Inteligencia Artificial

Guia 1.2. Prolog

4° Licenciatura En Sistemas de Información

2020

Universidad: UADER FcyT Concepción del Uruguay

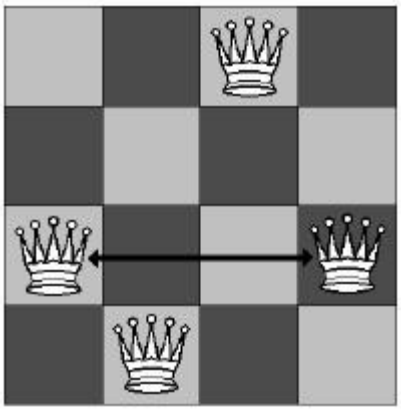
Profesor: Lopez de Luise Daniela, Bel Walter

Alumnos: Cepeda Leandro,

Guia 1.2. Prolog

1. El problema de K-reinas

El problema de las ocho reinas es un pasatiempo que consiste en poner ocho reinas en el tablero de ajedrez sin que se amenacen. Fue propuesto por el ajedrecista alemán Max Bezzel en 1848. En el juego del ajedrez la reina amenaza a aquellas piezas que se encuentren en su misma fila, columna o diagonal. El juego de las 8 reinas consiste en poner sobre un tablero de ajedrez ocho reinas sin que estas se amenacen entre ellas. Para resolver este problema emplearemos un esquema vuelta atrás (o Backtracking).



A[1]	A[2]	A[3]	...	A[k]
1	2	3	...	k

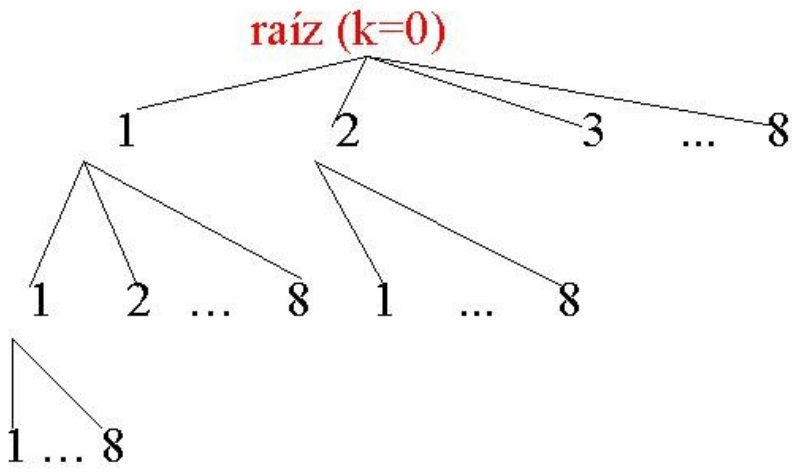
Bactracking significa ir aumentando el k en cada paso hasta llegar a n y asignar un valor posible para la última variable. Si esto no tiene éxito, hay que dar vuelta atrás (backtrack) en un árbol de búsqueda implícito.

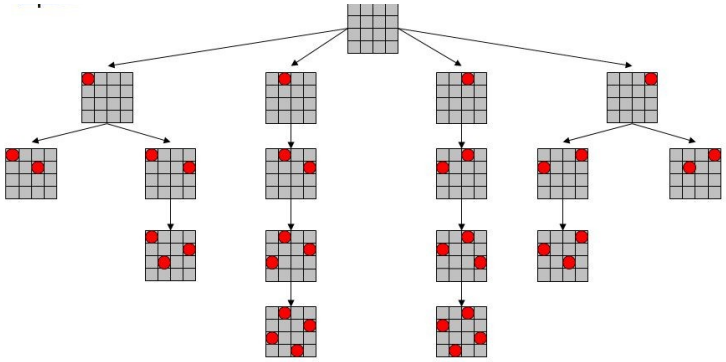
- Empezamos desde una solución parcial a = (a1, a2,...,ak); (al principio, vacía).
- Tratamos de extenderla agregando otro elemento al final.
- Después, comprobamos si tenemos una solución.
  - Si la tenemos, podemos imprimirla, guardarla, contarla, etc.
- Si no es solución, comprobamos si la solución parcial es todavía extensible para completar una solución
  - Si lo es, nos llamamos recursivamente y continuamos.
- Si no es extensible, borramos el último elemento de a e intentamos otra posibilidad para esa solución, si hay.

2. Representar el vector del problema

El vector del problema es: $a = [a_1, a_2, \ldots, a_K]$, donde cada indice representa a la fila y el valor a la columna donde se encuentra la reina. Cada reina estaría en la posición (k, a[k]) para k = 1 – 8.
Ejemplo:
.[4, 2, 5, 8, 6, 1, 3, 7]

3. Realizar el árbol de los hechos del problema para k=8





4. Realizar el seudocódigo

```
h: nos indica en cual renglón vamos a poner la reina
d: digonal derecha (135°)
i: digonal izquierda (45°)
v: vector que representa las columnas [1..8]
r: vector guarda la solucion actual [1..8]
di: vector que representa la diagonal izquierda [1..15]
dd: vector que representa la diagonal derecha [1..15]
sol: matriz [1..92,1..8]

procedure reinas(h)
    var d, i, j, k: entero

    para j desde 1 hasta 8 hacer
        d := j+h-1; i := 8+j-h

        % si las posiciones de los vectores estan en 0
        % significa que tanto las columnas como las diagonales no presentan
        si (v[j]-0) y (di[i]-0) y (dd[d]-0) entonces

            % guardamos en las posiciones correspondientes
            % de los vectores de (columas, diagonal derecha e izquierda)
            % las filas donde se coloca la reina y
            % en el vector de solucion actual guardamos la columa donde esta
            % ubicada la reina
            v[j] := h; di[i] := h; dd[d] := h; r[h] := j;

            % si la reina esta en la fila 8
            % significa que se encontro una solucion prometedora
            Si (h-8) entonces

                % incrementa el vector de soluciones en uno
                inc(s)

                para k desde 1 hasta 8 hacer

                % guardamos la solucion en la matriz de soluciones
                sol[s,k] := r[k]
            fin si

            % llamada recursiva
            sino
                reinas(h+1)

        v[j] := 0; di[i] := 0; dd[d] := 0;
        fin si
    fin para
fin procedure
```

5. Calcular el número de soluciones distintas y totalas conforma aumenta la cantidad de reinas en n=1, n=2, … , n=14.

	Soluciones distintas	Soluciones totales
1	1	1
2	0	0

3	0	0
4	1	2
5	2	10
6	1	4
7	6	40
8	12	92
9	46	352
10	92	724
11	341	2,680
12	1787	14,200
13	9,233	73,712
14	45,752	365,596

6. Detallar las 12 soluciones únicas del problema cuando k=8.

# Solución	Vector
1	[4,7,3,8,2,5,1,6]
2	[5,2,4,7,3,8,6,1]
3	[4,2,7,3,6,8,5,1]
4	[4,6,8,3,1,7,5,2]
5	[3,6,8,1,4,7,5,2]
6	[5,3,8,4,7,1,6,2]
7	[5,7,4,1,3,8,6,2]
8	[4,1,5,8,6,3,7,2]
9	[3,6,4,1,8,5,7,2]
10	[6,2,7,1,4,8,5,3]
11	[4,7,1,8,5,2,6,3]
12	[6,4,7,1,8,2,5,3]

7. implementar en PROLOG

*/*1.- La siguiente línea añade reglas de nivel superior, es muy importante para poner en ejecución sin el cual SWI-Prolog nos dará error.*/*
[user].

/*2.- nreinas (+N,?Sol). Es el predicado principal que nos permite conocer el resultado de la operación.*/
nreinas(N,Sol) :- generarTablero(N,Tablero), permutar(Tablero,Sol), buenTablero(Sol).

/*3.- generarTablero(+X,?Y). Este predicado genera un tablero de dimensión variable (N).*/
generarTablero(0,[]).
generarTablero(X,[X|R]) :- XMenos1 is X - 1, XMenos1 >= 0, generarTablero(XMenos1,R).

/*4.- permutar(?LX,?LY). Verifica si LY es una permutación de los elementos de LX, la única permutación de la lista vacía es la lista vacía.*/
permutar([],[]).
permutar(X,[C|Z]) :- seleccionar(X,C,R), permutar(R,Z).

/*5.- seleccionar(L,X,R). Verifica si X es un elemento de L y R, es la lista L sin el elemento X.*/
seleccionar([X|R],X,R).
seleccionar([C|R],X,[C|Y]) :- seleccionar(R,X,Y).

/*6.- buenTablero(+X). Verifica si en el tablero X, ninguna reina amenaza a otra; considerando que amenazar también se entiende ser amenazado.*/
buenTablero([]).
buenTablero([C|R]) :- not(amenaza(C,R)), buenTablero(R).

/*7.- amenaza(X,Y). Verifica si una reina colocada en la columna X de la fila n de un tablero amenaza a cualquiera de las demás reinas colocadas en las filas 0.n-1 del resto del tablero, y cuyas columnas vienen especificadas en la lista Y.*/
amenaza(X,Prof,[C|_]) :- X is C+Prof;
X is C-Prof;
X = C.
amenaza(X,Prof,[_|R]) :- ProfMas1 is Prof + 1, amenaza(X,ProfMas1,R).
amenaza(_,[]) :- fail.
amenaza(X,Y) :- amenaza(X,1,Y).

/*El predicado análogo amenaza(X,Prof,L), se cumple cuando X y otro elemento de la lista L se encuentran en la misma columna (mismo valor) o especifican reinas situadas en diagonal.*/

8. explicar el programa

Implementación

nreinas (+N,?Sol). El primer predicado y el cual llamamos desde el intérprete de Prolog . Se satisface si Sol es la solución al problema de las N-Reinas en un tablero de tamaño N.

```
nreinas(N,Sol):-generarTablero(N,Tablero),
permutar(Tablero,Sol),
buenTablero(Sol).
```

Este predicado generar un tablero de dimensión N, generar una permutación de ese tablero y por último comprueba si esa permutación contiene reinas es posiciones que no se amenacen unas a otras.

generarTablero(+X,?Y). se verifica si Y es una lista de X elementos que contiene los naturales comprendidos entre 1 y X, ambos inclusive. Nótese que:

- Como cada reina habrá de ocupar una columna distinta, el tablero (o tableros) solución será (serán) una permutación de un tablero así generado.

```
generarTablero(0,[]).
generarTablero(X,[X|R]):-
XMenos1 is X-1,
XMenos1 >= 0,
generarTablero(XMenos1,R).
```

Este predicado genera un tablero de la forma [N,N-1,N-2,….,1].

permutar(?LX,?LY). se verifica si LY es una permutación de los elementos de LX, la única permutación de la lista vacía es la lista vacía.

```
permutar([],[]).
permutar(X,[C|Z]) :- seleccionar(X,C,R), permutar(R,Z).
```

Para realizar la permutación de la lista de entrada, se seleccióna el primer valore de la lista y se permuta con el resto de la lista.

seleccionar(L,X,R). se verifica si X es un elemento de L y R es la lista L sin el elemento X.

```
seleccionar([X|R],X,R).
seleccionar([C|R],X,[C|Y]) :- seleccionar(R,X,Y).
```

buenTablero(+X). se verifica si en el tablero X, ninguna reina amenaza a otra. Hay que tener en cuenta que:

- Amenazar es una relación simétrica, es decir, lo mismo da amenazar que ser amenazada.
- Lo que se ha llamado Tablero es una lista de n elementos con los naturales de 1 a n donde cada elemento especifica la columna que ocupa la reina en el tablero, y la fila se especifica por el orden del elemento en la lista. (En resumidas cuentas, acorde a las especificaciones de la práctica).

```
buenTablero([]).
buenTablero([C|R]):- not(amenaza(C,R)),
buenTablero(R).
```

Se comprueba que la reina C no amena a las del resto del tablero y despues se comprueba el resto del tablero.

amenaza(X,Y). se verifica si una reina colocada en la columna X de la fila n de un tablero amenaza a cualquiera de las demás reinas colocadas en las filas 0..n-1 del resto del tablero, y cuyas columnas vienen especificadas en la lista Y. Utiliza para comprobar todo esto el predicado análogo amenaza(X,Prof,L), que se cumple cuando X y otro elemento de la lista L se encuentran en la misma columna (mismo valor) o especifican reinas situadas en diagonal.

```
amenaza(X,Prof,[C|_]):- X is C+Prof;
X is C-Prof;
X = C.
amenaza(X,Prof,[_|R]):- ProfMas1 is Prof + 1,
amenaza(X,ProfMas1,R).


amenaza(_,[]):- fail.
amenaza(X,Y):- amenaza(X,1,Y).
```

9. mostrar al menos 1 corrida con 9 reinas

```
?- consult('nreinas.pl').
?- nreinas(9,S).

S = [9, 7, 4, 2, 8, 6, 1, 3, 5] ;
S = [9, 7, 3, 8, 2, 5, 1, 6, 4] ;
S = [9, 7, 2, 4, 1, 8, 5, 3, 6] ;
S = [9, 6, 8, 2, 4, 1, 7, 5, 3] ;
S = [9, 6, 4, 7, 1, 8, 2, 5, 3] ;
```