PRÁCTICA 2: JOC

Heurísticas implementadas:

- **Heurística v1**

```
calcularHeuristicaV1(DamasNode node, int player_id)
int end = obtenerFilaFinal(player_id);
return obtenerNumeroDeCeldasCon(end, player_id);
```

Heurística más sencilla de todas. Cuenta el nº de damas del jugador actual y devuelve este número como valor heurístico, de modo que se dará preferencia a aquellos nodos que tengan más damas.

- **Heurística v2**

```
calcularHeuristicaV2(DamasNode node, int player_id)
int delta_nTokens=numberOfTokens(player_id)-numberOfTokens(opponent);
int opponent_tokens=0;
for (int row=0; row < getBoardSize(); row++){
   for (int col=0; col < getBoardSize(); col++){
       if (celda(row, col) == player_id){
           //Buscar fichas del contrario en los alrededores
           if (celda(row+1, col+1)==opponent) opponent_tokens++;
           if (celda(row+1, col-1)==opponent) opponent_tokens++;
           if (celda(row-1, col+1)==opponent) opponent_tokens++;
           if (celda(row-1, col-1)==opponent) opponent_tokens++;
return pow(delta_nTokens, 3)+ opponent_tokens;
```

Esta heurística favorece la eliminación de fichas del contrario: por un lado, calculando la diferencia entre el nº de fichas restantes para el jugador actual y el nº de fichas restantes para el contrario (cuanto mayor esta diferencia mejor) y por otro lado, calculando el nº de fichas del oponente que están alrededor del jugador actual (mas probabilidades de poder eliminar una ficha del contrario en la siguiente ronda). Esto último nos es útil sobretodo cuando hay empate en el nº de fichas, algo que es muy habitual. El segundo componente también es útil para que el algoritmo mueva las fichas “en conjunto” hacia el lado contrario (y no solo centrarse en que una ficha llegue hasta el final), aunque esto solo nos beneficiaría al principio de la partida (cuando las fichas del contrario se encuentren cerca de sus celdas iniciales).

La diferencia entre el nº de fichas restantes la he elevado a 3 para darle mayor importancia a este componente frente al nº de oponentes alrededor porque realmente lo que me interesa es siempre se favorezca aquellos nodos en los que se eliminan fichas del contrario. Elevado a 3 y no al cuadrado porque sino los valores negativos me los contaría también como buenos, y no es el caso.

- **Heurística v3**

```
public int calcularHeuristicaV3(DamasNode node, int player_id)
int hamming_distance=0;
Integer[][] currentBoard = node.getBoard();
Integer[][] goalBoard = createGoalBoard(node, player_id);
for (int row=0; row < node.getBoardSize(); row++){
    for (int col=0; col < node.getBoardSize(); col++){
        if (currentBoard[row][col] != goalBoard[row][col]){
            hamming_distance++;

return -hamming_distance;
```

Esta heurística está pensada para favorecer la similitud con un tablero "ideal" (goalBoard) mediante el cálculo de la similitud de éste con el tablero actual, utilizando para ello alguna distancia métrica (Hamming distance en este caso, pero podría haber sido otra como la distancia Euclidea o la similitud coseno). El tablero “ideal” sería un tablero en el que todas las fichas del jugador actual están en el lado contrario (las 12) y el resto de celdas estarían vacías (0 fichas del contrario). Esto es una situación que nunca se daría puesto que las fichas del jugador actual no pueden estar todas en el lado contrario y a la vez haber suficientes como para eliminar fichas del contrario. No obstante, nos sirve para priorizar las situaciones en o bien el jugador actual tiene fichas en el lado contrario o bien elimina fichas contrarias en su campo.

Tiempos de ejecución:

- MINIMAX:

Nivel máximo	Heurística v1	Heurística v2	Heurística v3
1	0.19s	null	0.23s
2	0.29s	0.30s	0.32s
3	0.38s	0.49s	0.52s
4	1.20s	1.23s	1.43s
5	4.03s	7.00s	7.43s
6	18.26s	10.22s	24.04s

- ALFABETA:

Nivel máximo	Heurística v1	Heurística v2	Heurística v3
1	0.22s	null	0.21s
2	0.29s	0.30s	0.25s
3	0.45s	0.35s	0.73s
4	0.57s	0.68s	0.74s
5	0.90s	0.98s	1.26s
6	2.17s	3.17s	4.92s
7	2.63s	5.10s	6.14s
8	59.3s	58.14s	59.34s

Calidad del juego: Heurísticas

A continuación, una comparativa de los ganadores enfrentando entre sí a las diferentes heurísticas:

	FICHAS NEGRAS: MINIMAX (nivelMaximo=6)			
FICHAS BLANCAS: MINIMAX (nivelMaximo=6)		Heuristica v1	Heuristica v2	Heuristica v3
	Heuristica v1		negras	negras
	Heuristica v2	blancas		blancas
	Heuristica v3	negras	negras	

*Las fichas negras empiezan la partida

Queda bastante patente que las negras juegan con ligera ventaja al empezar la partida. Contra todo pronóstico, la heurística que creo que consigue mejores resultados es la heurística 2, ya que pese a no tener la ventaja de empezar la partida consigue hacerse con la victoria.

Calidad del juego: Algoritmos de búsqueda

Ganadores enfrentando mismas heurísticas pero con algoritmos de búsqueda diferentes.

	FICHAS NEGRAS: MINIMAX (nivelMaximo =6)			
FICHAS BLANCAS: ALFABETA (nivelMaximo =7)		Heuristica v1	Heuristica v2	Heuristica v3
	Heuristica v1	blancas		
	Heuristica v2		blancas	
	Heuristica v3			blancas

*Las fichas negras empiezan la partida

De aquí creo que la conclusión que podría sacar es que poder llegar a un nivel más se nota, y según los resultados que he obtenido, influye más que la heurística.