**Árboles de Juegos**

Instructor: Fabricio Mendoza Clase 5
Fecha: 14/03/2020

---

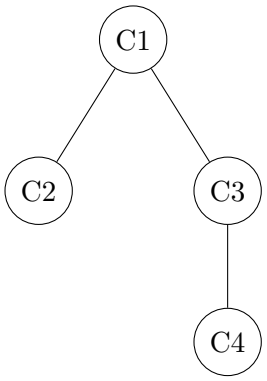
En este capítulo veremos como los juegos de mesa, tales como *Ajedrez, Tic Tac Toe o InpheXion* pueden ser resueltos por programas de computadoras. En particular veremos las *búsqueda minimax*, un método básico que ayuda a decidir que movimiento hacer en cada jugada. Además también veremos el método de corte *alpha-beta*, este método ayuda a descartar opciones *perdedoras* antes de analizarlas.

# 1 Representación de los Juegos

Los juegos de mesa están constituidos por tres piezas:

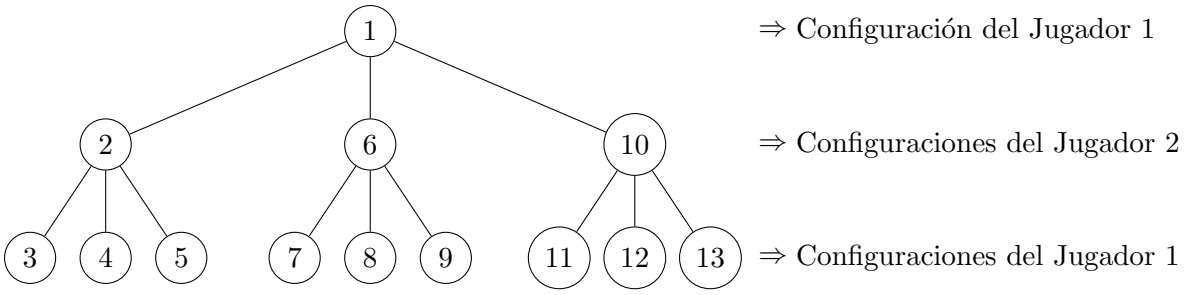
- Un tablero común a los dos jugadores,
- Un conjunto de piezas para cada jugador,
- Un conjunto de reglas mediante las cuales los jugadores mueven las piezas.

Las posiciones de las piezas en el tablero definen la configuración del tablero. El tablero tiene una configuración inicial y dependiendo del juego varias configuraciones finales. El movimiento de una pieza por parte de un jugador lleva de una configuración a otra en el tablero. Así, en cada turno el tablero ofrece al jugador distintas configuraciones posibles a las cuales acceder. Esto puede representarse computacionalmente como un árbol donde la raíz es la configuración inicial (o actual) del juego:




Ejemplo 1: Instancias de un juego donde C1 es la configuración actual y las demás son configuraciones que dependen de los movimientos en C1

Donde los nodos representan las configuraciones y las ramas indican como se pasa de una configuración a otra mediante el movimiento de una pieza. Cada nivel del árbol contiene las configuraciones posibles a las cuales puede acceder solo uno de los dos jugadores, como se observa en el siguiente ejemplo: Notar que, en el Ejemplo 2, el jugador 1 es el que debe decidir que movimiento realizar, por ende los demás movimientos son simulaciones de los movimientos que podrá realizar el jugador 2 y las respuestas posibles del jugador 1. Mientras más simulaciones de movimientos haga el jugador 1, mayor será la profundidad del árbol.




Ejemplo 2: Los niveles del árbol representan turnos de los jugadores

## 2 El procedimiento Minimax

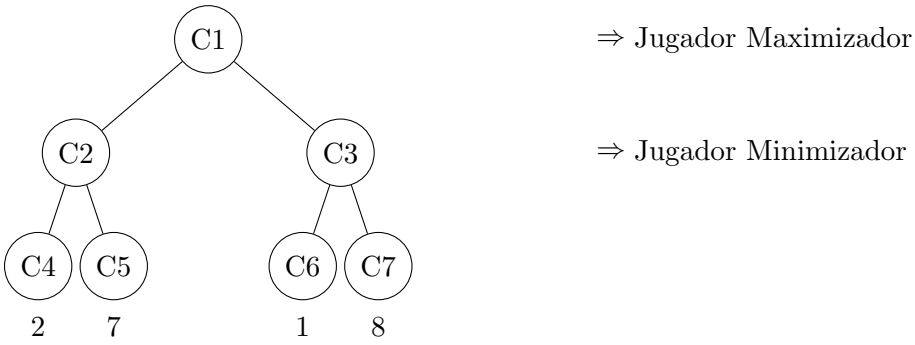
Dado que cada jugador desea alcanzar una configuración que le de ventaja sobre el oponente, algunas ramas representan mejores *jugadas* que otras. Para poder ejecutar el procedimiento minimax debemos suponer que todas las sucesiones de movimientos que llevan a una hoja del árbol tienen asignadas un valor númerico en la hoja del árbol. De esta manera podemos distinguir que jugadas son mejores que otras. Los valores obtenidos en cada camino que se construye desde la raíz hasta una hoja es llamado **puntaje de evaluación estática**.

Los jugadores se dividen en dos tipos:

- **Jugador maximizante o maximizador**, busca obtener los puntajes de evaluación estática más altos entre las distintos caminos que toma la computación.
- **Jugador minimizante o minimizador**, busca obtener los puntajes de evaluación estatica más bajos.

**Nota:** Debemos tener en cuenta que los niveles del árbol indican los turnos de un jugador, por tanto cuando el jugador maximizante deba tomar una decisión debe tener en cuenta que en el nivel de arriba el jugador oponente buscara minimizar el puntaje obtenido en el nivel actual.

Veamos el siguiente ejemplo:




Ejemplo 3: Observemos que el mayor valor al que puede acceder el jugador maximizante es 8, sin embargo la mejor jugada lo lleva a un valor de 2

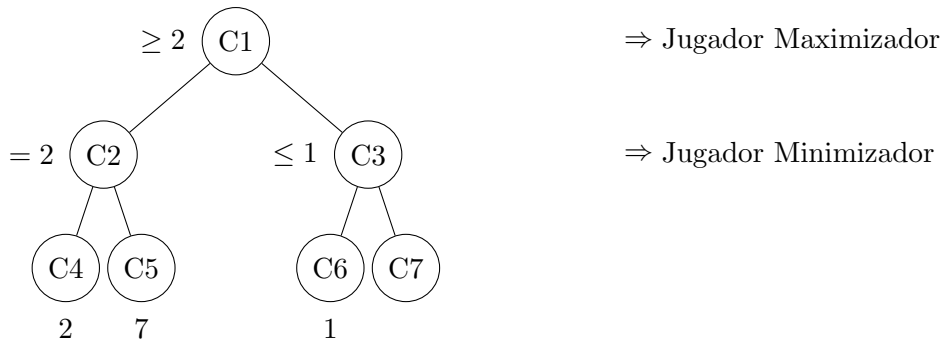
Supongamos que los puntajes estáticos ya han sido evaluados y están abajo de los nodos hoja. El jugador maximizador tiene como mejor jugada realizar un movimiento para pasar

a una configuración $C3$ y esperar que el jugador minimizante juegue la configuración $C7$. Pero el jugador minimizante tiene como mejor opción la configuración $C6$ que le retorna un puntaje estático de 1. Realizando el mismo análisis para una jugada desde la rama derecha el jugador maximizante obtiene un mejor puntaje jugando $C2$ dado que su oponente jugará $C5$. Así el jugador maximizante obtiene un 2 en lugar de 1. El procedimiento minimax es un procedimiento recursivo que visita todos los nodos hijos de un nodo en particular. Una vez que obtiene todos los puntajes de los nodos hay el máximo o mínimo valor dependiendo de que jugador este simulando, luego retorna ese valor un nivel arriba en la llamada recursiva. Esto se puede resumir en el siguiente algoritmo genérico:

- Si el limite de la búsqueda ha sido alcanzado, se computa el valor estático de la posición actual relativa al jugador apropiado.
- Sino, si es un nivel minimizante, llamar a **minimax** sobre los nodos hijos del nodo actual. Retornar el valor mínimo de los resultados.
- Sino, si es un nivel maximizante, llamar a **minimax** sobre los nodos hijos del nodo actual. Retornar el valor máximo.

## 3 Corte Alpha-Beta

Consideremos ahora, un análisis más minucioso sobre el árbol de juegos anterior. Supongamos que el jugador de a raíz (C1) decide ir por la rama izquierda. Luego, el minimizador del nivel 2 computa el mínimo sobre dos valores estáticos, el resultado es 2. Este resultado implica que el maximizador del nivel 1 (raiz) solo se le interesan valores estáticos al menos tan grandes como 2. Ahora, supongamos que se recorre el camino de la derecha y luego el minimizador del nivel 2 computa el valor estático 1 del nodo C6. Como es un minimizador, los valores que le interesan de los siguientes nodos deben ser menores o iguales a 1. Luego, como al maximizador del nivel 1 (raíz) solo le interesan los valores estáticos mayores o iguales a 2 no hace falta computar los demás valores estáticos de los hijos siguientes (en este caso C7). Esto se puede observar en la siguiente figura




Ejemplo 4: En este caso tenemos que la mejor opción explorada por el jugador maximizante es $\alpha = 2$ y la mejor opción explorada por el jugador minimizante es $\beta = 1$

El procedimiento de corte *alpha-beta* se encarga de cortar ramas del árbol que los suficientemente malas para ser examinadas. En el contexto de juegos el principio de alpha-beta especifica que *si se descubre algún hecho acerca del nodo actual, se debe verificar que se*

*sabe acerca del nodo ancestro.* El valor $\alpha$ (alpha) es la mejor opción que ha sido explorada por el jugador maximizante. El valor $\beta$ (beta) es la mejor opción que ha sido explorada por el jugador minimizante.

En general, el procedimiento *alpha-beta* se caracteriza por el siguiente algoritmo:

- Si el nivel del árbol es el más alto, iniciar $\alpha = -\infty$ y $\beta = \infty$.
- Si ha sido alcanzado el limite de la búsqueda, computar el valor estático de la posición actual relativa al jugador apropiado. Retornar el resultado.
- Si el nivel es un nivel minimizante,
  - Realizar hasta que todos los nodos hijos hayan sido visitados o $\alpha \geq \beta$,
    - Llamar al procedimiento alpha-beta, con los valores actuales de $\alpha$ y $\beta$, sobre el nodo hijo actual, almacenar el valor reportado,
    - Comparar el valor reportado con el valor de $\beta$, si el valor reportado es menor, almacenar en $\beta$ dicho valor
  - Retornar $\beta$.
- Sino, el nivel es maximizante,
  - Realizar hasta que todos los nodos hijos hayan sido visitados o $\alpha \geq \beta$
    - Llamar al procedimiento alpha-beta, con los valores actuales de $\alpha$ y $\beta$, sobre el nodo hijo actual, almacenar el valor reportado,
    - Comparar el valor reportado con el valor de $\alpha$, si el valor reportado es mayor, almacenar en $\alpha$ dicho valor
  - Retornar $\alpha$.

Ahora bien, analicemos que sucede cuando estamos en un nodo de un nivel minimizante. Sabemos que $\beta$ es la mejor opción hasta ese momento para el jugador minimizador. Pero si al realizar las evaluaciones sobre sus nodos hijos (maximizantes) se encuentra un valor menor a $\beta$ este se debe actualizar. De la misma manera ocurre cuando el nodo esta en un nivel maximizante.