Inteligencia Artificial

Guia 1.3. Prolog

4° Licenciatura En Sistemas de Información

2020

Universidad: UADER FcyT Concepción del Uruguay

Profesor: Lopez de Luise Daniela, Bel Walter

Alumnos: Cepeda Leandro,

1. El problema de K-reinas en 3GL: describir las adaptaciones del problema de k-reinas para JAVA a partir del programa en Prolog. Justificar

	Prolog	Java
Adaptaciones	Lenguaje interpretado. Lenguaje con un paradigma lógico. Lenguaje declarativo. Lenguaje de 5 generación.	Lenguaje compilado. Lenguaje de paradigma orientado a objetos. Lenguaje imperativo. Lenguaje de 3 generación.
Código	El algoritmo está compuesto de una serie de hechos y reglas definidos, los cuales permiten resolver el problema mediante la implementación de la recursividad.	El algoritmo está compuesto por una clase, que contiene un conjunto de métodos, los cuales permiten definir las operaciones necesarias para resolver el problema de manera recursiva,

Las diferencias nombradas en las adaptaciones, llevan a realizar una implementación distinta, pero con el mismo objetivo, en la cual ambos algoritmos funcionan de manera recursiva.

2. Mostrar el diseño del programa.

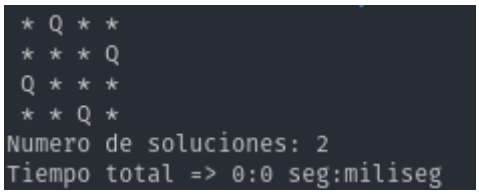
```
import java.util.ArrayList;
import java.util.List;

// PROGRAMA N-REINAS

class Main {

  static List calculateSolutions(int n) {
      long TInicio, TFin, tiempo;
      TInicio = System.currentTimeMillis();
      int[] arr = createZeroToNIntArray(n - 1);
      List<List<Integer>> validSolutions = new ArrayList<>();// The Integer Lists represent the positions of the
      // queens. The indexes are the x positions, and the values are the y positions.
      populateValidSolutions(arr, validSolutions, new ArrayList<>());
      TFin = System.currentTimeMillis();
      tiempo = TFin - TInicio;
      System.out.println("Tiempo de ejecución en milisegundos: " + tiempo);
      return validSolutions;
    }

    private static int[] createZeroToNIntArray(int n) {
      int[] arr = new int[n + 1];
      for (int i = 0; i < n + 1; i++) {
        arr[i] = i;
      }
      return arr;
    }

    static void populateValidSolutions(int[] arr, List<List<Integer>> validSolutions, List<Integer> queensYPosition) {
      /*
       *  Getting the permutations of the vertical positions of the queens, using DFS algorithm.
       */
      if (queensYPosition.size() == arr.length && !areDiagonalThreatens(queensYPosition)) {
        validSolutions.add(new ArrayList<>(queensYPosition));
      }
      for (int i : arr) {
        if (!queensYPosition.contains(i)) {
          queensYPosition.add(i);
          populateValidSolutions(arr, validSolutions, queensYPosition);
          queensYPosition.remove(queensYPosition.size() - 1);
        }
      }
    }

    static boolean areDiagonalThreatens(List<Integer> queensYPosition) {
      int n = queensYPosition.size();
      for (int i = 0; i < n; i++) {
        int min = Math.min(i, queensYPosition.get(i));
        int x = i - min;
        int y = queensYPosition.get(i) - min;
        while(x < n && y < n) {
          if(x != i && queensYPosition.get(x) == y) {
            return true;
          }
          x++;
          y++;
        }
        x = n;
        y = queensYPosition.get(i) - n + i;
        while(x > 0 && y < n - 1) {
          x--;
          y++;
          if(x != i && queensYPosition.get(x) == y) {
            return true;
          }
        }
      }
      return false;
    }

  public static void main(String[] args) {
    System.out.println("-- SOLUCIONES --");
    System.out.println("  ");
    System.out.println(calculateSolutions(8));
  }

}
```

3. Implementar el programa en C

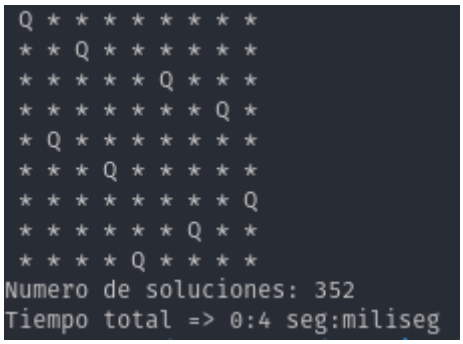
```
/------------------------------------------------------------------------+
// nreinas: Problema de las N reinas en secuencial |
//------------------------------------------------------------------------+

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/time.h>
#define TRUE 1
#define FALSE 0
#define MAX_REINAS 25
//------------------------------------------------------------------------+
// VARIABLES GLOBALES |
//------------------------------------------------------------------------+
// Se indexa por columna e indica fila en la que hay reina
int reinaEnFila[MAX_REINAS];
int numReinas, soluciones=0;
//------------------------------------------------------------------------+
int aceptable (int reinaFila, int reinaColumna) {
   int col;
   if (reinaEnFila[reinaColumna]!=0)
      return FALSE; // Reina en la misma columna
   for(col=1; col<=numReinas; col++)
   // Si hay una reina que me come en diagonal no es aceptable
   if ( ((reinaEnFila[col] != 0) && (abs(reinaEnFila[col]-reinaFila))==(abs(reinaColumna-col))))
      return FALSE;
   return TRUE;
}

//------------------------------------------------------------------------+
void NReinas(int reina) {
   int fila,columna,col;

   for(col=1;col<=numReinas;col++) {
      if(aceptable(reina,col)) {
         reinaEnFila[col]=reina; // Reina i siempre se ubica en fila i
         if (reina==numReinas) {
            if (soluciones==0) {
                  // printf("\n\n");
               for (fila=1;fila<=numReinas;fila++) {

                  for(columna=1;columna<=numReinas;columna++) {
                     if (fila!=reinaEnFila[columna])
                        printf(" *");
                     else printf(" Q");
                  }
                  printf("\n");
               }
            }
            soluciones++;
         } else NReinas(reina+1);
            reinaEnFila[col]=0;
      }
   }
}

//------------------------------------------------------------------------+
int main (int argc, char *argv[]) {
   int i;
   struct timeval t0, tf, t;
   for (i=0; i<MAX_REINAS; i++) reinaEnFila[i] = 0;
   soluciones=0;
   if (argc != 2) {
      printf ("Uso: nreinas numReinas\n");
      return 0;
   }
   numReinas = atoi(argv[1]);
   if (numReinas >= MAX_REINAS) {
      printf ("Error: El numero de reinas no puede superar %d\n",
      (MAX_REINAS-1));
      return 0;
   }
   gettimeofday (&t0, NULL);
   NReinas(1);
   gettimeofday (&tf, NULL);
   printf("Numero de soluciones: %d \n",soluciones);
   timersub (&tf, &t0, &t);
   printf ("Tiempo total => %ld:%ld seg:miliseg\n", t.tv_sec,
   t.tv_usec/1000);
   return 0;
}
```

4. documentar la corrida para diferentes valores de n, midiendo el tiempo que tarda.

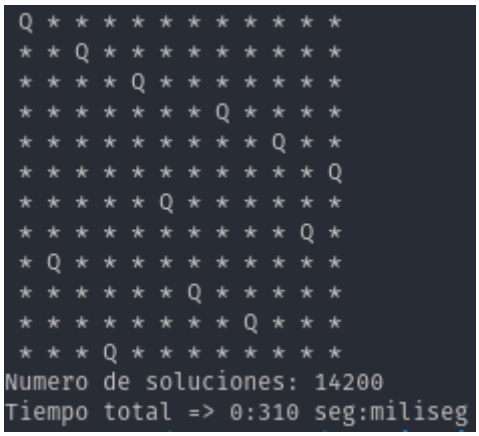
$./nreinas 4



$./nreinas 9



$./nreinas 12



5. comparar con SWI-PL: cantidad de líneas de código, tiempo de ejecución, cantidad de espacio (Mb) que ocupa el programa y sus librerías.

	C	Java	Prolog
Lineas de código	42	53	15
Tiempo de ejecución: 8 reinas.	Execution took 0,1 ms.	Execution took 0,35 ms.	Execution took 19 ms.
Espacio MB	0.029	0.027	0.02
Librerías	<stdio.h> <stdlib.h> <unistd.h> <sys/time.h>	java.util.ArrayList; java.util.List;	
O-grande	(O(n!) - O((n+1)!))	(O(n!) - O((n+1)!))	(O(n!) - O((n+1)!))