TECNM - INSTITUTO TECNOLÓGICO DE CULIACÁN

ASIGNATURA:
INTELIGENCIA ARTIFICIAL

CARRERA: INGENIERÍA EN SISTEMAS COMPUTACIONALES

TRABAJO:
TABLAS COMPARATIVAS | EFICIENCIA DE MÉTODOS
DE RESOLUCIÓN 8-PUZZLE

NOMBRE DEL MAESTRO:
JOSE MARIO RIOS FELIX

NOMBRE DEL ESTUDIANTE(S):
AMILCAR RODRIGUEZ MORENO

CULIACÁN SINALOA, 26 DE FEBRERO DEL 2025.

INTRODUCCIÓN

Se muestran tablas comparativas de las diferentes implementaciones de algoritmos de resolución del juego “8-puzzle”. Se comparan las eficiencias con relación al tiempo de ejecución y nodos de memoria utilizados. Los algoritmos de resolución a comparar pueden (o no) implementar heurística, los algoritmos con heurística implementada se diferencia a través de la palabra “priority” en su nombre del método. Los algoritmos por comparar son los siguientes:

- **widthSolve**
  Método de resolución simple por anchura, analiza todo el árbol de escenarios posibles priorizándolos por su nivel de profundidad (la cantidad de movimientos hechos hasta llegar a el estado). Antes de insertar cada escenario se valida que no se halla insertado uno equivalente antes en el árbol. Siempre obtiene una solución óptima de ser posible.
- **depthSolve**
  Método de resolución simple por profundidad, analiza los nodos hijos de cada nodo de manera iterativa, validando que cada nodo no se haya validado antes en su línea de antecesores para evitar análisis cíclicos, pero sin guardar un historial completo de nodos visitados para obtener un ahorro de memoria. Las soluciones encontradas suelen ser poco óptimas y su uso de memoria es igual al de las profundidades alcanzadas.
- **limitDepthSolve**
  Una implementación restringida del algoritmo depthSolve, requiere de un parámetro de entrada que indica una profundidad máxima alcanzable en este método de búsqueda, una vez alcanzado un nodo con esta profundidad no se generan nodos hijos para seguir analizando. De forma predeterminada el valor de profundidad máxima asignado es de 31, ya que, según estudios, está comprobado que ningún estado de 8-puzzle tiene una resolución óptima mayor a 31 movimientos. Las soluciones encontradas no necesariamente son las óptimas.
- **iterativeDepthSolve**
  Una implementación iterativa del algoritmo limitDepthSolve, combina la lógica de los algoritmos widthSolve y depthSolve aplicando iterativamente limitDepthSolve con un valor de entrada desde 0 hasta 31, con el propósito de explorar todo el árbol de nodos nivel por nivel. Al igual que el algoritmo widthSolve, sus soluciones son óptimas, con la ventaja de requiere de un uso de memoria similar al del depthSolve, pero con la desventaja de que requiere de un mayor numero de operaciones que el widthSolve.
- **priorityWidthSolve**
  Una implementación con heurística del algoritmo widthSolve, analiza los nodos priorizando su nivel de profundidad y después su valor de prioridad heurística.
- **priorityDepthSolve**
  Una implementación con heurística del algoritmo depthSolve, analiza los sucesores de cada nodo en el orden de su valor de prioridad heurística.

- **priorityLimitDepthSolve**
  La implementación con profundidad restringida (de manera predeterminada a 31) del algoritmo priorityDepthSolve, la lógica es idéntica a la del algoritmo limitDepthSolve aplicando heurística.
- **priorityIterativeDepthSolve**
  La implementación iterativa del algoritmo priorityLimitDepthSolve, la lógica es idéntica a la de iterativeDepthSolve aplicando heurística. Las soluciones encontradas son las óptimas.
- **priorityQueueSolve**
  Método de resolución que establece un valor de prioridad heurística de cada nodo, el cual también toma en cuenta la profundidad de los nodos, pero no prioriza su orden de análisis en base a ésta profundidad como en el priorityWidthSolve, sino basado enteramente en su nivel de prioridad. Las soluciones encontradas no siempre son las óptimas.

Se compara la eficiencia de los algoritmos a través de los siguientes escenarios:

- **Escenario 1**

{4, 1, 3},
{7, 2, 5},
{0, 8, 6}

Un escenario simple de 3x3 cuya solución óptima se obtiene en 6 movimientos.

- **Escenario 2**

{4, 3, 1},
{5, 6, 8},
{7, 0, 2}

Solución óptima en 15 movimientos.

- **Escenario 3**

{3, 1, 0},
{4, 6, 8},
{5, 7, 2}

Solución óptima en 20 movimientos.

- **Escenario 4**

{2, 8, 7},
{3, 6, 4},
{5, 0, 1}

Solución óptima en 25 movimientos.

- **Escenario 5**

{8, 6, 7},
{2, 5, 4},
{3, 0, 1}

Solución óptima en 31 movimientos.

- **Escenario 6**

{3, 2},
{1, 0}

Escenario de 2x2 sin solución.

- **Escenario 7**

{1, 2, 3},
{4, 5, 6},
{8, 7, 0}

Escenario de 3x3 sin solución.

- **Escenario 8**

{0, 2, 3, 4},
{1, 5, 7, 8},
{9, 6, 10, 12},
{13,14,11,15}

Un escenario de 4x4 cuya solución óptima se encuentra a 6 movimientos.

TABLAS DE RESULTADOS

Escenario 1 (6 movimientos)

Algoritmo	*Tiempo (Milisegundos)*	*Memoria (Nodos guardados)*	*Coste de solución (movimientos)*
widthSolve	9	57	6
depthSolve	7	55	54
limitDepthSolve	65	32	30
iterativeDepthSolve	9	7	6
priorityWidthSolve	11	53	6
priorityDepthSolve	6	7	6
priorityLimitDepthSolve	6	7	6
priorityIterativeDepthSolve	9	7	6
priorityQueueSolve	6	13	6

Escenario 2 (15 movimientos)

Algoritmo	*Tiempo (Milisegundos)*	*Memoria (Nodos guardados)*	*Coste de solución (movimientos)*
widthSolve	1470	6664	15
depthSolve	4191	12362	12361
limitDepthSolve	360	32	31
iterativeDepthSolve	117	16	15
priorityWidthSolve	971	5175	15
priorityDepthSolve	6	36	35
priorityLimitDepthSolve	262	32	29
priorityIterativeDepthSolve	116	16	15
priorityQueueSolve	14	220	17

Escenario 3 (20 movimientos)

Algoritmo	*Tiempo (Milisegundos)*	*Memoria (Nodos guardados)*	*Coste de solución (movimientos)*
widthSolve	73495	41098	20
depthSolve	60793	34213	34212
limitDepthSolve	498	32	30
iterativeDepthSolve	678	21	20
priorityWidthSolve	61332	37811	20
priorityDepthSolve	8	41	40
priorityLimitDepthSolve	144	32	30
priorityIterativeDepthSolve	551	21	20
priorityQueueSolve	62	740	22

Escenario 4 (25 movimientos)

Algoritmo	*Tiempo (Milisegundos)*	*Memoria (Nodos guardados)*	*Coste de solución (movimientos)*
widthSolve	2179628 (36m 19.628s)	159199	25
depthSolve	245813 (4m 5.628s)	60126	60125
limitDepthSolve	1794	32	31
iterativeDepthSolve	11091	26	25
priorityWidthSolve	1699855 (28m 19.855s)	142089	25
priorityDepthSolve	18	278	277
priorityLimitDepthSolve	44772	32	31
priorityIterativeDepthSolve	14178	26	25
priorityQueueSolve	67	953	25

Escenario 5 (31 movimientos)

Algoritmo	*Tiempo (Milisegundos)*	*Memoria (Nodos guardados)*	*Coste de solución (movimientos)*
widthSolve	3747554 (1h 2m 27.554s)	181440	31
depthSolve	747223 (12m 27.223s)	87952	87951
limitDepthSolve	14454	32	31
iterativeDepthSolve	256917 (4m 16.917s)	32	31
priorityWidthSolve	3366207 (56m 6.207s)	181439	31
priorityDepthSolve	15	108	107
priorityLimitDepthSolve	26234	32	31
priorityIterativeDepthSolve	314375 (5m 14.375s)	32	31
priorityQueueSolve	9572	12561	31

Escenario 6 (sin solución)

Algoritmo	*Tiempo (Milisegundos)*	*Memoria (Nodos guardados)*	*Coste de solución (movimientos)*
widthSolve	13	12	NA
depthSolve	20	13	NA
limitDepthSolve	11	13	NA
iterativeDepthSolve	25	13	NA
priorityWidthSolve	14	12	NA
priorityDepthSolve	8	13	NA
priorityLimitDepthSolve	14	13	NA
priorityIterativeDepthSolve	25	13	NA
priorityQueueSolve	12	12	NA

Escenario 7 (sin solución)

Algoritmo	*Tiempo (Milisegundos)*	*Memoria (Nodos guardados)*	*Coste de solución (movimientos)*
widthSolve	3497577 (56m 17.577s)	181440	NA
depthSolve	0	0	NA
limitDepthSolve	141691 (2m 21.691s)	32	NA
iterativeDepthSolve	0	0	NA
priorityWidthSolve	3766621 (1h 2m 46.621s)	181440	NA
priorityDepthSolve	0	0	NA
priorityLimitDepthSolve	162435	32	NA
priorityIterativeDepthSolve	0	0	NA
priorityQueueSolve	3228533 (53m 48.533s)	181441	NA

Escenario 8 (6 movimientos, matriz 4x4)

Algoritmo	*Tiempo (Milisegundos)*	*Memoria (Nodos guardados)*	*Coste de solución (movimientos)*
widthSolve	14	177	6
depthSolve	0	0	0
limitDepthSolve	1084	32	30
iterativeDepthSolve	8	7	6
priorityWidthSolve	10	97	6
priorityDepthSolve	6	7	6
priorityLimitDepthSolve	8	7	6
priorityIterativeDepthSolve	11	7	6
priorityQueueSolve	5	16	6

(Los resultados en 0 son resultados tan tardados que no pudieron ser registrados)