INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Estructura de Datos



Practica 5

Prof.: Franco Martínez Edgardo Adrián

Integrantes:

Mendoza Parra Sergio.

Salcedo Barrón Ruben Osmair.

Tejeda Martínez José Miguel.

MEXICO, D.F. a 9 de Enero del 2016

**Introducción:**

En este programa se implementa el algoritmo de backtracking con el problema de las “N-Reinas” que dependiendo el número de reinas que un usuario ingrese, no se deben de tocar o poner en el mismo renglón que las demás reinas. Y a su vez se implementa el backtracking para que si una reina llega a estar en un renglón donde hay otra reina se hace el regreso de posición y hasta que se resuelva el problema.

**Planteamiento del problema:**

Lo que se hizo primeramente fue hacer varios ejemplo con papel y lápiz y buscar varios ejemplos en para que nos diéramos una idea de cómo se iban a hacer o se iban a poner cada reina y con base a eso implementar una manera en la cual pudiéramos resolver el problema de las reinas.

Lo que utilizamos fue una matriz ya que así pudimos ver o verificar en que posiciones están las reinas y hacer el backtracking.

**Diseño y funcionamiento de la solución:**

**“Problema de la N-Reinas”**

**“Backtracking”**

En base al programa y en base al planteamiento del problema, tomamos en cuenta la parte del backtracking ya que lo que hicimos como ya lo habíamos mencionado fue:

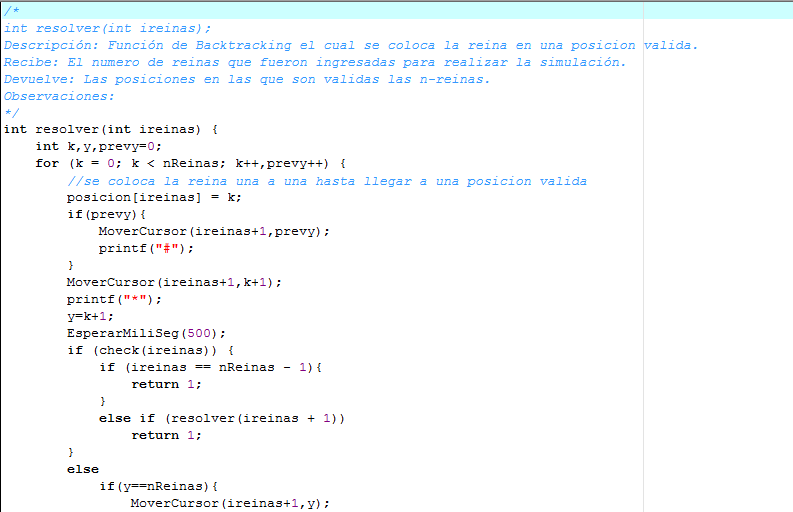
1. Ver ejemplos de cómo se ponían cada reina
2. Y buscar de los que se trataba el problema

El diseño que le pusimos fue con base a una matriz la cual al momento de que se ingresa un numero de reinas se puede ver la resolución del problemas en forma de matriz, su funcionamiento es en forma de tipo el juego del gato porque lo hicimos basándonos en eso las posiciones en las que se mueve y pues solo hacer lo mismo pero que se regrese a la posición en la que no debe de ser colocado.

También en la parte del funcionamiento se le puede dar un numero cualquiera de reinas y el programa realizara la resolución del problema puede que en algunos casos no se pueda resolver, pero si es muy eficiente el programa.

**Implementación de la solución en el código:**

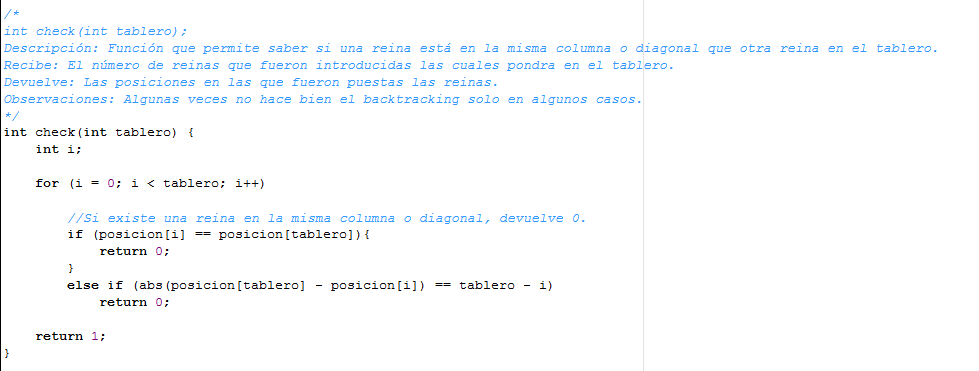
**Backtracking**

****

****

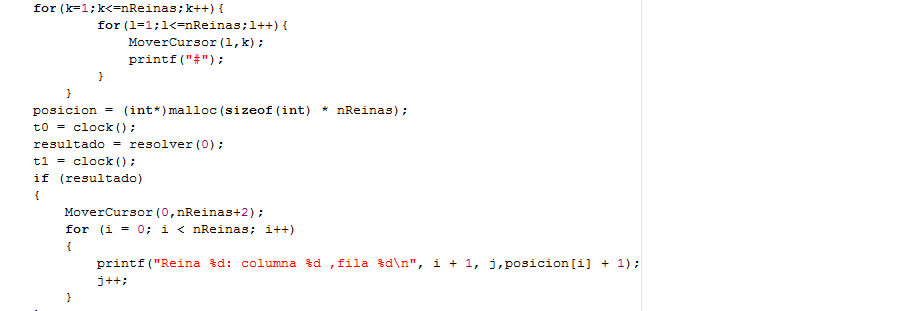
Ésta es la función del Backtracking en la cual dependiendo el numero de reinas que se ingresen va a hacer la función del Backtracking se puede ver que hay un contador de pasos que es la variable “prevy” la cual va a estar contando el numero de pasos que da cada reina, una vez que se verifica la posición que está en la condición “check” que como su nombre lo dice checa la posición de la reina de tal manera que no se pongan en la misma posición las reinas, de tal manera que una vez checando la posición de la reina hay dos casos que son: El backtracking si una reina esta en el mismo renglón que otra reina y la otra condición que es para seguir movimiendo la reina hasta que haya una incongruencia o no, si la hay cae en la condición del Backtracking y si no pues se sigue moviendo la reina hasta que quede en un posición que sea válida.

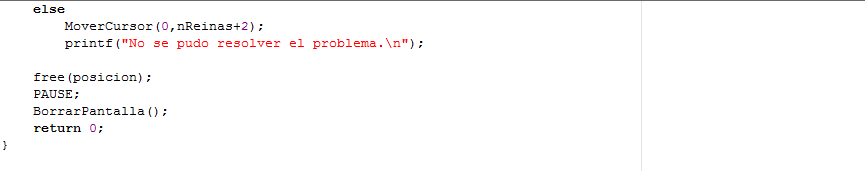
**Checar posiciones en el tablero.**



Aquí lo que podemos ver es que si la posición en la que está una reina es igual a la misma posición en el tablero de otra reina se regresa a la función del Backtracking de tal manera que se regresa “i” posiciones para que ya no haya una incongruencia.

**Posiciones en el tablero.**

****

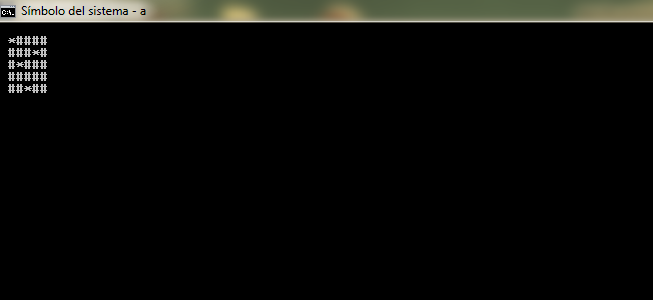
****

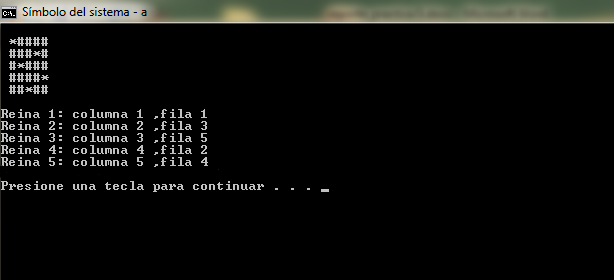
Aquí lo que podemos observar es que una vez ingresado el numero de reinas que se quiere resolver se pinta la parte del tablero primero como se puso “Movercursor(x,y)”, después se reserva en memoria dinámica el tamaño de las reinas que se vayan a ingresar y después se hace el tiempo con forme a la operación “ clock (); ” y por último se ponen los pasos que se hicieron y por lo contrario si no se pudo resolver el problema solo se ponen las incongruencias de posiciones y sale un mensaje de que no se pudo resolver el problema.

**Funcionamiento**

Funcionamiento del programa:

Aquí se puede ver que una vez ingresado el numero de reinas se empieza a resolver el problema de las “N-REINAS” de tal manera que como se puede ver en la imagen se empiezan a poner las reinas en posiciones que pueden estar sin tocar o estar en la misma fila o renglón de una reina.



Después una vez que se ubicaron se ponen las coordenadas en las que quedaron cada reina. 

**Errores detectados**

Los errores detectados los siguientes:

1. En algunas ocasiones al final de que se resuelve el problema no se porque se pone el mensaje de que no se pudo resolver el problema
2. En otras ocasiones hay incongruencias porque a veces las reinas se ponen en la misma fila o columna que las demás reinas pero solo en algunos casos.

**Posibles mejoras**

Las posibles mejoras que se podrían poner en el programa serian las siguientes:

1. Las incongruencias ya que se pueden checar las posiciones en las que se mueven las reinas.
2. El mensaje de que no se pudo resolver también se puede omitir y solo al momento en el que se resuelva el problema si no se puede resolver solo tener o que se pongan las incongruencias de las posiciones.

**Anexos**

(Código Fuente):

/\*

PROBLEMA DE LAS N-REINAS

AUTOR: Mendoza Parra Sergio, Salcedo Barron Ruben Osmair, Tejeda Martinez Jose Miguel (C) Diciembre 2015

VERSIÓN: 1.0

DESCRIPCIÓN: Con la implementación de "Backtracking"

realizar la implementación de las N-Reinas,

el problema de las N-Reinas consiste en colocar n reinas en

un tablero de ajedrez de tamaño N\*N de forma la reinas no

se amenacen según las normas del ajedrez. Se busca

encontrar una solución o todas las soluciones posibles.

OBSERVACIONES:

COMPILACIÓN: gcc reinas.c presentacionWin.c -o "Nombre del ejecutable"

EJECUCION: ./"Nombre del ejecutable" (Linux)

"Nombre del ejecutable".exe (Windows)

\*/

//Librerias

#include <stdlib.h>

#include <stdio.h>

#include <time.h>

#include "presentacion.h"

#define PAUSE system("pause")

static int nReinas = 0; // Numero de reinas

static int \*posicion = 0; // Posicion en el ajedrez (empieza en 0)

// ireinas : Número de la primera reina de colocar (profundidad del gráfico)

int resolver(int ireinas);

/\*

Compruebe si la reina en la posición "tablero" se las reinas anteriores bien situada respecto.

\*

\* Tablero: Indice de la reina estamos colocando (profundidad del gráfico).

\*

\* Devuelve 1 si está bien situado a la reina en la posición "tablero",

otra cosa devuelve 0.

\*/

int check(int tablero);

int main(void)

{

int resultado = 1, i,j=1,k,l;

clock\_t t0, t1;

/\*Se debe ingresar un numero mayor a 0 de reinas de lo contrario

no se realizara la simulación\*/

while (nReinas <= 0 || resultado == 0) {

printf("Ingrese cuantas reinas : ");

resultado = scanf("%d", &nReinas);

system("cls");

}

for(k=1;k<=nReinas;k++){

for(l=1;l<=nReinas;l++){

MoverCursor(l,k);

printf("#");

}

}

posicion = (int\*)malloc(sizeof(int) \* nReinas);

t0 = clock();

resultado = resolver(0);

t1 = clock();

if (resultado)

{

MoverCursor(0,nReinas+2);

for (i = 0; i < nReinas; i++)

{

printf("Reina %d: columna %d ,fila %d\n", i + 1, j,posicion[i] + 1);

j++;

}

}

else

MoverCursor(0,nReinas+2);

printf("Se resolvio el problema.\n");

free(posicion);

PAUSE;

BorrarPantalla();

return 0;

}

/\*

int resolver(int ireinas);

Descripción: Función de Backtracking el cual se coloca la reina en una posicion valida.

Recibe: El numero de reinas que fueron ingresadas para realizar la simulación.

Devuelve: Las posiciones en las que son validas las n-reinas.

Observaciones:

\*/

int resolver(int ireinas) {

int k,y,prevy=0;

for (k = 0; k < nReinas; k++,prevy++) {

//se coloca la reina una a una hasta llegar a una posicion valida

posicion[ireinas] = k;

if(prevy){

MoverCursor(ireinas+1,prevy);

printf("#");

}

MoverCursor(ireinas+1,k+1);

printf("\*");

y=k+1;

EsperarMiliSeg(500);

if (check(ireinas)) {

if (ireinas == nReinas - 1){

return 1;

}

else if (resolver(ireinas + 1))

return 1;

}

else

if(y==nReinas){

MoverCursor(ireinas+1,y);

printf("#");

}

}

return 0;

}

/\*

int check(int tablero);

Descripción: Función que permite saber si una reina está en la misma columna o diagonal que otra reina en el tablero.

Recibe: El número de reinas que fueron introducidas las cuales pondra en el tablero.

Devuelve: Las posiciones en las que fueron puestas las reinas.

Observaciones: Algunas veces no hace bien el backtracking solo en algunos casos.

\*/

int check(int tablero) {

int i;

for (i = 0; i < tablero; i++)

//Si existe una reina en la misma columna o diagonal, devuelve 0.

if (posicion[i] == posicion[tablero]){

return 0;

}

else if (abs(posicion[tablero] - posicion[i]) == tablero - i)

return 0;

return 1;

}

**Conclusiones**

**Tejeda Martínez José Miguel**: El problema de las N-Reinas es un problema clásico en el uso de backtracking; lo que pudimos observar al desarrollar este programa es que solo existe una solución para cada caso, y que cada uno de los elementos (en este caso las reinas), son colocadas en el tablero de ajedrez de manera que no se amenacen entre sí.

En la ejecución y simulación del problema nos dimos cuenta que existen distintas configuraciones para 1, 2,…, n-1 reinas, pudiendo cumplirse la condición de que sus vectores de desplazamiento no coincidan entre sí; el punto esencial se encuentra en el término N, ya que es en este donde se deben cumplir la condición; de otro modo, el programa “deshace” los pasos o acciones realizadas hasta que exista un resultado válido.

Aunque este ejemplo podemos catalogarlo como “juego o pasatiempo”, la implementación de backtracking para resolver este y muchos otros problemas nos ayuda a encontrar soluciones intentando por todos los caminos posibles hasta hallar un resultado válido.

**Mendoza Parra Sergio**: En ésta práctica del problema de las N-Reinas fue un poquito engorroso la parte de estar checando las posiciones en las que eran validas las reinas de tal modo que las reinas no tenían que estar ni en la misma diagonal o en la misma fila, utilizamos lo que era una matriz la cual una vez que se checaba las posiciones de una reina la otra obviamente, si era igual a la posición en la que estaba la otra reina (la fila), ese paso ya no se hacía y se saltaba al siguiente de tal manera que se acomodaran. Por otra parte tuvimos algunos problemas en la parte de que a veces se encimaban las reinas porque a veces las reinas se ponían en posiciones erróneas y eran incorrectas las posiciones en las que se ponían.

En mi punto de vista fue una práctica en la que yo lo asocie como una toma de decisiones en las que sí, se iba de un lado y no funcionaba se regresaba al paso en el que no había estado para que tuviera una forma en la cual se resolviera el problema o fuera la mejor opción.

**Salcedo Barrón Ruben**: En esta práctica se vio backtracking el cual vimos que es parte de recursividad pero en bruto para cierto casos y en este problema de las N-reinas el cual pudimos investigar que para hacerlo de n-reinas vimos que para cada número de reinas había una solución diferente y que en algunas situaciones había 2 o más soluciones, una vez partiendo de eso tuvimos que hacerlo gráfico y no fue tan difícil pero fue un poco sencillo ya que tuvimos un problema cuando se mostraba gráficamente ya que se ponían 2 reinas en una columna por el bactracking pero ya solucionado solo fue acomodar bien las cosas y decir en que columna y fila esta la reina.

# BIBLIOGRAFÍA

#### [1] Edgardo Adrián Franco Martínez, Practica 05: ["El problema de las N-Reinas"](http://eafranco.com/docencia/estructurasdedatos/files/practicas/Practica05.pdf) Recursos Adicionales:[Juega Reinas](http://eafranco.com/docencia/estructurasdedatos/files/practicas/05/8reinas.swf), Noviembre de 2015. [En línea]. Disponible en: <http://eafranco.com/docencia/estructurasdedatos/files/practicas/Practica05.pdf>