Breadth First Search: Shortest Reach

Big-O Blue – Lecture 5: BFS

Tóm tắt đề bài

Tóm tắt đề bài

Cho đồ thị vô hướng gồm **N** đỉnh, các đỉnh được đánh số từ **1** đến **N** và đỉnh bắt đầu **S**.

Tất cả cạnh trong đồ thị đều có độ dài là **6**.

Tìm độ dài ngắn nhất từ **S** đến tất cả các đỉnh còn lại.

- Lưu ý: Có nhiều testcase trong 1 input.

Giải Thích Ví Dụ

Mô tả Input/Output

Input:

Q
N M
u1 v1
u2 v2
…
uM vM
S
...

2 ≤ N ≤ 1000, 1 ≤ M ≤ N(N-1)/2
1 ≤ u, v, S ≤ N

Output:

- Mỗi testcase: In **N-1** số nguyên là độ dài ngắn nhất từ **S** đến đỉnh tương ứng. Nếu không tồn tại đường đi, in **-1**.
- Lưu ý: In theo thứ tự đỉnh tăng dần, và không in độ dài ngắn nhất từ đỉnh **S** đến chính nó.

Ví dụ 1

Input:

2

4 2

1 2

1 3

1

3 1

2 3

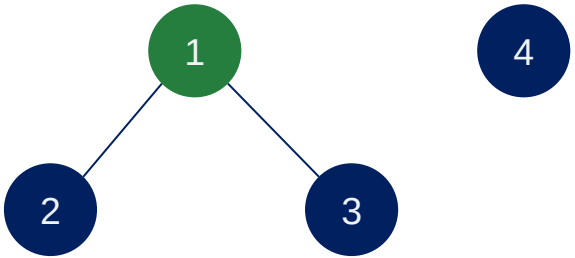
2

Output:

6 6 – 1

-1 6

Đồ thị của testcase đầu tiên:



Độ dài ngắn nhất từ đỉnh 1 □ 2: 6.

Độ dài ngắn nhất từ đỉnh 1 □ 3: 6.

Không có đường đi từ 1 đến 4.

□ 6 6 -1

Ví dụ 1

Input:

2

4 2

1 2

1 3

1

3 1

2 3

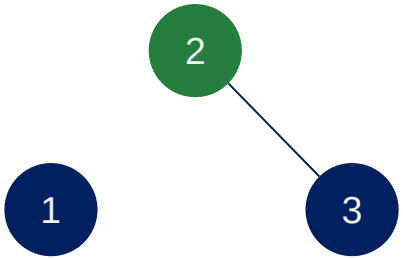
2

Output:

6 6 – 1

-1 6

Đồ thị của testcase thứ hai:



Không có đường đi từ 2 đến 1.

Độ dài ngắn nhất từ đỉnh 2 -> 3: 6.

-1 6

Ví dụ 2

Input:

1

6 6

5 6

5 4

4 2

2 3

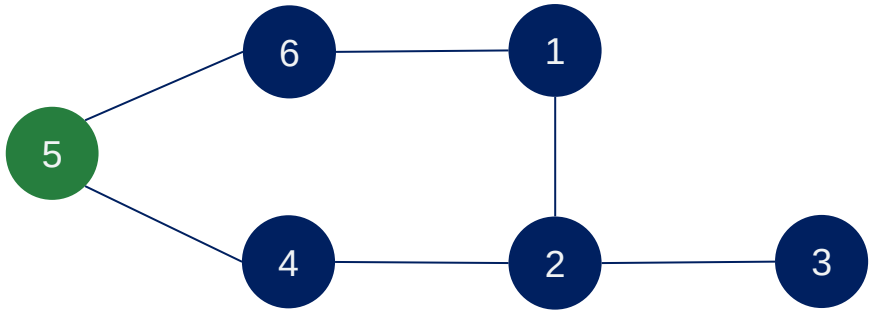
6 1

1 2

5

Output:

12 12 18 6 6



Độ dài ngắn nhất từ đỉnh 5 đến:

- Đỉnh 1: 12
- Đỉnh 2: 12.
- Đỉnh 3: 18.
- Đỉnh 4: 6.
- Đỉnh 6: 6.

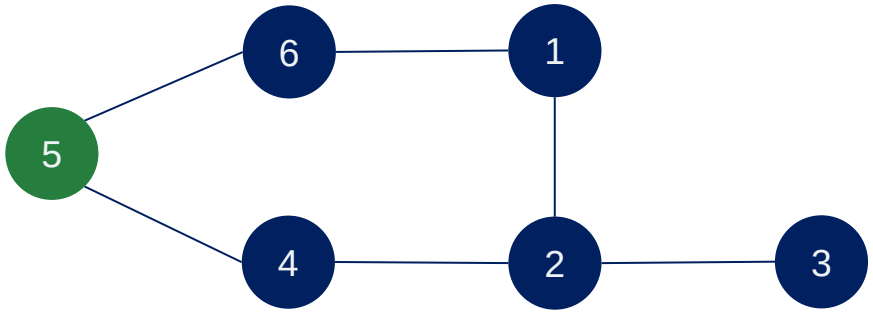
 12 12 18 6 6

Ý Tưởng

Ý tưởng

- Tuy là cạnh có trọng số, nhưng tất cả trọng số đều bằng nhau -> Độ dài đường đi phụ thuộc vào số cạnh đi qua.
- Tìm đường đi ngắn nhất từ 1 đỉnh đến tất cả đỉnh khác trong đồ thị không trọng số -> Dùng thuật toán BFS.
- Dùng mảng **dist[v]** để lưu độ dài đường đi ngắn nhất từ **S** đến **v**.
- Giả sử đã có được độ dài ngắn nhất từ **S** đến **u** và từ **u** có cạnh nối đến **v** (và chưa tìm được đường đi ngắn nhất từ **S** tới **v**) -> Độ dài đường đi ngắn nhất từ **S** đến **v** là **dist[u] + 1**.

Ví dụ 2

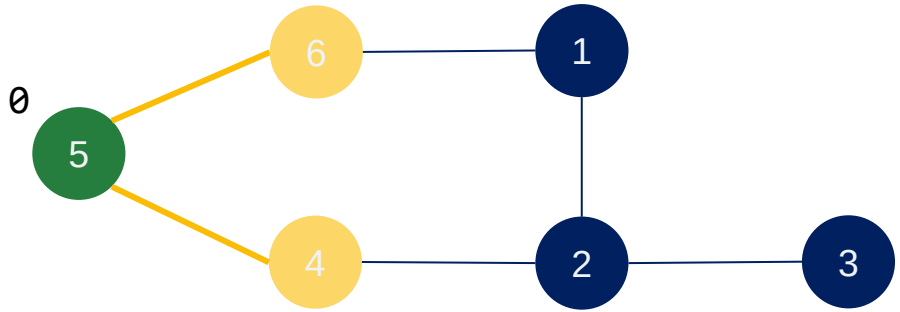


Queue:

- Bắt đầu từ đỉnh 5.

- dist[5] = 0, visited[5] = true, và cho đỉnh 5 vào queue.

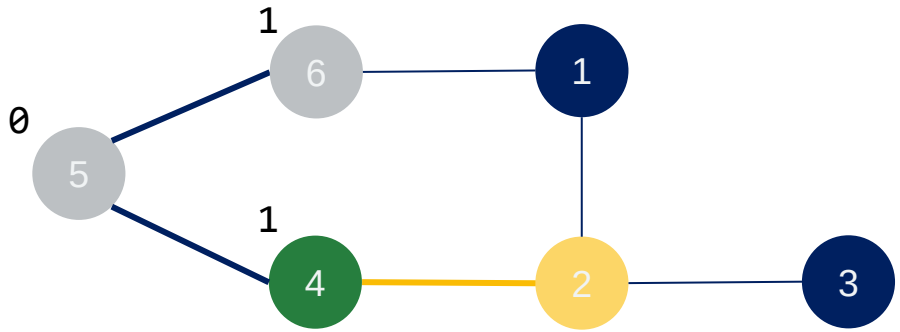
Ví dụ 2



Queue:

- Lấy đỉnh đầu trong queue: Đỉnh 5.
- Duyệt qua các đỉnh kề của 5: đỉnh 4 và 6
  - dist[4] = dist[5] + 1, dist[6] = dist[5] + 1
  - visited[4] = true, visited[6] = true.
  - Cho đỉnh 4 và 6 vào queue

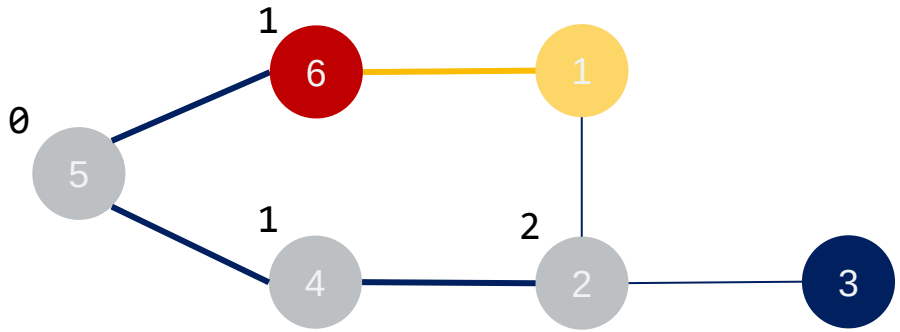
Ví dụ 2



Queue:

- Lấy đỉnh đầu trong queue: Đỉnh 4.
- Duyệt qua các đỉnh kề của 4: đỉnh 2.
  - dist[2] = dist[4] + 1
  - visited[2] = true
  - Cho đỉnh 2 vào queue

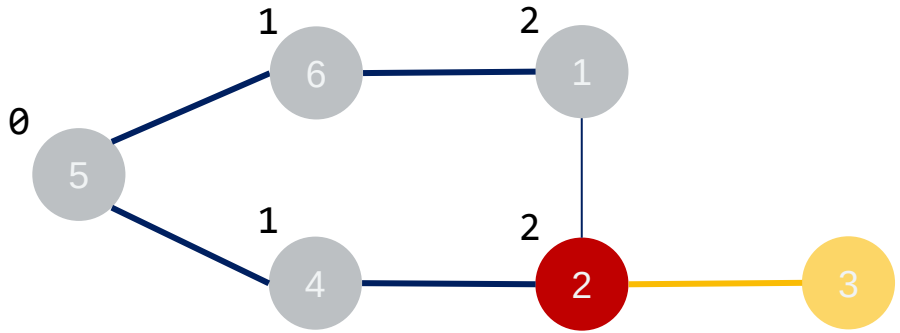
Ví dụ 2

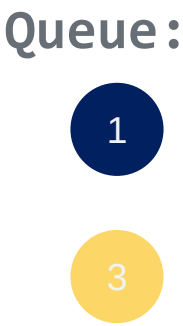


Queue:

- Lấy đỉnh đầu trong queue: Đỉnh 6.
- Duyệt qua các đỉnh kề của 6: đỉnh 1.
  - dist[1] = dist[6] + 1
  - visited[1] = true
  - Cho đỉnh 1 vào queue

Ví dụ 2



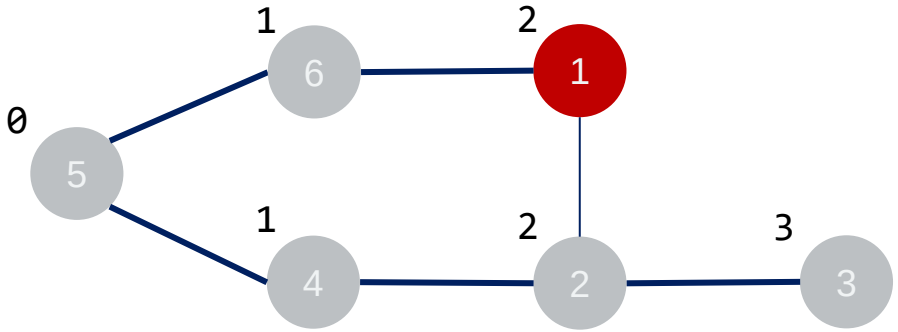


- Lấy đỉnh đầu trong queue: Đỉnh 2.

- Duyệt qua các đỉnh kề của 2: đỉnh 3 và đỉnh 1 (nhưng đỉnh 1 đã được thăm nên không xét).

  - dist[3] = dist[2] + 1
  - visited[3] = true
  - Cho đỉnh 3 vào queue

Ví dụ 2



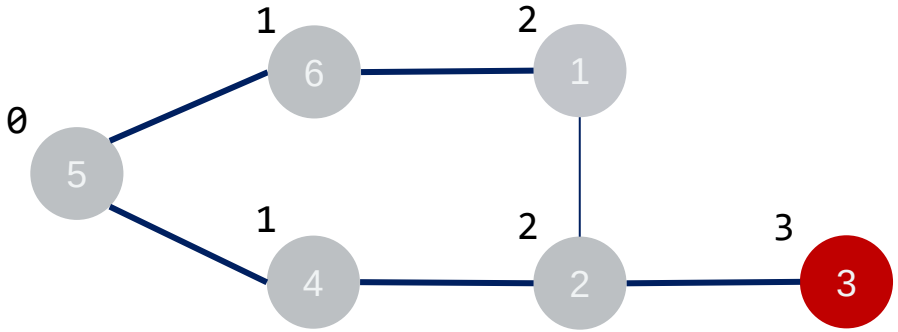
Queue:

3

- Lấy đỉnh đầu trong queue: Đỉnh 1.

- Duyệt qua các đỉnh kề của 1: đỉnh 6 và đỉnh 2 (nhưng cả 2 đỉnh đã được thăm nên không xét).

Ví dụ 2

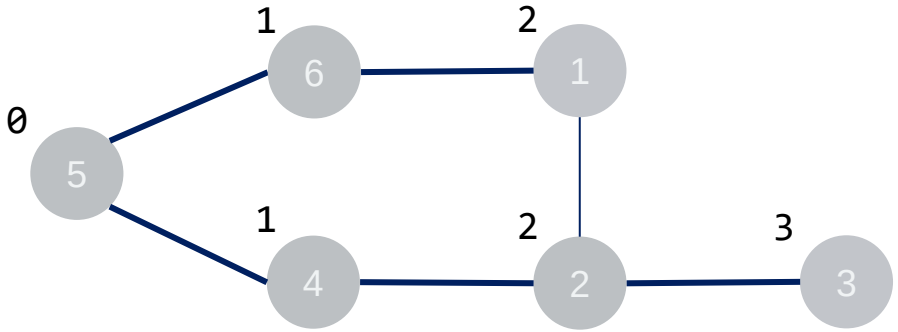


Queue:

- Lấy đỉnh đầu trong queue: Đỉnh 3.

- Duyệt qua các đỉnh kề của 3: đỉnh 2 (nhưng cả đỉnh 2 đã được thăm nên không xét).

Ví dụ 2



Queue:

- Queue rỗng Dừng thuật toán

Hướng dẫn giải

Các bước giải

- Bước 1: Đọc Q – số lượng testcase.
- Bước 2: Với mỗi testcase: **dist = [-1]*MAX, visited = [false]*MAX**
  - Bước 2.1: Đọc vào danh sách cạnh của đồ thị và chuyển thành danh sách đỉnh k ề.
  - Bước 2.2: Chạy thuật BFS bắt đầu từ đỉnh **S**:
    - Cho **dist[S] = 0**, **visited[S] = true**, và cho **S** vào queue.
    - Xét đến khi queue rỗng:
      - Lấy đỉnh đầu tiên trong queue (đỉnh **u**)
      - Duyệt qua các đỉnh kề của **u** (đỉnh **v**), nếu **v** chưa được thăm thì tiến hành cập nhật **dist[v], visited[v]**.
  - Bước 2.3: Xuất kết quả.

Mã giả

Mã giả

```
read Q
MAX = 1000 + 1
for q = 1 → Q:
   //reset các biến
   dist = [-1]*MAX
   visited = [False]*MAX
   graph = [[] for i in range(MAX)]
#graph[i] chứa danh sách đỉnh kề của đỉnh i
   read N, M
   for i = 1 → M:
      read u, v
      graph[u].append(v)
      graph[v].append(u)
   read S
   BFS(S, graph, visited, dist)
   for i = 1 → N:
      if i != S:
         if visited[i] == True:
            print(dist[i]*6)
         else: print(-1)
```

```
function BFS(source, graph, visited, dist):
   q = Queue
   q.push(source)
   dist[source] = 0
   visited[source] = true
   while q is not empty:
      u = q.front(), q.pop()
      for v in graph[u]:
         if visited[v] == False:
            dist[v] = dist[u] + 1
            visited[v] = True
            q.push(v)
```