Ampliación de Programación

Práctica 6

Juegos

Sergio García Mondaray

Escuela Superior de Informática de Ciudad Real
Universidad de Castilla-La Mancha

1 Enunciado

Tenemos un conjunto de N fichas de colores, cada una de un determinado valor, de entre K valores posibles (K<N). Dos jugadores cogen alternativamente fichas del conjunto, ganando el que coge la última ficha. En cada jugada un jugador puede coger bien todas las fichas de un color, o bien todas las fichas del mismo valor. Diseñe una función que dada una posición del juego la evalúe y devuelva la decisión a tomar.

2 Planteamiento y resolución del problema

Antes de nada, indicar que el programa realizado no sólo selecciona la mejor opción para una situación dada. He optado por implementar el juego completo, dando la posibilidad al usuario de enfrentarse a la máquina. Para dar alguna posibilidad al jugador humano, siempre se le deja empezar (en caso contrario la máquina ganaría siempre).

2.1 Representación

Nuestra representación del tablero es un array, donde cada elemento es una *ficha*. La estructura ficha contiene la siguiente información:

```
struct ficha{
    int color; //Color de la ficha (codificado como numero)
    int numero; //Numero de la ficha
    int cogida; //Si esta cogida ya o esta en juego
};
```

La situación actual del juego está formada por todas las fichas con las que el juego inició, diferenciando las que ya no están en juego (han sido cogidas por los jugadores) con una X. Veamos un ejemplo:

```
[ ] Ficha  0: Color =  0, Numero =  0
[X] Ficha  1: Color =  2, Numero =  2
[ ] Ficha  2: Color =  0, Numero =  0
[X] Ficha  3: Color =  1, Numero =  0
[X] Ficha  4: Color =  1, Numero =  2
```

2.2 Resolución del problema

La idea es generar el árbol de posibles movimientos, donde cada nivel representa los movimientos posibles de uno de los dos jugadores. Para cada movimiento posible, analizamos todos los movimientos posibles del rival, y para cada movimiento del rival todas las posibilidades que tenemos nosotros, así sucesivamente. ¿Cómo sabremos si una jugada nos hará ganar? es sencillo: si la situación que le dejamos al rival con esa jugada es una en la que no pueda ganar.

Para resolver el problema de analizar si una jugada nos hará ganar, nos basamos en 2 reglas sencillas:

1. Una jugada es ganadora si todas sus sucesoras son perdedoras.
2. Una jugada es perdedora si alguna de sus sucesoras es ganadora.

El caso base será cuando el tablero esté vacío, donde nuestra situación será perdedora (en la jugada anterior el rival quitó la última ficha).

2.3 Algoritmo utilizado

El algoritmo que, dada una situación, selecciona una jugada (será ganadora, si la hay), es el siguiente:

```
FUNCION mejorMovimiento(situacion: FICHA[], n: ENTERO, numColores: ENTERO,
                                                numNumeros: ENTERO, j: JUGADA*)
    VARIABLES
        i: ENTERO; situacion2[n]: FICHA[]; jugada2: JUGADA;
    FIN VARIABLES

    INICIO
        jugada2.ganadora = NO;
        SI "El tablero esta vacio" ENTONCES
            j->ganadora = NO; /* El jugador anterior gano */
        SI NO
            /* Recorremos todas las posibilidades para quitar un numero */
            PARA i = 0, j->ganadora = NO, MIENTRAS i < numNumeros Y
                                            j->ganadora == NO, INCREMENTO i++
                SI "Hay alguna ficha en juego con ese numero"
                    situacion2 = situacion /*copia*/
                    "Quitar numero i de situacion2"
                    mejorMovimiento(&situacion2[0], n, numColores, numNumeros,
                                                                    &jugada2);
                    j->tipoQuitar = NUMERO;
                    j->cualQuitar = i;
                    SI jugada2.ganadora = NO //la jugada que dejamos es perdedora
                        j->ganadora = SÍ;
                    FIN SI
                FIN SI
            FIN PARA
            /* ahora con los colores */
            PARA i = 0, j->ganadora = NO, MIENTRAS i < numNumeros Y
                                            j->ganadora = NO, INCREMENTO i++
                SI "Hay alguna ficha en juego con ese color"
                    situacion2 = situacion /*copia*/
                    "Quitar color i de situacion2"
                    mejorMovimiento(situacion2, n, numColores, numNumeros,
                                                                    &jugada2);
                    j->tipoQuitar = COLOR;
                    j->cualQuitar = i;
                    SI jugada2.ganadora = NO
                        j->ganadora = SI;
                    FIN SI
                FIN SI
            FIN PARA
        FIN SI
    FIN
```

```
struct jugada{
    int tipoQuitar; // Color o numero
    int cualQuitar; // Que color o numero
    int ganadora; // SI o NO
};
```

Podemos observar que buscamos un movimiento ganador empezando por comprobar quitando los números. Y sólo intentaremos encontrar una solución eliminando algún color cuando ninguna jugada de quitar un número nos haga ganar.

3 Valoración y tiempo de trabajo

El tiempo empleado en la realización de esta práctica ha sido de unas 11 horas aproximadamente, y la dificultad encontrada de 8 sobre 10. Sin duda ha sido la práctica más interesante, sobre todo por haber implementado el juego completo.