PRÁCTICA 4 DE IA: TORNEO DE MANCALA

Fecha de publicación: 04 de Abril de 2018
Fechas de entrega:
Parte 0 (entrenamiento, opcional): [Todos los grupos: **del 2018/04/09 al 2018/04/18**]
No es posible garantizar que el torneo pueda ser actualizado en vacaciones o festivos.
Parte A (torneo): [Todos los grupos: **del 2018/04/19 al 2018/05/03**]
No es posible garantizar que el torneo pueda ser actualizado en vacaciones o festivos.
Parte B (memoria): [grupos jueves: **miércoles, 2018/05/03, 23:55**]
[grupos viernes: **jueves, 2018/05/04, 23:55**]

Índice del enunciado de la práctica 4:

IMPORTANTE: Para esta práctica podéis usar la versión **Allegro CL 6.2 ANSI with IDE** o la versión **Free Express actual** o **SBCL**.

Forma de entrega: En general, según lo dispuesto en las normas de entrega, accesibles en Moodle (en Normativa de prácticas), se detalla más información en la sección A.3 del enunciado.

PARTE A (5 puntos)

A.1: Funcionamiento del juego

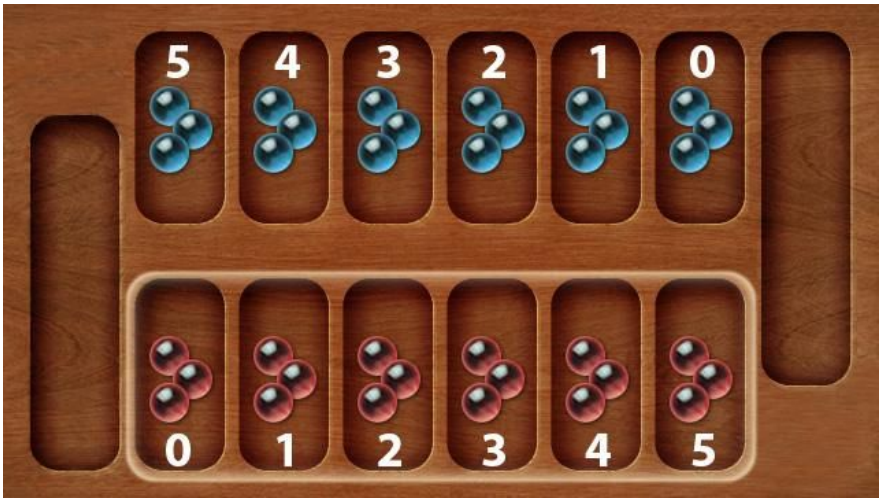
Introducción

En esta parte de la práctica se pide implementar un jugador para una variante de la familia de juegos mancala [1]. En este juego dos jugadores realizan movimientos en turnos alternados. En cada uno de ellos el jugador redistribuye (siembra) las piezas (semillas) de un hoyo con el objetivo de capturar las piezas de su oponente. El tablero del juego está formado por 2 filas de 6 hoyos cada una. Una fila pertenece a un jugador y la otra a su oponente. El número total de semillas en el tablero permanece constante, y cada jugador empieza sembrando en su fila, pero, tras pasar el depósito lateral llamado mancala o kalaha, puede acabar sembrando en los hoyos del contrario. Las semillas capturadas del otro jugador se depositan únicamente en el kalaha. Antes de programar los contenidos de la práctica, se recomienda familiarizarse con la dinámica del mancala, para lo que se proporciona en un anexo el tablero de mancala.

Elementos del juego

Tablero

En nuestra variante **mancala11**, el tablero está compuesto por 2 filas de 6 hoyos cada una numerados de 0 a 5, más un depósito o kalaha. Cada fila pertenece a un jugador. Los valores de cada fila indican el número de semillas que el jugador tiene en el hoyo correspondiente. La disposición inicial del juego es la siguiente:



En la disposición inicial, cada jugador parte con 3 semillas en cada hoyo, lo que hace un total de 18 semillas para cada uno.

[1] Juegos de este tipo (incluyendo múltiples variantes) son de los más antiguos conocidos. Actualmente son más populares en África y Asia pero pueden encontrarse tableros parecidos excavados en suelos y escaleras de templos y mercados egipcios, griegos y romanos.

Semillas

Son las piezas que se depositan en los hoyos. Todas tienen el mismo valor.

Dinámica del juego

El juego consiste en recolocar o redistribuir las semillas de un hoyo en otros de la misma fila con el objetivo de capturar el mayor número posible de semillas del oponente. El juego termina cuando uno de los dos jugadores queda sin semillas en su fila (en tal caso el ganador de la partida es quien tiene más semillas), pero puede terminar cuando se dan otras circunstancias que se detallan más adelante.

Reglas del juego

Las acciones que puede realizar un jugador en su turno son las siguientes:

- ***Siembra***: se selecciona un hoyo que contenga semillas y un sentido de siembra y se siembran las semillas del hoyo seleccionado en los sucesivos hoyos. El sentido de siembra es elegido por el jugador en cada jugada. Se denomina a derechas (siembra-R) la que distribuye las fichas de izquierda a derecha, relativas al jugador situado frente a su fila, y depositando una semilla en cada hoyo. Se denomina siembra a izquierdas (siembra-L) al caso contrario. En la versión actual sólo se permite la siembra a derechas. En el caso de que se llegara sembrando al final de la fila (hoyo 5) se continúa sembrando en el mismo sentido, primero en el kalaha y luego, en el sentido contrario a las agujas del reloj, en los hoyos 0, 1, 2… del contrario. La siguiente secuencia muestra cómo, desde la situación inicial de partida, el jugador 1 seleccionó el hoyo 1 y sentido R para la siembra. De esta forma, depositó una semilla en cada uno de los hoyos a su derecha (2, 3 y 4). Se indica en rojo la situación en el hoyo con la última semilla.

Tablero inicial indicando los sentidos de juego para cada jugador. El asterisco indica el turno (lado del tablero al que le corresponde jugar):

```
     ←R  L→
    5  4  3  2  1  0
   ----------------
    3  3  3  3  3  3      18    Lado 1 (Jugador 2)
 0                        0
*   3  3  3  3  3  3      18    Lado 0  (Jugador 1)
   ----------------
    0  1  2  3  4  5
     ←L  R→
```

Tablero después de la siembra inicial del jugador 1 (nótese que el turno es ahora del jugador 2):

```
    5  4  3  2  1  0
   ----------------
*   3  3  3  3  3  3      18
 0                        0
    3  0  4  4  4  3      18
   ----------------
    0  1  2  3  4  5
```

- ***Captura/Entrega/Robo***: se da sólo cuando tras la siembra, la última semilla se deposita en un hoyo (hoyo de última siembra) con ciertas características. Se pueden dar las siguientes alternativas:

1. ***Captura semillas***: Si la siembra termina en un hoyo que no sea el kalaha y el hoyo del contrario contiene semillas, se capturan las semillas del hoyo opuesto y se depositan, junto con la propia, en el kalaha del jugador (el de la derecha).

2. ***Repetición turno***. Si la siembra termina en el kalaha propio, no se captura, pero se juega de nuevo (salvo si estuviera vacío, en cuyo caso se cede el turno). Tras la nueva siembra volverá a analizarse el nuevo hoyo de última siembra y en su caso se repetiría todo el proceso hasta que finalicen las capturas o se cambie de turno.

3. **Robo de semillas**: Si la siembra termina en un hoyo contrario (que no sea el kalaha) con 5 o más semillas (antes de la siembra), y si el propio opuesto no tuviera ninguna, el jugador, llevado de su avaricia, ignora el procedimiento habitual y se lleva a su propio hoyo todas las semillas del hoyo del contrario antes de cederle el turno.

4. ***Cambio de turno***: En cualquier otro caso, se termina la jugada y se pasa el turno al oponente.

A continuación, se expone un ejemplo que ilustra el proceso. Gracias a unas líneas descomentadas al final del código entregado, este ejemplo arranca automáticamente (en modo de máxima verbosidad) al ejecutarlo.

Desarrollo del juego

Dado el siguiente tablero, en el que el jugador 1 tiene el turno:

```
   5  4  3  2  1  0
   ----------------
   0  3  5  2  1  2     18  Ac:  0  Jugador 2
 5                    7
*  1  0  1  3  2  4     18  Ac:  0  Jugador 1
   ----------------
   0  1  2  3  4  5
```

decide jugar desde el hoyo 4, que tiene dos semillas. Para especificar su jugada introduce (R 4) o simplemente 4 en modo abreviado. Las semillas que había en 4 se depositarán en el hoyo 5 y en su kalaha, resultando el tablero:

```
   5  4  3  2  1  0
   ----------------
   0  3  5  2  1  2     18  Ac:  0  Jugador 2
 5                    8
*  1  0  1  3  0  5     18  Ac:  0  Jugador 1
   ----------------
   0  1  2  3  4  5
```

Como la última semilla ha sido depositada en su kalaha, el jugador 1 continúa jugando. Ahora elige el hoyo 3, quedando el tablero como se muestra a continuación:

```
   5  4  3  2  1  0
   ----------------
   0  3  5  2  1  2     18  Ac:  0  Jugador 2
 5                    9
*  1  0  1  0  1  6     18  Ac:  0  Jugador 1
   ----------------
   0  1  2  3  4  5
```

Al igual que en la jugada anterior, como la última semilla ha caído nuevamente en el kalaha del jugador 1, sigue jugando. Elige el hoyo 5, que contiene 6 semillas. Eso le hace perder 5 fichas como indica el marcador, pero nótese que el marcador de acumulado no varía puesto que no hay captura.

```
   5  4  3  2  1  0
   ----------------
*  0  4  6  3  2  3     23  Ac:  0  Jugador 2
 5                    10
   1  0  1  0  1  0     13  Ac:  0  Jugador 1
   ----------------
   0  1  2  3  4  5
```

Ahora es el turno del jugador 2. Dicho jugador ve oportunidad de captura, por lo que juega 2. Tras sembrar las 3 semillas de dicho hoyo, el tablero quedaría (previo a la captura):

```
   5  4  3  2  1  0
   ----------------
*  0  5  7  0  2  3     23  Ac:  0  Jugador 2
 7                    10
   0  0  1  0  1  0     13  Ac:  0  Jugador 1
   ----------------
   0  1  2  3  4  5
```

Y así sucesivamente hasta que algún jugador quede sin fichas (fin de juego).

- ***Fin del juego***: El juego termina cuando uno de los dos jugadores se queda sin semillas en sus hoyos. El ganador será quién más semillas tenga, sumando las de sus hoyos y las de su kalaha. Si ambos tienen el mismo número el resultado es tablas. También se dará el juego por tablas si el jugador posicionalmente más fuerte no consigue vencer en un número suficientemente grande de jugadas.

Mancala como Juego de Suma Cero: Minimax y Negamax

Juegos de suma constante son aquellos en que la suma de las ganancias de los jugadores en cada posible situación es siempre la misma. Un caso particular de juegos de suma constante son los juegos de suma cero.

Para el caso de 2 jugadores, en juegos de suma cero toda ganancia de un jugador es a costa de una pérdida idéntica del contrario, lo cual excluye la posibilidad de colaboración. El término juego se usa aquí en su sentido académico, es decir, describe una situación en la que, a diferencia de cuando se compite contra el azar o contra la naturaleza, los contrincantes deben tomar decisiones sabiendo que el contrario tiene una estrategia cuyo objetivo es maximizar su ganancia, la cual se produce a costa del sacrificio propio. Esta situación se da no sólo en los juegos de mesa, sino también en conflictos sociales o bélicos. Ejemplos de juegos de suma constante (pero no cero): distribución de un botín entre ladrones, reparto de un segmento de mercado entre empresas competidoras, reparto del territorio entre tribus o naciones rivales. Ejemplos de juegos de suma cero: ajedrez, damas, mancala.

Puede demostrarse que para los juegos de suma constante existe al menos una estrategia óptima, es decir, un método de decisión que minimiza para ambos contrincantes la máxima pérdida esperable. Este es el origen del algoritmo minimax. Para el caso de juegos de suma nula dicho algoritmo se simplifica notablemente, pues, debido a la simetría entre beneficios y pérdidas, el beneficio de un jugador frente a cada posible situación es simplemente el opuesto del beneficio del contrario. La variante de minimax que utiliza esta simplificación se denomina negamax, y es la versión implementada en el código que se entrega.

Como parte del código del juego se proporciona al alumno una implementación del algoritmo minimax, en su versión negamax, que inicia la búsqueda y devuelve el siguiente estado elegido por el jugador que tiene el turno.

A.2: Implementación de la heurística.

Función de evaluación del alumno

La función de evaluación, aparte de la función de búsqueda (que es la misma para todos), es la parte más importante y crítica para conseguir un buen jugador automático. **Esta función debe tomar como argumento el estado de la partida (disposición del tablero y jugador que tiene el turno) y devolver un valor numérico que cuantifique la confianza en que, desde esa posición, el jugador en posesión del turno gane la partida**. En general, la evaluación se debe hacer sin generar sucesores, del mismo modo que si se pidiera construir un sensor que estimara la distancia a un objetivo, no sería aceptable tener que viajar hasta el objetivo para conocer la distancia.

Si en la función de evaluación se generarán de manera sistemática los sucesores, el jugador diseñado sería igual o peor que un jugador cuya función de evaluación esté basada únicamente en el estado de la partida, pero que utilizara un nivel de búsqueda más profundo. En ocasiones, cuando el estado de la partida es tal que en el tablero hay situaciones no resueltas (por ejemplo, en ajedrez cuando hay una secuencia incompleta de sacrificios) se podrían generar de manera selectiva algunos estados sucesores que sean relevantes para la resolución de dichas situaciones. Sin embargo, la generación indiscriminada de sucesores es una estrategia no recomendable pues consume una cantidad exponencial de computación.

Para el diseño de funciones de evaluación, además de consultar el libro de Russell & Norvig, se espera que el alumno haga una labor de investigación bibliográfica. Como punto de partida, se deben leer los artículos (para acceder al texto de los artículos, se puede pinchar sobre los enlaces desde un ordenador en la UAM, o activando el acceso remoto a la UAM mediante una sesión VPN):

A. M. Turing (1950), "Computing machinery and intelligence". Mind, Vol. 59, No. 236, pp. 433-460

A. L. Samuel (1959), "Some Studies in Machine Learning Using the Game of Checkers". IBM Journal of Research and Development, 3(3), pp. 210–229

A. L. Samuel (1967), "Some Studies in Machine Learning Using the Game of Checkers II. Recent Progress". IBM Journal of Research and Development, 11(6), pp. 601-617

Para la investigación bibliográfica, se debe evitar la utilización de fuentes derivadas (la mayoría de las que se encuentran en Internet). La web puede ser un buen punto de partida para localizar las fuentes originales. Una vez identificadas dichas fuentes, se pueden utilizar los recursos de la biblioteca y la hemeroteca de la EPS (libros, artículos de revista en papel o electrónicos).

Una vez familiarizado con el juego, el alumno debe implementar su propia función de evaluación. Para ello se puede probar la calidad de dicha función contra otras funciones alternativas o contra las funciones de evaluación de otras parejas. A tal efecto, al final del código suministrado se incluyen tres funciones de evaluación: aleatoria, regular y buena. Se pide implementar una función de evaluación que al menos iguale o supere, con el saque tanto a favor como en contra, a la `f-eval-Regular`. No se deben enviar funciones al torneo que no cumplan este requisito. La función de evaluación debe tener la siguiente cabecera:

```
(defun mi-f-ev (estado) …)
```

Adicionalmente a `f-eval-Regular`, que sería correspondiente a un jugador regular, se proporcionan uno bueno y uno aleatorio, contra los que el alumno podrá compararse.

El jugador *aleatorio* podría ganar en un caso muy improbable, pero a la larga su puntuación será inferior al resto de jugadores. El jugador *regular* hace una simple estimación del estado a base de calcular las fichas que puede capturar. Es una estrategia a muy corto plazo y se espera que el alumno le consiga ganar con facilidad. Es bastante más difícil ganar al jugador llamado "*bueno*", aunque su estrategia es la misma que la del jugador regular. Su ventaja radica en que expande un nivel el estado en curso, por lo que cuenta con una visión de más alcance sobre la partida, sin embargo, sigue siendo una estrategia pobre y a costa de un tiempo de cálculo considerablemente mayor.

A fin de implementar la función de evaluación se facilitan un conjunto de funciones útiles especificadas en el apartado siguiente. Las funciones utilizadas por cada jugador son supervisadas antes de ponerlas en funcionamiento. **No se permite el uso de funciones que pretendan modificar la mecánica del juego, ni de `setq`/`setf`, ni de funciones destructivas**. Por otra parte, en la segunda parte de la práctica también se tendrá en cuenta la eficiencia de las mismas, es decir, dos funciones que obtengan resultados similares en el juego dentro del límite de tiempo establecido pueden obtener puntuaciones distintas en la segunda parte.

Construcción del jugador

A fin de probar su función de evaluación el alumno deberá crear un jugador que utilice el algoritmo para la búsqueda suministrado con el código y una función de evaluación para la evaluación de los estados. El nombre puede ser cualquier alias que se quiera dar al jugador. **Esta estructura no debe ser entregada,** es sólo a efectos de pruebas personales del alumno.

```
(setf *mi-jugador* (make-jugador
                            :nombre       'Mi-Jugador
                            :f-juego      #'negamax
                            :f-eval       #'mi-f-ev))
```

La expansión de nodos es una **práctica no recomendable** (por los motivos anteriormente explicados).

En esta práctica ya no aplican algunas de las restricciones sobre programación en Lisp de prácticas anteriores, pero sigue estando **prohibido utilizar funciones destructivas, macros, modificación de variables, acceso al sistema operativo, etc.**

Funciones a disposición del alumno

Se listan en este apartado las funciones que el alumno tiene disponible para realizar la heurística:

`(cuenta-fichas tablero lado desde)`
Devuelve la suma de las fichas que hay en la zona del tablero de un jugador desde la posición **`desde`** hasta el final de la fila. Obviamente si **`desde`**=0 cuenta todas las fichas.

`(get-fichas tablero lado posicion)`
Devuelve el número de fichas en determinado hoyo del lado del tablero de un jugador

`(list-lado estado lado)`
Devuelve una lista con el estado de un lado del tablero en orden inverso

`(posiciones-con-fichas-lado tablero lado desde)`

Devuelve la lista de posiciones, para un jugador, que tienen alguna ficha

(lado-contrario lado)
Devuelve el jugador oponente

(estado-tablero estado)
Devuelve el tablero como array de 2 x 7 dimensiones.

(estado-lado-sgte-jugador estado)
Devuelve el lado del tablero a cuyo jugador le corresponde mover.

(estado-debe-pasar-turno estado)
Devuelve T si el siguiente jugador debe pasar turno, porque el otro realizó una acción que le permite volver a mover. Se usa sólo en versiones de mancala que lo permitan.

(juego-terminado-p estado)
Devuelve T si el estado es de fin de partida.

(get-pts i)
Devuelve los puntos del marcador del jugador i en este momento.

(get-tot i)
Devuelve el acumulador total del jugador i en la partida en curso.

(reset-tablero-aux &opcional x)
Inicializa o reinicializa a x el tablero auxiliar `*tablero-aux*`. Uso discrecional.

(put-tablero-aux i j k)
Pone la celda (i,j) de `*tablero-aux*` a k. Uso discrecional.

Las dos últimas funciones utilizan la variable global `*tablero-aux*`, que contiene una estructura semejante al tablero (*array* de 2 x *N* elementos) de uso discrecional por el alumno. El único acceso permitido en escritura a dicha estructura es a través de las dos mencionadas funciones. El acceso en lectura es mediante la habitual `aref` (pueden verse ejemplos de uso en la ayuda o en varias de las funciones entregadas).

Limitaciones en la implementación de la heurística

No se admitirán jugadores que usen funciones de control del juego o manipulen sus estructuras de datos, produciéndose la exclusión del juego a la pareja que lo intente (aparecerán en la clasificación como ***proscritos***). A diferencia de los ***descalificados***, que pueden seguir jugando si solucionan el motivo de su descalificación, los proscritos no podrán enviar más jugadores y los que tengan serán eliminados.

La duración máxima de una jugada está limitada, pero no se especifica de antemano, pues la duración máxima aceptable se irá disminuyendo según aumente la calidad de los jugadores. En todo caso, siempre será inferior al 50 % del tiempo que precise el jugador bueno en evaluar una posición a profundidad 4. Nótese que el baremo de cálculo no es un tiempo absoluto (que depende, además de la máquina, de diversos factores, entre ellos del SO utilizado y de su granularidad) sino el tiempo que requiere el jugador bueno, lo cual sí es reproducible en cualquier máquina y permite al alumno saber de antemano a qué atenerse[2]. Es decir, **jugadores que requieran tiempos superiores al 50 % del llamado bueno a profundidad 4 serán muy**

[2] En un ordenador mediano de sobremesa equivale aproximadamente a 0.1s por jugada con código compilado o aproximadamente 10s con código interpretado.

probablemente eliminados por *timeout*. No es posible asegurar que siempre serán eliminados, porque la duración de una evaluación depende también del n.º de movimientos posibles, lo que introduce cierta variabilidad. Para evitar riesgo de eliminación se recomienda no superar el 40 % de dicho máximo.

La variable global `*timeout*` permite experimentar con el tiempo de juego permitido a los jugadores. **El objetivo del alumno es producir la mejor y más rápida estrategia de evaluación** del estado. La expansión del nodo, aunque no está prohibida (la usa el jugador llamado "bueno" por los motivos explicados anteriormente), se desaconseja, pues es una opción sumamente costosa que no aporta nada a la estrategia. El torneo se realizará limitando la evaluación a profundidad 2, pero el alumno tiene la posibilidad de utilizar cualquier otra profundidad para sus pruebas y deberá documentar en la memoria los resultados observados. Pruebas específicas para evaluación del jugador en el apartado C pueden utilizar profundidades mayores.

La siguiente instrucción permite determinar el valor máximo de `*timeout*` para nuestra máquina. Tal cual se entrega realiza pruebas desde 0.5 a 5 s con incrementos de 0.5 s, ajústese a lo necesario en cada caso. Si aparece en pantalla el turno del jugador humano ello indica que para ese t el jugador bueno ha podido terminar su jugada. Compilar el código puede reducir en dos niveles de magnitud (10^{-2}) el tiempo de ejecución, pero las relaciones serán aproximadamente las mismas.

```
(dotimes (i 10)
  (setq *timeout* (* 0.5 (1+ i)))
  (format t "~%----- Probando t = ~D ----- (si turno Humano = pasa la prueba)" *timeout*)
  (partida 1 4 (list *jdr-humano* *jdr-nmx-bueno*)))
```

Inicio de partida forzando una posición para su análisis detallado

El siguiente procedimiento permite reproducir cualquier situación que se desee investigar. El primer comando define una posición para analizar o jugar a partir de ella. El segundo crea el estado correspondiente a la anterior posición. El tercero inicia una partida entre humanos forzando como posición inicial el estado anterior y estableciendo el saque para el primer jugador.

```
(setq mi-posicion (list '(1 0 1 3 2 4 7) (reverse '(5 0 3 5 2 1 2))))
(setq estado (crea-estado-inicial 0 mi-posicion))
(partida 0 2 (list *jdr-humano* *jdr-human2*) mi-posicion)
```

La situación elegida reproduce el tablero discutido anteriormente (pág. 5):

```
     5  4  3  2  1  0
     ----------------
     0  3  5  2  1  2     18  Ac:  0  Jugador 2
  5                    7
*    1  0  1  3  2  4     18  Ac:  0  Jugador 1
     ----------------
     0  1  2  3  4  5

[J 1] El turno es de HUMANO
```

A.3: Formato de la entrega.

Normas de entrega de jugadores

Formato del nombre de archivo

En cada entrega se enviará un único fichero con extensión .cl con nombre en la forma:

grupoGGparejaPPNCRCDDMM.cl

Donde:

- **GG:** dos últimos dígitos del grupo de prácticas (i.e. eliminando el 23).
- **PP:** dos dígitos (con 0 a la izda. si es necesario) del n.º de pareja.
- **N:** 1-3 para distinguir los 3 jugadores permitidos a la pareja.
- **CRC:** Código de 2 cifras que se proporcionará a los alumnos antes de empezar la práctica 3.
- **DDMM:** Día y mes en el que se entrega el fichero.

Los ficheros que no se ajusten exactamente a esta norma serán descartados.

Formato del contenido

No se debe incluir ningún otro tipo de función o ejecución, aunque se admite el uso de comentarios. El formato será como en el siguiente ejemplo: La pareja 2 del grupo 2314, cuya clave secreta es Xf entrega su tercer jugador el 06/04/2018 en el fichero **`grupo14pareja023Xf0604.cl`** con el siguiente contenido:

```
(defpackage :grupo14pareja023Xf0604 ; se declara un paquete lisp que usa common-lisp
  (:use :common-lisp :mancala)      ; y mancala, y exporta la función de evaluación
  (:export :heuristica :*alias*))   ; heurística y un alias para el torneo

(in-package grupo14pareja023Xf0604)

(defun heuristica (estado) …)       ; función de evaluación heurística a implementar

(defvar *alias* '|Campeon|)         ; alias que aparecerá en el ranking

(defun …)                           ; funciones auxiliares usadas por heurística
```

El alias permite a las parejas no revelar su identidad al resto de jugadores. Cada uno de los tres jugadores puede tener un alias distinto, y ese alias puede cambiar en cada entrega. Es responsabilidad del alumno asegurarse de utilizar un alias único. En cualquier caso, el sistema no utiliza internamente el alias sino el identificador de pareja, por lo que aún en caso de alias repetidos el torneo funcionará correctamente, pero la clasificación publicada no permitirá distinguir a un jugador de otro. Lo mismo puede decirse de los diferentes jugadores entregados por una pareja: si tienen el mismo alias, no habrá modo de distinguirlos.

Valoración:

Esta práctica está compuesta de tres partes, cada una de las cuales se debe entregar por separado:

- La parte A (7 puntos) consiste en la implementación y presentación de jugadores a un torneo entre todos los alumnos de la asignatura.
- La parte B (3 puntos) consiste en una memoria respondiendo a preguntas sobre el trabajo realizado.

Como resultado de los enfrentamientos entre los jugadores se publicará una clasificación, accesible mediante un enlace que se mostrará en la página de prácticas. Esta clasificación será actualizada de manera regular con el fin reflejar la incorporación al torneo de nuevos jugadores. En cierto momento se introducirán de manera anónima jugadores suministrados por los profesores, cuyo objetivo es servir de indicadores de nivel de juego. La nota de la parte B (torneo) corresponderá al resultado del mejor de los 3 jugadores presentados por cada pareja en el momento de cierre del torneo en relación con los jugadores de referencia, según el baremo siguiente:

- jugador que supera a 5 de los jugadores de referencia: 4 puntos

- jugador que supera a 4 de los jugadores de referencia: 3 puntos
- jugador que supera a 3 de los jugadores de referencia: 2 puntos
- jugador que supera a 2 de los jugadores de referencia: 1 punto
- jugador que supera a 1 de los jugadores de referencia: 1/2 punto

Se concederá un premio adicional de 1 punto a los 5 jugadores que ocupen las primeras posiciones de la clasificación final.

Para aplicar este baremo se considerará únicamente el mejor situado de entre los jugadores de la pareja que, en el momento de la entrega final esté activo en el torneo. Es obligatoria la participación en el torneo con al menos un jugador. El baremo se refiere sólo a jugadores que sean parte de la clasificación, es decir, las parejas sin jugadores en la clasificación (sea por no haberlos entregado, por haber sido descalificados por errores de forma o compilación, por no reunir los mínimos requisitos de calidad o por cualquier otro motivo), obtendrán 0 puntos.

Habrá un torneo al día. En Moodle se publicará la hora límite para la subida de funciones de evaluación heurística. En principio se habilitan subidas hasta las 9 de la noche, y los resultados se publican a las 9 de la mañana del día siguiente. Se podrán subir hasta tres funciones en cada torneo. Las clasificaciones diarias serán publicadas en Moodle.

Del 9 de abril al 18 de abril se podrán obtener puntos subiendo todos los días funciones: 1 al día = 1 punto, 2 al día = 2 puntos y 3 al día = 3 puntos. Esta regla excluye festivos y fines de semana.

Del 19 de abril hasta el 3 de Mayo (día anterior al torneo final) deberán subirse al menos 1 función al día, en caso contrario se perderá 1 punto de la nota. Esta regla excluye festivos y fines de semana.

Nótese que quien obtenga estas bonificaciones podría obtener una nota de 13 para esta práctica.

En la evaluación de la entrega se tendrán en cuenta la calidad del diseño del jugador entregado, como rapidez, eficacia y eficiencia.

Referencias

En la memoria se debe hacer referencia a las fuentes utilizadas con un formato estándar, como el empleado en este enunciado para citar el trabajo de Turing y Samuel. En caso de que se utilicen textos extraídos de fuentes consultadas, deben aparecer entrecomillados y con una referencia a la fuente (e. g. *"We may hope that machines will eventually compete with men in all purely intellectual fields."* [Turing, 1950]). En cualquier caso, lo que se valora es la comprensión y asimilación de las lecturas, no la mera inclusión de citas. Para distinguir el plagio de la contribución original se valorará la relevancia de los comentarios propios al problema en curso.

PARTE B (3 puntos)

Preguntas sobre el código entregado

Pregunta C1. Explique los fundamentos, el razonamiento seguido, la bibliografía utilizada y las pruebas realizadas para la entrega de la/s función/es de evaluación entregada/s en el torneo.

Pregunta C2. El jugador llamado bueno sólo se distingue del regular en que expande un nivel más. ¿Por qué motivo no es capaz de ganar al regular? Sugiera y pruebe alguna solución que remedie este problema.

Evaluación de la calidad de la Función de Evaluación

Esta parte no requiere contestación específica ya que se realiza sobre el mejor de los jugadores entregados, evaluando características distintas a su mera posición en el torneo. Se valoran características como la rapidez, eficiencia, eficacia y resistencia del jugador.

A modo de ayuda para la realización de esta práctica se recomienda la lectura de:

Russell, S. and Norvig, P. Imperfect Real-Time Decisions. In Artificial Intelligence: A Modern Approach, Chapter 6, pp. 171-175. New Jersey, USA, 2003.

ANEXO A: Partida expuesta paso a paso

A continuación se muestran las primeras jugadas a profundidad 2 entre dos jugadores humanos con las variables `*debug-level*` a nivel 2 y `*verb*`, `*verjugada*` y `*vermarcador*` puestas a `T`. Dichas variables permiten regular la cantidad de salidas: `*verb*` informa sobre la evolución del juego, `*verjugada*` presenta el tablero tras cada movimiento y `*vermarcador*` presenta el marcador. Para todas ellas el valor por defecto es `T`. Además, `*debug-nmx*` (por defecto a `NIL`) permite obtener detalles de la evolución de negamax. Modificando sus valores pueden obtenerse diferentes niveles de detalle. Para experimentar intensivamente con jugadores automáticos de modo más eficiente conviene poner a `NIL` las mencionadas variables. El acumulador tras el marcador indica el acumulado de semillas obtenidas del contrario, podría servir de indicador a largo plazo del desempeño de cada jugador.

```
(partida 0 2 (list *jdr-humano* *jdr-nmx-human2*))

  Juego: (1) Humano vs Human2 (2)
Local game Humano-Human2 depth=2
TABLERO:

   5  4  3  2  1  0
  ------------------
   3  3  3  3  3  3     18  Ac:  0
 0                   0
*  3  3  3  3  3  3     18  Ac:  0
  ------------------
   0  1  2  3  4  5
[J 1] El turno es de Humano

Jugador Humano.
  Elija entre: ((R 0) (R 1) (R 2) (R 3) (R 4) (R 5))
  o en modo abreviado: (0 1 2 3 4 5)
  Introduzca jugada o x para terminar : 3

Juega (R 3)  => Ultima ficha en 6, con 0, contra: 0 => Kalaha contrario o vacio. Cede turno,
[J 1] Humano juega (R 3)
TABLERO:

   5  4  3  2  1  0
  ------------------
*  3  3  3  3  3  3     18  Ac:  0
 0                   1
   3  3  3  0  4  4     18  Ac:  0
  ------------------
   0  1  2  3  4  5
[J 2] El turno es de Human2

Jugador Humano.
  Elija entre: ((R 0) (R 1) (R 2) (R 3) (R 4) (R 5))
  o en modo abreviado: (0 1 2 3 4 5)
  Introduzca jugada o x para terminar : 0

Juega (R 0)  => Ultima ficha en 3, con 3, contra: 3: (18 18)
[J 2] Human2 juega (R 0)
TABLERO:

   5  4  3  2  1  0
  ------------------
   3  3  4  4  4  0     18  Ac:  0
 0                   1
*  3  3  3  0  4  4     18  Ac:  0
  ------------------
   0  1  2  3  4  5
[J 3] El turno es de Humano

Jugador Humano.
  Elija entre: ((R 0) (R 1) (R 2) (R 4) (R 5))
  o en modo abreviado: (0 1 2 4 5)
```

```
  Introduzca jugada o x para terminar : x
1
"Error en func. o abandono"
```

ANEXO B: Tablero de mancala