## Ví dụ

Input:

2 ← số queries | đồ thị

4 2 ← số đỉnh và cạnh của đồ thị 1

1 2

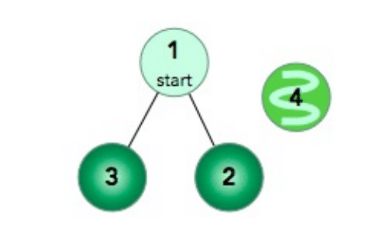
1 3

1

3 1 ← số đỉnh và cạnh của đồ thị 2

2 3

2



Đồ thị 1:

4 2

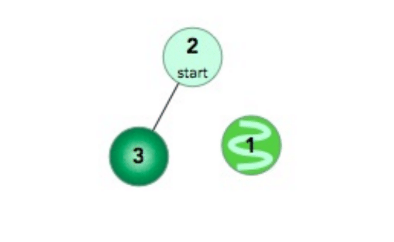
1 2

1 3

1

⇒ Độ dài từ đỉnh 1 đến đỉnh 2 và 3 là 6. Không có đường đi từ 1 đến 4

⇒ Output: 6 6 -1



Đồ thị 2:

3 1

2 3

2

⇒ Không có đường đi từ 2 đến 1.

Độ dài đường đi từ đỉnh 2 đến 3 là 6. ⇒ Output: -1 6

## Hướng giải:

* Dùng BFS bắt đầu từ đỉnh S để tìm đường đi ngắn nhất từ đỉnh S đến tất cả các đỉnh khác.
* Dùng mảng dist[v] để lưu lại khoảng cách ngắn nhất từ đỉnh S đến đỉnh v. Khi đó, nếu mình đến được đỉnh v lúc duyệt các đỉnh kề của u thì dist[v] = dist[u] + 1.

## **Giải thuật cho 1 đồ thị:**

B1: Đọc dữ liệu, tạo danh sách kề từ dữ liệu input.

B2: Khởi tạo:

Mảng dist = [] có V phần tử với giá trị -1

Mảng visited = [] có V phần tử với giá trị False.

1 queue rỗng

B3: Dùng phép duyệt BFS:

Thêm đỉnh S vào queue, đánh dấu dist[S] = 0, visited[S] = True.

Thực hiện cho đến khi queue rỗng:

Lấy đỉnh u khỏi queue

Duyệt qua tất cả đỉnh kề v của u:

Nếu visited[v] == False:

Thêm đỉnh v vào queue

cập nhật dist[v] = dist[u] + 1

visited[v] = True

B4: Xuất dist[v]\*6 nếu v khác S và 1 <= v <= V.

## Mã giả

BFS(S):

q = Queue() // Khởi tạo queue rỗng

dist[S] = 0

visited[S] = True

q.push(S)

while q not empty:

u = q.pop()

for v in graph[u]:

if visited[v] == False:

q.push(v)

dist[v] = dist[u] + 1

visited[v] = True

main():

read(V, E)

graph = [] \* V

for i = 0 to E:

read(u, v)

u--

v--

graph[u].append(v)

graph[v].append(u)

read(S)

dist = [-1] \* V

visited = [False] \* V

BFS(S)

for i = 0 to V:

if i == S:

continue

if dist[i] != -1:

print(dist[i] \* 6)

else:

print(-1)

Độ phức tạp: O(Q\*(V + E))