Proyecto – 4

Antes que cualquier cosa: indicarme en un "readme" el ambiente de desarrollo que usaron. Por lo general es obvio, pero igual quiero que me lo digan. El proyecto tiene que compilar PERFECTAMENTE sin que yo tenga que hacer absolutamente nada. Si ese no es el caso, eso influirá en su nota de una manera negativa.

Implementar los algoritmos minimax y alfa-beta para el juego de tic-tac-toe. Como sabemos, tic-tac-toe es un juego clásico de dos jugadores, donde los jugadores tratan de formar tres Xs o Os en columnas, filas o diagonales en un tablero de 3 x 3 cuadrillas. Yo he proporcionado código que tienes que integrar con tus dos algoritmos.

Tu algoritmo calculara el siguiente movimiento utilizando los algoritmos minimax y alfa-beta, dependiendo de la configuración del código adjunto. La convención será que el jugador jugando X será el jugador "Max" y el jugador O será el jugador "Min".

Para la implementación de los algoritmos minimax y alfa-beta, utilizar estrictamente los seudocódigos de minimax y alfa-beta presentados en nuestro texto:

```
function MINIMAX-DECISION(state) returns an action
  return arg max_{a ∈ ACTIONS(s)} MIN-VALUE(RESULT(state, a))
```

```
function MAX-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for each a in ACTIONS(state) do
    v ← MAX(v, MIN-VALUE(RESULT(s, a)))
  return v
```

```
function MIN-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← ∞
  for each a in ACTIONS(state) do
    v ← MIN(v, MAX-VALUE(RESULT(s, a)))
  return v
```

Figure 5.3 An algorithm for calculating minimax decisions. It returns the action corresponding to the best possible move, that is, the move that leads to the outcome with the best utility, under the assumption that the opponent plays to minimize utility. The functions MAX-VALUE and MIN-VALUE go through the whole game tree, all the way to the leaves, to determine the backed-up value of a state. The notation $\text{argmax}_{a \in S} f(a)$ computes the element a of set S that has the maximum value of $f(a)$.

```
function ALPHA-BETA-SEARCH(state) returns an action
  v ← MAX-VALUE(state, −∞, +∞)
  return the action in ACTIONS(state) with value v
```

```
function MAX-VALUE(state, α, β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for each a in ACTIONS(state) do
    v ← MAX(v, MIN-VALUE(RESULT(s,a), α, β))
    if v ≥ β then return v
    α ← MAX(α, v)
  return v
```

```
function MIN-VALUE(state, α, β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← +∞
  for each a in ACTIONS(state) do
    v ← MIN(v, MAX-VALUE(RESULT(s,a) , α, β))
    if v ≤ α then return v
    β ← MIN(β, v)
  return v
```

Figure 5.7 The alpha–beta search algorithm. Notice that these routines are the same as the MINIMAX functions in Figure 5.3, except for the two lines in each of MIN-VALUE and MAX-VALUE that maintain α and β (and the bookkeeping to pass these parameters along).

El juego de tic-tac-toe es lo bastante pequeño en su alcance para poder generar todo el árbol del juego cuando hacemos la búsqueda de minimax. El valor terminal cuando gana Max es de 1 y cuando gana Min es de -1, y en caso de tablas es de 0.

Como parte de los materiales del proyecto, adjunto código escrito en Java que genera el interfaz grafico adecuado. Tus algoritmos van a ser integrados a este código. El programa de interfaz grafico incluido requiere de dos argumentos, el primer argumento define al jugador que mueve primero, y el segundo argumento define al jugador que mueve segundo. Los argumentos posibles son:

(1) "human"
(2) "random"
(3) "minimax"
(4) “alfa-beta”

Nota: cualquiera de los argumentos puede ser el primer o segundo argumento del programa.

Obviamente, los jugadores "random", "minimax" y "alfa-beta" corresponden a la computadora. El jugador "random" ya está implementado, y los jugadores "minimax" y "alphabeta" son los jugadores que implementaras tú. Analizar cómo esta implementado el jugador computarizado "random", para saber cómo será integrada tu implementación de "minimax" y "alfa-beta".