Trabajo 2: Árbol Mastermind

Grado de Ingeniería Informática en Sistemas de Información

Lenguajes de Programación

2º año: GIISI

Curso: 2022 - 2023

Ángel Custodio Eyama Alogo Okomo

Miguel Iñesta Garza

Contenido

1. Enunciado

El mastermind es un juego de lógica y deducción que se realiza entre dos participantes donde cada uno tiene un número secreto de cuatro cifras sin ceros ni cifras repetidas.

En cada ronda, el jugador intenta adivinar la combinación secreta del rival y si no ha acertado, el rival le proporciona pistas que le indican la cantidad de cifras correctas, pero en posición equivocada (considerados **tocados**) y la cantidad de cifras correctas en la posición correcta (considerados **hundidos**).

La dinámica del juego es básicamente ir adivinando la combinación completa poco a poco hasta obtenerla y ganar la partida.

Para este trabajo de Lenguajes de Programación se debe hacer un predicado en Prolog llamado **arbolmaster(A),** que devuelva un árbol que permita jugar al ordenador al mastermind.

Un árbol consta de dos partes, la raíz (X), que es la jugada que se pretende realizar y una lista de 14 árboles:
rama (X, [A40,A22,A13,A04,A30,A21,A12,A03,A20,A11,A02,A10,A01,A00,vacio]) siendo los Aij árboles con i hundidos y j tocados, y por supuesto un árbol vacío, que se denotará como **vacío.**

El árbol debe empezar por rama([1,2,3,4],...) es decir la primera jugada siempre será [1,2,3,4], las jugadas sucesivas se eligen yendo por la rama correspondiente a los hundidos y tocados que tenga la jugada anterior, si al hacer este proceso llegamos a una rama vacía es que el contrario nos ha engañado, y no hay ningún número acertado posible, o que hemos acertado en la jugada anterior, y en consecuencia no hay que hacer más jugadas.
Se debe intentar que la profundidad del árbol sea de 6, sin contar los vacíos del final, es decir, que permita acertar al ordenador siempre como mucho en 6 jugadas .

1. Construcción del árbol

Pese a que la llamada principal se encapsula en la llamada a **arbolmaster/1,** esta crea una lista con todas las combinaciones posibles de 4 cifras del 1 al 9 sin repetir y una lista con todas las combinaciones posibles de tocados y hundidos en forma de listas de dos elementos y llama a la regla que realmente genera el árbol. La regla encargada de crear el árbol recursivamente es **creaArbol/3** con su auxiliar **creaArboles/4.**

La regla **creaArbol/3** crea el árbol comenzando por la raíz [1,2,3,4] y realiza una primera llamada a **creaArboles/4** pasándole la primera jugada posible (en este caso [1,2,3,4]) como jugada anterior y la lista entera de combinaciones posibles para formar los subárboles correspondientes, luego se realiza otra llamada a **creaArboles/4** esta vez decrementando la lista de combinaciones de tocados y hundidos para formar los 14 árboles posibles de la jugada raíz.

La regla **creaArboles/4** recibe la jugada en un momento dado, el total de jugadas posibles y una lista con las combinaciones de tocados y hundidos posibles o restantes, con esos datos crea una nueva rama con la jugada siguiente y de manera recursiva los árboles hijos de esa jugada. Lo primero que hace es filtrar las jugadas en base a la jugada anterior y una respuesta de tocados y hundidos de la lista de posibles, una vez filtrada la lista de jugadas posibles utiliza

la primera combinación de dicha lista como nueva jugada y crea el nuevo árbol, una vez que se alcanza la profundidad máxima, mediante backtracking se crean ahora los árboles con el resto de respuestas posibles de tocados y hundidos recursivamente en los niveles que se descendieron inicialmente hasta llegar al primer nivel y pasar a crear el resto de árboles con las otras respuestas posibles de tocados y hundidos desde la jugada raíz.

Nota: Debido a la larga cantidad de datos que ha de estudiar cada función de filtrado y la inmensa cantidad de combinaciones posibles, la generación del árbol toma tiempo (más de un minuto).

2. Reglas básicas

En el código se utilizarán varias funciones o reglas que se consideran las básicas para la generación de jugadas y combinaciones.

- La regla **valido/1** recibe una lista de cuatro elementos y devuelve true si la lista cumple con la condición de las jugadas del mastermind, es decir, que los números vayan del 1 al 9 y no haya cifras repetidas.
- La regla **todos/1** genera una lista de listas con todas las combinaciones de jugadas que cumplen la regla **valido**.

3. Filtro de jugadas

Como se explicó en el enunciado, en cada rama se encontrará una jugada y una lista de árboles, en cada árbol se agruparán las siguientes jugadas y árboles según el número de tocados y hundidos que se reciban de respuesta.

Para filtrar dentro del total de jugadas posibles el conjunto de jugadas correspondiente a los tocados y hundidos posibles dada una jugada, se ha creado la regla **filtraJugadas/4** que recibe una lista de jugadas y devuelve una lista reducida dependiendo del resultado de tocados y hundidos que se contemple de la jugada pasada.

- **FiltraJugadas para [0,0].** En este caso tenemos cero hundidos y cero tocados, por lo tanto, se recopilan las jugadas que no contengan ningún número de la jugada anterior. Por ejemplo, si la jugada es [1,2,3,4] las jugadas filtradas serían todas las combinaciones distintas de los valores 5,6,7,8,9.
- **FiltraJugadas para [0,1].** En este caso tenemos cero hundidos y un tocado con lo cual se recogen las jugadas que contienen un único valor de la jugada anterior, pero en una posición distinta y se cambian los tres valores restantes.
- **FiltraJugadas para [0,2].** Filtro para cero hundidos y 2 tocados, se mantienen dos de los valores de la jugada anterior permutados y se cambian los dos restantes.
- **FiltraJugadas para [0,3].** Filtro para cero hundidos y 3 tocados, se mantienen tres de los valores de la jugada anterior permutados y se intercambia el valor restante.
- **FiltraJugadas para [0,4].** Filtro para cero hundidos y 4 tocados, se mantienen los cuatro valores de la jugada anterior permutados.
- **FiltraJugadas para [1,0].** Se mantiene una de las cifras de la jugada anterior en su misma posición y se cambian los tres valores restantes.

- **FiltraJugadas para [1,1].** Se mantiene un valor en su posición de la jugada anterior, se permuta la posición de un valor y se cambian los valores restantes.
- **FiltraJugadas para [2,2].** Se mantienen dos valores en su posición original y se permutan los otros dos.
- **FiltraJugadas para [2,0].** Se mantienen dos valores en la misma posición de la jugada anterior y se cambian los valores restantes.
- **FiltraJugadas para [1,2].** Se mantiene un valor en la misma posición, se permutan dos y se cambia el valor restante.

Tanto la combinación [4,0] como cualquier otra combinación (caso que no debería suceder) con la que se llame a filtraJugadas devolverá una lista vacía.

Para todos los filtros se ha utilizado la regla predefinida **findall/3,** incluyendo en ella todas las condiciones que se han considerado necesarias para restringir las jugadas.

4. Funciones auxiliares

- La regla **quitarepetidos/2** elimina los elementos duplicados de una lista.
- La regla **cuatroValoresEn/2** comprueba si una lista L contiene exactamente cuatro valores [X, Y, Z, T]. Es decir, comprueba que X, Y, Z, T son parte de L, se utiliza para encontrar cuatro valores que pertenezcan a una lista y poder utilizarlos para permutar o mantener sin tomar en cuenta un valor y posición específica de la jugada anterior.
- La regla **mismoValorNoPos/4** verifica si un par de elementos X e Y de las listas L1 y L2 tienen el mismo valor, pero se encuentran en posiciones distintas.
- La regla **mismaPosValor/4** verifica si un par de elementos X e Y de las listas L1 y L2 tienen el mismo valor y posición.
- La regla **mismosValores/2** verifica si dos listas contienen los mismos valores. Utiliza la regla predefinida **sort**/2 para comparar los elementos sin importar el orden.

Pruebas

Para comprobar que el programa puede llegar al resultado hemos simulado una partida a mano, imaginando una clave y manualmente respondiendo los tocados y hundidos. Al ver las jugadas siguientes con las que probaría el programa podemos comprobar que llega a la clave en 6 intentos, el resultado de la prueba es el siguiente:

Clave = [7432]

Primera jugada = [1234]

Resultado = [1,2]

Segunda jugada = [1325]

Resultado = [0,2]

Tercera jugada = [2164]

Resultado = [0,2]

Cuarta jugada = [3247]

Resultado = [0,4]

Quinta jugada = [4,7,3,2]

Resultado = [2,2]

Sexta jugada = [7,4,3,2]

Resultado = [4,0]