UNIVERSIDAD PRIVADA BOLIVIANA

ALGORITMICA 2

Proyecto Final

Resolución interactiva del problema Nqueens

Nombres: Nicole Góngora Rodríguez

Fabian Segurondo Ferrel

Código: 40796

52489

Carrera: Ingeniería de Sistemas Computacionales

Docente: Ing. Paul Wilker Flores Landaeta

Bolivia, La Paz, Noviembre 2020

Dedicatoria

Dedicamos este trabajo final a nuestra clase y a la carrera en su totalidad, esperamos que nuestras arduas horas de trabajo, consenso y búsqueda de soluciones sean motivo y ejemplo para ejecutar un trabajo con gran profesionalismo; que el reto asumido sea de gran valor para nuestro aprendizaje y otorgue valor a nuestro largo camino de investigación y trabajo con la tecnología.

Agradecimientos

Aprovechamos esta sección para demostrar y consagrar nuestro trabajo a nuestros padres y familia, a los cuales les estaremos eternamente agradecidos por ser los pilares fundamentales de nuestros estudios y apoyarnos incondicionalmente en todas las situaciones. A nuestro docente el ingeniero Paul W. Landaeta, sus enseñanzas en la materia, el desarrollo y búsqueda de soluciones, son de vital importancia para nuestra carrera profesional, además de ser un gran mentor y consejero.

Resumen

El trabajo realizado, es el desarrollo de un famoso problema en el mundo del desarrollo de algoritmos y de la ciencia de computación. “Nqueens” que es un reto planteado, busca emplear la lógica y los principios del backtracking/recursividad para ser resuelto, trabajando con mecanismos de análisis de las posiciones de una determinada pieza, que en este caso son las reinas (Queens) en un tablero, similar al del juego de ajedrez. Este problema se destaca por las diversas soluciones y caminos que se pueden presentar. Nuestro trabajo está enfocado en poder ejecutar un programa que interactúe con un usuario y este pueda jugar el juego. El proyecto se realizó en su totalidad en el lenguaje Java y se emplearon ciertas condiciones para su funcionamiento.

Abstract

The work done is the development of a famous problem in the world of algorithm development and computer science. “Nqueens”, which is a challenge, seeks to use the logic and principles of backtracking / recursion to be solved, working with mechanisms for analyzing the positions of a certain piece, which in this case are the queens (Nqueens) on a board, similar to the chess game. This problem is highlighted by the various solutions and paths that can be presented. Our work is focused on being able to execute a program that interacts with a user and this user can play the game. The project was carried out entirely in the Java language and certain conditions were used for its operation.

Introducción

El objetivo de este trabajo y proyecto final tiene como fin presentar el uso y ventajas de emplear el algoritmo de tipo “Backtraking” y de explicar el planteamiento del problema “Nqueens” famoso por presentar muchas soluciones posibles y de tener cierta complejidad al ser resuelto. Principalmente este reto consiste de: colocar ocho reinas en el tablero de ajedrez sin que estas se amenacen. Fue propuesto y desarrollado 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 N-reinas consiste en poner sobre un tablero de ajedrez, representado por una matriz de un determinado número de filas por columnas (una matriz NxM), donde se posicionan las N reinas sin que estas se presenten una amenaza entre ellas mismas. Para resolver este problema emplearemos la resolución de un mecanismo de recursividad o ir “vuelta atrás” que es el principio del “Back tracking”.

Es importante destacar que este famoso problema, desde que fue planteado el año 1848 a presentado varias aproximaciones para ser resuelto y destacados científicos y afamados matemáticos presentaron opciones de resolver este interesante problema. Matemáticos como el famoso Gauss y Georg Cantor, han trabajado en él y lo han generalizado a n-reinas. Las primeras soluciones fueron ofrecidas por Franz Nauck en 1850. Nauck también se abocó a las n-reinas (en un tablero de N x M de tamaño arbitrario). En 1874, S. Günther propuso un método para hallar las soluciones usando determinantes, y J.W.L. Glaisher redefinió su aproximación.

Edsger Dijkstra usó este problema en 1972 para ilustrar el poder de la llamada programación estructurada. Publicó una descripción muy detallada del desarrollo del algoritmo de backtracking, "depth-first".

Este acertijo apareció en el popular juego de computadora de los '90 llamado "The 7th Guest".

Tabla de Contenidos

Parte 1

Planteamiento del problema

Resolución de “Nqueens” incompletos

Desde un principio se levantaron diversas conjeturas de los posibles números de soluciones y de los límites y condiciones que debe cumplir este problema, concluyendo que principalmente que el reto de las “Nqueens”, tiene 92 soluciones, de las cuales 12 son esencialmente distintas, es decir las 92 soluciones existentes se pueden obtener a partir de traslaciones, simetrías y rotaciones de las 12 soluciones únicas. Por lo tanto, principalmente la solución necesita que, no más de dos reinas, compartan las mismas filas, columnas o estén bajo las diagonales de las matrices (representadas en el tablero de ajedrez). Inicialmente, las primeras soluciones terminaban siendo incompletas o el número de casillas que debería tener una reina era falta, por lo tanto, la implementación del algoritmo del backtracking resuelve el reto por medio de la incrementación y construcción de posibles soluciones y por cada iteración cambia las posibles soluciones, si es que estas ya no satisfacen las condiciones iniciales o ya no son válidas. Cabe recalcar que al colocar n reinas en un tablero de ajedrez de N x N de tal manera que ninguna de las reinas quede atacando a otras. El análisis y solución es isomorfo al de las ocho reinas. Finalmente, es necesario destacar que el tablero de mayor espacio en una matriz es el de 27 x 27 (filas por columnas) es el más grande hasta ahora numerado.

Posibles soluciones y planteamientos

Análisis del problema

Se han desarrollado varios algoritmos basados en búsquedas. Planteados para generar todos los conjuntos de soluciones posibles para un dado n por n tablero. Los algoritmos de retroceso [3, 4, 5] son uno de los algoritmos de búsqueda más utilizados para la resolución de problemas de n-reinas que generan sistemáticamente todas las soluciones posibles. La idea básica de retroceder algoritmos es construir el vector de solución, de un componente a la vez y probarlo de acuerdo con el criterio función para determinar si el vector que se está formando, todavía tiene posibilidades de éxito. Para explorar todo los posibles vectores el

algoritmo comienza colocando una reina en la primera columna de la primera fila y continúa colocando las reinas en las otras filas, manteniendo las restricciones impuestas por la iteración anterior, previamente colocado reinas. Si el algoritmo alcanza una fila para la cual todos los cuadrados ya son atacados por las otras reinas, este retrocede a la fila anterior y explora los otros vectores. En la práctica, sin embargo, los enfoques de retroceso proporcionan una clase muy limitada de soluciones para grandes tableros porque es difícil que una búsqueda de retroceso encontrar soluciones que sean significativamente distintas en la solución.

Como cada reina puede amenazar a todas las reinas que estén en la misma fila, cada una ha de situarse en una fila diferente. Podemos representar las N reinas mediante un vector[1-N], teniendo en cuenta que cada índice del vector representa una fila y el valor una columna. Así cada reina estaría en la posición (i, v[i]) para i = 1-N.

El vector (3,1,6,2,8,6,4,7) significa que la reina 1 está en la fila 1, columna 3; la reina 2 en la fila 2, columna 1; la reina 3 en la fila 3, columna 6; la reina 4 en la fila 4, columna 2; etc... Como se puede apreciar esta solución es incorrecta ya que la reina 3 y la 6 estarían en la misma columna. Por tanto, el vector correspondería a una permutación de los N primeros números enteros.

El problema de las filas y columnas lo tenemos resuelto, Sin embargo, es muy importante considerar la posición respecto a las diagonales, Para las posiciones sobre una misma diagonal descendente, se cumple que tienen el mismo valor fila-columna; mientras que para las posiciones en la misma diagonal ascendente, se cumple que tienen el mismo valor fila+columna.

Condiciones del problema para ser resuelto

Principalmente el juego debe cumplir las siguientes condiciones:

- No ser de mayor tamaño que 27 x 27.
- Las reinas son colocadas en cierto orden ascendente en las filas, columnas y diagonales.
- Por cada verificación de la reina, la posición es validada y por consiguiente es posible continuar con las siguientes movidas.

- Por lo tanto, las posiciones con respecto a las filas y columnas (i y j) deberán cumplir con:
    - i- j = k –l
    - i + j = k +l
    - j – l = i – k
    - j – l = k – i

    Teniendo en cuenta todas estas consideraciones, podemos aplicar el esquema retroactivamente para colocar las ocho reinas de una manera realmente eficiente. Para ello, reformulamos el problema como problema de búsqueda en un árbol. Decimos que en un vector de enteros entre 1 y N es-prometedor, para el numero de posiciones que recorra, si ninguna de las reinas colocadas en las posiciones con respecto a los vectores amenaza a ninguna de las otras. Las soluciones a nuestro problema se corresponden con aquellos vectores que son N-prometedores.

Parte 2

Algoritmo que resuelve el problema

Algoritmo de "Bakctracking/recursividad"

Código Back tracking

```
void search(int y) {
        if (y == n) {
        count++;
        return;
}
        for (int x = 0; x < n; x++) {
        if (col[x] || diag1[x+y] || diag2[x-y+n-1]) continue;
        col[x] = diag1[x+y] = diag2[x-y+n-1] = 1;
```

```
        search(y+1);
        col[x] = diag1[x+y] = diag2[x-y+n-1] = 0;
        }
}
```

La búsqueda comienza llamando a buscar (0). El tamaño del tablero es n, y el. El código calcula el número de soluciones a contar. El código asume que las filas y las columnas del tablero están numeradas del 0 al n - 1. Cuando se llama a buscar con el parámetro y, coloca una reina en la fila y luego se llama a sí mismo con el parámetro y + 1. Entonces, si y = n, se ha encontrado una solución y se aumenta el valor de un contador por uno. La matriz col realiza un seguimiento de las columnas que contienen una reina y las matrices con las respectivas: diagonal 1 y diagonal 2 realizan un seguimiento de las diagonales. No se permite agregar otra reina a una columna o diagonal que ya contiene una reina.

El algoritmo de retroceso anterior nos dice que hay 92 formas de colocar 8 reinas en el tablero de 8 × 8. Cuando n aumenta, la búsqueda rápidamente se vuelve lenta, porque el número de soluciones crece exponencialmente. Por ejemplo, ya se necesita minuto en una computadora moderna para calcular que hay 14772512 formas de colocar 16reinas en el tablero de 16 × 16.

De hecho, nadie conoce una forma eficaz de contar el número de combinaciones de reinas. para valores mayores de n. Actualmente, el mayor valor de n para el que se conoce el resultado es27: hay 234907967154122528 combinaciones en este caso. Esto fue descubierto en 2016 por un grupo de investigadores que utilizó un grupo de computadoras para calcular el resultado.

Implementación del código

```
public class
Nqueensprocess
{

        static int[] rows = new int[32];
```

```
        static boolean findSolution =false;


        static boolean valid(int row, int column) {
                for (int lastPlayed = 0; lastPlayed < column;
lastPlayed++) {
                        if (rows[lastPlayed] == row ||
Math.abs(rows[lastPlayed] - row) == Math.abs(lastPlayed-column)) {
                                return false;
                        }
                }
                return true;
        }


        static void solution(int column) {
                Scanner kb = new Scanner(System.in);
                int n = kb.nextInt();
                if(n>0 || n<=32) {
                if (column == n) {
                        for (int i = 0; i < n; i++) {
                                System.out.print("[" + (rows[i] + 1) + "]
");
                        }
                        System.out.println();
                        findSolution = true;
                        return;
                }
                for (int row = 0; row < n; row++) {
                        if (valid(row, column)) {
```

```
                        rows[column] = row;
                        solution(column + 1);
                        if(findSolution) {
                                break;
                        }
                }
        }
}


        public static void main(String[] args) throws IOException {
                solution(0);
        }
}
```

Parte 3

Elementos de la Interfaz Grafica

Figura 1. –

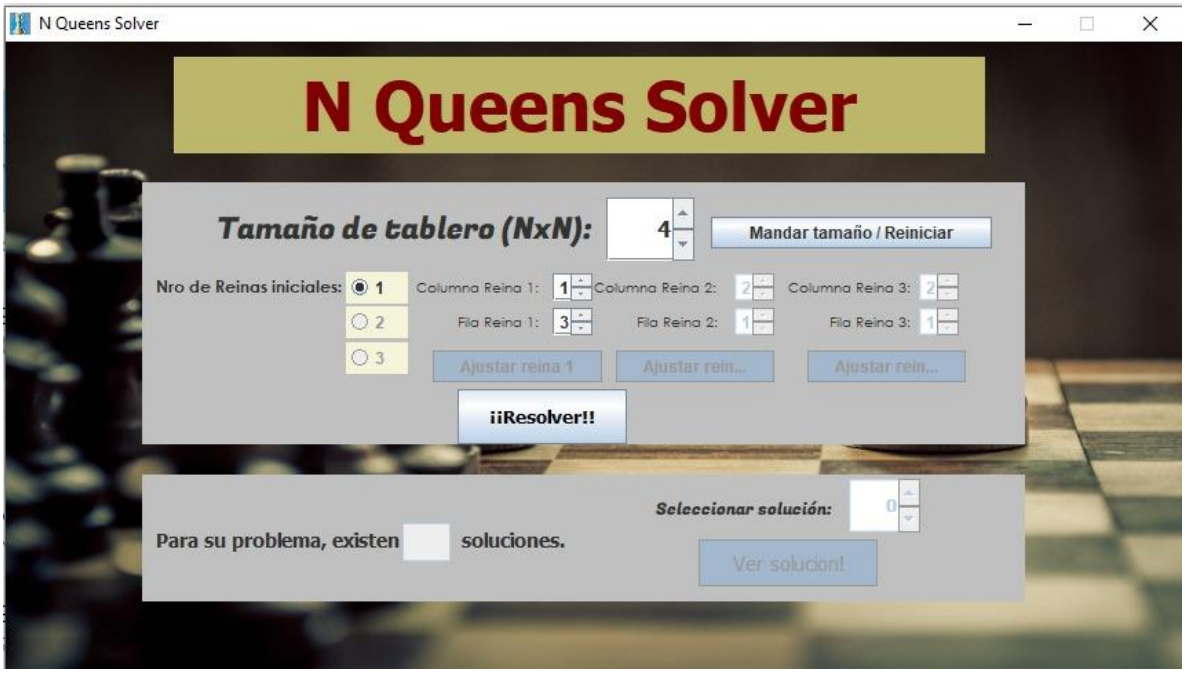

Figura 1.- Interfaz que muestra los niveles y número de filas y columnas con los que desee jugar el usuario.

Figura 2. –

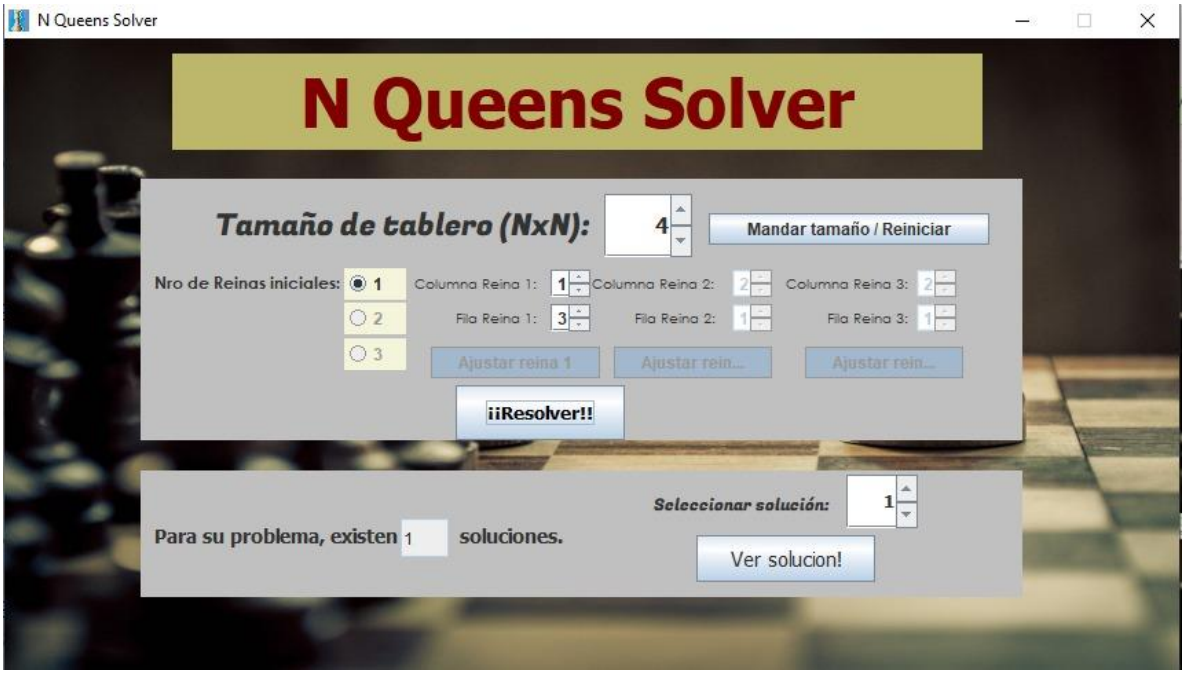

Figura 2.- Interfaz que muestra las opciones escogidas por el usuario y que la solución ya ha sido presentada y aceptada.

Figura 3. -

Figura 3.- Interfaz que muestra el tablero con el número de reinas posibles y la solución aceptada.

Aplicación en funcionamiento

Figura 4. –

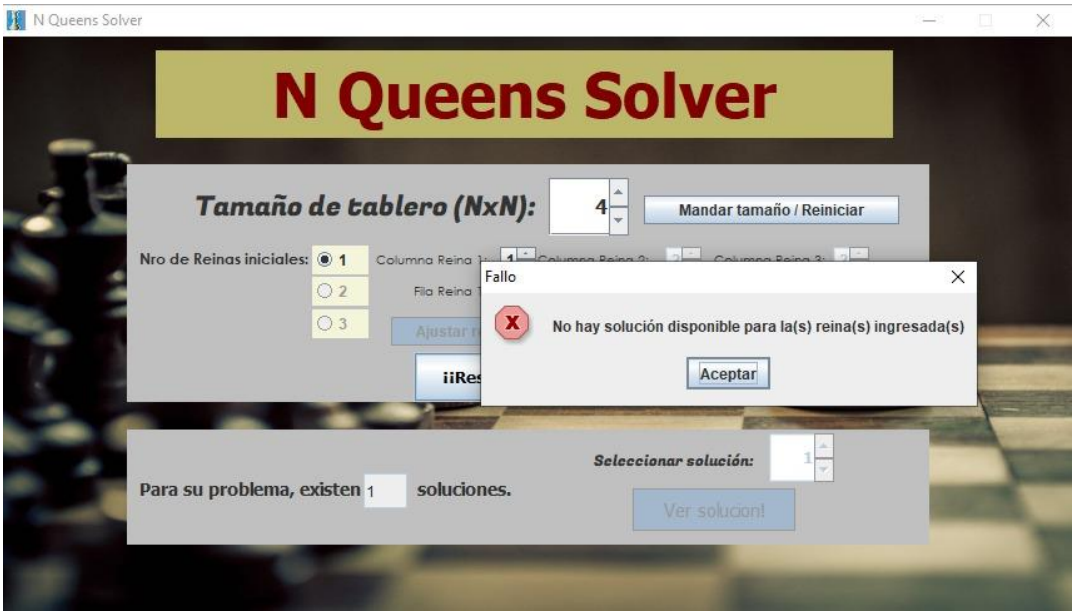

Figura 4.- Interfaz que muestra un error al no tener una solución para el numero de reinas y para las casillas.

Figura 5. –

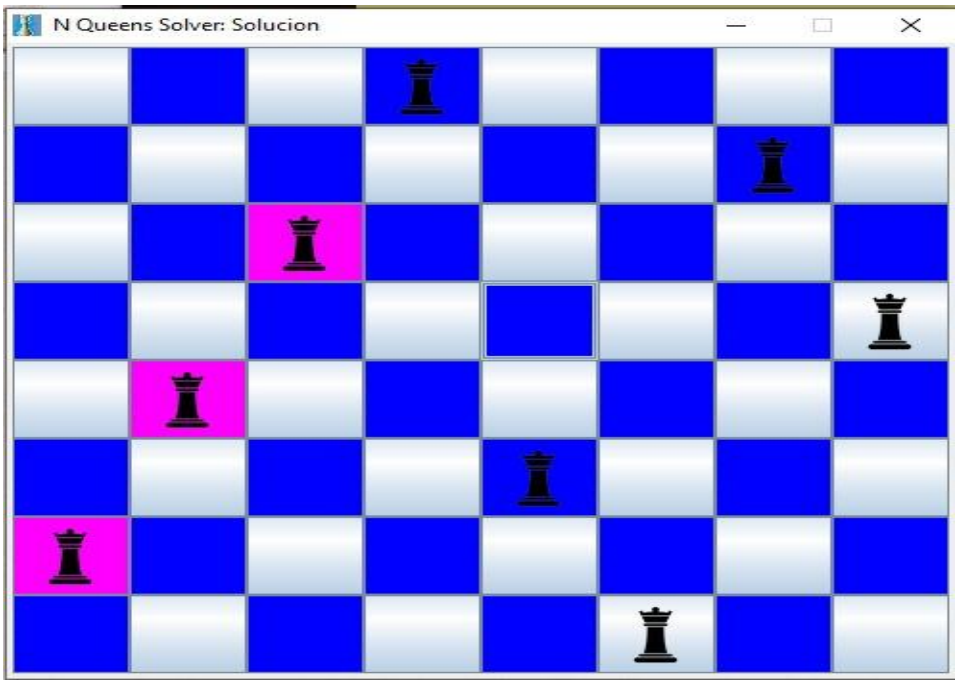

Figura 5.- Interfaz que muestra un tablero mucho más robusto y con un mayor número de soluciones posibles para las reinas.

Figura 6. –

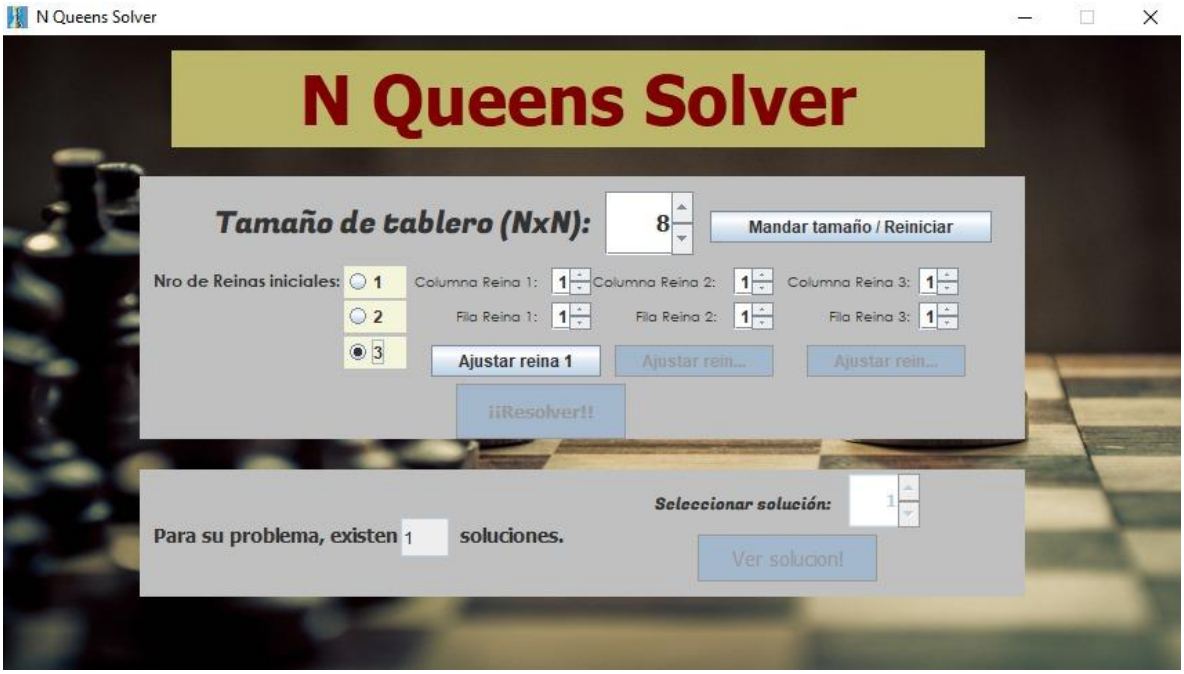

Figura 6.- Menú de interacción principal del usuario para continuar con el juego.

Parte 4

Resultados y Conclusiones

La realización y desarrollo de este proyecto, nos condujo a puntuales y objetivas conclusiones con respecto al problema planteado, su análisis, las condiciones mínimas que debe cumplir, la implementación del algoritmo de resolución backtracking y el desarrollo de una interfaz para la interacción con un usuario y mostrar todas las posibles soluciones. Se presentó un algoritmo rápido que funciona colocando directamente las reinas en el tablero usando esa serie. Empleando una serie de arreglos de reinas que se consideran para resolver el problema que encaja en la serie y una disposición única, que funciona para cualquier valor de n que no encaje en la serie.

Hemos verificado la solidez del algoritmo mediante comprobar el conflicto entre dos o más reinas cualesquiera y considerando el tamaño del tablero. El algoritmo implementado para que resuelva las n-reinas presenta también un problema en un tiempo de ejecución considerablemente menor, ocupando un espacio de O(N). Destacamos con satisfacción que el proyecto, de manera fructífera, logro presentar todas las soluciones posibles dentro de los límites de la colocación directa de reinas.

Lista de referencias (Bibliografía)

- Brassard, Gilles; Bratley, Paul (1997). «Exploración de grafos». Fundamentos de Algoritmia. Madrid: Prentice Hall.
- https://es.wikipedia.org/wiki/Problema_de_las_ocho_reinas
- https://www.techiedelight.com/print-possible-solutions-n-queens-problem/
- R. Sosic, J. Gu, "Fast search algorithms for the n-queens problem", Systems Man and Cybernetics IEEE Transactions on, vol. 21, no. 6, pp. 1572-1576, 1991.
- H.S. Stone and J.M. Stone. “Efficient search
- techniques - An empirical study of the n-queens
- problem”. IBM Journal of Research and Development,
- 31: 464-474, 1987.
- http://www.geocities.ws/certificatessayed/publications/sci_nqueens.pdf
- Antti Laaksonen.2017. Guide to Competitive Programming: Learning and Improving Algorithms Through Contests (Undergraduate Topics in Computer Science) (Inglés) 1st ed.Springer
- https://codeforces.com/blog/entry/70648
- https://www.geeksforgeeks.org/n-queen-problem-backtracking-3/