



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Cuaderno de trabajo: Búsqueda en profundidad iterativa¹

Albert Sanchis

DSIC

Departamento de Sistemas
Informáticos y Computación

¹Para una correcta visualización, se requiere Acrobat Reader v. 7.0 o superior

Objetivos formativos

- ▶ Caracterizar la búsqueda convencional en un grafo de estados.
- ▶ Describir búsqueda en profundidad iterativa.
- ▶ Construir el árbol de búsqueda en profundidad iterativa.
- ▶ Aplicar búsqueda en profundidad iterativa a un problema clásico.
- ▶ Analizar la calidad de búsqueda en profundidad iterativa.

Problema: La ruta más corta entre dos puntos

Búsqueda de una ruta más corta desde Arad a Bucarest [1]:



$\text{Acciones}(\text{Arad}) = \{\text{Ir}(\text{Sibiu}), \text{Ir}(\text{Timisoara}), \text{Ir}(\text{Zerind})\}.$

Búsqueda en profundidad iterativa [2]

PI(G, s) // *Profundidad Iterativa*

para $m = 0, 1, 2, \dots$: **si** ($r = \text{DFS}(G, s, m)$) $\neq \text{NULL}$: **retorna** r

DFS(G, s', m) // *Depth-first search* con profundidad máxima m

$O = \text{IniPila}(s')$ // *Open: frontera-pila* de la búsqueda

mientras no $\text{PilaVacía}(O)$:

$s = \text{Desapila}(O)$ // selección *LIFO (Last in, first out)*

si $\text{Objetivo}(s)$ **retorna** s // solución encontrada!

si $\text{Profundidad}(s) < m$: // no a profundidad máxima

para toda $(s, n) \in \text{Adyacentes}(G, s)$: // generación: n hijo de s

$\text{Apila}(O, n)$ // añadimos n a la pila

retorna **NULL** // ninguna solución encontrada

- **Cuestión 1:** Haz una traza del algoritmo *profundidad iterativa* aplicado al problema de búsqueda de una ruta más corta desde Arad a Bucarest.

Con $m = 0$

O	S
{Arad (p=0)}	—
{}	Arad (p=0)

Con $m = 1$

O	S
{Arad (p=0)}	—
{Sibiu (p=1), Timisoara (p=1), Zerind (p=1)}	Arad (p=0)
{Timisoara (p=1), Zerind (p=1)}	Sibiu (p=1)
{Zerind (p=1)}	Timisoara (p=1)
{}	Zerind (p=1)

Con $m = 2$

O	s
{Arad (p=0)}	—
{Sibiu (p=1), Timisoara (p=1), Zerind (p=1)}	Arad (p=0)
{Arad (p=2), Fagaras (p=2), Oradea (p=2), Rimnicu(p=2), Timisoara (p=1), Zerind (p=1)}	Sibiu (p=1)
{Fagaras (p=2), Oradea (p=2), Rimnicu(p=2), Timisoara (p=1), Zerind (p=1)}	Arad (p=2)
{Oradea (p=2), Rimnicu(p=2), Timisoara (p=1), Zerind (p=1)}	Fagaras (p=2)
{Rimnicu(p=2), Timisoara (p=1), Zerind (p=1)}	Oradea (p=2)
{Timisoara (p=1), Zerind (p=1)}	Rimnicu (p=2)
{Arad (p=2), Lugoj (p=2), Zerind (p=1)}	Timisoara (p=1)
{Lugoj (p=2), Zerind (p=1)}	Arad (p=2)
{Zerind (p=1)}	Lugoj (p=2)
{Arad (p=2), Oradea (p=2)}	Zerind (p=1)
{Oradea (p=2)}	Arad (p=2)
{}	Oradea (p=2)

Con $m = 3$

O	s
{Arad (p=0)}	—
{Sibiu (p=1), Timisoara (p=1), Zerind (p=1)}	Arad (p=0)
{Arad (p=2), Fagaras (p=2), Oradea (p=2), Rimnicu(p=2), Timisoara (p=1), Zerind (p=1)}	Sibiu (p=1)
{Sibiu (p=3), Timisoara (p=3), Zerind (p=3), Fagaras (p=2), Oradea (p=2), Rimnicu(p=2), Timisoara (p=1), Zerind (p=1)}	Arad (p=2)
{Timisoara (p=3), Zerind (p=3), Fagaras (p=2), Oradea (p=2), Rimnicu(p=2), Timisoara (p=1), Zerind (p=1)}	Sibiu (p=3)
{Zerind (p=3), Fagaras (p=2), Oradea (p=2), Rimnicu(p=2), Timisoara (p=1), Zerind (p=1)}	Timisoara (p=3)
{Fagaras (p=2), Oradea (p=2), Rimnicu(p=2), Timisoara (p=1), Zerind (p=1)}	Zerind (p=3)
{Bucharest (p=3), Sibiu (p=3), Oradea (p=2), Rimnicu(p=2), Timisoara (p=1), Zerind (p=1)}	Fagaras (p=2)
{Sibiu (p=3), Oradea (p=2), Rimnicu(p=2), Timisoara (p=1), Zerind (p=1)}	Bucharest (p=3)

- **Cuestión 2:** Construye el árbol de búsqueda resultante de aplicar el algoritmo *profundidad iterativa* al problema de búsqueda de una ruta más corta desde Arad a Bucarest.

Amb $m = 0$

Amb $m = 1$

Amb $m = 2$

Amb $m = 3$

- ▶ **Cuestión 3:** ¿El algoritmo encuentra solución? **Sí**
- ▶ **Cuestión 4:** Si la respuesta es “Sí”:
 - ▷ ¿Cuántas iteraciones utilizando el algoritmo DFS se han necesitado hasta encontrar la solución? **Cuatro**
 - ▷ ¿De qué depende el número de iteraciones que necesita el algoritmo para encontrar la solución? **De la profundidad a la cual se encuentra la solución más corta en número de acciones**
 - ▷ ¿Cuál ha sido la solución encontrada? **El camino solución encontrado ha sido: Arad, Sibiu, Fagaras, Bucharest**
 - ▷ ¿Cuál es el coste de esta solución? **450**
 - ▷ ¿Se trata de la solución óptima? **No, porque hay otra solución con menor coste de 418: Arad, Sibiu, Rimnicu, Pitesti, Bucharest**
 - ▷ ¿Qué tipo de solución encuentra el algoritmo profundidad iterativa? **Aplicando un recorrido en profundidad encuentra la solución que se encuentra a menos profundidad en el árbol de búsqueda**

Referencias

- [1] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, third edition, 2010.
- [2] R. E. Korf. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, 1985.