



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Cuaderno de trabajo:

Búsqueda en coste uniforme

Algoritmo de Dijkstra¹

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 coste uniforme (o algoritmo de Dijkstra).
- ▶ Construir el árbol de búsqueda en coste uniforme.
- ▶ Aplicar búsqueda en coste uniforme a un problema clásico.
- ▶ Analizar la calidad de búsqueda en coste uniforme.

Problema: La ruta más corta entre dos puntos

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



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

Coste uniforme o algoritmo de Dijkstra [1, 2, 3]

```
UCS( $G, s'$ )      // Uniform-cost search;  $G$  grafo ponderado,  $s'$  start
 $O = \text{IniCola}(s', g_{s'} \triangleq 0)$       // Open: cola de prioridad  $g$ 
 $C = \emptyset$       // Closed: nodos explorados
mientras no  $\text{ColaVacía}(O)$ :      // 1ro el mejor:  $s = \arg \min_{n \in O} g_n$ 
     $s = \text{Desencola}(O)$       // desempates a favor de objetivo
    si  $\text{Objetivo}(s)$  retorna  $s$       // solución encontrada!
     $C = C \cup \{s\}$       //  $s$  explorado
    para toda  $(s, n) \in \text{Adyacentes}(G, s)$ : // generación:  $n$  hijo de  $s$ 
         $x = g_s + w(s, n)$  // coste del camino de  $s'$  a  $n$  pasando por  $s$ 
        si  $n \notin C \cup O$ :  $\text{Encola}(O, n, g_n \triangleq x)$ 
        si no si  $n \in O$  y  $x < g_n$ :  $\text{Modcola}(O, n, g_n \triangleq x)$ 
retorna NULL      // ninguna solución encontrada
```

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

O	C	s
{Arad (c=0)}	{}	—
{Zerind (c=75), Timisoara (c=118), Sibiu (c=140)}	{Arad (c=0)}	Arad (c=0)
{Timisoara (c=118), Sibiu (c=140), Oradea (c=146)}	{Arad (c=0), Zerind (c=75)}	Zerind (c=75)
{Sibiu (c=140), Oradea (c=146), Lugoj (c=229)}	{Arad (c=0), Zerind (c=75), Timisoara (c=118)}	Timisoara (c=118)
{Oradea (c=146), Rimnicu (c=220), Lugoj (c=229), Fagaras (c=239)}	{Arad (c=0), Zerind (c=75), Timisoara (c=118), Sibiu (c=140)}	Sibiu (c=140)
{Rimnicu (c=220), Lugoj (c=229), Fagaras (c=239)}	{Arad (c=0), Zerind (c=75), Timisoara (c=118), Sibiu (c=140), Oradea (c=146)}	Oradea (c=146)
{Lugoj (c=229), Fagaras (c=239), Pitesti (c=317), Craiova (c=366)}	{Arad (c=0), Zerind (c=75), Timisoara (c=118), Sibiu (c=140), Oradea (c=146), Rimnicu (c=220)}	Rimnicu (c=220)
{Fagaras (c=239), Mehadia (c=299), Pitesti (c=317), Craiova (c=366)}	{Arad (c=0), Zerind (c=75), Timisoara (c=118), Sibiu (c=140), Oradea (c=146), Rimnicu (c=220), Lugoj (c=229)}	Lugoj (c=229)
{Mehadia (c=299), Pitesti (c=317), Craiova (c=366), Bucharest (c=450)}	{Arad (c=0), Zerind (c=75), Timisoara (c=118), Sibiu (c=140), Oradea (c=146), Rimnicu (c=220), Lugoj (c=229), Fagaras (c=239)}	Fagaras (c=239)

O	C	s
{Pitesti (c=317), Craiova (c=366), Dobreta (c=374), Bucharest (c=450)}	{Arad (c=0), Zerind (c=75), Timisoara (c=118), Sibiu (c=140), Oradea (c=146), Rimnicu (c=220), Lugoj (c=229), Fagaras (c=239), Mehadia (c=299)}	Mehadia (c=299)
{Craiova (c=366), Dobreta (c=374), Bucharest (c=418)}	{Arad (c=0), Zerind (c=75), Timisoara (c=118), Sibiu (c=140), Oradea (c=146), Rimnicu (c=220), Lugoj (c=229), Fagaras (c=239), Mehadia (c=299), Pitesti (c=317)}	Pitesti (c=317)
{Dobreta (c=374), Bucharest (c=418)}	{Arad (c=0), Zerind (c=75), Timisoara (c=118), Sibiu (c=140), Oradea (c=146), Rimnicu (c=220), Lugoj (c=229), Fagaras (c=239), Mehadia (c=299), Pitesti (c=317), Craiova (c=366)}	Craiova (c=366)
{Bucharest (c=418)}	{Arad (c=0), Zerind (c=75), Timisoara (c=118), Sibiu (c=140), Oradea (c=146), Rimnicu (c=220), Lugoj (c=229), Fagaras (c=239), Mehadia (c=299), Pitesti (c=317), Craiova (c=366), Dobreta (c=374)}	Dobreta (c=374)
{}	{Arad (c=0), Zerind (c=75), Timisoara (c=118), Sibiu (c=140), Oradea (c=146), Rimnicu (c=220), Lugoj (c=229), Fagaras (c=239), Mehadia (c=299), Pitesti (c=317), Craiova (c=366), Dobreta (c=374)}	Bucharest (c=418)

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

- ▶ **Cuestión 3:** ¿El algoritmo encuentra solución? **Sí**
- ▶ **Cuestión 4:** Si la respuesta es “Sí”:
 - ▷ ¿Cuál ha sido la solución encontrada? **El camino solución encontrado ha sido: Arad, Sibiu, Rimnicu, Pitesti, Bucharest**
 - ▷ ¿Cuál es el coste de esta solución? **418**
 - ▷ ¿Se trata de la solución óptima? **Sí**
 - ▷ ¿Qué tipo de solución encuentra el algoritmo BFS? **Soluciones óptimas si los costos de las acciones son positivos**

Referencias

- [1] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, third edition, 2010.
- [2] E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1959.
- [3] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer, 2018.