PRÁCTICA 3 DE IA. TORNEO, JUEGOS Y MINIMAX

Fecha de publicación: 10 de marzo de 2015.
Duración: 2 sesiones de laboratorio.

Forma de entrega: En general, según lo dispuesto en las normas de entrega, accesibles en Moodle, pero en lo que respecta a la parte A, con las excepciones mencionadas en las ***consideraciones adicionales***.

Calendario de Entregas:
Parte A (torneo): del 13-mar, 12:00 al 27-mar, 12:00.
Parte B (código y memoria): 27-mar-15, 23:55h

Valoración:

Esta práctica está compuesta de dos partes, cada una de las cuales se debe entregar por separado: la primera consiste en la implementación y presentación de jugadores a un torneo entre todos los alumnos de la asignatura. La segunda parte sigue el formato habitual en prácticas anteriores y requiere la entrega de código y memoria. Cada una de las dos partes se valora en 5 puntos.

Como resultado de los enfrentamientos entre los jugadores se publicará una clasificación cuyo enlace se mostrará en la página de prácticas y que será refrescado periódicamente para reflejar la situación de los nuevos jugadores. Entre los jugadores habrá algunos suministrados por los profesores, cuyo objetivo es servir de *indicadores* de nivel de juego. La nota de la parte A (torneo) corresponderá al resultado del mejor de los 3 jugadores presentados por cada pareja en el momento de cierre del torneo en relación a los jugadores de referencia, según el baremo siguiente:

- jugador que supera a 5 de los jugadores de referencia: 5 puntos
- jugador que supera a 4 de los jugadores de referencia: 4 puntos
- jugador que supera a 3 de los jugadores de referencia: 3 puntos
- jugador que supera a 2 de los jugadores de referencia: 2 puntos
- jugador que supera a 1 de los jugadores de referencia: 1 punto
- jugador que entra en competición pero no supera a ninguna referencia: 1/2 punto

Además se concederá un premio adicional de 1 punto (lo que permitiría obtener una nota de 11) a los 5 jugadores que ocupen las primeras posiciones del ranking final, siempre que no estén ocupadas por uno de los indicadores.

Nótese que, a pesar de que en la clasificación puede haber hasta 3 jugadores por pareja, a efectos de aplicar el baremo se considerará únicamente uno (el mejor situado). 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.

Los restantes 5 puntos de la nota serán asignados según el criterio habitual, por evaluación de la entrega de la parte B, donde también se tendrán en cuenta criterios de diseño del jugador entregado.

Consideraciones adicionales:

- En la parte A de esta práctica (torneo) se podrán realizar múltiples entregas, aunque tanto el número de entregas por día como el total de jugadores activos por pareja están limitados según se especifica más adelante.
- En las entregas para el torneo se presentará **un solo fichero** no comprimido, con extensión .cl, cuyo nombre y contenido viene descrito más adelante. El lugar de entrega no será el habitual, sino un

directorio específico, situado en la Zona de Entrega de Prácticas de la EPS, denominado Torneo. Para poder entrar en la página de entregas es preciso usuario y contraseña. Dado que las entregas a este grupo no son visibles en el registro, conviene revisar la confirmación: **`El envío del fichero XYZ.cl ha sido almacenado en IA/torneo`**. En caso de duda, se puede intentar enviar por segunda vez. Si el envío anterior fue correcto aparecerá el mensaje: **`Ese fichero ya existe`**.

- Cuanto antes se empiece a concursar más experiencia se irá ganando en el desarrollo de una buena función de evaluación, lo que permitirá escalar puestos en la clasificación. Por el contrario, esperar a los últimos días reduce la cantidad de pruebas que se podrán hacer.
- El torneo funciona en modo automático, publicando resultados periódicamente (una vez estabilizado será cada 2 horas).
- Existen una página de avisos y otra de preguntas frecuentes (FAQ) que conviene visitar periódicamente. Se ruega no preguntar a los profesores sin haber leído antes las FAQ.
- Las entregas inválidas serán eliminadas. Se dispone de plazo suficiente para corregir cualquier deficiencia, por lo que no se harán excepciones.
- Debe tenerse en cuenta que por motivos inesperados (mantenimiento, averías …) puede haber interrupciones de servicios de la UAM que afecten al torneo. Ello no será motivo para aplazar la fecha de cierre, dado que el plazo de que se dispone para realizar entregas es suficientemente amplio. Por ello se recomienda comenzar lo antes posible probando los jugadores.

PARTE A (5 puntos)

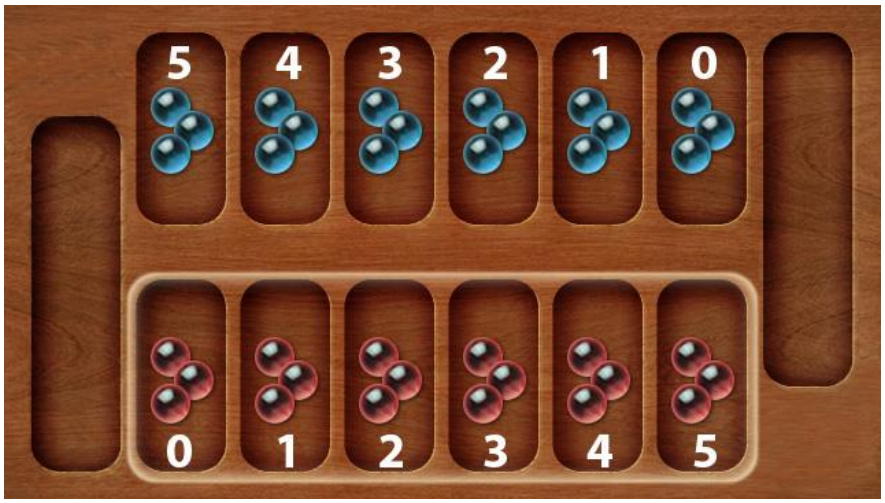
Introducción

En esta parte de la práctica se pide implementar un jugador para una variante de la familia de juegos Mancala. 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.

Elementos del juego

Tablero

En nuestra variante Mancala8, 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.

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 consigue que su oponente posea una o ninguna semilla en su fila (en tal caso es el ganador de la partida), 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***: se da sólo cuando tras la siembra, la última semilla se deposita en un hoyo vacío (hoyo de última siembra). En este caso, 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 en el Kalaha del jugador (el de la derecha).
    2. ***Repetición turno***. Si la siembra termina en el Kalaha no se captura pero se juega de nuevo. 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. ***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 des-comentadas 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[1]:

```
   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:  5  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 (nótese que el asterisco ha cambiado de lado). 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
   ----------------
*  1  5  7  0  2  3     23  Ac:  5  Jugador 2
 5                  10
   1  0  1  0  1  0     13  Ac:  0  Jugador 1
   ----------------
   0  1  2  3  4  5
```

[1] Téngase en cuenta que, pese a activar los comentarios, cuando juega un jugador automático los tableros que aparecen al ejecutar el código son los resultantes de toda la jugada. Los tableros intermedios que aquí se muestran no serían sólo visibles.

Puesto que su último hoyo estaba vacío y el contrario contiene fichas, el jugador 2 las captura, llevándolas a su Kalaha, quedando el tablero:

```
    5  4  3  2  1  0
    ----------------
*   0  5  7  0  2  3     24  Ac:  6  Jugador 2
 7                    10
    0  0  1  0  1  0     12  Ac:  0  Jugador 1
    ----------------
    0  1  2  3  4  5
```

Y así sucesivamente hasta 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.

Fase de Familiarización con el juego

Se proporciona al alumno el código del juego que cumple las reglas anteriores. Se pide que en primer lugar se familiarice con el juego, haciendo ejecuciones y examinando el código suministrado. La implementación del tablero en el código entregado sigue el formato mostrado anteriormente, donde las cifras superiores (5 a 0) e inferiores (0 a 5) indican los hoyos de cada jugador. Para realizar una jugada, el jugador sólo tiene que indicar la posición del hoyo cuyas fichas recoge para empezar a sembrar. El resto de la mecánica es automático y puede originar una cascada de movimientos con los que es preciso familiarizarse antes de empezar a codificar. El asterisco (*) a la izquierda de una de las filas indica el jugador que tiene el turno. Los números que aparecen a la derecha de las filas indican el número total de semillas en la fila más las que tenga cada jugador en su Kalaha, lo que llamaremos los puntos del jugador. En la configuración inicial por defecto de toda partida, ambas filas tienen 18 semillas, distribuidas como se indica en el apartado tablero, pero es posible forzar posiciones de inicio distintas para experimentar con el juego o analizar problemas.

El programa muestra en cada jugada las opciones posibles, es decir, los hoyos en los que el jugador puede comenzar una siembra (en esta versión hay sentido único). En modo manual no se aceptan otros valores, pero puede abortarse el juego en cualquier momento respondiendo **x** en lugar de una acción. En modo automático, un valor inválido o fuera de rango producirá un error que implica la pérdida de la partida.

En la última página de esta práctica se ofrece un tablero de juego para facilitar la experimentación con elementos físicos como semillas o fichas (la mente humana suele procesar mejor información motora-visual que numérica).

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.

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 generaran 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 utilizará 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 texto de Russel & 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):

1. A.M. Turing, "Computing machinery and intelligence" Mind, Vol. 59, No. 236, pp. 433-460 (1950)
2. A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers" IBM Journal of Research and Development, 3(3), pp. 210‑229 (1959)
3. A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers II. Recent Progress" IBM Journal of Research and Development, 11(6), pp. 601-617 (1967)

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). 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 (eg. *"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.

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. Con tal motivo, 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)  …)
```

Además de dicha función se pueden entregar un número limitado (véase apartado de normas de entrega) de funciones auxiliares. 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. Hay que tener en cuenta que el tiempo disponible para la partida está limitado, por lo que funciones lentas podrán originar una descalificación por tiempo (timeout) que implica la pérdida de la partida. 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     #'minimax
                          :f-eval      #'mi-f-ev))
```

En el código se incluyen, además de un jugador humano que simplemente espera una entrada manual por consola, varios ejemplos de jugadores, entre ellos los denominados: *aleatorio*, *regular* y *bueno* a fin de que el alumno pueda comparar su jugador jugando contra ellos. Al final del código se incluyen ejemplos de cómo ejecutar el juego cambiando los jugadores, así como para forzar una determinada situación del tablero para realizar pruebas o análisis.

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.

La expansión de nodos es una **práctica no recomendable** (por los motivos anteriormente explicados). La usa el jugador bueno con el fin de que sirva de *sparring-partner* suficientemente correoso contra los jugadores del alumno sin por ello dar a conocer ninguna otra técnica de evaluación, pero el objetivo del alumno debe ser superar a este jugador por otros medios, por ejemplo realizando estrategias basadas en la evaluación del despliegue posicional y no sólo en la cantidad de fichas robadas.

Funciones a disposición del alumno

A fin de separar los conceptos de búsqueda y evaluación de estados, la función de evaluación debería basar su criterio exclusivamente en la observación del estado del tablero objetivo, y no en la expansión de nodos, por lo cual se recomienda limitarse a las funciones siguientes:

`(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 2x7 dimensiones.

(estado-lado-sgte-jugador estado)
Devuelve el estado 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 2xN 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).

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 es sí 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**

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

muy probablemente eliminados por timeout. No es posible asegurar que siempre serán eliminados, porque la duración de una evaluación depende también del nº 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 B 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 5s con incrementos de 0.5s, 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-mmx-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
```

Inclusión del Juego en otros programas

Como sería esperable de cualquier otra función, la función partida devuelve el resultado de la partida: 1 si gana el primer jugador, 2 si gana el segundo y 0 en caso de tablas. Ello permite al alumno incorporar el juego a otros programas que puedan ser de utilidad en mejorar su juego.

Partidas P2P

Asimismo, se proporcionan funciones para el juego peer-to-peer (P2P), que permiten a dos jugadores enfrentarse en tiempo real en modo remoto (en casa o en los laboratorios) tanto en modo manual como automático, sin por ello dar a conocer su estrategia. Estos enfrentamientos son de carácter privado entre alumnos, y no serán tenidos en cuenta en el torneo, aunque se ofrece un servicio de publicación web. Dicha página web publicará una lista con los jugadores disponibles ("chulos"), que desafían a quien quiera

enfrentarse a ellos ("aspirantes"), aunque pueden limitar el nº de partidas que aceptan jugar. Para ello publicarán su IP (sea de su casa o del laboratorio) y el puerto por el que escuchan. Los aspirantes interesados en desafiarles establecerán una comunicación TCP/IP con ellos mientras dure la partida. Una vez terminada la partida se publicará el resultado de la misma. Las instrucciones de uso del sistema de torneos P2P se publicarán en un documento específico en la página de prácticas. La participación en estas partidas es discrecional y no tienen influencia ni en la clasificación ni en la nota final.

http://arantxa.ii.uam.es/~ia/practicas/tor/prod/avisos-p2p.html

http://arantxa.ii.uam.es/~ia/practicas/tor/prod/faqs-p2p.html

Partida paso a paso

A continuación se muestran las primeras jugadas a profundidad 2 entre un jugador humano y otro automático 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-mmx* (por defecto a nil) permite obtener detalles de la evolución de minimax. 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-mmx-Regular*))

  Juego: (1) Humano vs Ju-Mmx-Regular (2)
Local game Humano-Ju-Mmx-Regular 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 y sigue jugando,
[J 1] Humano juega (R 3)
[J 2] Ju-Mmx-Regular pasa turno.
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 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 : 0

Juega (R 0)  => Ultima ficha en 3, con 0, contra: 3 => Captura 3 y termina,
```

```
[J 3] Humano juega (R 0)
TABLERO:

   5  4  3  2  1  0
  ------------------
*  3  3  3  0  3  3     15  Ac:  0
 0                   5
   0  4  4  0  4  4     21  Ac:  3
  ------------------
   0  1  2  3  4  5
[J 4] El turno es de Ju-Mmx-Regular

[J 4] Ju-Mmx-Regular juega (R 0)
TABLERO:

   5  4  3  2  1  0
  ------------------
   3  3  4  1  4  0     15  Ac:  0
 0                   5
*  0  4  4  0  4  4     21  Ac:  3
  ------------------
   0  1  2  3  4  5
[J 5] El turno es de Humano

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

Juega (R 1)  => Ultima ficha en 5, con 4, contra: 0: (21 15)
[J 5] Humano juega (R 1)
TABLERO:

   5  4  3  2  1  0
  ------------------
*  3  3  4  1  4  0     15  Ac:  0
 0                   5
   0  0  5  1  5  5     21  Ac:  3
  ------------------
   0  1  2  3  4  5
[J 6] El turno es de Ju-Mmx-Regular

[J 6] Ju-Mmx-Regular juega (R 1)
TABLERO:

   5  4  3  2  1  0
  ------------------
   4  4  5  2  0  0     15  Ac:  0
 0                   5
*  0  0  5  1  5  5     21  Ac:  3
  ------------------
   0  1  2  3  4  5
[J 7] El turno es de Humano

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

Juega (R 2)  => Ultima ficha en 0, con 0, contra: 6: (20 16)
[J 7] Humano juega (R 2)
TABLERO:

   5  4  3  2  1  0
  ------------------
*  4  4  5  2  0  1     16  Ac:  0
 0                   6
   0  0  0  2  6  6     20  Ac:  3
  ------------------
   0  1  2  3  4  5
[J 8] El turno es de Ju-Mmx-Regular

[J 8] Ju-Mmx-Regular juega (R 0)
TABLERO:

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

```
   4  4  5  2  0  0     22  Ac:  6
 7                  6
*  0  0  0  2  0  6     14  Ac:  3
  -----------------
   0  1  2  3  4  5
[J 9] El turno es de Humano

...
```

Normas de entrega de jugadores

Formato del nombre

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

ggppnaammddKK.cl

Donde:

- **gg:** dos últimos dígitos del grupo de prácticas (i.e. eliminando el 23)
- **pp:** dos dígitos (con 0 a la izda. si necesario) del nº de pareja.
- **n:** 1-3 para distinguir los 3 jugadores permitidos a la pareja.
- **aammdd:** año, mes y día de la entrega, justificados con 0 a la izda.
- **KK:** dos primeros caracteres de la clave personal y secreta (se distinguen mayúsculas y minúsculas), para evitar que contrincantes malintencionados pudieran bloquear las entregas de otras parejas.

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

A lo largo de las próximas semanas cada pareja podrá presentar múltiples jugadores al torneo, donde serán enfrentados a otros y listados en la clasificación que será publicada periódicamente en la web de prácticas. El número de entregas por día estará limitado a 1 por jugador (es decir, se podrán entregar hasta 3 jugadores en el mismo día pero deben tener distinto **n**). Asimismo, el número máximo de jugadores activos por pareja es 3, todos con nombre distinto (para lo cual se usará la **n** del nombre).

El envío de un jugador con un **n** que ya ha sido utilizado antes implica que el nuevo jugador reemplaza al antiguo. Esto permite al alumno substituir selectivamente de entre sus tres jugadores a aquél que esté peor situado en la clasificación y mantener el resto.

Todos los jugadores que entran de nuevo en el juego empiezan sin puntos, que irán ganando tras múltiples enfrentamientos que se realizarán de modo automático. Esta situación incluye el caso del nuevo jugador que sustituye a otro, es decir no se *heredan* posiciones, cada jugador tiene que ganarse su posición en la clasificación. Sin embargo esta posición en la clasificación debe considerarse aproximada, pues no todos los jugadores han realizado las mismas partidas ni contra los mismos contrincantes (estos cambian durante la evolución del torneo). Al menos una vez cada 48 horas y al terminar el plazo de entrega de jugadores se realizará un campeonato "*todos-contra-todos*" entre los jugadores que estén activos en ese momento. Sólo el enfrentamiento *todos-contra-todos* es reproducible y exacto, por eso es el resultado de ese enfrentamiento final el que dará la nota.

Formato del contenido

El fichero contendrá en su primera línea un punto y coma (;) separado con un espacio de la clave personal e intransferible que será suministrada por el profesor a cada pareja durante la primera hora de prácticas, conservando mayúsculas y minúsculas.

Seguidamente podrá venir otra línea de comentario con un alias, según se describe más abajo.

A continuación, deberá contener la función de evaluación según la cabecera antes mencionada, seguida de un máximo de hasta 9 funciones auxiliares. No se debe incluir ningún otro tipo de función o ejecución, aunque se admite el uso de comentarios.

Ejemplo: La pareja 2 del grupo 2314, cuya clave secreta es AbCd123456 entrega su tercer jugador, con fecha 18-03-14, haciendo uso de la opción de alias (**nótese que el acento en *Campeón* no sería válido**):

Nombre Fichero: **`14023140318Ab.cl`**

Contenido:

```
; AbCd123456
; Campeon

(defun mi-f-ev (estado)  …)

(defun abcproc (a b c) …)

(defun xyzproc (x y z) …)
```

El alias permite dar al jugador en la clasificación un nombre distinto al nombre por defecto **`ggppn`**. Ello 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. Para ello basta con incluir en el fichero un segundo comentario, justo debajo de la contraseña, incluyendo una cadena de hasta 10 caracteres con el alias deseado para ese jugador. El alias debe ir separado por un espacio del ";" inicial y puede contener **números, mayúsculas, minúsculas y signos básicos de puntuación, pero no acentos, espacios o paréntesis**. Es responsabilidad del alumno asegurarse de utilizar un alias único. En cualquier caso, el sistema no utiliza 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.

PARTE B (5 puntos)

Pregunta B1. Función de Evaluación

Apdo B1.1. 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.

Apdo B1.2. Nótese que según el código entregado el jugador Bueno, pierde (si saca) con el Regular a profundidad 2, y pierde (si no saca) a profundidad 4. ¿No debería ganar siempre dado que analiza un nivel más de juego? Analice y explique el motivo. ¿Cómo evitar el problema en el jugador entregado?

Pregunta B2. Minimax y Minimax a-b. Se proporciona al alumno el código que implementa el algoritmo minimax, en su versión negamax, que inicia la búsqueda y devuelve el siguiente estado elegido por el jugador que tiene el turno. Se pide:

Apdo B2.1. Explique el código entregado:

a. Utilice ejemplos de evaluaciones que ilustren para qué sirve cada función.
b. Comente línea a línea las funciones implicadas en la implementación del algoritmo minimax.
c. Compare y comente la ejecución del código cuando la profundidad es par o impar.

Apdo B2.2. Codifique una función minimax-a-b que implemente el algoritmo minimax con poda alfa-beta. Los prototipos de las funciones a entregar en este apartado son los siguientes:

```
;;; Función minimax con poda alfa-beta
(defun minimax-a-b (estado profundidad-max f-evaluacion)    ...)
```

Es importante documentar de forma clara el algoritmo empleado en la función minimax-a-b realizada, así como su funcionamiento y sus características más relevantes. Para poder probar esta función, se creará un jugador que utilice el algoritmo minimax con poda alfa-beta para la búsqueda, y la función de evaluación definida en el apartado anterior por el alumno

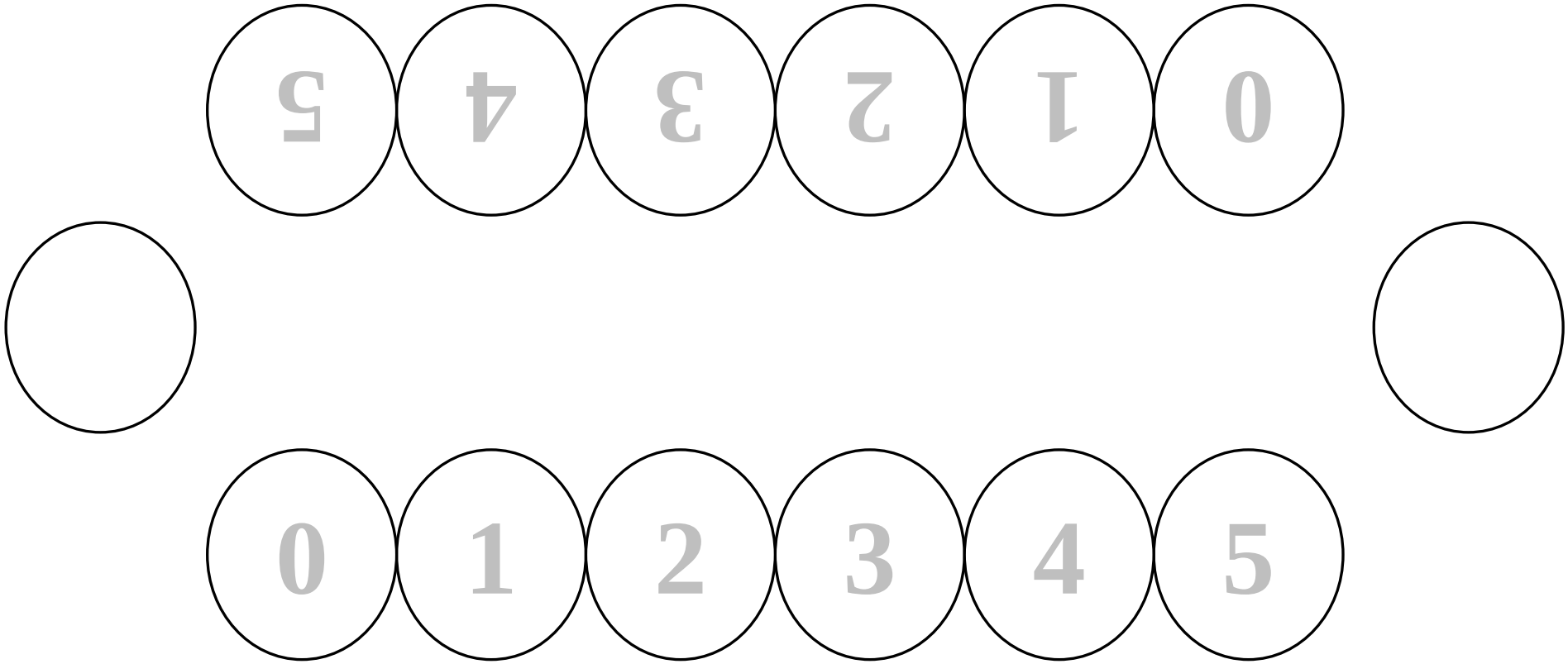
```
(setf *mi-jugador-a-b* (make-jugador
                          :nombre      'Jugador-Minimax-a-b
                          :f-juego     #'minimax-a-b
                          :f-eval      #'mi-f-ev))
```

Apdo B2.3. Compare el tiempo que tarda un jugador utilizando minimax y utilizando minimax con poda alfa-beta. Comente los resultados.

Apdo B2.4. Modifique el orden el que se exploran las jugadas. Comente el efecto que tiene en la poda alfa-beta modificar dicho orden.

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

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


5 4 3 2 1 0
0 1 2 3 4 5