Características	BFS (Breadth-First Search)
Definición	Es un algoritmo de búsqueda que explora los nodos de un grafo o árbol en **niveles**: primero los más cercanos al nodo inicial.
Estrategia	Utiliza una **cola** para almacenar los nodos que se deben explorar, procesando primero los más cercanos al nodo inicial.
Orden de Exploración	Explora los nodos por **niveles**, de cerca a lejano.
Estructura de Datos	Cola (FIFO).
Ventajas	- Garantiza encontrar el camino más corto (en grafos no ponderados).
	- Es adecuado para grafos poco profundos.
Desventajas	- Consume más memoria, especialmente para grafos grandes.
	- Menos eficiente en grafos profundos.
Complejidad Temporal	O(V + E), donde V es el número de vértices y E es el número de aristas.
Complejidad Espacial	O(V), ya que almacena todos los nodos a medida que se exploran.
Aplicaciones	- Encontrar el camino más corto en un grafo no ponderado.
	- Algoritmos de planificación de tareas.
	- Juegos de búsqueda como "puzzle" (ejemplo, 8-puzzle).
	- Redes sociales, recomendaciones.
	- Análisis de rutas en mapas y GPS.

Usos típicos	- Resolución de problemas de conectividad.

DFS (Depth-First Search)
Es un algoritmo de búsqueda que explora los nodos de un grafo o árbol **profundizando** lo más posible en cada rama antes de retroceder.
Utiliza una **pila** (o recursión) para explorar un nodo, luego explora sus vecinos antes de retroceder y continuar con otros nodos.
Explora los nodos **en profundidad**, primero explora un camino completo antes de volver.
Pila (LIFO) o recursión (automáticamente usa la pila del sistema).
- Es más sencillo de implementar.
- Requiere menos memoria que BFS para grafos profundos.
- Puede encontrar soluciones rápidamente si están cerca de la raíz.
- No garantiza encontrar la solución más corta.
- Puede quedar atrapado en ramas profundas si el grafo es infinito.
O(V + E), donde V es el número de vértices y E es el número de aristas.
O(V), pero generalmente se usa menos espacio en grafos muy profundos, ya que solo se almacenan los nodos actuales.
- Búsqueda en laberintos.
- Encontrar componentes conexos en grafos no dirigidos.
- Exploración de redes.
- Resolver problemas de laberintos o puzzles.
- Generación de árboles de decisión.

- Comprobación de conectividad.