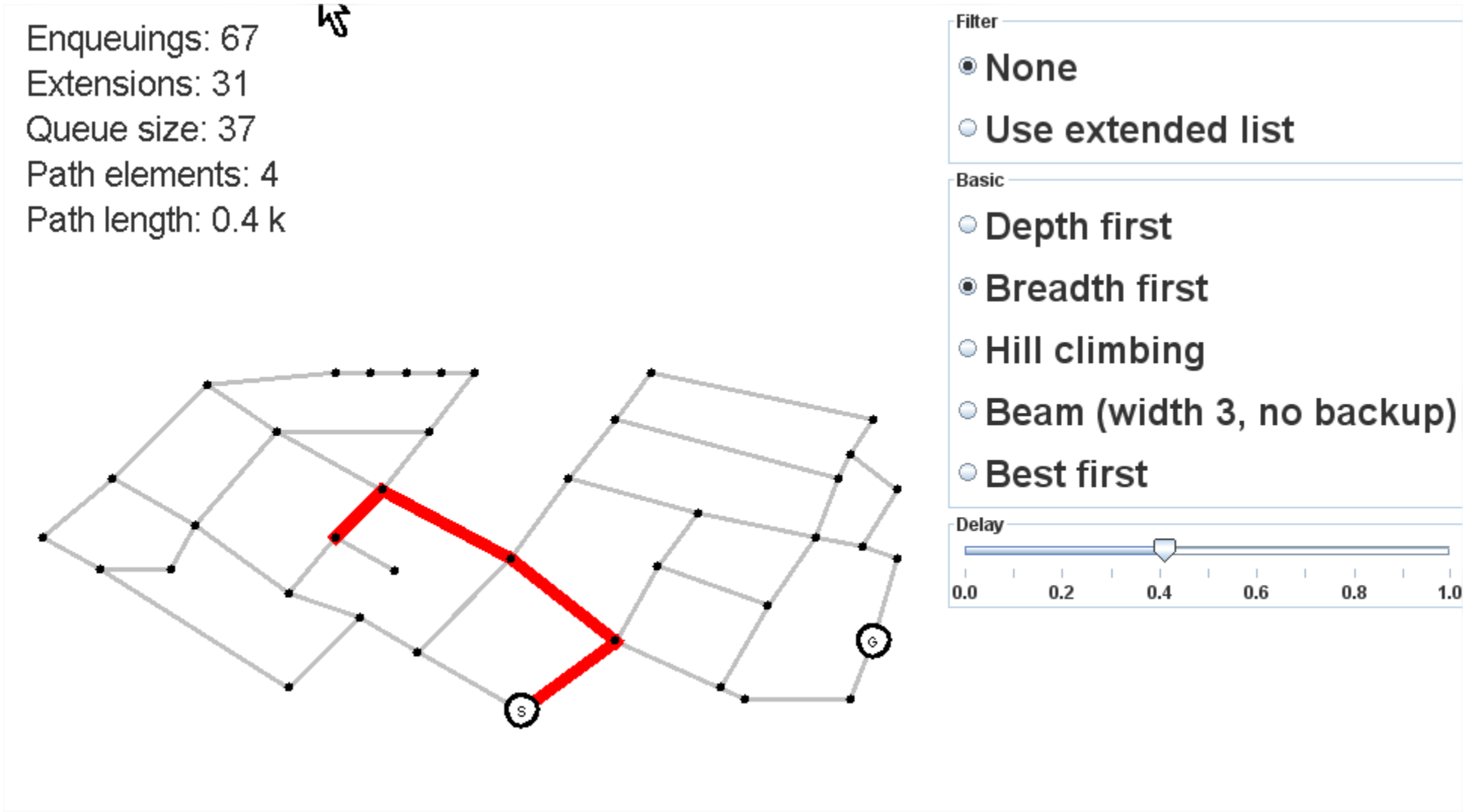
- Búsquedas ciegas (no informadas)

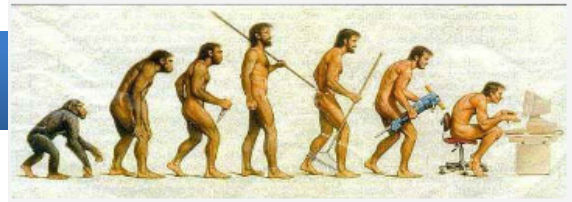
Vs

- Búsquedas heurísticas (informadas)

Usamos conocimiento específico del problema **para guiar la búsqueda**
Aparte del conocimiento de la representación del problema

¿Los humanos resolvemos problemas con búsqueda?





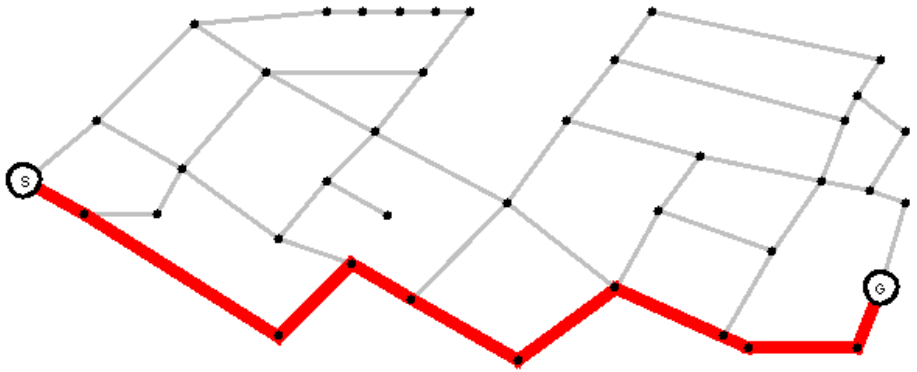
Basic search

10:44:18 CEST 25-sep-2018

- Goal trees
  - Blocks
- Search
  - Basic search
  - Optimal search
  - Games
  - Monte Carlo
- Constraint satisfaction
- Biological mimetics
- Learning
- Neural nets
- Bayes nets
- Miscellaneous

Start | Stop | Pause | Set start | Set goal | Thief | Beginner | Expert | Hide choices

Enqueuings: 661
Extensions: 386
Queue size: 277
Path elements: 10
Path length: 0.8 k



Filter
- None
- Use extended list

Basic
- Depth first
- Breadth first
- Hill climbing
- Beam (width 3, no backup)
- Best first

Delay
0.0 0.2 0.4 0.6 0.8 1.0

La búsqueda se refiere a las elecciones a las decisiones que hay que tomar para encontrar una solución.

DEMO PACMAN Y ALGORITMOS DE BUSQUEDA

- _pycache_
- layouts
- test_cases
- autograder
- commands
- eightpuzzle
- game
- ghostAgents
- grading
- graphicsDisplay
- graphicsUtils
- keyboardAgents
- layout
- pacman
- pacmanAgents
- projectParams
- search
- searchAgents
- searchTestClasses
- submission_autograder
- testClasses
- testParser
- textDisplay
- util
- VERSION

Descargar y descomprimir el archivo search.zip dado

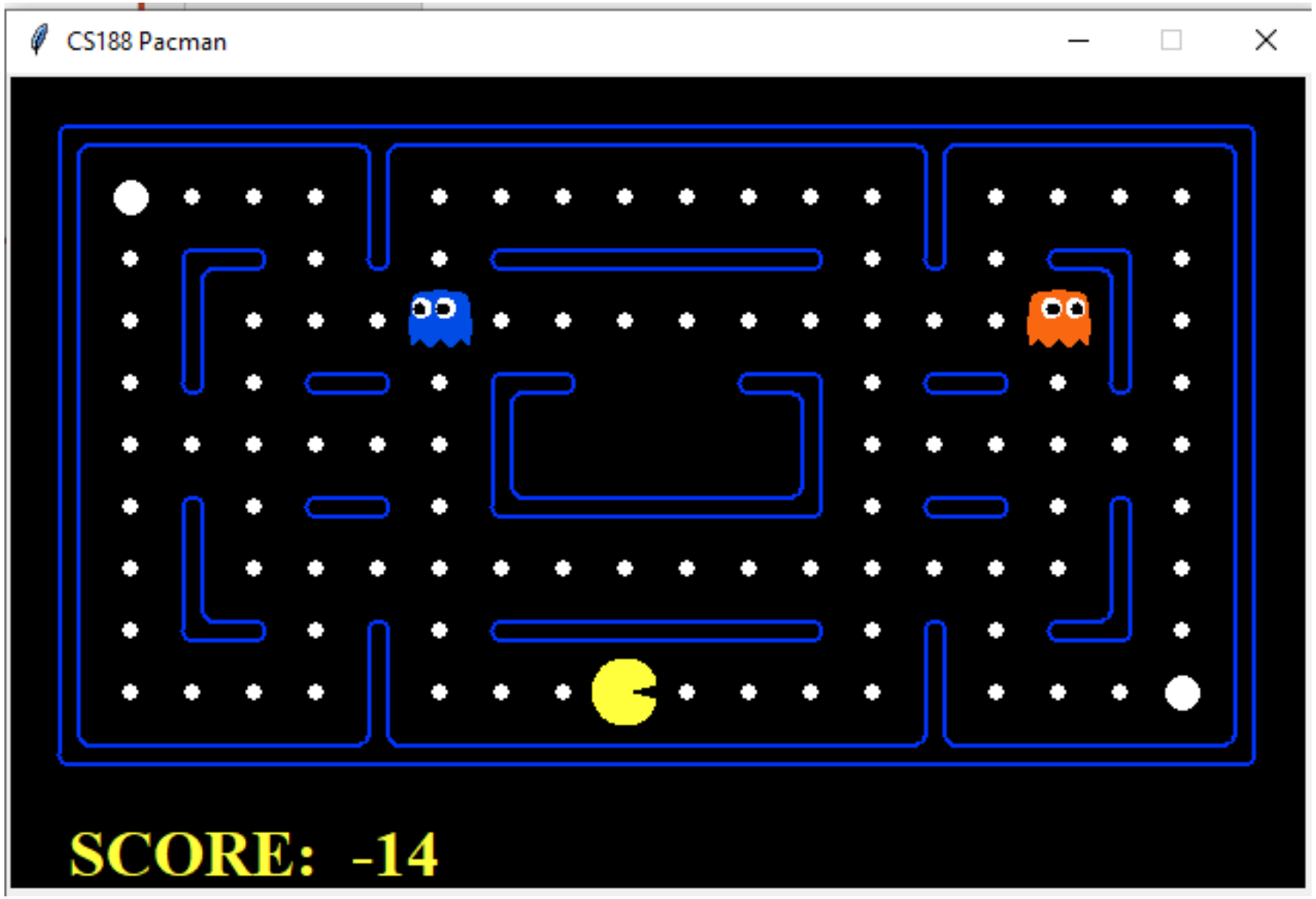
El entorno se ejecuta con

python pacman.py

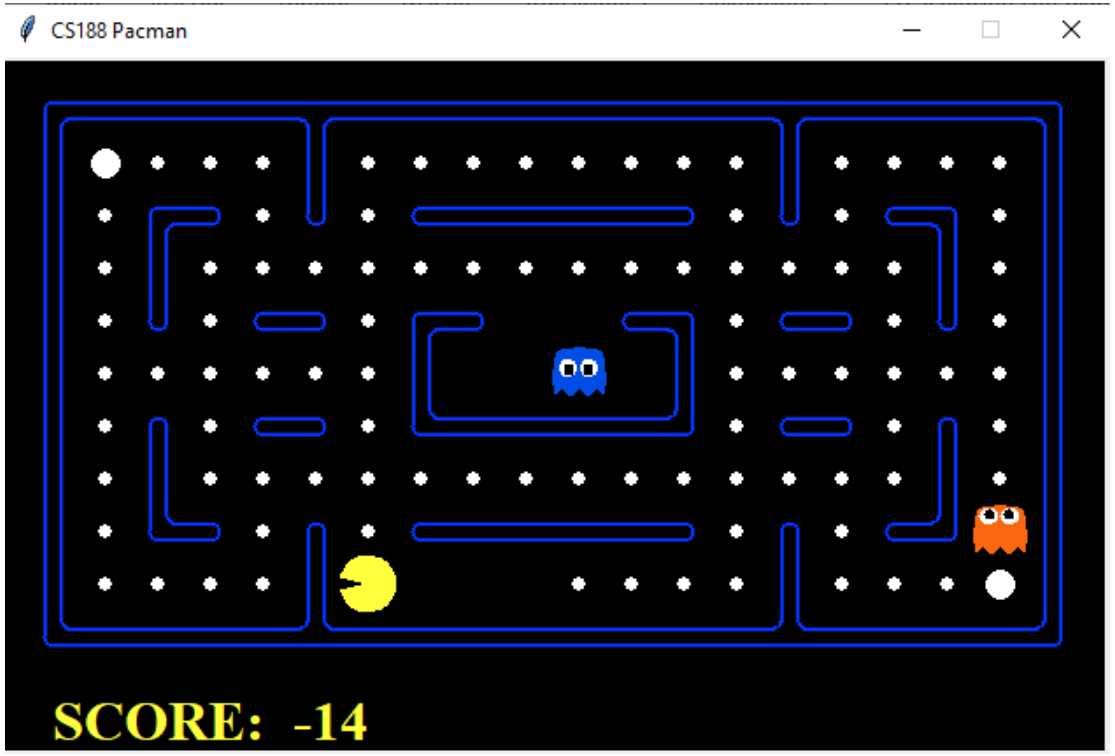
No estamos indicando el comportamiento del pacman → por defecto PACMAN manual (teclado)

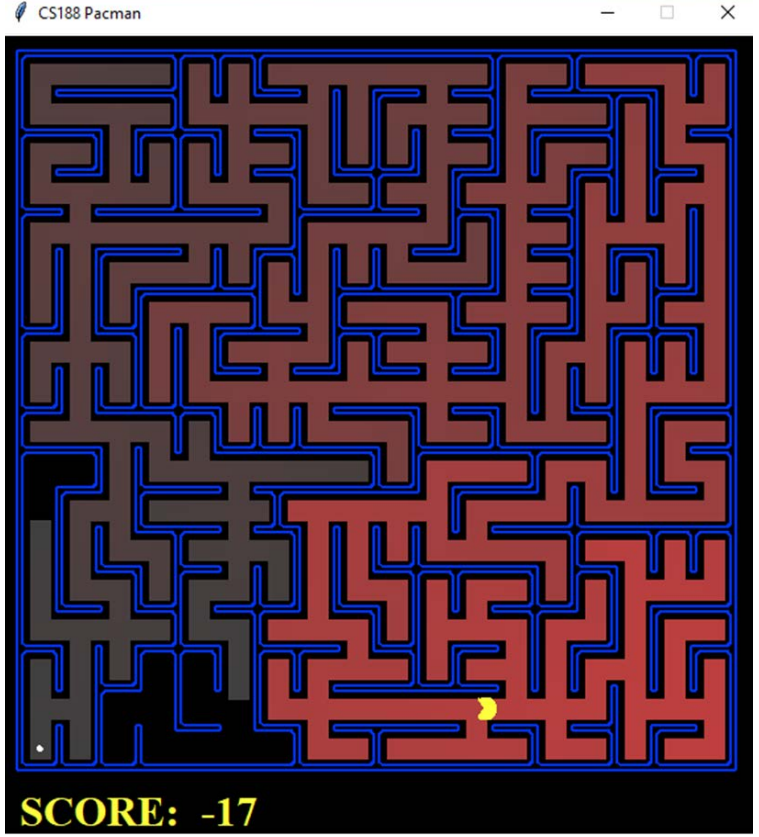
Se cambia con el parámetro –p

Los fantasmas tienen por defecto un comportamiento random



python pacman.py --pacman GoWestAgent

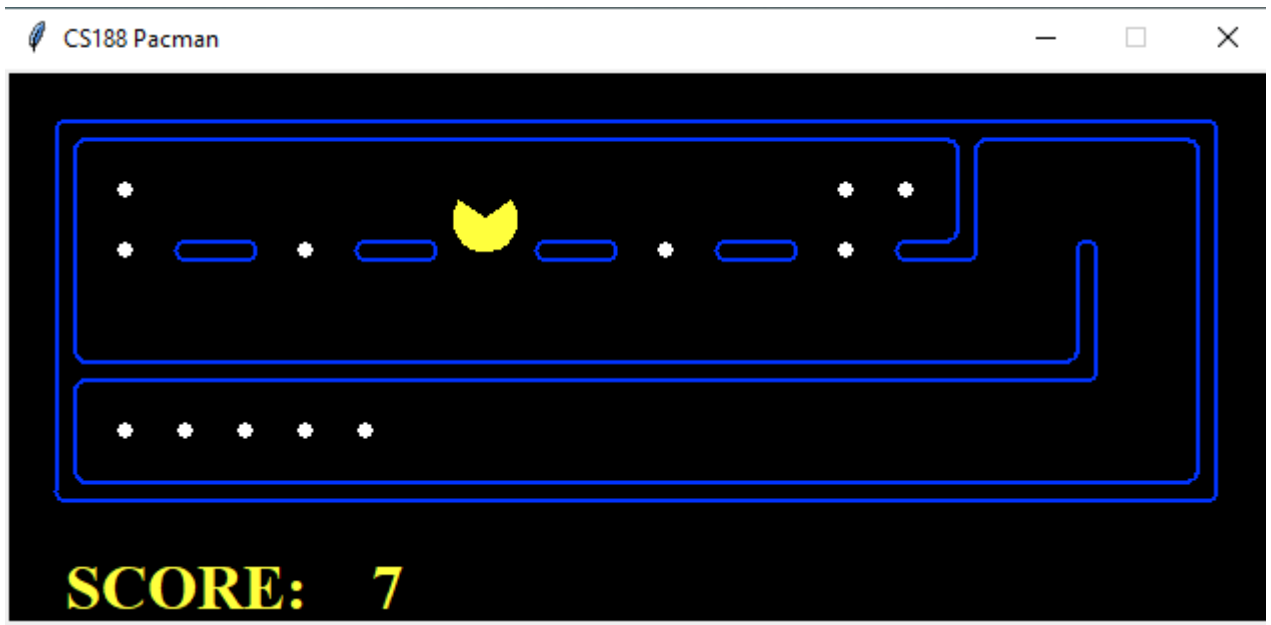





```
C:\Datos\Clases\IA\IAIC\2122\ejercicios resueltos AIMA\search>python pacman.py --layout testMaze --pacman GoWestAgent
Pacman emerges victorious! Score: 503
Average Score: 503.0
Scores:        503.0
Win Rate:      1/1 (1.00)
Record:        Win

C:\Datos\Clases\IA\IAIC\2122\ejercicios resueltos AIMA\search>python pacman.py --pacman GoWestAgent
Pacman died! Score: -541
Average Score: -541.0
Scores:        -541.0
Win Rate:      0/1 (0.00)
Record:        Loss
```

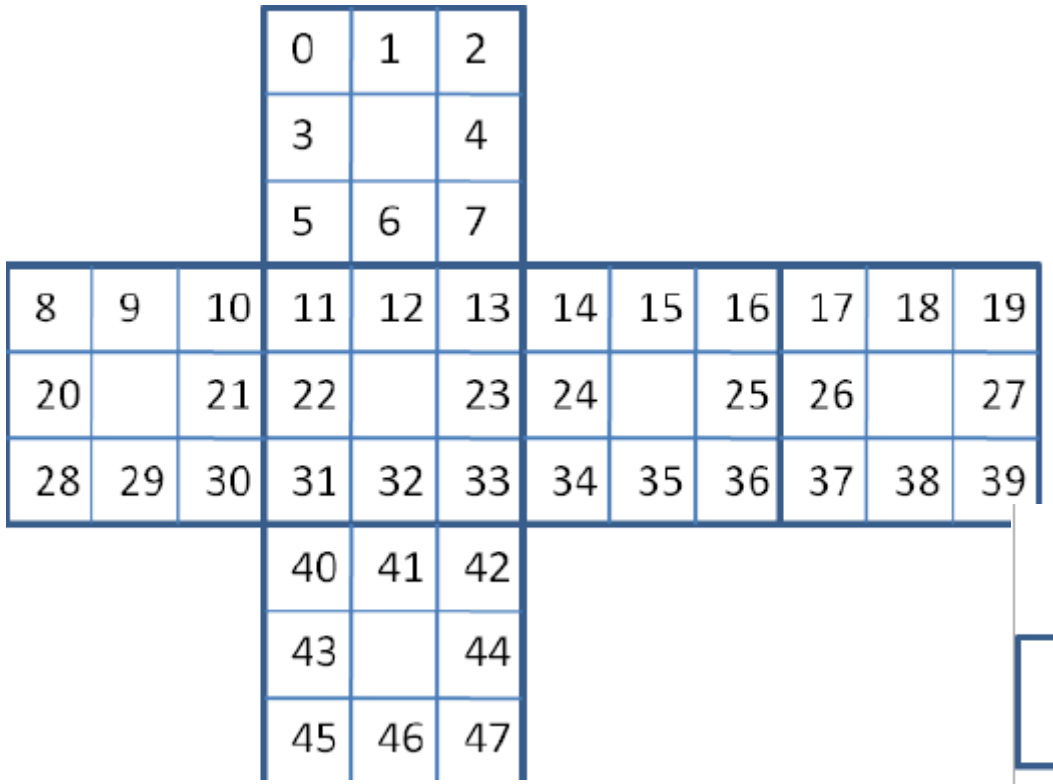
Siempre va al oeste y se queda bloqueado. Lo normal es que pierda..

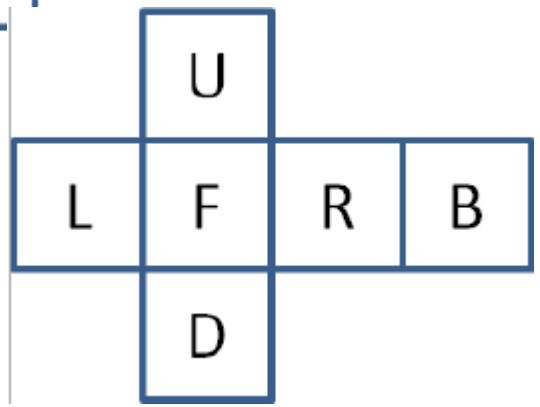
```
C:\Datos\Clases\IA\IAIC\2122[redacted]\search>python pacman.py -l trickySearch -p AStarFoodSearchAgent
The cost is  1
Path found with total cost of 60 in 1.7 seconds
Search nodes expanded: 16688
Pacman emerges victorious! Score: 570
Average Score: 570.0
Scores:        570.0
Win Rate:      1/1 (1.00)
Record:        Win
```

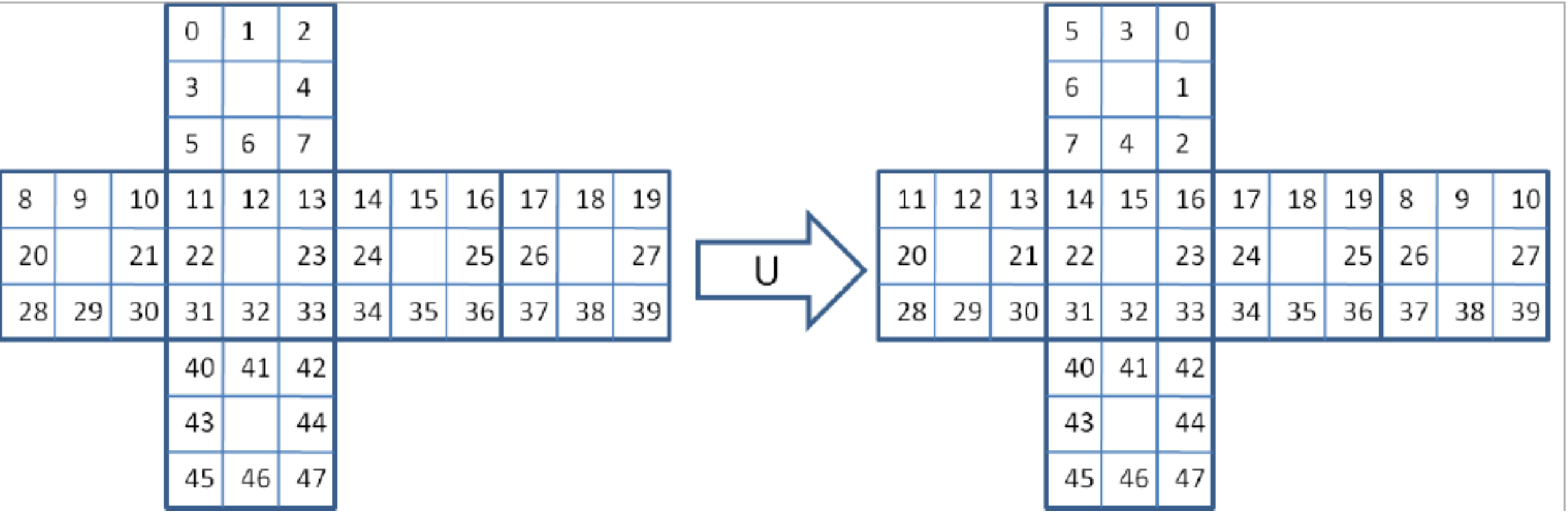



0
1
2
3
4
0
1
2
3
4
R
G
Y
B

Cubo de Rubik

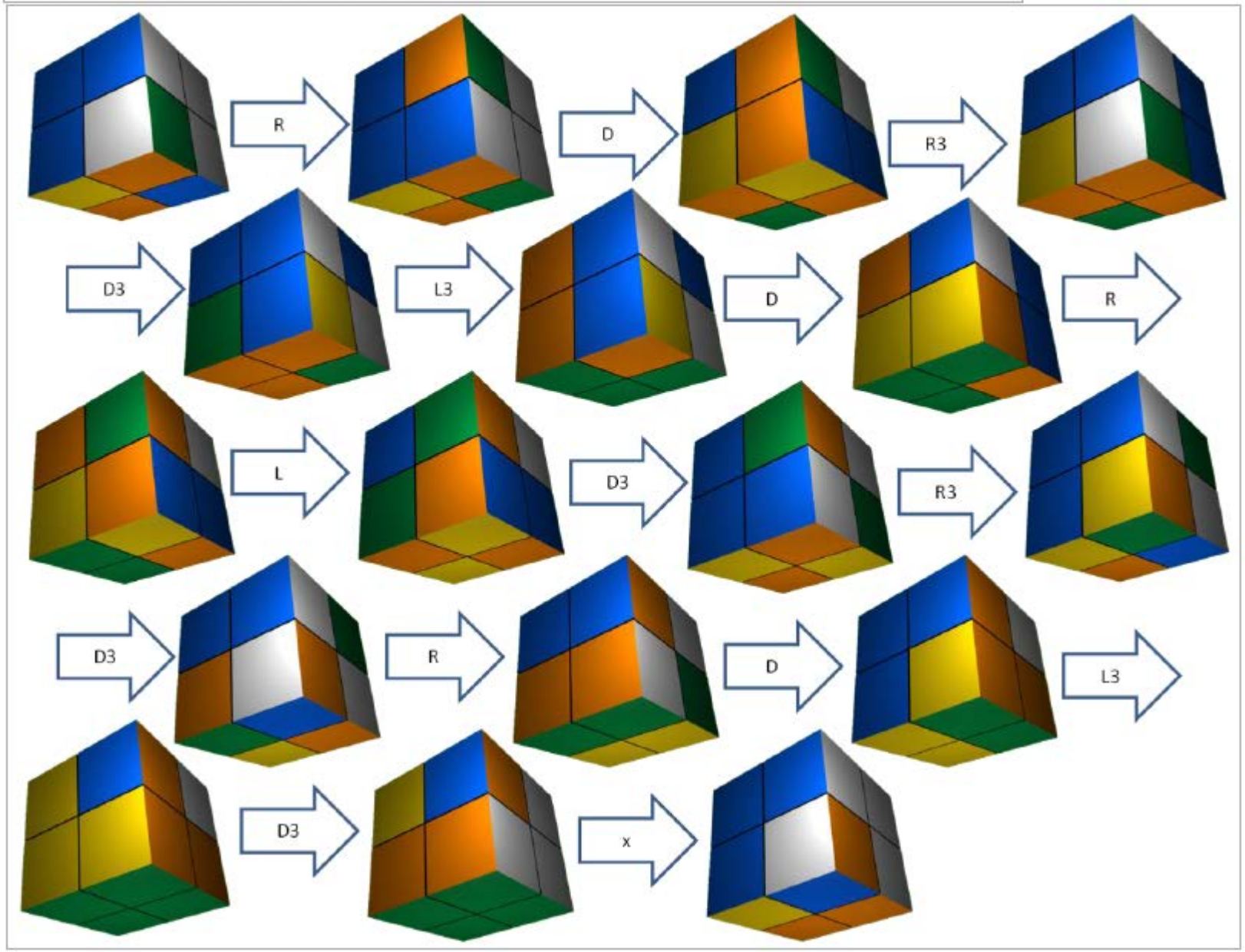






Movimiento de la cara superior ‘U’, que se gira 90º en el sentido de las agujas del reloj
‘U2’ sería una giro de 180º, y ‘U3’ un giro de 270º (o de 90º en sentido contrario a las agujas del reloj).

Como el cubo de 2x2x2 sólo tiene esquinas lo que conseguimos es cambiar la posición de dos esquinas adyacentes. Los movimientos son los siguientes:
RDR3D3L3DRLD3R3D3RDL3D3x



Bibliografía

- **Russell, S., Norvig, P.** Artificial Intelligence. A Modern Approach. Prentice Hall, 2013, 3ª edición. https://faculty.psau.edu.sa/filedownload/doc-7-pdf-a154ffbcec538a4161a406abf62f5b76-original.pdf