TECNOLÓGICO NACIONAL DE MÉXICO

TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE CULIACÁN

TABLA COMPARATIVA

Materia: Inteligencia Artificial
Docente: Jose Mario Rios Felix

Alumna:
Miranda Contreras Montserrat

Horario: 6:00 pm- 7:00 pm

Búsqueda en Anchura (BFS - Breadth-First Search)

- **Tiempo**: BFS explora todos los nodos nivel por nivel, asegurándose de explorar completamente un nivel antes de pasar al siguiente. La complejidad temporal de BFS es O(b^d), donde b es el factor de ramificación del árbol (número promedio de hijos por nodo) y d es la profundidad del árbol.
- **Espacio**: BFS necesita almacenar todos los nodos de un nivel para explorar los del siguiente nivel, lo que puede requerir un espacio significativo. La complejidad espacial es también O(b^d) porque en el peor caso, necesita almacenar todo el nivel más ancho del árbol en la memoria.

Búsqueda en Profundidad (DFS - Depth-First Search)

- **Tiempo**: DFS explora un camino hasta el final antes de retroceder y probar otra ruta. La complejidad temporal es O(b^m) en el peor caso, donde m es la profundidad máxima del árbol. Sin embargo, si se encuentra una solución cerca de la raíz, puede ser mucho más rápido.
- **Espacio**: DFS solo necesita almacenar el camino actual desde la raíz hasta el nodo actual, además de los nodos no explorados de cada nodo en el camino. Esto da como resultado una complejidad espacial de O(bm), donde m es la profundidad máxima del árbol.

Tabla Comparativa

Algoritmo	Complejidad en Tiempo	Complejidad en Espacio
Búsqueda en Anchura	O(b^d)	O(b^d)
Búsqueda en Profundidad	O(b^m)	O(bm)

- **b** = factor de ramificación (número promedio de hijos por nodo)
- **d** = profundidad de la solución
- **m** = profundidad máxima del árbol

Observaciones:

- BFS es óptimo, siempre encuentra la solución menos profunda primero, pero puede ser impracticable en espacio para árboles muy anchos o infinitos.
- DFS puede ser más eficiente en espacio y útil en árboles muy profundos o infinitos, pero no garantiza encontrar la solución óptima primero.

Estas complejidades asumen una representación implícita del árbol de búsqueda, donde los sucesores de un nodo se generan en tiempo de ejecución, como es el caso de muchos problemas de espacio de estado como el puzzle de 8.

En el código se muestra así primero:

```
  System.out.println("Resolviendo con BFS:");
       juego.busquedaEnAnchura();
```

Con **Búsqueda en Anchura (BFS - Breadth-First Search)**

```
123
056
478

123
456
078

123
456
708

123
456
780

Número de movimientos: 13
Tiempo total para resolver el puzzle en ms: 17
PS C:\Users\monts\Desktop\IA>
```

Es mas rápido y en menos movimientos, en cambio en

Búsqueda en Profundidad (DFS - Depth-First Search)

```
123
405
786

123
450
786

123
456
780

Número de movimientos: 31309
Tiempo total para resolver el puzzle en ms: 6866
```

Son mas movimientos y mas tiempo de ejecución.