DFS y BFS

Jorge Baier

Departamento de Ciencia de la Computación
Pontificia Universidad Católica de Chile

Santiago, Chile

Objetivos

- Conocer los algoritmos de búsqueda DFS y BFS
- Comprender cómo estos algoritmos se derivan del algoritmo de búsqueda genérica

Búsqueda Genérica

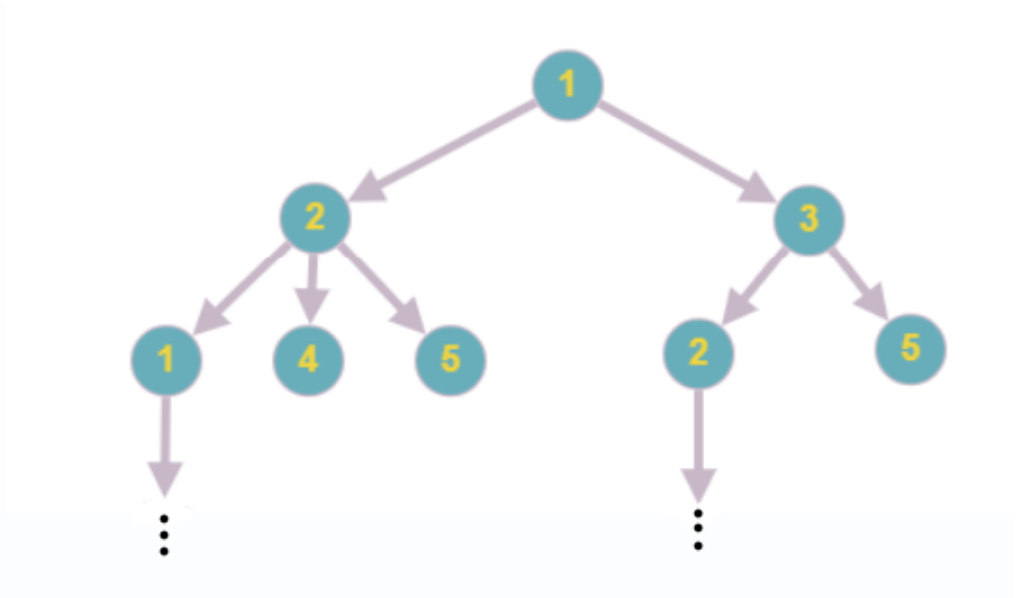
El siguiente es un algoritmo de búsqueda genérico.
Input: Un problema de búsqueda $(\mathbf{S}, A, s_{init}, G)$
Output: Un nodo objetivo

1. *Open* es un contenedor vacío
2. *Closed* es un conjunto vacío
3. Inserta s_{init} a *Open*
4. $parent(s_{init}) = null$
5. **while** $Open \neq \emptyset$
6. $\quad u \leftarrow \texttt{Extraer}(Open)$
7. $\quad$ Inserta u en *Closed*
8. $\quad$ **for each** $v \in Succ(u) \setminus (Open \cup Closed)$
9. $\quad\quad parent(v) = u$
10. $\quad\quad$ **if** $v \in G$ **return** v
11. $\quad\quad$ Inserta v a *Open*

Grafo del problema y Árbol de búsqueda

- Todos los problemas de búsqueda pueden ser representados por un **grafo**, donde cada nodo es un estado, los cuales tienen una arista hacia sus sucesores.
- Todos los algoritmos de búsqueda que veremos en el curso generan un **árbol de búsqueda**, el cual tiene como raíz el estado inicial y cada nodo tiene como hijos sus sucesores.

Grafo del problema



Árbol de búsqueda

Búsqueda en Profundidad (*Depth-First Search*)

- Usualmente abreviado como DFS.
- Resulta de implementar a *Open* como un stack.
- Siempre se extrae el elemento al tope de *Open* (línea 6; alg. principal).

Ejemplo: En pizarra.

Búsqueda en Amplitud (*Breadth-First Search*)

- Abreviado como BFS
- Resulta de implementar a *Open* como una cola.
- Siempre se extrae el primer elemento al principio de *Open* (línea 6; alg. principal).

Ejemplo: En pizarra.

Propiedades

Teorema

Si el espacio de estados es finito, búsqueda en profundidad con detección de ciclos es completo (es decir, encuentra una solución si ésta existe).

Propiedades

Teorema

Si el espacio de estados es finito, búsqueda en profundidad con detección de ciclos es completo (es decir, encuentra una solución si ésta existe).

Teorema

Si el espacio de búsqueda es finito, búsqueda en amplitud es completo y óptimo para problemas de búsqueda con costos uniformes.

Tiempo y Espacio

Para los siguientes resultados, suponemos:

- b: factor de ramificación promedio.
- p: profundidad a la que se encuentra la solución.
- m: largo de la rama más larga del árbol de búsqueda.

Teorema

La memoria usada por DFS es $\mathcal{O}(bm)$, mientras que breadth-first necesita memoria de tamaño $\mathcal{O}(b^p)$.

Teorema

DFS requiere tiempo $\mathcal{O}(b^m)$, mientras que breadth-first necesita tiempo $\mathcal{O}(b^p)$.

Lo mejor de los dos mundos

Profundidad Limitada

Funciona como DFS, pero recibe como parámetro un límite ℓ de profundidad para la búsqueda. Se ejecuta DFS sobre el subárbol de profundidad ℓ del espacio de búsqueda.

Lo mejor de los dos mundos

Profundidad Limitada

Funciona como DFS, pero recibe como parámetro un límite ℓ de profundidad para la búsqueda. Se ejecuta DFS sobre el subárbol de profundidad ℓ del espacio de búsqueda.

Profundización Iterativa (*Iterative Deepening DFS*)

1. ℓ=1;
2. realice búsqueda en profundidad limitada con límite ℓ.
3. si hubo éxito, retorne el estado encontrado; en otro caso incremente ℓ y vuelva al paso anterior.

Resultados sobre IDDFS

Teorema

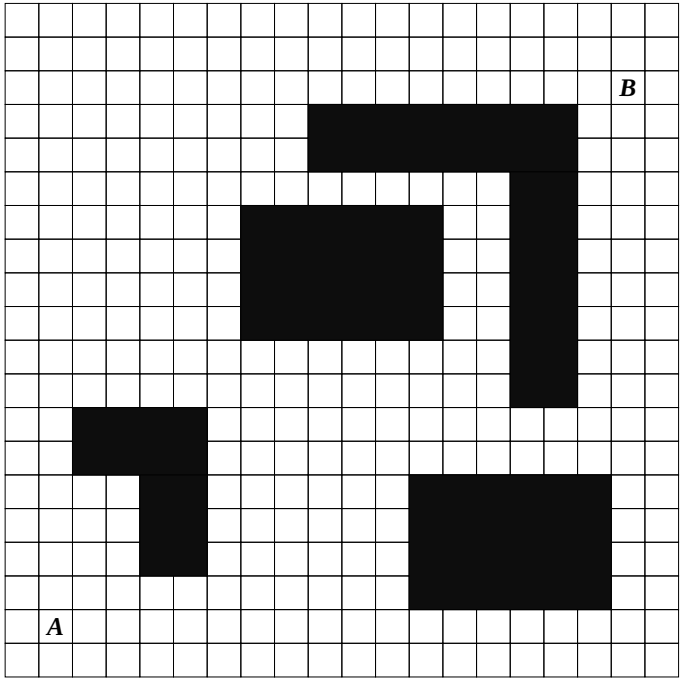
Profundización Iterativa es completo.

- b: factor de ramificación promedio.
- p: profundidad a la que se encuentra la solución.
- m: largo de la rama más larga del árbol de búsqueda.

Teorema

El tiempo requerido por IDDFS es $\mathcal{O}(b^p)$ y memoria de tamaño $\mathcal{O}(bp)$.

Ejemplo



Objetivos

- Conocer los algoritmos de búsqueda DFS y BFS
- Comprender cómo estos algoritmos se derivan del algoritmo de búsqueda genérica