Ayudantía 7

Repaso y Tips Tarea 3

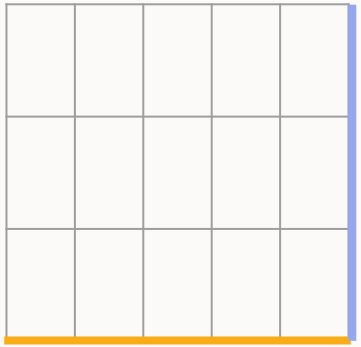
Por Sofía Hosiasson

4 de Octubre de 2024

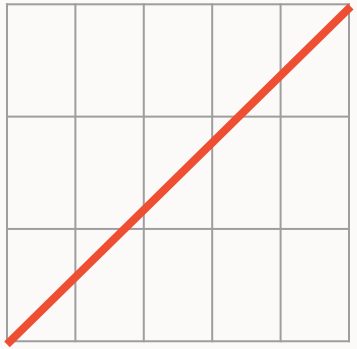
Repaso Heurísticas de Búsqueda

Una heurística es una función que estima el costo restante desde un nodo hasta el objetivo.

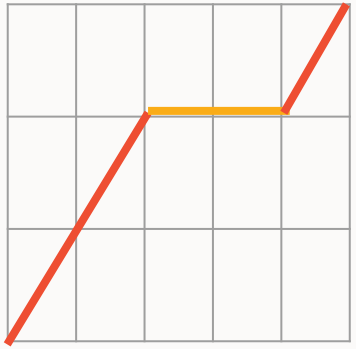
Manhattan



Euclidiana



Octile



Admisibilidad

Definición

Una heurística **h** se dice **admisible** si para todo **estado s** se cumple que:

$$h(s) \leq h^*(s)$$

Donde **h*(s)** es el costo de un camino óptimo desde el **estado s** a un estado objetivo.

Repaso algoritmo A*

A* es un algoritmo utilizado en problemas de **búsqueda**, su característica distintiva es que utiliza heurística, utilizando la siguiente función de evaluación:

- La **rapidez y complejidad** computacional está estrechamente ligada a la **heurística** seleccionada
- Siempre entrega soluciones **óptimas** si se usa con una heurística **admisible**

$$f(n) = g(n) + h(n)$$

- g(n): Costo de un camino desde s0 hasta un nodo n (NO ES FUNCIÓN, SE ACTUALIZA A MEDIDA QUE SE VA EXPLORANDO)
- h(n): Heurística

Repaso algoritmo A*

Input: S, A, s0 , G

Output: nodo objetivo

for each $s \in S$ **do** $g(s) \leftarrow \infty$

Open $\leftarrow \{ s0 \}$

$g(s0) \leftarrow 0$

$f(s0) \leftarrow h(s0)$

while Open $\neq \emptyset$

$u \leftarrow$ Extraer(**Open**)

if $u \in G$ **return** u

for each $v \in Succ(u)$ **do**

Inserta v a Open

Extracción del menor valor
f(s) = g(s) + h(s)

Repaso algoritmo A*: Insertar v

Costo de llegar de v por u

Seguimos si costo es menor al ya explorado

Inserta *v* a Open

costo$_v$ = g(**u**) + c(u, v)

if costo$_v$ ≥ g(**v**) **return**

Parent(v) ← **u**

g(v) ← costo$_v$

f(v) ← g(v) + h(v)

if v ∈ Open **then** reorder Open

else insertar v en Open

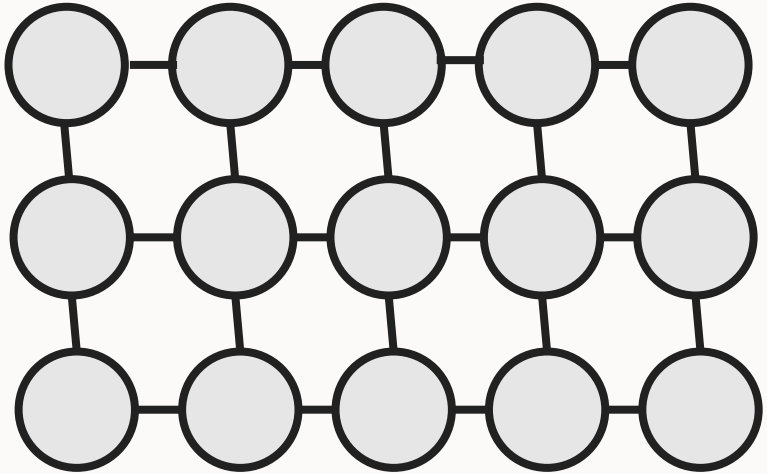
Tarea 3.1: A* vs Early A*

```
1. for each s in S do :  g(s) ← ∞
2. g(s0) ← 0
3. f(s0) ← h(s0)
4. Insertar s0 en Open
5. while Open ≠ ø do
6.     Extraer un u desde Open con menor valor de f
7.     if u es objetivo then :  return u
8.     for each v in Succ(u) do
9.         cost_v ← g(u) + c(u, v)
10.        if cost_v < g(v) then:
11.            parent(v) ← u
12.            g(v) ← cost_v
13.            f(v) ← g(v) + h(v)
14.            if v not in Open then : Insertar v en Open
15.            else : Ajustar Open
```

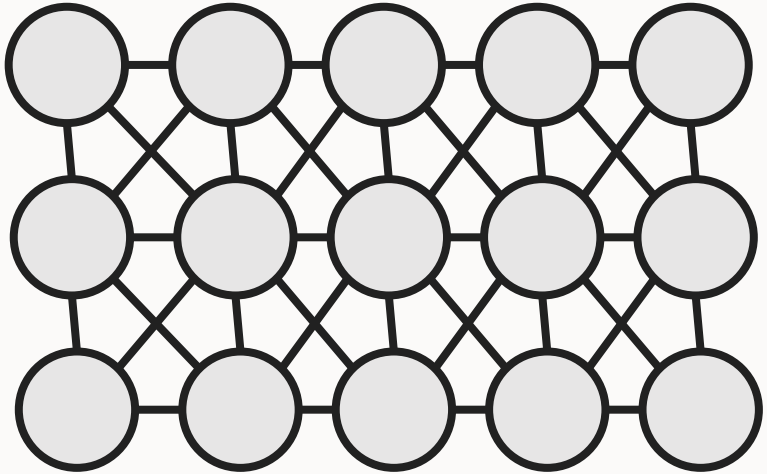
```
1. for each s in S do :  g(s) ← ∞
2. g(s0) ← 0
3. f(s0) ← h(s0)
4. U ← ∞
5. Insertar s0 en Open
6. while Open ≠ ø do
7.     Extraer un u desde Open con menor valor de f
8.     if U ≤ f(u) then : return sol
9.     for each v in Succ(u) do
10.        cost_v ← g(u) + c(u, v)
11.        if cost_v ≥ g(v) then:  continue
12.        if v es objetivo and g(v) < U then: sol ← v , U ← f(v)
13.        parent(v) ← u
14.        g(v) ← cost_v
15.        f(v) ← g(v) + h(v)
16.        if f(v) ≤ U then:
17.            if v not in Open then : Insertar v en Open
18.            else : Ajustar Open
```

Tarea 3.2: DCCarrera

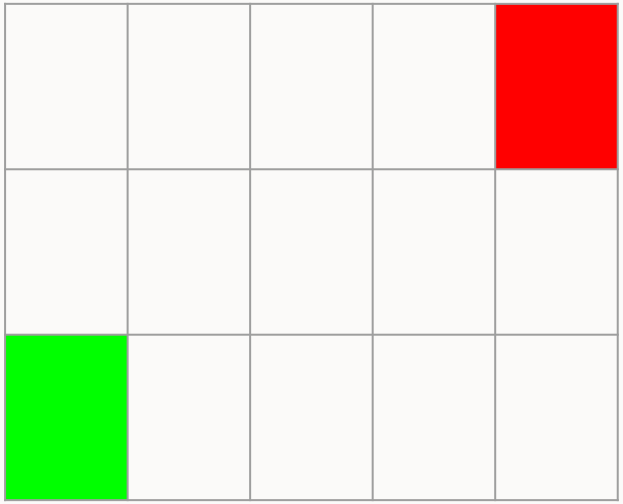
Conectividad 4

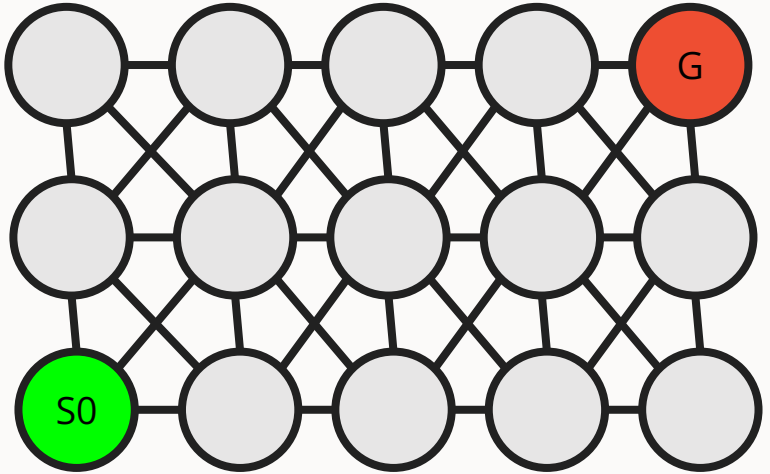


Conectividad 8

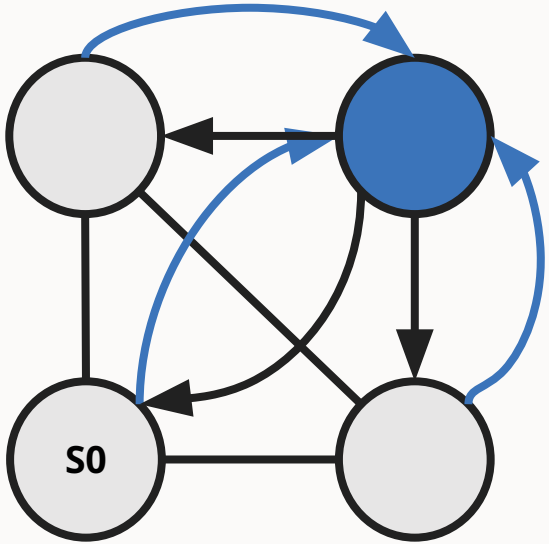


Tarea 3.2: DCCarrera



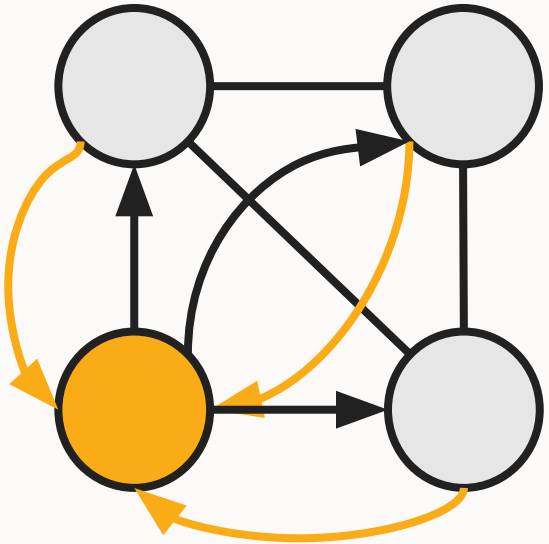


Tarea 3.2: DCCarrera



TERRENO HIELO

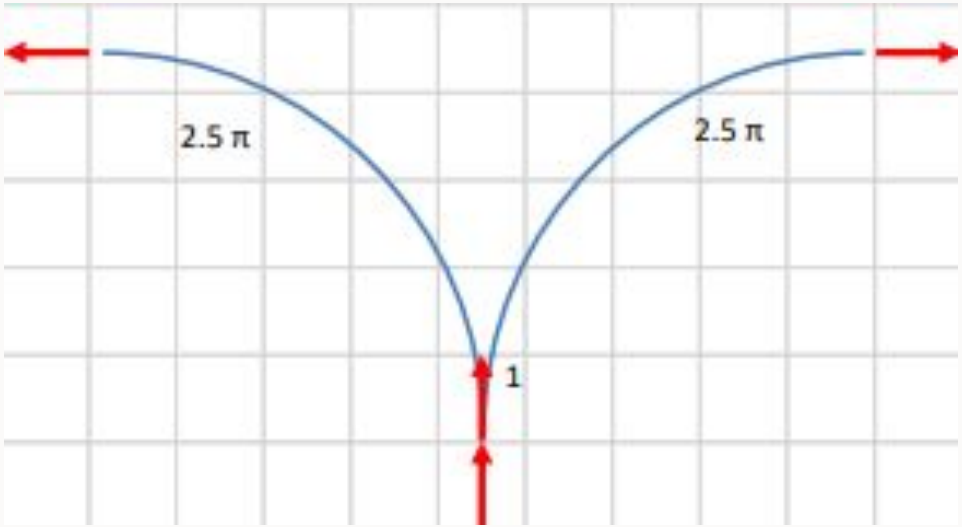
Tarea 3.2: DCCarrera



TERRENO ARENA

Tarea 3.2: DCCarrera

State Lattices



Repaso Minimax

El algoritmo Minimax es una técnica utilizada principalmente en juegos de dos jugadores para determinar la **mejor jugada posible**. Este algoritmo asume que ambos jugadores actúan óptimamente: uno intenta **maximizar su ganancia** (Max), mientras que el otro intenta **minimizar la ganancia del oponente** (Min).

Repaso Minimax

Se representa el juego como un árbol donde:

- Nodos Max: Representan las decisiones del jugador que intenta maximizar el valor.
- Nodos Min: Representan las decisiones del jugador que intenta minimizar el valor

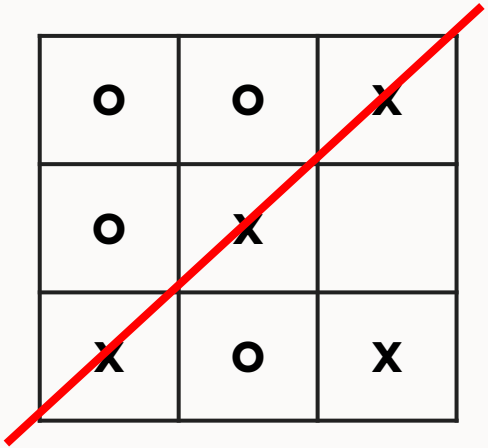
Ejemplo Minimax: Gato

O: yo
X: adversario

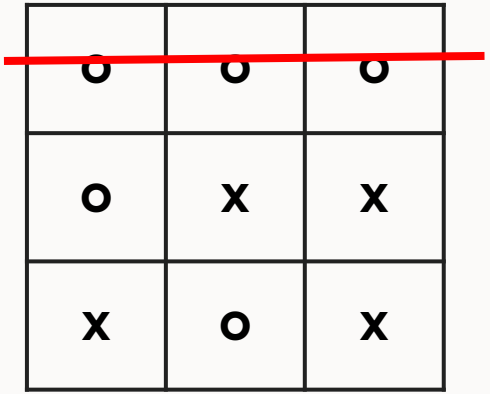
O		
O	X	
X	O	X

Partimos desde este estado (es mi turno, aún no juego)

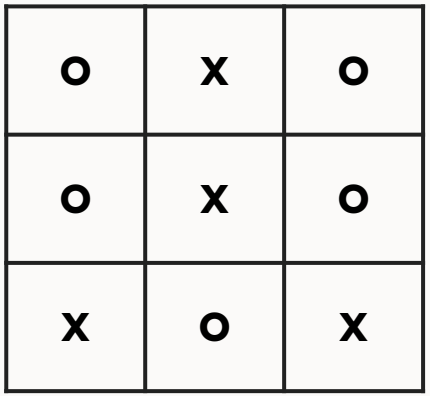
Ejemplo Minimax: Gato



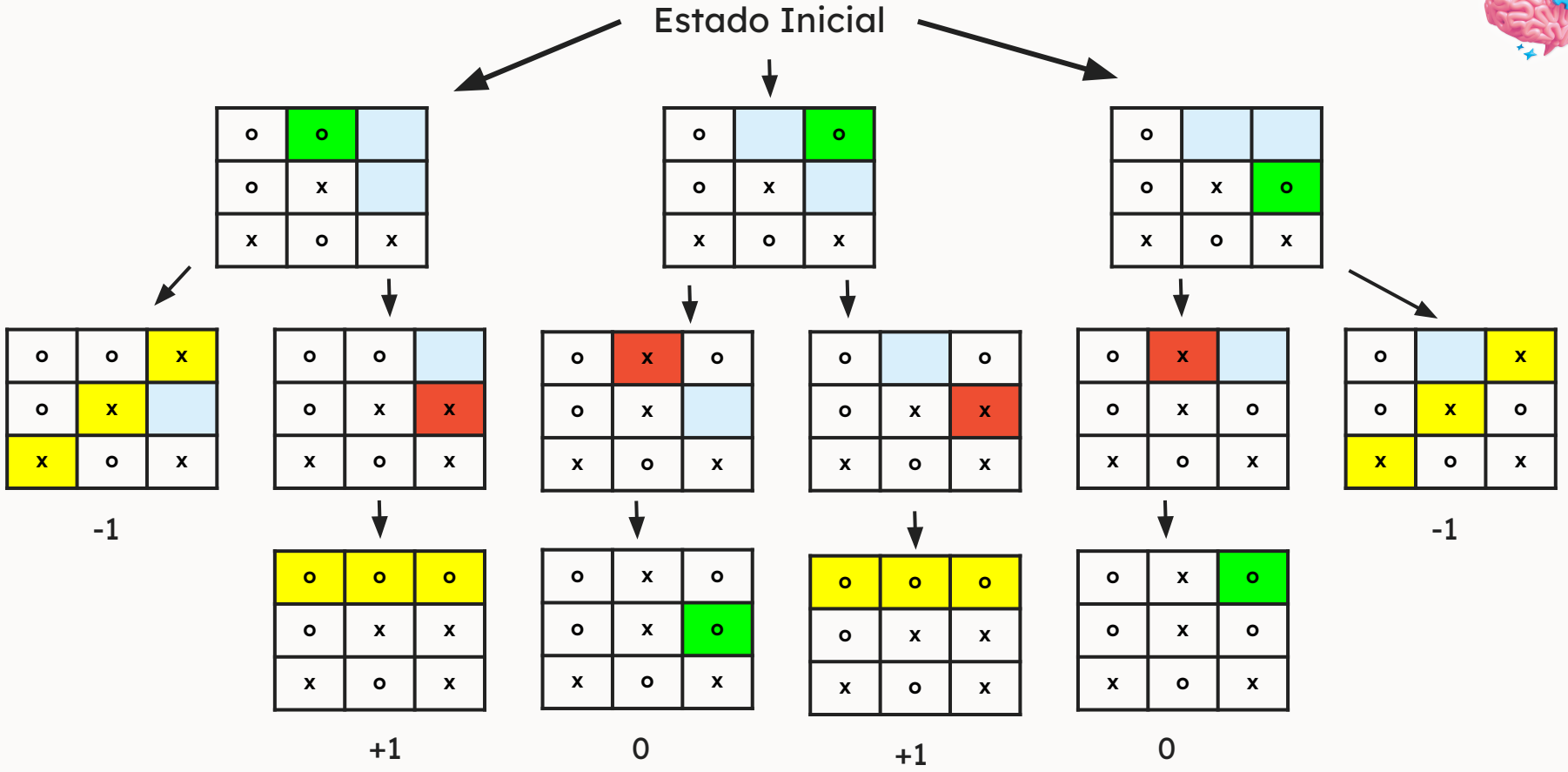
PERDI (-1)

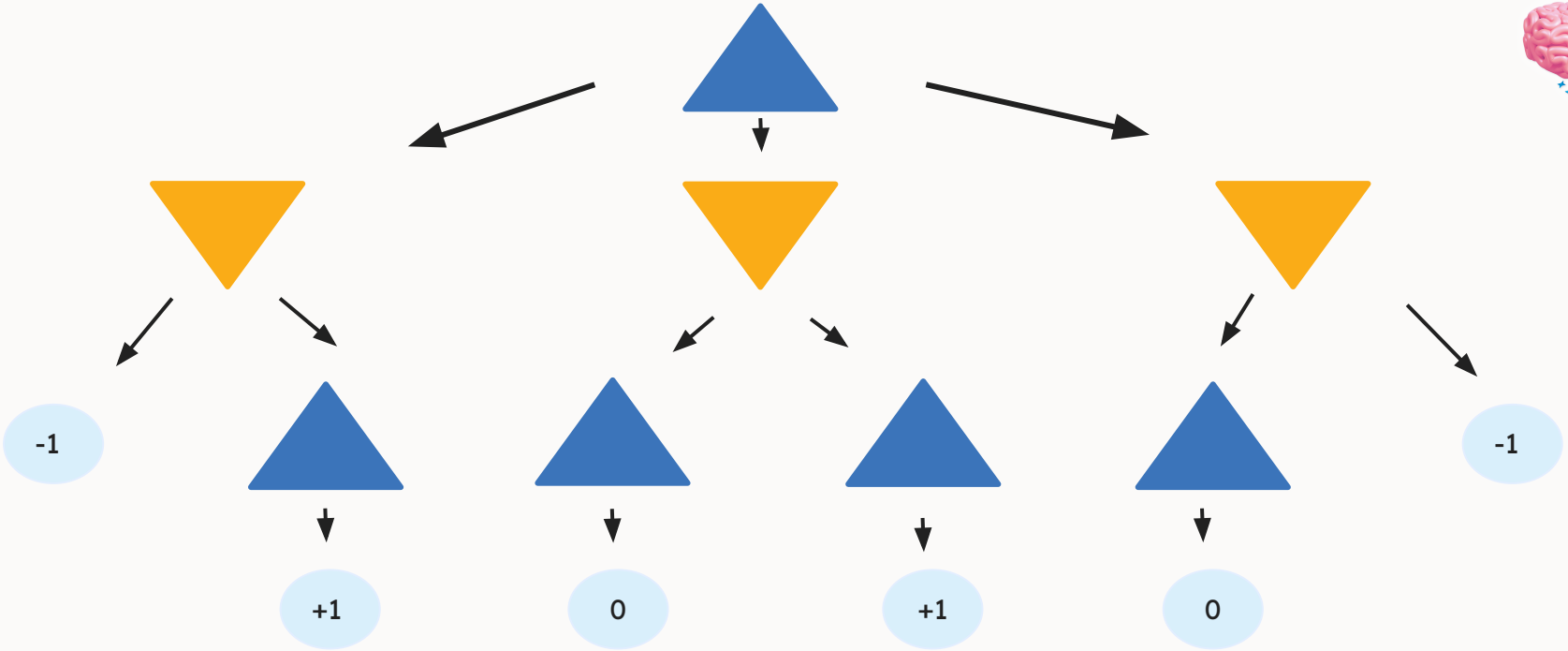


GANE (+1)

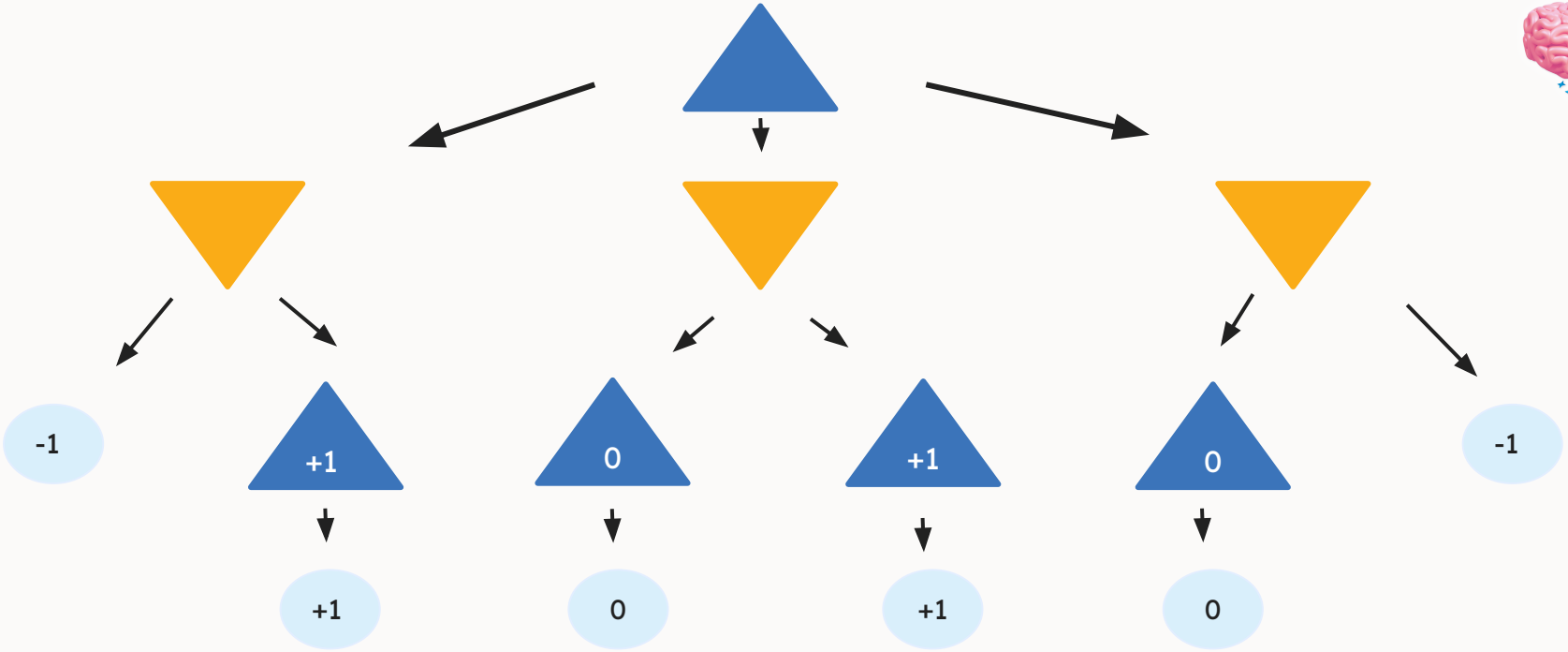


EMPATE (0)

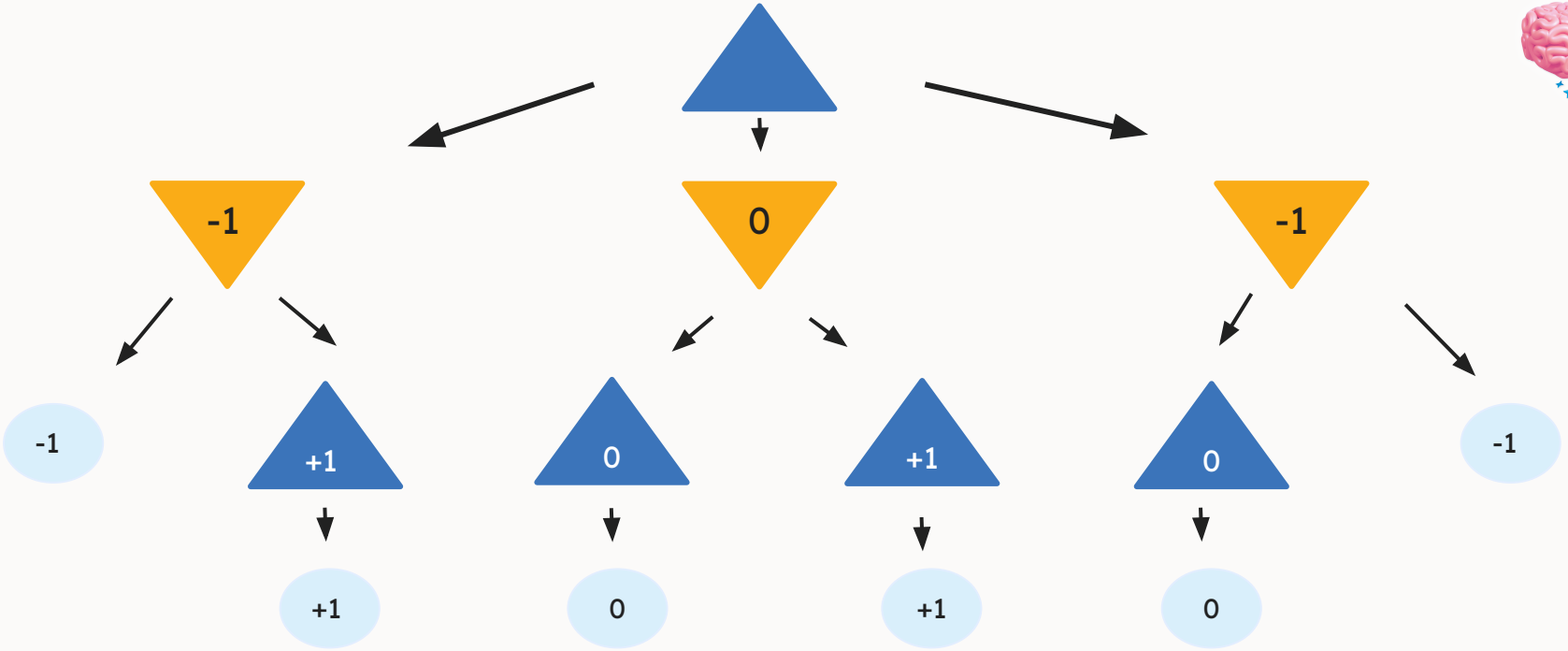

Estado Inicial
-1
+1
0
+1
0
-1



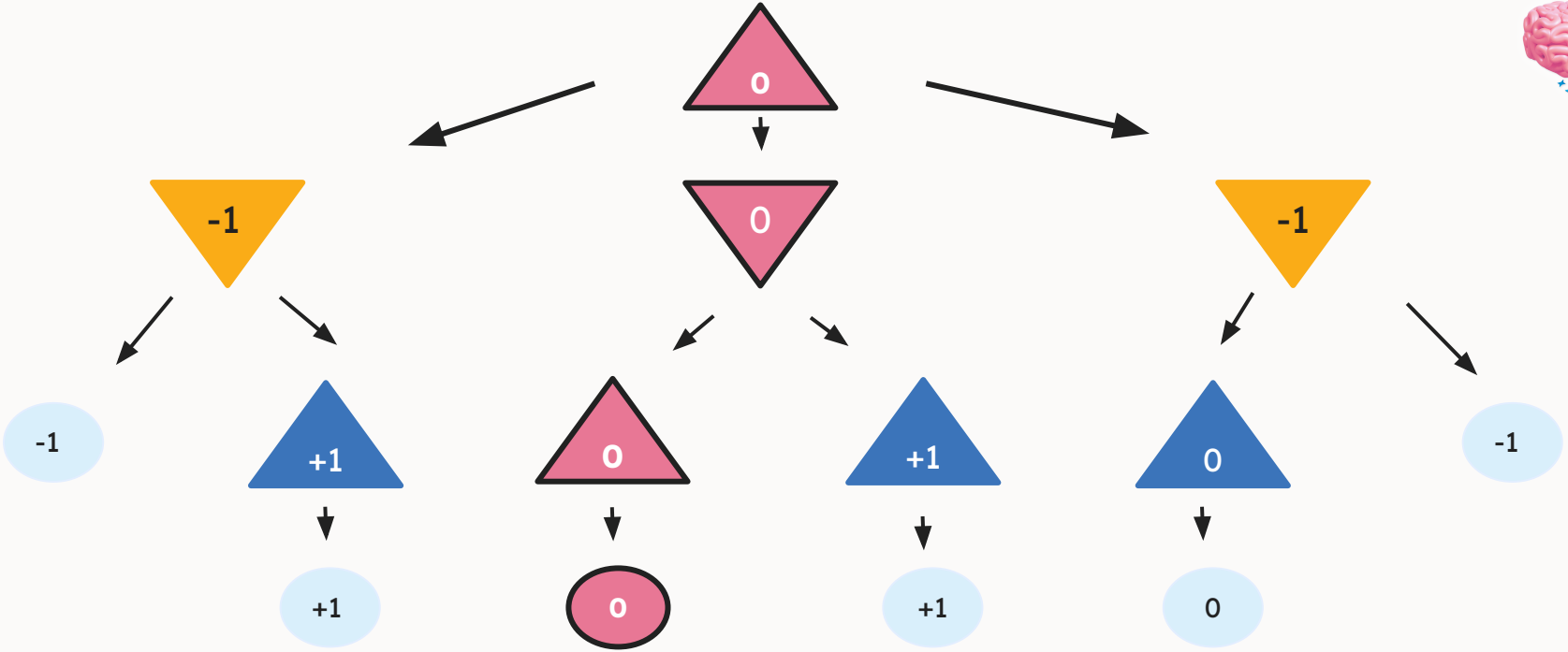
ADVERSARIO QUIERE MINIMIZAR MI PTJE vs YO QUIERO MAXIMIZAR MI PTJE



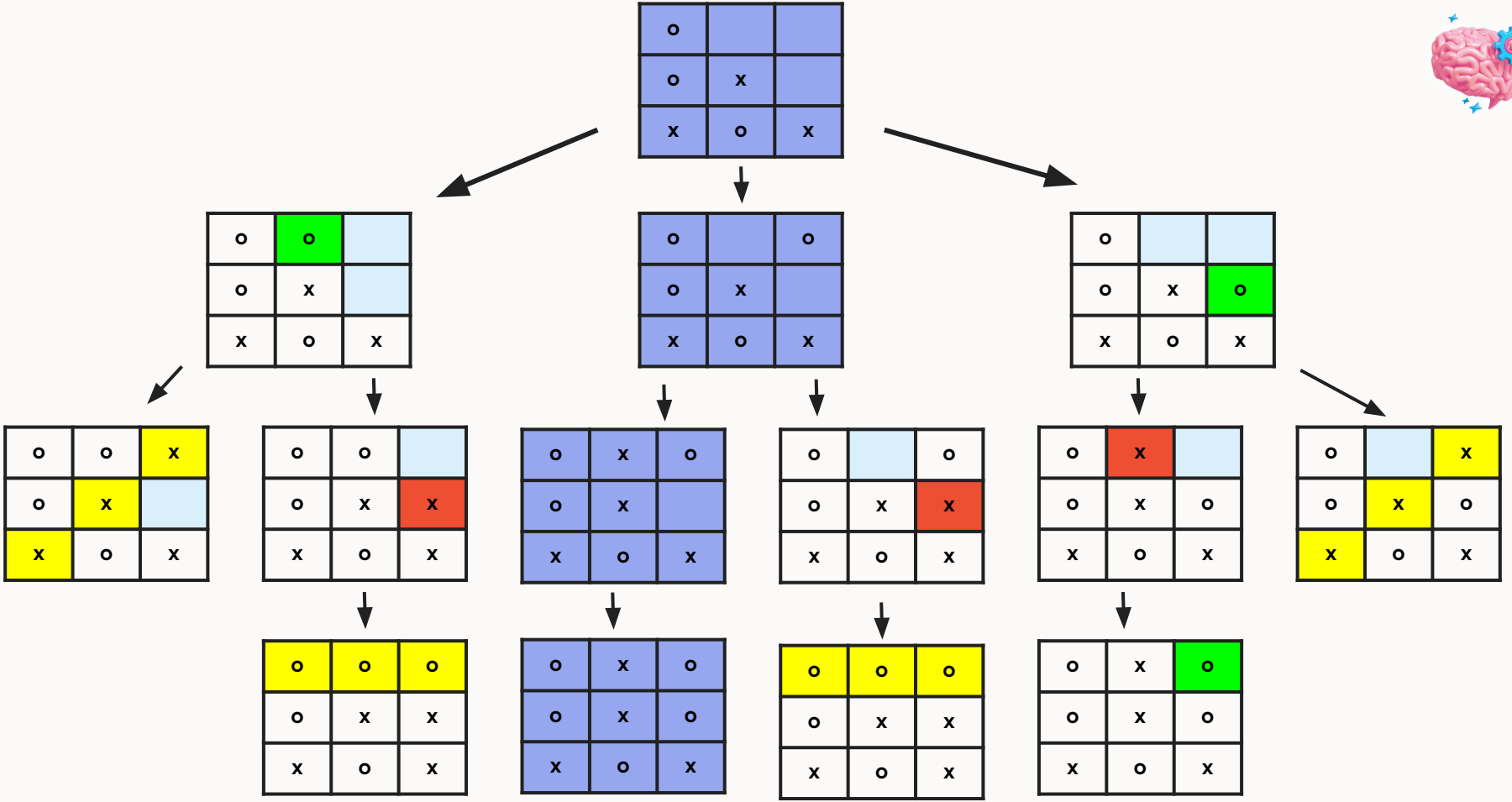
ADVERSARIO QUIERE MINIMIZAR MI JUGADA **vs** YO QUIERO MAXIMIZAR MI JUGADA



ADVERSARIO QUIERE MINIMIZAR MI JUGADA **vs** YO QUIERO MAXIMIZAR MI JUGADA

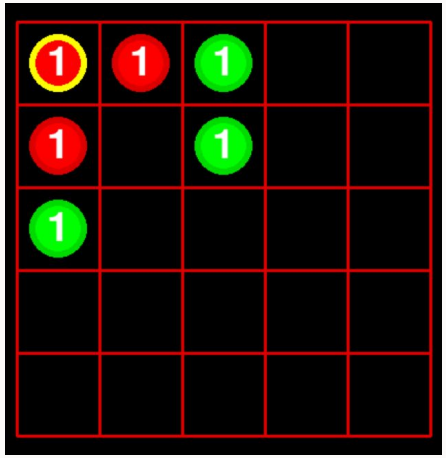


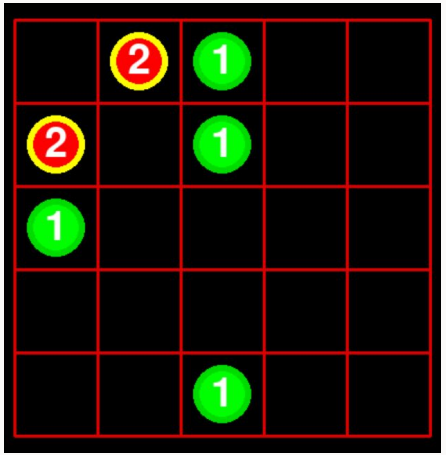
ADVERSARIO QUIERE MINIMIZAR MI JUGADA vs YO QUIERO MAXIMIZAR MI JUGADA


o
o x
x o x
o o
o x
x o x
o o
o x
x o x
o
o x
x o x
o o x
o x
x o x
o o
o x x
x o x
o x o
o x
x o x
o o
o x x
x o x
o x x
o o x
o x
x o x
o o o
o x x
x o x
o x o
o x o
x o x
o o o
o x x
x o x
o x o
o x o
x o x
o x
o x o
x o x
o x
o x o
x o x

Tarea 3.3: DCChain

En cada turno, el jugador hace un movimiento, y luego "el virus" (oponente) responde adaptando su estrategia para minimizar la curación.





Ayudantía 7

Repaso y Tips Tarea 3

Por Sofía Hosiasson

4 de Octubre de 2024