

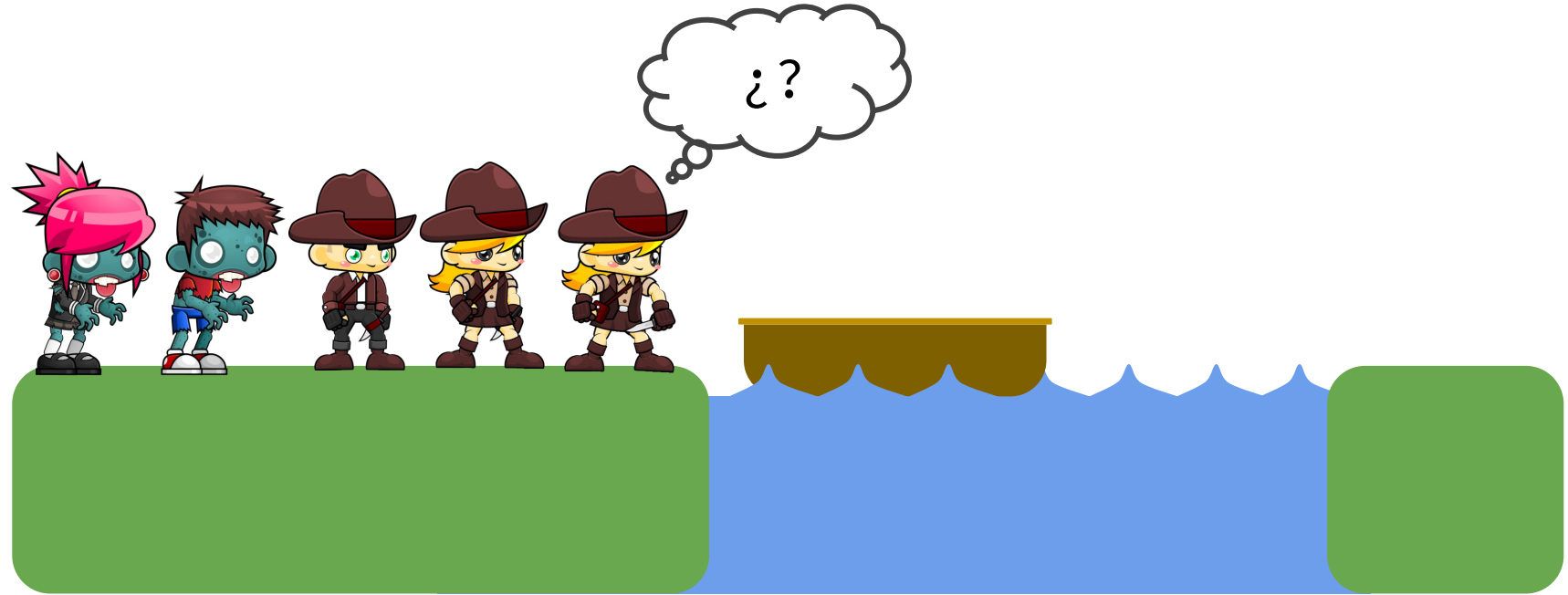
1. RESOLUCIÓN DE PROBLEMAS MEDIANTE BÚSQUEDA (PARTE 1)

IA 3.2 - Programación III

2° C - 2025

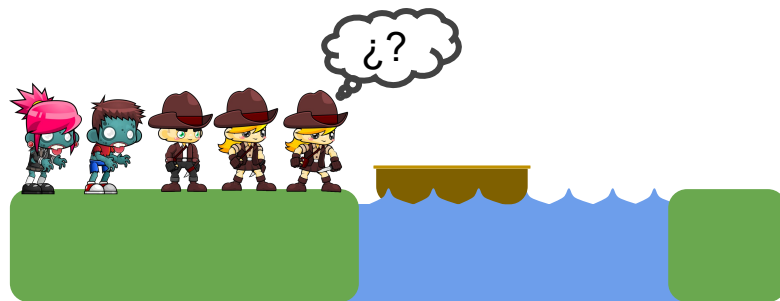
Dr. Mauro Lucci

Problema de cruce de río



Descripción

— — —



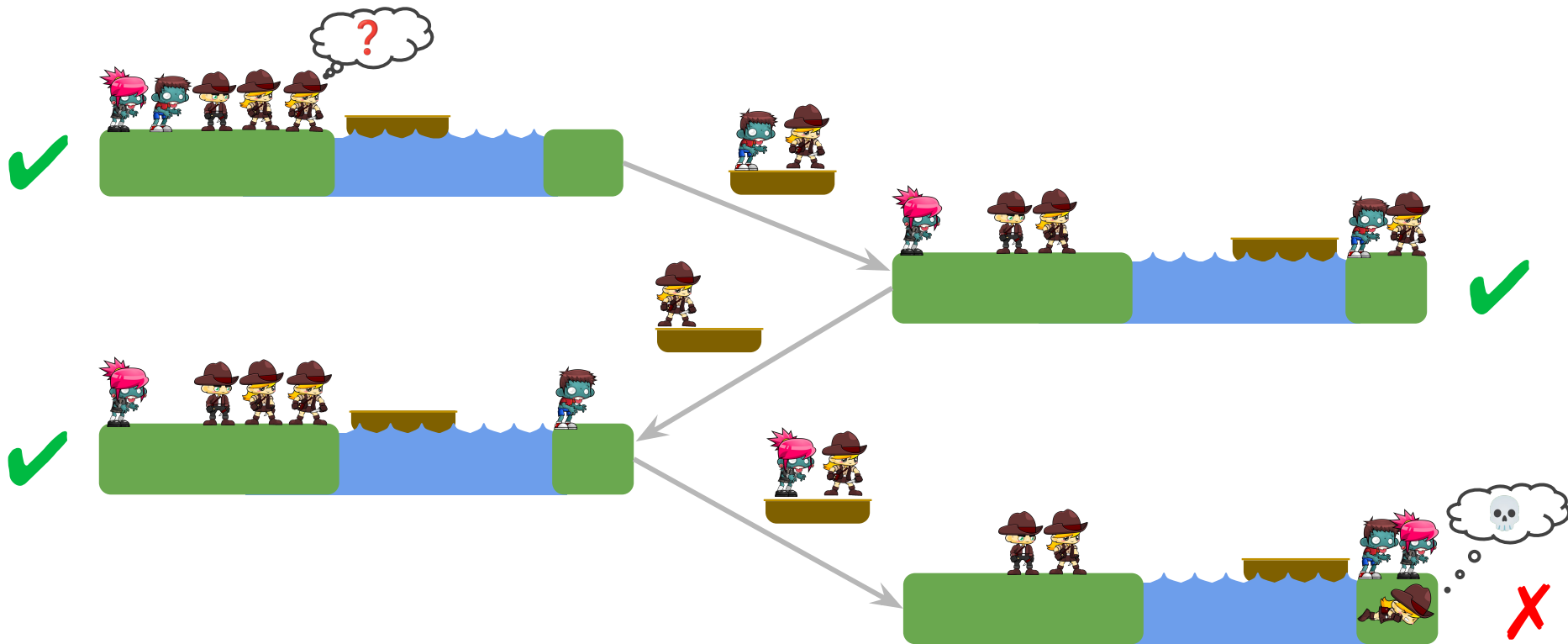
Objetivo. Todas las personas (humanas y zombies) deben llegar al otro lado del río sanas y salvas, en el menor número de cruces.

Reglas.

1. El río solo se cruza en bote.
2. La capacidad del bote es de 1 o 2 personas (humanas o zombies).
3. Al menos 1 persona (humana o zombie) debe estar en el bote para navegar.
4. Si en algún lado del río la cantidad de zombies supera a la de humanos (incluso por un momento), los zombies comerán a los humanos.



Ejemplo





Desafío

— — —

¿Es posible resolver el problema?

En caso afirmativo, ¿cuántos cruces requiere la solución que encontró?

Problema de búsqueda

Un **problema de búsqueda** se define por:

1. **Estado inicial.**
2. **Acciones.**
3. **Modelo de transiciones.**
4. **Test objetivo.**
5. **Costo de camino.**

Muchos problemas de **factibilidad** y de **optimización** se pueden expresar como problemas de búsqueda.

— — —

Formular y Abstraer

— — —

- **Formular.** Proceso de expresar a un problema como un problema de búsqueda, debiendo elegir una representación para los **estados** y las **acciones** que lo describen.
- **Abstraer.** Proceso de remover de una representación detalles **irrelevantes**.

Estados y Acciones

— — —

- **Estados.** Un **estado** es una representación abstracta de los elementos del problema en un instante dado.

Deben incluirse aquellos elementos que cambian en el tiempo y que no pueden determinarse a partir de otros elementos.

- **Acciones.** Una **acción** modifica el estado actual.








Ejemplo – Cruce de río

Podemos representar un estado como una **matriz 3x2** con las cantidades de zombies, humanos y bote en cada lado del río.

Cantidad en
lado izq.

Cantidad en
lado der.

		→	Zombies
		→	Humanos
		→	Bote







Sin emojis:

$$\begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 0 & 1 \end{bmatrix}$$

¿Hay información redundante en esta representación? 🤔

Información redundante

— — —

		→ La fila suma 2
 		→ La fila suma 3
		→ La fila suma 1

Conociendo las cantidades del lado izquierdo, es posible determinar las del lado derecho, y viceversa.

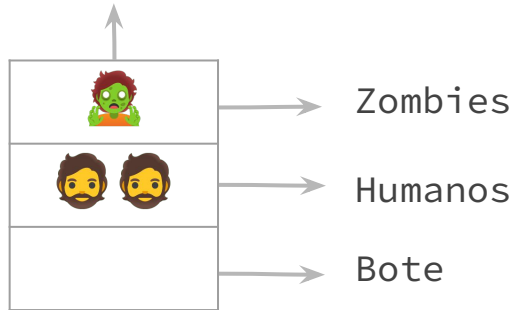


Ejemplo – Cruce de río

— — —

Luego, podemos representar un estado como una **terna** con las cantidades de zombies, humanos y bote del lado izquierdo.

Cantidad en
lado izq.



Sin emojis:

$(1, 2, 0)$

Luego en el lado
derecho hay:





Ejemplo – Cruce de río

— — —

Estados. Por extensión:

$\{(0,0,0), (0,0,1), (0,1,0), (0,1,1), (0,2,0), (0,2,1),$
 $(0,3,0), (0,3,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1),$
 $(1,2,0), (1,2,1), (1,3,0), (1,3,1), (2,0,0), (2,0,1),$
 $(2,1,0), (2,1,1), (2,2,0), (2,2,1), (2,3,0), (2,3,1)\}$

Por comprensión:

$$\{(z, h, b) \in \{0, 1, 2\} \times \{0, 1, 2, 3\} \times \{0, 1\}\}$$



Ejemplo – Cruce de río

— — —

Estado genérico:

(z, h, b)



Cantidades en el lado
derecho:

- Zombies: 2 - z
- Humanos: 3 - h
- Bote: 1 - b



Ejemplo – Cruce de río

— — —

Cantidad total de estados:

$$|\{0,1,2\} \times \{0,1,2,3\} \times \{0,1\}| =$$

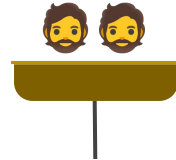
$$|\{0,1,2\}| \times |\{0,1,2,3\}| \times |\{0,1\}| =$$

$$3 \times 4 \times 2 = 24$$



Ejemplo – Cruce de río

Una **acción** consiste en cruzar al menos 1 y a lo sumo 2 personas:



Podemos representar una acción con un **par** con las cantidades de zombies y humanos que cruzan.

↓
 $(1,0)$

↓
 $(2,0)$

↓
 $(0,1)$

↓
 $(0,2)$

↓
 $(1,1)$



Ejemplo – Cruce de río

Acciones. Por extensión:

$$\{(1,0), (2,0), (0,1), (0,2), (1,1)\}$$

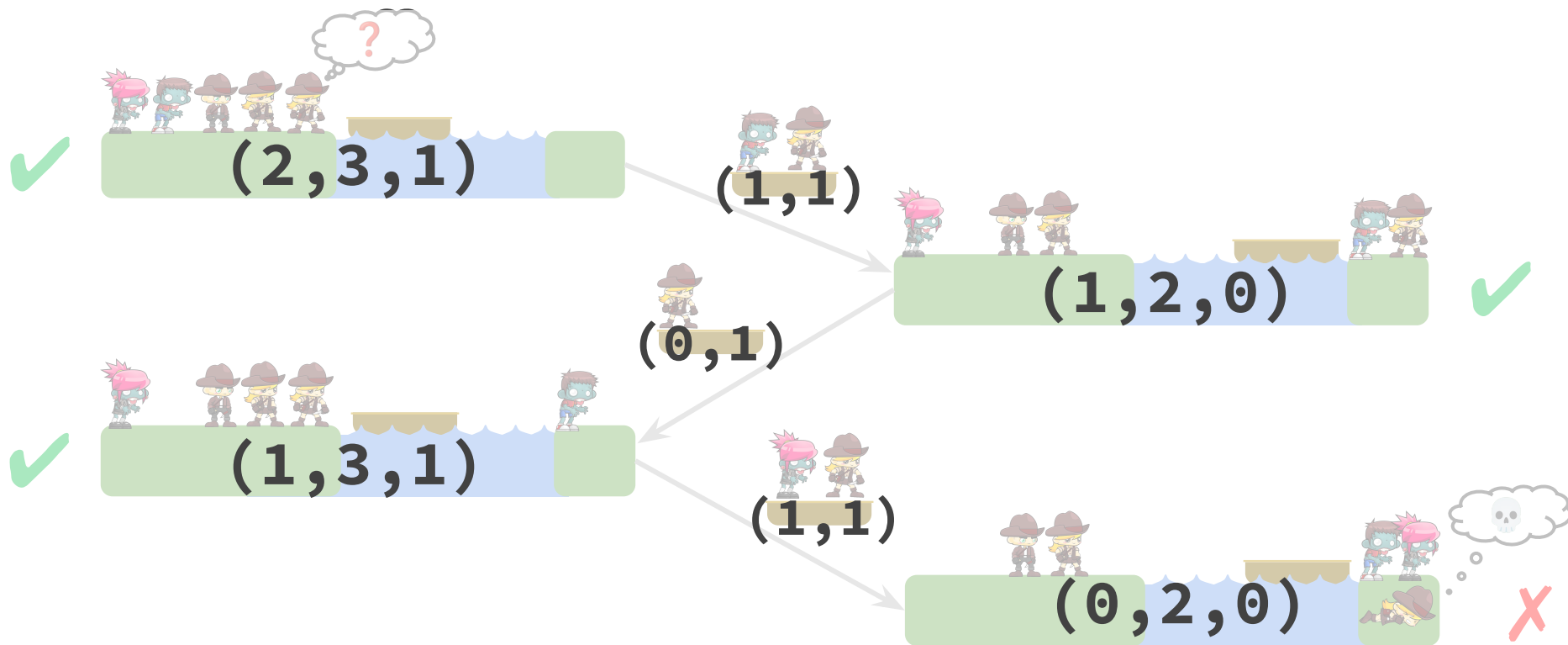
Por comprensión:

$$\{(z', h') \in \{0, 1, 2\} \times \{0, 1, 2, 3\} : \underbrace{1 \leq z' + h' \leq 2}_{\text{Al menos 1 y a lo sumo 2 personas.}}\}$$

Al menos 1 y a lo
sumo 2 personas.



Ejemplo – Cruce de río



Problema de búsqueda

— — —

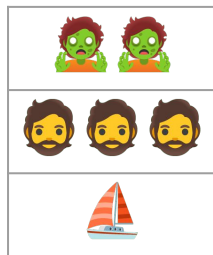
Una vez elegida una representación para los **estados** y **acciones**, hay que definir las 5 componentes de un problema de búsqueda.

1. Estado inicial

— — —

Estado inicial. Estado del problema en el instante inicial.

📖 **Ejemplo - Cruce de río.**



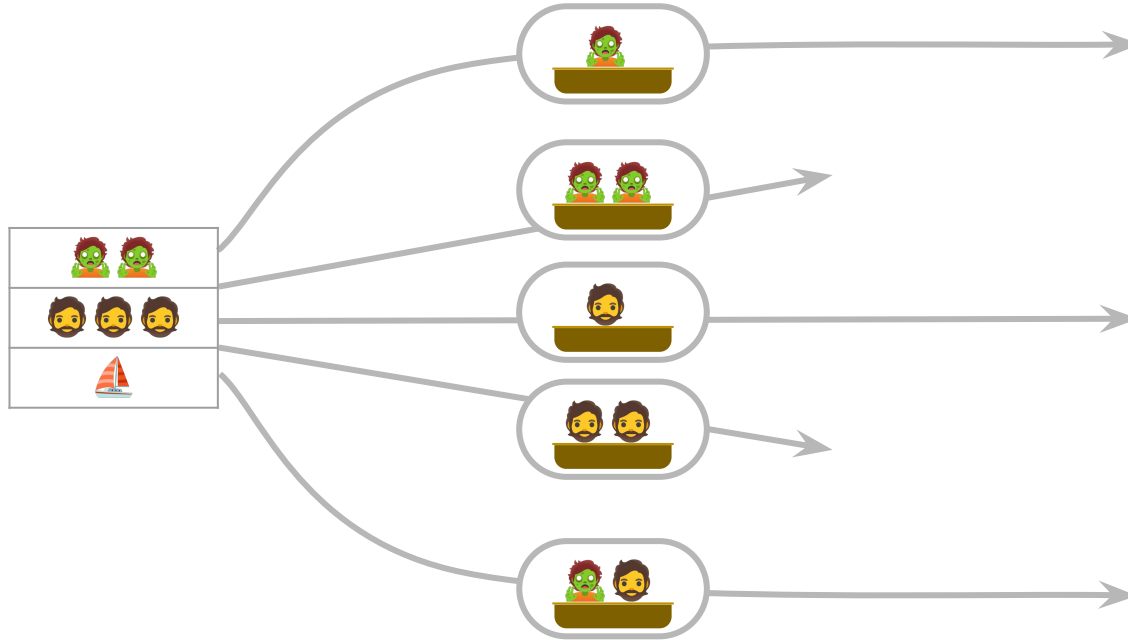
(2,3,1)

2. Acciones

Acciones. Dado un estado S , la función **ACCIONES(S)** retorna el conjunto de acciones que se pueden ejecutar en S .



Ejemplo – Cruce de río

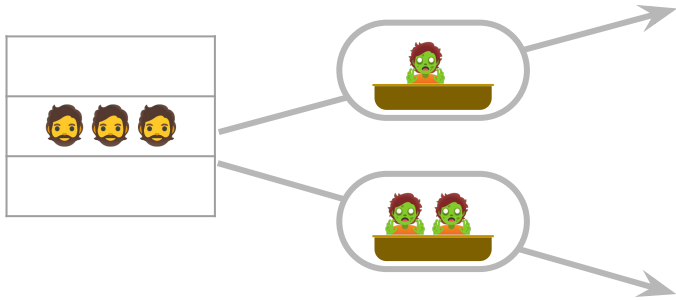


$$\text{ACCIONES}(2,3,1) = \{(1,0), (2,0), (0,1), (0,2), (1,1)\}$$



Ejemplo – Cruce de río

— — —



$\text{ACCIONES}(0,3,0) = \{(1,0), (2,0)\}$

$\text{ACCIONES}(2,1,0) = \{\}$



Ejemplo – Cruce de río

Podemos definir la función **ACCIONES** por extensión:

$ACCIONES(2,3,1) = \{(1,0), (2,0), (0,1), (0,2), (1,1)\}$

$ACCIONES(0,3,0) = \{(1,0), (2,0)\}$

$ACCIONES(2,1,0) = \{\}$

... ¡Así con cada uno de los 24 estados!

Pero en general es conveniente dar su definición por comprensión, es decir, mediante una **ley** lógica/matemática.



Ejemplo – Cruce de río

— — —



Idea.

- Si en cualquiera de los lados del río hay más zombies que humanos, no podemos aplicar acciones.
- Sino, podemos aplicar toda acción (z', h') tal que z' y h' sean menores o iguales a la cantidad de zombies y humanos que hay respectivamente en ese lado del río.



Ejemplo – Cruce de río

Para un estado genérico (z, h, b) donde el bote está en el lado **izquierdo**, es decir $b = 1$.

$\text{ACCIONES}(z, h, 1) =$

$$\left\{ \begin{array}{l} \{\} \\ \{(z', h') \in \{0, \dots, z\} \times \{0, \dots, h\} : 1 \leq z' + h' \leq 2\} \end{array} \right.$$

Pueden cruzar a
lo sumo z zombies
y h humanos

No se excede
la capacidad
del bote

Hay más zombies que humanos
en el lado izquierdo

si $\overbrace{z > h}$

si $z \leq h$



Ejemplo – Cruce de río

Para un estado genérico (z, h, b) donde el bote está en el lado **derecho**, es decir $b = 0$.

$\text{ACCIONES}(z, h, 0) =$

$$\left\{ \begin{array}{l} \{ \} \\ \{(z', h') \in \underbrace{\{0, \dots, 2 - z\} \times \{0, \dots, 3 - h\}}_{\text{Pueden cruzar a lo sumo } 2-z \text{ zombies y } 3-h \text{ humanos}} : \underbrace{1 \leq z' + h' \leq 2}_{\text{No se excede la capacidad del bote}} \} \end{array} \right.$$

Hay más zombies que humanos
en el lado derecho

si $\overbrace{2 - z > 3 - h}$

si $2 - z \leq 3 - h$

3. Modelo de transiciones

Modelo de transiciones. Describe el resultado de aplicar las acciones a los estados.

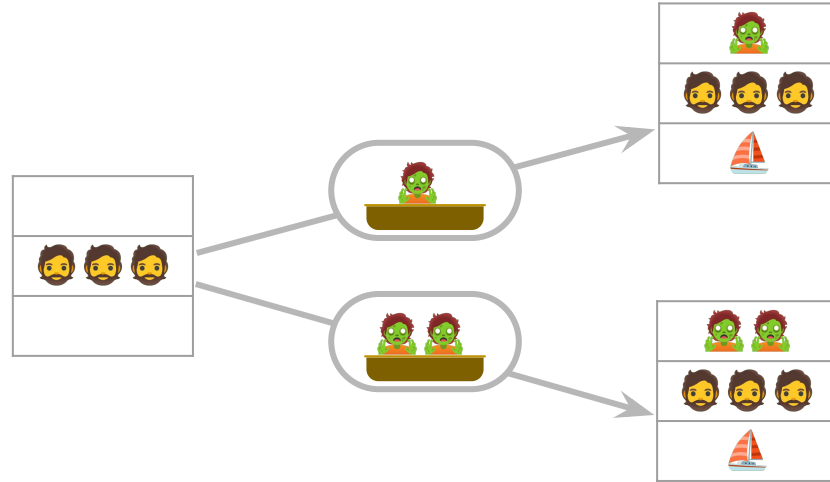
Dada un estado S y una acción $A \in \text{ACCIONES}(S)$, la función **RESULTADO(S, A)** retorna el estado que resulta de aplicar A en S .

Si $\text{RESULTADO}(S, A) = S'$, entonces S' es un **sucesor** de S .



Ejemplo – Cruce de río

— — —



RESULTADO((0,3,0), (1,0)) = (1,3,1)

RESULTADO((0,3,0), (2,0)) = (2,3,1)



Ejemplo – Cruce de río

— — —

Podemos definir la función **RESULTADO** por extensión:

`RESULTADO((0,3,0), (1,0)) = (1,3,1)`

`RESULTADO((0,3,0), (2,0)) = (2,3,1)`

... ¡A lo sumo $24 \times 5 = 120$ casos!

Para evitar enumerarlos a todos, la definimos por comprensión.

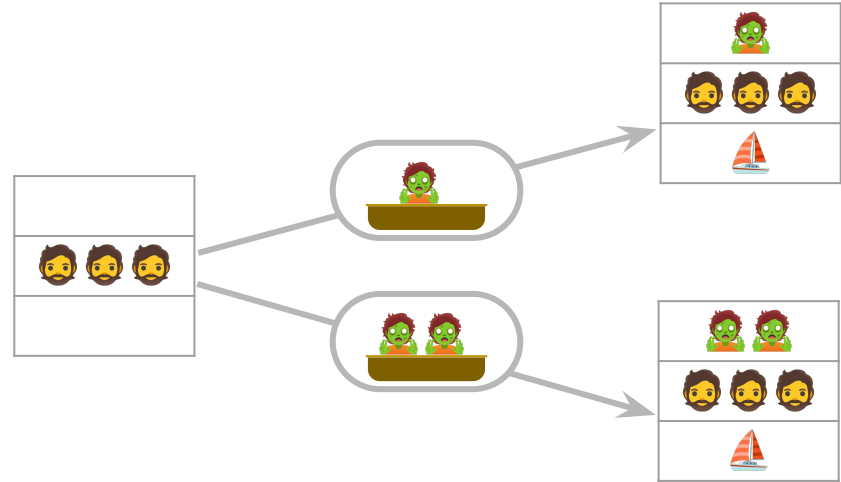


Ejemplo – Cruce de río



Idea.

- Si el bote va a la derecha, las personas se van (**restan**).
- Si el bote va a la izquierda, las personas llegan (**suman**).
- El bote siempre cambia de lado.



$$\text{RESULTADO}((0, 3, 0), (1, 0)) = (1, 3, 1) =$$

$$(0 + 1, 3 + 0, 1)$$

$$\text{RESULTADO}((0, 3, 0), (2, 0)) = (2, 3, 1) =$$

$$(0 + 2, 3 + 0, 1)$$



Ejemplo – Cruce de río

— — —

$$\text{RESULTADO}((z, h, b), (z', h')) = \begin{cases} (z - z', h - h', 0) & \text{si } b = 1 \\ (z + z', h + h', 1) & \text{si } b = 0 \end{cases}$$

Espacio de estados

— — —

- El estado inicial, las acciones y el modelo de transiciones definen implícitamente el **espacio de estados** del problema.
- Es una abstracción que nos permite razonar sobre los estados que podemos alcanzar mediante cualquier secuencia de acciones.
- Cuando es finito, se puede representar con un **grafo dirigido** (o **digrafo**) donde los **nodos** son estados y los **arcos** son acciones.
- El **tamaño** del espacio de estados es el total de estados/nodos.



Digrafo de estados - Construcción

— — —



Ejemplo.

INICIAL = A

ACCIONES(A) = $\{a_1, a_2\}$

ACCIONES(B) = $\{b\}$

ACCIONES(C) = $\{c\}$

ACCIONES(D) = $\{d\}$

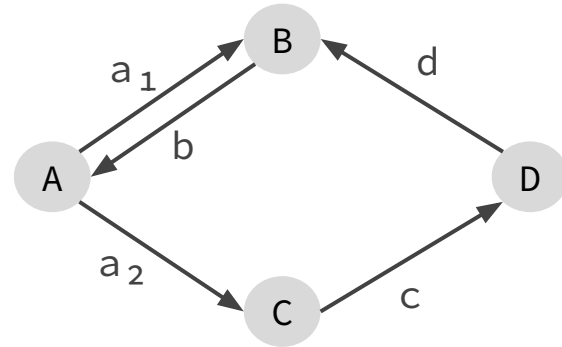
RESULTADO(A, a_1) = B

RESULTADO(A, a_2) = C

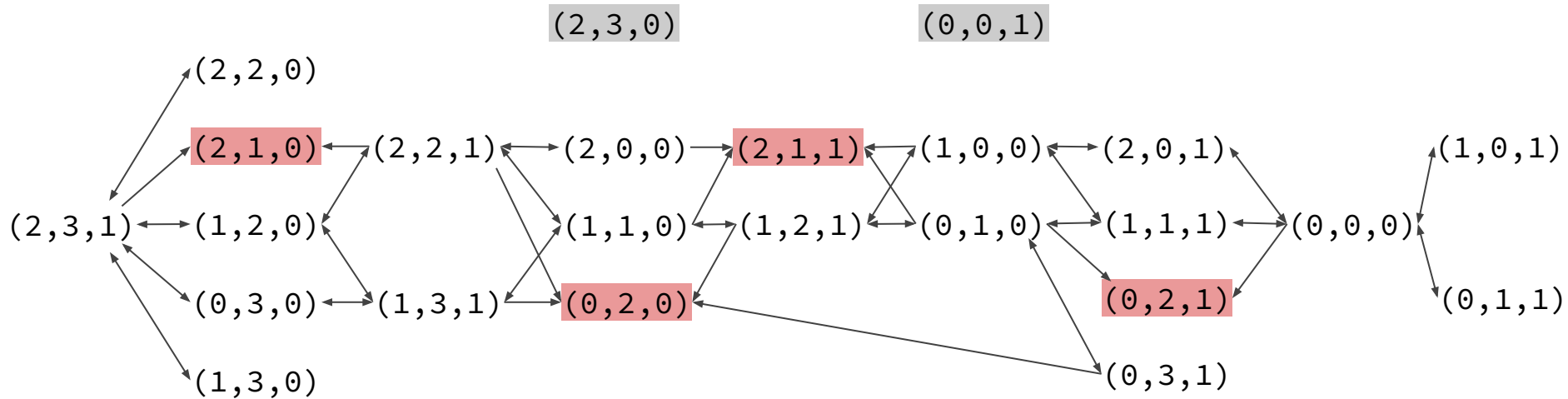
RESULTADO(B, b) = A

RESULTADO(C, c) = D

RESULTADO(D, d) = B







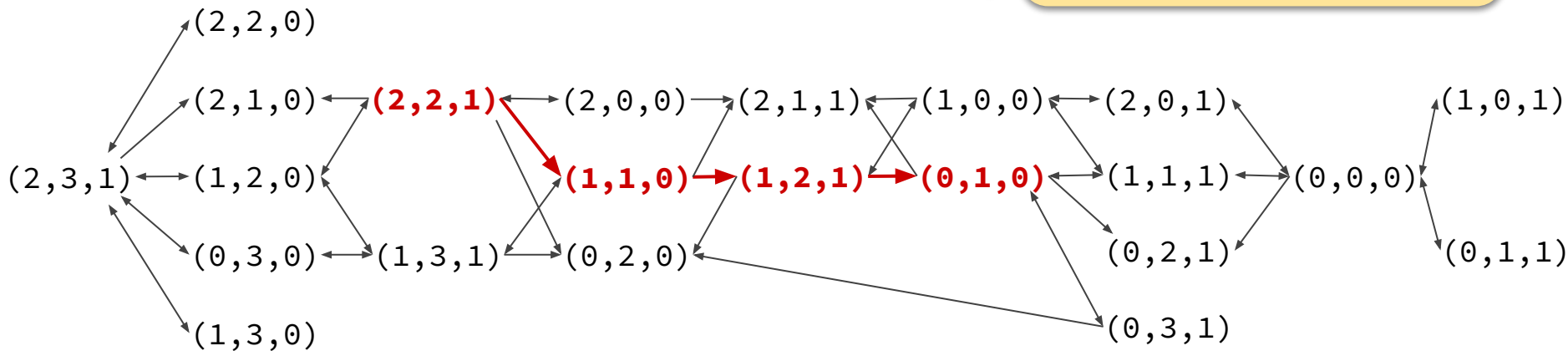
- Sin sucesores: Un estado s para el cual $\text{ACCIONES}(s) = \emptyset$.
- No alcanzable: Un estado s para el cual $\text{RESULTADO}^{-1}(s) = \emptyset$.



Ejemplo – Cruce de río

— — —

Un **camino** en el espacio de estados es una secuencia de estados conectados mediante una secuencia de acciones.



Estado objetivo

— — —

Objetivo. Es un estado al que se busca llegar.

Pueden existir múltiples estados objetivos.



Ejemplo - Cruce de río.



$(0,0,0)$

En este caso,
hay único
objetivo.

4. Test objetivo

Un **predicado** es una función que retorna **True** o **False**.

Dado un estado S , el predicado **TEST-OBJETIVO(S)** determina si S es un estado objetivo.

Puede darse mediante una **enumeración explícita** de todos los objetivos o mediante una **propiedad**.

 **Ejemplo - Cruce de río.**

$$\text{TEST-OBJETIVO}(S) = \begin{cases} \text{True} & \text{si } S = (0,0,0). \\ \text{False} & \text{en caso contrario.} \end{cases}$$

 **Ejemplo - Ajedrez.**

$\text{TEST-OBJETIVO}(S) = \text{True}$ si y sólo si hay jaque mate en S .

Costo de camino

— — —

A cada **camino** del espacio de estados se le puede asignar un costo numérico (no negativo), denominado **costo de camino**.



Ejemplo - Cruce de río.

$$(2,3,1) \xrightarrow{(2,0)} (0,3,0) \xrightarrow{(1,0)} (1,3,1) \xrightarrow{(0,2)} (1,1,0) \xrightarrow{(0,1)} (1,2,1)$$

Dado que en este camino se realizan 4 cruces en bote, podemos pensar que su costo de camino es 4.

5. Costo de camino

- Dado un camino P , la función **COSTO-CAMINO(P)** retorna el costo de camino de P .

Recordar que está definida para todo camino del espacio de estados.
En las primeras unidades, asumiremos que **el costo de un camino es igual a la suma de los costos individuales de cada acción del camino.**

Por lo tanto, si $P = \langle S_0, A_0, \dots, A_{n-1}, S_n \rangle$,

$$\mathbf{COSTO-CAMINO(P)} = \sum_{i=0}^{n-1} \mathbf{COSTO-INDIVIDUAL(S_i, A_i)}$$

- Dado un estado S y una acción $A \in \mathbf{ACCIONES(S)}$, la función **COSTO-INDIVIDUAL(S, A)** retorna el costo individual de aplicar A en S .



Ejemplo – Cruce de río

— — —

Para todo estado S y acción $A \in \text{ACCIONES}(S)$,

$$\text{COSTO-INDIVIDUAL}(S,A) = 1.$$

Luego, el costo de camino es exactamente la cantidad de acciones (cruces en bote) realizadas.



Formulación – Cruce de río

— — —

En general, **un problema puede admitir múltiples formulaciones**, en base a la representación de estado y acción elegida.

Repasando... El problema de cruce de río puede formularse como problema de búsqueda de la siguiente forma.

- **Estado:** $(z, h, b) \in \{0, 1, 2\} \times \{0, 1, 2, 3\} \times \{0, 1\}$

representa que en el lado izquierdo del río hay z zombies, h humanos y b botes.

El número total de estados es $3 \cdot 4 \cdot 2 = 24$.

- **Acción:** $(z', h') \in \{0, 1, 2\} \times \{0, 1, 2, 3\}$ tal que $1 \leq z' + h' \leq 2$

representa que cruzan el río z' zombies y h' humanos.



Formulación – Cruce de río

— — —

1. Estado inicial.

$(2, 3, 1)$

2. Acciones. Dado un estado (z, h, b) ,

$$\text{ACCIONES}(z, h, 1) = \begin{cases} \{ \} & \text{si } z > h \\ \{(z', h') \in \{0, \dots, z\} \times \{0, \dots, h\} : 1 \leq z' + h' \leq 2\} & \text{si } z \leq h \end{cases}$$

$$\text{ACCIONES}(z, h, 0) = \begin{cases} \{ \} & \text{si } 2 - z > 3 - h \\ \{(z', h') \in \{0, \dots, 2 - z\} \times \{0, \dots, 3 - h\} : 1 \leq z' + h' \leq 2\} & \text{si } 2 - z \leq 3 - h \end{cases}$$



Formulación – Cruce de río

— — —

3. **Modelo de transiciones.** Dado un estado (z, h, b) y una acción (z', h') ,

$$\text{RESULTADO}((z, h, b), (z', h')) = \begin{cases} (z - z', h - h', 0) & \text{si } b = 1 \\ (z + z', h + h', 1) & \text{si } b = 0 \end{cases}$$

4. **Test objetivo.**

$$\text{TEST-OBJETIVO}(S) = \begin{cases} \text{True} & \text{si } S = (0, 0, 0). \\ \text{False} & \text{en caso contrario.} \end{cases}$$

5. **Costo de camino.** Es la suma de los costos individuales de cada acción del camino.

$$\text{COSTO-INDIVIDUAL}(S, A) = 1, \text{ para toda acción } A \text{ y estado } S.$$

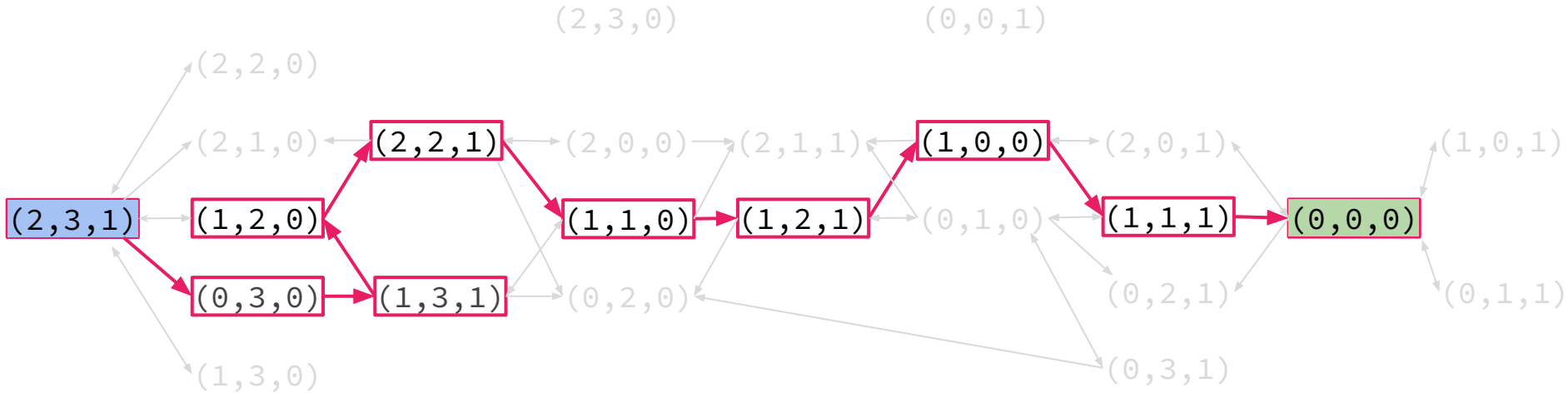
Solución

Dado un problema de búsqueda, una **solución** es un **camino** en el espacio de estados desde el **estado inicial** a un **objetivo**.

Una **solución óptima** tiene el **menor costo de camino** entre todas las soluciones.



Ejemplo – Cruce de río.

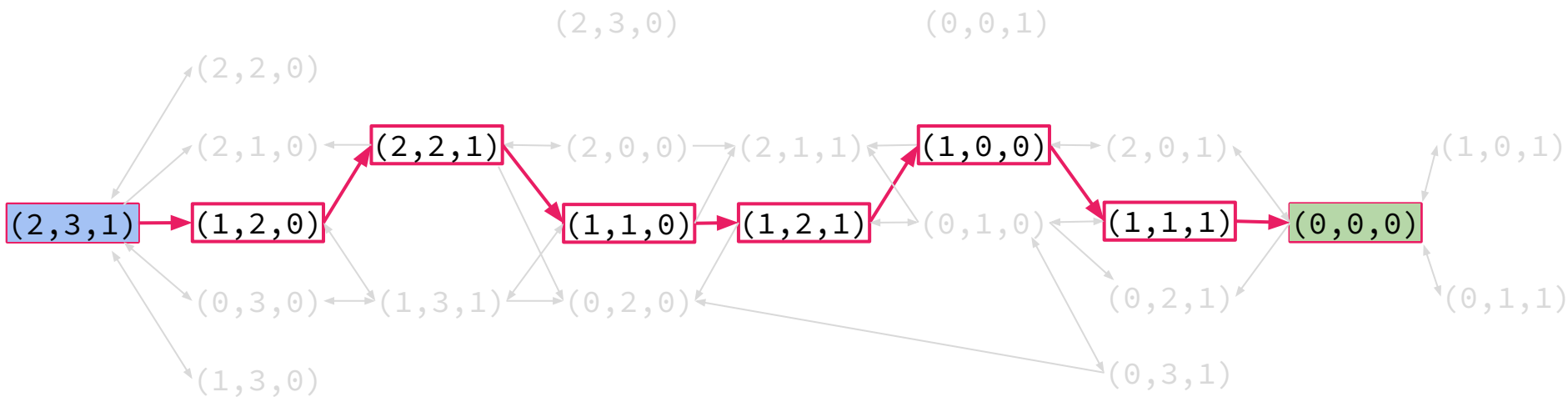


Referencias de estados: inicial y objetivo.

Solución con costo
de camino 9.
No es óptima.



Ejemplo – Cruce de río.

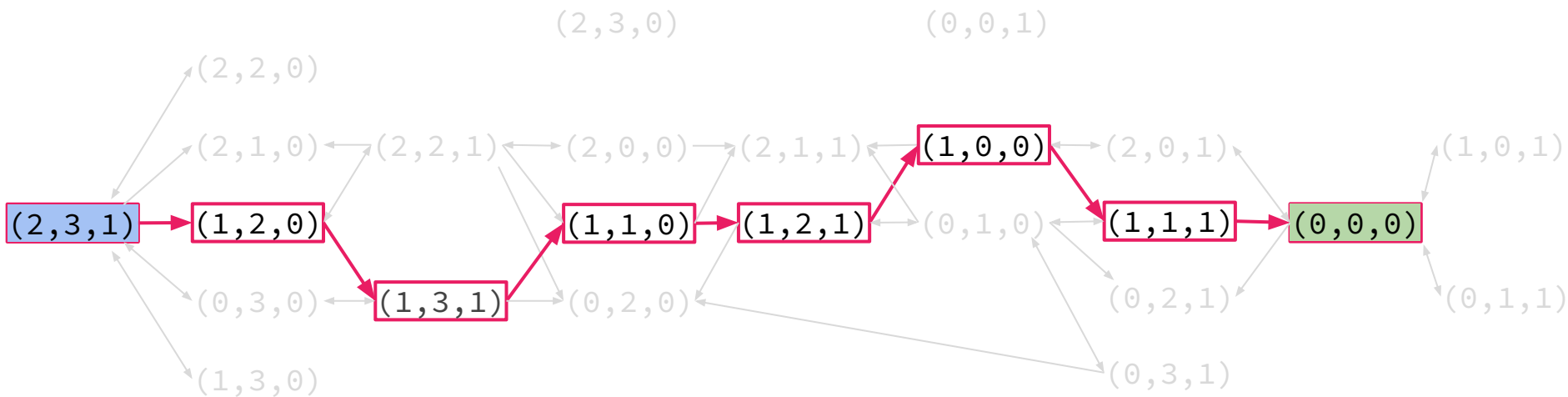


Referencias de estados: inicial y objetivo.

Solución con costo de camino 7.
Es óptima, pero no es única.



Ejemplo – Cruce de río.



Referencias de estados: inicial y objetivo.

Otra solución
óptima.

Algoritmos de búsqueda

Formulado nuestro problema como problema de búsqueda, lo siguiente es **resolverlo**, es decir, encontrar una **solución**, preferentemente **óptima**.

Este proceso se conoce como **búsqueda** y a los algoritmos específicos para esta tarea, **algoritmos de búsqueda**.

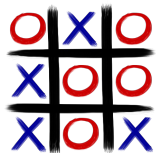
En problemas chicos, es posible hacer búsquedas a mano. **Pero esto no escala con el tamaño del problema y no será posible en general.**

— — —

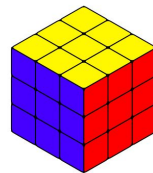
Grandes espacios de búsqueda

— — —

Problemas más interesantes suelen tener miles o millones de estados.



Problema	Estados
Ta-te-ti	$3^9 = 19.683$ (765)
Cubo Rubik 2x2	3.674.160
Cubo Rubik 3x3	4.3×10^{19}
Ajedrez 8x8	$\approx 10^{44}$





Motivación

Con lo visto en **Programación 2**, ya conocemos un algoritmo de búsqueda.

1. Construir el digrafo de espacio de estados.
2. Definir el peso de cada arco como el costo individual de la acción que representa.
3. Usar el algoritmo de **Dijkstra** (adaptado a grafos dirigidos) para computar el camino más corto desde el estado inicial a cada uno de los estados objetivo.
4. Devolver el camino más corto.

¿Cuáles son las limitaciones de este algoritmo?



Motivación

-- --



El espacio de estados puede ser infinito.



Aún siendo finito, podría ser demasiado grande para ser almacenado en la memoria.

Nos interesan problemas de búsqueda con miles o millones de estados, en los cuales no es eficiente, y muchas veces posible, almacenar por completo el digrafo de espacio de estados en la memoria.



Próximamente

— — —

Buscar soluciones de manera eficiente en espacios de estados grandes o infinitos es una de las áreas más importantes de la IA.

Veremos **algoritmos de búsqueda generales**, que pueden aplicarse para resolver cualquier problema de búsqueda.