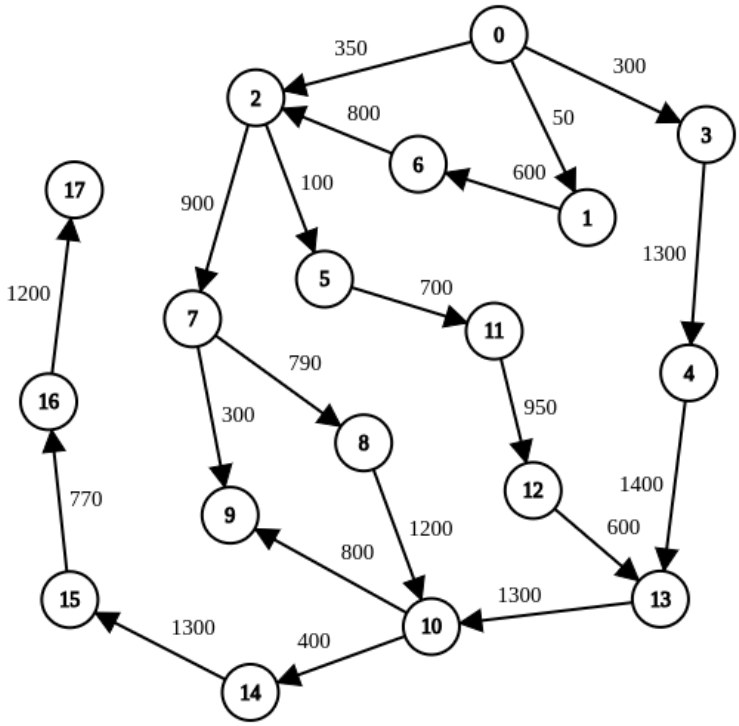
BTTH - NM TTNT - TUAN 1

21280099 - Nguyễn Công Hoài Nam

Ngày 30 tháng 10 năm 2023

I. Chạy tay thuật toán BFS, DFS, UCS

Vì cấu trúc dữ liệu trong Python chỉ mục bắt đầu bằng 0 nên để thuận tiện so sánh ta chuyển tập đỉnh $(\mathbf{V_1}, \ldots, \mathbf{V_{18}})$ thành $(\mathbf{0}, \ldots, \mathbf{17})$



Như vậy bài toán tìm đường đi ngắn nhất từ Đại học Khoa học Tự nhiên $(\mathbf{V_1})$ đến Tân Sơn Nhất $(\mathbf{V_{18}})$ trở thành tìm đường đi ngắn nhất từ đỉnh **0** đến đỉnh **17**

1. Breadth-first Search (BFS)

Các bước thực hiện của thuật toán DFS:

1. Frontier = [0] = START
2. Node 0, Frontier = [1,2,3], Parent[1,2,3] = 0
3. Node 1, Frontier = [2,3,6], Parent[6] = 1
4. Node 2, Frontier = [3,6,5,7], Parent[5,7] = 2
5. Node 3, Frontier = [6,5,7,4], Parent[4] = 3
6. Node 6, Frontier = [5,7,4]
7. Node 5, Frontier = [7,4,11], Parent[11] = 5
8. Node 7, Frontier = [4,11,8,9], Parent[8,9] = 7

9. Node 4, Frontier = [11,8,9,13], Parent[13] = 4
10. Node 11, Frontier = [8,9,13,12], Parent[12] = 11
11. Node 8, Frontier = [9,13,12,10], Parent[10] = 8
12. Node 9, Frontier = [13,12,10]
13. Node 13, Frontier = [12,10]
14. Node 12, Frontier = [10]
15. Node 10, Frontier = [14], Parent[14] = 10
16. Node 14, Frontier = [15], Parent[15] = 14
17. Node 15, Frontier = [16], Parent[16] = 15
18. Node 16, Frontier = [17], Parent[17] = 16
19. Node 17 = GOAL

Đường đi từ đỉnh 0 tới đỉnh 17 là: 0 → 2 → 7 → 8 → 10 → 14→ 15 → 16→ 17 (GOAL).

2. Depth-first Search (DFS)

Các bước thực hiện của thuật toán DFS:

1. Frontier = [0] = START
2. Node 0, Frontier = [3,2,1], Parent[3,2,1] = 0
3. Node 3, Frontier = [4,2,1], Parent[4] = 3
4. Node 4, Frontier = [13,2,1], Parent[13] = 4
5. Node 13, Frontier = [10,2,1], Parent[10] = 13
6. Node 10, Frontier = [14,9,2,1], Parent[14,9] = 10
7. Node 14, Frontier = [15,9,2,1], Parent[15] = 14
8. Node 15, Frontier = [16,9,2,1], Parent[16] = 15
9. Node 16, Frontier = [17,9,2,1], Parent[17] = 16
10. Node 17 = GOAL

Đường đi từ đỉnh 0 tới đỉnh 17 là: 0 → 3 → 4 → 13 → 10 → 14→ 15 → 16→ 17 (GOAL).

3. Uniform-cost Search (UCS)

Các bước thực hiện của thuật toán UCS:

1. PQ = (0, 0) = (START, 0). (PQ là Priority Queue)
2. PQ = (1, 50), (3, 300), (2, 350), Parent[1, 2, 3] = 0
3. PQ = (3, 300), (2, 350), (6, 650), Parent[6] = 1
4. PQ = (2, 350), (6,650), (4, 1600), Parent[4] = 3
5. PQ = (5, 450), (6, 650), (7,1250), (6,1600), Parent[5, 7] = 2
6. PQ = (6,650), (11,1150), (7,1250), (4,1600), Parent[11] = 5
7. PQ = (7,1250), (4,1600), (12,2100), Parent[12] = 11
8. PQ = (9,1550), (4,1600), (8,2040), (12,2100), Parent[8, 9] = 7
9. PQ = (4,1600), (8,2040), (12,2100), (10,3240), Parent[10] = 8
10. PQ = (8,2040), (12,2100), (13,2200), (10,3240), Parent[13] = 4

11. PQ = (12, 2100), (13, 2200), (10,3240)
12. PQ = (10, 3240), (13, 2200)
13. PQ = (10, 3240)
14. PQ = (14, 3640), Parent[14] = 10
15. PQ = (15, 4940), Parent[15] = 14
16. PQ = (16, 5710), Parent[16] = 15
17. PQ = (17, 6910) = GOAL

Đường đi từ đỉnh 0 tới đỉnh 17 là: 0 → 2 → 7 → 8 → 10 → 14→ 15 → 16→ 17 (GOAL) với chi phí 6910.

II. Cài đặt và thực thi thuật toán

1. Kiểm tra tính đúng đắn của thuật toán

Nhìn chung code chạy cho ra kết quả đúng và không có sai sót gì. Tuy nhiên còn một đoạn code thừa

```
def BFS(graph, start, end):
    visited = []
    frontier = Queue()

    #Them node start vao frontier va visited
    frontier.put(start)
    visited.append(start)

    #Start khong co node cha
    parent = dict()
    parent[start] = None

    path_found = False

    while True:
        if frontier.empty():
            raise Exception("No way Exception")
        current_node = frontier.get()
        #Doan code thua
        visited.append(current_node)
        #

        #Kiem tra current node co la end hay khong
        if current_node == end:
            path_found = True
            break

        for node in graph[current_node]:
            if node not in visited:
                frontier.put(node)
                parent[node] = current_node
                visited.append(node)

    #Xay dung duong di
    path = []
    if path_found:
        path.append(end)
        while parent[end] is not None:
            path.append(parent[end])
            end = parent[end]
        path.reverse()
    return path
```

Trong code BFS, ta thấy current_node được thêm vào tập visited này sau khi lấy ra frontier, điều này sẽ gây ra trùng lặp vì nó có thể sẽ được thêm vào tập visited một ở vòng lặp trước đó rồi.
Ta quan tâm đến hai đoạn code

```
    while True:
        current_node = frontier.get()
        #Doan code thua
        visited.append(current_node)
        #
        for node in graph[current_node]:
```

```
            if node not in visited:
                frontier.put(node)
                parent[node] = current_node
                visited.append(node)
```

Ví dụ, đỉnh 0 có 3 đỉnh kề 1,2,3. Ở vòng lặp trước đã thêm 1,2,3 vào visited. Vòng lặp sau, khi lấy ra đỉnh 1 ta lại thêm đỉnh 1 vào visited.
Ngoài ra ta đã thêm đỉnh START (0) vào visited ở đầu. Sau đó vào vòng lặp ta lại thêm nó vào một lần nữa. Để kiểm chứng ta có thể in tập visited khi bỏ và chưa bỏ đoạn code đó

```
#Truoc khi bo doan code
Tap Visited:
[0, 0, 1, 2, 3, 1, 6, 2, 5, 7, 3, 4, 6, 5, 11, 7, 8, 9, 4, 13, 11, 12, 8, 10, 9, 13, 12, 10,
    14, 14, 15, 15, 16, 16, 17, 17]
#Sau khi bo
Tap Visited:  [0, 1, 2, 3, 6, 5, 7, 4, 11, 8, 9, 13, 12, 10, 14, 15, 16, 17]
```

Ở cả 3 thuật toán đều có đoạn code thừa tương tự như vậy.

2. Kiểm tra kết quả thuật toán với kết quả chạy tay

Kết quả chạy thuật toán

```
#Ket qua su dung thuat toan BFS:
 [0, 2, 7, 8, 10, 14, 15, 16, 17]
 #Ket qua su dung thuat toan DFS:
 [0, 3, 4, 13, 10, 14, 15, 16, 17]
#Ket qua su dung thuat toan UCS:
 [0, 2, 7, 8, 10, 14, 15, 16, 17] voi tong chi phi la 6910
```

Như vậy kết quả chạy tay và thuật toán là giống nhau.