

# BIỂU DIỄN TRI THỨC

## Bài tập 3

Nhóm 07

June 19th, 2021

### **Bài 3**

#### **Thu thập tri thức**

##### **Lý thuyết đồ thị**

##### **Bài toán tìm đường đi ngắn nhất**

## Câu a. Xác định nguồn tri thức

- **Xác định bài toán**

Cho đồ thị vô hướng  $G$  không có trọng số âm, đỉnh bắt đầu  $x_0$ . Tìm đường đi ngắn nhất từ  $x_0$  đến đỉnh bất kỳ  $x_k$  theo thuật toán Dijkstra.

- **Nguồn thu thập tri thức**

- Kenneth H. Rosen. 2018. Discrete mathematics and its applications. Boston: McGraw-Hill

- **Phân loại tri thức**

- **Khái niệm:** Đỉnh, Cạnh

- **Quan hệ:** Cạnh là khoảng cách giữa hai đỉnh liền kề

- **Hàm:** Dijkstra(Đỉnh bắt đầu, Đỉnh kết thúc); Degree(Đỉnh)

## Câu b. Mô hình biểu diễn tri thức

Mô hình Hệ luật dẫn dạng (F,R) cho kiến thức thu thập được thể hiện như sau:

- **Facts (F)** = tập đỉnh bắt đầu (start\_node), đỉnh kết thúc (end\_node), và các cặp đỉnh cùng khoảng cách của nó trong đồ thị gọi là (path):

- start\_node
- end\_node
- path\_1, path\_2, path\_3, ...path\_n

với path là bộ các (from\_node, to\_node, distance).

- **Rules (R)**

- $\forall i \in \{1, n\}, \text{from\_node\_i}, \text{to\_node\_j} \in \text{path\_i},$   
 $\text{adjacency\_list}[\text{from\_node\_i}] = \{\text{to\_node\_j} | \text{from\_node\_j} = \text{from\_node\_i}, \forall j \in \{1, n\}\} \Rightarrow \text{adjacency\_list}$
- $\exists i \in n, \min(\text{path\_i}[\text{distance}]) \Rightarrow \min\_path$

### Bài toán

Cho đồ thị vô hướng, trọng số không âm. Tìm đường đi ngắn nhất giữa hai đỉnh lần lượt là đỉnh bắt đầu và đỉnh kết thúc.

- **GT:** Danh sách các cặp đỉnh cùng trọng số (khoảng cách) của chúng, và hai đỉnh bắt đầu, kết thúc.

Ví dụ: A B 3, B C 5,...; start\_node = A, end\_node = C

- **KL:** Trả về đường đi ngắn nhất, và khoảng cách của hai đỉnh đã cho.

Ví dụ: [A,B,C], 8

```
1 def shortest_path(self, start_node, end_node):
2     # Ban đầu, tất cả các đỉnh đều được coi như chưa được xét.
3     unvisited_nodes = self.nodes.copy()
4
5     # Khởi tạo một CTDL từ điển distance_from_start chứa các khoảng cách ban đầu (INFINITY)
6     # từ từng đỉnh đến đỉnh start_node. Từ điển này sẽ được cập nhật khi tìm đường ngắn nhất.
7     distance_from_start = {
8         node: (0 if node == start_node else INFINITY) for node in self.nodes
9     }
10
11     # Khởi tạo một CTDL từ điển previous_node chứa các đỉnh trước đó khi xét tới từng điểm
12     # có đường đi ngắn nhất.
13     previous_node = {node: None for node in self.nodes}
14
15     # Xét tập các đỉnh chưa được xét
16     while unvisited_nodes:
```

```

17 # B1: Chọn một đỉnh trong danh sách đỉnh chưa xét mà có khoảng cách ngắn nhất trong
18 # từ điển [distance_from_start]
19 current_node = min(
20 unvisited_nodes, key=lambda node: distance_from_start[node]
21 )
22
23 # B2: Loại đỉnh đang xét khỏi danh sách đỉnh chưa xét
24 unvisited_nodes.remove(current_node)
25
26 # B3: Nếu đỉnh đang xét có khoảng cách tới đỉnh bắt đầu là INFINITY,
27 # nghĩa là hai đỉnh này không liên thông với nhau, quay lại B1
28 if distance_from_start[current_node] == INFINITY:
29     break
30
31 # B4: Xét các đỉnh láng giềng của đỉnh đang xét,
32 # nếu tổng [khoảng cách của đỉnh láng giềng so với đỉnh đang xét] và [khoảng cách đỉnh bắt
33 # đầu so với đỉnh đang xét] bé hơn khoảng cách [đỉnh bắt đầu so với đỉnh láng giềng]
34 # thì cập nhật khoảng cách ngắn nhất, và đỉnh trước đó của láng giềng là đỉnh đang xét
35 for neighbor, distance in self.adjacency_list[current_node]:
36     new_path = distance_from_start[current_node] + distance
37     if new_path < distance_from_start[neighbor]:
38         distance_from_start[neighbor] = new_path
39         previous_node[neighbor] = current_node
40
41 # B5: Nếu đỉnh đang xét là đỉnh kết thúc thì kết thúc bài toán.
42 if current_node == end_node:
43     break
44
45 # Xây dựng CTDL hàng đợi hai đầu cho đường đi
46 path = deque()
47
48 # Thêm vào đầu hàng đợi lần lượt các đỉnh từ cuối đến đầu
49 current_node = end_node
50 while previous_node[current_node] is not None:
51     path.appendleft(current_node)
52     current_node = previous_node[current_node]
53 path.appendleft(start_node)
54
55 # Trả về đường đi ngắn nhất và khoảng cách cần tìm
56 return path, distance_from_start[end_node]

```

Listing 1: Thuật toán Dijkstra

### Câu c. Tổ chức lưu trữ miền tri thức

Tổ chức lưu trữ miền tri thức trong hệ quản trị cơ sở dữ liệu (CSDL) MS SQL Server với các bảng sau:

- Bảng **Vertices**: mỗi dòng định danh đỉnh của đồ thị ( $a, b, c, \dots$ ) với:
  - $id$  là dãy số duy nhất cho từng đỉnh,  $name$  là tên đỉnh thường được đặt theo ký hiệu toán học.
  - $degree$  là thuộc tính bậc của đỉnh ( $deg_a = 2, deg_b = 1, \dots$ ), trong đó một số trường hợp đặc biệt số bậc của đỉnh như  $deg_b = 1$ , cho biết  $b$  là đỉnh treo hay nếu  $deg_b = 0$  thì  $b$  là đỉnh cô lập.

Vertices	
PK	<b>id int NOT NULL</b>
	name char(50) NOT NULL
	degree int NOT NULL

- Bảng **Edges**: mỗi dòng chứa thông tin của một cạnh trong đồ thị bao gồm hai cặp  $id$  của đỉnh từ bảng vertices, khoảng cách (distance) giữa hai đỉnh đó và tên (name) của cạnh đó.

Edges	
FK	<b>from_id char(50) NOT NULL</b>
FK	<b>to_id char(50) NOT NULL</b>
	distance int NOT NULL
	name char(100) NOT NULL

Miền tri thức trên cho phép giải các bài toán liên quan tới phân loại đồ thị, tìm đường đi và chu trình đường đi trong đồ thị.

Ví dụ minh họa chi tiết miền tri thức lý thuyết đồ thị bằng cấu trúc bảng dữ liệu với đồ thị gồm 5 đỉnh A,B,C,D,E với độ dài các cạnh tương ứng như bên dưới:

vertices		
id	name	degree
1	A	2
2	B	3
3	C	4
4	D	5
5	E	2

edges			
from_id	to_id	distance	name
1	2	4	Nguyễn Văn Trỗi
1	3	1	Cộng Hòa
1	4	1	Trường Sơn
2	3	2	Lý Chính Thắng
3	4	5	Nam Kỳ Khởi Nghĩa
4	5	3	Hoàng Văn Thụ

Hình 1: Minh họa miền tri thức định dạng bảng trong CSDL