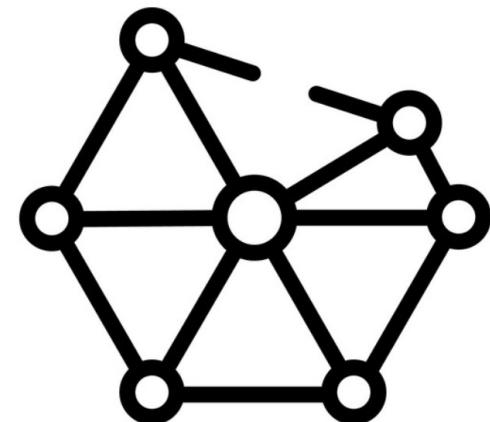




THỰC HÀNH LÝ THUYẾT ĐỒ THỊ

Trình bày:
KS. BÙI PHÚ KHUYÊN





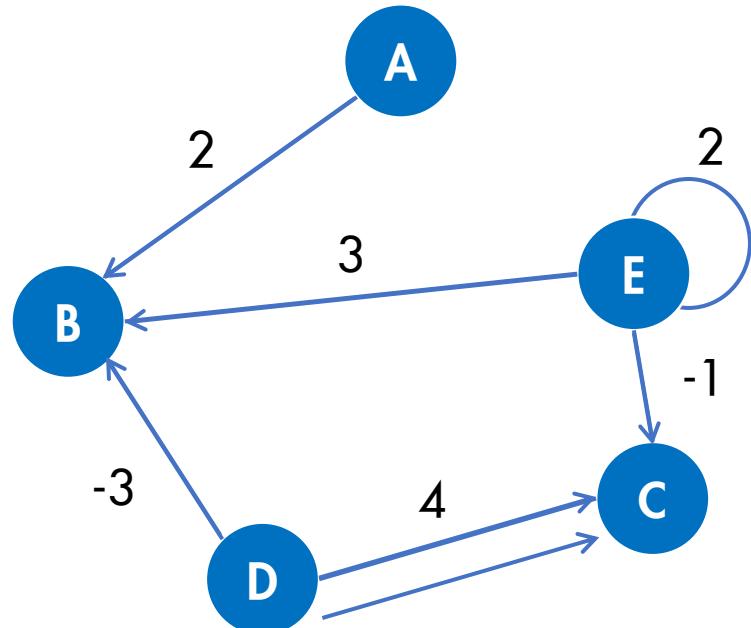
TUẦN 1

ĐỒ THỊ, MA TRẬN KÈ VÀ

XÉT TÍNH LIÊN THÔNG

ĐỒ THỊ LÀ GÌ?

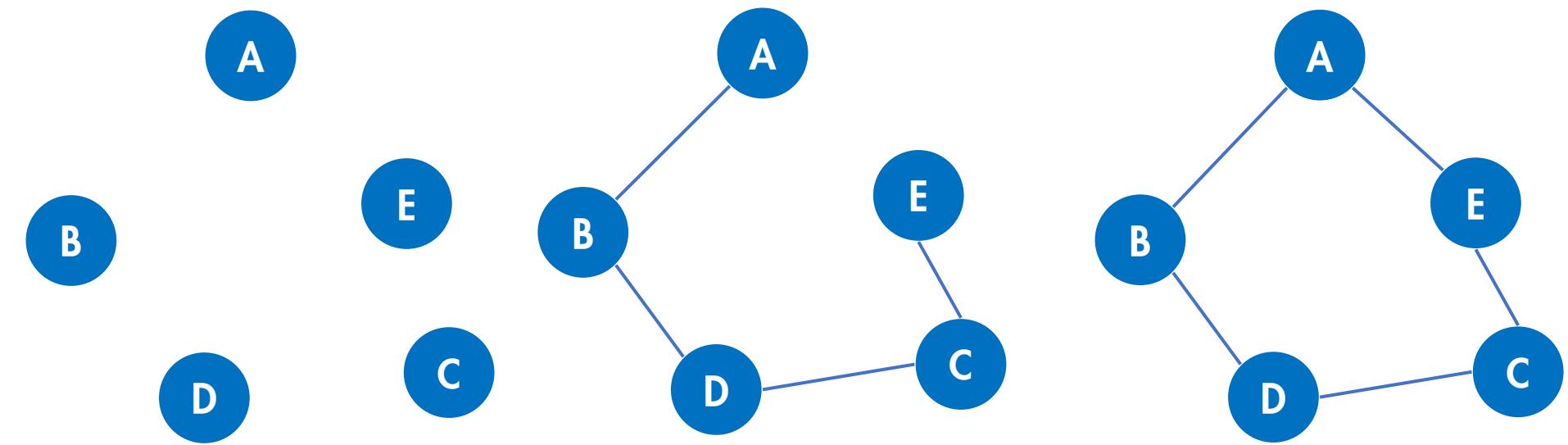
Đồ thị là 1 cấu trúc rời rạc gồm **các đỉnh** và **các cạnh** nối với các đỉnh đó



Về cạnh thì có các dạng sau:

- Không có hướng
- Có hướng
- Có vòng/khuyên
- Có cạnh song song
- Có trọng số

MỘT SỐ DẠNG ĐỒ THỊ



Đồ thị rỗng

Đồ thị đơn

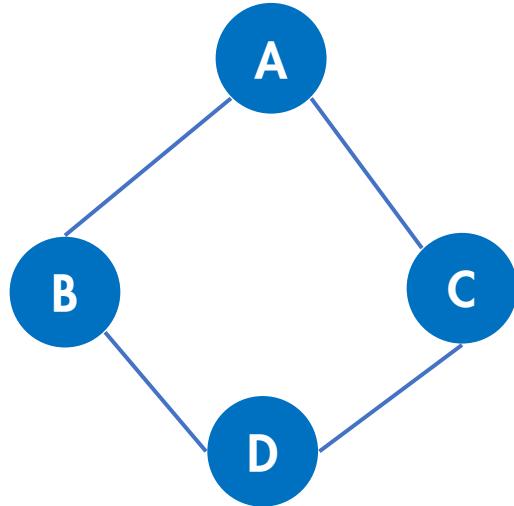
Đồ thị đầy đủ

MA TRẬN KỀ ĐỒ THỊ KHÔNG TRỌNG SỐ

Để biểu diễn một đồ thị (có hướng/vô hướng), người ta sử dụng dạng ma trận kề.

Một **giá trị $a[i, j]$** trong ma trận (dòng i , cột j) **bằng 1** nếu **có cạnh nối** từ đỉnh i đến đỉnh j trong đồ thị. Nếu giữa 2 đỉnh của đồ thị **không có cạnh nối** thì phần tử $a[i, j] = 0$

- Xác định nhanh thông qua số đỉnh.
- Thiết lập ma trận



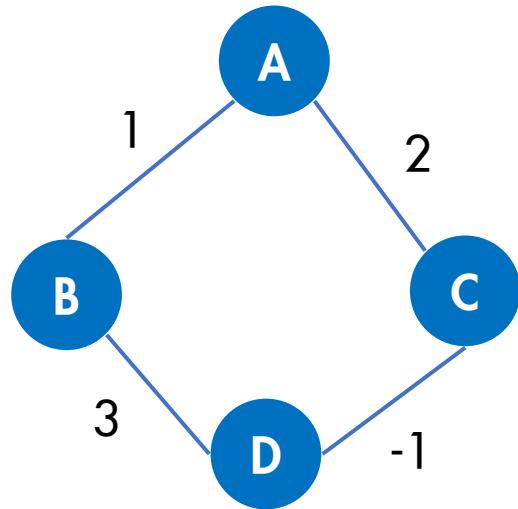
	A	B	C	D
A				
B				
C				
D				

	A	B	C	D
A	0	1	1	0
B	1	0	0	1
C	1	0	0	1
D	0	1	1	0

MA TRẬN KỀ CÓ TRỌNG SỐ

Một giá trị $a[i, j]$ trong ma trận (dòng i , cột j) ứng với **trọng số của cạnh** từ đỉnh i đến đỉnh j trong đồ thị. Nếu giữa 2 đỉnh của đồ thị **không có cạnh nối** thì phần tử $a[i, j] = 0$

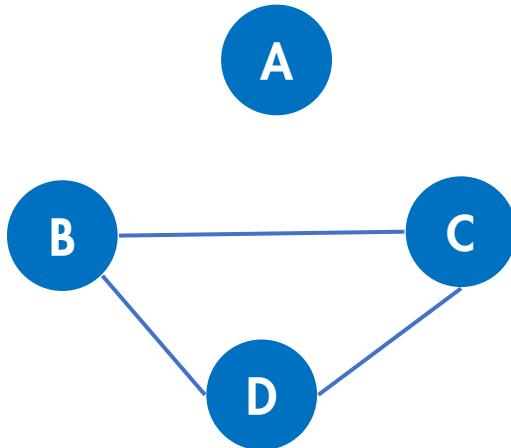
- Xác định nxn thông qua số đỉnh.
- Thiết lập ma trận



	A	B	C	D
A				
B				
C				
D				

	A	B	C	D
A	0	1	2	0
B	1	0	0	3
C	2	0	0	-1
D	0	3	-1	0

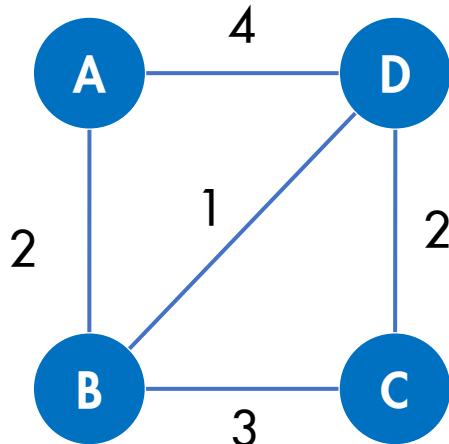
VÍ DỤ MỘT SỐ MA TRẬN KỀ



	A	B	C	D
A				
B				
C				
D				

	A	B	C	D
A	0	0	0	0
B	0	0	1	1
C	0	1	0	1
D	0	1	1	0

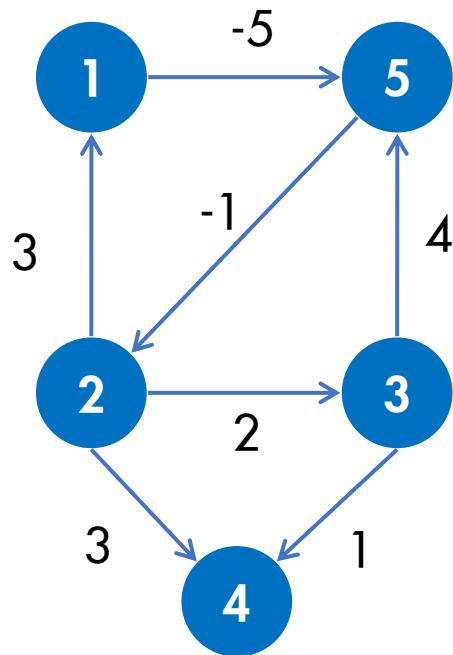
VÍ DỤ MỘT SỐ MA TRẬN KỀ



	A	B	C	D
A				
B				
C				
D				

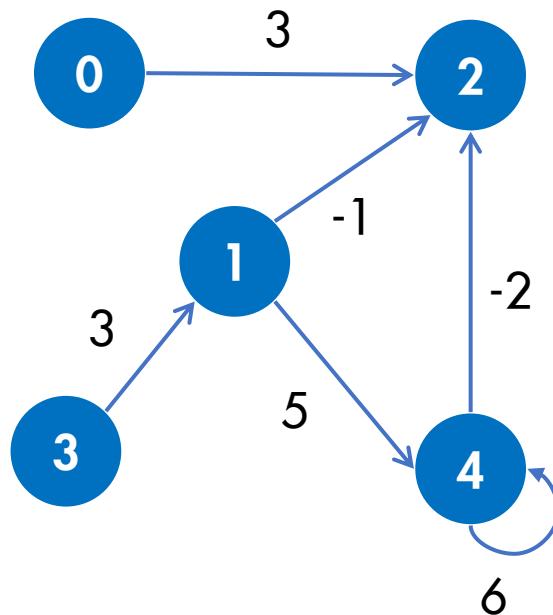
	A	B	C	D
A	0	2	0	4
B	2	0	3	1
C	0	3	0	2
D	4	1	2	0

VÍ DỤ MỘT SỐ MA TRẬN KỀ



	1	2	3	4	5
1	0	0	0	0	-5
2	3	0	2	3	0
3	0	0	0	1	4
4	0	0	0	0	0
5	0	-1	0	0	0

VÍ DỤ MỘT SỐ MA TRẬN KỀ



	0	1	2	3	4
0	0	0	3	0	0
1	0	0	-1	0	5
2	0	0	0	0	0
3	0	3	0	0	0
4	0	0	-2	0	6

NHẬP – XUẤT MA TRẬN KÈ

Số đỉnh



test1.txt

b

0	1	1	0	0	0	0
1	0	1	1	1	0	0
1	1	0	1	1	0	0
0	1	1	0	1	1	0
0	1	1	1	0	1	0
0	0	0	1	1	0	0
0	0	0	1	1	0	0

Ma trận kè

BÀI TẬP

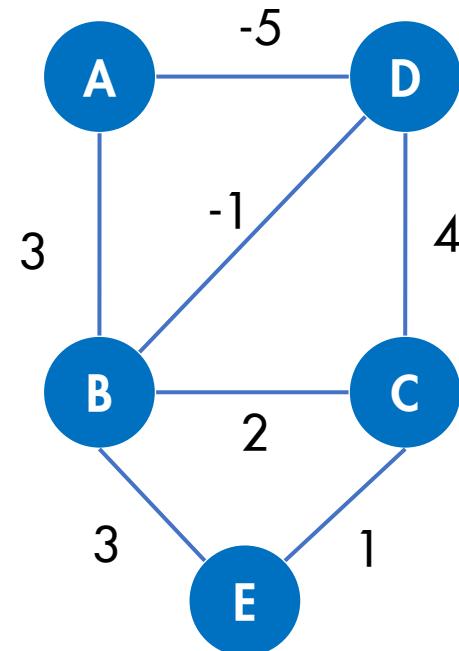
- Đọc thông tin của đồ thị gồm số đỉnh n, ma trận kè từ 1 file nào đó (chẳng hạn như C:/test1.txt có cấu trúc như trên).
- Xuất thông tin của đồ thị ra màn hình:
 - Dòng đầu tiên là số đỉnh của đồ thị
 - Những dòng tiếp theo (n dòng) là thông tin ma trận kè của đồ thị
- Kiểm tra đồ thị được đọc vào từ file đó có hợp lệ không?
- Nếu thông tin đồ thị đó là hợp lệ thì hãy cho biết đồ thị đó là vô hướng hay có hướng?

ĐỒ THỊ LIÊN THÔNG LÀ GÌ?

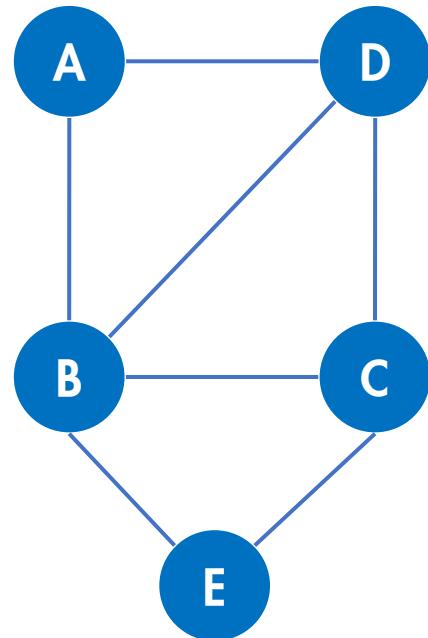
Đồ thị liên thông là đồ thị:

- Chỉ 1 thành phần liên thông
- Hay lấy bất kỳ 2 đỉnh i, j nào đều có đường đi trực tiếp hoặc gián tiếp (thông qua các đỉnh khác của đồ thị) nối từ đỉnh i đến đỉnh j.

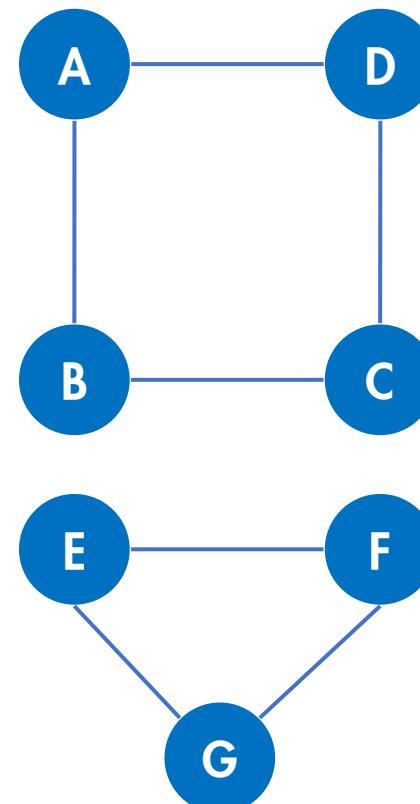
Đồ thị không liên thông là đồ thị tồn tại nhiều thành phần liên thông con.



ĐỒ THỊ KHÔNG LIÊN THÔNG

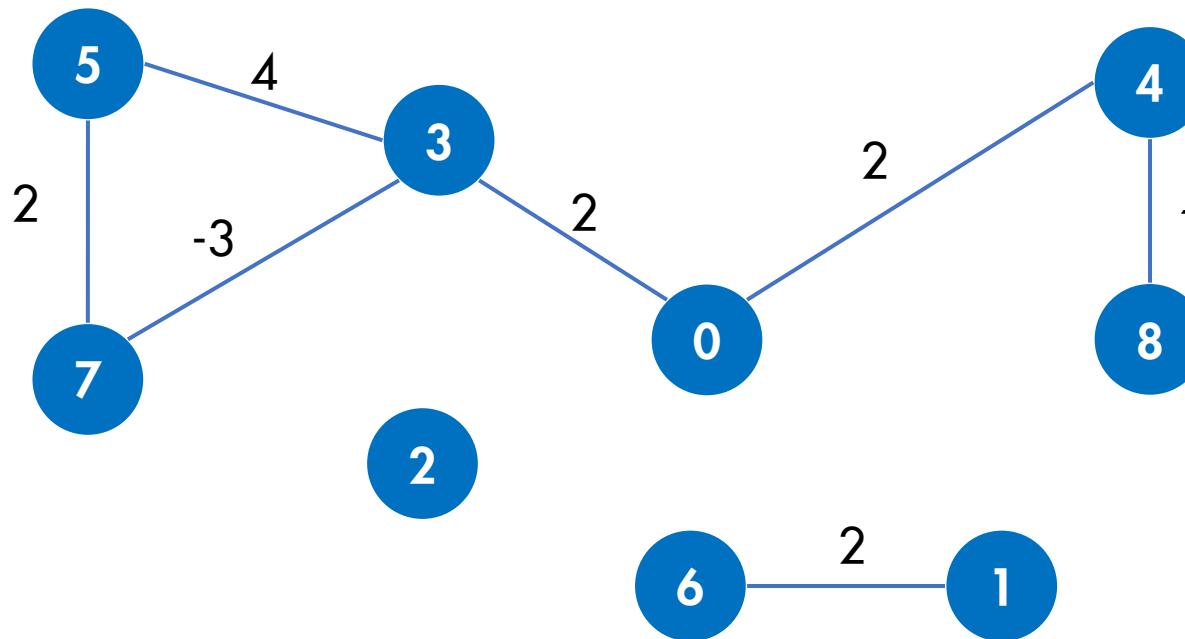


Đồ thị liên thông



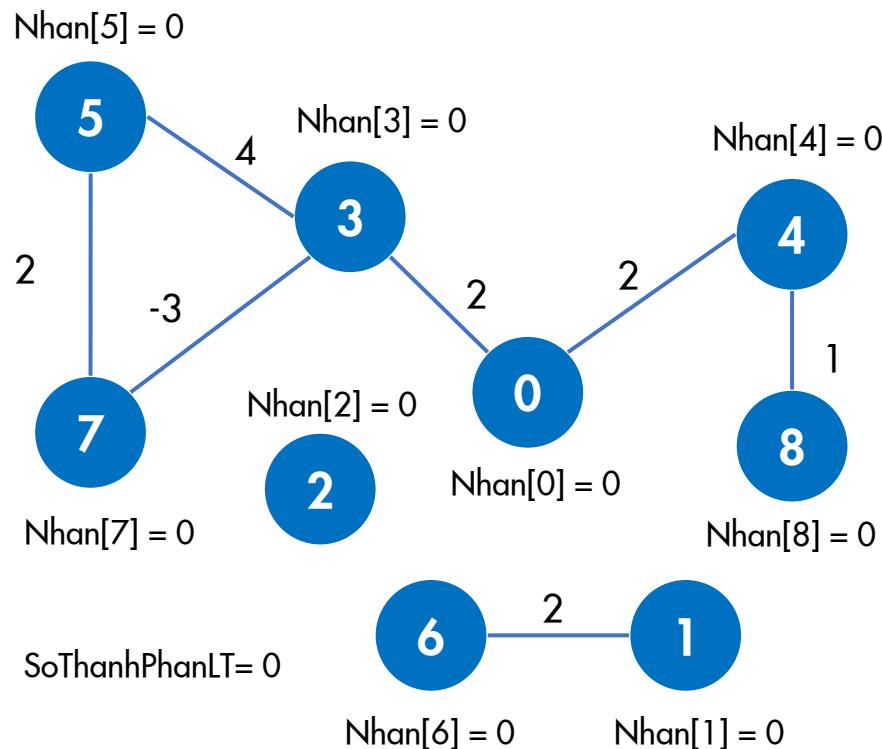
Đồ thị không
liên thông

THUẬT GIẢI ĐI TÌM CÁC THÀNH PHẦN LIÊN THÔNG



Tìm các thành phần liên thông của đồ thị trên?

THUẬT GIẢI ĐI TÌM CÁC THÀNH PHẦN LIÊN THÔNG

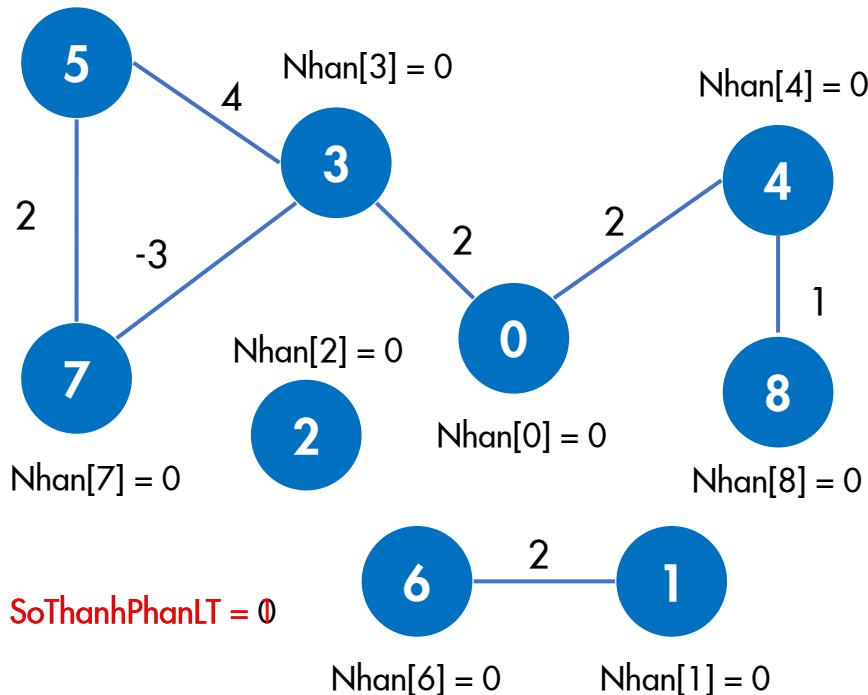


Bước 1: Khởi tạo:

- Tất cả các đỉnh của đồ thị có nhãn bằng 0. Tức $Nhan[i] = 0$. Điều này có nghĩa là đỉnh của đồ thị này chưa thuộc thành phần liên thông nào
- Biến $SoThanhPhanLienThong = 0$

THUẬT GIẢI ĐI TÌM CÁC THÀNH PHẦN LIÊN THÔNG

Nhan[5] = 0

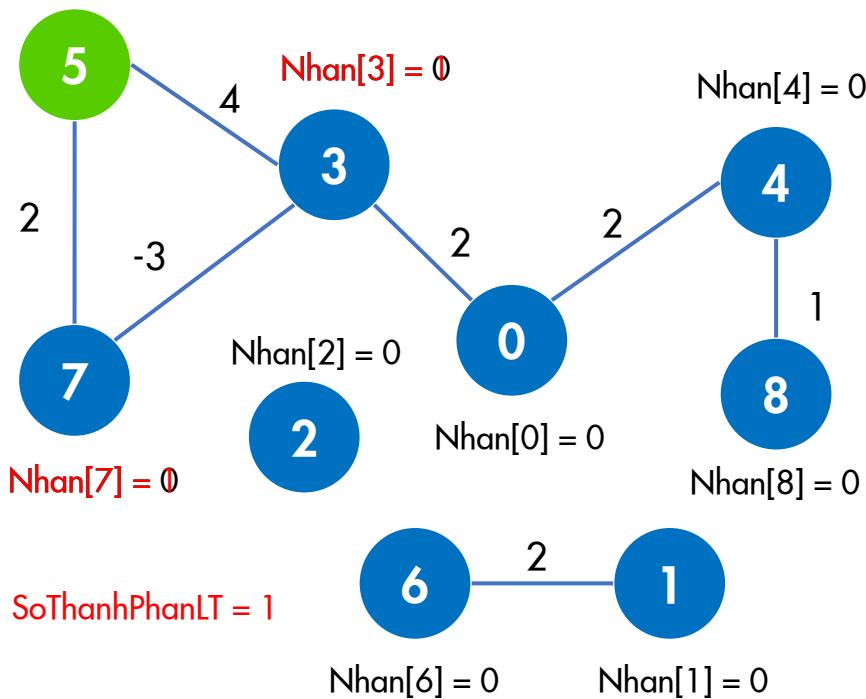


Bước 2:

- Chọn một đỉnh bất kỳ có nhãn của nó là 0. (giả sử chọn đỉnh 5)
- Tăng SoThanhPhanLT lên 1 đơn vị.
- Gán nhãn của đỉnh 5 = SoThanhPhanLT (tức Nhan[5] = 1)

THUẬT GIẢI ĐI TÌM CÁC THÀNH PHẦN LIÊN THÔNG

Nhan[5] = 1

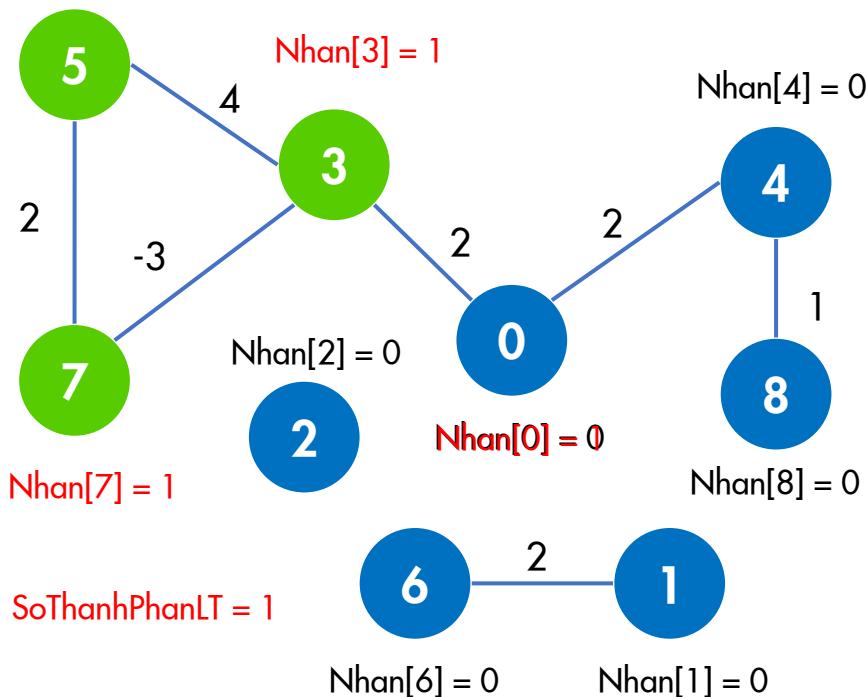


Bước 3:

- Tiến hành đi tìm các đỉnh khác nối với đỉnh 5. Đó chính là đỉnh 3 và đỉnh 7
- Ta thấy đỉnh 3 và 7 có nhän là 0. (khác với nhän của đỉnh 5 là 1)
- Tiến hành cập nhật nhän của đỉnh 3 và 7 bằng nhän của đỉnh 5.

THUẬT GIẢI ĐI TÌM CÁC THÀNH PHẦN LIÊN THÔNG

Nhan[5] = 1

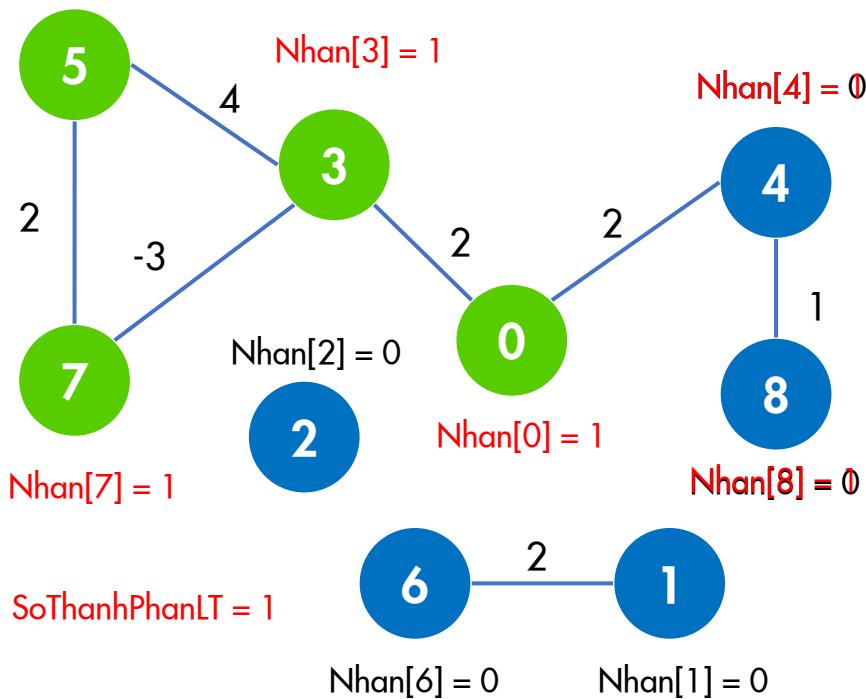


Bước 4:

- Ta tiếp tục xét đỉnh 3, đỉnh 3 có 3 cạnh nối
 $3 \rightarrow 5; 3 \rightarrow 7; 3 \rightarrow 0.$
- Ta thấy nhãn của đỉnh 3, 5, 7 là 1. Do đó, ta không cập nhật giá trị nhãn của đỉnh 5, 7.
- Trong khi đó có đỉnh 0 có nhãn khác đỉnh 3. Ta tiến hành cập nhật nhãn của đỉnh 0:
 $Nhan[0] = Nhan[3]$

THUẬT GIẢI ĐI TÌM CÁC THÀNH PHẦN LIÊN THÔNG

Nhan[5] = 1

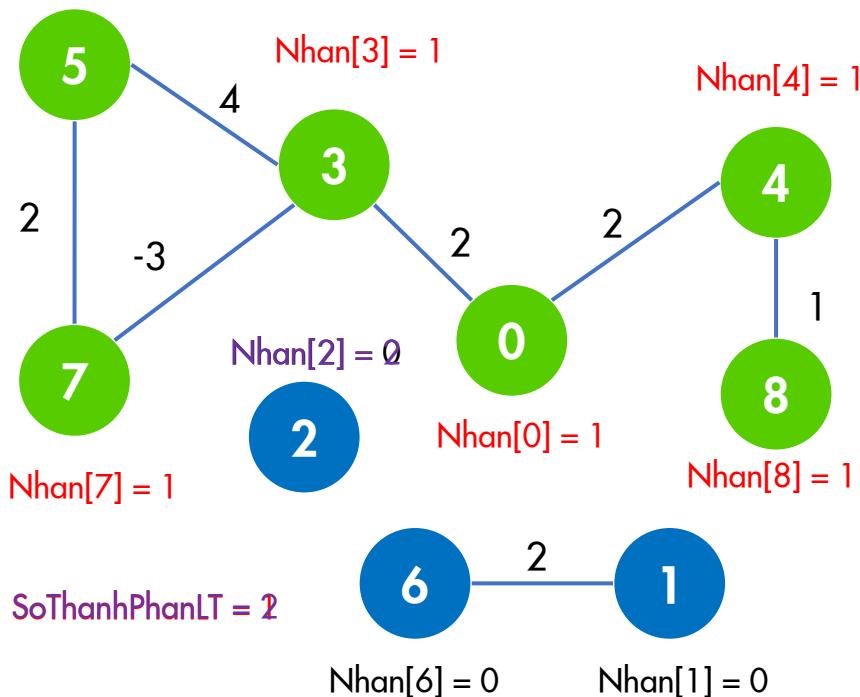


Bước 5:

- Ta tiếp tục thay đổi nhãn tương tự.
- Tại thời điểm hiện tại không tìm được đỉnh mới (nối với 8) để thay đổi nhãn tiếp. Do đó ta kết thúc thành phần liên thông đầu tiên (1) ở đây. Bao gồm các đỉnh:
5, 7, 3, 0, 4, 8
- Nhận thấy trong đồ thị vẫn còn đỉnh có nhãn là 0 (tức là chưa thuộc một thành phần liên thông nào). Do đó, ta quay lại bước đầu

THUẬT GIẢI ĐI TÌM CÁC THÀNH PHẦN LIÊN THÔNG

Nhan[5] = 1

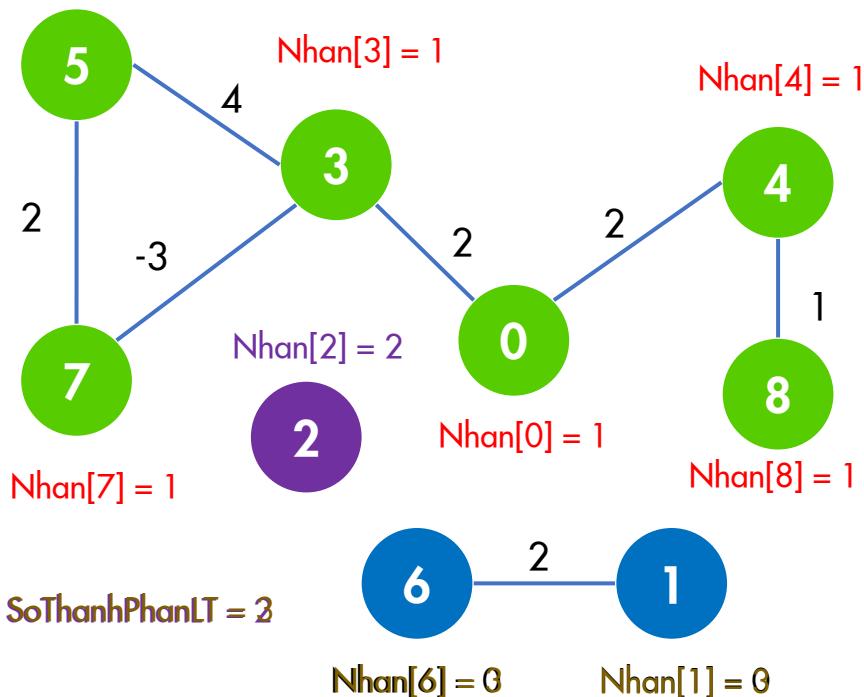


Bước 6:

- Chọn một đỉnh bất kỳ có nhãn của nó là 0. (giả sử chọn đỉnh 2)
- Tăng SoThanhPhanLT lên 1 đơn vị.
- Gán nhãn của đỉnh 2 = SoThanhPhanLT (tức Nhan[2] = 2)
- Tại thời điểm hiện tại không tìm được đỉnh mới (nối với 2) để thay đổi nhãn tiếp. Do đó ta kết thúc thành phần liên thông (2) ở đây. Chỉ gồm đỉnh 2

THUẬT GIẢI ĐI TÌM CÁC THÀNH PHẦN LIÊN THÔNG

Nhan[5] = 1

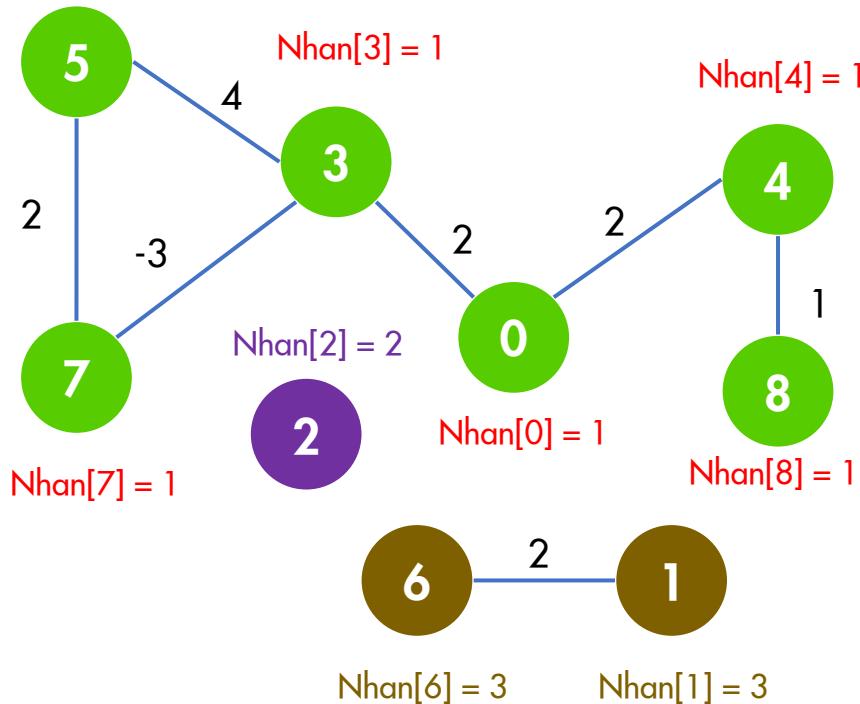


Bước 7:

- Nhận thấy trong đồ thị vẫn còn đỉnh có nhãn là 0 (6, 7). Do đó, ta quay lại bước đầu.
- Chọn một đỉnh bất kỳ có nhãn của nó là 0. (giả sử chọn đỉnh 6)
- Tăng SoThanhPhanLT lên 1 đơn vị.
- Gán nhãn của đỉnh 6 = SoThanhPhanLT (tức Nhan[6] = 3)
- Tiến hành đi tìm các đỉnh khác nối với đỉnh 6. Đó chính là đỉnh 1. Cập nhật lại nhãn cho đỉnh 1

THUẬT GIẢI ĐI TÌM CÁC THÀNH PHẦN LIÊN THÔNG

$Nhan[5] = 1$



Kết luận: Có 3 Thành phần liên thông

- TPLT1: 0, 3, 4, 5, 7, 8
- TPLT2: 2
- TPLT3: 1, 6



TUẦN 2

CHU TRÌNH, ĐƯỜNG ĐI

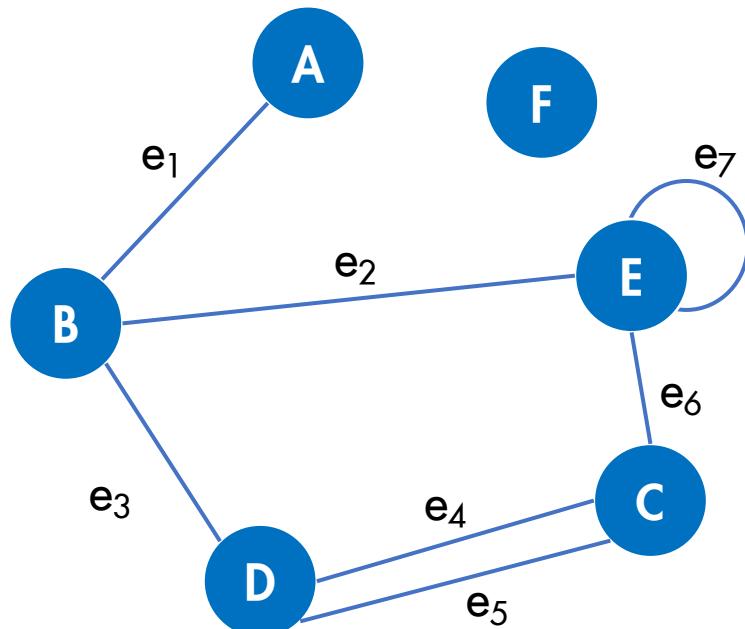
EULER

BẬC CỦA ĐỈNH TRÊN ĐỒ THỊ VÔ HƯỚNG

Bậc của đỉnh x trong đồ thị G là số các cạnh kề với đỉnh x.

Mỗi vòng (khuyên) được tính 2 lần.

Ký hiệu: $d_G(x)$ hoặc $d(x)$



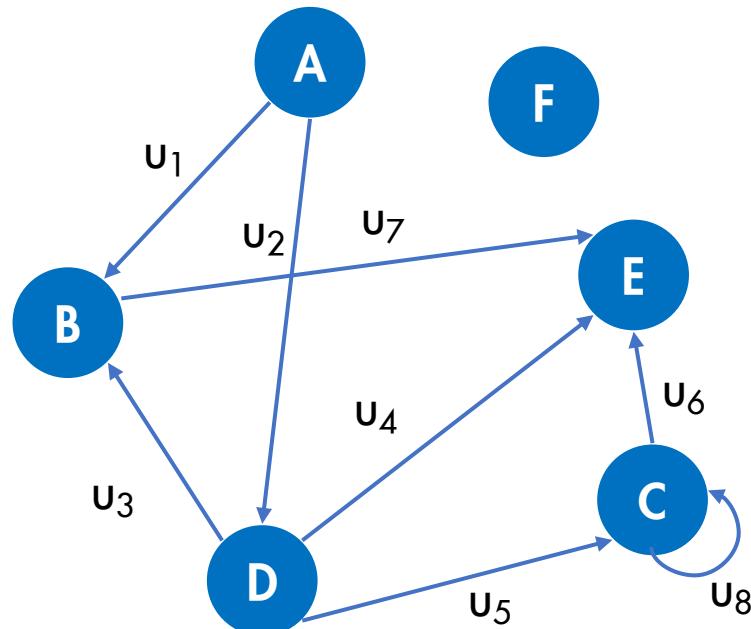
Tính bậc của tất cả các đỉnh của đồ thị bên:

- $d(A) = ?$ 1
- $d(B) = ?$ 3
- $d(C) = ?$ 3
- $d(D) = ?$ 3
- $d(E) = ?$ 4
- $d(F) = ?$ 0

BẬC CỦA ĐỈNH TRÊN ĐỒ THỊ CÓ HƯỚNG

- Nửa bậc ngoài của đỉnh x là số các cạnh **đi ra** khỏi đỉnh x , ký hiệu $d^+(x)$
- Nửa bậc trong của đỉnh x là số các cạnh **đi vào** đỉnh x , ký hiệu $d^-(x)$
- Bậc của đỉnh x : $d(x) = d^+(x) + d^-(x)$

Tính bậc của tất cả các đỉnh của đồ thị bên:

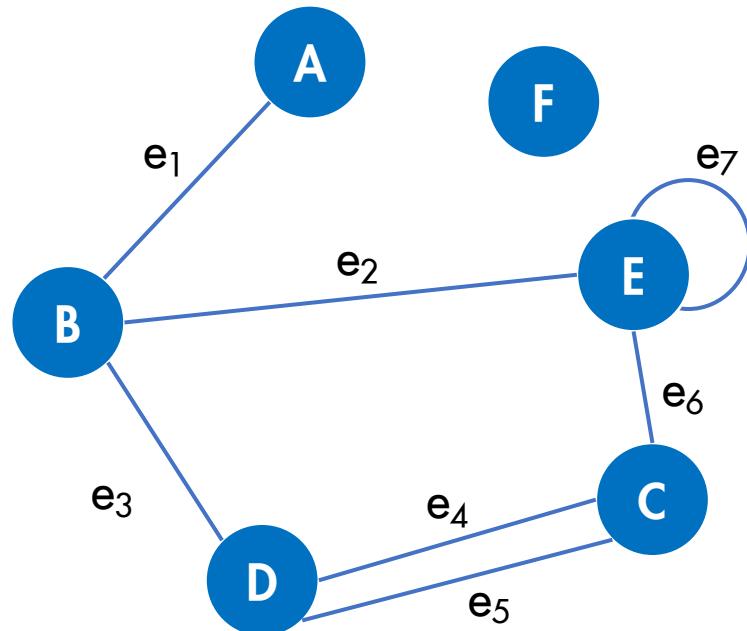


	$d^+(x)$	$d^-(x)$	$d(x)$
• $d(A) = ?$	2	0	2
• $d(B) = ?$	1	2	3
• $d(C) = ?$	2	2	4
• $d(D) = ?$	3	1	4
• $d(E) = ?$	0	3	3
• $d(F) = ?$	0	0	0

ĐỈNH TREO, ĐỈNH CÔ LẬP

Từ cách tính bậc của đồ thị, người ta định nghĩa:

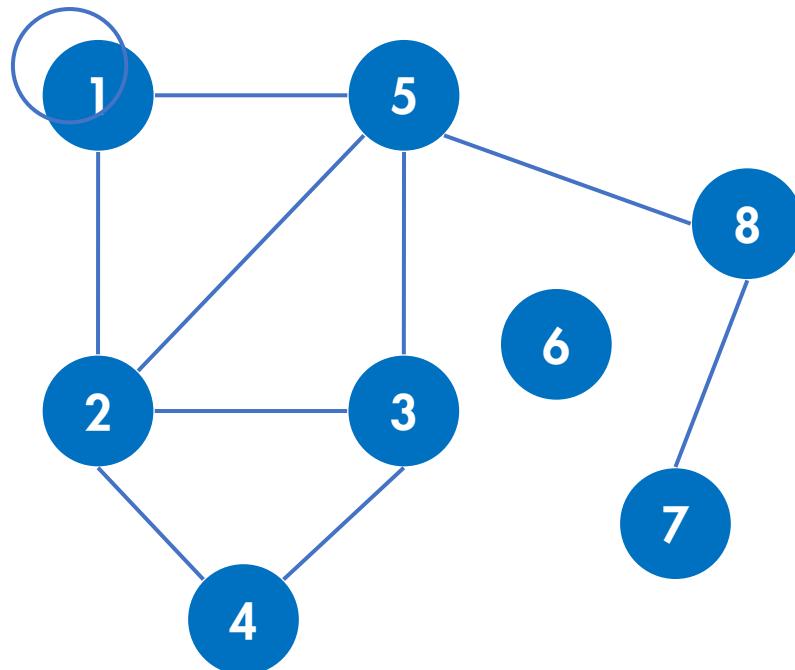
- Đỉnh treo: là đỉnh có bậc bằng 1
- Đỉnh cô lập: là đỉnh có bậc bằng 0



- $d(A) = 1 \longrightarrow$ **Đỉnh treo**
- $d(B) = 3$
- $d(C) = 3$
- $d(D) = 3$
- $d(E) = 4$
- $d(F) = 0 \longrightarrow$ **Đỉnh cô lập**

BÀI TẬP VẬN DỤNG

Tính bậc của tất cả các đỉnh của đồ thị sau. Và cho biết đỉnh treo và đỉnh cô lập?



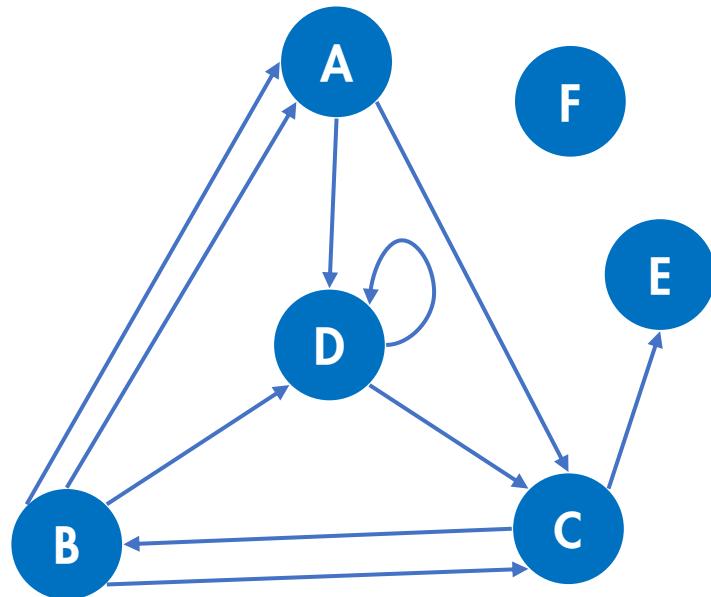
Đỉnh $G(x)$	Bậc $dG(x)$
1	4
2	4
3	3
4	2
5	4
6	0
7	1
8	2

Đỉnh cô lập

Đỉnh treo

BÀI TẬP VẬN DỤNG

Tính bậc của tất cả các đỉnh của đồ thị sau. Và cho biết đỉnh treo và đỉnh cô lập?

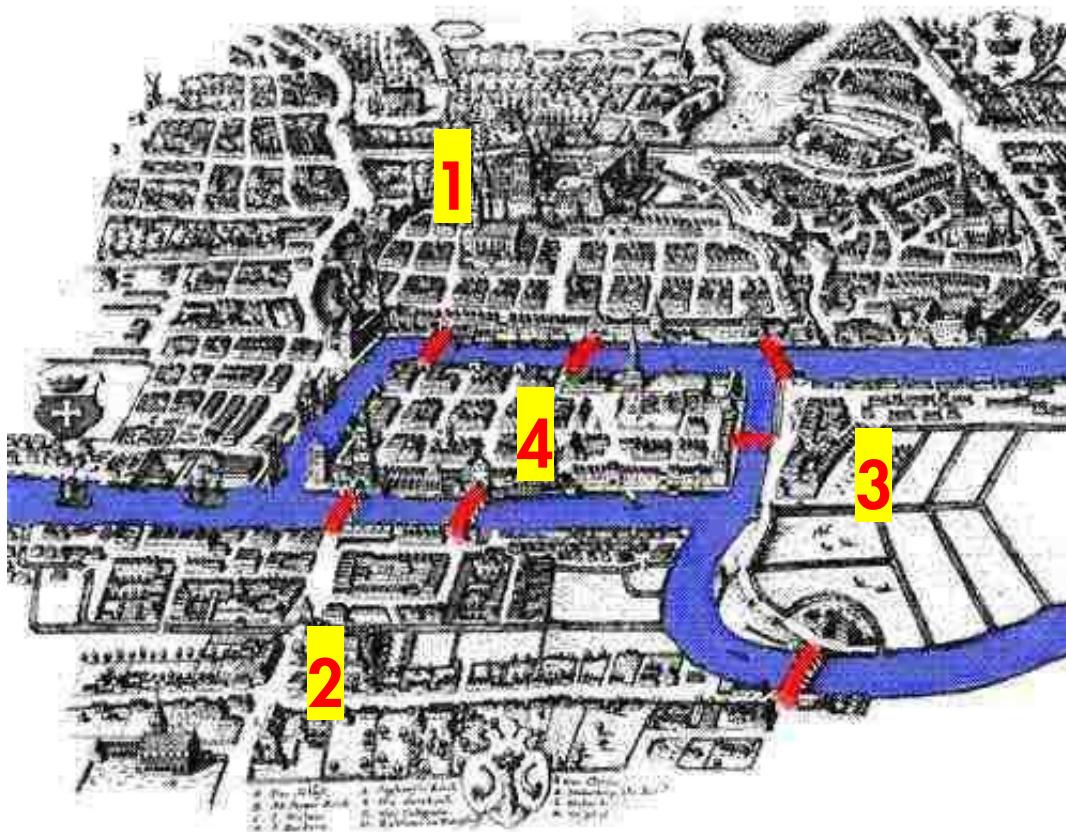


Đỉnh $G(x)$	Bậc ngoài	Bậc trong	Bậc $dG(x)$
A	2	2	4
B	4	1	5
C	2	3	5
D	2	3	5
E	0	1	1
F	0	0	0

Đỉnh treo

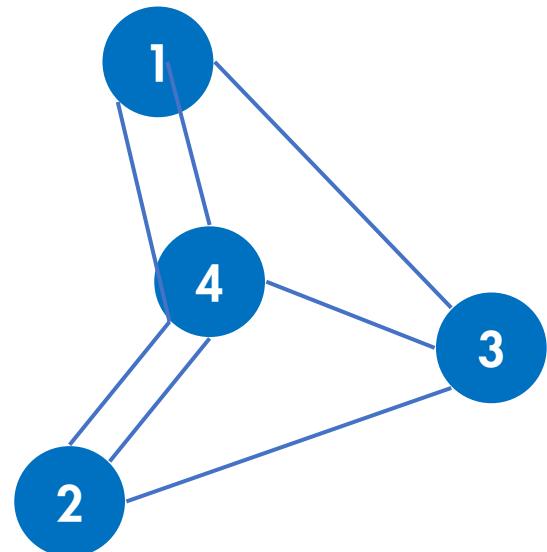
Đỉnh cô lập

BÀI TOÁN CỔ ĐIỂN THÀNH PHỐ KONIGSBERG



7 cây cầu (màu đỏ) ở thành phố Konigsberg.

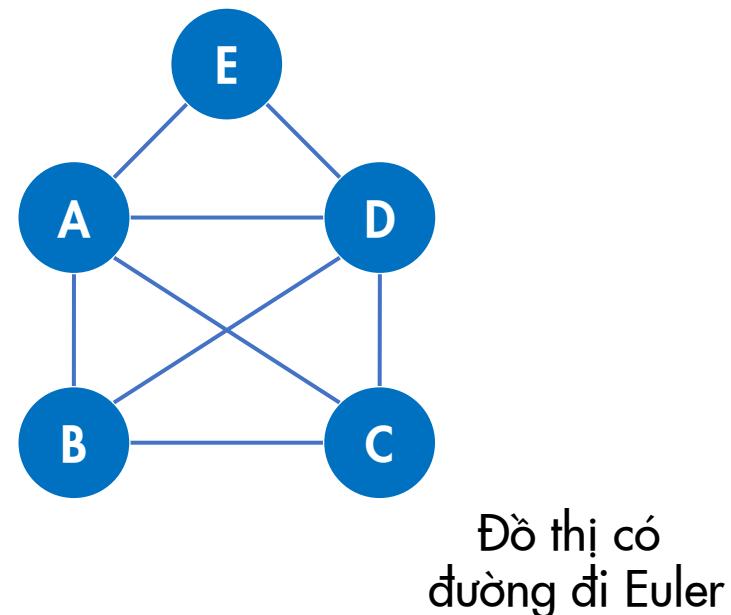
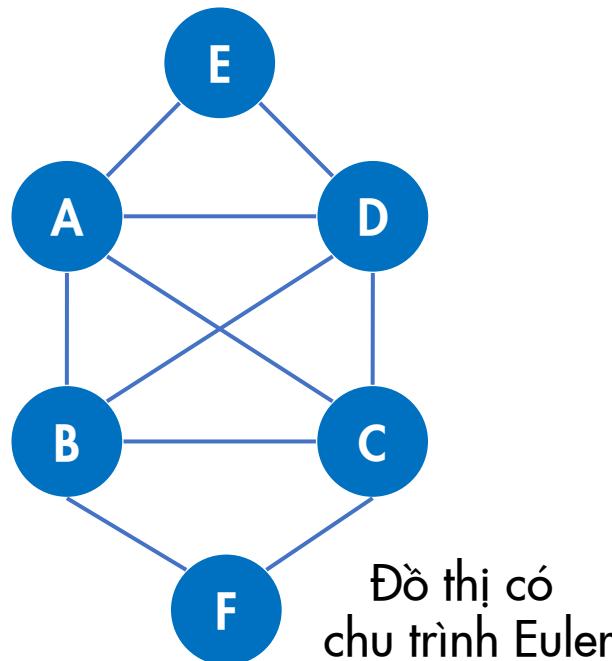
Bài toán: Có thể nào đi dạo qua tất cả bảy cầu, mỗi cầu chỉ một lần thôi không?

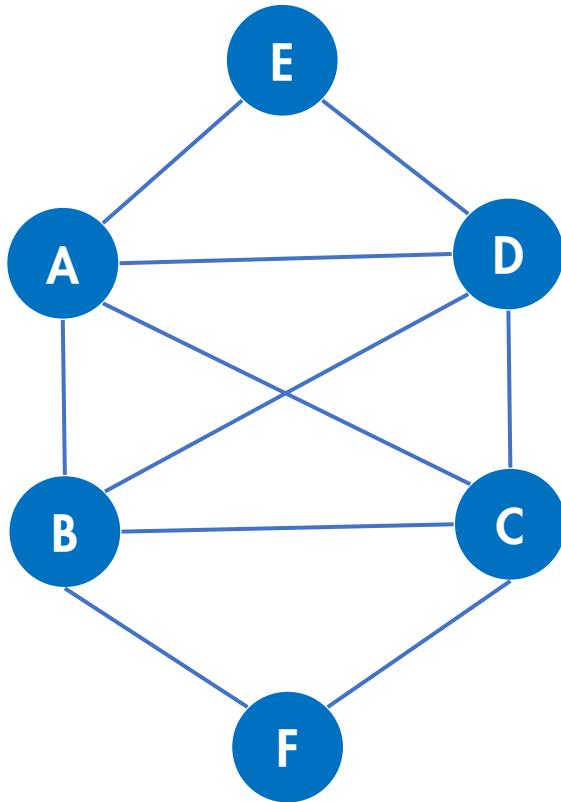


Euler → Không giải được

ĐỊNH NGHĨA

- Chu trình Euler là chu trình đi qua tất cả các đỉnh của đồ thị, mỗi cạnh đúng 1 lần trong đó đỉnh đầu và đỉnh cuối trùng nhau (đồ thị có thể có các đỉnh cô lập).
- Tương tự, đường đi Euler, ngoài trừ điểm đầu và điểm cuối không trùng nhau.





Đồ thị có
chu trình Euler

CHU TRÌNH EULER

Điều kiện cần để tồn tại chu trình Euler:

- Trong đồ thị vô hướng, bậc của tất cả các đỉnh phải là số chẵn.
- Trong đồ thị có hướng, bậc ngoài và bậc trong của mỗi đỉnh phải bằng nhau.

Trường hợp chắc chắn tồn tại chu trình Euler:

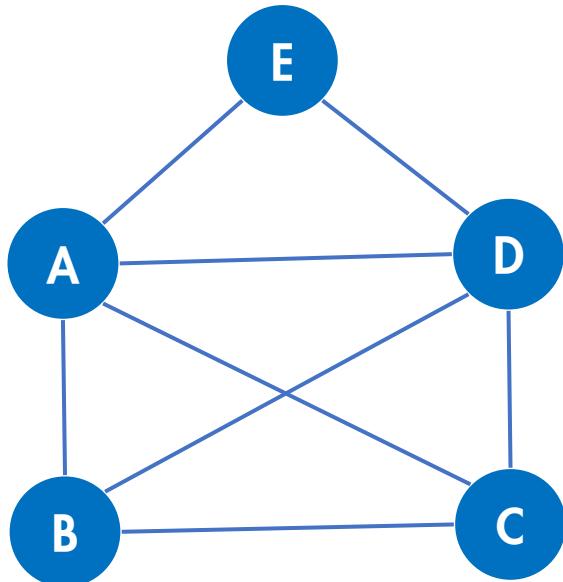
- Đồ thị vô hướng
- Liên thông
- Bậc của tất cả các đỉnh trong đồ thị đều là bậc chẵn (không tồn tại đỉnh bậc lẻ nào)

Trong một đồ thị, nếu không tìm ra chu trình Euler, vẫn có thể tồn tại đường đi Euler

ĐƯỜNG ĐI EULER

Điều kiện cần để tồn tại đường đi Euler:

- Trong đồ thị vô hướng, tồn tại duy nhất 2 đỉnh có bậc lẻ, tất cả các đỉnh còn lại là bậc chẵn
- Trong đồ thị có hướng, bậc ngoài và bậc trong của mỗi đỉnh phải bằng nhau ngoại trừ một đỉnh tại đó bậc ngoài lớn hơn bậc trong 1 đơn vị (làm đỉnh bắt đầu) và một đỉnh tại đó bậc trong lớn hơn bậc ngoài 1 đơn vị (làm đỉnh kết thúc)

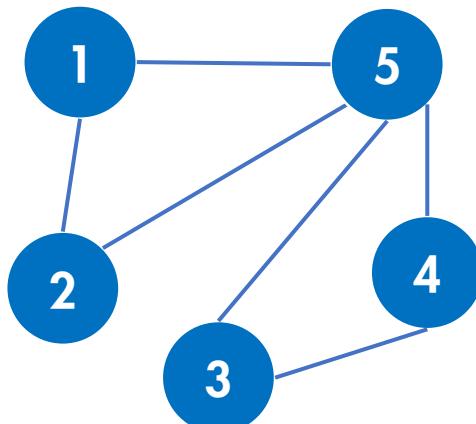


Đồ thị có
đường đi Euler

Trường hợp chắc chắn tồn tại đường đi Euler:

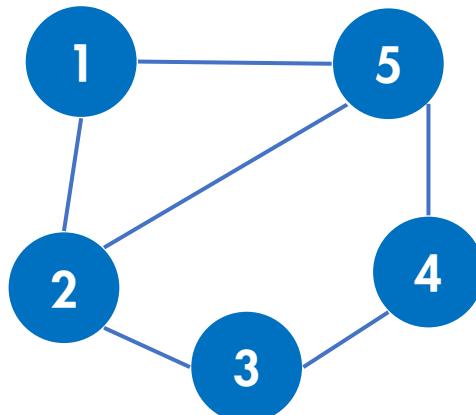
- Đồ thị vô hướng
- Liên thông
- Tồn tại đúng 2 đỉnh bậc lẻ của đồ thị.

TÌM CÁC ĐỒ THỊ CÓ CHU TRÌNH – ĐƯỜNG ĐI EULER

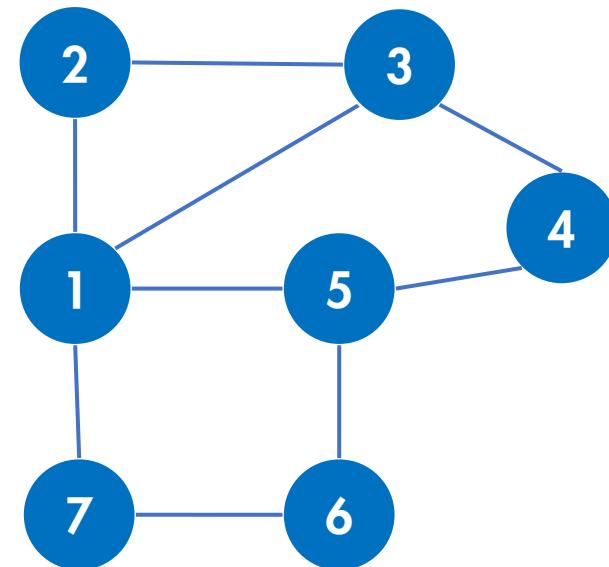


Chu trình Euler
 $1 \rightarrow 2 \rightarrow 5 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 1$

Đường đi Euler
 $2 \rightarrow 1 \rightarrow 5 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$



Đường đi Euler
 $3 \rightarrow 2 \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 1 \rightarrow 7 \rightarrow 6 \rightarrow 5$

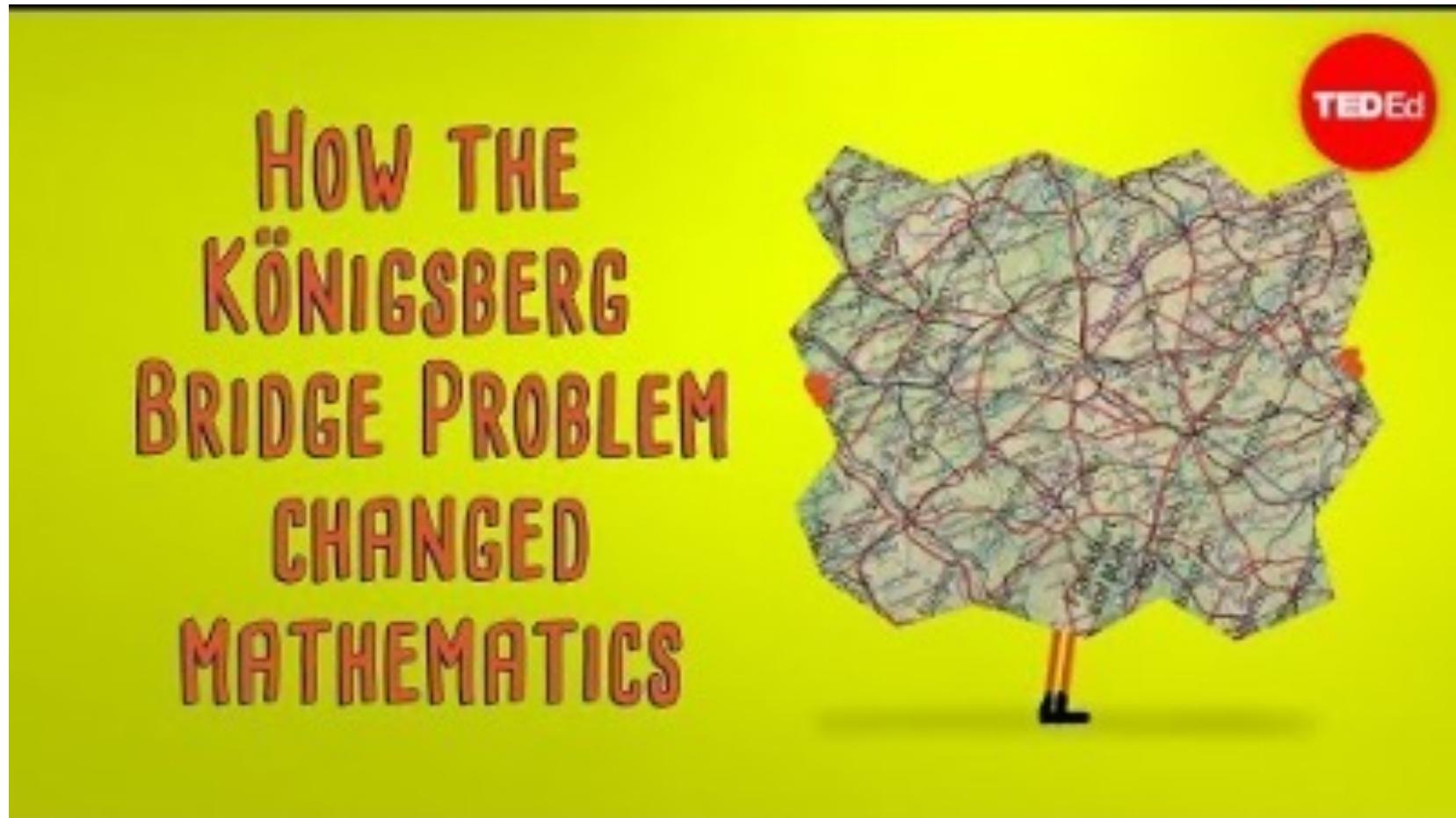




TÌM CÁC ĐỒ THỊ CÓ CHU TRÌNH – ĐƯỜNG ĐI EULER

Tiêu chí	Chu trình Euler	Đường Đi Euler
Đồ thị vô hướng	Có	Có
Đồ thị liên thông	Có	Có
Bậc của đỉnh	Ko có đỉnh bậc lẽ, tức toàn bộ đỉnh là bậc chẵn.	Có đúng 2 đỉnh bậc lẽ.
Điểm xuất phát	Bất kỳ đỉnh nào bậc chẵn của đồ thị	Là một đỉnh bậc lẽ của đồ thị.
Điểm kết thúc	Đỉnh bậc chẵn của đồ thị	Đỉnh bậc lẽ còn lại của đồ thị.

BÀI TOÁN CÂY CẦU Ở KONIGSBERG





THUẬT TOÁN FLEURY

Ta có thể vạch được 1 chu trình Euler trong đồ thị liên thông (G) có bậc của mọi đỉnh là chẵn theo thuật toán Fleury sau:

Xuất phát từ 1 đỉnh bất kỳ của đồ thị (G) và tuân theo 2 quy tắc sau:

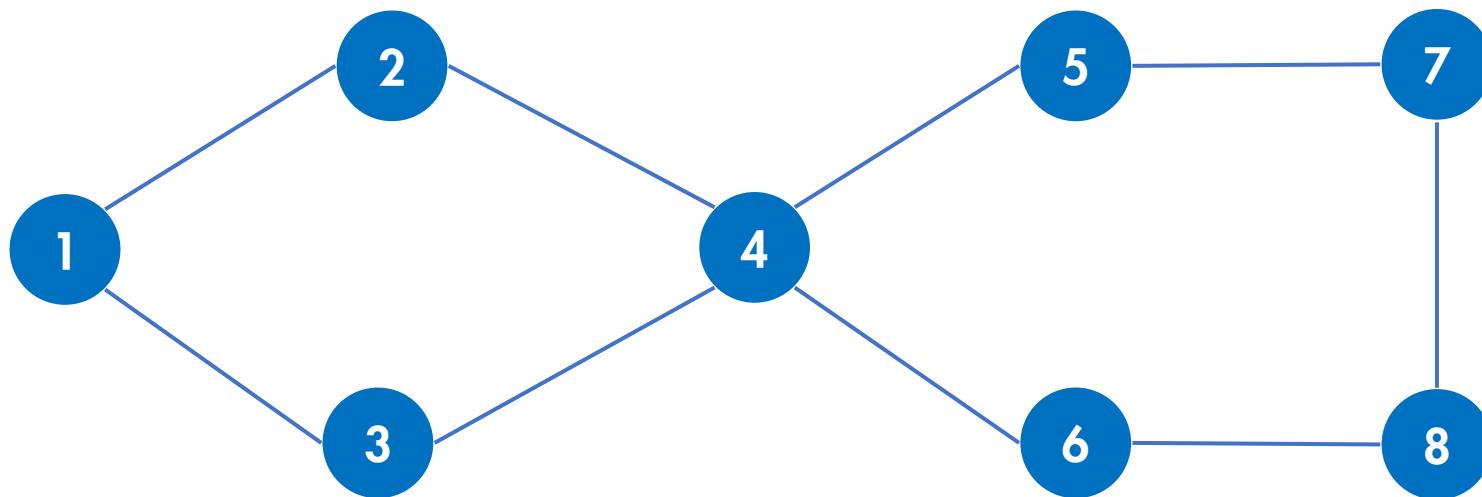
1. Mỗi khi đi qua một cạnh nào đó thì xóa nó đi, sau đó xóa đỉnh cô lập (nếu có).
2. Không bao giờ đi qua cạnh cầu trừ khi không còn cách đi nào khác.

(cạnh cầu là cạnh khi xoá đi không làm cho đồ thị mất liên thông)

Và ta cứ chọn cạnh đi một cách thoải mái như vậy cho đến khi không đi tiếp được nữa

Nhận xét: Thuật toán này tuy đơn giản nhưng thường không được sử dụng bởi việc xác định “cầu” trong đồ thị không phải đơn giản.

VÍ DỤ THUẬT TOÁN FLEURY



Nếu xuất phát từ đỉnh 1, có hai cách đi tiếp: hoặc sang 2 hoặc sang 3.

Giả sử ta sẽ sang 2 và xoá cạnh (1, 2) vừa đi qua.



THUẬT TOÁN TÌM CHU TRÌNH/ĐƯỜNG ĐI EULER

Cho $G = (N, E)$ là một đồ thị gồm N đỉnh và tập E cạnh.

Thuật toán tìm chu trình Euler xuất phát từ u với u là đỉnh có bậc khác 0 như sau:
'tour' là một stack;

Find_tour (u)

Với mỗi cạnh $e = (u,v)$ trong E

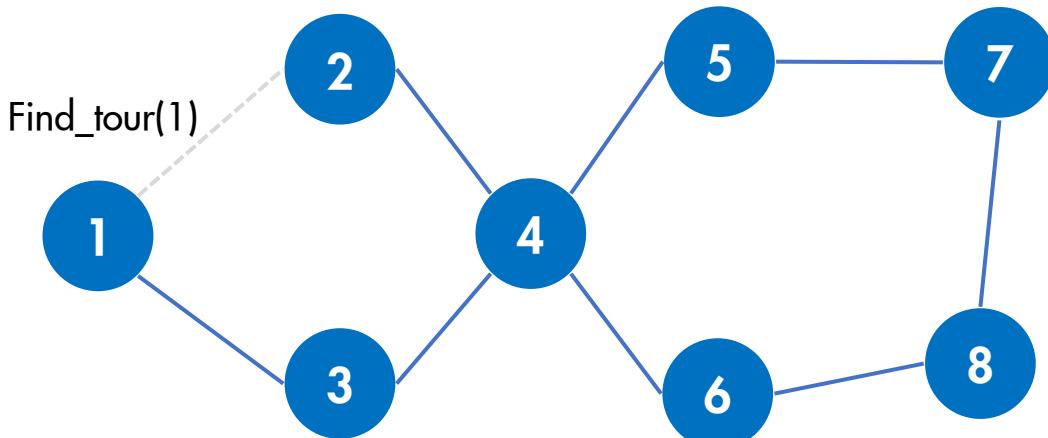
 Loại cạnh e khỏi tập E ;

Find_tour (v);

 Khi hết cạnh nối, thêm u vào stack 'tour';

Cuối hàm

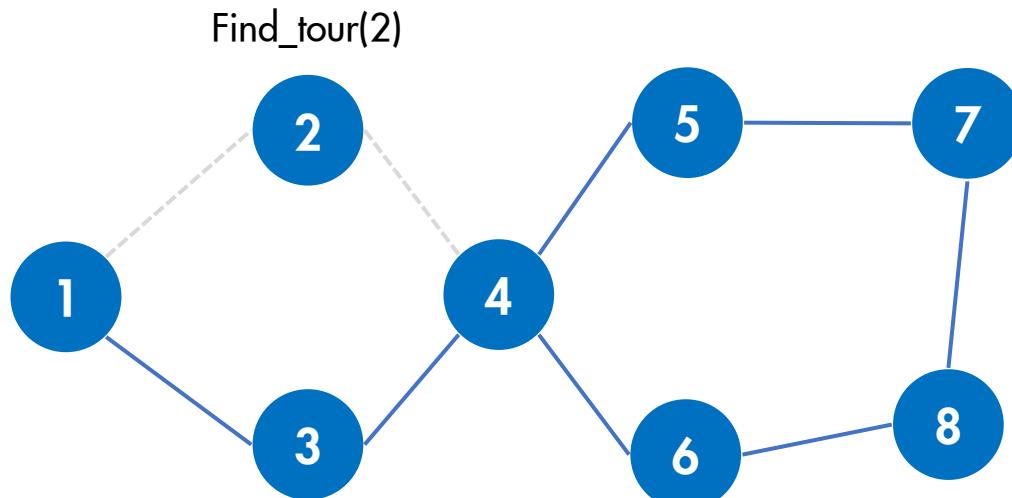
THUẬT TOÁN TÌM CHU TRÌNH/ĐƯỜNG ĐI EULER



Xuất phát từ đỉnh 1, giả sử chọn cạnh (1,2) để xét.
Loại cạnh (1,2) ra khỏi tập hợp



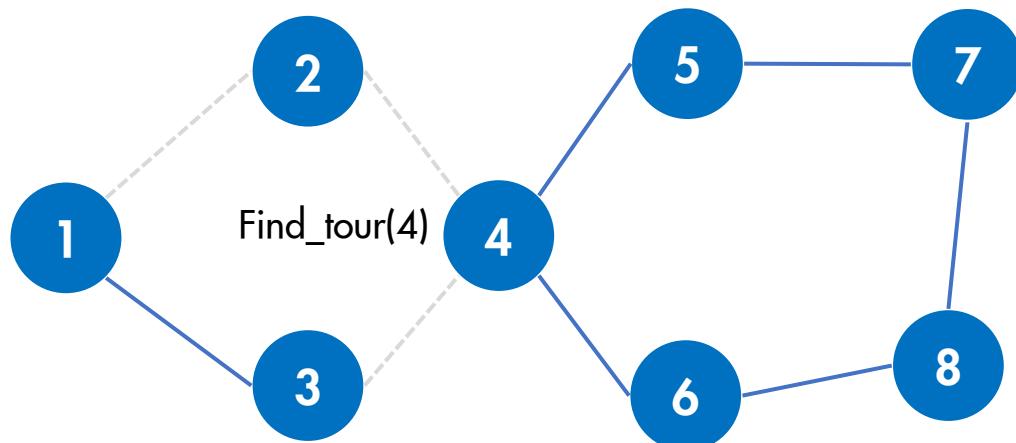
THUẬT TOÁN TÌM CHU TRÌNH/ĐƯỜNG ĐI EULER



Từ đỉnh 2, ta tìm thấy một cạnh (2, 4) và tương tự
ta cũng loại khỏi tập hợp



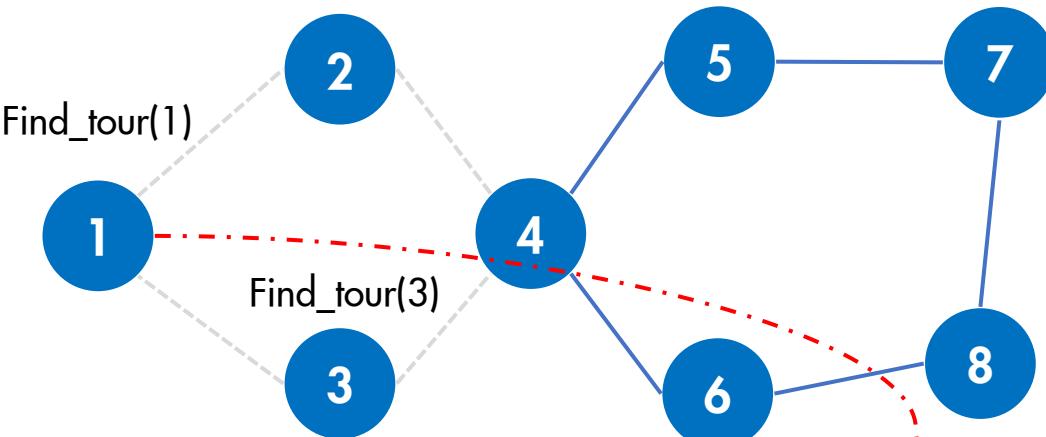
THUẬT TOÁN TÌM CHU TRÌNH/ĐƯỜNG ĐI EULER



Tại đỉnh 4 có 4 cạnh nối $(4,3)$, $(4,5)$, $(4,6)$.
Giả sử xét cạnh $(4,3)$ trước bằng cách xoá cạnh



THUẬT TOÁN TÌM CHU TRÌNH/ĐƯỜNG ĐI EULER

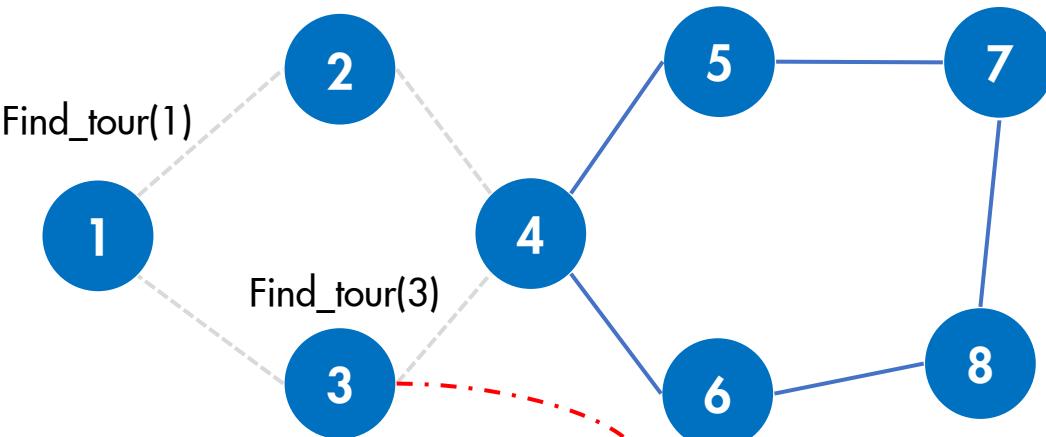


Tiếp đến xét đỉnh 3 và ta loại cạnh $(3,1)$.

Lúc này từ 1 không còn cạnh nào chưa xét,
vì vậy ta đưa đỉnh 1 vào stack



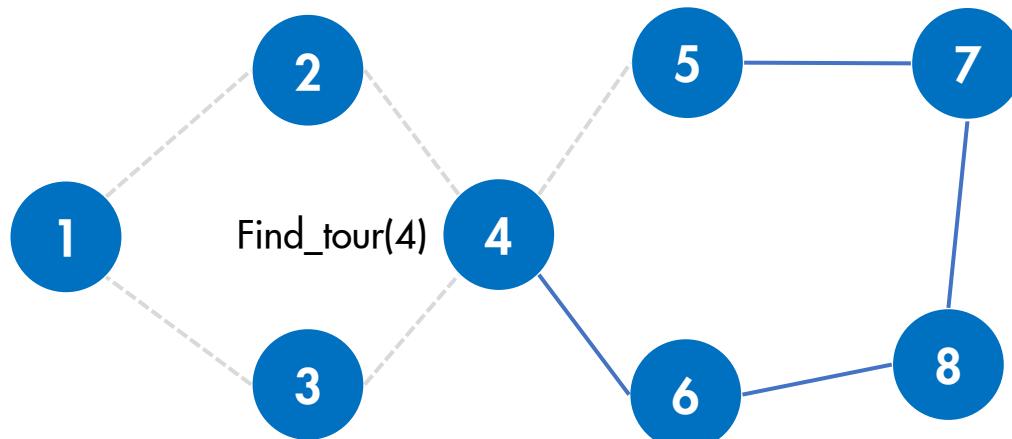
THUẬT TOÁN TÌM CHU TRÌNH/ĐƯỜNG ĐI EULER



Từ 3 cũng không còn cạnh nào chưa xét.
Đưa đỉnh 3 vào stack 'tour'



THUẬT TOÁN TÌM CHU TRÌNH/ĐƯỜNG ĐI EULER

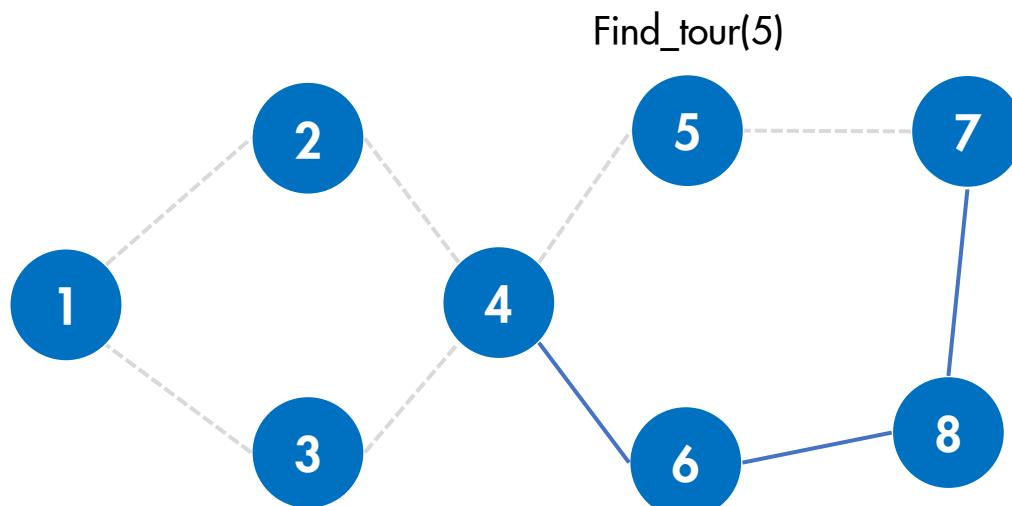


Quay lui lại Find_tour(4). Xét tiếp cạnh (4,5)

Tương tự ta sẽ xoá
lần lượt các cạnh (5,7), (7,8), (8,6) và (6,4)



THUẬT TOÁN TÌM CHU TRÌNH/ĐƯỜNG ĐI EULER

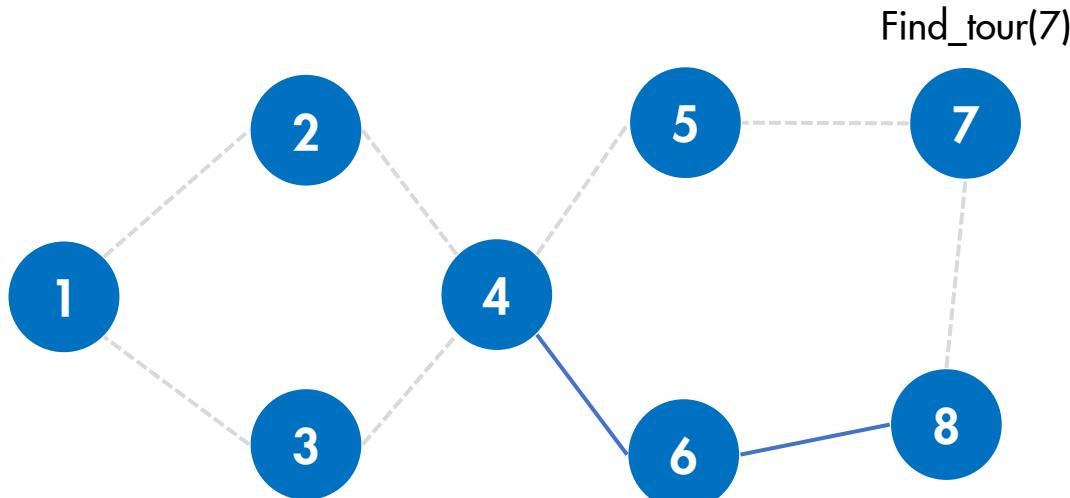


Quay lui lại Find_tour(4). Xét tiếp cạnh (4,5)

Tương tự ta sẽ xoá
lần lượt các cạnh (5,7), (7,8), (8,6) và (6,4)



THUẬT TOÁN TÌM CHU TRÌNH/ĐƯỜNG ĐI EULER

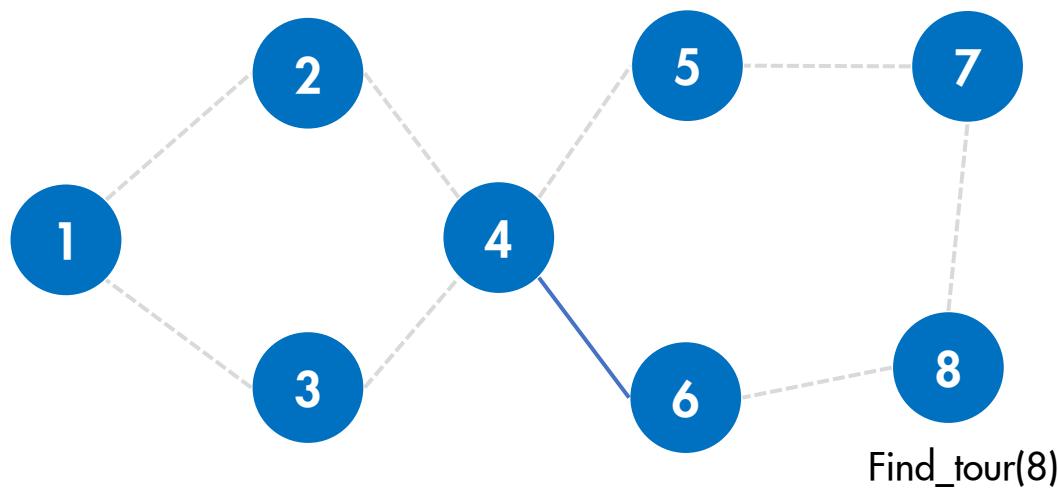


Quay lui lại Find_tour(4). Xét tiếp cạnh (4,5)

Tương tự ta sẽ xoá
lần lượt các cạnh (5,7), (7,8), (8,6) và (6,4)

3
1
'tour' stack

THUẬT TOÁN TÌM CHU TRÌNH/ĐƯỜNG ĐI EULER

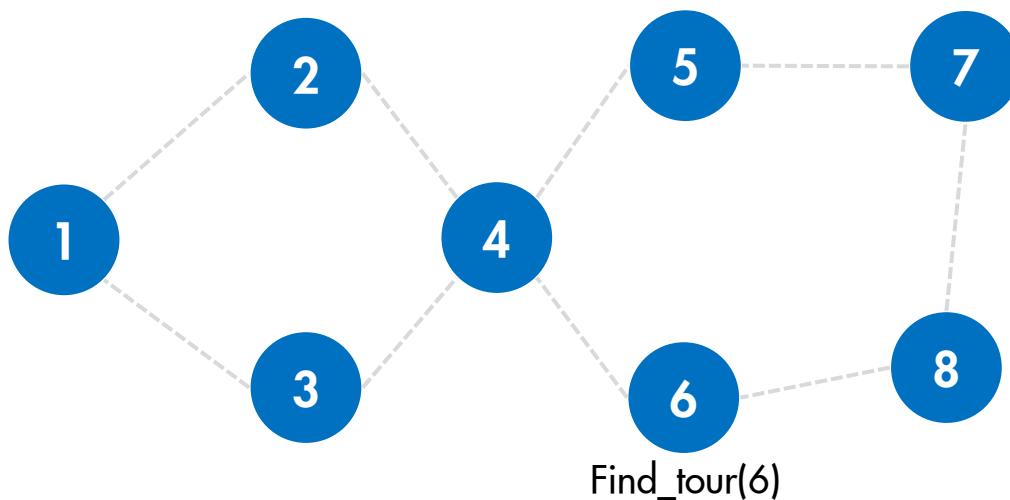


Quay lui lại Find_tour(4). Xét tiếp cạnh (4,5)

Tương tự ta sẽ xoá
lần lượt các cạnh (5,7), (7,8), (8,6) và (6,4)



THUẬT TOÁN TÌM CHU TRÌNH/ĐƯỜNG ĐI EULER

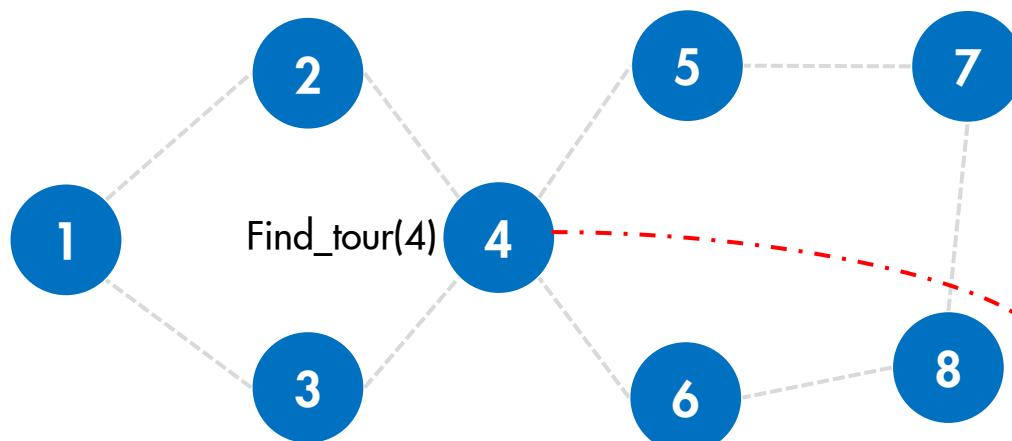


Tương tự ta sẽ xoá
lần lượt các cạnh $(5,7)$, $(7,8)$, $(8,6)$ và $(6,4)$

Tại đỉnh 4, không tìm thấy cạnh nối nữa.
Cho vào stack 'tour' và quay lui.

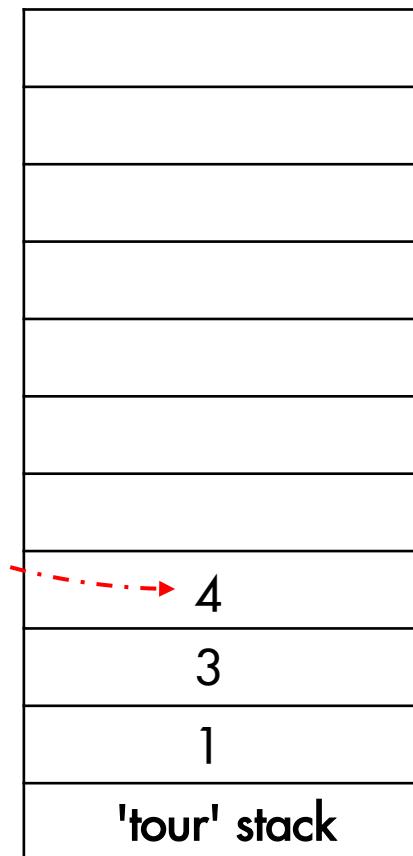


THUẬT TOÁN TÌM CHU TRÌNH/ĐƯỜNG ĐI EULER

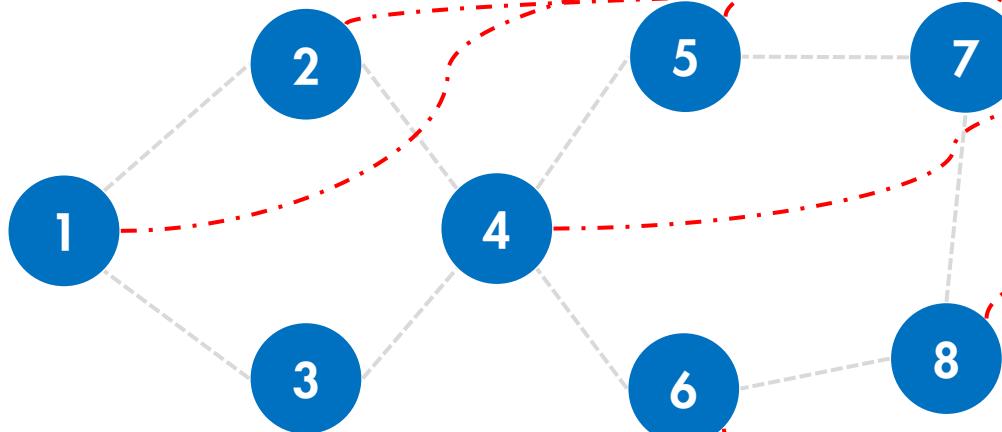


Tương tự ta sẽ xoá
lần lượt các cạnh $(5,7)$, $(7,8)$, $(8,6)$ và $(6,4)$

Tại đỉnh 4, không tìm thấy cạnh nối nữa.
Cho vào stack 'tour' và quay lui.

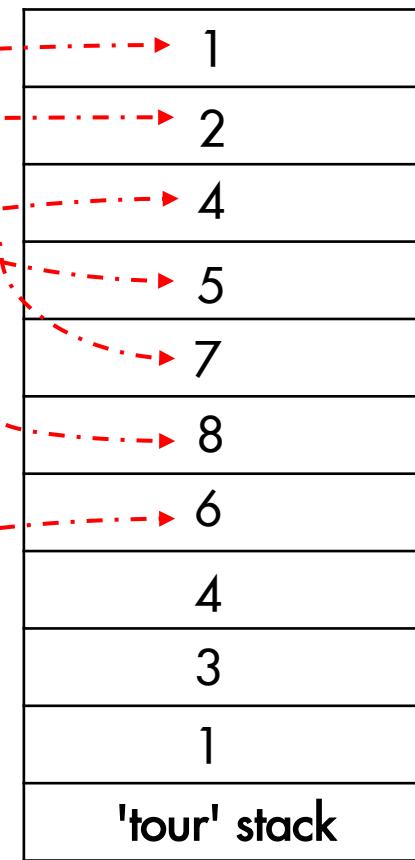


THUẬT TOÁN TÌM CHU TRÌNH/ĐƯỜNG ĐI EULER



Lần lượt quay lui và cho vào stack 'tour'

Khi đó lấy theo nguyên tắc của stack là LIFO
(Last In - First Out) sẽ được chu trình Euler



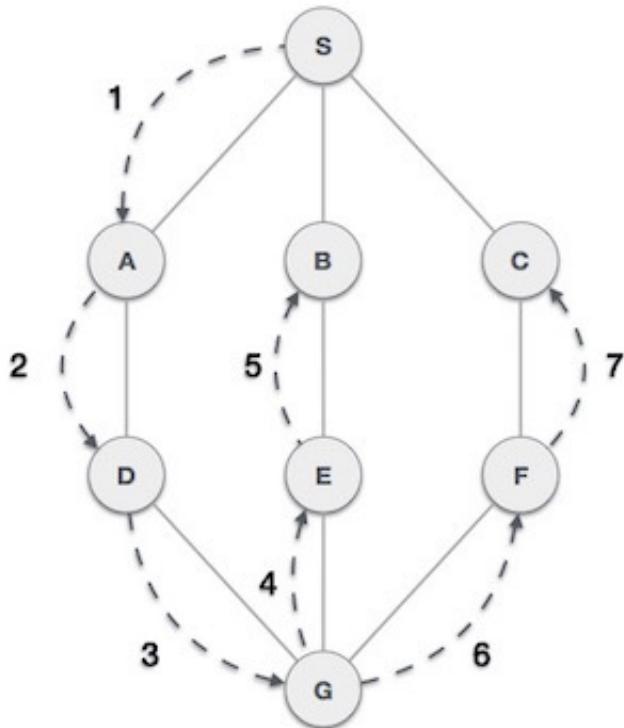


TUẦN 3

KỸ THUẬT TÌM KIẾM CHIỀU RỘNG VÀ CHIỀU SÂU TRÊN ĐỒ THỊ

TÌM KIẾM CHIỀU SÂU - DEPTH-FIRST SEARCH (DFS)

Ý TƯỞNG THUẬT TOÁN

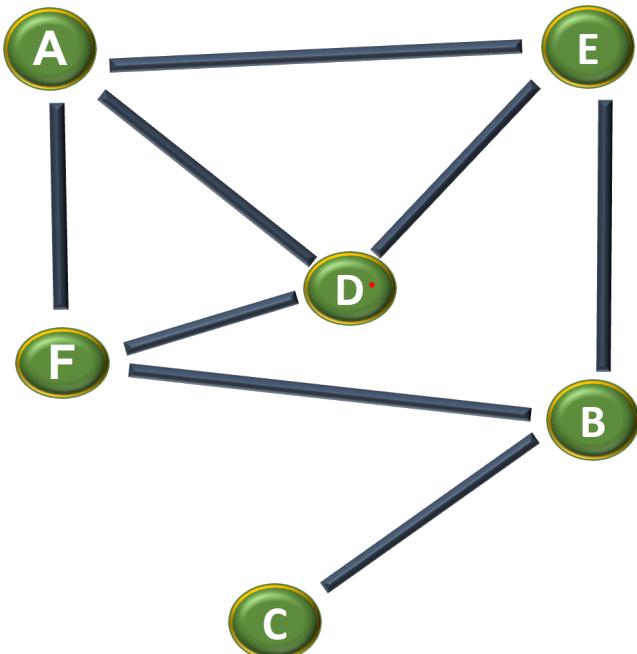


- Đây là thuật toán tìm các đỉnh bằng cách duyệt theo chiều sâu.
- Xuất phát từ 1 đỉnh và đi mãi cho đến khi không thể đi tiếp, sau đó đi về lại đỉnh đầu.
Trong quá trình quay lại:
 - + Nếu gặp đường đi khác thì đi cho đến khi không đi tiếp được nữa
 - + Nếu không tìm ra đường đi nào khác thì ngừng việc tìm kiếm.

TÌM KIẾM CHIỀU SÂU - DEPTH-FIRST SEARCH (DFS)

MỘT SỐ VÍ DỤ

Tìm đường đi theo chiều sâu với các điều kiện sau:



1. Từ đỉnh A đến đỉnh B?

A – D – E – B

2. Từ đỉnh A đến đỉnh C?

A – D – E – B – C

3. Từ đỉnh C đến đỉnh D?

C – B – E – A – D

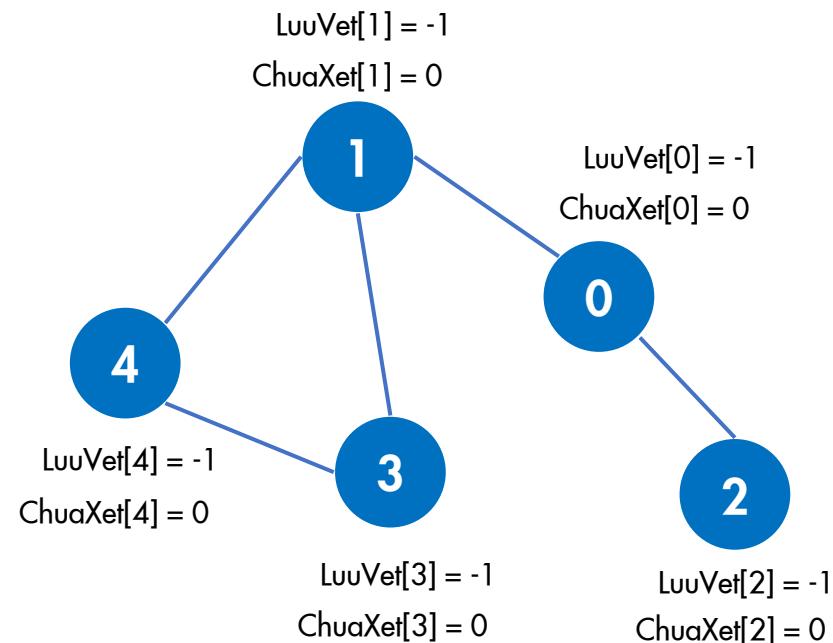
4. Từ đỉnh D đến đỉnh C?

D – A – E – B – C

TÌM KIẾM CHIỀU SÂU - DEPTH-FIRST SEARCH (DFS)

GIẢI THUẬT

Giả sử ta có đồ thị $G=(N, E)$ gồm có 5 đỉnh như sau:



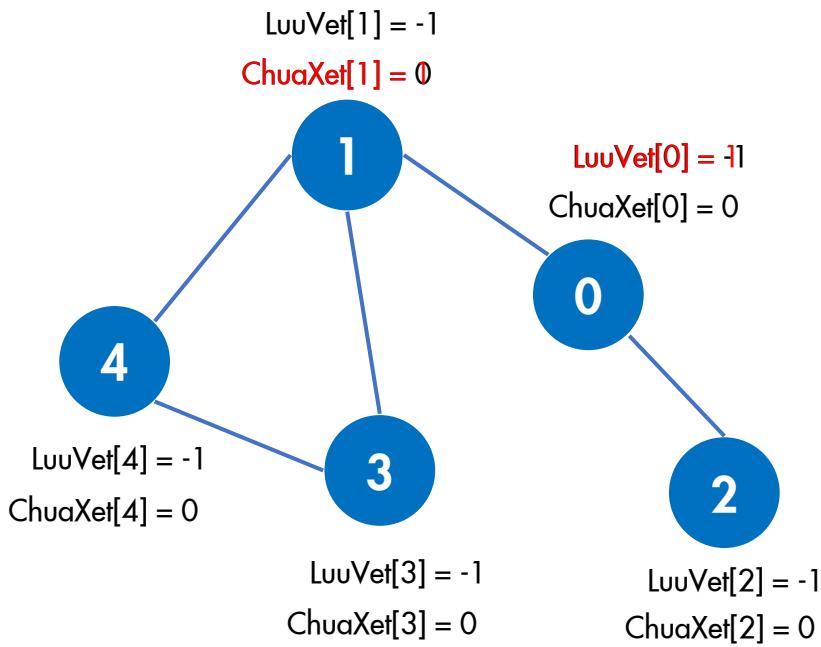
Bước 1: Tạo ra 2 mảng 1 chiều **LuuVet** và **ChuaXet** có kích thước là n .

Bước 2: Gán

- **LuuVet[i] = -1;** // Vì ban đầu chưa chạy thuật toán nên các đỉnh i đều chưa có vết đi đến đó nên đặt giá trị là -1
- **ChuaXet[i] = 0;** // Vì ban đầu chưa chạy thuật toán nên các đỉnh i trong đồ thị g đều chưa được xét đến nên gán giá trị 0

TÌM KIẾM CHIỀU SÂU - DEPTH-FIRST SEARCH (DFS)

GIẢI THUẬT

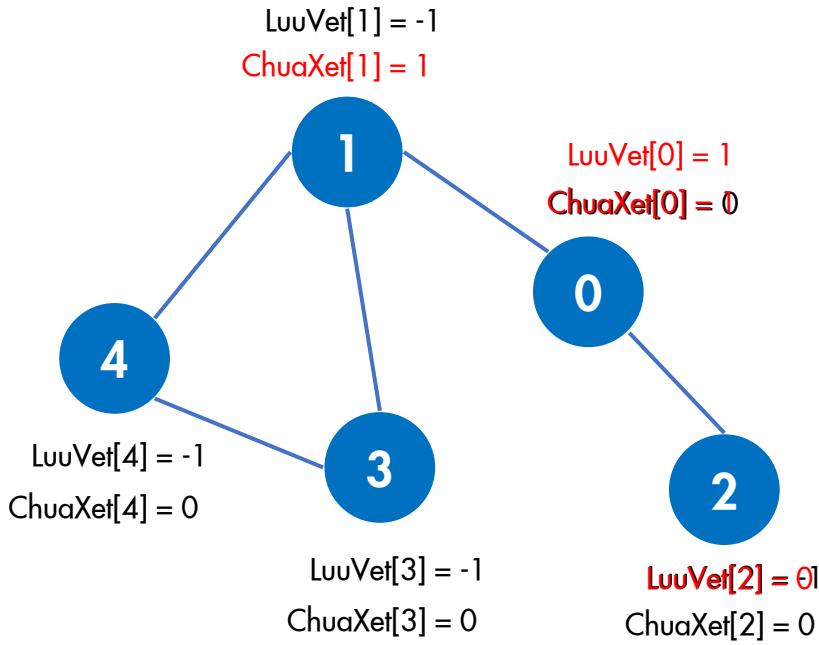


Bước 3: Bắt đầu duyệt và xét 1 đỉnh v nào đó (bắt đầu bước này đầu tiên thì đỉnh v là đỉnh $S = 1$)

- **ChuaXet[1] = 1;** // gán lại giá trị đỉnh 1 trong mảng ChuaXet là 1, điều này có nghĩa là đỉnh 1 đã và đang được xét hay duyệt đến theo thuật toán.
- Từ đỉnh 1 ta xem xét có đường đi đến đỉnh 0, 3, 4. Ở đây ta chọn theo thứ tự là đỉnh 0.
 - **LuuVet[0] = 1;** // đánh dấu lại đỉnh 0 được đi đến từ đỉnh 1 trong quá trình duyệt theo thuật toán DFS
 - Khi đó ta lại quay lui lại bước 3 và xét đỉnh 0.

TÌM KIẾM CHIỀU SÂU - DEPTH-FIRST SEARCH (DFS)

GIẢI THUẬT

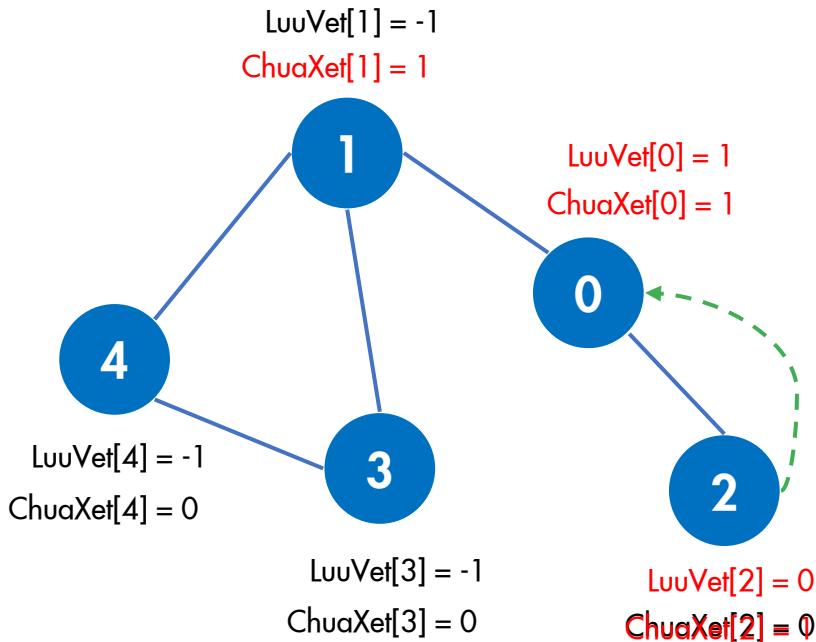


Bước 4: Bắt đầu duyệt và xét đỉnh 0

- **ChuaXet[0] = 1;** // gán lại giá trị đỉnh 0 trong mảng ChuaXet là 1, điều này có nghĩa là đỉnh 0 đã và đang được xét hay duyệt đến theo thuật toán.
- Từ đỉnh 0 ta xem xét có đường đi đến đỉnh 1, 2. Mà đỉnh 1 ta đã xét rồi (**ChuaXet[1] = 1**) nên ta chọn đỉnh 2 vì đỉnh 2 ta chưa xét (**ChuaXet[2] = 0**)
 - **LuuVet[2] = 0;** // đánh dấu lại đỉnh 2 được đi đến từ đỉnh 0 trong quá trình duyệt theo thuật toán DFS
 - Khi đó ta lại quay lui lại bước 3 và xét đỉnh 2.

TÌM KIẾM CHIỀU SÂU - DEPTH-FIRST SEARCH (DFS)

GIẢI THUẬT

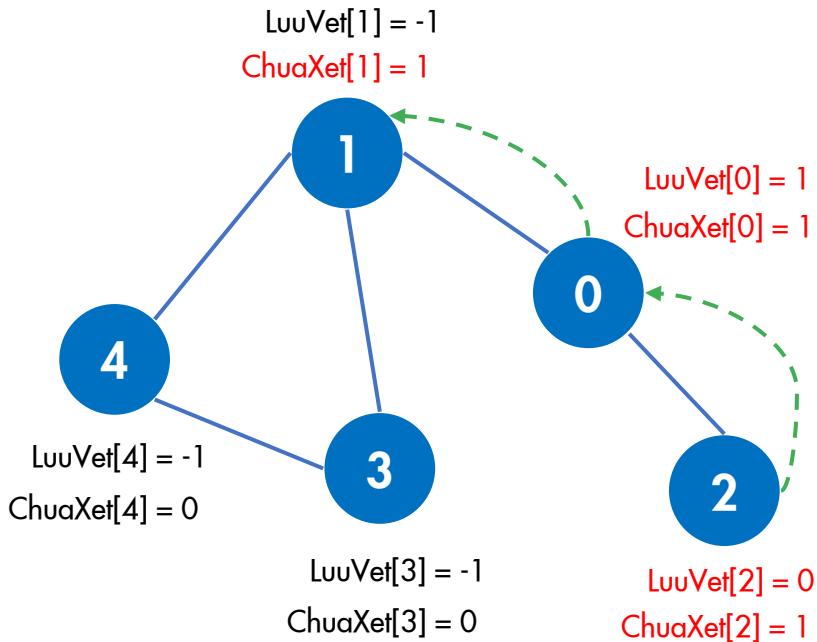


Bước 5: Bắt đầu duyệt và xét đỉnh 2

- **ChuaXet[2] = 1;** // gán lại giá trị đỉnh 2 trong mảng ChuaXet là 1, điều này có nghĩa là đỉnh 2 đã và đang được xét hay duyệt đến theo thuật toán.
- Từ đỉnh 2 ta xem xét có đường đi đến đỉnh 0. Mà đỉnh 0 ta đã xét rồi (**ChuaXet[0] = 1**) và cũng ko còn có đỉnh nào từ đỉnh 2 mà chưa xét nên ta quay lại đỉnh trước đó (đỉnh 0) và xét tiếp đỉnh đó (đỉnh 0) (đệ qui_mảng lưu vết)

TÌM KIẾM CHIỀU SÂU - DEPTH-FIRST SEARCH (DFS)

GIẢI THUẬT

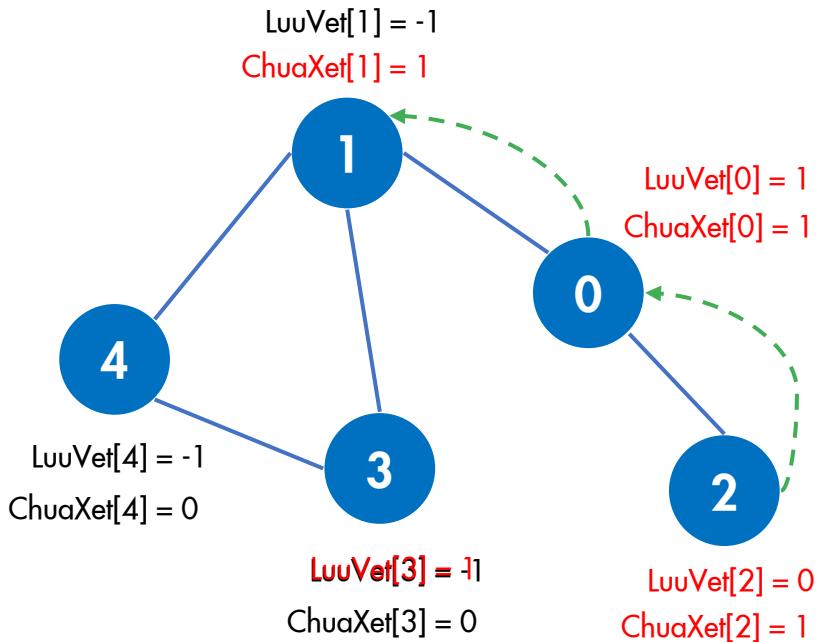


Bước 6: Xét đỉnh 0

Từ đỉnh 0 ta xem xét có đường đi đến đỉnh 1, 2. Mà đỉnh 1, 2 ta đã xét rồi ($\text{ChuaXet}[1] = 1$, $\text{ChuaXet}[2] = 1$) và cũng ko còn có đỉnh nào từ đỉnh 0 mà chưa xét nên ta quay lại đỉnh trước đó (đỉnh 1) và xét tiếp đỉnh đó (đỉnh 1) (đệ qui_mảng lưu vết).

TÌM KIẾM CHIỀU SÂU - DEPTH-FIRST SEARCH (DFS)

GIẢI THUẬT

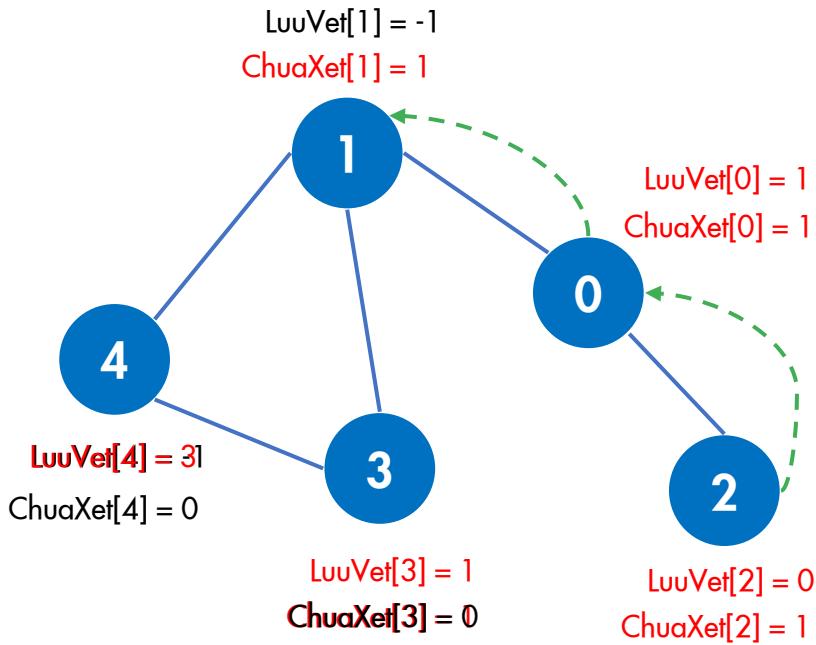


Bước 7: Xét định 1

- Từ đỉnh 1 ta xem xét có đường đi đến đỉnh 0, 3, 4.
Mà đỉnh 0 ta đã xét rồi (**ChuaXet[0] = 1**) nên ta
chọn đỉnh 3 (theo thứ tự) (**ChuaXet[3] = 0**)
 - **LuuVet[3] = 1**; // đánh dấu lại đỉnh 3 được đi đến từ đỉnh 1
trong quá trình duyệt theo thuật toán DFS
 - Khi đó ta lại quay lại bước 3 và xét đỉnh 3

TÌM KIẾM CHIỀU SÂU - DEPTH-FIRST SEARCH (DFS)

GIẢI THUẬT

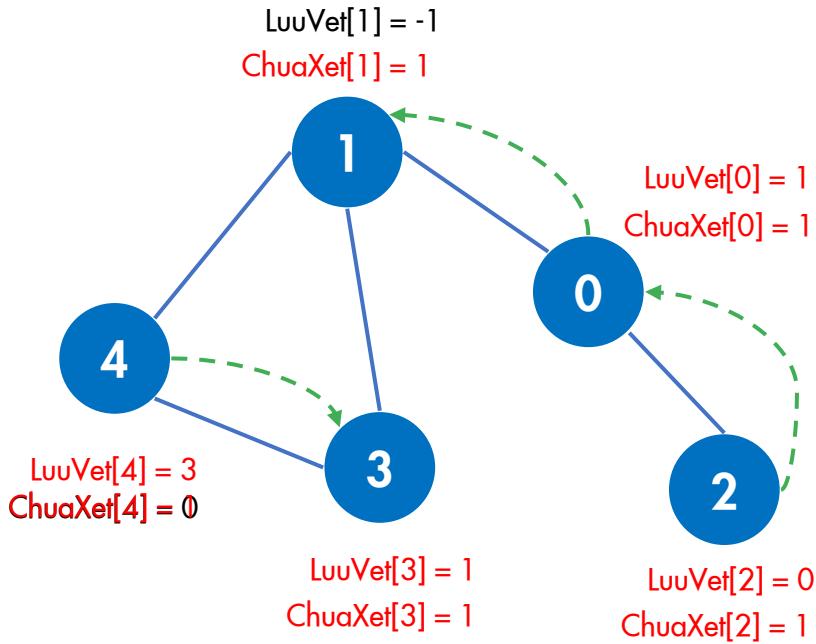


Bước 8: Bắt đầu duyệt và xét đỉnh 3

- `ChuaXet[3] = 1`; // gán lại giá trị đỉnh 3 trong mảng `ChuaXet` là 1, điều này có nghĩa là đỉnh 3 đã và đang được xét hay duyệt đến theo thuật toán.
- Từ đỉnh 3 ta xem xét có đường đi đến đỉnh 1, 4. Mà đỉnh 1 ta đã xét rồi (`ChuaXet[1] = 1`) nên ta chọn đỉnh 4 vì đỉnh 2 ta chưa xét (`ChuaXet[4] = 0`)
- `LuuVet[4] = 3`; // đánh dấu lại đỉnh 4 được đi đến từ đỉnh 3 trong quá trình duyệt theo thuật toán DFS
- Khi đó ta lại quay lại bước 3 và xét đỉnh 4

TÌM KIẾM CHIỀU SÂU - DEPTH-FIRST SEARCH (DFS)

GIẢI THUẬT

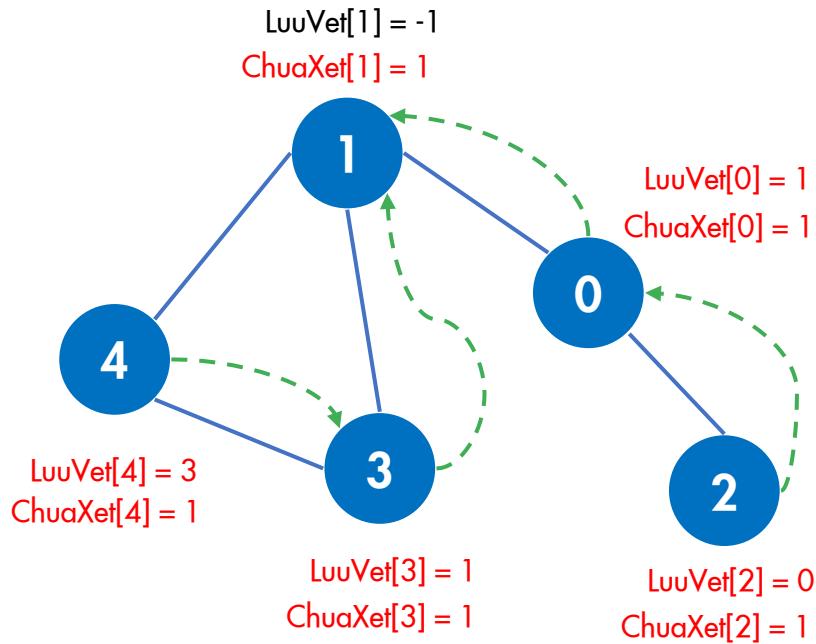


Bước 9: Bắt đầu duyệt và xét đỉnh 4

- **ChuaXet[4] = 1**; // gán lại giá trị đỉnh 4 trong mảng ChuaXet là 1, điều này có nghĩa là đỉnh 4 đã và đang được xét hay duyệt đến theo thuật toán.
- Từ đỉnh 4 ta xem xét có đường đi đến đỉnh 1, 3. Mà đỉnh 1, 3 ta đã xét rồi (**ChuaXet[1] = 1**, **ChuaXet[3] = 1**) nên ta không còn đỉnh nào chưa xét mà đi từ 4. Do đó ta quay lại đỉnh đi tới nó là đỉnh 3 và xét đỉnh 3.

TÌM KIẾM CHIỀU SÂU - DEPTH-FIRST SEARCH (DFS)

GIẢI THUẬT

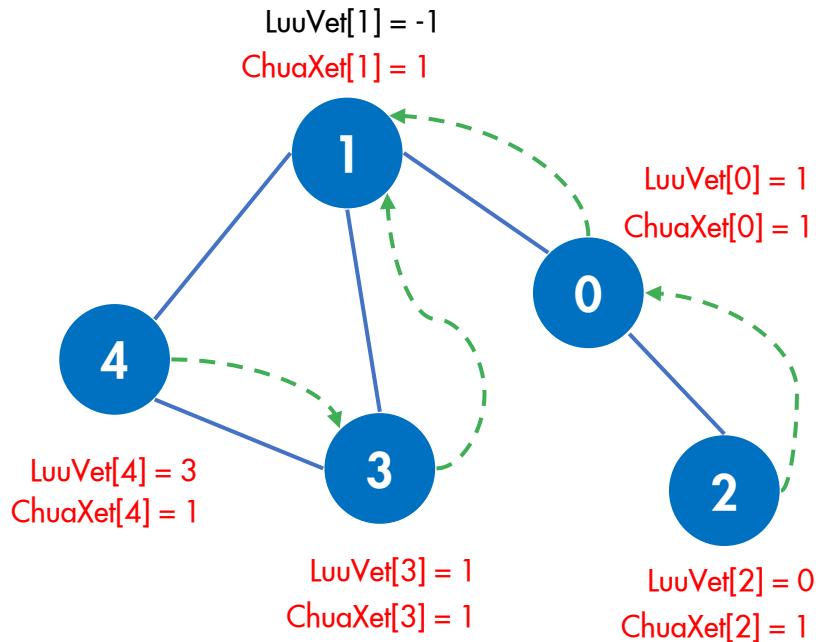


Bước 11: Xét đỉnh 3

Từ đỉnh 3 ta xem xét có đường đi đến đỉnh 1, 4. Mà đỉnh 1, 4 ta đã xét rồi ($\text{ChuaXet}[1] = 1$, $\text{ChuaXet}[4] = 1$) nên ta không còn đỉnh nào chưa xét mà đi từ 3. Do đó ta quay lại đỉnh đi tới nó là đỉnh 1 và xét đỉnh 1.

TÌM KIẾM CHIỀU SÂU - DEPTH-FIRST SEARCH (DFS)

GIẢI THUẬT



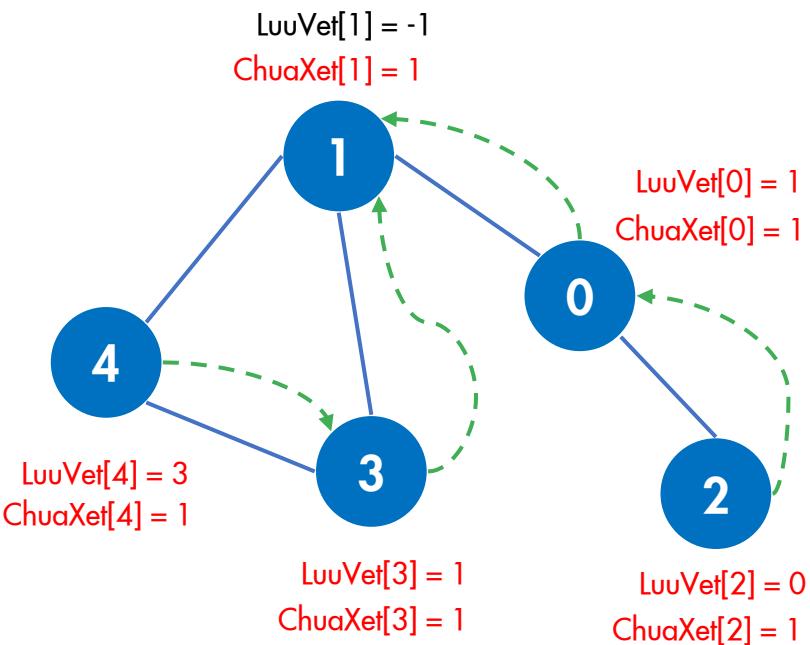
Bước 12: Xét đỉnh 3

Từ đỉnh 1 ta xem xét có đường đi đến đỉnh 0, 3, 4. Mà đỉnh 1, 4 ta đã xét rồi ($\text{ChuaXet}[0] = 1$, $\text{ChuaXet}[3] = 1$, $\text{ChuaXet}[4] = 1$) nên ta không còn đỉnh nào chưa xét mà đi từ 1. Do đó ta quay lại đỉnh đi tới nó mà đỉnh đi tới nó là -1 nên dừng thuật toán duyệt DFS ở đây.

TÌM KIẾM CHIỀU SÂU - DEPTH-FIRST SEARCH (DFS)

GIẢI THUẬT

Sau khi duyệt DFS xong, bạn muốn biết đường đi từ đỉnh S (là đỉnh 1 trong trường hợp này) thì bạn căn cứ vào mảng LuuVet và ChuaXet.

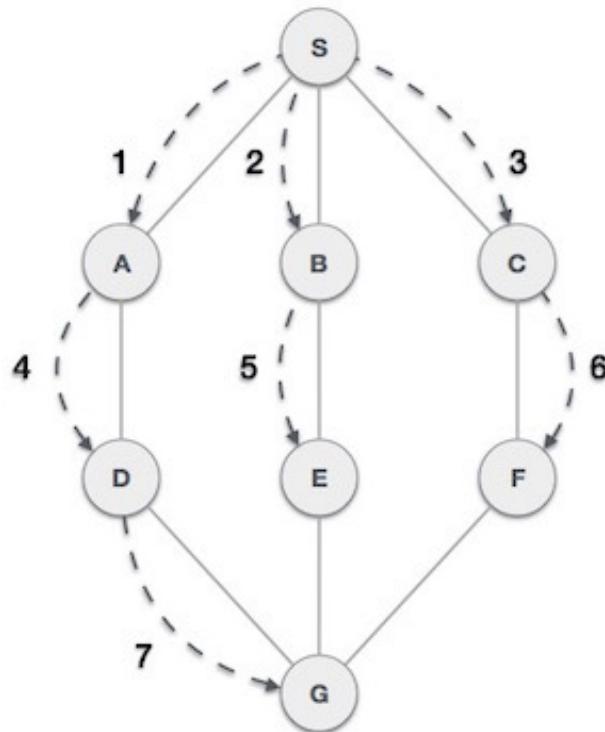


LuuVet[0]	LuuVet[1]	LuuVet[2]	LuuVet[3]	LuuVet[4]
1	-1	0	1	3
ChuaXet[0]	ChuaXet[1]	ChuaXet[2]	ChuaXet[3]	ChuaXet[4]
1	1	1	1	1

Giả sử, bạn muốn tìm đường đi từ đỉnh 1 đến đỉnh 4. Thì đầu tiên căn cứ vào giá trị của mảng ChuaXet. **ChuaXet[4] = 1**
LuuVet[4] = 3 → LuuVet[3] = 1 → LuuVet[1] = -1
Ta thấy đỉnh 1 là đỉnh xuất phát đường đi nên dừng.
Vậy đường đi từ 1 đến 4 là **4 → 3 → 1**.

TÌM KIẾM CHIỀU RỘNG – BREADTH-FIRST SEARCH (BFS)

Ý TƯỞNG THUẬT TOÁN



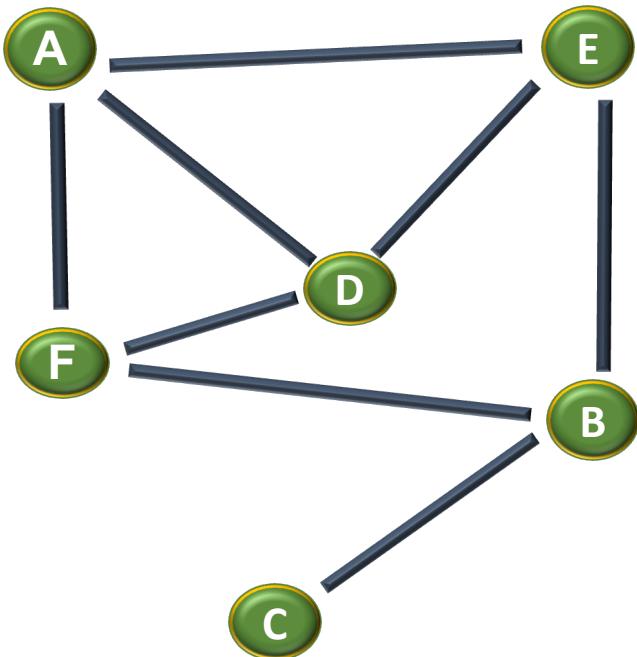
Từ một đỉnh (nút) gốc ban đầu.

- Xác định và lần lượt duyệt các đỉnh kề xung quanh đỉnh gốc vừa xét.
- Tiếp tục quá trình duyệt qua các đỉnh kề đỉnh vừa xét cho đến khi đạt được kết quả cần tìm hoặc duyệt qua tất cả các đỉnh.

TÌM KIẾM CHIỀU RỘNG – BREADTH-FIRST SEARCH (BFS)

MỘT SỐ VÍ DỤ

Tìm đường đi theo chiều rộng với các điều kiện sau:



1. Từ đỉnh A đến đỉnh B?

A – E – B

2. Từ đỉnh B đến đỉnh A?

B – E – A

3. Từ đỉnh C đến đỉnh D?

C – B – E – D

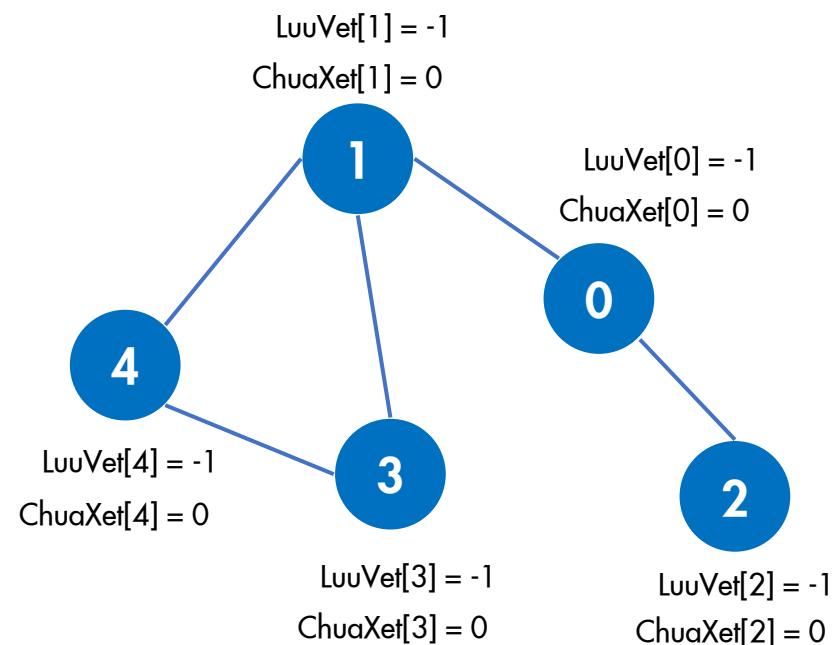
4. Từ đỉnh D đến đỉnh F?

D – F

TÌM KIẾM CHIỀU RỘNG – BREADTH-FIRST SEARCH (BFS)

GIẢI THUẬT

Giả sử ta có đồ thị $G=(N, E)$ gồm có 5 đỉnh như sau:



Bước 1: Tạo ra 2 mảng 1 chiều **LuuVet** và **ChuaXet** có kích thước là n .

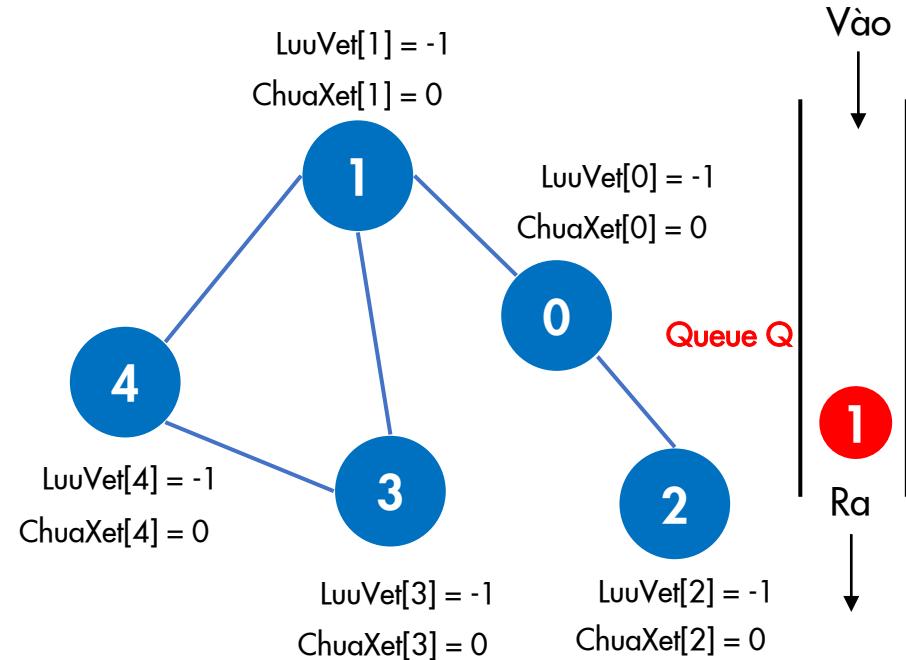
Bước 2: Gán

- **LuuVet[i] = -1;** // Vì ban đầu chưa chạy thuật toán nên các đỉnh i đều chưa có vết đi đến đó nên đặt giá trị là -1
- **ChuaXet[i] = 0;** // Vì ban đầu chưa chạy thuật toán nên các đỉnh i trong đồ thị g đều chưa được xét đến nên gán giá trị 0

TÌM KIẾM CHIỀU RỘNG – BREADTH-FIRST SEARCH (BFS)

GIẢI THUẬT

Giả sử ta có đồ thị $G=(N, E)$ gồm có 5 đỉnh như sau:



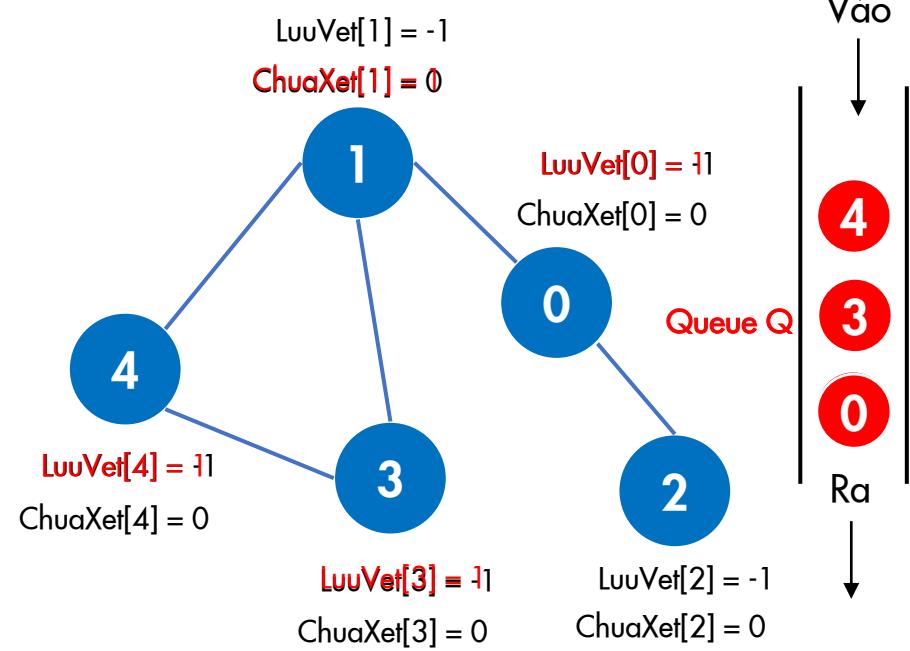
Bước 3: Xây dựng với QUEUE các hàm cần thiết:

- KhoiTaoQueue.
- DayGiaTriVaoQueue
- LayGiaTriRaKhoiQueue
- KiemTraQueueRong.

Đẩy đỉnh 1 (tức là đỉnh S) vào hàng đợi Q

TÌM KIẾM CHIỀU RỘNG – BREADTH-FIRST SEARCH (BFS)

GIẢI THUẬT



Bước 4: Kiểm tra hàng đợi Queue Q có rỗng ko.

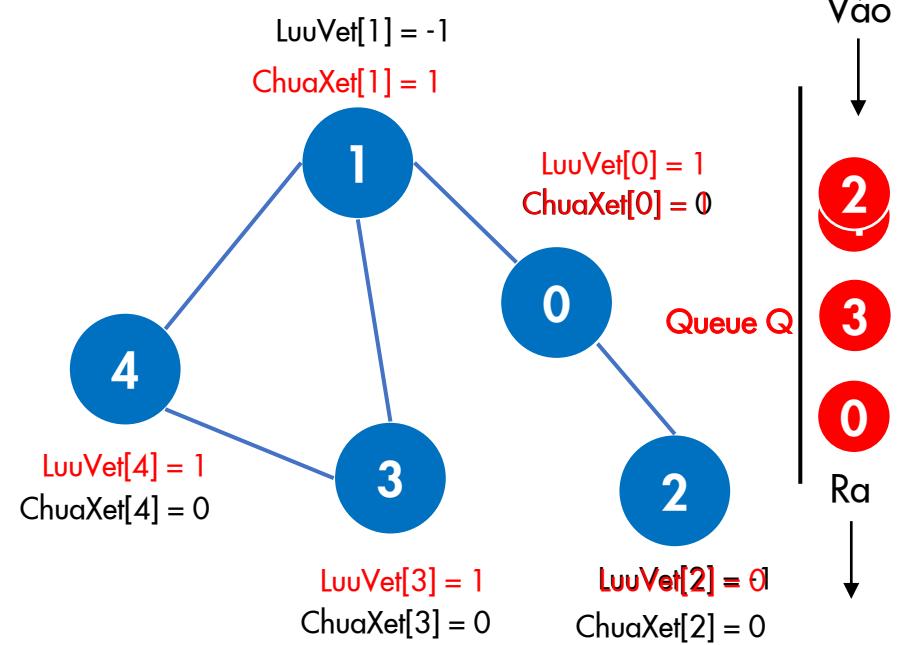
Hàng đợi Queue Q không rỗng.

Lấy điểm dưới cùng ra. Là đỉnh 1 và tiến hành xét đỉnh 1.

- **ChuaXet[1] = 1;**
- Từ đỉnh 1 ta xem xét có đường đi đến đỉnh 0, 3, 4. Và 3 đỉnh này đều có giá trị ChuaXet là 0.
- Đẩy 3 đỉnh 0, 3, 4 vào Queue Q.
- **LuuVet[0] = 1; LuuVet[3] = 1; LuuVet[4] = 1;**
Sau đó ta quay lại đầu bước này

TÌM KIẾM CHIỀU RỘNG – BREADTH-FIRST SEARCH (BFS)

GIẢI THUẬT



Bước 5: Kiểm tra hàng đợi Queue Q có rỗng ko.

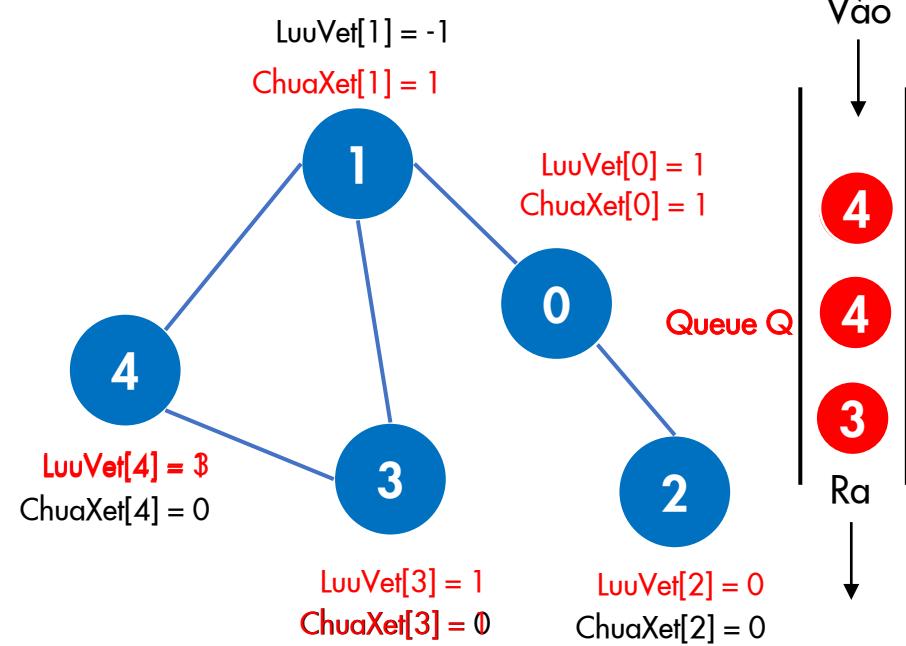
Hàng đợi Queue Q không rỗng.

Lấy điểm dưới cùng ra. Là đỉnh 0 và tiến hành xét đỉnh 0.

- **ChuaXet[0] = 1;**
 - Từ đỉnh 0 ta xem xét có đường đi đến đỉnh 1 và 2. Vì đỉnh 1 đã xet rồi (**ChuaXet[1] = 1**) nên bỏ qua, đỉnh 2 thì chưa xét nên:
 - Đẩy đỉnh 2 vào Queue Q.
 - **LuuVet[2] = 0;**
- Sau đó ta quay lại đầu bước này.

TÌM KIẾM CHIỀU RỘNG – BREADTH-FIRST SEARCH (BFS)

GIẢI THUẬT



Bước 6: Kiểm tra hàng đợi Queue Q có rỗng ko.

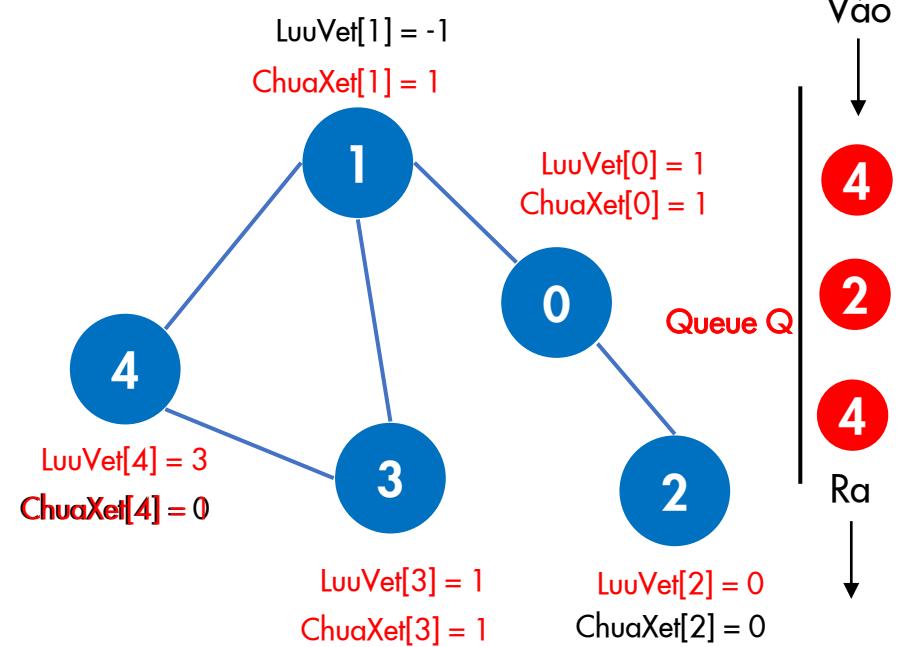
Hàng đợi Queue Q không rỗng.

Lấy điểm dưới cùng ra. Là đỉnh 3 và tiến hành xét đỉnh 3.

- **ChuaXet[3] = 1;**
 - Từ đỉnh 3 ta xem xét có đường đi đến đỉnh 1 và 4. Và đỉnh 1 thì xét rồi (**ChuaXet[1] = 1**) nên bỏ qua, đỉnh 4 thì chưa xét nên:
 - Đẩy đỉnh 4 vào Queue Q.
 - **LuuVet[4] = 3;**
- Sau đó ta quay lại đầu bước này.

TÌM KIẾM CHIỀU RỘNG – BREADTH-FIRST SEARCH (BFS)

GIẢI THUẬT



Bước 7: Kiểm tra hàng đợi Queue Q có rỗng ko.

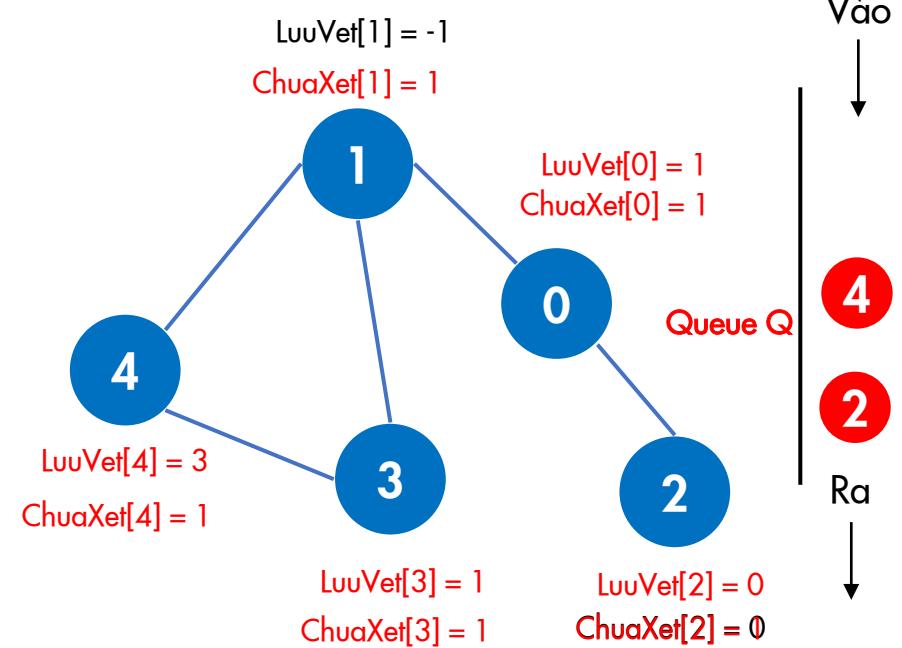
Hàng đợi Queue Q không rỗng.

Lấy điểm dưới cùng ra. Là đỉnh 4 và tiến hành xét đỉnh 4.

- **ChuaXet[4] = 1;**
- Từ đỉnh 4 ta xem xét có đường đi đến đỉnh 1, 3. Vì đỉnh 1, 3 thì xét rồi (**ChuaXet[1] = 1, ChuaXet[3] = 1**) nên bỏ qua. Vậy ko có đỉnh nào mới đưa vào QUEUE Q.
- Sau đó ta quay lại đầu bước này.

TÌM KIẾM CHIỀU RỘNG – BREADTH-FIRST SEARCH (BFS)

GIẢI THUẬT



Bước 8: Kiểm tra hàng đợi Queue Q có rỗng ko.

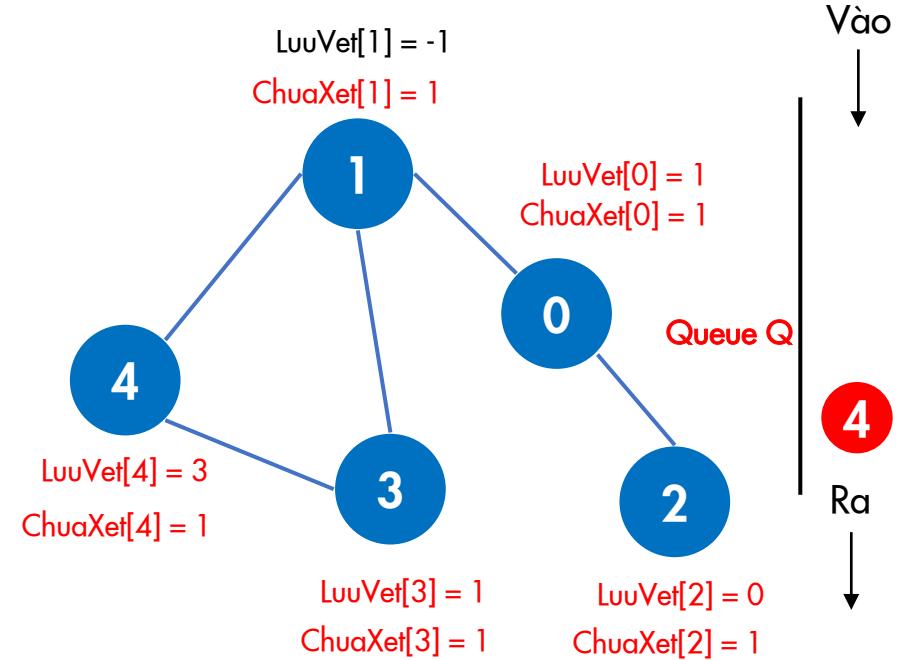
Hàng đợi Queue Q không rỗng.

Lấy điểm dưới cùng ra. Là đỉnh 2 và tiến hành xét đỉnh 2.

- ChuaXet[2] = 1;
- Từ đỉnh 2 ta xem xét có đường đi đến đỉnh 0. Vì đỉnh 0 đã được thăm dò (ChuaXet[0] = 1) nên bỏ qua. Vậy ko có đỉnh nào mới đưa vào QUEUE Q.
- Sau đó ta quay lại bước này.

TÌM KIẾM CHIỀU RỘNG – BREADTH-FIRST SEARCH (BFS)

GIẢI THUẬT



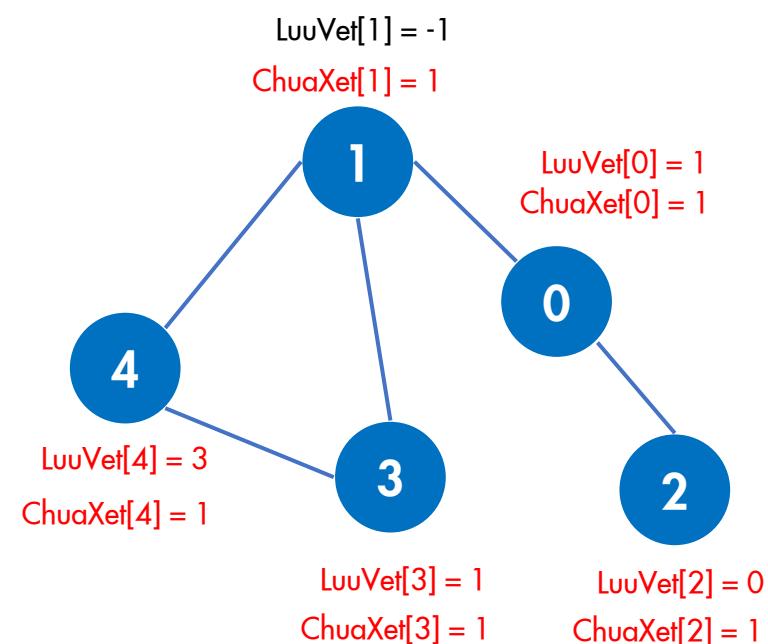
Bước 9: Kiểm tra hàng đợi Queue Q có rỗng ko.
Hàng đợi Queue Q không rỗng.

- Lấy điểm dưới cùng ra. Là đỉnh 4 và tiến hành xét đỉnh 4. Mà đỉnh 4 có giá trị $ChuaXet[4] = 1$; nên không cần phải xét nữa.
- Tới đây, ta thấy Queue Q rỗng nên thuật toán dừng.

TÌM KIẾM CHIỀU RỘNG – BREADTH-FIRST SEARCH (BFS)

GIẢI THUẬT

Sau khi duyệt BFS xong, bạn muốn biết đường đi từ đỉnh S (là đỉnh 1 trong trường hợp này) thì bạn căn cứ vào mảng LuuVet và ChuaXet.



LuuVet[0]	LuuVet[1]	LuuVet[2]	LuuVet[3]	LuuVet[4]
1	-1	0	1	3
ChuaXet[0]	ChuaXet[1]	ChuaXet[2]	ChuaXet[3]	ChuaXet[4]
1	1	1	1	1

Giả sử, bạn muốn tìm đường đi từ đỉnh 1 đến đỉnh 4. Thì đầu tiên căn cứ vào giá trị của mảng ChuaXet. $ChuaXet[4] = 1$

$$LuuVet[4] = 3 \rightarrow LuuVet[3] = 1 \rightarrow LuuVet[1] = -1$$

Ta thấy đỉnh 1 là đỉnh xuất phát đường đi nên dừng.

Vậy đường đi từ 1 đến 4 là $4 \rightarrow 3 \rightarrow 1$.



TUẦN 4

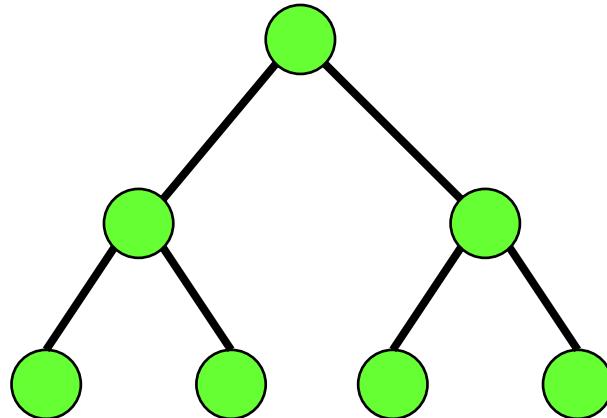
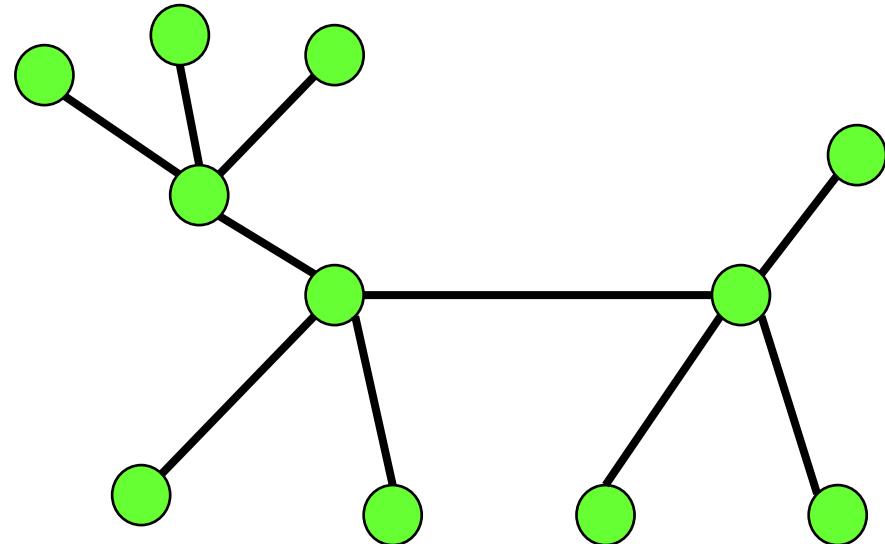
CÂY KHUNG NHỎ NHẤT

PRIM - KRUSKAL

CÂY LÀ GÌ?

Ta gọi cây là đồ thị vô hướng liên thông không có chu trình.

Rừng là hợp (disjoint union) của các cây.

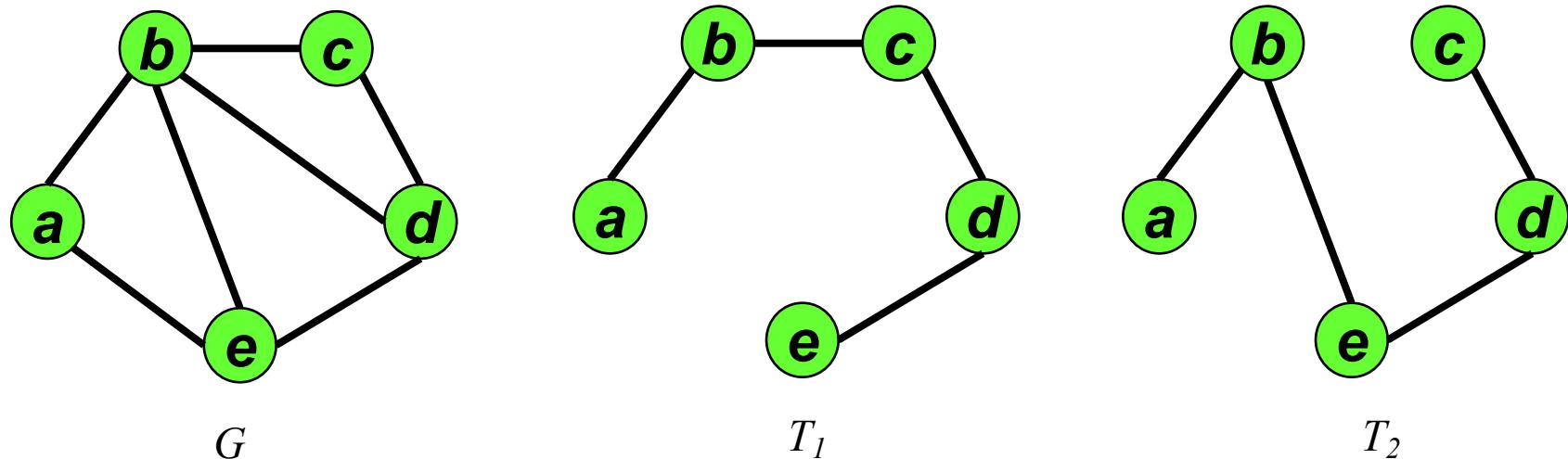
 T_1  T_2

Rừng F gồm 2 cây T_1, T_2

CÂY KHUNG ĐỒ THỊ?

Giả sử $G = (N, E)$ là đồ thị vô hướng liên thông.

Cây $T = (N, F)$ với $F \subset E$ được gọi là cây khung của đồ thị G .

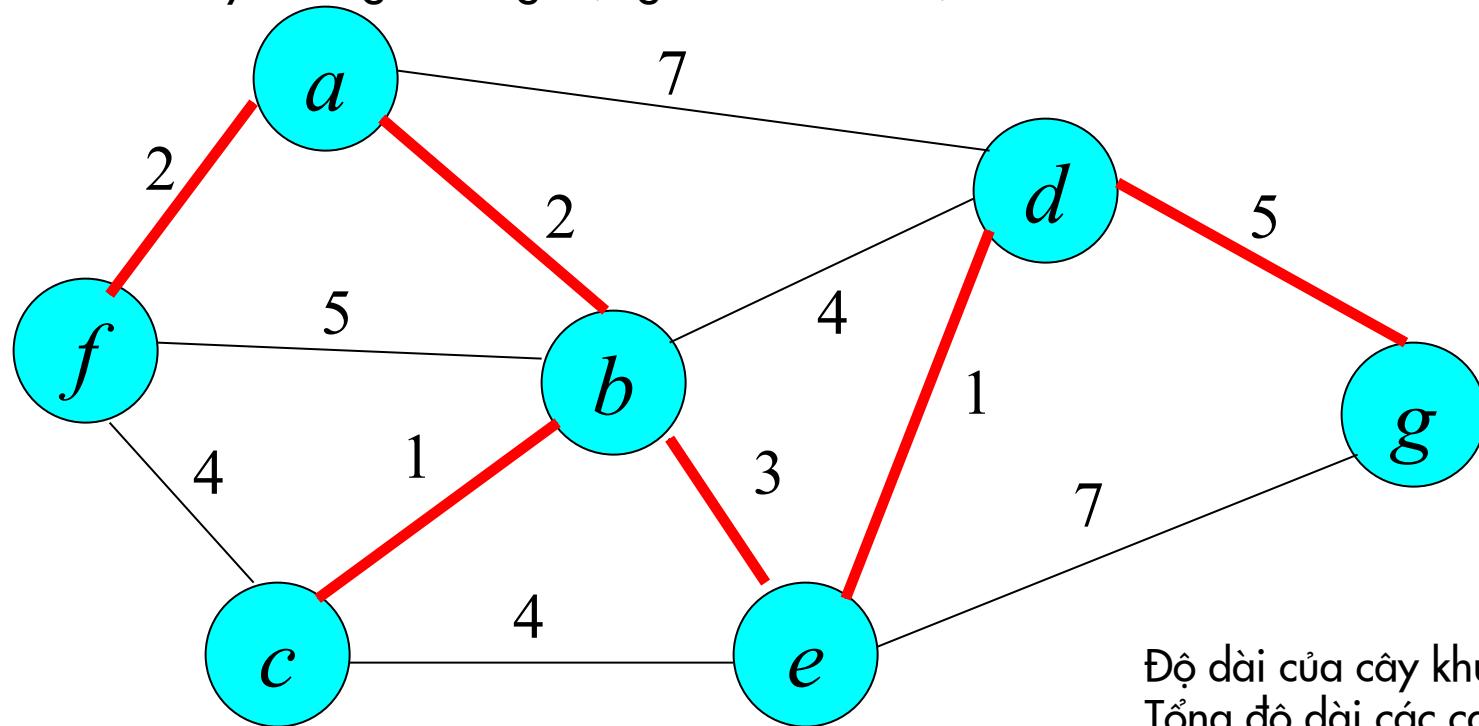


Đồ thị G và 2 cây khung T_1 và T_2 của nó

CÂY KHUNG NHỎ NHẤT

Cho đồ thị vô hướng liên thông $G=(N,E)$ với trọng số $c(e)$, $e \in E$.

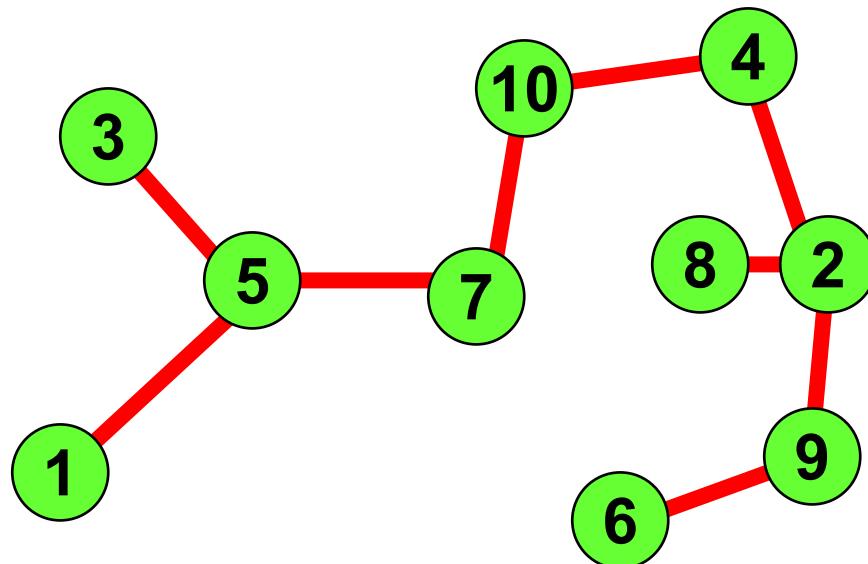
Độ dài của cây khung là tổng trọng số trên các cạnh của nó.



Độ dài của cây khung là
Tổng độ dài các cạnh: 14

CÂY KHUNG NHỎ NHẤT

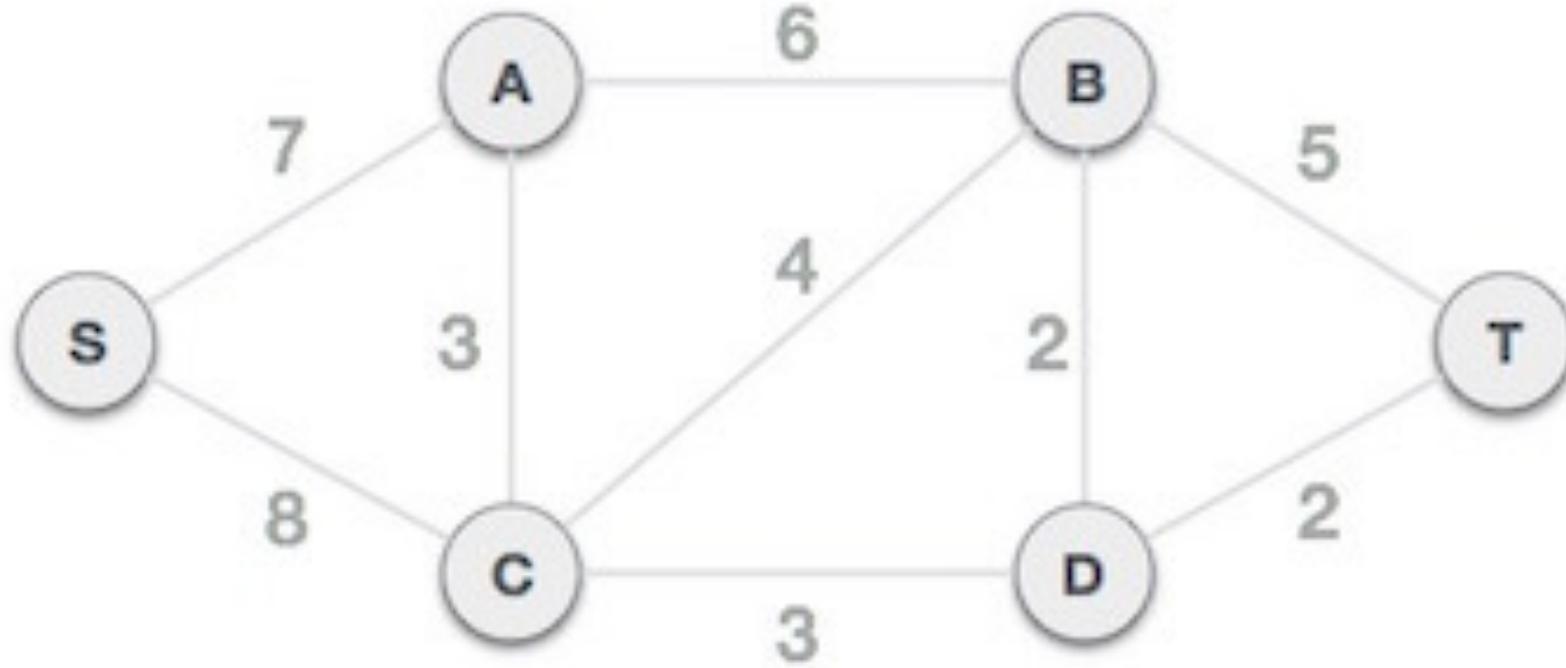
Công ty truyền thông Viettel cần xây dựng mạng truyền thông kết nối n khách hàng. Chi phí thực hiện kênh nối i và j là c_{ij} . Hỏi chi phí nhỏ nhất để thực hiện việc kết nối tất cả các khách hàng là bao nhiêu?



Giả thiết là: Chỉ có cách kết nối duy nhất là đặt kênh nối trực tiếp giữa hai nút.

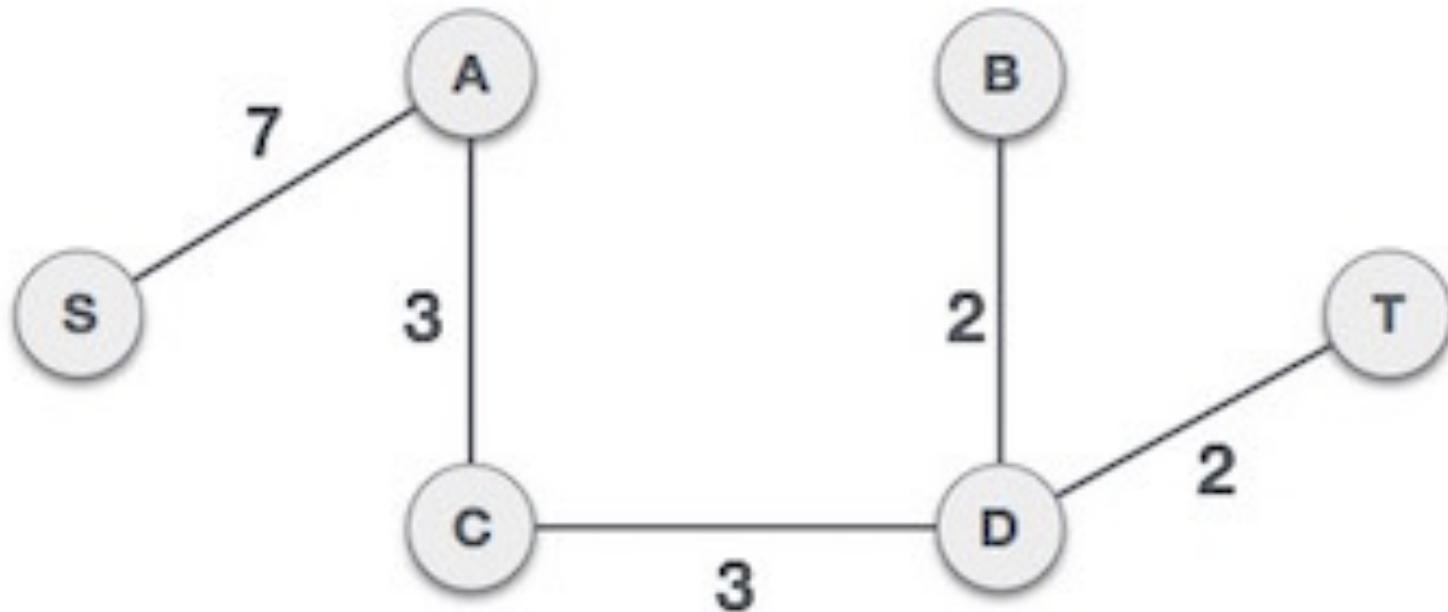
VÍ DỤ CÂY KHUNG NHỎ NHẤT

Tìm cây khung nhỏ nhất của đồ thị sau:



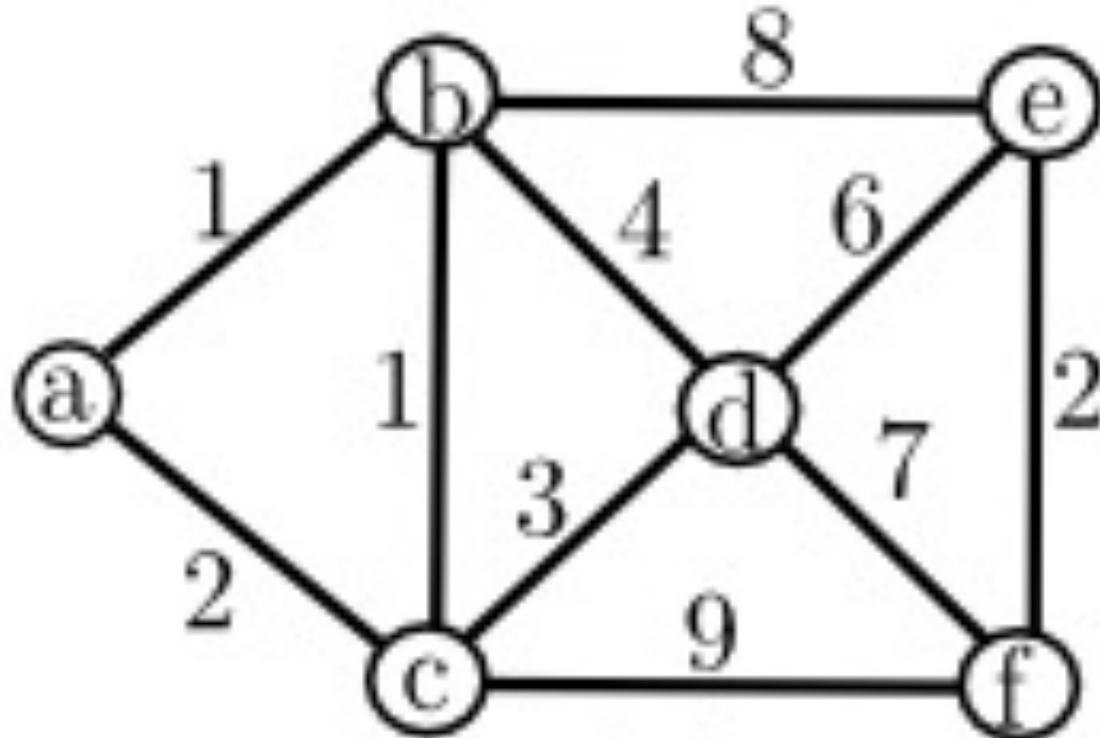
VÍ DỤ CÂY KHUNG NHỎ NHẤT

Tìm cây khung nhỏ nhất của đồ thị sau:



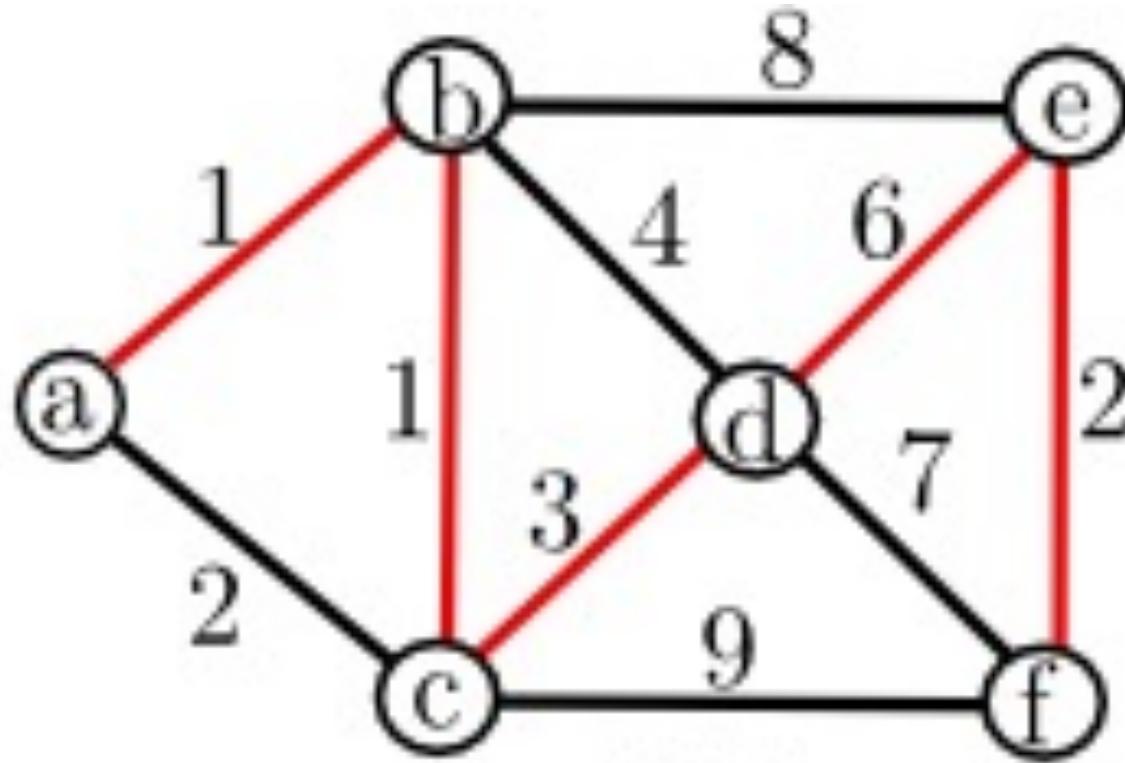
VÍ DỤ CÂY KHUNG NHỎ NHẤT

Tìm cây khung nhỏ nhất của đồ thị sau:



VÍ DỤ CÂY KHUNG NHỎ NHẤT

Tìm cây khung nhỏ nhất của đồ thị sau:



THUẬT TOÁN PRIM

Thuật toán xuất phát từ một cây chỉ chứa đúng một đỉnh và mở rộng từng bước một, mỗi bước thêm một cạnh mới vào cây, cho tới khi bao trùm được tất cả các đỉnh của đồ thị.

Dữ liệu vào: Một đồ thị có trọng số liên thông với tập hợp đỉnh N và tập hợp cạnh E (trọng số có thể âm). Đồng thời cũng dùng N và E để ký hiệu số đỉnh và số cạnh của đồ thị.

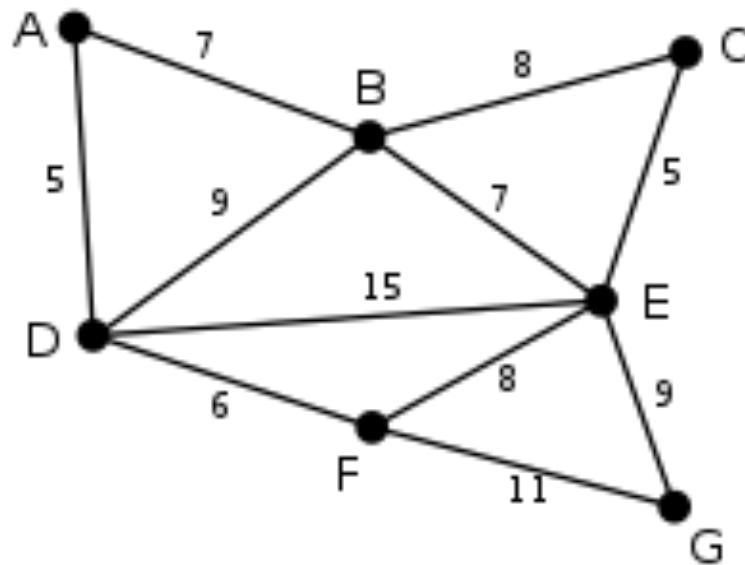
Khởi tạo: $N_{\text{mới}} = \{x\}$, trong đó x là một đỉnh bất kì (đỉnh bắt đầu) trong N , $E_{\text{mới}} = \{\}$

Lặp lại cho tới khi $N_{\text{mới}} = N$:

- Chọn cạnh (u, v) có trọng số nhỏ nhất thỏa mãn u thuộc $N_{\text{mới}}$ và v không thuộc $N_{\text{mới}}$ (nếu có nhiều cạnh như vậy thì chọn một cạnh bất kì trong chúng)
- Thêm v vào $N_{\text{mới}}$, và thêm cạnh (u, v) vào $E_{\text{mới}}$

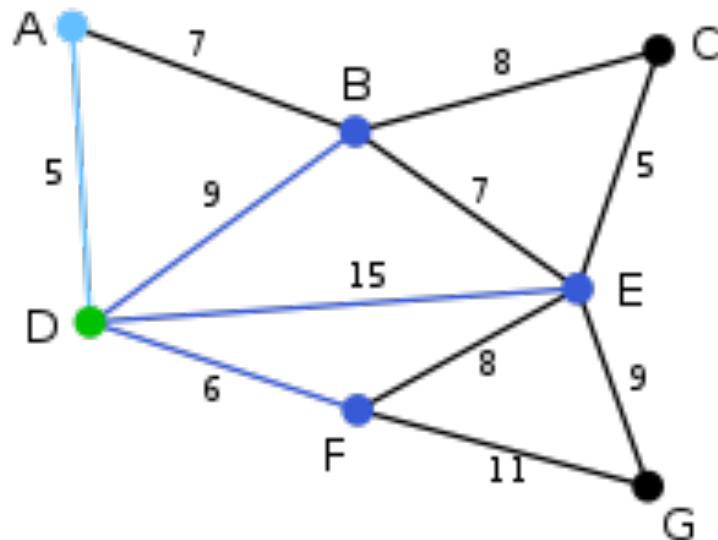
Dữ liệu ra: $N_{\text{mới}}$ và $E_{\text{mới}}$ là tập hợp đỉnh và tập hợp cạnh của một cây bao trùm nhỏ nhất

THUẬT TOÁN PRIM



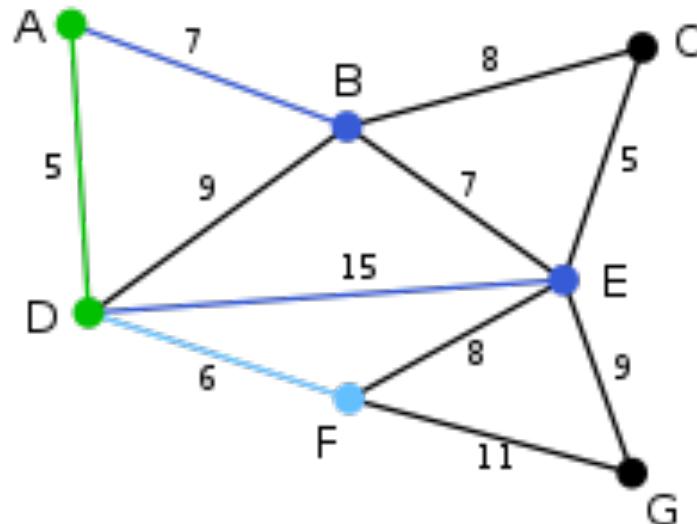
X	Cạnh (u,v)	$N \setminus X$	Mô tả
{}		{A,B,C,D,E,F,G}	Đây là đồ thị có trọng số ban đầu. Các số là các trọng số của các cạnh.

THUẬT TOÁN PRIM



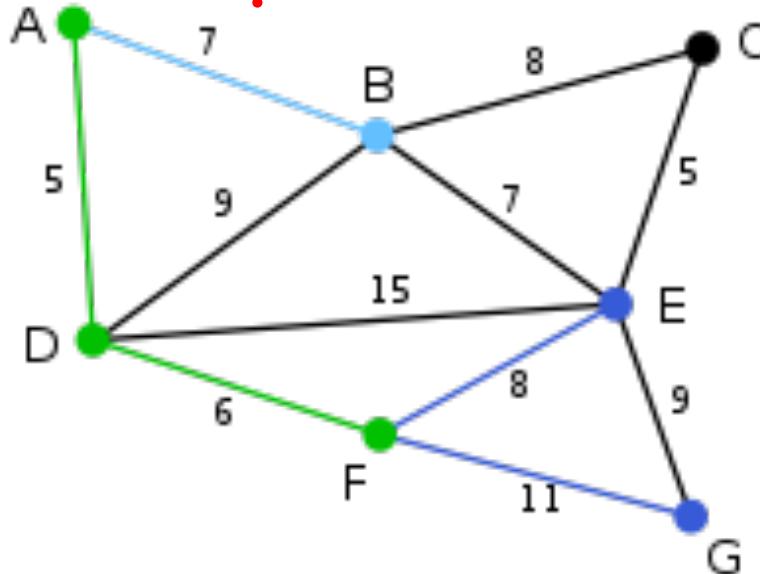
X	Cạnh (u,v)	N \ X	Mô tả
{D}	(D,A) = 5 (D,B) = 9 (D,E) = 15 (D,F) = 6	{A,B,C,E,F,G}	Chọn một cách tùy ý đỉnh D là đỉnh bắt đầu. Các đỉnh A, B, E và F đều được nối trực tiếp tới D bằng cạnh của đồ thị. A là đỉnh gần D nhất nên ta chọn A là đỉnh thứ hai của cây và thêm cạnh AD vào cây.

THUẬT TOÁN PRIM



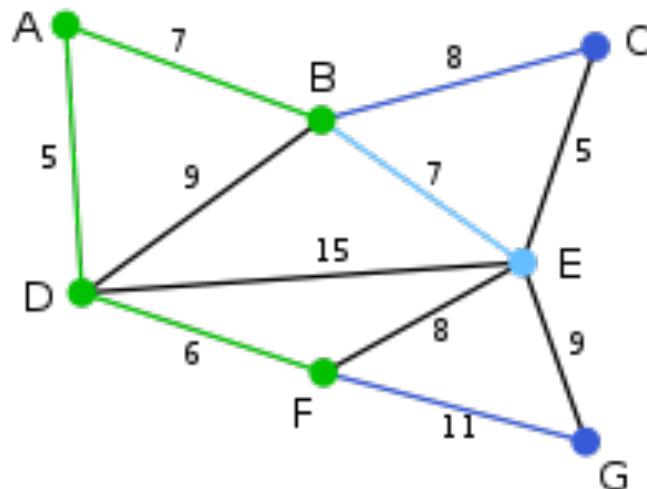
X	Cạnh (u,v)	N \ X	Mô tả
{A,D}	(D,B) = 9 (D,E) = 15 (D,F) = 6 (A,B) = 7	{B,C,E,F,G}	Đỉnh được chọn tiếp theo là đỉnh gần D hoặc A nhất. B có khoảng cách tới D bằng 9 và tới A bằng 7, E có khoảng cách tới cây hiện tại bằng 15, và F có khoảng cách bằng 6. F là đỉnh gần cây hiện tại nhất nên chọn đỉnh F và cạnh DF.

THUẬT TOÁN PRIM



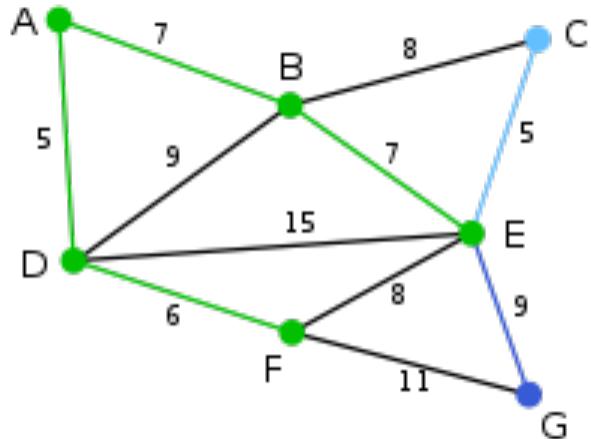
X	Cạnh (u,v)	$N \setminus X$	Mô tả
{A,D,F}	$(D,B) = 9$ $(D,E) = 15$ $(A,B) = 7$ $(F,E) = 8$ $(F,G) = 11$	{B,C,E,G}	Thuật toán tiếp tục tương tự như bước trước. Chọn đỉnh B có khoảng cách tới A bằng 7.

THUẬT TOÁN PRIM



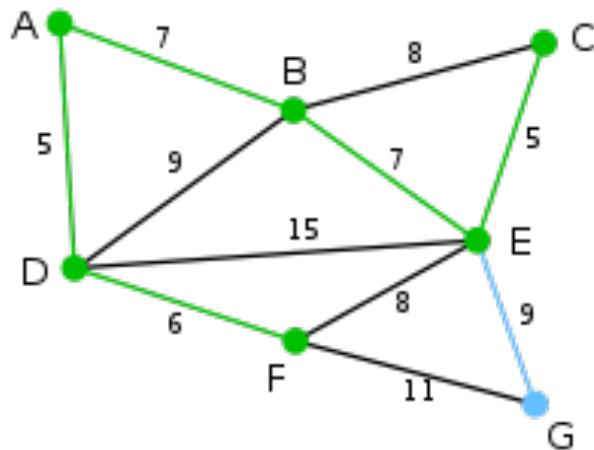
X	Cạnh (u,v)	$N \setminus X$	Mô tả
$\{A,B,D,F\}$	$(B,C) = 8$ $(B,E) = 7$ $(D,B) = 9$ chu trình $(D,E) = 15$ $(F,E) = 8$ $(F,G) = 11$	$\{C,E,G\}$	Ở bước này ta chọn giữa C , E , và G . C có khoảng cách tới B bằng 8 , E có khoảng cách tới B bằng 7 , và G có khoảng cách tới F bằng 11 . E là đỉnh gần nhất, nên chọn đỉnh E và cạnh BE .

THUẬT TOÁN PRIM



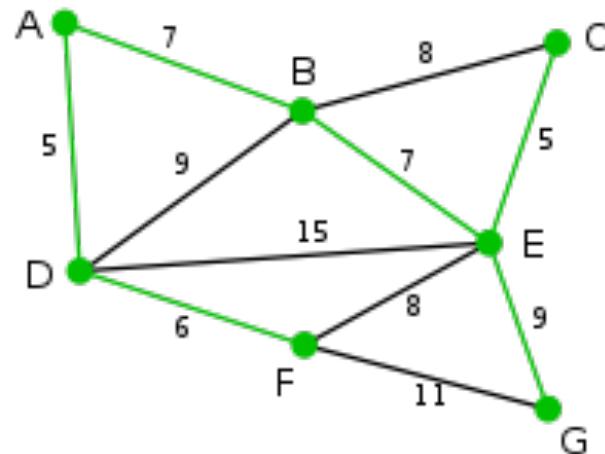
X	Cạnh (u,v)	N \ X	Mô tả
{A,B,D,E,F}	(B,C) = 8 (D,B) = 9 chu trình (D,E) = 15 chu trình (E,C) = 5 (E,G) = 9 (F,E) = 8 chu trình (F,G) = 11	{C,G}	Ở bước này ta chọn giữa C và G . C có khoảng cách tới E bằng 5 , và G có khoảng cách tới E bằng 9 . Chọn C và cạnh EC .

THUẬT TOÁN PRIM

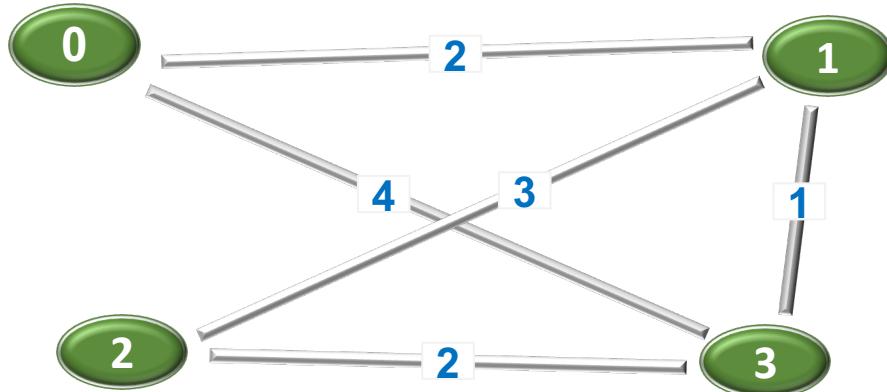


X	Cạnh (u,v)	$N \setminus X$	Mô tả
$\{A,B,C,D,E,F\}$	$(B,C) = 8$ chu trình $(D,B) = 9$ chu trình $(D,E) = 15$ chu trình $(E,G) = 9$ $(F,E) = 8$ chu trình $(F,G) = 11$	{G}	Đỉnh G là đỉnh còn lại duy nhất. Nó có khoảng cách tới F bằng 11 , và khoảng cách tới E bằng 9 . E ở gần hơn nên chọn đỉnh G và cạnh EG .

THUẬT TOÁN PRIM



X	Cạnh (u,v)	N \ X	Mô tả
{A,B,C,D,E,F, G}	(B,C) = 8 chu trình (D,B) = 9 chu trình (D,E) = 15 chu trình (F,E) = 8 chu trình (F,G) = 11 chu trình	{}	Hiện giờ tất cả các đỉnh đã nằm trong cây và cây bao trùm nhỏ nhất được tô màu xanh lá cây. Tổng trọng số của cây là 39 .



X	Cạnh (u,v)	$N \setminus X$



TUẦN 5

CÁC THUẬT TOÁN TÌM ĐƯỜNG ĐI NGẮN NHẤT

GIỚI THIỆU

- Trường Đại học Công nghệ TP.HCM - HUTECH
- Hutech, 36 Ung Văn Khiêm, Phường 25, B
- Hutech - Đại học Công Nghệ Tp HCM - Kh
- HUTECH University - E Campus (SHTP), 1
- Đại học HUTECH cơ sở R, Khu Công Nghệ
- Thêm điểm đến

Tùy chọn

 Gửi đường đi đến điện thoại của bạn

 qua Xô Viết Nghệ Tĩnh

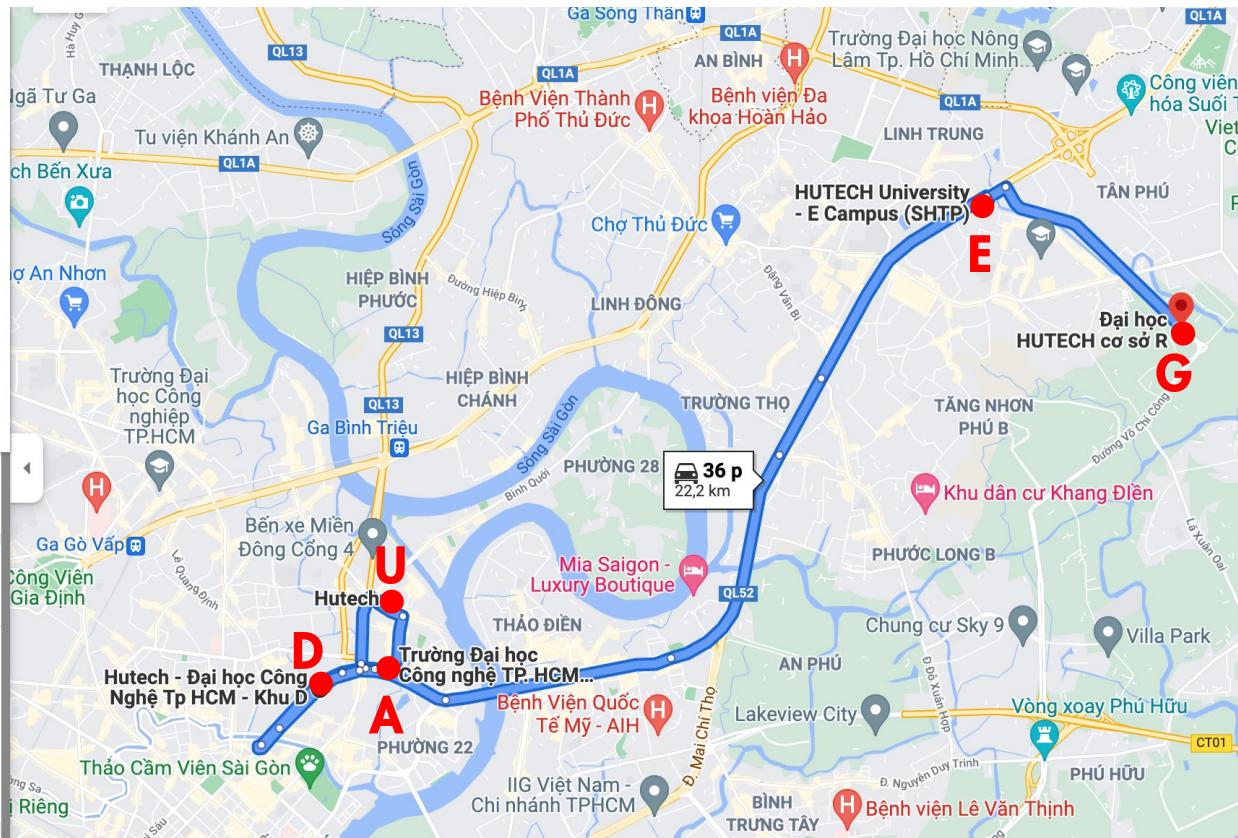
36 p ko có giao thông

 Tuyến này có thu phí.

[Chi tiết](#)

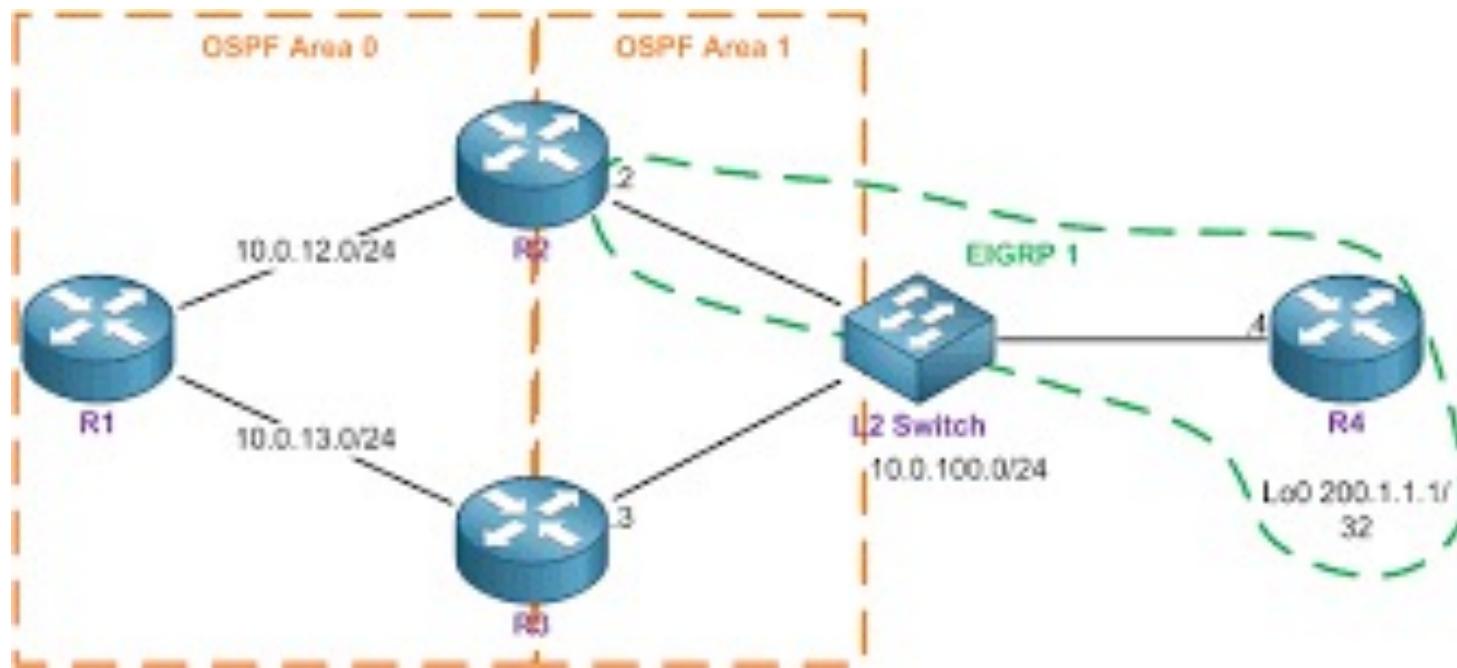
36 p

22,2 km



Bài toán tìm đường đi ngắn nhất đến một địa điểm trên bản đồ.

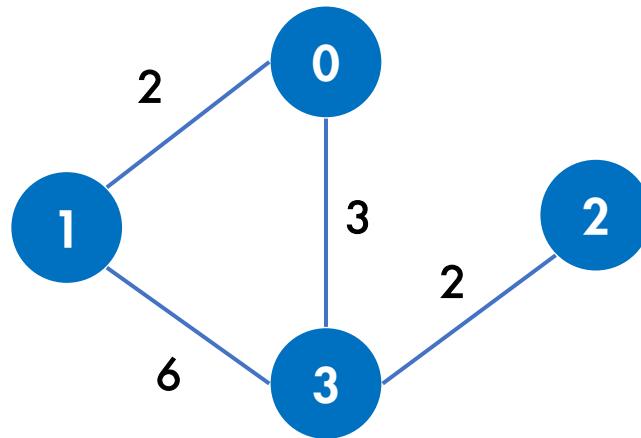
GIỚI THIỆU



Bài toán định tuyến đường đi của một hệ thống mạng

THUẬT TOÁN TÌM ĐƯỜNG ĐI NGẮN NHẤT

Cho đơn đồ thị có hướng $G = (N, E)$ với hàm trọng số $w: E \rightarrow R$ ($w(e)$ được gọi là độ dài hay trọng số của cạnh e)

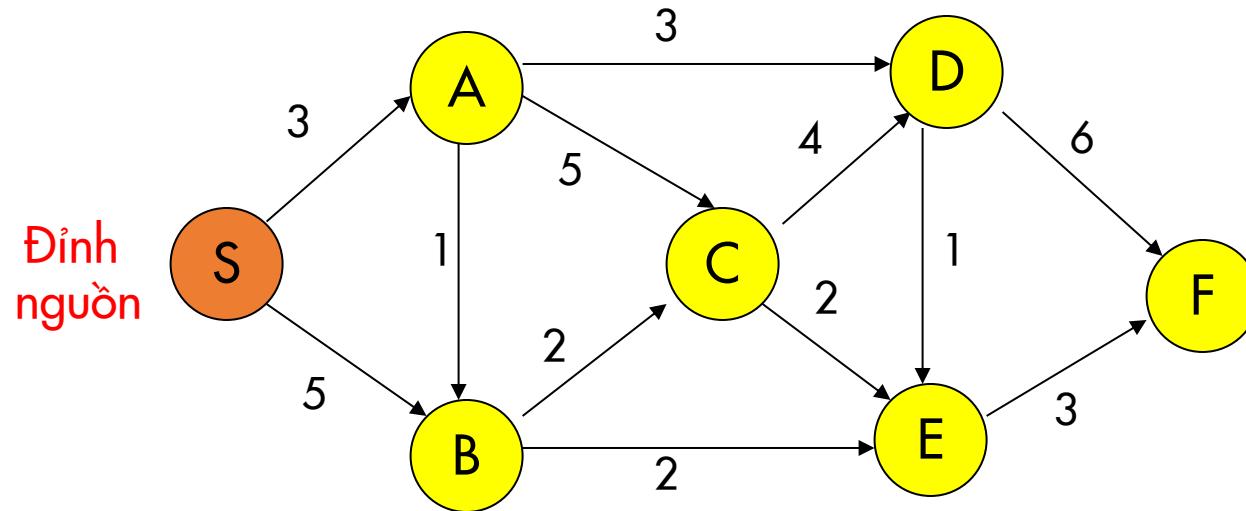


Đường đi ngắn nhất từ đỉnh u đến đỉnh v là đường đi có độ dài ngắn nhất trong số các đường đi nối u với v .

Độ dài của đường đi ngắn nhất từ u đến v còn được gọi là **khoảng cách từ u tới v** và ký hiệu là $\delta(u,v)$.

VÍ DỤ

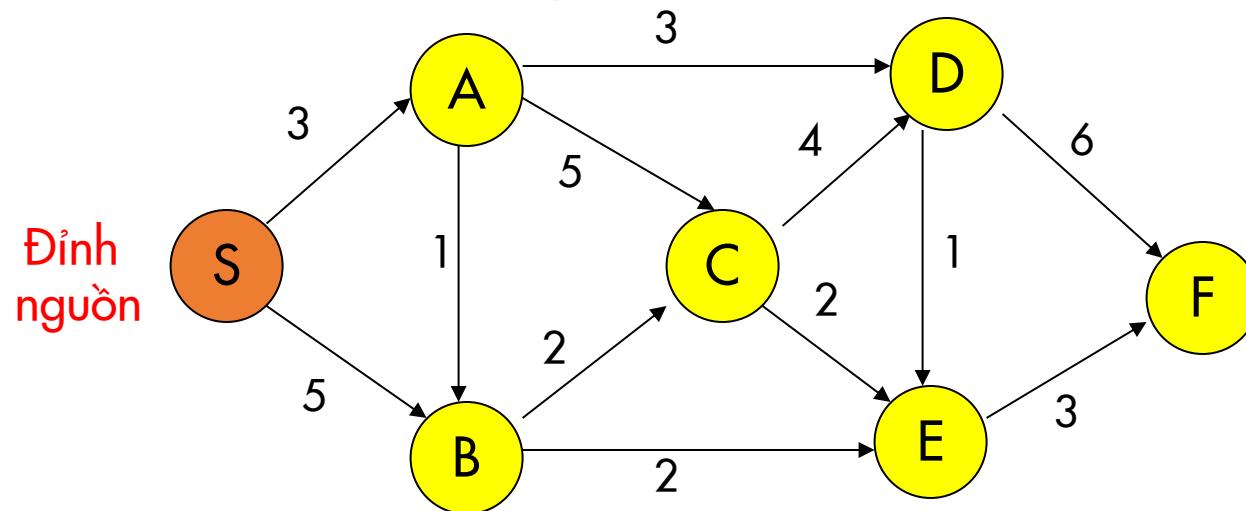
Cho đồ thị có trọng số $G = (N, E)$, và đỉnh nguồn $S \in N$, hãy tìm đường đi ngắn nhất từ S đến mỗi đỉnh còn lại.



Đỉnh	S	A	B	C	D	E	F
Trọng số	0						
Đường đi	S						

VÍ DỤ

Cho đồ thị có trọng số $G = (N, E)$, và đỉnh nguồn $S \in N$, hãy tìm đường đi ngắn nhất từ S đến mỗi đỉnh còn lại.



Đỉnh	S	A	B	C	D	E	F
Trọng số	0	3	4	6	6	6	9
Đường đi	S	S,A	S,A,B	S,A,B,C	S,A,D	S,A,B,E	S,A,B,E,F

THUẬT TOÁN DIJKSTRA

Cho $G=(N,E)$ là một đồ thị có trọng số không âm gồm n đỉnh.

Thuật toán Dijkstra dùng để tìm đường đi ngắn nhất giữa 2 đỉnh S(Start) và F(Finish) trong đồ thị G như sau:

Ta sử dụng 1 mảng 1 chiều LuuVet dùng để lưu vết đường đi từ $S \rightarrow F$, 1 mảng 1 chiều khác với tên là ChuaXet dùng để đánh dấu đỉnh nào trong đồ thị đã xét rồi, đỉnh nào chưa xét trong quá trình tìm đường đi từ $S \rightarrow F$, 1 mảng DoDaiDuongDiToi để lưu lại độ dài nhỏ nhất trong quá trình tìm đường đi từ $S \rightarrow F$.

- ChuaXet[MAX]: đánh dấu đỉnh nào trong đồ thị đã xét rồi, đỉnh nào chưa xét.
- LuuVet[MAX]: lưu đỉnh liền trước nó trên đường đi.
- DoDaiDuongDiToi[MAX]: Lưu độ dài từ đỉnh đầu i đến các đỉnh trong đồ thị.



THUẬT TOÁN DIJKSTRA

Các bước thi hành thuật toán Dijkstra như sau:

Bước 1: Khởi tạo:

- $ChuaXet[i] = 0; \forall i \in N$
- $LuuVet[i] = -1; \forall i \in N$
- $DoDaiDuongDiToi[S] = 0$
- $DoDaiDuongDiToi[i] = +\infty, \forall i \in N \setminus \{S\}$

Bước 2: Nếu $ChuaXet[F] == 1$ (tức đã xét tới F) thì dừng thuật toán và giá trị $DoDaiDuongDiToi[F]$ là độ dài đường đi ngắn nhất từ S → F và $LuuVet[F]$ lưu đỉnh nằm ngay trước F trên đường đi đến F.

Bước 3: Chọn đỉnh v ∈ N sao cho $ChuaXet[v]=0$ và $DoDaiDuongDiToi[v]$ nhỏ nhất. Khi đó gán $ChuaXet[v] = 1$.

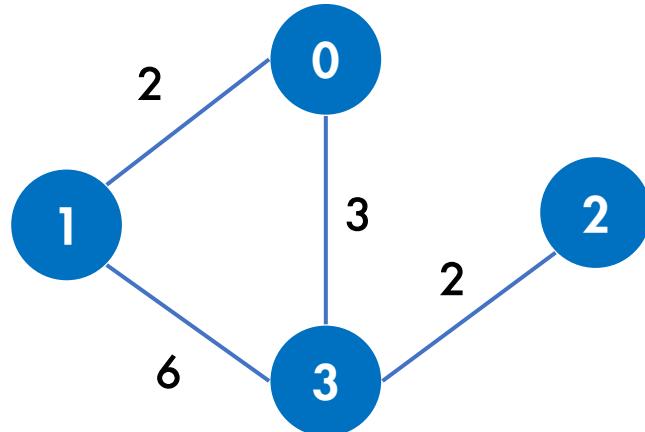
Bước 4: ∀k ∈ N mà $ChuaXet[k] == 0$ và có cạnh nối từ v đến k:

Nếu $DoDaiDuongDiToi[k] > DoDaiDuongDiToi[v] + e(v,k)$ thì
 $DoDaiDuongDiToi[k] = DoDaiDuongDiToi[v] + e(v,k);$
 $LuuVet[k] = v;$

Trở về bước 2

THUẬT TOÁN DIJKSTRA

Giả sử ta có đồ thị $G=(N, E)$ gồm có 4 đỉnh như sau. Thi hành thuật toán Dijkstra, Tìm đường đi ngắn nhất từ đỉnh 1 đến đỉnh 2



Bước 0: Khởi tạo: Khởi tạo đỉnh 1 với độ dài min hiện tại là 0, và nhãn đỉnh trước là -1. Các đỉnh còn lại đều được gán độ dài min là $+\infty$. Ta có bảng mô tả sau:

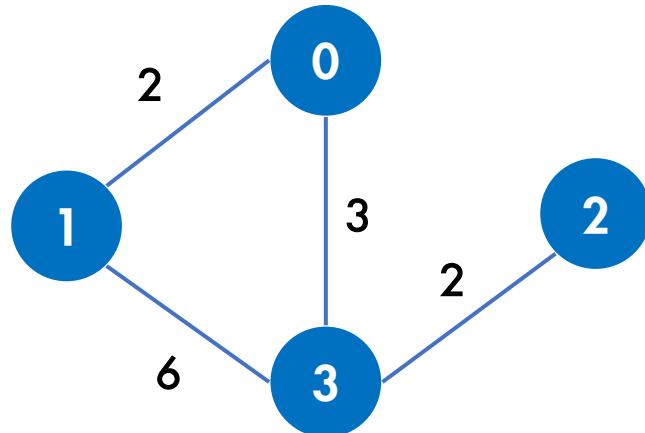
Mảng\Đỉnh	0	1	2	3
ChuaXet[i]	0	0	0	0
LuuVet[i]	-1	-1	-1	-1
DoDaiDuongDiToi[i]	$+\infty$	0	$+\infty$	$+\infty$

THUẬT TOÁN DIJKSTRA

Bước 0.1: Đỉnh 2 vẫn có giá trị ChuaXet[2] = 0 nên ta sang bước kế tiếp

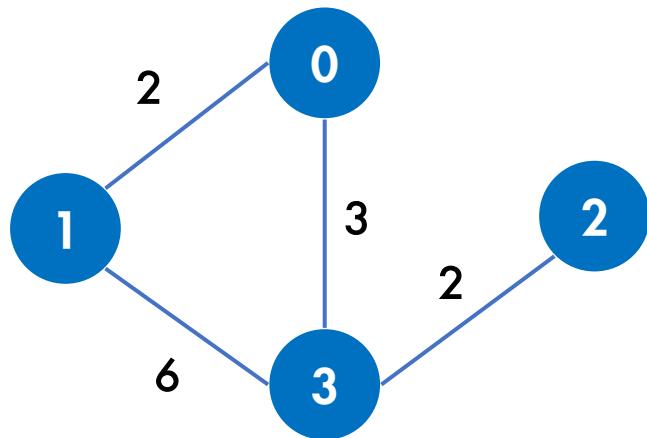
Bước 0.2: Chọn đỉnh có độ dài nó nhỏ nhất (trong mảng Độ dài) mà đỉnh giá trị chưa xét của nó là 0. Ở đây trong các đỉnh chưa xét đến là 0, 1, 2, 3.

Đỉnh 1 có độ dài nhỏ nhất là 0. Ta xét giá trị ChuaXet[1] = 1



Mảng\Đỉnh	0	1	2	3
ChuaXet[i]	0	1	0	0
LuuVet[i]	-1	-1	-1	-1
DoDaiDuongDiToi[i]	$+\infty$	0	$+\infty$	$+\infty$

THUẬT TOÁN DIJKSTRA



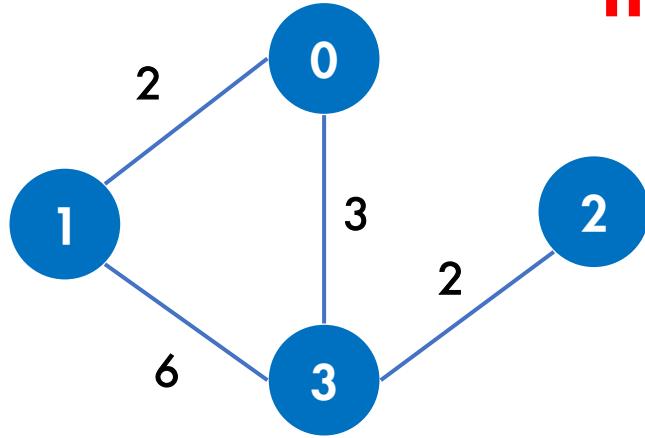
Buoc\ChuaXet[i]	0	1	2	3
0	0	0	0	0
1	0	1	0	0

Bước 1: Từ đỉnh 1 ta có đường đi đến đỉnh 0 và đỉnh 3.
Chi phí từ đỉnh 1 đến các đỉnh 0, 3 đều mang $+\infty$ nên ta
sẽ cập nhật lại như sau:

Buoc\DoDaiDuongDiToi[i]	0	1	2	3
0	$+\infty$	0	$+\infty$	$+\infty$
1	2	0	$+\infty$	6

Buoc\LuuVet[i]	0	1	2	3
0	-1	-1	-1	-1
1	1	-1	-1	1

THUẬT TOÁN DIJKSTRA



Bước 2: Đỉnh 2 vẫn có giá trị ChuaXet[2] = 0 nên ta sang bước kế tiếp. Chọn đỉnh có độ dài nó nhỏ nhất. Ở đây là đỉnh 0, ta xét giá trị ChuaXet[0] = 1

Buoc\ChuaXet[i]	0	1	2	3
0	0	0	0	0
1	0	1	0	0
2	1	1	0	0

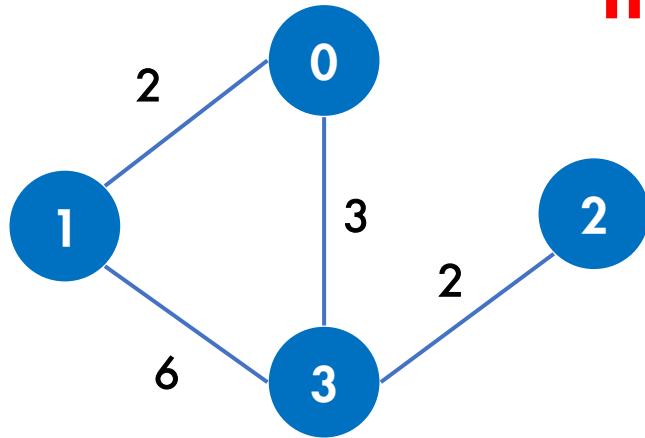
Bước 2.1: Tính độ dài từ đỉnh 0 vừa xét đến những còn lại có giá trị ChuaXet[i]=0.

Đỉnh 3 có chi phí mới $2+3=5 < 6$ nên ta cập nhật lại chi phí lại của đỉnh này đồng thời cập nhật lại lưu vết đỉnh 3.

Buoc\DoDaiDuongDiToi[i]	0	1	2	3
0	$+\infty$	0	$+\infty$	$+\infty$
1	2	0	$+\infty$	6
2	2	0	$+\infty$	5

Buoc\LuuVet[i]	0	1	2	3
0	-1	-1	-1	-1
1	1	-1	-1	1
2	1	-1	-1	0

THUẬT TOÁN DIJKSTRA



Bước 3: Đỉnh 2 vẫn có giá trị ChuaXet[2] = 0 nên ta sang bước kế tiếp. Chọn đỉnh có độ dài nó nhỏ nhất. Ở đây là đỉnh 3, ta xét giá trị ChuaXet[3] = 1.

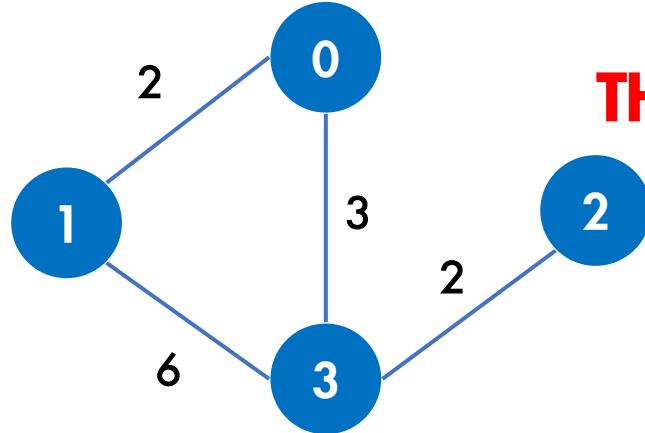
Buoc\ChuaXet[i]	0	1	2	3
0	0	0	0	0
1	0	1	0	0
2	1	1	0	0
3	1	1	0	1

Bước 2.1: Tính độ dài từ đỉnh 3 vừa xét đến những còn lại có giá trị ChuaXet[i]=0.

Đỉnh 2 có chi phí mới $5+2=7 < +\infty$ nên ta cập nhật lại chi phí lại của đỉnh này đồng thời cập nhật lại lưu vết đỉnh 2.

Buoc\DoDaiDuongDiToi[i]	0	1	2	3
0	$+\infty$	0	$+\infty$	$+\infty$
1	2	0	$+\infty$	6
2	2	0	$+\infty$	5
3	2	0	7	5

Buoc\LuuVet[i]	0	1	2	3
0	-1	-1	-1	-1
1	1	-1	-1	1
2	1	-1	-1	0
3	1	-1	3	0



THUẬT TOÁN DIJKSTRA

Bước 4: Đỉnh 2 vẫn có giá trị ChuaXet[2] = 0 nên ta sang bước kế tiếp. Chọn đỉnh có độ dài nó nhỏ nhất. Ở đây là đỉnh 2, ta xét giá trị ChuaXet[2] = 1

Buoc\LuuVet[i]	0	1	2	3
0	-1	-1	-1	-1
1	1	-1	-1	1
2	1	-1	-1	0
3	1	-1	3	0

Buoc\DoDaiDuongDiToi[i]	0	1	2	3
0	$+\infty$	0	$+\infty$	$+\infty$
1	2	0	$+\infty$	6
2	2	0	$+\infty$	5
3	2	0	7	5

Buoc\ChuaXet[i]	0	1	2	3
0	0	0	0	0
1	0	1	0	0
2	1	1	0	0
3	1	1	0	1
4	1	1	1	1

Bước 4.1: Lúc này đỉnh 2 đã được xét và chúng ta kết thúc thuật toán

Kết luận: Đường đi là: 1-0-3-2 với độ dài là 7.



Q&A

Cảm ơn

Thầy và các bạn đã chú ý lắng nghe

