INSTITUTO POLITÉNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

ESTRUCTURAS DE DATOS

PRÁCTICA #5

PROBLEMA DE LAS N-REINAS

GRUPO: 1CM8

EQUIPO: LAS MÁS PERRAS
INTEGRANTES:

- JIMÉNEZ DELGADO LUIS DIEGO 2019630461
- SÁNCHEZ CASTRO AARÓN GAMALIEL 2019630079
- SÁNCHEZ TIRADO CITLALI YASMÍN 2019630096

PROFESOR: EDGARDO ADRIÁN FRANCO MARTÍNEZ

FECHA DE ENTREGA: 13 DE MAYO 2019

INTRODUCCIÓN

En el desarrollo de esta práctica se implemento la teoría del backtracking, estrategia para encontrar soluciones a problemas con restricciones definidos sobre espacios discretos (de elevado tamaño) como es el caso de la práctica de las reinas.

MARCO TEÓRICO

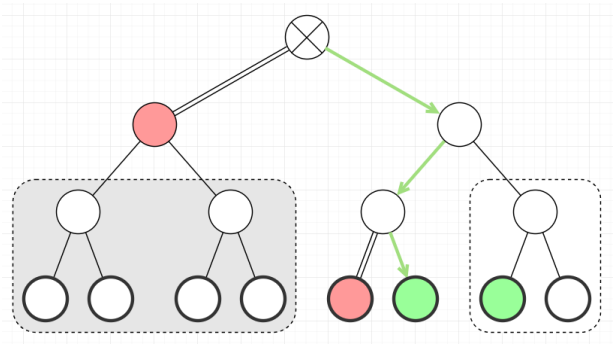
QUÉ ES BACKTRACKING

Cuando un problema no tiene un método algorítmico para resolverse, en general la única forma posible de resolverlo es la búsqueda exhaustiva de soluciones entre todas las posibilidades del problema; este método se conoce como fuerza bruta: se generan todos los casos posibles y se testean uno a uno hasta encontrar las soluciones necesarias (a veces basta con encontrar una, en otras ocasiones hay que encontrar todas ellas, o quedarse con la mejor). Sin embargo, en muchos de estos problemas no es necesario crear completamente un caso para ver si es una solución o no.

Cuando resolver un problema puede hacerse por etapas se puede comprobar paso a paso si se está creando una solución o si se han tomado decisiones que no conseguirán resolver el problema: en cada etapa se estudian las propiedades cuya validez ha de examinarse con objeto de seleccionar las adecuadas para proseguir con el siguiente paso. La gestión de las etapas, las posibles decisiones que se toman y las relaciones entre ellas, suponen la generación de un árbol de decisiones. En los nodos se encuentran las soluciones parciales, y los enlaces entre ellos son las decisiones tomadas para cambiar de etapa. El avance a lo largo del árbol se detiene cuando se llega a una situación en que no puede tomarse ninguna decisión que permita obtener una solución, o cuando efectivamente se llega a resolver el problema. En estos casos se recupera una solución parcial anterior y se continua buscando si es necesario.

Backtracking (o Vuelta Atrás) es una técnica de recursión intensiva para resolver problemas por etapas, que utiliza como árbol de decisiones la propia organización de la recursión. Cuando se “avanza” de etapa se realiza una llamada recursiva, y cuando se “retrocede” lo que se hace es terminar el correspondiente proceso recursivo, con lo que efectivamente se vuelve al estado anterior por la pila de entornos creada en la recursión. Como el árbol de decisiones no es una estructura almacenada en memoria, se dice que Backtracking utiliza un árbol implícito y que se habitualmente se denomina árbol de expansión o espacio de búsqueda. Básicamente, Backtracking es un método recursivo para buscar de soluciones por etapas de manera exhaustiva, usando prueba y error para obtener la solución completa del problema añadiendo decisiones a las soluciones parciales.

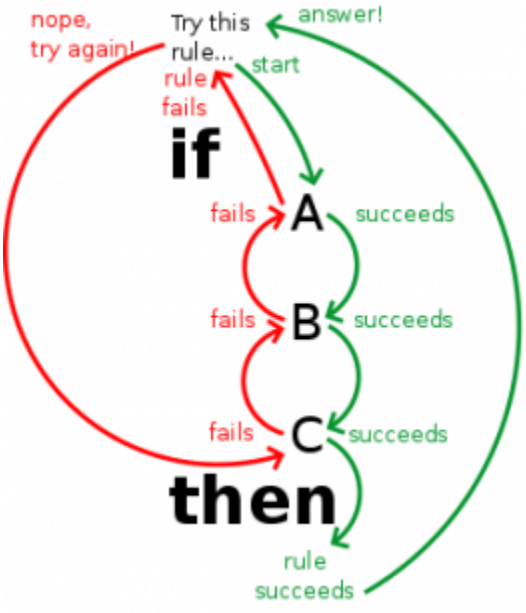
COSTES Y EFICIENCIA EN BACKTRACKING



La forma de generar y escoger entre las distintas posibilidades determina la forma del árbol, la cantidad de descendientes de un nodo, la profundidad del árbol, etc., y determina la cantidad de nodos del árbol y por tanto posiblemente la eficiencia del algoritmo, pues el tiempo de ejecución depende en gran medida del número de nodos que se generen y se visiten. Habitualmente, tendremos como máximo tantos niveles como valores tenga una secuencia solución.

En general, la eficiencia depende de:

- El tiempo necesario para generar las decisiones posibles en el punto
- La cantidad de decisiones posibles que se obtienen
- Incorporar una decisión a una solución parcial en el punto
- El número de soluciones parciales que satisfacen es completable
- Y en mucha menor medida, del tiempo que se tarde en comprobar es solución y tratar solución.



DISEÑO Y FUNCIONAMIENTO DE LA SOLUCIÓN

IMPLEMENTACIÓN DE LA SOLUCIÓN

```
/*
BACKTRACKING N-REINAS
```

```
AUTORES:
JIMÉNEZ DELGADO LUIS DIEGO
SÁNCHEZ CASTRO AARÓN GAMALIEL
SÁNCHEZ TIRADO CITLALI YASMÍN
VERSIÓN 1.5
DESCRIPCIÓN: EL PROGRAMA COLOCA N NÚMERO DE REINAS DADAS POR EL USUARIO EN UN
TABLERO DE DIMENSIÓN NxN
EL OBJETIVO ES POSICIONAR A LAS REINAS DE MANERA QUE NINGUNA SE AMENACE CON EL
RESTO.
*/
#include <stdio.h>
#include <stdlib.h>
#include "Gotoxy.h"
//CONSTANTES A UTILIZAR
#define TRUE  1
#define FALSE 0

//VARIABLES GLOBALES POR COMODIDAD
int i,j,x,y,aux;

/*
FUNCIÓN: validarPosicion(int fila, int posReinas[], int n)
RECIBE: int fila (FILA DE LA REINA A VALIDAR), int[] posReinas (POSICIONES DE LAS
REINAS EN EL TABLERO)
int n (NÚMERO DE REINAS INGRESADO POR EL USUARIO)
DEVUELVE: TRUE/FALSE SEGÚN SEA EL CASO.
DESCRIPCIÓN: LA FUNCIÓN VALIDA QUE LA REINA ESTÉ BIEN COLOCADA, SE COMPRUEBA QUE
PARA CUALQUIER FILA i
UNA REINA NO ESTÉ EN LA MISMA COLUMNA (SE COMPRUEBA QUE TODOS LOS NÚMEROS EN EL
ARREGLO SENA DIFERENTES).
ADEMÁS, EL PROBLEMA DE LAS DIAGONALES ASCENDENTES/DESCENDENTES SE ARREGLA
REALIZANDO LA OPERACIÓN
fila-columna, ESTA OPERACIÓN DEBE TENER UN RESULTADO DIFERENTE PARA CADA PAR
ORDENADO DEL ARREGLO.
OBSERVACIONES:
*/
int validarPosicion(int fila,int posReinas[],int n){
for(i=0; i<fila; i++){
//SI posReinas[i]==posReinas[fila] ENTONCES ESTÁN EN LA MISMA COLUMNA
//SI posReinas[fila]-posReinas[i]=(fila-columna) ENTONCES ESTÁN EN LA MISMA
DIAGONAL
if((posReinas[i]==posReinas[fila])||(abs(fila-i)== abs(posReinas[fila]-
posReinas[i]))){
return FALSE;
}
}
return TRUE;
}
```

La función *validarPosicion* recibe la fila en la que se ubica la reina agregada, mediante una sentencia for para recorrer el arreglo de posiciones se comprueba si dicha fila es una posición válida según la condición establecida en el problema.

```
/*
FUNCIÓN: mostrarTablero( int posReinas[], int n, int *alto)
RECIBE: int[] posReinas (POSICIONES DE LAS REINAS EN EL TABLERO)
int n (NÚMERO DE REINAS INGRESADO POR EL USUARIO), int* alto (ORDENADA PARA SITUAR
EL CURSOR E IMPRIMIR)
DEVUELVE: NADA.
```

```
DESCRIPCIÓN: IMPRIME EL TABLERO CON LAS POSICIONES ALMACENADAS EN EL ARREGLO AL
MOMENTO.
OBSERVACIONES: PARA CASOS DONDE n (NÚMERO DE REINAS) ES "GRANDE" (P.E: 10) ES
NECESARIO QUE EL TAMAÑO DEL BÚFER
DE PANTALLA SEA IGUALMENTE GRANDE PARA EVITAR UN DESBORDAMIENTO Y LA PÉRDIDA DE LOS
GRÁFICOS.
*/
void mostrarTablero(int posReinas[],int n,int*alto)
{
x=0;
y=*alto;
aux=0;
for(i=0; i<n; i++){
x=0;//SERÁ REUTILIZADA, POR LO TANTO ES NECESARIO REINICIAR EL CONTADOR
for(j=0; j<n; j++){
if(posReinas[i]==j){//SE COLOCA UNA REINA
MoverCursor(x,y);
printf("%c",219);
}else{
if((j+aux)%2==0){//SE COLOCA UN "ESPACIO BLANCO"
MoverCursor(x,y);
printf("%c",176);
}else{
MoverCursor(x,y);
printf("%c",176);
}
}
x+=2;
}
y+=1;
aux=1-aux;
*alto = y+2;
}
printf("\n");
}
```

La función *mostrarTablero* imprime el tablero gracias al arreglo de posiciones, mediante una sentencia *for* se recorre dicho arreglo, se coloca el cursor en las coordenadas correspondientes (x,y o alto) y se imprime el caracter correspondiente. Posteriormente se hacen los respectivos incrementos a las variables (x,y).

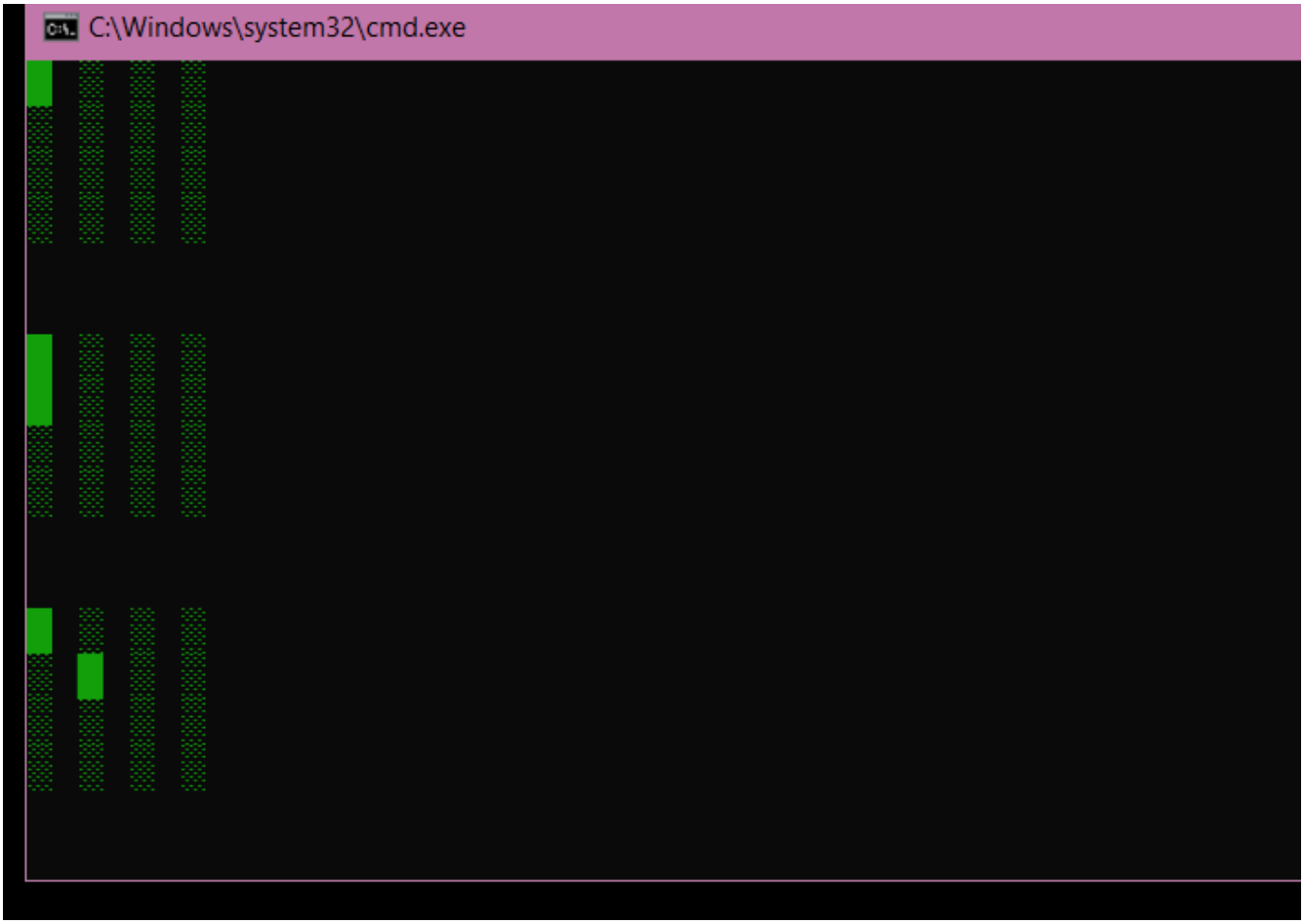
```
/*
FUNCIÓN: Reina(int fila, int posReinas[], int *alto).
RECIBE: int fila (FILA DONDE SE COLOCARÁ A LA NUEVA REINA), int[] posReinas
(TABLERO DE POSICIONES DE LAS REINAS)
,int* alto (ORDENADA PARA SITUAR EL CURSOR).
DEVUELVE: LLAMA A LA FUNCIÓN NUEVAMENTE, SEGÚN SEA EL CASO.
DESCRIPCIÓN: COLOCA UNA REINA EN EL TABLERO. LLAMA A LA FUNCIÓN validarPosicion
CADA VEZ QUE LO HACES
OBSERVACIONES:
*/
void Reina (int fila,int posReinas[],int n,int*alto)
{
int ok = FALSE;
if(fila<n){
// Quedan reinas por colocar
for(posReinas[fila]=0;posReinas[fila]<n;posReinas[fila]++){
//COMPROBAMOS SI LA POSICIÓN ES VÁLIDA ¿PUEDO MANDAR A IMPRIMIR?
mostrarTablero(posReinas,n,alto);
EsperarMiliSeg(400);
```

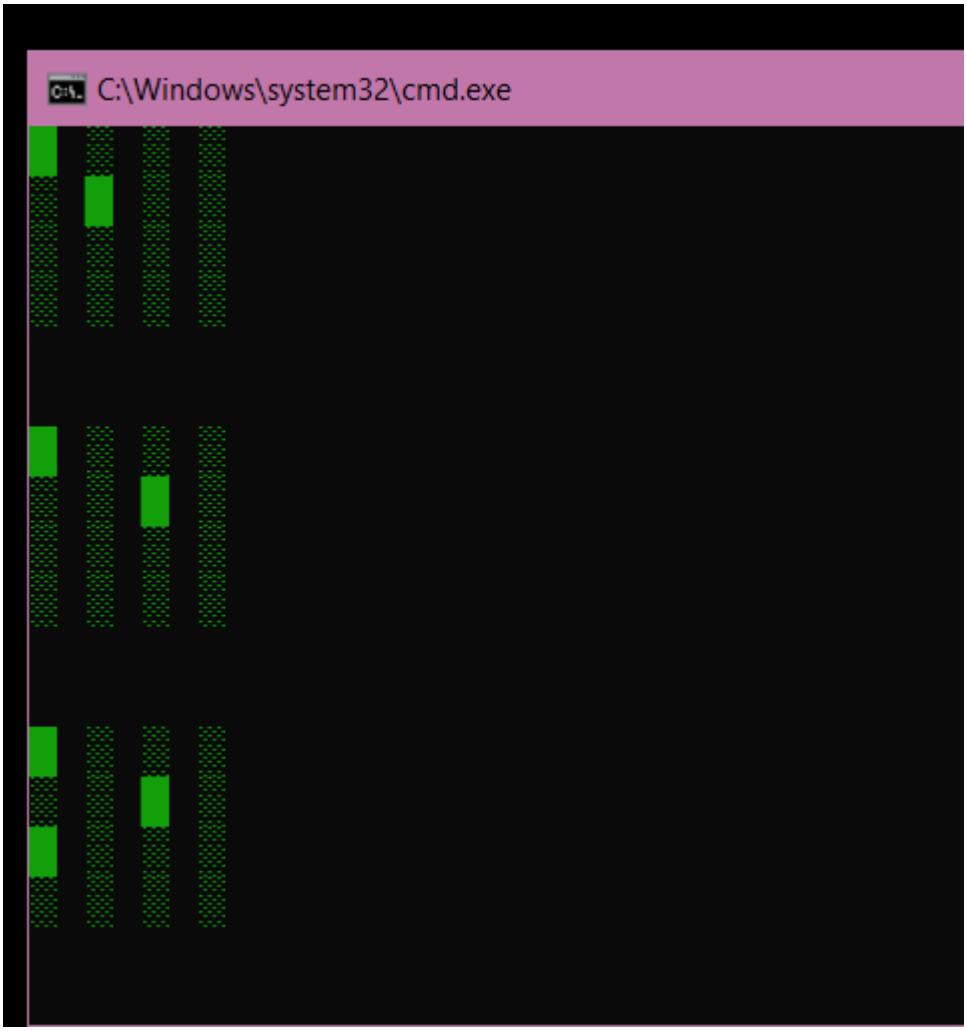
```
if(validarPosicion(fila,posReinas,n)){
//SI LA POSICIÓN ES VÁLIDA, PROCEDEMOS A COLOCAR LA(S) SIGUIENTE(S) REINA(S)
Reina(fila+1,posReinas,n,alto);
}
//AQUÍ SE HACE EL BACKTRACKING, EN CASO DE NO SER VÁLIDA, SIMPLEMENTE NO SE LLAMA A
LA FUNCIÓN Y SE SIGUE EL for
}
}else{
printf("SOLUCION");
}
return;
}
```

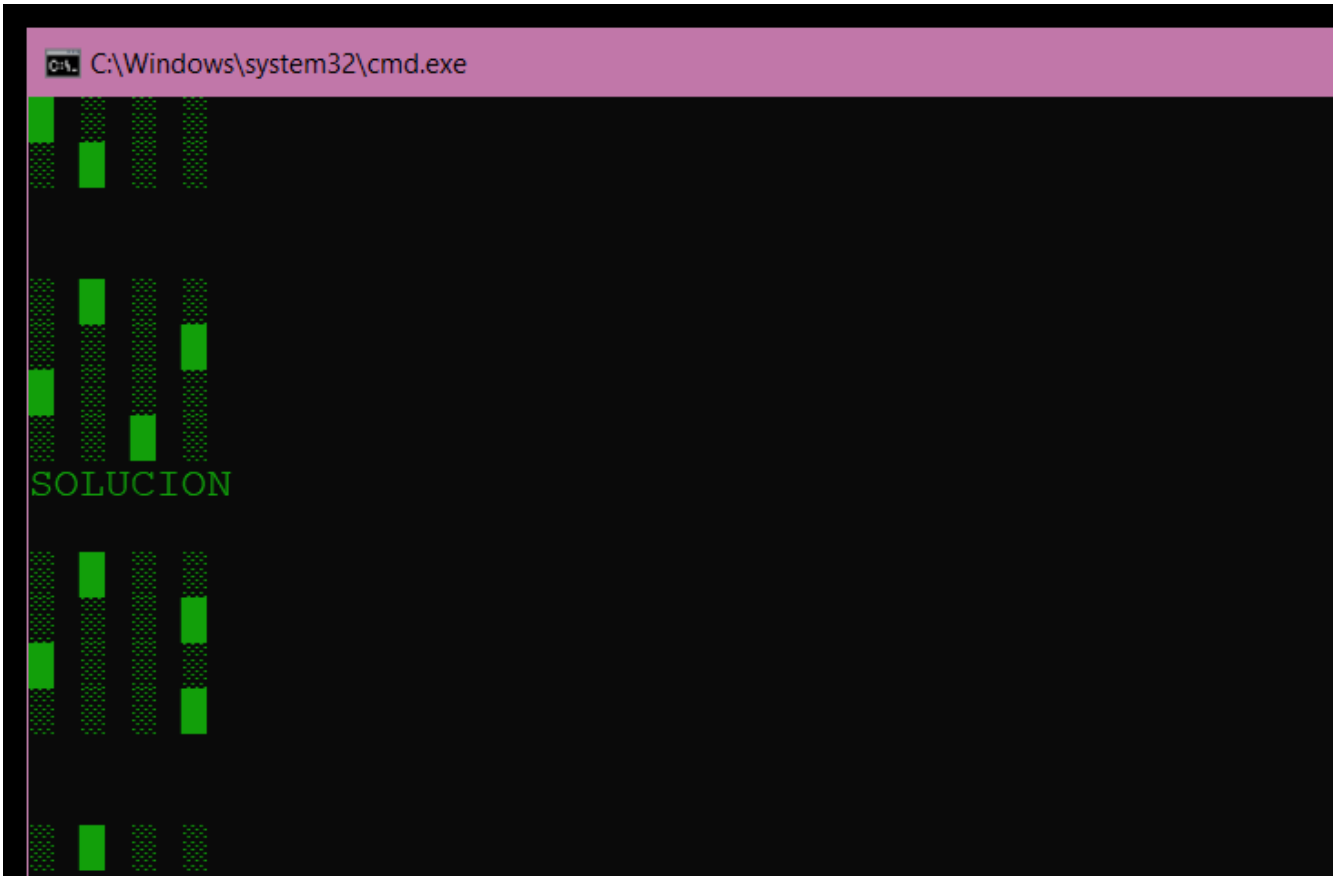
La función *Reina* es una función recursiva, que en caso de ser válida la posición para una reina agregada al arreglo se llama a sí misma para agregar una nueva reina al tablero. En caso de que el número de reinas agregadas sea igual al número de reinas por agregar y todas las posiciones sean válidas, se dice que se ha encontrado una solución al problema.

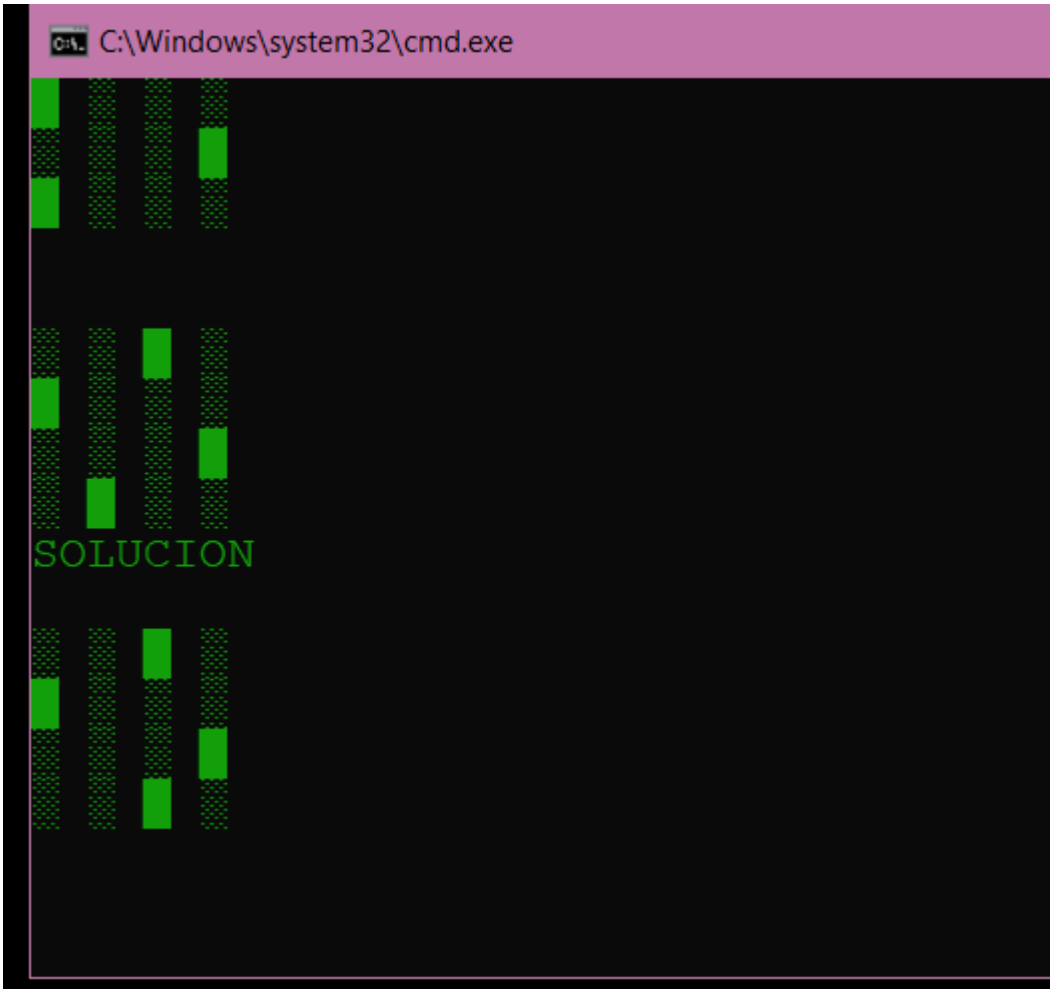
```
int main (){
int numReinas;// Número de reinas
int alto=0;
printf("--------------------\nBIENVENIDO\n--------------------\n");
printf("Ingresa la cantidad de reinas a colocar (4<=numero<=10):\n>");
scanf("%i",&numReinas);
int posReinas[numReinas];
system("cls");
//INICIALIZAMOS EL ARREGLO CON UN VALOR NEGATIVO (NO ENTRA EN EL RANGO DEL ARREGLO)
PARA FACILITAR SU MANIPULACIÓN
for(i=0; i<numReinas; i++)
posReinas[i]=-1;
//COMENZAMOS CON LA RECURSIVIDAD PARA COLOCAR LAS REINAS
Reina(0,posReinas,numReinas,&alto);
}
```

FUNCIONAMIENTO




C:\Windows\system32\cmd.exe


C:\Windows\system32\cmd.exe
SOLUCION



ERRORES DETECTADOS

Como observación se tiene que el programa puede limpiar la pantalla cada vez que imprima un nuevo tablero.

POSIBLES MEJORAS

El ingresar un número de reinas muy grande involucra un gran número de soluciones para el tablero, por lo tanto, el tiempo de ejecución se prolonga indefinidamente. Es posible modificar la manera en la que el programa imprime el camino para llegar a la solución, imprimiendo un solo tablero y limpiando la pantalla.

CONCLUSIONES

Jiménez Delgado Luis Diego: La recursividad es un tema de bastante interés y que facilita demasiado la solución de diferentes algoritmos de programación. Los métodos de búsqueda de información, ordenamiento y de solución son fundamentales de conocer en esta rama de la computación, ya que se usa en todos los lugares posibles.

Sánchez Castro Aarón Gamaliel: Gracias a los conocimientos adquiridos de la práctica anterior es posible dar una solución recursiva para este problema utilizando la técnica *backtracking.* Una observación importante es que la eficiencia del algoritmo es directamente proporcional al tamaño del tablero especificado.

Sanchez Tirado Citlalli Yasmin:
En está practica me di cuenta lo útil que puede ser el backtracking ya que resuelve los problemas por fuerza bruta, esto tiene ciertas desventajas y ventajas como todo. El backtracking es un esquema algorítmico que nos va a permitir resolver una gran variedad de tipos de problemas, pero, por término medio, es sumamente costoso (en cuanto a complejidades temporales) debido al tamaño que adoptan sus espacios de búsqueda. Aplicándole una serie de mejoras podremos conseguir que dicho espacio para reducir en lo posible las altas complejidades que suelen tener.