Universidad modelo

Ingeniería en desarrollo de tecnología y software

Proyecto de las 8 reinas

Joaquin Casillas González

Algoritmos

20/04/23

Codigo de reinas video link:
https://drive.google.com/file/d/1Ce_X_6EpOcEL5Qo6LE_aXymEgXDKZlJK/view?usp=sharing

Codigo 1.

```
using System;

class Programa
{
    static int N = 8; //Sera el tamaño del arreglo bidimensional
    static bool[] fila = new bool[N]; //Se usara para revisar las filas
disponibles
    static int[,] tablero = new int[N, N]; //Tablero
    static bool[] diagonalderecha = new bool[2 * N - 1];
    static bool[] diagonalizquierda = new bool[2 * N - 1];
    static void imprimir()
    {
        for(int i=0; i< N; i++)
        {
            for(int j = 0; j < N; j++)
            {
                if (tablero[i,j] == 1)
                {
                    Console.Write(" Queen ");
                }
                else
                {
                    Console.Write(" - ");
                }
            }
            Console.WriteLine();
        }
    }
    static bool resolver(int columna)
    {
        if(columna >= N)
        {
            return true;
        }
        for(int f = 0; f < N; f++)
        {
            if (!fila[f] && !diagonalderecha[f + columna] &&
!diagonalizquierda[N - 1 - f + columna])
            {
                tablero[f, columna] = 1;
                fila[f]= true;
                diagonalderecha[f + columna] = true;
                diagonalizquierda[N -1 - f + columna]= true;
                if (resolver(columna + 1))
                {
                    return true;
                }
                //--------------------
                tablero[f, columna] = 0;
                fila[f] = false;
                diagonalderecha[f + columna]= false;
```

```
                diagonalizquierda[N-1-f+columna]= false;
            }
        }
        return false;
    }
    static void Main(string[] args)
    {
        Console.WriteLine("Tablero resuelto: \n");
        if (resolver(0))
        {
            imprimir();
        }
        else
        {
            Console.WriteLine("No hay solucion"); //solo necesario para
problemas de nxn en este caso no
        }
    }
}
```

Codigo 2.

```
using System;


class Program
{
   public static int Tamaño = 8;
   public static int[,] tablero = new int[Tamaño, Tamaño];
   public static int[,] tablero2 = new int[Tamaño, Tamaño];
   public static int contador = 0;


   static void Imprimir()
   {
      for (int i = 0; i < Tamaño; i++)
      {
         for (int j = 0; j < Tamaño; j++)
         {
            Console.Write(" - ");
         }
```

```
        Console.WriteLine();
    }
}
static void Realizado()
{
    for (int i = 0; i < Tamaño; i++)
    {
        for (int j = 0; j < Tamaño; j++)
        {
            if (tablero[i, j] == 1)
            {
                Console.Write(" Q ");
            }
            else
            {
                Console.Write(" 0 ");
            }
        }
        Console.WriteLine();
    }
}

static bool VerificarIzquierda(int[,] arreglo, int fila)
{
    int longitud = arreglo.GetLength(1);

    for (int i = 0; i < longitud; i++)
    {
        if (arreglo[fila, i] == 1 && i < longitud - 1 && arreglo[fila, i + 1] == 0)
```

```
        {
            return true;
        }
    }

    return false;
}

static bool VerificarAbajo(int[,] arreglo, int columna)
{
    int longitud = arreglo.GetLength(0);

    for (int i = 0; i < longitud; i++)
    {
        if (arreglo[i, columna] == 1 && i < longitud - 1 && arreglo[i + 1, columna] == 0)
        {
            return true;
        }
    }

    return false;
}

static bool DiagonalI(int[,] arreglo, int fila, int columna)
{
    int tamaño = arreglo.GetLength(0);

    // Verificar diagonal hacia la izquierda inferior
    for (int i = fila + 1, j = columna - 1; i < tamaño && j >= 0; i++, j--)
```

```
        {
            if (arreglo[i, j] == 1)
            {
                return true;
            }
        }

        return false;
    }

    static bool DiagonalS(int[,] arreglo, int fila, int columna)
    {
        int tamaño = arreglo.GetLength(0); //se vuelve a definir, debido a un error que la matriz no coincidia

        // Verificar hacia la izquierda superior
        for (int i = fila - 1, j = columna - 1; i >= 0 && j >= 0; i--, j--)
        {
            if (arreglo[i, j] == 1)
            {
                return true;
            }
        }

        return false;
    }

    static void Algoritmo()
```

```
  {
    for (int i = 0; i < Tamaño; i++)
    {
      for (int j = 0; j < Tamaño; j++)
      {
        if (!VerificarIzquierda(tablero, i) && !DiagonalS(tablero, i, j) && !DiagonalI(tablero, i, j) &&
!VerificarAbajo(tablero, j) && !VerificarIzquierda(tablero, j))
        {
          // colocar la reina
          tablero[i, j] = 1;
        }
      }
    }
  }
  static void verificacion()
  {
    for(int i = 0; i < Tamaño; i++)
    {
      for(int j = 0; j < Tamaño; j++)
      {
        if (tablero[i,j] == 1)
        {
          contador = contador + 1;
        }
      }
    }
    if(contador == 8)
    {
      Realizado();
```

```
        }
        else
        {

        }
    }

    static void Main(string[] args)
    {
        Console.Write("El tablero es: \n");
        Imprimir();
        Algoritmo();
        Console.Write("El tablero con las reinas es: \n");
        Realizado();
    }
}
```

Codigo 3.

```
using System;

class Program
{
    public static int Tamaño = 8;
    public static int[,] tablero = new int[Tamaño, Tamaño];

    static void Imprimir()
    {
        for (int i = 0; i < Tamaño; i++)
        {
            for (int j = 0; j < Tamaño; j++)
```

```
        {
            Console.Write(" - ");
        }
        Console.WriteLine();
    }
}
static void Realizado()
{
    for (int i = 0; i < Tamaño; i++)
    {
        for (int j = 0; j < Tamaño; j++)
        {
            if (tablero[i, j] == 1)
            {
                Console.Write(" Q ");
            }
            else
            {
                Console.Write(" 0 ");
            }
        }
        Console.WriteLine();
    }
}

static bool VerificarIzquierda(int[,] arreglo, int fila)
{
    int longitud = arreglo.GetLength(1);
```

```
    for (int i = 0; i < longitud; i++)
    {
        if (arreglo[fila, i] == 1 && i < longitud - 1 && arreglo[fila, i + 1] == 0)
        {
            return true;
        }
    }

    return false;
}

static bool VerificarAbajo(int[,] arreglo, int columna)
{
    int longitud = arreglo.GetLength(0);

    for (int i = 0; i < longitud; i++)
    {
        if (arreglo[i, columna] == 1 && i < longitud - 1 && arreglo[i + 1, columna] == 0)
        {
            return true;
        }
    }

    return false;
}

static bool DiagonalI(int[,] arreglo, int fila, int columna)
{
    int tamaño = arreglo.GetLength(0);
```

```
    // Verificar diagonal hacia la izquierda inferior
    for (int i = fila + 1, j = columna - 1; i < tamaño && j >= 0; i++, j--)
    {
        if (arreglo[i, j] == 1)
        {
            return true;
        }
    }

    return false;
}

static bool DiagonalS(int[,] arreglo, int fila, int columna)
{
    int tamaño = arreglo.GetLength(0); //se vuelve a definir, debido a un error que la matriz no coincidia

    // Verificar hacia la izquierda superior
    for (int i = fila - 1, j = columna - 1; i >= 0 && j >= 0; i--, j--)
    {
        if (arreglo[i, j] == 1)
        {
            return true;
        }
    }

    return false;
}
```

```
    static void Algoritmo()
    {
        for (int i = 0; i < Tamaño; i++)
        {
            for (int j = 0; j < Tamaño; j++)
            {
                if (!VerificarIzquierda(tablero, i) && !DiagonalS(tablero, i, j) && !DiagonalI(tablero, i, j) &&
!VerificarAbajo(tablero, j) && !VerificarIzquierda(tablero, j))
                {
                    // colocar la reina
                    tablero[i, j] = 1;
                }
            }
        }
    }

    static void Main(string[] args)
    {
        Console.Write("El tablero es: \n");
        Imprimir();
        Algoritmo();
        Console.Write("El tablero con las reinas es: \n");
        Realizado();
    }
}
```