Resumen de Estrategias de Búsqueda: DFS, BFS e IDS

Búsqueda en Profundidad (DFS)

A partir de la información obtenida en el documento *11.busquedas.pdf*, estos son los aspectos esenciales sobre la Búsqueda en Profundidad (DFS, por sus siglas en inglés *Depth First Search*):

- **Estrategia de búsqueda:** DFS es una técnica que explora los nodos del árbol de búsqueda profundizando lo máximo posible en cada rama antes de retroceder y examinar otras alternativas.
- **Implementación:** Se implementa generalmente mediante una pila (stack) LIFO, donde los sucesores de un nodo se agregan al inicio de la estructura. Esto asegura que se explore primero el nodo más recientemente insertado.
- **Tipo de búsqueda:** DFS corresponde a una búsqueda tipo árbol, ya que parte desde un estado inicial y genera un árbol de posibles caminos.
- **Estados repetidos:** La ausencia de control sobre los estados ya visitados puede generar exploraciones redundantes o incluso infinitas. Esto provoca un aumento exponencial en el trabajo requerido.
- **Comparación con BFS:** A diferencia de la Búsqueda en Anchura (BFS), DFS no garantiza encontrar el camino más corto. BFS utiliza una cola FIFO, mientras que DFS se basa en una pila.
- **Búsqueda en grafo vs. árbol:** En la búsqueda en grafo se evita expandir estados duplicados mediante una estructura auxiliar (como un conjunto o diccionario). Esto optimiza el proceso en comparación con la búsqueda en árbol.

- **Resumen:** DFS prioriza explorar profundamente cada camino antes de cambiar de dirección. Es eficiente en memoria en algunos casos, pero puede ser ineficiente si no se controlan los ciclos.

Complejidad:

- **Tiempo:** En el peor de los casos, explora todo el árbol hasta una profundidad máxima m con un factor de ramificación b. Su complejidad es $O(b^m)$.
- **Memoria:** Requiere memoria proporcional a la profundidad explorada, es decir, $O(b \cdot m)$ para búsqueda en árbol. En búsqueda en grafo, puede necesitar memoria proporcional al número total de estados.

Búsqueda en Anchura (BFS)

Basado en la misma fuente, estos son los puntos más relevantes sobre la Búsqueda en Anchura (BFS, *Breadth First Search*):

- **Estrategia de búsqueda:** BFS explora primero todos los nodos en la profundidad actual antes de avanzar al siguiente nivel, expandiendo el espacio de búsqueda por niveles.
- **Implementación:** Utiliza una cola (queue) FIFO, donde los nodos se agregan al final y se procesan por orden de llegada, garantizando así la expansión ordenada por niveles.
- **Tipo de búsqueda:** Al igual que DFS, se trata de una búsqueda en árbol que recorre de manera sistemática los caminos posibles desde el nodo raíz.
- **Estados repetidos:** Es esencial evitar la expansión de estados ya visitados, especialmente en espacios de búsqueda cíclicos. La implementación de búsqueda en grafo lo soluciona manteniendo un registro de estados explorados.
- **Camino más corto:** Una propiedad fundamental de BFS es que garantiza encontrar la solución con la menor cantidad de transiciones, siempre que el costo de cada acción sea uniforme.

- **Complejidad en tiempo:** Si el factor de ramificación es b y la profundidad de la solución más cercana es d, entonces la complejidad temporal es $O(b^d)$.
- **Complejidad en memoria:** Dado que se almacenan todos los nodos del nivel actual, la memoria requerida también es $O(b^d)$. Esto representa una desventaja frente a DFS, que generalmente consume menos memoria.

Búsqueda con Profundización Iterativa (IDS)

Aunque la fuente no menciona explícitamente IDS (*Iterative Deepening Search*), podemos entenderla como una combinación de DFS y BFS.

- **Concepto:** IDS realiza búsquedas en profundidad limitadas, comenzando con una profundidad de 0 y aumentando el límite en cada iteración hasta encontrar la meta.
- **Relación con DFS y BFS:** Internamente, IDS ejecuta DFS, pero al realizarlo por niveles, hereda la propiedad de BFS de encontrar la solución más superficial. Es óptimo en espacios donde las acciones tienen costo uniforme.
- **Ventajas:**
  - **Completitud:** Garantiza encontrar una solución si esta existe a profundidad finita.
  - **Óptimo:** Encuentra el camino más corto en términos del número de pasos cuando el costo de las acciones es igual.
  - **Memoria:** Tiene un consumo de memoria bajo, similar a DFS: $O(b \cdot d)$, siendo d la profundidad de la solución.
- **Desventajas:**
  - **Redundancia:** IDS repite la exploración de nodos en niveles bajos múltiples veces.
  - **Tiempo:** Aunque repite trabajo, su complejidad sigue siendo $O(b^d)$, como BFS, aunque con un coeficiente mayor.

- **Complejidad:**
  - **Tiempo:** $O(b^d)$, siendo b el factor de ramificación y d la profundidad de la solución.
  - **Memoria:** $O(b \cdot d)$, ya que solo necesita almacenar la ruta actual como DFS.