Departamento de Computación
FCEFQyN, Universidad Nacional de Río Cuarto
Asignaturas: Diseño de Algoritmos - Algoritmos II
Primer Cuatrimestre de 2025

Guía Práctica No. 6: Búsqueda con Adversario

Esta guía práctica corresponde a las Técnicas de Búsqueda con Adversario.
Esta práctica no tiene entrega formal. Su resolución es opcional.

1. Utilizando mecanismos de reutilización propios de la programación orientada a objetos, realice una implementación de min max para problemas de búsqueda con adversarios. La implementación debe permitir incorporar problemas de búsqueda con adversarios concretos con facilidad, sobre los cuales el algoritmo definido pueda utilizarse sin realizar modificaciones. La implementación debe tener en cuenta una profundidad máxima para la construcción del árbol de juego, de manera tal que si ésta es alcanzada y aún no se consiguió una hoja, permita valorar los estados del último nivel, y así completar la valoración de los nodos del árbol. La técnicade búsqueda, aplicada a un problema particular en un estado particular, debe indicar cuál es la mejor forma de "jugar" en la movida corriente.

2. Desarrolle una aplicación que permita jugar al conocido juego del *Ta-Te-Ti*, instancie la solución genérica de búsqueda Min Max, definida en el ejercicio previo, para conseguir estrategias óptimas de juego.

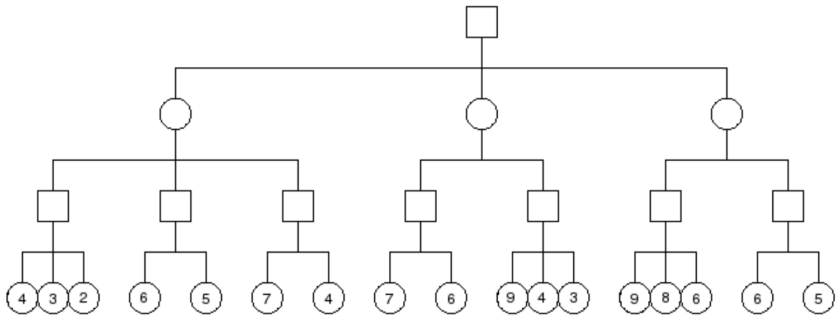
3. El algoritmo de búsqueda adversaria MinMax puede mejorarse significativamente utilizando poda alfa-beta. Dé dos ejemplos de árboles de juego, en uno de los cuales la poda alfa-beta mejore significativamente a MinMax podando al menos dos tercios de los nodos del árbol, mientras que en el otro no produzca ninguna poda.

4. Utilizando mecanismos de reutilización propios de la programación orientada a objetos, realice una implementación de *min max con poda alfa-beta* para problemas de búsqueda con adversarios. La implementación debe permitir incorporar problemas de búsqueda con adversarios concretos con facilidad, sobre los cuales el algoritmo definido pueda utilizarse sin realizar modificaciones. La implementación debe tener en cuenta una profundidad máxima para la construcción del árbol de juego, de manera tal que si ésta es alcanzada y aún no se consiguió una hoja, permita valorar los estados del último nivel, y así completar la valoración de los nodos del árbol. La técnica de búsqueda, aplicada a un problema particular en un estado particular, debe indicar cuál es la mejor forma de "jugar" en la movida corriente.

5. El *Duidoku* es una versión de Sudoku de dos jugadores. En este juego, los jugadores alternan movimientos, que consisten en posicionar valores en el tablero de manera tal de no generar conflictos en el mismo. Para un tablero clásico de 9 x 9, los conflictos son los usuales de Sudoku: no puede haber valores repetidos en una misma fila, columna, o región de 3 x 3; y los valores permitidos en el tablero son enteros entre 1 y 9. Gana el juego aquel jugador que es capaz de realizar el último movimiento en el mismo, es decir, aquel que fuerza que su adversario no pueda colocar ningún valor en el tablero.
   Como referencia, puede considerar la versión online del juego en `http://www.duidoku.com`.
   Se requiere implementar un programa que juegue automáticamente a *Duidoku* contra un jugador humano, en un tablero clásico de 9 x 9. La solución debe implementarse utilizando la técnica *Minmax* con *poda alfa-beta*.

6. Dé dos ejemplos de árboles de juego, en uno de los cuales la poda alfa-beta mejore significativamente a MinMax podando al menos dos tercios de los nodos del árbol, mientras que en el otro no produzca ninguna poda.

7. Considere el siguiente árbol de juego, en el cual los cuadrados son nodos max y los círculos son nodos min:



Ejecute sobre este árbol el algoritmo minimax con poda alfa-beta, indicando qué porciones del árbol se podarían con este algoritmo.