# Definición del Problema

El sudoku es un pasatiempo que se publicó por primera vez a finales de la década de 1970 y se popularizó en Japón en 1986, dándose a conocer en el ámbito internacional en 2005 cuando numerosos periódicos empezaron a publicarlo en su sección de pasatiempos. 1 El objetivo del sudoku es rellenar una cuadrícula de 9 × 9 celdas (81 casillas) dividida en sub-cuadrículas de 3 × 3 (también llamadas "cajas" o "regiones") con las cifras del 1 al 9 partiendo de algunos números ya dispuestos en algunas de las celdas. Aunque se podrían usar colores, letras, figuras, se conviene en usar números para mayor claridad, lo que importa, es que sean nueve elementos diferenciados, que no se deben repetir en una misma fila, columna o sub-cuadrícula. Un sudoku está bien planteado si la solución es única. La solución de un sudoku siempre es un cuadrado latino, aunque el recíproco en general no es cierto ya que el sudoku establece la restricción añadida de que no se puede repetir un mismo número en una región.

Tabla de contenido

[Definición del Problema 1](#_Toc357721633)

[Análisis 1](#_Toc357721634)

[Diseño 2](#_Toc357721635)

[Código Fuente 10](#_Toc357721636)

[Condiciones de Error 23](#_Toc357721637)

[Ejemplos de Uso 24](#_Toc357721638)

[Notas al Profesor 25](#_Toc357721639)

# Análisis

Existen varios métodos para los humanos resolver sudokus, pero para un computador la manera más eficiente es el backtrackin, que es un tipo de fuerza bruta no tan bruta que mediante prueba y error verifica cueles "caminos" puede seguir y cuales no, asi hace que el proceso no sea tan lento y que no sea nescesario probar con todas las soluciones posibles y así encontrar la solución mas rápidamente.

# Diseño

PSEUDOCODIGO

Inicio

Declarar comprobarfila

Declarar comprobarcolumna

Declarar comprobarcuadro

Declarar sudoku

Declarar bactracking

Declarar matriz\_3x3

Declara mostrar

Declarar caracter opcion=1;

Declarar struct datos: caracter nombre, carácter apellido, entero num, destreza

Funcion Main

Declarar datnombre tipo datos

Declarar i,j,aceptado, matriz[9][9],filas[9]

Establecer aceptado a 1, contador a 0

Imprimir "BUSCADOR SOLUCION FACTIBLE-SUDOKU"

Imprimir "Digite su nombre : "

Leer datnombre.nombre, datnombre.apellido

Imprimir "Especifique su nivel de destreza en SUDOKU ""0: Principiante,1: Intermedio, 2: Experto"

Leer datnombre.destreza

Mientras (datnombre.destreza sea diferente de0 y atnombre.destreza sea diferente de 1 y datnombre.destreza sea diferente de 2)

Imprimir "Digite una opcion valida:"

Leer datnombre.destreza

Fin mientras

Mientras (aceptado sea igual a 1)

Imprimir "Digite los valores de la matriz inicial del SUDOKU:"

Imprimir"(Use el cero para indicar una casilla libre):"

Para i=0 hasta 9

Imprimir " Fila "i

Leer fila

Para j=0 hasta 9

filas[j] - = '0'

matriz[i][j]=filas[j];

Fin para

Fin para

Declarar cont =0

Establecer aceptado a -1

Si sudoku(matriz) retorna 1

Establecer aceptado a 1

Fin si

Si aceptado equivale a-1

Imprimir "Los valores no son validos"

Imprimir "Desea intentar otra vez," datnombre.nombre " ? (s/n):"

Establecer aceptado a 1;

Leer opcion

Si opcion equivale a 's' {

Establecer aceptado a 1;

Continuar

Fin si

Fin si

Si no

Establecer aceptado a -1

Fin sino

Fin si

Si aceptadoequivale a 1

Incrementar contador a 1;

Imprimir " El sudoku parece ser valido, procediendo a su solucion..."

aceptado=1;

Imprimir "Desea mostrar por animación el algoritmo (S/N)? "

Leer opción

Imprimir" Muy bien" datnombre.nombre,datnombre.apellido", mediante el algoritmo de Vuelta Atras, se ha

Encontrado la siguiente solución factible al SUDOKU digitado:"

Llamar bactracking con matriz y 0

Mostrar matriz

Imprimir " Usted ha realizado " contador " búsqueda de solución factible."

Imprimir "Desea realizar otra búsqueda? (S/N): ");

Leer opción

Si opción equivale al carácter 'S

Establecer aceptado a 1

Si no

Establecer aceptado a -1

Fin si

Si aceptado equivale a -1

Declarar carácter hora[6]

Declarar t tipo time\_t

Declarar Tiempo tipo punter a struct tm ;

Establecer t a time(0)

Establecer tiempo=localtime(&t);

Imprimir string de time " hora " tiempo

Imprimir "Son las"hora

Imprimir "!que tenga feliz resto del dia " datnombre.nombre,datnombre.apellido"! Gracias por utilizar esta

herramienta"

retornar 0;

Fin si

retornar 0

Fin main

Funcion funcion entera bactracking (int matriz[9][9],posicion)

Declarar k;

Si posicionequivale a 82

retornar 1

fin si

Declarar i , j

Establecer i a posicion/9, j aposicion%9

Si matriz[i][j] diferente de 0

retornar bactracking(matriz,posicion+1)

Fin si

Para k=1 hasta9

matriz[i][j] igual a k;

Si sudoku(matriz)==1

Si bactracking(matriz,posicion+1)equivale a 1

retornar 1

Fin si

Fin si

matriz[i][j]=0;

Fin para

retornar 0

Fin funcion

Funcion sudoku (int matriz[9][9])

Declarar i

Establecer i a 0

Mientras(comprobarcolumna(matriz,i) equivalga a 1 y comprobarfila(matriz,i) equivale1)

Y comprobarcuadro(matriz)equivale a 1 y i menor que 9)

Aumentar i a 1

Fin mientras

Si(i==9)

Retornar 1

sino

rettornar 0

Fin si

Fin funcion

Funcion comprobarfila(int arr[9][9], int fila)

Declarar i,j;

Declarar copyfilas[9];

Para i=0 hasta 9

copyfilas[i]=arr[fila][i];

Fin para

para i=0 hasta 9

para j=i+1 hasta 9

si(copyfilas[j] equivale a arr[fila][i] y arr[fila][i] diferente de 0)

retornar 0

Fin si

Fin para

retornar 1

Fin funcion

Funcion comprobarcolumna(int arr[9][9], int columna)

Declarar i,j;

Declarar copycolumnas[9];

Para i=0 hasta 9

copycolumnas[i]=arr[i][columna]

Fin para

para i=0 hasta9

para(j=i+1; j<9; j++)

si(copycolumnas[j]==arr[i][columna]&&arr[i][columna]!=0)

return 0

fin si

fin para

fin para

retornar 1

Fin funcion

funcion comprobarcuadro(int arr[9][9])

Declarar i,j;

Para i=0 hasta 9 de 3 en 3

paraj=0 hast 9 de 3 en 3

si(matriz\_3x3(arr,i,i+3,j,j+3)equivale a 0)

retornar 0

fin si

Fin para

Fin para

Retornar 1

Fin funcion

int matriz\_3x3(int matriz[9][9],int limix,int limsx,int limiy,int limsy)

{

Declarar i,j, k,arrcopy[9];

Inicializar k a 0

para i=limix hasta limsx

para j=limiy hasta limsy

arrcopy[k]=matriz[i][j]

k incrementa en 1

Fin para

Fin para

Para i=0hasta 9

Para j=i+1hasta 9

si(arrcopy[i]==arrcopy[j]&&arrcopy[j] es diferente de 0)

retornar 0

fin si

fin para

fin para

retornar 1

fin funcion

# Código Fuente

//Codigo en bitBucket: bitbucket.org/eduardoveras94/proyectos-isc-105

//Este codigo fue desarrollado por Eduardo Veras y Ernesto Rodriguez

//Matriculas 2012-0844 y 2012-0201

//inclusion de librerias para el manejo de colores y del tiempo

#include <stdlib.h>

#include <stdio.h>

#include <conio.c>

#include <time.h>

#include <windows.h>

#include <string.h>

//Declaracion de funciones

int comprobarfila(int arr[9][9], int fila);

int comprobarcolumna(int arr[9][9], int columna);

int comprobarcuadro(int arr[9][9]);

int sudoku(int arr[9][9]);

int bactracking(int matriz[9][9],int posicion);

int matriz\_3x3(int matriz[9][9],int limix,int limsx,int limiy,int limsy);

void mostrar(int matriz[9][9]);

char opcion=1;

//Estructura para los datos del usuario

struct datos

{

char nombre[30];

char apellido[30];

int num;

int destreza;

} ;

typedef struct datos Datos;

int main()

{

system("color 4e");

Datos datnombre;

int i,j,aceptado=1;

char filas[9],contador=0;

int matriz[9][9];

//Pantalla de bienvenida

printf("\n\n");

printf(" .\_\_\_ \_\_ .\_.\n");

printf(" \_\_\_\_\_\_ \_\_ \_\_ \_\_| \_/ \_\_\_\_ | | \_\_ \_\_ \_\_ | |\n");

printf(" / \_\_\_/ | | \\ / \_\_ | / \_ \\ | |/ / | | \\ | |\n");

printf(" \\\_\_\_ \\ | | / / /\_/ | ( <\_> ) | < | | / \\|\n");

printf(" /\_\_\_\_ > |\_\_\_\_/ \\\_\_\_\_ | \\\_\_\_\_/ |\_\_|\_ \\ |\_\_\_\_/ \_\_\n");

printf(" \\/ \\/ \\/ \\/\n");

printf("\n\n\n\n\t\t BUSCADOR SOLUCION FACTIBLE-SUDOKU");

printf("\n\t\t Presione cualquier letra");

getch();

system("CLS");

//imprime instrucciones para el sudoku

printf("\n\n\n\t\tINSTRUCCIONES DEL SUDOKU\n\n");

printf("\t1.Tienes que completar todas las casillas existentes, teniendo en \n\tcuenta que no pueden coincidir 2 numeros iguales en la misma fila, \n\tcolumna o cuadro.\n\n");

printf("\t2.Tan solo puedes rellenar numeros del 1 al 9 (inclusive).\n\n");

printf("\t3.El sudoku esta terminado cuando el cuadro esta completo sin \n\tningun valor vacio");

getch();

system("CLS");

printf("\n\n\tDigite su nombre y apellido: ");

scanf(" %s %s",&datnombre.nombre,&datnombre.apellido);

system("CLS");

printf("\n\n\tEspecifique su nivel de destreza en SUDOKU \n\t\t0: Principiante \n\t\t1: Intermedio \n\t\t2: Experto\n\t\t");

datnombre.destreza=getch()-'0';

while(datnombre.destreza!=0&&datnombre.destreza!=1&&datnombre.destreza!=2)

{

printf("\nDigite una opcion valida:");

datnombre.destreza=getch()-'0';

}

system("CLS");

//Introduccion de los datos del sudoku

//para proceder a su verificacion

while(aceptado==1)

{

printf("\nDigite los valores de la matriz inicial del SUDOKU:");

printf("\n(Use el cero para indicar una casilla libre):\n\n");

for(i=0; i<9; i++)

{

printf("\tFila%d: ",i+1);

scanf("%s",&filas);

for(j=0; j<9; j++)

{

filas[j]-='0';

matriz[i][j]=filas[j];

}

}

system("CLS");

printf("\n\n\tVerificando filas");

for(i=0; i<3; i++)

{

printf("\t.");

Sleep(500);

}

system("CLS");

printf("\n\n\n\n\tVerificando columnas");

for(i=0; i<3; i++)

{

printf("\t.");

Sleep(500);

}

system("CLS");

printf("\n\n\n\n\n\n\tVerificando cuadros");

for(i=0; i<3; i++)

{

printf("\t.");

Sleep(500);

}

//Esta parte es solo una ilusion para hacer creer

//que se estan verificando los datos los cuales ya

//han sido verificados

int cont =0;

aceptado=-1;

if(sudoku(matriz)==1)

aceptado=1;

if(aceptado==-1)

{

printf("\nLos valores no son validos\n");

printf("Desea intentar otra vez,%s ? (s/n):",datnombre.nombre);

aceptado=1;

opcion=getch();

if(opcion=='s'||opcion=='S')

{

aceptado=1;

continue;

}

else

aceptado=-1;

}

if(aceptado==1)

{

system("CLS");

contador++;

printf("\nEl sudoku parece ser valido, procediendo a su solucion...\n");

aceptado=1;

printf("Desea mostrar por animaci%cn el algoritmo (S/N)? ",162);

opcion=getch();

system("CLS");

printf("\n\nMuy bien %s %s , mediante el algoritmo de \nVuelta Atr%cs, se ha encontrado",datnombre.nombre,datnombre.apellido,160);

printf("la siguiente \nsoluci%cn factible al SUDOKU digitado:\n\n",162);

bactracking(matriz,0);

mostrar(matriz);

printf("\nUsted ha realizado %d busqueda de solucion factible.",contador);

printf("\nDesea realizar otra busqueda? (S/N): ");

opcion=getch();

if(opcion=='S'||opcion=='s')

aceptado=1;

else aceptado =-1;

}

if(aceptado==-1)

{

//Aqui se usa la libreria time.h para poder imprimir la hora actual del sistema

//la hora se imprime en formato de 24 horas

system("CLS");

char hora[6];

time\_t t;

struct tm \*tiempo;

t=time(0);

tiempo=localtime(&t);

strftime(hora, 6, "%H:%M", tiempo);

printf("\n\n\t\tSon las%s", hora);

printf("!que tenga feliz resto del dia %s %s!.\n\t\t Gracias por utilizar esta herramienta.",datnombre.nombre,datnombre.apellido);

getch();

return 0;

}

}

return 0;

}

//funcion para imprimir la matriz despues que esta arreglada

void mostrar (int matriz[9][9])

{

int i,j;

printf("\t%c%c%c%c%c%c%c%c%c%c%c%c%c\n",201,205,205,205,203,205,205,205,203,205,205,205,187);

for (i=0; i<9; i++)

{

printf("\t");

printf("%c",186);

for (j=0; j<9; j++)

{

if((j+1)%3)

{

textbackground(RED);

textcolor(WHITE);

printf("%d",matriz[i][j]);

textcolor(YELLOW);

}

else

{

textbackground(RED);

textcolor(WHITE);

printf("%d", matriz[i][j]);

textcolor(YELLOW);

printf("%c",186);

}

//Este if define si se imprime con animacion o sin animacion

//if(opcion=='S'||opcion=='s')

// Sleep(100);

// else

// continue;

}

putchar('\n');

if (!((i+1)%3))

printf("\t%c%c%c%c%c%c%c%c%c%c%c%c%c\n",204,205,205,205,206,205,205,205,206,205,205,205,185);

}

puts("\n");

}

//Esta funcion hace la fuerza bruta del backtracking

int bactracking(int matriz[9][9],int posicion)

{

//axioma

int k;

if(posicion==82)

{

return 1;

}

int i=posicion/9,j=posicion%9;

if(matriz[i][j]!=0)

{

if(opcion=='S'||opcion=='s')

{

printf("\n\n\n");

mostrar(matriz);

Sleep(86);

system("CLS");

}

return bactracking(matriz,posicion+1);

}

for(k=1; k<=9; k++)

{

matriz[i][j]=k;

//Llamada recursiva, tomando los caminos correctos

if(sudoku(matriz)==1)

{

if(bactracking(matriz,posicion+1)==1)

{

return 1;

}

}

matriz[i][j]=0;

}

return 0;

//si no encuentra un resultado, retorna 0

}

//Esta es la funcion principal de verificacion

//atravez de esta se llaman a las demas funciones

//de verificar columna, fila y cuadrado

int sudoku(int matriz[9][9])

{

int i=0;

while(comprobarcolumna(matriz,i)==1&&comprobarfila(matriz,i)==1

&&comprobarcuadro(matriz)==1&&i<9)

{

i++;

}

if(i==9)

return 1;

else

return 0;

//En el caso de que el tablero no sea valido retorna 0

}

//Comprabando las filas linealmente

int comprobarfila(int arr[9][9], int fila)

{

int i,j;

int copyfilas[9];

for(i=0; i<9; i++)

{

copyfilas[i]=arr[fila][i];

}

for (i=0; i<9; i++)

{

for(j=i+1; j<9; j++)

{

if(copyfilas[j]==arr[fila][i]&&arr[fila][i]!=0)

return 0;

}

}

return 1;

}

//Comprabando las columnas linealmente

int comprobarcolumna(int arr[9][9], int columna)

{

int i,j;

int copycolumnas[9];

for(i=0; i<9; i++)

{

copycolumnas[i]=arr[i][columna];

}

for (i=0; i<9; i++)

{

for(j=i+1; j<9; j++)

{

if(copycolumnas[j]==arr[i][columna]&&arr[i][columna]!=0)

return 0;

}

}

return 1;

}

//Para comprabar el cuadrado, se recorre la matriz

//cuadro por cuadro y se transforma cada cuadro en

//una cadena de enteros,luego se verifica que no

//haya ningun numero repetido en la cadena

//Para convertir de cuadro a cadena se usa otra funcion

int comprobarcuadro(int arr[9][9])

{

int i,j;

for(i=0; i<9; i+=3)

for(j=0; j<9; j+=3)

{

if(matriz\_3x3(arr,i,i+3,j,j+3)==0)

return 0;

}

return 1;

}

//Esta es la funcion que convierte el cuadro a una cadena de caracteres para verificar su validez

int matriz\_3x3(int matriz[9][9],int limix,int limsx,int limiy,int limsy)

{

int i,j,k=0,arrcopy[9];

for( i=limix ; i<limsx; i++)

{

for(j=limiy; j<limsy; j++)

{

arrcopy[k]=matriz[i][j];

k+=1;

}

}

for(i=0; i<9; i++)

{

for(j=i+1; j<9; j++)

{

if(arrcopy[i]==arrcopy[j]&&arrcopy[j]!=0)

return 0;

}

}

return 1;

}

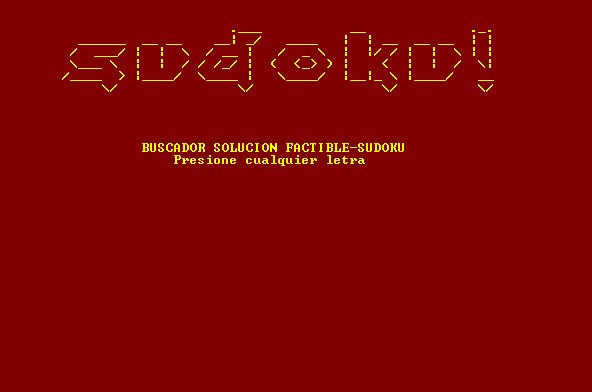
# Condiciones de Error

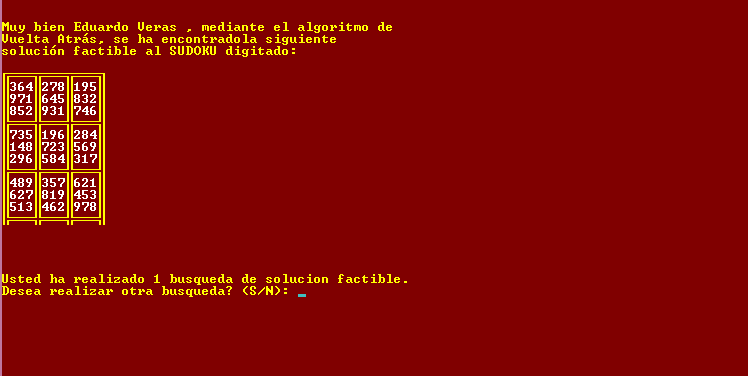
Nuestro programa funciona siempre y cuando se le inserten los datos indicados, han ocurrido algunos pocos casos en los que no logra resolver algunos sudokus como por ejemplo este:

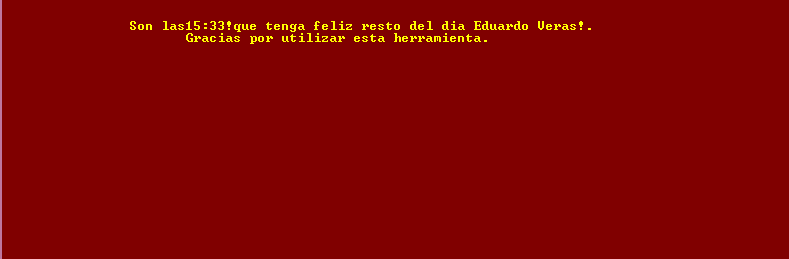
900100005  
005090201  
800040000  
000080000  
000700000  
000026009  
200300006  
000200900  
001904570

En todos los demas casos las soluciones han sido correctas

# Ejemplos de Uso







# Notas al Profesor

<http://www.lchs.info/Puss-In-Boots-Shrek-497126.jpeg>