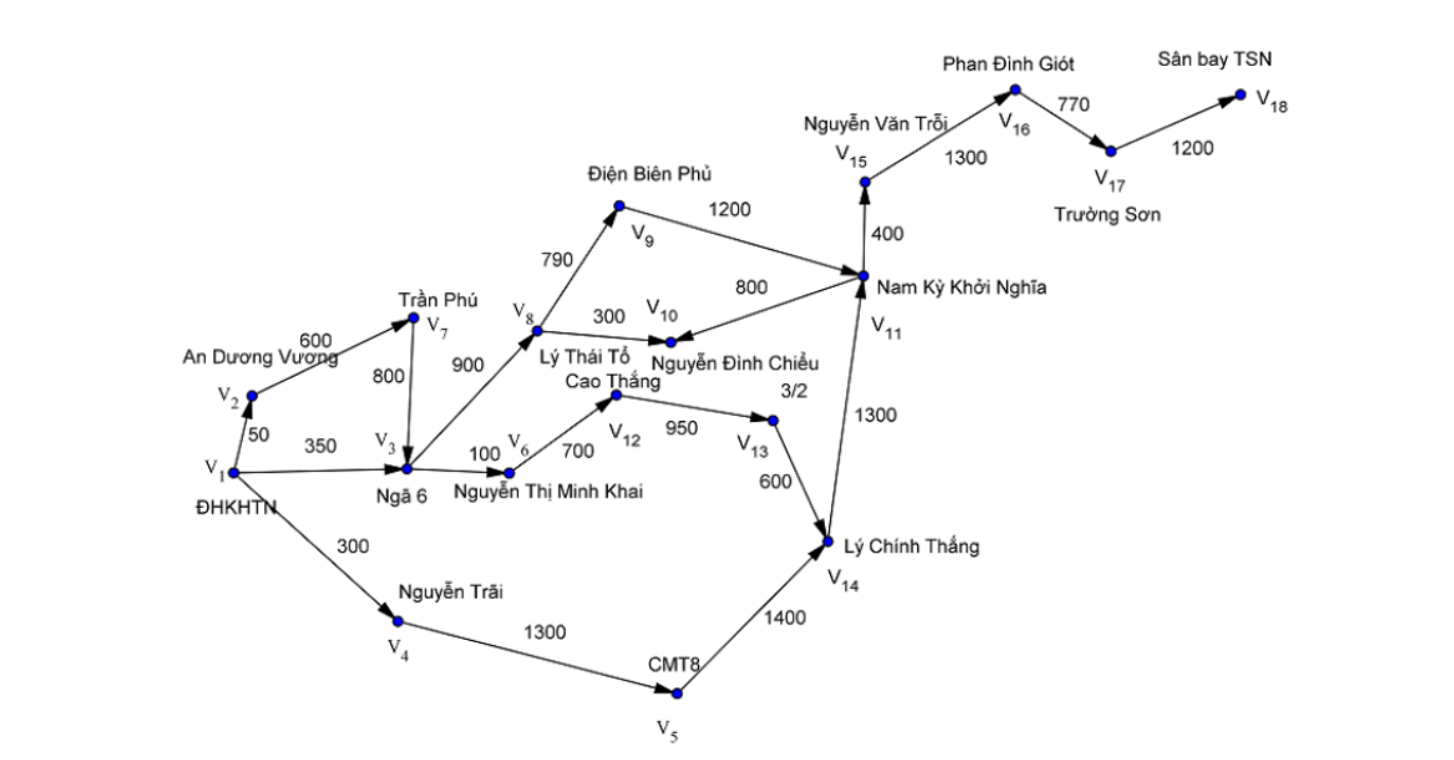
Họ và tên: Nguyễn Vạn Phúc Huy

MSSV: 23110163

Lớp: Thực hành thứ 4 (tiết 4-5)

**Bài tập thực hành tuần 1**



Tìm đường đi ngắn nhất từ trường Đại học Khoa học Tự Nhiên (V1) 🡪 Sân bay Tân Sơn Nhất (V18)

**BFS (Breadth – First Search)**

1. L = [V1] (trạng thái ban đầu)
2. Node = V1, L =[V2, V3, V4], father[V2, V3, V4] = V1
3. Node = V2, L= [V3, V4, V7], father[V7] = V2
4. Node = V3, L = [V4, V7, V6, V8], father[V6, V8] = V3
5. Node = V4, L = [V7, V6, V8, V5], father[V5] = V4
6. Node = V7, L = [V6, V8, V5], father[V3] = V7
7. Node = V6, L = [V8, V5, V12], father[V12] = V6
8. Node = V8, L = [V5, V12, V9, V10], father[V9, V10] = V8
9. Node = V5, L = [V12, V9, V10, V14], father[V14] = V5
10. Node = V12, L = [V9, V10, V14, V13], father[V13] = V12
11. Node = V9, L = [V10, V14, V13, V11], father[V11] = V9
12. Node = V10, L = [V14, V13]
13. Node = V14, L = [V13, V11], father[V11] = V14
14. Node = V13, L = [V11], father V[14] = V13
15. Node = V11, L = [V15], father[V15] = V11
16. Node = V15, L = [V16], father[V16] = V15
17. Node = V16, L = [V17], father[V17] = V16
18. Node = V17, L = [V18], father[V18] = V17
19. Node = 18 (trạng thái kết thúc) => dừng

**Đường đi ngắn nhất mà BFS tìm được:**

1→3→8→9→11→15→16→17→18

1→4→5→14→11→15→16→17→18

**DFS (Depth First Search)**

1. L = [V1] (trạng thái ban đầu)
2. Node = V1, L = [V4, V3, V2], father[V4, V3, V2] = V1
3. Node = V4, L = [V5, V3, V2], father[V5] = V4
4. Node = V5, L = [V14, V3, V2], father[V14] = V5
5. Node = V14, L = [V11, V3, V2], father[V11] = V14
6. Node = V11, L = [V15, V3, V2], father[V15] = V11
7. Node = V15, L = [V16, V3, V2], father[V16] = V15
8. Node = V16, L = [V17, V3, V2], faher[V17] = V16
9. Node = V17, L = [V18, V3, V2], father[V18] = V17
10. Node = V18 (trạng thái kết thúc) => dừng

**Đường đi ngắn nhất mà BFS tìm được:**

1→4→5→14→11→15→16→17→18

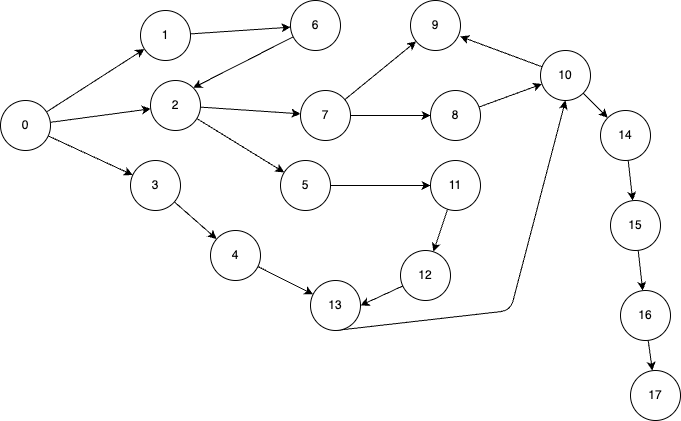
**UCS (Unifrom – Cost Search)**

1. PQ = {(V1, 0)}
2. PQ = {(V2, 50), (V4, 300), (V3, 350)}
3. PQ = {(V4, 300), (V3, 350), (V7, 650)}
4. PQ = {(V6, 450), (V8, 1250), (V5, 1600)}
5. PQ = {(V12, 1150), (V8, 1250), (V5, 1600)}
6. PQ = {(V10, 1550), (V5, 1600), (V9, 2040), (V13, 2100)}
7. PQ = {(V11, 3240), (V14, 2700)}
8. PQ = {(V15, 3640)}
9. PQ = {(V16, 4940)}
10. PQ = {(V17, 5710)}
11. PQ = {(V18, 6910)}

**Đường đi ngắn nhất mà BFS tìm được:**

1→3→8→9→11→15→16→17→18 (6910)

**Đồ thị Input.txt có dạng:**



**BFS (Breadth – First Search)**

Node đích: 17

1. L = [0] (trạng thái ban đầu)
2. Node = 0, L = [1, 2, 3], father[1, 2, 3] = 0
3. Node = 1, L = [2, 3, 6], father[2, 3, 6] = 1
4. Node = 2, L = [3, 6, 5, 7], father[5, 7] = 2
5. Node = 3, L = [6, 5, 7, 4], father[4] = 3
6. Node = 6, L = [5, 7, 4]
7. Node = 5, L = [7, 4, 11], father[11] = 5
8. Node = 7, L = [4, 11, 8, 9], father[8, 9] = 7
9. Node = 4, L = [11, 8, 9, 13], father[13] = 4
10. Node = 11, L = [8, 9, 13, 12], father[12] = 11
11. Node = 8, L = [9, 13, 12, 10], father[10] = 8
12. Node = 9, L = [13, 12, 10]
13. Node = 13, L = [12, 10], father[10] = 13
14. Node = 12, L = [10]
15. Node = 10, L = [14], father[14] = 10
16. Node = 14, L = [15], father[15] = 14
17. Node = 15, L = [16], father[15] = 15
18. Node = 16 L = [17], father[15] = 16
19. Node = 17 (trạng thái kết thúc) => dừng

**Đường đi do BFS tìm được:**

0→2→7→8→10→14→15→16→17

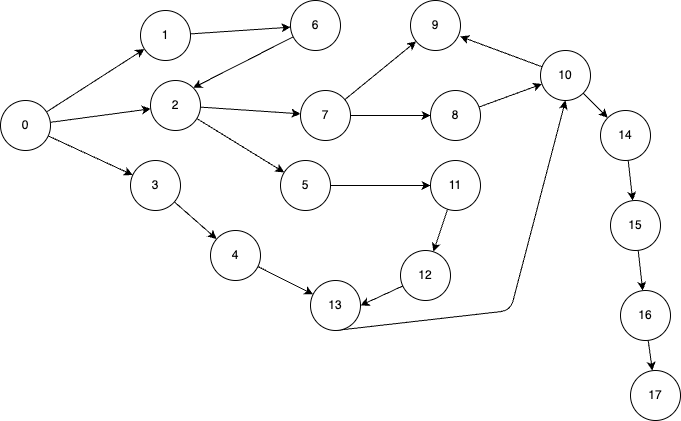
0→3→4→13→10→14→15→16→17

**Đường đi cho BFS (code) tìm được:**

0→2→7→8→10→14→15→16→17

Nhận xét: Đường đi do thuật toán BFS cài sẵn trong code cho ra giống với đường đi giải tay, chỉ đưa ra 1 đường trong hai đường đi tìm được

* Vậy thuật toán cài sẵn trong code là đúng, nhưng chưa đủ



**DFS (Depth First Search)**

1. L = [0] (trạng thái ban đầu)
2. Node = 0, L = [3, 2, 1], father[3, 2, 1] = 0
3. Node = 3, L = [4, 2, 1], father[4] = 3
4. Node = 4, L = [13, 2, 1], father[13] = 4
5. Node = 13, L= [10, 2, 1], father[10] = 13
6. Node = 10, L = [14, 2, 1], father[14] = 10
7. Node = 14, L = [15, 2, 1], father[15] = 14
8. Node = 15, L = [16, 2, 1], father[16] = 15
9. Node = 16, L = [17, 2, 1], father[17] = 16
10. Node = 17 (trạng thái kết thúc) => dừng

**Đường đi do DFS tìm được:**

0→3→4→13→10→14→15→16→17

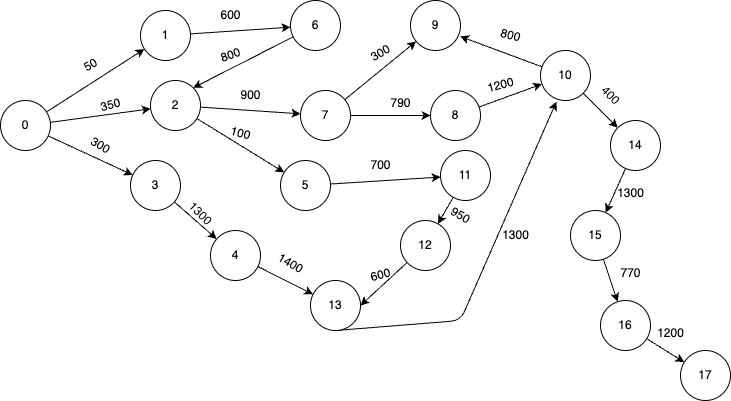
**Đường đi cho DFS (code) tìm được:**

0→3→4→13→10→14→15→16→17

Nhận xét: Đường đi do thuật toán DFS cài sẵn trong code cho ra giống với đường đi giải tay

* Vậy thuật toán cài sẵn trong code là đúng

**Đồ thị InputUCS.txt có dạng:**



**UCS ( Uniform Cost Search)**

1. PQ = {(0, 0)}
2. PQ = {(1, 50), (3, 300), (2, 350)}
3. PQ = {(3, 300), (2, 350), (6, 650)}
4. PQ = {(2, 350), (6, 650), (4, 1600)}
5. PQ = {(5, 450), (6, 650), (7, 1250), (4, 1600)}
6. PQ = {(11, 1050), (7, 1250), (4, 1600)}
7. PQ ={(9, 1550), (11, 2000), (8, 2040), (13, 3000)}
8. PQ = {(8, 2040), (12, 2950), (13, 3000)}
9. PQ = {(13, 3000), {(10, 3240)}
10. PQ = {(10, 3240)}
11. PQ = {(14, 3640)}
12. PQ = {(15, 4940)}
13. PQ = {(16, 5710)}
14. PQ = {(17, 6910)}

**Đường đi ngắn nhất do UCS tìm được:**

0→2→7→8→10→14→15→16→17

**Đường đi ngắn nhất do UCS (code) tìm được:**

0→2→7→8→10→14→15→16→17

**Nhận xét:** Đường đi do thuật toán UCS cài sẵn trong code cho ra giống với đường đi giải tay

* Vậy thuật toán cài sẵn trong code là đúng

**Kếtl luận**

Thuật toán đã cho trên code chạy ra kết quả hoàn toàn khớp với kết quả giải tay. Nhưng có một vấn đề ở chỗ visited.append(start) ở đầu hàm là dư thừa. Trong thuật toán , node bắt đầu (start) đã được thêm vào visited ngay sau khi nó được lấy ra từ hàng đợi frontier. Vì vậy, không cần thiết phải thêm nó vào visited ngay từ đầu.