

ALGORITMOS BT.

GENERAL: Realizamos una búsqueda exhaustiva, generando todas las posibles soluciones pudiendo buscar 1, mejor, o varias soluciones

Complejidad: $\Theta(e^n)$

Arboles de rec. profundidad

Pseudo código:

Función BT (datos)

if (datos Es Solucion)

La devolvemos o la almacenamos una copia

else if (esValida Siguiente op)

BT (datos (con opcion))

Quitamos el cambio

ALGORITMOS BT

LABERINTO: Buscamos solución a un laberinto dado

Pseudocódigo:

Función Lab (tablero, fila, columna, valor)

Ponemos la posición dada el valor

if (terminado(tablero))

Hacemos con la solución lo que nos pidan

else

if (valida \rightarrow Izq) Lab (fila - 1)

if (valida \rightarrow der) Lab (fila + 1)

if (valida \rightarrow aba) Lab (columna - 1)

if (valida \rightarrow arr) Lab (columna + 1)

si no, dejaremos el cambio poniendo a 0 la posición

Función Valida (tablero, fila, columna)

Comprobamos que la posición no sea un muro y este dentro del tablero

Función Terminado (tablero, fila, columna)

Si tanto filas como columnas están \rightarrow

SUDOKU: Resolver un sudoku dado

Psudocódigo:

Función solucionar (sudoku, listaSol)

if (lleno (sudoku))

Lo copiamos y añadimos la copia a la lista

else

ponemos fila y columna = -1 , y con un bucle vacío recorremos el sudoku

if (sudoku[] = 0)

Guardamos fila y columna y ponemos vacío a true , salimos del primer for **Break**

if (vacío) Break

if (!vacío) Devolvemos true

for (numeros)

if (puedo (sudoku, fila, columna))

Neto el numero

if (solucionar (sudoku, sol))

Ponemos un 0 deshaciendo el cambio

Devolvemos false

Función Puedo (sudoku, fila, columna)

Miramos si no esta rep en fila/columna y para cuadrado llamamos $inicol = col - col \% 3$
y es mismo a fila.

BT GRAFOS

HAMILTONIANO: Visitar una única vez todos los nodos

Pseudocódigo:

grafo = int [][]

camino = int []

Complejidad:

$\Theta(n!)$

Función Hamilton (grafo, camino, pos)

if (pos = grafo.leng)

if (grafo[cam[pos-1]][0] = 1)

Hemos visitado el nodo inicial devolvemos true

Declararímos booleano resultado a false

for (grafo.leng)

if (Valido (pos, i, camino, grafo))

Añadimos al camino el vértice

Resultado = Hamilton (grafo, cam, pos+1)

}

return resultado

Función Válido { pos, vértice, camino, grafo }

if (grafo [cam [pos-1]] [v] = 0) Devuelve false

for (posiciones)

if (camino [i] = v) Return false

Return true

COLOREADO: Cuantos colores son necesarios para colorear un grafo

Complejidad: $\Theta(\text{colores}^{\text{vertices}})$

Pseudo código:

grafo = int[][] color = int[]
 $V = \text{num_vertices}$

Funcion Colorear (grafo, color, vert, numcolores)

$\text{if } (\text{vert} = V)$ Devolvemos true

for (numcolores)

$\text{if } (\text{Buena}(\text{vert}, \text{grafo}, i, \text{color}))$

Añadimos a color[i] el i

$\text{if } (\text{Colorear}(\text{grafo}, \text{color}, \text{vert} + 1, \text{numcol}))$ Devuelve true

Paramos color[i] = 0

Devolvemos falso

Funcion Buena (vertice, grafo, color, listacolor)

for (V)

$\text{if } (\text{grafo}[v][i] = 1 \wedge \text{color} = \text{listacolor}[i])$

Devuelve falso

Devuelve true