Pràctica 2
Joc de dames

ESCOLA TÈCNICA SUPERIOR
D'ENGINYERIA
Universitat Rovira i Virgili

David Domènech Vilà

Intel·ligència Artificial

2018-2019

Joc de dames

En aquesta pràctica implementarem una versió senzilla del joc de les dames amb intel·ligència artificial. El joc es juga en un taulell estàndard d'escacs de 8x8 amb 12 dames per cada jugador (blanques i negres), que quedaran col·locades de la següent forma

```
    0  1  2  3  4  5  6  7
  ------------------------
0 | .  N  .  N  .  N  .  N |
1 | N  .  N  .  N  .  N  . |
2 | .  N  .  N  .  N  .  N |
3 | .  .  .  .  .  .  .  . |
4 | .  .  .  .  .  .  .  . |
5 | B  .  B  .  B  .  B  . |
6 | .  B  .  B  .  B  .  B |
7 | B  .  B  .  B  .  B  . |
  ------------------------
```

A cada torn, el jugador corresponent mourà una fitxa. L'ordinador ens demana per teclat quin moviment volem fer i, a continuació, farà el seu moviment a partir de la implementació de l'algorisme de resolució de jocs corresponent.

Guanya el jugador que tingui més dames

Heurístiques

Per la resolució d'aquest joc es dissenyen 3 heuristiques diferents:
-A (Moure fitxa més llunyana)

En aquesta heurística es vol trobar el moviment que ens donarà una fitxa més llunyana en el taulell i més a l'esquerra del taulell.
Es recorrer tot el taulell buscan les nostres fitxes i si trobem una simulem els seu moviments possibles duran 2 nivells i escollim el taulell que ens dona una fitxa més llunyana a la nostra posició.

Per el cas de les fitxes blanques:

```
for(int i=0;i<8;i++){
    for(int j=0;j<8;j++){
        if(copia[i][j].equals("B")){
            if(primerCop==false){
                primerCop=true;
                heuristicaResult=i;
            }
        }
    }
}
```

-B (Moure fitxa més possibilitats de matar)

En aquesta heurística es simulen dos nivells i el que es vol es trobar una tirada que ens doni més possibilitats de matar, per aconseguir-ho es simulen tots els possibles moviments del taulell en aquest dos nivells i si en un nivell es pot matar una fitxa contraria retornem infinit. Si en cap de els possibles moviments durant els dos nivells trobem un infinit(taulell amb possibilitat de matar) fem un moviment aleatori dintre dels possibles.

```
for(int i=0;i<8;i++){
  for(int j=0;j<8;j++){
     if(tablero[x-1][j+1].equals("N")&&tablero[x-2][j+2].equals(".")){
           heuristicaResult=heuristicaResult+1;
           this.xInici = x;
           this.yInici = j;
           this.moviment = 1;
        }
     }
}
```

-C (Moure fitxa que dona més fitxes més lluny)

Per aquesta heurística el que es vol aconseguir és fer el moviment que ens donarà més fitxes més llunyanes de la nostre posició. Per fer-ho simulem tots els taulers possibles i de cada tauler sumem a la heurística la posició de cada una de les nostres fitxes en l'eix y.

```
for(int i=0;i<8;i++){
    for(int j=0;j<8;j++){
        if(copia[i][j].equals("B")){
            heuristicaResult=heuristicaResult+i;
        }
    }
}
```

Algorisme Minimax

En el nostre algoritme simulem dos nivells, el primer es el moviment de el primer jugador i el segon es el moviment del segon jugadors.

En el taule busquem totes les fitxes del nostre color i generem els següents taulers segons els moviments que puguin fer.

```
if(color.equals(COLOR)){
   for (int x = 0; x <= 7; x++) {
       for (int j = 0; j <= 7; j++) {
           if (nouTauler[x][j].equals("FITXA")) {
               generarFills(x, j, nouTauler,"COLOR",1);
           }
       }
   }
```

Ara de cada fitxa trobada simulem els seus possibles fills, de cada fitxa poden sortir dos fills un moviment cap a la dreta i un moviment cap a l'esquerra.
De cada fill que sigui possible cridem a la funció aplicarHueristica que ens retornarà el valor de heurística en aquell fill.

```
public int generarFills(x,j,tauler,color,nivell) {

fillPosible = ComprovarMovimientos(x, j, color, tauler, 1);
if (fillPosible == true) { //FILL DRET ES POSSIBLE
     resultatHeuristica1 = aplicarHeuristica();
     if ((heuristicaGLOBAL >resultatHeuristica1))) {
        heuristicaGLOBAL=resultatHeuristica1;
        this.xInici = x;
        this.yInici = j;
        this.moviment = 1;
     }
}
```

```
if (fillPosible == true) { //FILL ESQUERRA ES POSSIBLE
      resultatHeuristica2 = aplicarHeuristica();
      resultatHeuristica1 = aplicarHeuristica();
      if ((heuristicaGLOBAL >resultatHeuristica1))) {
         heuristicaGLOBAL=resultatHeuristica1;
         this.xInici = x;
         this.yInici = j;
         this.moviment = 1;
      }
```

Si l'heurística de un fill és més gran que la dels fills anteriors, actualitzen el moviment a fer.

Dintre de la funció aplicar heurística a més a més de calcular l'heurística del nivell segons les funcions que hem vist en l'apartat 1 d'aquest document, també generem els possibles fills del segon nivell.

Per fer.ho cridem a la funció generarFillsContrari, que ens simularà els fills de el nostre rival.

```
public int aplicarHeuristica() {
// calculem heuristiques(vist en l'apartat 1)
if(nivell==2){
   int heuristicaFills=generarFillsContrari(copia,color);
   if(heuristicaFills>heuristicaResult){
       heuristicaResult=heuristicaFills;
   }
}
```

La funció generarFillsContrari crida a la funció generarFills vista anteriorment però amb el tauler de el nivell 1 simulat i canvian el color de les fitxes a moure.

```
public int generarFillsContrari(){
for(int i=0; i<=7;i++){
   for(int j=0; j<=7;j++){
       if(tablero[i][j].equals(ficha)){
           heuristica=generarFills(i,j,tablero,color,2);
       }
   }
}
```

Ara aquesta funció calcula unes noves heuristiques i si es la heurística calculada és més gran que la que teníem anteriorment s'actualitzarà el moviment a realitzar per el moviment del pare d'aquest node.

Joc de proves

En el joc de proves es simularan partides de la maquina vs maquins amb diferents heuristiques i s'estudiarà en quina heurística la maquina guanya més partides.

-Heuristica A

A vs A

	Algoritme Negre	Algoritme Blanca	Guanyador	Temps milliseconds
Partida 1	A	A	Blanques	4945
Partida 2	A	A	Blanques	4042
Partida 3	A	A	Blanques	7022

A vs B

	Algoritme Negre	Algoritme Blanca	Guanyador	Temps
Partida 1	A	B	Blanques	7722
Partida 2	A	B	Blanques	3286
Partida 3	A	B	Blanques	4004

A vs C

	Algoritme Negre	Algoritme Blanca	Guanyador	Temps
Partida 1	A	C	Empat	3826
Partida 2	A	C	Empat	4521
Partida 3	A	C	Empat	3564

B vs A

	Algoritme Negre	Algoritme Blanca	Guanyador	Temps
Partida 1	B	A	Negres	3363
Partida 2	B	A	Negres	4343
Partida 3	B	A	Blanques	3270

B vs B

	Algoritme Negre	Algoritme Blanca	Guanyador	Temps
Partida 1	B	B	Empat	9744
Partida 2	B	B	Blanques	3815
Partida 3	B	B	Blanques	5248

B vs C

	Algoritme Negre	Algoritme Blanca	Guanyador	Temps
Partida 1	B	C	Negres	3707
Partida 2	B	C	Negres	2995
Partida 3	B	C	Negres	6169

C vs A

	Algoritme Negre	Algoritme Blanca	Guanyador	Temps
Partida 1	C	A	Blanques	4352
Partida 2	C	A	Blanques	4215
Partida 3	C	A	Blanques	5896

C vs B

	Algoritme Negre	Algoritme Blanca	Guanyador	Temps
Partida 1	C	B	Negres	4983
Partida 2	C	B	Blanques	4147
Partida 3	C	B	Blanques	3187

C vs C

	Algoritme Negre	Algoritme Blanca	Guanyador	Temps
Partida 1	C	C	Negres	2786
Partida 2	C	C	Blanques	2524
Partida 3	C	C	Negres	3588

Segons aquest joc de proves podem treure les següents conclusions.

-Segons victories:
L'heurística B és la millor de totes amb un total de 10 victòries, la segona és la A amb 4 victòries i l'última heurística és C que només ha guanyat 1 partida.

-Temps d'execució:
Segons el temps podem veure que la millor és la C amb una mitjana de 2966 milisegons, la segona millor és la A amb una mitjana de 5336 milisegons i la última és B amb una mitjana de 6269 milisegons.