



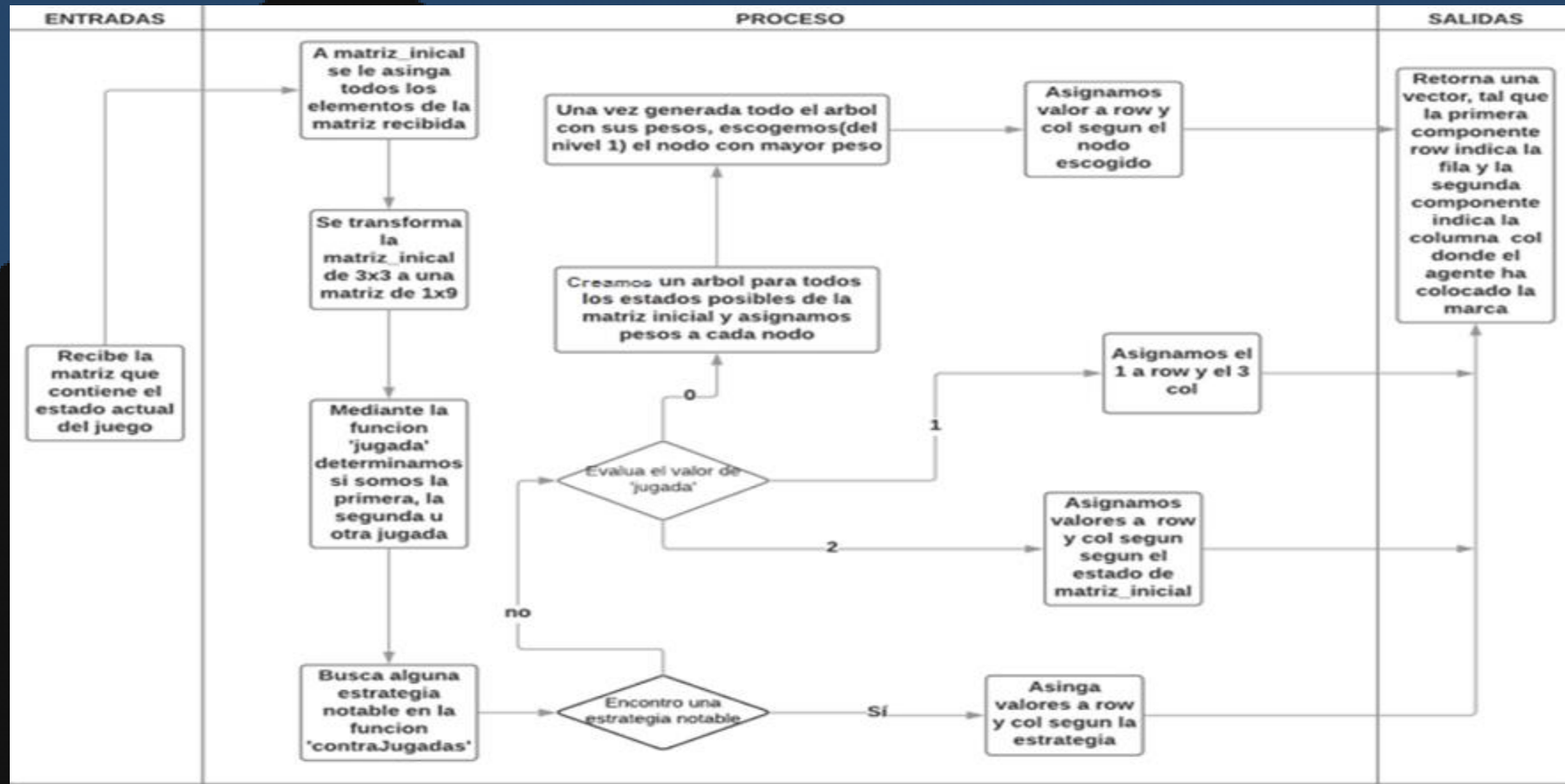
Matemáticas Discretas

Proyecto de Segundo Bimestre, Tic Tac Toe

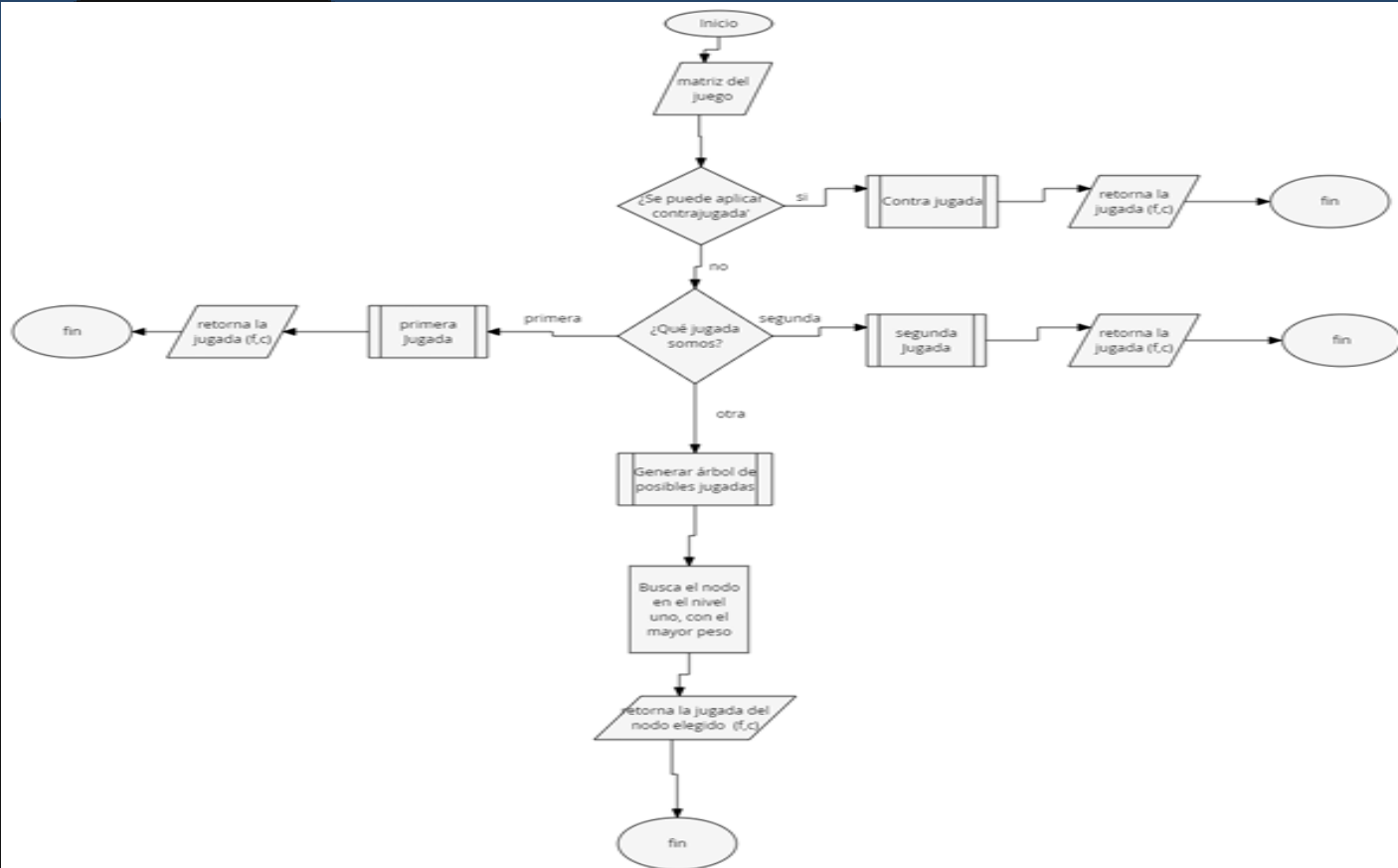
Objetivo

Aplicar la teoría de árboles para el diseño de un agente capaz de jugar el tic-tac-toe (juego del gato o tres en raya) con un bajo costo computacional y un desempeño mejor que el promedio de las personas.

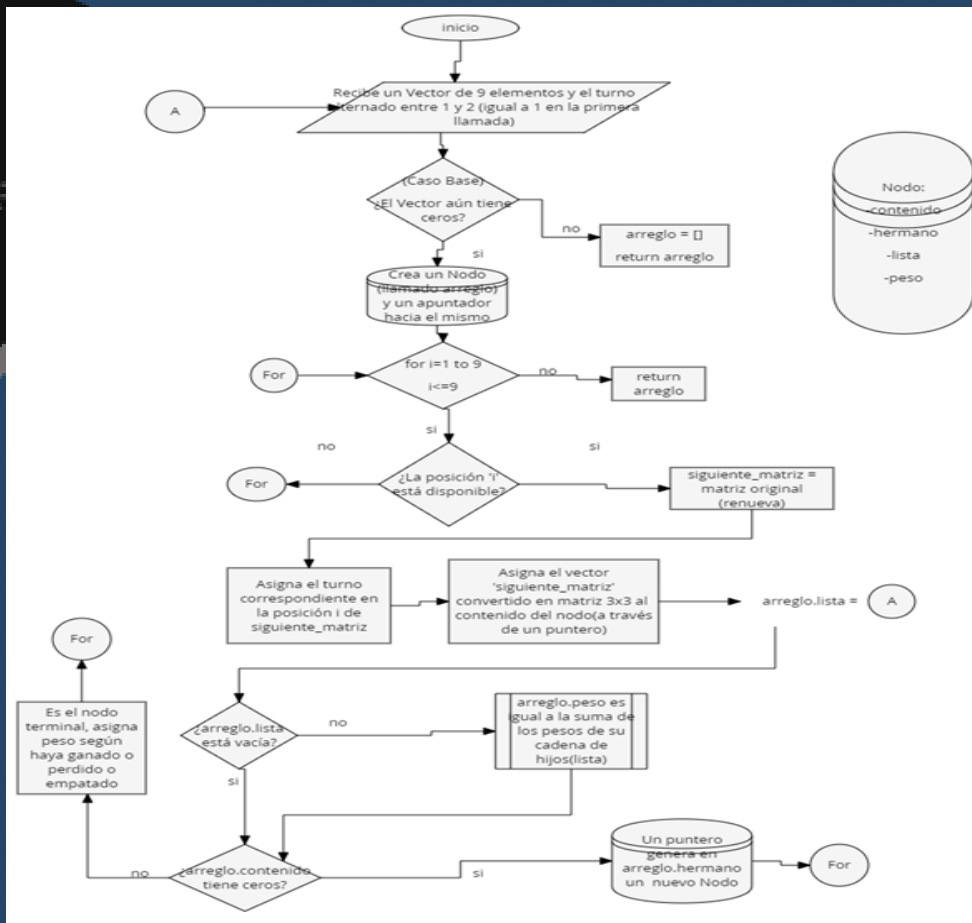
DIAGRAMA DE FLUJO



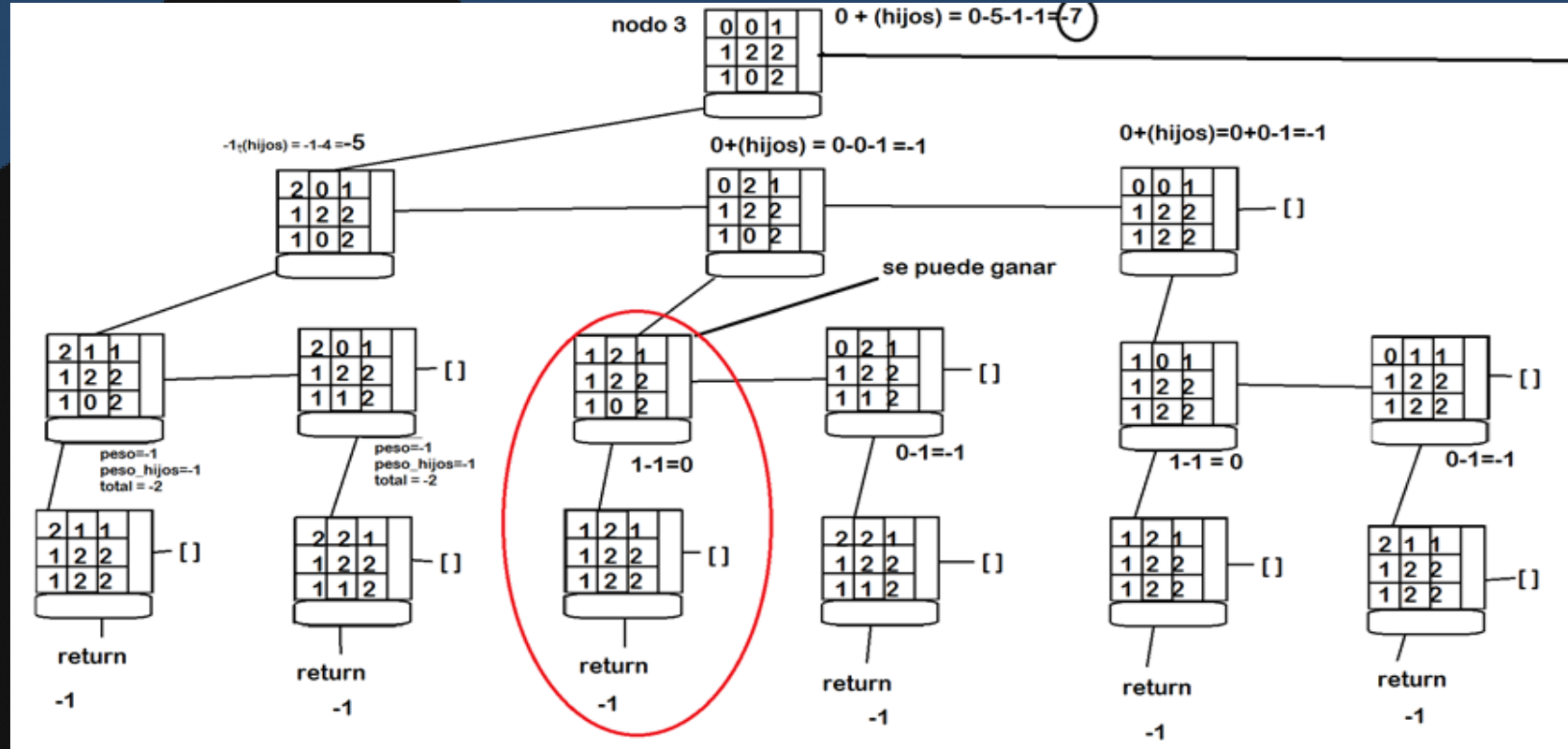
Otra forma de ver el mismo diagrama de flujo de *playAgentStudent*



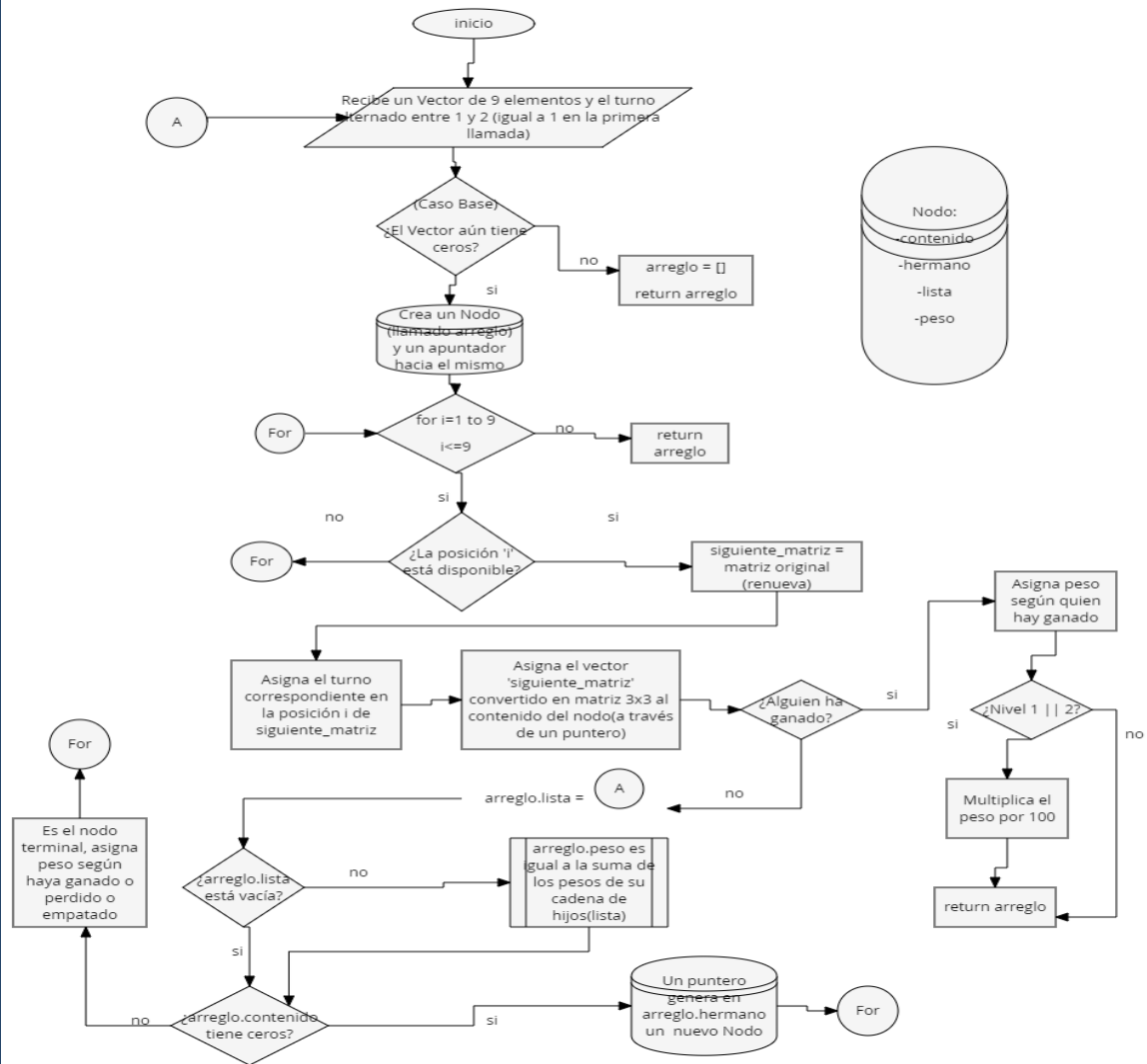
Función que genera todas las posibles jugadas del 'tres en raya' a partir de una matriz dada



Deficiencia de la estrategia (mala asignación)

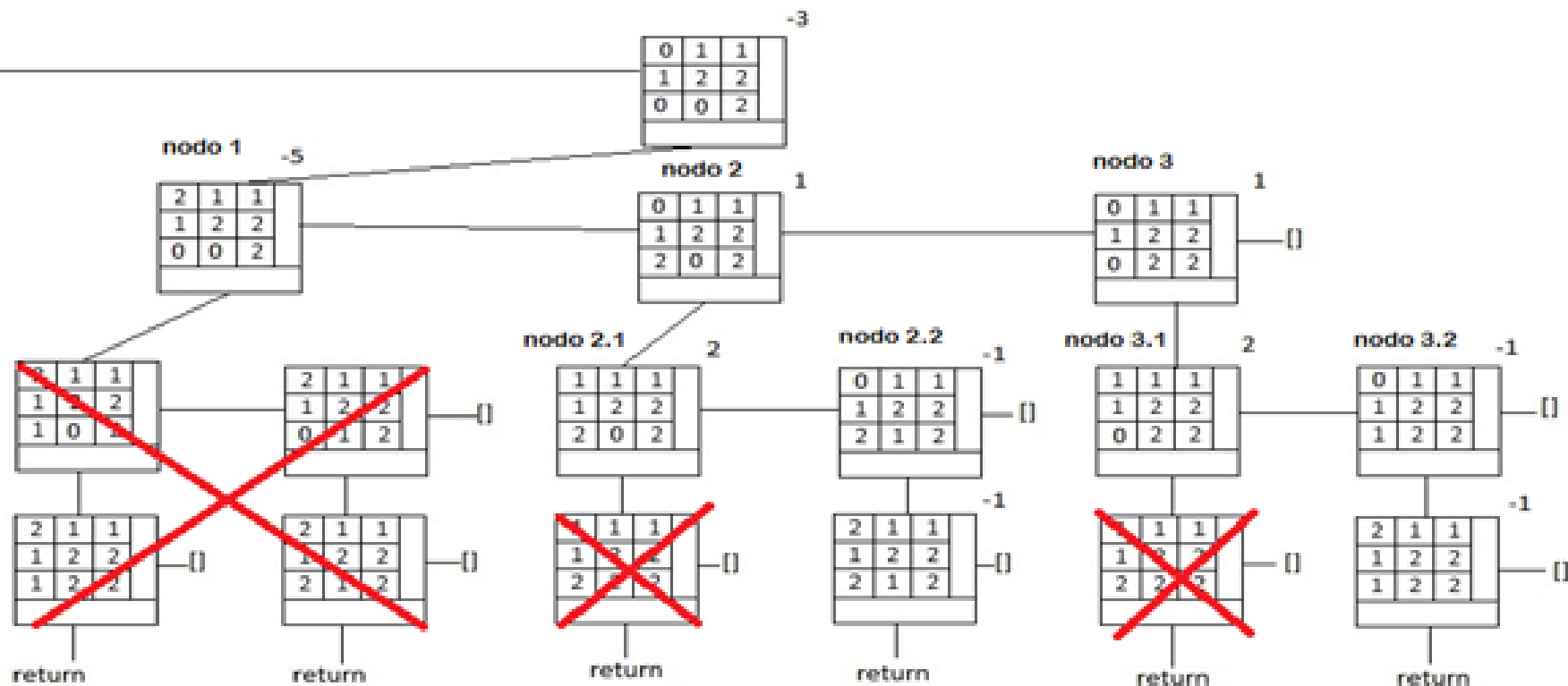


Función mejorada
que genera todas
las posibles jugadas
del 'tres en raya' a
partir de una matriz
dada

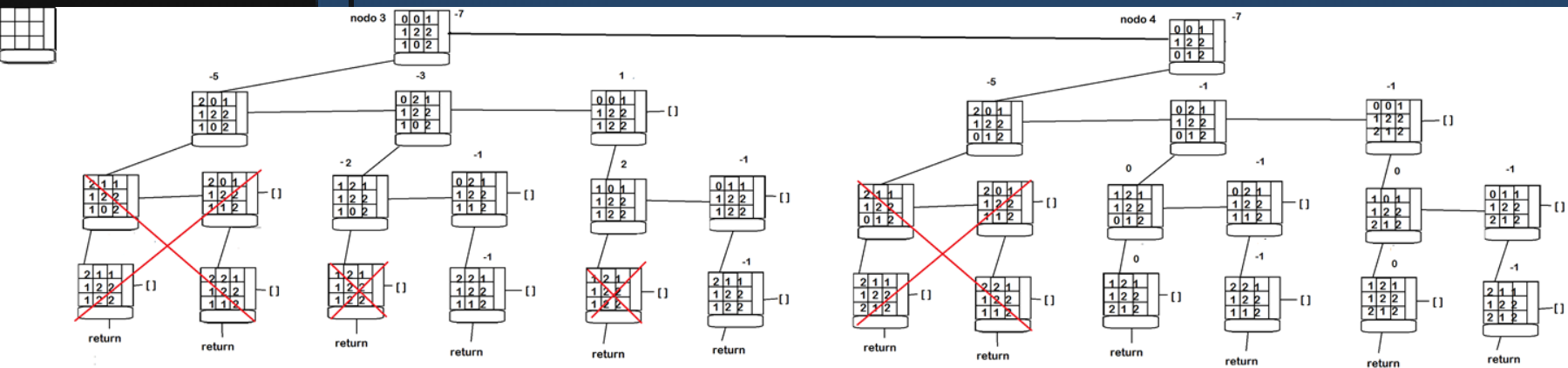


Estrategia solventada.

Ejemplos de la forma actual de asignación de pesos



5



Cantidad de nodos que genera la matriz con 'n' espacios disponibles

De manera analítica se obtuvo una fórmula para obtener la cantidad de nodos que cada padre generaba (esto es parte fundamental para la estrategia que usamos), para tres espacios disponibles ($n=3$) es:

$$1+(n-1)+(n-1)(n-2)]*n \text{ (uno)}$$

Lo que se interpreta como: el primer descendiente (1) más el número de nodos que genera el primer descendiente ($n-1$) más el número de nodos que genera cada hijo ($n-2$) multiplicado por los hermanos de ese hijo ($n-1$) y como solo habíamos considerado el primer descendiente, debemos multiplicar por todos los demás descendientes del primer nivel (n). De manera general:

$$[1+(1)(n-1)+\dots+(1)(n-1)(n-2)\dots(n-(n-1))]*n$$

Si desarrollamos la ecuación (uno):

$$=n+n(n-1)+n(n-1)(n-2)$$

$$= \frac{n(n-1)!}{(n-1)!} + \frac{n(n-1)(n-2)!}{(n-2)!} + \frac{n(n-1)(n-2)(n-3)!}{(n-3)!}$$

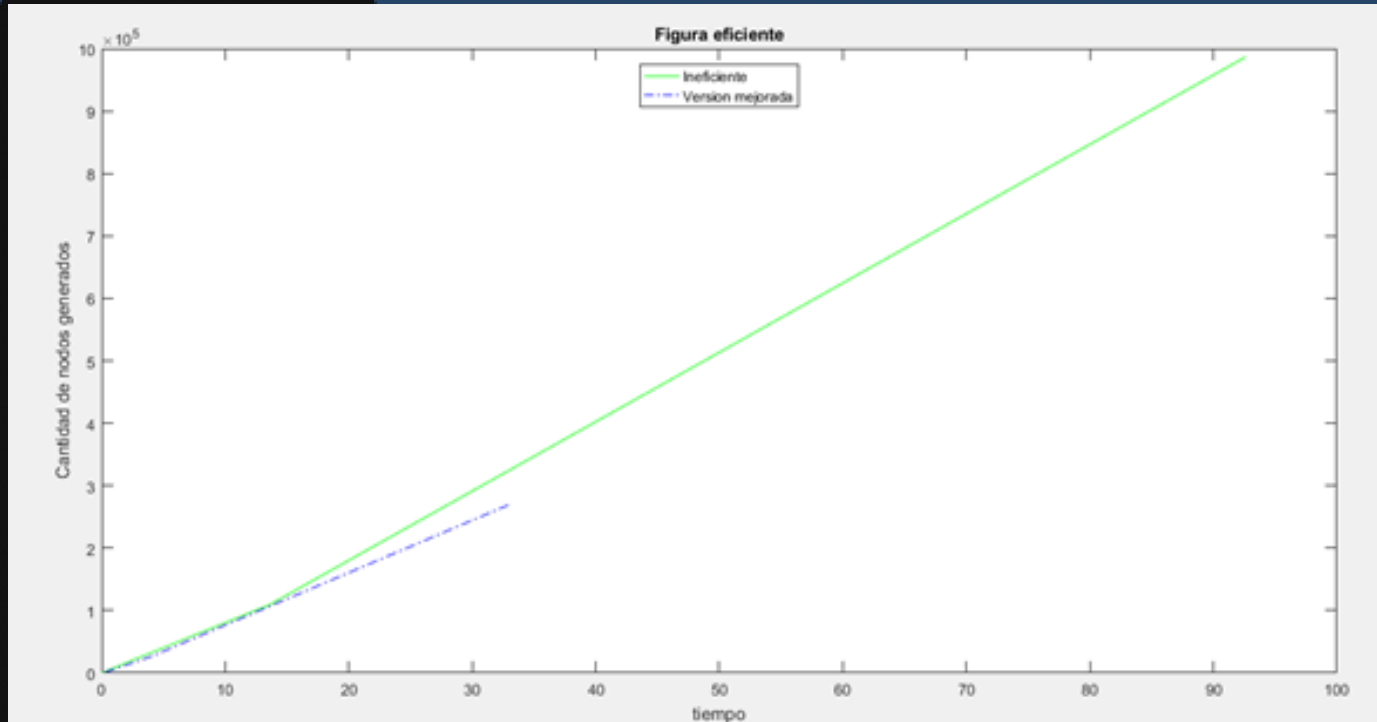
$$= n! \left(\frac{1}{(n-1)!} + \frac{1}{(n-2)!} + \frac{1}{(n-3)!} \right) = n! \sum_{j=1}^n \frac{1}{(n-j)!} \quad (\text{dos})$$

Esta fórmula (dos) se reafirmó en el momento en que ubicamos un contador en el programa que aumentaba cada vez que se asignaba un contenido a un nodo generado. Con lo que de paso obtuvimos esta relación.

Home	About Us	Contact Us	Privacy Policy
Sitemap	Terms & Conditions	FAQ	Feedback

Espacios disponibles de la matriz (n)	Cantidad de nodos que genera (sin recortes)	Tiempo promedio de ejecución
1	1	0.0000
2	4	0.0003
3	15	0.0015
4	64	0.0100
5	325	0.0400
6	1956	0.2200
7	13699	1.5900
8	109600	13.3400
9	986409	90.7200

Sin embargo el árbol actualmente (con la implementación de recortar ramales innecesarios) no genera esa cantidad de nodos como se puede ver en la figura 2 (línea azul):



Comparación entre el anterior y actual algoritmo

Espacios disponibles de la matriz (n)	Nodos que debe generar(línea verde)	Nodos reales que genera (línea azul)	Tiempo que debe tardar (línea verde)	Tiempo que se demora (línea azul)
1	1	1	0.0000	0.0000
2	4	4 <u>aprox</u>	0.0003	0.0003
3	15	15 <u>aprox</u>	0.0015	0.0015
4	64	43 <u>aprox</u>	0.0140	0.0120
5	325	130 <u>aprox</u>	0.0400	0.0250
6	1956	691 <u>aprox</u>	0.2200	0.0970
7	13699	3000aprox	1.5900	0.1400
8	109600	26852aprox	13.3400	4.0600
9	986409	269173aprox	90.7200	33.2800

ESTRATEGIAS

Estrategia de la primera y segunda jugada :

La 'Primera' y 'Segunda' jugada nacieron como estrategia para asegurar una victoria o al menos empate y porque resultaba más eficiente crear el árbol a partir de 7 espacios en blancos.

Según las reglas establecidas, el programa debe responder en menos de dos segundos, por lo que, si generamos el árbol con 9 espacios en blanco, perderíamos instantáneamente el juego

La gráfica de espacios disponibles por llenar vs tiempo (ver figura 3) nos ayudó a determinar que con 7 espacios en blancos era suficiente para cumplir dicha regla.

Primera Jugada:

Los jugadores más experimentados colocan la primera jugada en una esquina cada vez que empiezan primero. Esto le da al oponente mayor oportunidad para cometer un error. Si el oponente responde colocando en cualquier parte que no sea el centro, las probabilidades de victoria aumentan. Es por ello que el algoritmo elige una de las 4 esquinas como primera jugada.

1		

Segunda Jugada:


Si el oponente juega primero y comienza en una esquina, siempre se coloca la segunda jugada en el centro. La segunda jugada no debe ubicarse en una esquina contraria, porque perderíamos automáticamente, sin vuelta atrás. Con esta estrategia, se estima que el juego terminará seguramente en empate. En teoría, se puede ganar desde esta posición, pero el oponente tendría que cometer un gran error.

2		
	1	

- Cuando el oponente comience colocando en el centro, se coloca la segunda jugada en una esquina. Básicamente, no hay forma de ganar, solo de empatar, desde esta posición; a menos que el oponente cometa un error.

		1
	2	

- Si el oponente coloca la primera jugada en un borde (lado) en lugar de en una esquina o en el centro, hay posibilidad de ganar. Se coloca la segunda jugada en una de las esquinas.

	2	1
		

Contra-Jugadas

La función implementada “contra Jugadas” sirve para solventar algunas debilidades que pueda tener el algoritmo y también para alimentarlo de “buenas jugadas”. Es decir, si encuentra alguna “jugada notable” en donde la victoria es clara, aplicamos esa jugada para ganar. Por ejemplo:

Supongamos que tenemos esta jugada. Si colocamos un 1 en cualquiera de las esquinas sobrantes perderemos inmediatamente.

2			2		1	2		1	2		1	2		1
	1			1			1		1	1		1	1	
		2			2	2		2	2		2	2	2	2

La 'contra jugada' está programada para ver casos simétricos de esta jugada con la función rot90, y ubicar su jugada en uno de los **lados**. Ya que si ubicamos en esas posiciones nuestro '1' contrarrestamos esa jugada

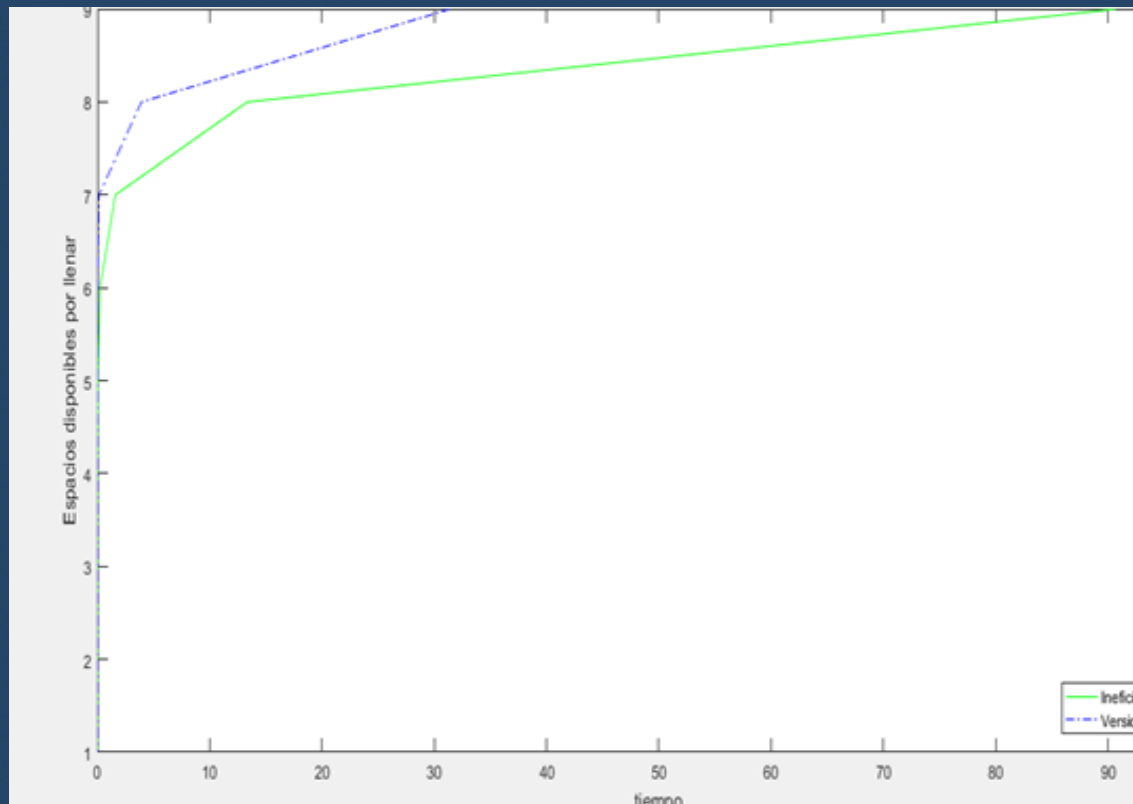
2			2			2		2	2	1	2	2	1	2
	1	1	2	1	1	2	1	1	2	1	1	2	1	1
		2			2	1		2	1		2	1	2	2

Por qué se da este problema?

Porque los pesos nos pueden traer ambigüedad. El caso anterior es un ejemplo en donde la estrategia usada en el algoritmo genera ambigüedad.

REDUCCIÓN DE CÓMPUTO

Elaborado por: [illegible]
Fecha: [illegible]
Versión: [illegible]
Página: [illegible]

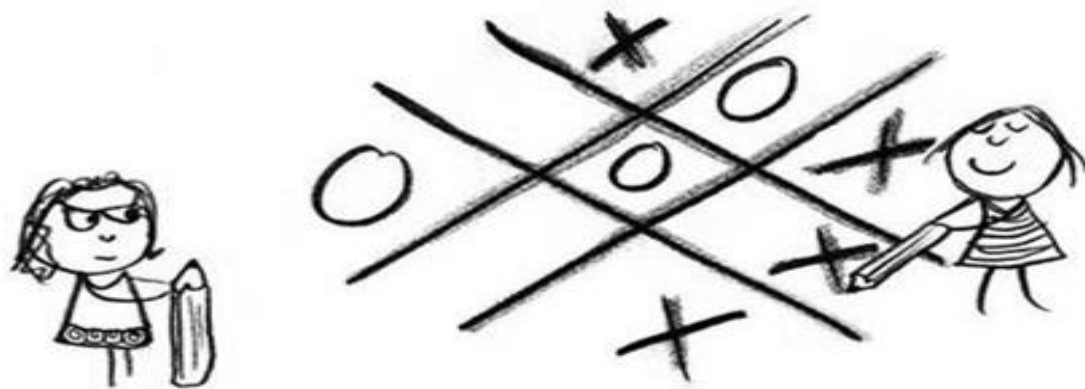




Gracias por su atención y por su
colaboración en el desarrollo de
este proyecto. Esperamos que
sea de su interés y de su utilidad.

GRACIAS

HAPPINESS IS



...winning at tic-tac-toe.