

Razón Artificial

Informática, robótica, razón artificial...

El salto del caballo, backtracking

8 ENERO 2010 2 COMENTARIOS

([HTTP://RAZONARTIFICIAL.WORDPRESS.COM/2010/01/08/EL-SALTO-DEL-CABALLO-BACKTRACKING/#COMMENTS](http://RAZONARTIFICIAL.WORDPRESS.COM/2010/01/08/EL-SALTO-DEL-CABALLO-BACKTRACKING/#COMMENTS))



(<http://razonartificial.files.wordpress.com/2010/01/caballo.jpg>) El salto del caballo es un viejo problema que existe mucho antes que los ordenadores. Consiste en, dada una posición inicial, en un tablero de ajedrez recorrer todas las casillas del tablero únicamente con los movimientos del caballo sin repetir ninguna casilla.

Este problema antiguamente se resolvía “a mano” probando posibles soluciones y anotando las que eran erróneas, es decir método prueba-error. Con la llegada de los ordenadores se solucionó mediante técnicas de inteligencia artificial. Usando algoritmos de búsqueda de caminos.


1	16	11	6	3
10	5	2	17	12
15	22	19	4	7
20	9	24	13	18
23	14	21	8	25

(<http://razonartificial.files.wordpress.com/2010/01/backtr3.gif>)

He creado una versión en Python de este problema típico de IA. Yo no se usar de momento los árboles en Python que sería lo ideal para resolver este problema así que lo he hecho de la manera que se me ha ido ocurriendo.

Partiendo de la idea de que el caballo puede mover como máximo a 8 posiciones diferentes el algoritmo consiste en pasar a una función los posibles movimientos del caballo descartar los que estén fuera del tablero y las casillas ya visitadas y probar con el resto. Si una posición se queda sin posibles siguientes

movimientos, volver a la anterior posición y probar la siguiente posible. Lo he implementado con una función recursiva, se llama así misma.

	3		2	
4				1
				
5				8
	6		7	

(<http://razonartificial.files.wordpress.com/2010/01/backtr2.gif>)

Heurística

En cuanto a la heurística, hay un método que dice que se debe visitar primero las casillas con menos movimientos posibles. Así que he hecho una función que analiza los posibles movimientos de los vecinos y ordena la lista de movimientos de menor a mayor.

El juego es de un tablero de 8×8, es decir 64 posiciones diferentes, que son las de un tablero de ajedrez, aunque se puede ajustar el juego para que halle la solución de un tablero de nxn

Bueno dejo el código después del salto.

```
[python]
# -*- coding: utf-8 -*-

#####
# Salto del Caballo
# Versión: 0.2
# Autor: Adrigm
# Email: adrigm.admin@gmail.com
# Web: http://www.adrigm.es (http://www.adrigm.es)
# Licencia: GPL
#####

# Bibliotecas.
import random
import sys

# Funciones.
# -----
```

```
# Dibuja el tablero.
def dibujar(tablero):
    tab = ""
    for f in range(8):
        for c in range(8):
            if tablero[f][c] < 10:
                celda = " " + str(tablero[f][c])
            else:
                celda = str(tablero[f][c])
            tab = tab + celda + " "
        tab = tab + "\n"
    print tab

# Devuelve una lista con las coordenadas del caballo.
def pos_c(tablero):
    for f in range(8):
        for c in range(8):
            if tablero[f][c] == "CC":
                pos = [f, c]
    return pos

# Devuelve 1 si el tablero está lleno.
def lleno(tablero):
    for f in range(8):
        for c in range(8):
            if tablero[f][c] == "__":
                return 0
    return 1

# Mueve desde CC desde inicial hasta final.
def mover(inicial, final, tablero, contador):
    tablero[inicial[0]][inicial[1]] = contador
    tablero[final[0]][final[1]] = "CC"

# Retrocede una posición.
def retroceder(inicial, final, tablero):
    tablero[inicial[0]][inicial[1]] = "__"
    tablero[final[0]][final[1]] = "CC"

# devuelve una lista con los movimientos posibles.
def posibles(mov, pos, tablero):
    posibles = []
    for n in range(8):
        if (pos[0] + mov[n][0]) in range(8) and (pos[1] + mov[n][1]) in range(8):
            if tablero[pos[0] + mov[n][0]][pos[1] + mov[n][1]] == "__":
```

```
v = [pos[0]+mov[n][0], pos[1]+mov[n][1]]
posibles.append(v)
return posibles
```

```
# Ordena dos listas en función de la primera.
```

```
def ordenar(lista, listb):
if listb == []:
return listb
else:
for i in range(len(lista)):
for j in range(len(lista) -1):
if lista[j] > lista[j+1]:
listb[j], listb[j+1] = listb[j+1], listb[j]
lista[j], lista[j+1] = lista[j+1], lista[j]
return listb
```

```
# Ordena los valores de manera que primero se mueva a las menos probables.
```

```
def heuristica(pos_mov, tablero, mov):
n_pos = []
for n in range(len(pos_mov)):
pos = len(posibles(mov, pos_mov[n], tablero))
n_pos.append(pos)
return ordenar(n_pos, pos_mov)
```

```
# Función recursiva principal.
```

```
def caballo(contador):
contador += 1
pos = pos_c(tablero)
mov = [[2, 1], [1, 2], [-2, 1], [2, -1], [-1, 2], [1, -2], [-2, -1], [-1, -2]]
```

```
pos_mov = posibles(mov, pos, tablero)
pos_mov = heuristica(pos_mov, tablero, mov)
```

```
for i in range(len(pos_mov)):
pos_ant = pos
mover(pos, pos_mov[i], tablero, contador)
dibujar(tablero)
if lleno(tablero):
sys.exit()
pos = pos_c(tablero)
caballo(contador)
retroceder(pos, pos_ant, tablero)
pos = pos_c(tablero)
dibujar(tablero)
```

```
# -----
```

```
# Crea un tablero de 8x8.
tablero = range(8)
for n in tablero:
    tablero[n] = range(8)

# Pone el tablero a 0.
for f in range(8):
    for c in range(8):
        tablero[f][c] = "&quot;__&quot;;

# Posición inicial caballo.
tablero[random.randint(0,7)][random.randint(0,7)] = "&quot;CC&quot;;

# Comienza el programa.
dibujar(tablero)
contador = 0
caballo(contador)
[/python]
```

ARCHIVADO EN INTELIGENCIA ARTIFICIAL PROGRAMACIÓN ETIQUETADO CON
BACKTRACKING, INTELIGENCIA ARTIFICIAL, JUEGOS, PROGRAMACIÓN, PYTHON

Acerca de adrigm

Pues soy yo mismo, único e incopiable ;)

2 Respuestas a *El salto del caballo, backtracking*

jacobo dice:

13 junio 2011 en 1:27 am

oie, que representa este pedazo de código?

tablero[f]1
es de la linea 24.

lo pregunto porque lo intenté correr pero da error de sintaxis en esa parte.

adrigm dice:

13 junio 2011 en 1:30 am

Es para Python 2.x asegúrate que no estás usando una versión 3.x

Blog de WordPress.com.

Seguir

Seguir “Razón Artificial”

Ofrecido por WordPress.com