FUNDAMENTOS DE ANÁLISIS Y DISEÑO DE ALGORITMOS

PRIMER MINIPROYECTO: DOCUMENTACION

JHOJAN RICARDO VELASCO - 1969042
KEREN BENAVIDES MUÑOZ - 1969076
CESAR AUGUSTO TELLO - 1969023

UNIVERSIDAD DEL VALLE
YUMBO
MAYO 2022

ANÁLISIS DE COSTOS

- Método para comprobar los valores

Instrucción	***Costo***	***Veces que se repite***
public boolean comprobar_valor(String valor1) {	C1	1
if (String.valueOf(valor1).equalsIgnoreCase("")) {		
return false;		
} else {	C2	2
int valor = Integer.valueOf(valor1);		
if (valor >= 0 && valor < 10) {		
return true;		
} else {	C3	1
return false;		
}		
}		
}		

- Método para comprobar filas

Instrucción	***Costo***	***Veces que se repite***
public boolean existe_fila(int numero, int fila) {		
boolean resultado = false;	C1	1
int a = matriz[0][2];	C2	1
for (int i = 0; i < matriz.length; i++) {	C3	n
if (numero == 0) {	C4	n-1
} else {		
if (matriz[fila][i] == numero) {	C5	n-1
resultado = true;		
break;		
}		
}		
}		
return resultado;	C6	1
}		

n-1+n-1+n+1+1+1= 3n+1

O(n)

- Método para comprobar columnas

Instrucción	***Costo***	***Veces que se repite***
public boolean existe_columna(int numero, int columna) {		
boolean resultado = false;	C1	1
int a = matriz[7][0];	C2	1
for (int i = 0; i < matriz.length; i++) {	C3	n
if (matriz[i][columna] == numero) {	C4	n-1
resultado = true;		
break;		
}		
}		
return resultado;	C5	1
}		

n-1+n+1+1+1 = 2n + 2

O(n)

- Metodos de determinacion

Instrucción	*Costo*	*Veces que se repite*
public boolean existe_caja(int valor, int fila, int columna) {		
int minimo_fila;	C1	1
int maximo_fila;	C2	1
int minimo_columna;	C3	1
int maximo_columna;	C4	1
boolean resultado = false;	C5	1
//DETERMINAMOS LAS FILAS		
if (fila >= 0 && fila < 3) {	C6	1
minimo_fila = 0;		
maximo_fila = 2;		
} else if (fila >= 3 && fila < 6) {	C7	1
minimo_fila = 3;		
maximo_fila = 5;		
} else {	C8	1
minimo_fila = 6;		
maximo_fila = 8;		
}		
//DETERMINAMOS LAS COLUMNAS		
if (columna >= 0 && columna < 3) {	C9	1
minimo_columna = 0;		
maximo_columna = 2;		
} else if (columna >= 3 && columna < 6) {	C10	1
minimo_columna = 3;		
maximo_columna = 5;		
} else {		
minimo_columna = 6;		
maximo_columna = 8;		
}		
//RECORREMOS EL RANGO, Y BUSCAMOS EL VALOR.		
for (int f = minimo_fila; f <= maximo_fila; f++) {	C11	n+1
for (int c = minimo_columna; c <= maximo_columna; c++) {	C12	n(n+1)
if (valor == 0) {	C13	n(n)
} else {		
if (matriz[f][c] == valor) {	C14	n(n)

resultado = true;		
break;		
}		
}		
}		
}		
//REGRESAMOS EL VALOR BOOLEANO.		
return resultado;	C15	1
}		

$$n^2 + n + n^2 + n^2 + n + 12 = 3n^2 + 2n + 12$$

O (n^2)

- Método guardar jugada

Instrucción	***Costo***	***Veces que se repite***
private int NuevoDato,ViejoDato;	C1	1
private int PosFila,PosColumna,PostArray;	C2	1
public void NuevaJugada(int fila, int columna, int ViejoDato, int NuevoDato)		
{		
if (PosJugada < this.Jugadas.size() - 1) {	C3	1
int Size = this.Jugadas.size() - 1;	C4	1
for (int i = Size; PosJugada < i; i--) {	C5	n
this.Jugadas.remove(i);	C6	n-1
}		
Jugada JugadaNueva = new Jugada(NuevoDato, ViejoDato, fila, columna, this.Jugadas.size());	C7	1
this.Jugadas.add(JugadaNueva);	C8	1
} else {		
Jugada JugadaNueva = new Jugada(NuevoDato, ViejoDato, fila, columna, this.Jugadas.size());	C9	1
this.Jugadas.add(JugadaNueva);	C10	1
}		
}		

n-1+n+1+1+1+1+1+1+1+1 = 2n+7

O(n)

- Metodo de ayuda

Instrucción	***Costo***	***Veces que se repite***
for (int i = 0; i < f.MatrizZudo.length;j++) {	C1	n
for (int j = 0; j < f.MatrizZudo.length;j++) {	C2	n(n-1)
if(f.MatrizZudo[i][j]==0){	C3	(n-1)(n-1)
JOptionPane.showMessageDialog(null, "en la fila "+i+ "columna "+j+"pon el numero "+f.MatrizSolucion[i][j]);		
break;		
}		
if(j==9){	C4	(n-1)(n-1)
i++;		
j=0;		
}		
}		
break;	C5	1
}		

$$n + n(n-1) + (n-1)(n-1) + (n-1)(n-1) + 1$$
$$= n + n^2 - n + n^2 - n - n + 1 + n^2 - n - n + 1 + 1 = 3n^2 - 4n + 3$$

O(n^2)

- Metodo de historial

Instrucción	***Costo***	***Veces que se repite***
public String mostrarjugadas(){		
String resultado="";	C1	1
for (int i = 0; i < HistorialJugada.size(); i++) {	C2	n
int a =i+1;	C3	n-1
resultado+="jugada numero :" +a+"puso el numero :" +HistorialJugada.get(i)+"\n";	C4	n-1
}		
return resultado;	C5	1
}		

n-1+n-1+n+1+1 = 3n

O(n)