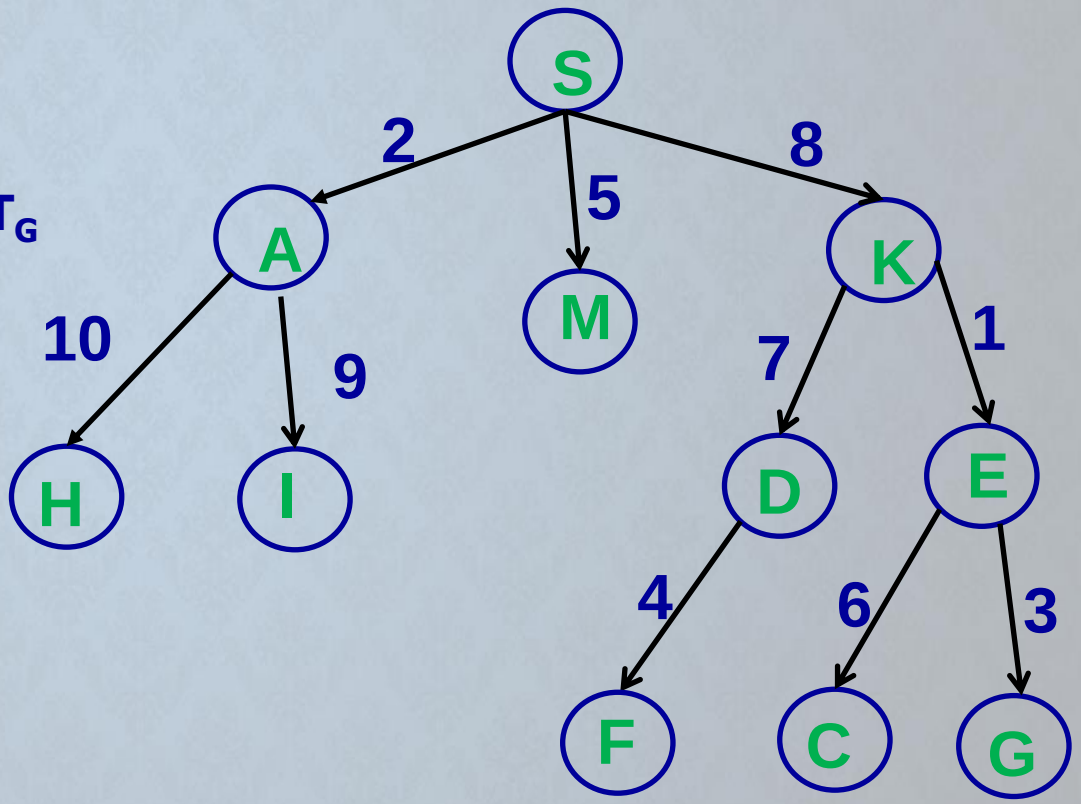
BÀI 10: THỰC NGHIỆM BÀI TOÁN TÌM KIẾM

Yêu cầu: Cho đồ thị như sau:

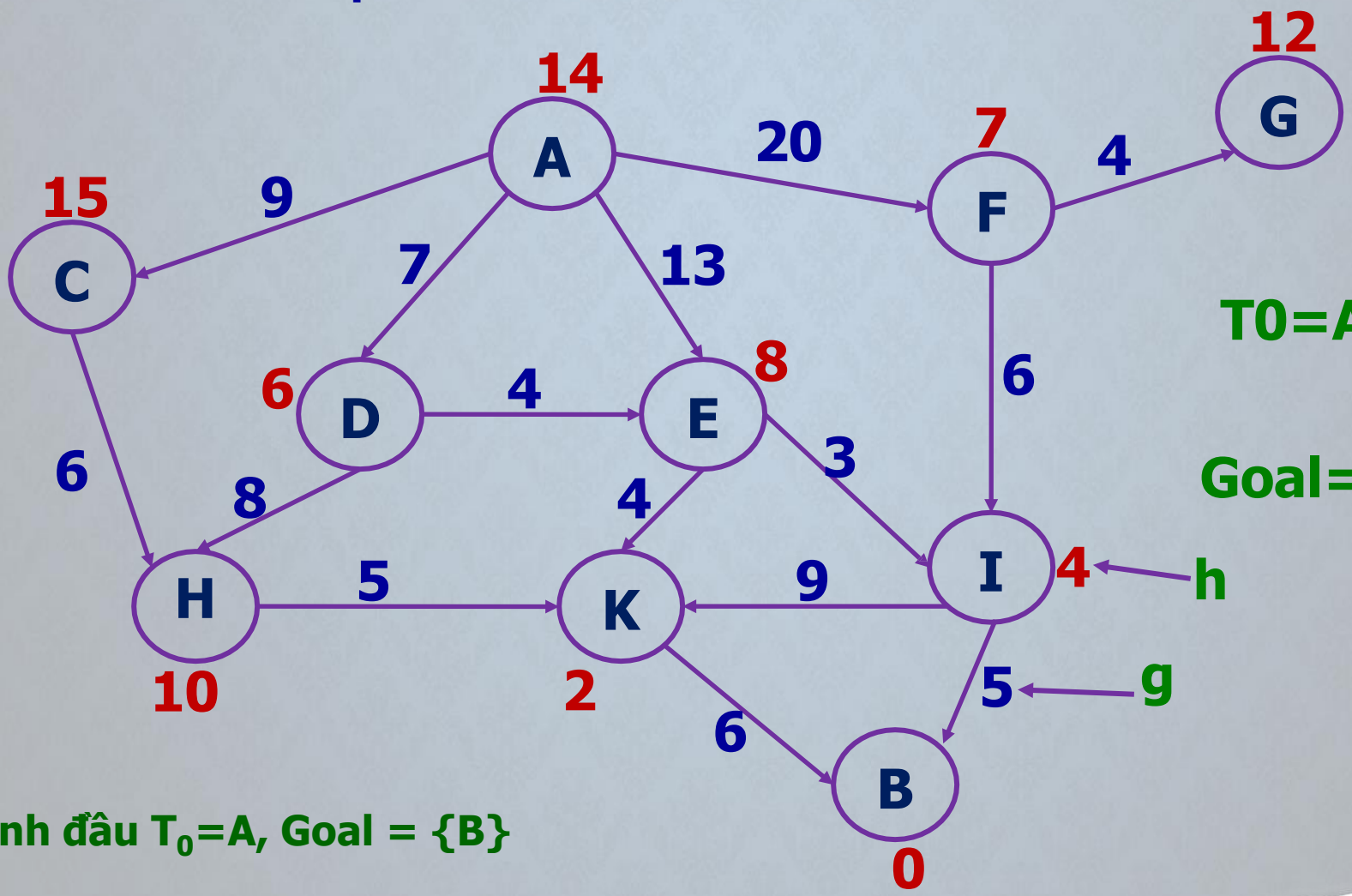
Đỉnh đầu T_0=S, T_G = {G}

Tìm đường đi p từ To đến T_G bằng phương pháp A^T?

?



Yêu cầu: Cho đồ thị như sau:



T0=A

Goal={B}

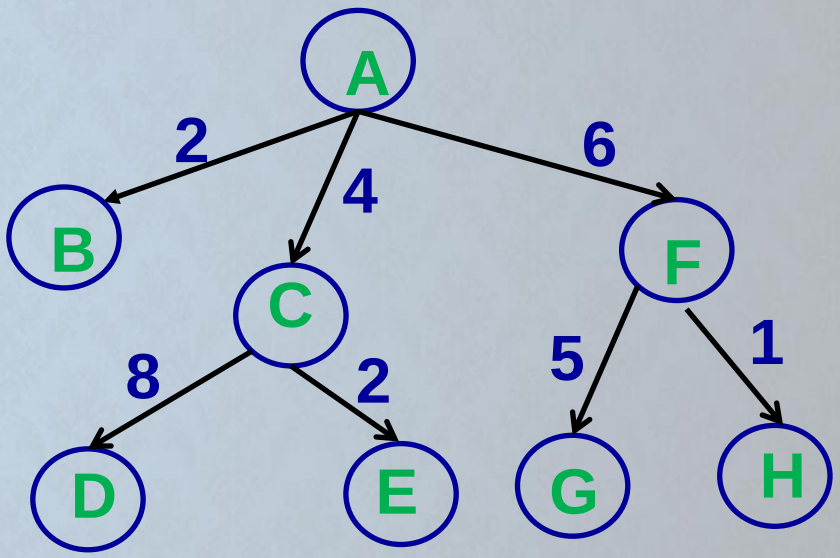
Đỉnh đầu T_0=A, Goal = {B}

Tìm đường đi p từ To đến Goal bằng phương pháp A^* ?

Ví dụ: Cho đồ thị như sau:

Đỉnh đầu T_0=A, T_G = {D, H}

Tìm đường đi p từ To đến T_G bằng phương pháp A^T?



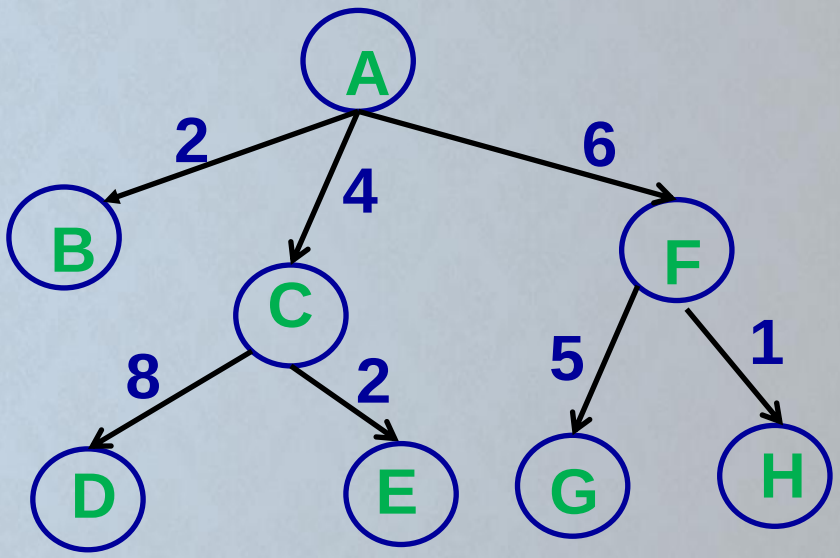
- **Input:**
- **Output:**

?

Cho đồ thị như sau:

Đỉnh đầu T_0=A, T_G = {D, H}

Tìm đường đi p từ To đến T_G bằng phương pháp A^T?



- **Input: Đồ thị, đỉnh đầu T_0=A, T_G = {D, H}**
- **Output: Có đường đi từ T_0 đến T_G không?**

Nếu có thì in ra đường đi, nếu không thì đưa ra thông báo

TÌM KIẾM THEO A^T

Tạo đồ thị theo đề bài

```
graph = {"A": {"B": 2, "C" : 4, "F": 6},
         "B": {},
         "C": {"D": 8, "E": 2},
         "D": {},
         "E": {},
         "F": {"G": 5, "H": 1},
         "G": {},
         "H": {}}
```

Hàm in đường đi và chi phí

```
def print_path_and_cost(start, goal, parent, g):
    path = []
    current = goal
    while current != start:
        path.append(current)
        current = parent[current]
    path.append(start)
    path.reverse()
    print("Đường đi:", ' -> '.join(path))
    print("C(p) = ", g[goal])
```

TÌM KIẾM THEO AT

Hàm AT

```python
def AT(graph, start, goals):
    MO = [start]  # Danh sách các đỉnh đã được duyệt
    g = {start: 0}  # Chi phí tới từng đỉnh
    DONG = []  # Danh sách các đỉnh đã xét xong
    parent = {}  # Lưu trữ cha của mỗi đỉnh

    while MO:
        # Lấy đỉnh n có chi phí g(n) nhỏ nhất từ tập MO
        min_cost = float('inf')
        for vertex in MO:
            if vertex in g:
                cost = g[vertex]
            else:
                cost = float('inf')
            if cost < min_cost:
                min_cost = cost
                n = vertex

        if n in goals:
            print_path_and_cost(start, n, parent, g)
            return True

        MO.remove(n)  # Xóa đỉnh n khỏi tập MO
        DONG.append(n)  # Thêm đỉnh n vào tập đã xét

        for m in graph.get(n, {}):  # Duyệt qua các đỉnh kề của n
            cost = graph[n][m]  # Chi phí từ n đến m
            new_cost = g.get(n, float('inf')) + cost

            # Nếu m đã có cha và đường đi mới ngắn hơn
            if m in parent and new_cost < g[m]:
                g[m] = new_cost
                parent[m] = n

            # Nếu m chưa được duyệt
            elif m not in MO and m not in DONG:
                g[m] = new_cost
                parent[m] = n
                MO.append(m)

    return False  # Không tìm thấy đường đi đến đỉnh đích
```

TÌM KIẾM THEO A^T

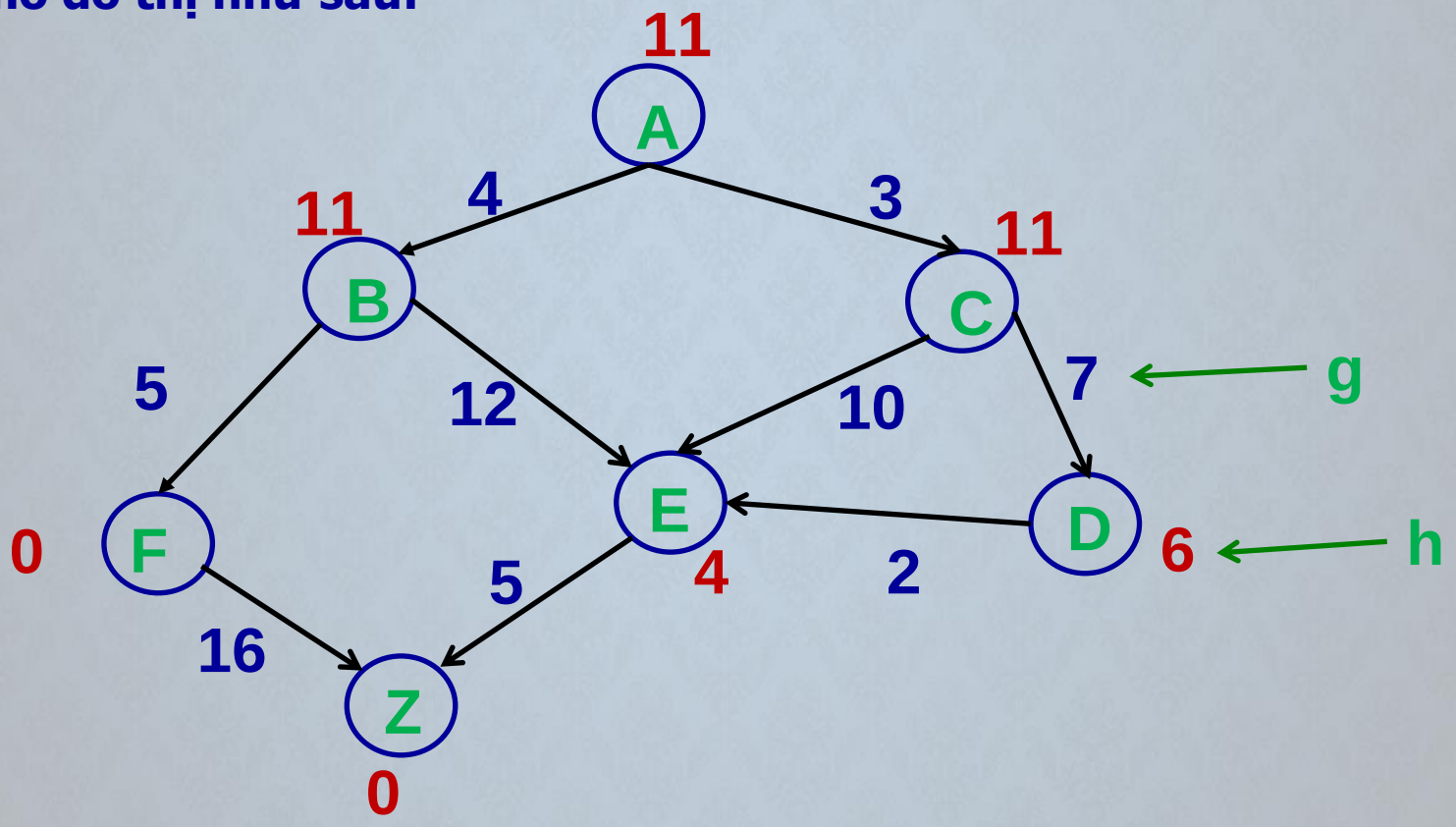
Kết quả sau khi thực hiện gọi hàm tìm kiếm A^T với đỉnh đầu là A và đích là {D,H}.

```
start = "A"
goals = ["D", "H"]
AT(graph, start, goals)
```

```
Đường đi: A -> F -> H
C(p) =  7
True
```

TÌM KIẾM THEO A*

Ví dụ: Cho đồ thị như sau:



Đỉnh đầu T_0=A, Goal = {F, Z}

Tìm đường đi p từ To đến Goal bằng phương pháp A* ?

TÌM KIẾM THEO A*

Tạo đồ thị theo đề bài

```
graph = {
    'A': [('B', 4, 11), ('C', 3, 11)],
    'B': [('F', 5, 11), ('E', 12, 4)],
    'C': [('D', 7, 6), ('E', 10, 4)],
    'D': [('E', 2, 4)],
    'E': [('Z', 5, 0)],
    'F': [('Z', 16, 0)]
}
```

TÌM KIẾM THEO A*

Tạo hàm h(n) theo đề bài

```
[ ]  def h(node):
         # Hàm tính h(n) - ở đây mình sẽ giả sử h(n) là chi phí trực tiếp từ n đến đích Z
         h = {
             'A': 11,
             'B': 11,
             'C': 11,
             'D': 6,
             'E': 4,
             'F': 0,
             'Z': 0
         }
         return h[node]
```

TÌM KIẾM THEO A*

Hàm in đường đi và chi phí

```
[ ]  def print_path_and_cost(start, goal, parent, g):
         path = []
         current = goal
         while current != start:
             path.append(current)
             current = parent[current]
         path.append(start)
         path.reverse()
         print("Đường đi:", ' -> '.join(path))
         print("C(p) = ", g[goal])
```

TÌM KIẾM THEO A*

Hàm A*

```python
def A_star(graph, start, goals):
    MO = [start]  # Tập đỉnh mở, ban đầu chứa đỉnh start
    DONG = []  # Tập đỉnh đã xét, ban đầu rỗng
    g = {start: 0}  # Chi phí từ start đến các đỉnh khác
    f = {start: h(start)}  # Giá trị f(n) cho mỗi đỉnh
    parent = {}  # Lưu trữ cha của mỗi đỉnh
    while MO:
        # Chọn đỉnh n có f(n) nhỏ nhất từ tập MO
        min_f = float('inf')
        min_node = None
        for node in MO:
            if f[node] < min_f:
                min_f = f[node]
                min_node = node
        n = min_node
        if n in goals:
            # In ra đường đi và kết thúc thuật toán
            print_path_and_cost(start, n, parent, g)
            print(parent)
            return True
        MO.remove(n)  # Xóa đỉnh n khỏi tập MO
        DONG.append(n)  # Thêm đỉnh n vào tập DONG
        for m, cost_g, cost_h in graph.get(n, []):
            cost_g_new = g[n] + cost_g  # Chi phí mới từ start đến m
            if m not in MO and m not in DONG:
                # Mở rộng đỉnh m
                g[m] = cost_g_new
                f[m] = g[m] + cost_h
                parent[m] = n
                MO.append(m)
            elif m in MO and g[m] > cost_g_new:
                # Cập nhật đỉnh m nếu có chi phí mới tốt hơn
                g[m] = cost_g_new
                f[m] = g[m] + cost_h
                parent[m] = n
    return False  # Không tìm thấy đường đi đến đỉnh đích
```

TÌM KIẾM THEO A*

Kết quả sau khi thực hiện gọi hàm tìm kiếm A* với đỉnh đầu là A và đích là {Z, F}.

```
print("A*")
A_star(graph, 'A', ['Z','F'])
```

```
A*
Đường đi: A -> C -> D -> E -> Z
C(p) =  17
{'B': 'A', 'C': 'A', 'D': 'C', 'E': 'D', 'F': 'B', 'Z': 'E'}
True
```