PRÁCTICA 4

TORNEO DE CONECTA4

Gloria del Valle Cano & Leyre Canales Antón
gloria.valle@estudiante.uam.es & leyre.canales@estudiante.uam.es

Índice

1. Introducción

Esta prácta consta de una competición de jugadores sobre el juego de conecta4. Ciertos días que se nos espeficaron tuvimos que subir los jugadores creando asi 11 diferentes. Los jugadores son una heurística en lisp que debemos pensar y crear, intentando que sea lo mejor posible para ganar la competición. Para ello se nos proporcionan una funciones bases del juego de conecta4, que nosotros podemos utilizar para realzar las heurísticas de nuestros jugadores.

2. Jugador 1

Esta heríustica solamente genera números aleatorios con la función que se nos proporciona `random`, los devulve cuando el juego a terminado o si no devuelve el valor pre-definido de `+val-max+`.

Implementación de la heurística:

```
1     (defun heuristica (estado)
2             (if (juego-terminado-p estado)
3                     (random estado)
4                 +val-max+))
```

3. Jugador 2

Esta heríustica devuelvo el valor `0` o el valor `1` en función de lo que devulva la función auxiliar que se nos proporciona `generar-sucesores`.

Implementación de la heurística:

```
1     (defun heuristica (estado)
2         (if (generar-sucesores estado)
3                 0
4                 1))
```

4. Jugador 3

Esta heríustica devuelve siempre un número aleatorio generado por la función auxiliar que se nos proporciona `random`.

Implementación de la heurística:

```
1     (defun heuristica (estado)
2         (random estado))
```

5. Jugador 4

Esta heurística cuenta las fichas del oponente que se encuentran en el tablero utilizando la función que se nos proporciona `contar-abajo`, según el número de fichas que se encuentren sumamos una puntuación. La `puntuacion-oponente` suma más cuando se encuentran más fichas de éste, por el contrario la `puntuacion-actual` suma los valores de forma contraria. Finalmente se devuelve la resta de las dos puntuaciones calculadas.

Implementación de la heurística:

```
(defun heuristica (estado)
    (let* ((tablero (estado-tablero estado))
           (ficha-actual (estado-turno estado))
           (ficha-oponente (siguiente-jugador ficha-actual))
           (puntuacion-actual 0)
           (puntuacion-oponente 0))
           (loop for columna from 0 below (tablero-ancho tablero) do
                  (let* ((altura (altura-columna tablero columna))
                         (fila (1- altura))
                         (num-fichas (contar-abajo tablero ficha-oponente columna fila)))
                         (setf  puntuacion-oponente
                                (+ puntuacion-oponente
                                (cond ((= num-fichas 0) 0)
                                        ((= num-fichas 1) 5)
                                        ((= num-fichas 2) 10)
                                        ((= num-fichas 3) 15))))
                         (setf  puntuacion-actual
                                (+ puntuacion-actual
                                (cond ((= num-fichas 0) 15)
                                        ((= num-fichas 1) 10)
                                        ((= num-fichas 2) 5)
                                        ((= num-fichas 3) 0))))))
           (- puntuacion-actual puntuacion-oponente)))
```

6. Jugador 5

Esta heurística cuenta las fichas del oponente que se encuentran en el tablero utilizando la función que se nos proporciona `contar-abajo`. En caso de que se encuentren tres `fichas-oponente` sumamos a `puntuacion-oponente` el valor de 100 puntos y en caso contrario sumamos a `puntuacion-actual` el valor de 50 puntos. Finalmente restamos a la `puntuacion-actual` la `puntuacion-oponente`.

Implementación de la heurística:

```
(defun heuristica (estado)
    (let* ((tablero (estado-tablero estado))
           (ficha-actual (estado-turno estado))
           (ficha-oponente (siguiente-jugador ficha-actual))
           (puntuacion-actual 0)
           (puntuacion-oponente 0))
           (loop for columna from 0 below (tablero-ancho tablero) do
                  (let* ((altura (altura-columna tablero columna))
                         (fila (1- altura))
                         (num-fichas (contar-abajo tablero ficha-oponente columna fila)))
                         (if (= num-fichas 3)
                                (setf  puntuacion-oponente (+ puntuacion-oponente 100))
                                (setf puntuacion-actual (+ puntuacion-actual 50)))))
           (- puntuacion-actual puntuacion-oponente)))
```

7. Jugador 6

Esta heurística esta basada en la heurística del `JUGADOR5`, a la cual hemos añadido la condición de que si encuentra dos `fichas-oponente` sumamos a `puntuacion-oponente` el valor de 50 puntos y en caso contrario a `puntuacion-actual` el valor de 25 puntos. Finalmente como en el caso anterior restamos a `puntuacion-actual` el valor de `puntuacion-oponente`.

Implementación de la heurística:

```
1     (defun heuristica (estado)
2         (let* ((tablero (estado-tablero estado))
3                (ficha-actual (estado-turno estado))
4                (ficha-oponente (siguiente-jugador ficha-actual))
5                (puntuacion-actual 0)
6                (puntuacion-oponente 0))
7                (loop for columna from 0 below (tablero-ancho tablero) do
8                       (let* ((altura (altura-columna tablero columna))
9                              (fila (1- altura))
10                             (num-fichas (contar-abajo tablero ficha-oponente columna fila)))
11                             (if (= num-fichas 3)
12                                    (setf  puntuacion-oponente (+ puntuacion-oponente 100))
13                                    (setf puntuacion-actual (+ puntuacion-actual 50)))
14                             (if (= num-fichas 2)
15                                    (setf puntuacion-oponente (+ puntuacion-oponente 50))
16                                    (setf puntuacion-actual (+ puntuacion-actual 25)))))
17               (- puntuacion-oponente puntuacion-actual)))
```

8. Jugador 7

Esta heurística cuenta el numero de `ficha-oponente` que se encuentran en las diagonales por separado que denominamos como `num-fichas-diagonal-asc` y `num-fichas-diagonal-desc`. Según el número de fichas la primera diagonal sumamos en `puntuacion-asc` unos valores y según el número de fichas de la segunda diagonal sumamos en `puntuacion-desc` unos valores. Finalmente sumamos estas dos puntuaciones.

Implementación de la heurística:

```
1     (defun heuristica (estado)
2         (let* ((tablero (estado-tablero estado))
3                (ficha-actual (estado-turno estado))
4                (ficha-oponente (siguiente-jugador ficha-actual))
5                (acciones (acciones-posibles estado))
6                (puntuacion-asc 0)
7                (puntuacion-desc 0))
8             (loop for columna from 0 below (tablero-ancho tablero) do
9                (let* ((altura (altura-columna tablero columna))
10                      (fila (1- altura))
11                      (num-fichas-diagonal-asc
12                          (+ (contar-abajo-izquierda tablero ficha-oponente columna fila)
13                                  (contar-arriba-derecha tablero ficha-oponente
14                                  (1+ columna) (1+ fila))))
15                      (num-fichas-diagonal-desc
16                          (+ (contar-abajo-derecha tablero ficha-oponente columna fila)
17                                  (contar-arriba-izquierda tablero ficha-oponente
18                                  (1- columna) (1+ fila)))))
19                      (cond ((null acciones) nil)
20                              (t (setf  puntuacion-asc
21                                      (+ puntuacion-asc
22                                      (cond ((= num-fichas-diagonal-asc 0) 0)
23                                              ((= num-fichas-diagonal-asc 1) 5)
24                                              ((= num-fichas-diagonal-asc 2) 10)
25                                              ((= num-fichas-diagonal-asc 3) 15))))
26                              (setf  puntuacion-desc
27                                      (+ puntuacion-desc
28                                      (cond ((= num-fichas-diagonal-desc 0) 15)
29                                              ((= num-fichas-diagonal-desc 1) 10)
30                                              ((= num-fichas-diagonal-desc 2) 5)
31                                              ((= num-fichas-diagonal-desc 3) 0))))))
32
33                      ))
34
35            (+ puntuacion-desc puntuacion-asc)
36
37        ))
```

9. Jugador 8

Esta heurística esta basa en la que se nos proporciona del `jugador-bueno`, pero cambiamos los valores que se suman a las diferentes puntuaciones. Estos valores los tomamos basandonos en las heurísticas anteriores y los resultados obtenidos de los torneos y de las pruebas con los jugaroes que nos dan, intentando asi colocar las fichas en las posiciones mas centrales para conseguir más oportunidades de ganar. Finalmente devolvemos el valor de la resta de la `puntuacion-actual` y la `puntuacion-oponente`.

Implementación de la heurística:

```
1     (defun heuristica (estado)
2         (let* ((tablero (estado-tablero estado))
3          (ficha-actual (estado-turno estado))
4          (ficha-oponente (siguiente-jugador ficha-actual)))
5     (if (juego-terminado-p estado)
6         (let ((ganador (ganador estado)))
7           (cond ((not ganador) 0)
8                 ((eql ganador ficha-actual) +val-max+)
9                 (t +val-min+)))
10      (let ((puntuacion-actual 0)
11            (puntuacion-oponente 0))
12        (loop for columna from 0 below (tablero-ancho tablero) do
13              (let* ((altura (altura-columna tablero columna))
14                     (fila (1- altura))
15                     (abajo (contar-abajo tablero ficha-actual columna fila))
16                     (arriba (contar-arriba tablero ficha-actual columna fila))
17                     (der (contar-derecha tablero ficha-actual columna fila))
18                     (izq (contar-izquierda tablero ficha-actual columna fila))
19                     (abajo-der (contar-abajo-derecha tablero ficha-actual columna fila))
20                     (arriba-izq (contar-arriba-izquierda tablero ficha-actual columna fila))
21                     (abajo-izq (contar-abajo-izquierda tablero ficha-actual columna fila))
22                     (arriba-der (contar-arriba-derecha tablero ficha-actual columna fila)))
23                (setf puntuacion-actual
24                      (+ puntuacion-actual
25                         (cond ((= abajo 0) 0)
26                               ((= abajo 1) 15)
27                               ((= abajo 2) 500)
28                               ((= abajo 3) 3500))
29                         (cond ((= arriba 0) 0)
30                               ((= arriba 1) 15)
31                               ((= arriba 2) 500)
32                               ((= arriba 3) 3500))
33                         (cond ((= der 0) 0)
34                               ((= der 1) 15)
35                               ((= der 2) 500)
36                               ((= der 3) 3500))
37                         (cond ((= izq 0) 0)
38                               ((= izq 1) 15)
39                               ((= izq 2) 500)
40                               ((= izq 3) 3500))
41                         (cond ((= abajo-izq 0) 0)
42                               ((= abajo-izq 1) 15)
43                               ((= abajo-izq 2) 500)
44                               ((= abajo-izq 3) 3500))
45                         (cond ((= abajo-der 0) 0)
46                               ((= abajo-der 1) 15)
47                               ((= abajo-der 2) 500)
48                               ((= abajo-der 3) 3500))
49                         (cond ((= arriba-der 0) 0)
50                               ((= arriba-der 1) 15)
51                               ((= arriba-der 2) 500)
52                               ((= arriba-der 3) 3500))
53                         (cond ((= arriba-izq 0) 0)
54                               ((= arriba-izq 1) 15)
55                               ((= arriba-izq 2) 500)
56                               ((= arriba-izq 3) 3500))
57                         )))
58              (let* ((altura (altura-columna tablero columna))
59                     (fila (1- altura))
60                     (abajo (contar-abajo tablero ficha-oponente columna fila))
61                     (arriba (contar-arriba tablero ficha-actual columna fila))
62                     (der (contar-derecha tablero ficha-oponente columna fila))
63                     (izq (contar-izquierda tablero ficha-oponente columna fila))
64                     (abajo-der (contar-abajo-derecha tablero ficha-oponente columna fila))
65                     (arriba-izq (contar-arriba-izquierda tablero ficha-oponente columna fila))
```

```
                    (abajo-izq (contar-abajo-izquierda tablero ficha-oponente columna fila))
                    (arriba-der (contar-arriba-derecha tablero ficha-oponente columna fila))
                    )
              (setf puntuacion-oponente
                    (+ puntuacion-oponente
                       (cond ((= abajo 0) 0)
                             ((= abajo 1) 10)
                             ((= abajo 2) 200)
                             ((= abajo 3) 1900))
                       (cond ((= arriba 0) 0)
                             ((= arriba 1) 10)
                             ((= arriba 2) 200)
                             ((= arriba 3) 1900))
                       (cond ((= der 0) 0)
                             ((= der 1) 10)
                             ((= der 2) 200)
                             ((= der 3) 1900))
                       (cond ((= izq 0) 0)
                             ((= izq 1) 10)
                             ((= izq 2) 200)
                             ((= izq 3) 1900))
                       (cond ((= abajo-izq 0) 0)
                             ((= abajo-izq 1) 10)
                             ((= abajo-izq 2) 200)
                             ((= abajo-izq 3) 2500))
                       (cond ((= abajo-der 0) 0)
                             ((= abajo-der 1) 10)
                             ((= abajo-der 2) 200)
                             ((= abajo-der 3) 2500))
                       (cond ((= arriba-der 0) 0)
                             ((= arriba-der 1) 10)
                             ((= arriba-der 2) 200)
                             ((= arriba-der 3) 2500))
                       (cond ((= arriba-izq 0) 0)
                             ((= arriba-izq 1) 10)
                             ((= arriba-izq 2) 200)
                             ((= arriba-izq 3) 2500))
                       ))))
        (- puntuacion-actual puntuacion-oponente)))))
```

10. Jugador 9

Esta heurítica esta basada en la del `JUGADOR5` y `JUGADOR6`, a la cual hemos añadido la condición de que si encuentra una `ficha-oponente` sumamos a `puntuacion-oponente` del valor de 25 puntos y en caso contrario a `puntuacion-actual` el valor de 20 puntos. Finalmente como en las heuríticas anteriores restamos a `puntuacion-actual` el valor de `puntuacion-oponente`.

Implementación de la heurística:

```
(defun heuristica (estado)
    (let* ((tablero (estado-tablero estado))
           (ficha-actual (estado-turno estado))
           (ficha-oponente (siguiente-jugador ficha-actual))
           (puntuacion-actual 0)
           (puntuacion-oponente 0))
           (loop for columna from 0 below (tablero-ancho tablero) do
                  (let* ((altura (altura-columna tablero columna))
                         (fila (1- altura))
                         (num-fichas (contar-abajo tablero ficha-oponente columna fila)))
                         (if (= num-fichas 3)
                                (setf  puntuacion-oponente (+ puntuacion-oponente 100))
                                (setf puntuacion-actual (+ puntuacion-actual 50)))
                         (if (= num-fichas 2)
                                (setf puntuacion-oponente (+ puntuacion-oponente 50))
                                (setf puntuacion-actual (+ puntuacion-actual 25)))
                         (if (= num-fichas 1)
                                (setf puntuacion-oponente (+ puntuacion-oponente 25))
                                (setf puntuacion-actual (+ puntuacion-actual 20)))))
           (- puntuacion-oponente puntuacion-actual)))
```

11. Jugador 10

Esta heurística esta basada en la heurística del `JUGADOR7`, a la cual hemos añadido una condición de verdad cada vez que sumamos un valor a las puntuaciones que suma el valor 10 a la puntuación. Y en la parte final restamos las puntuaciones y el resultado de esa resta lo multiplicamos por -1.

Implementación de la heurística:

```
(defun heuristica (estado)
    (let* ((tablero (estado-tablero estado))
                   (ficha-actual (estado-turno estado))
                   (ficha-oponente (siguiente-jugador ficha-actual))
                   (acciones (acciones-posibles estado))
                   (puntuacion-asc 0)
                   (puntuacion-desc 0))
            (loop for columna from 0 below (tablero-ancho tablero) do
                   (let* ((altura (altura-columna tablero columna))
                          (fila (1- altura))
                          (num-fichas-diagonal-asc
                              (+ (contar-abajo-izquierda tablero ficha-oponente columna fila)
                                                  (contar-arriba-derecha tablero ficha-oponente
                                                      (1+ columna) (1+ fila))))
                          (num-fichas-diagonal-desc
                              (+ (contar-abajo-derecha tablero ficha-oponente columna fila)
                                                  (contar-arriba-izquierda tablero ficha-oponente
                                                          (1- columna) (1+ fila)))))
                          (cond ((null acciones) 0)
                                  (t (setf  puntuacion-asc
                                          (+ puntuacion-asc
                                          (cond ((= num-fichas-diagonal-asc 0) 0)
                                                  ((= num-fichas-diagonal-asc 1) 100)
                                                  ((= num-fichas-diagonal-asc 2) 200)
                                                  ((= num-fichas-diagonal-asc 3) 3000)
                                                  (t (+ puntuacion-desc 10)))))
                                          (setf  puntuacion-desc
                                                  (+ puntuacion-desc
                                                  (cond ((= num-fichas-diagonal-desc 0) 3000)
                                                          ((= num-fichas-diagonal-desc 1) 200)
                                                          ((= num-fichas-diagonal-desc 2) 100)
                                                          ((= num-fichas-diagonal-desc 3) 0)
                                                          (t (+ puntuacion-desc 10)))))))

                   ))

            (* (- 0 1) (- puntuacion-desc puntuacion-asc))

    ))
```

12. Jugador 11

Esta heurística está basada en la del `JUGADOR10` y el `JUGADOR7`, a la cual hemos añadido un nueva puntuación que es `puntuacion-abajo`. Esta se consigue contando el número de `ficha-oponente` con la función de `contar-abajo` que se nos proporciona. Como en el resto de puntuaciones sumamos los valores de 100, 200, 3000 en función de si hay 1, 2 o 3 fichas. Además hemso cambiado la suma de la condicón de verdad que sumara a la puntuación correspondiente el valor de 10000. Finalmente se resta el valor de `puntuación-asc` al de `puntuacion-desc` que se multiplica por -1 y al resultado de las operaciones anteriores se le suma el valor de `puntuacion-abajo`.

Implementación de la heurística:

```
1   (defun prueba15 (estado)
2       (let* ((tablero (estado-tablero estado))
3                      (ficha-actual (estado-turno estado))
4                      (ficha-oponente (siguiente-jugador ficha-actual))
5                      (acciones (acciones-posibles estado))
6                      (puntuacion-asc 0)
7                      (puntuacion-desc 0)
8                      (puntuacion-abajo 0))
9              (loop for columna from 0 below (tablero-ancho tablero) do
10                     (let* ((altura (altura-columna tablero columna))
11                            (fila (1- altura))
12                            (num-fichas-abajo (contar-abajo tablero ficha-oponente columna fila))
13                            (num-fichas-diagonal-asc
14                            (+ (contar-abajo-izquierda tablero ficha-oponente columna fila)
15                                    (contar-arriba-derecha tablero ficha-oponente (1+ columna)
16                                    (1+ fila))))
17                            (num-fichas-diagonal-desc
18                            (+ (contar-abajo-derecha tablero ficha-oponente columna fila)
19                                    (contar-arriba-izquierda tablero ficha-oponente (1- columna)
20                                    (1+ fila)))))
21                            (cond ((null acciones) 0)
22                                    (t (setf  puntuacion-asc
23                                            (+ puntuacion-asc
24                                            (cond ((= num-fichas-diagonal-asc 0) 0)
25                                                    ((= num-fichas-diagonal-asc 1) 100)
26                                                    ((= num-fichas-diagonal-asc 2) 200)
27                                                    ((= num-fichas-diagonal-asc 3) 3000)
28                                                    (t (+ puntuacion-asc 10000)))))
29                                    (setf  puntuacion-abajo
30                                            (+ puntuacion-abajo
31                                            (cond ((= num-fichas-abajo 0) 0)
32                                                    ((= num-fichas-abajo 1) 100)
33                                                    ((= num-fichas-abajo 2) 200)
34                                                    ((= num-fichas-abajo 3) 3000)
35                                                    (t (+ puntuacion-abajo 10000)))))
36                                    (setf  puntuacion-desc
37                                            (+ puntuacion-desc
38                                            (cond ((= num-fichas-diagonal-desc 0) 3000)
39                                                    ((= num-fichas-diagonal-desc 1) 200)
40                                                    ((= num-fichas-diagonal-desc 2) 100)
41                                                    ((= num-fichas-diagonal-desc 3) 0)
42                                                    (t (+ puntuacion-desc 10000)))))))
43
44                     ))
45
46             (+ (* (- 0 1) (- puntuacion-desc puntuacion-asc)) puntuacion-abajo)
47
48         ))
```