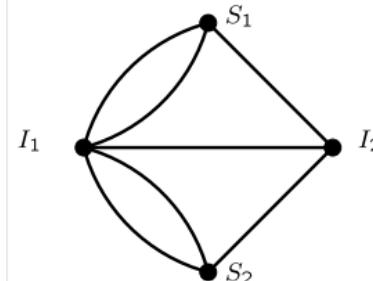
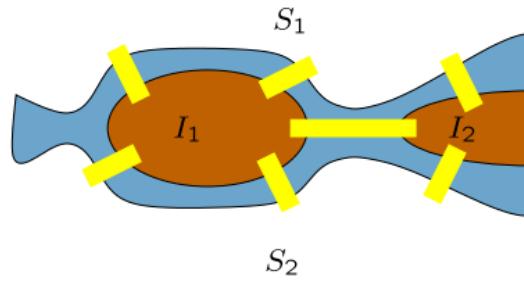


LÝ THUYẾT ĐỒ THỊ

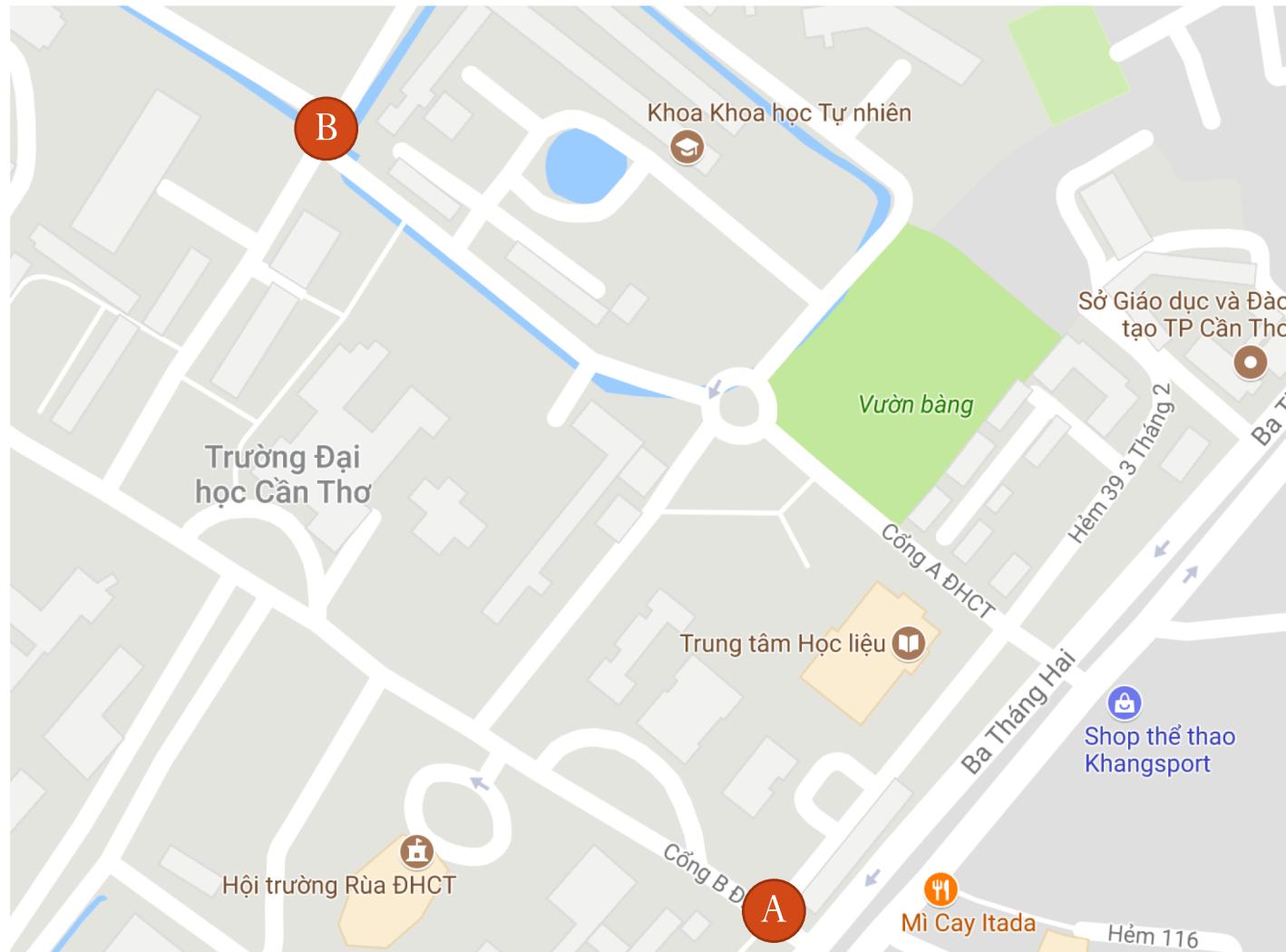
Tìm đường đi ngắn nhất

Phạm Nguyên Khang
BM. Khoa học máy tính, CNTT
pnkhang@cit.ctu.edu.vn



Cần Thơ, 8/2021

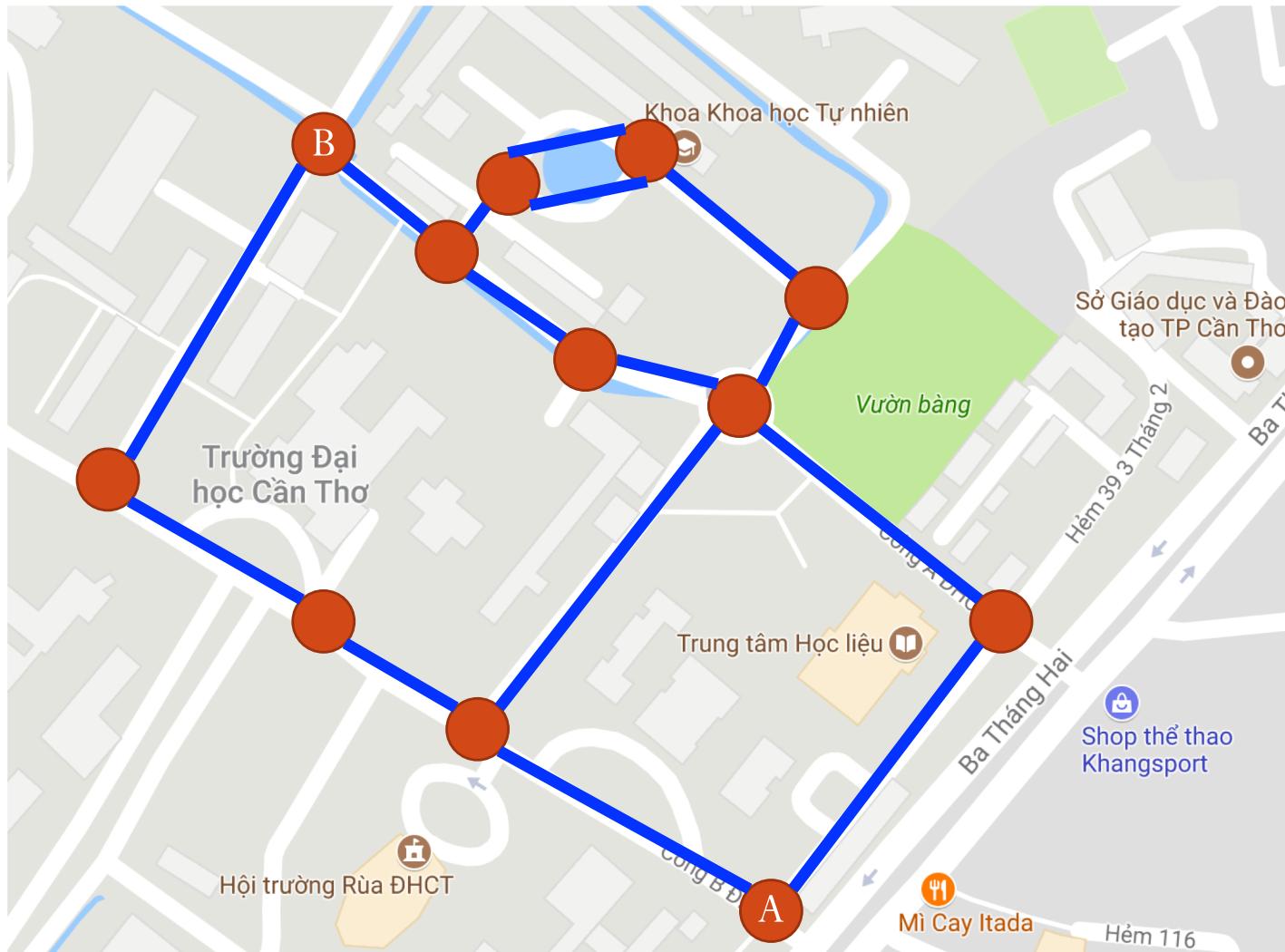
Từ bản đồ đến đồ thị



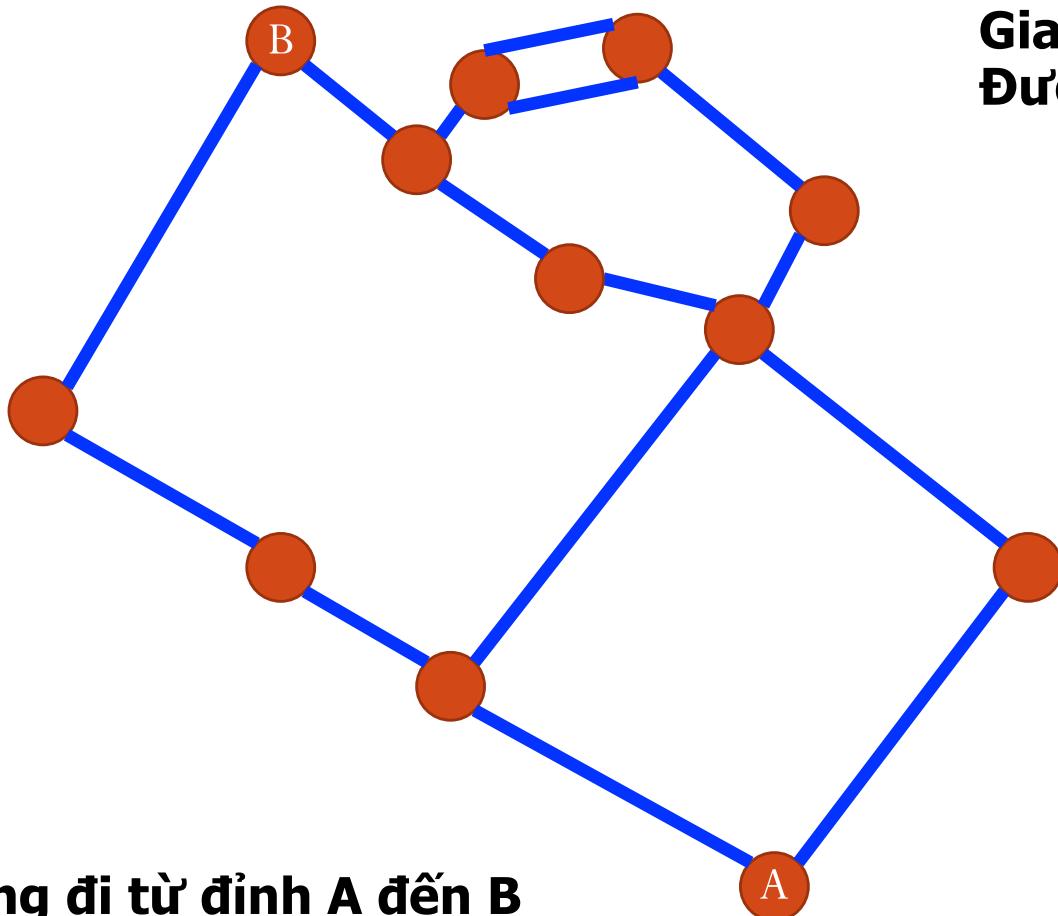
Từ bản đồ đến đồ thị



Từ bản đồ đến đồ thị



Từ bản đồ đến đồ thị



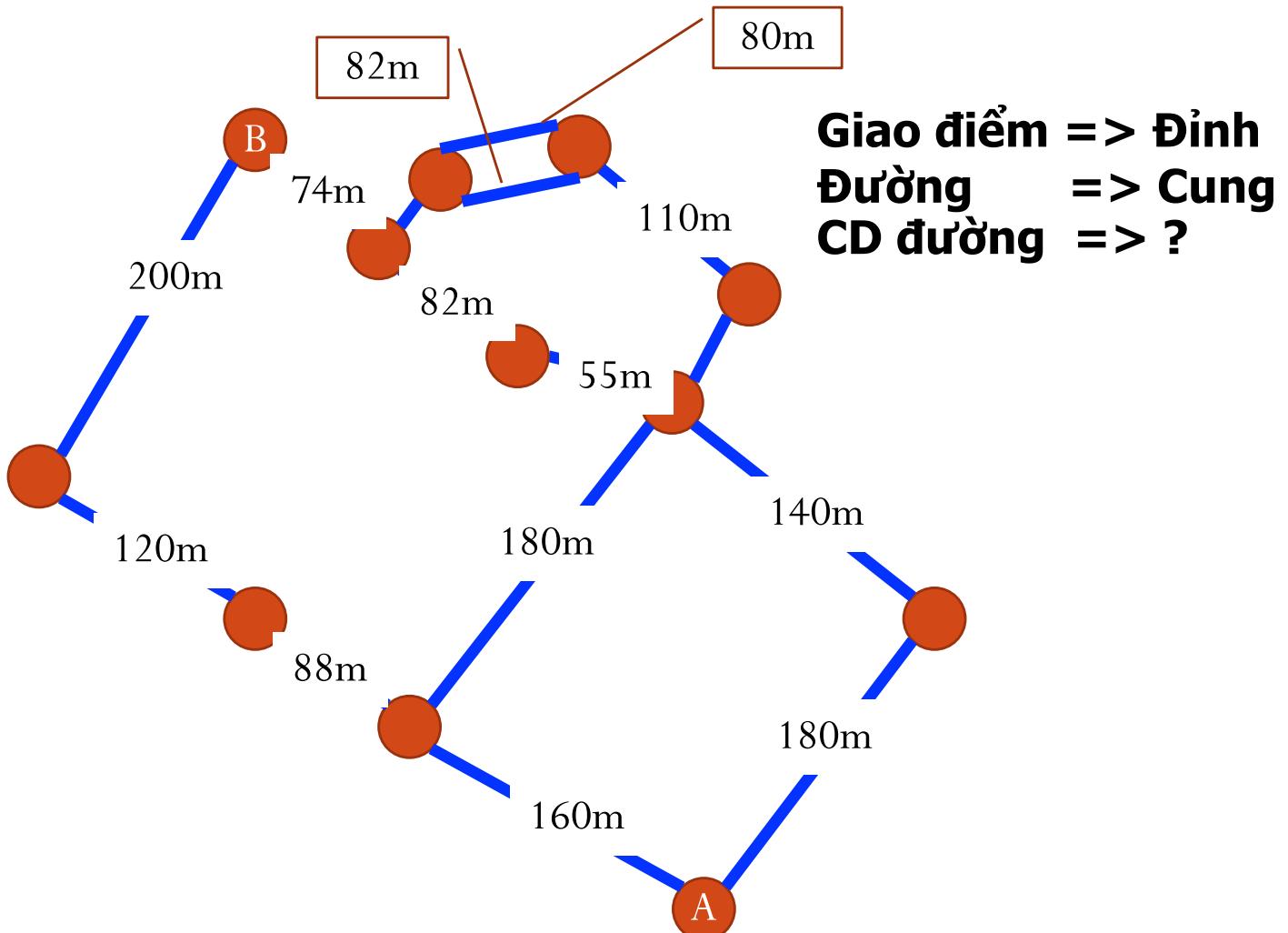
Giao điểm => Đỉnh
Đường => Cung

Tìm đường đi từ đỉnh A đến B

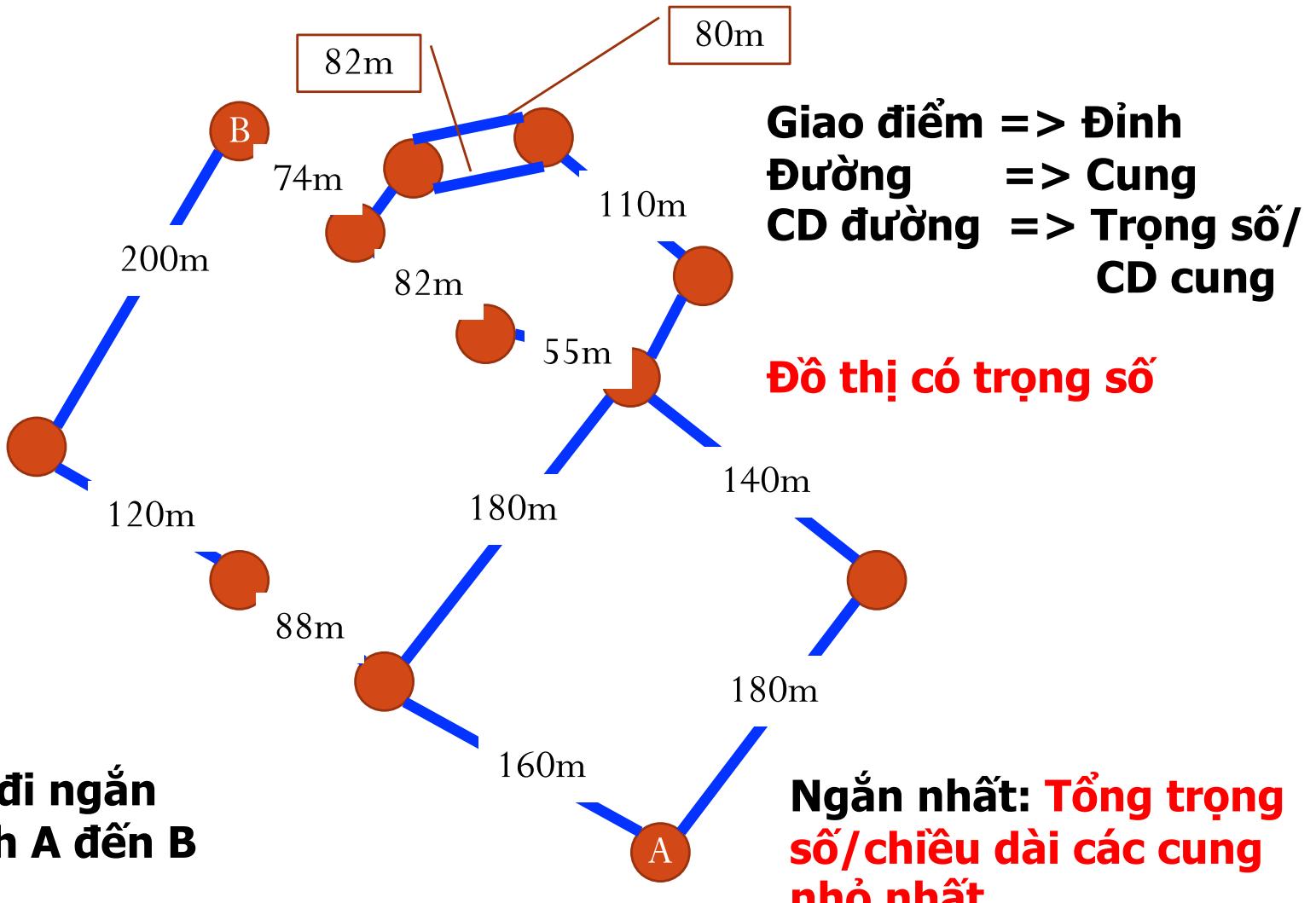
Từ bản đồ đến đồ thị



Từ bản đồ đến đồ thị

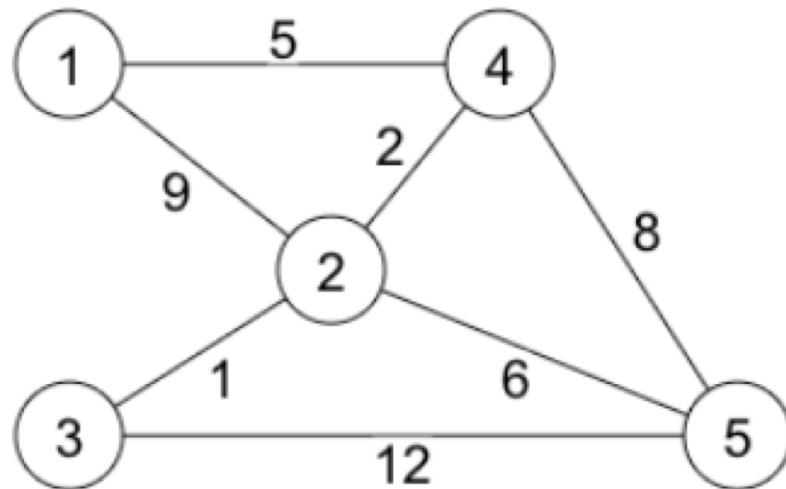


Từ bản đồ đến đồ thị



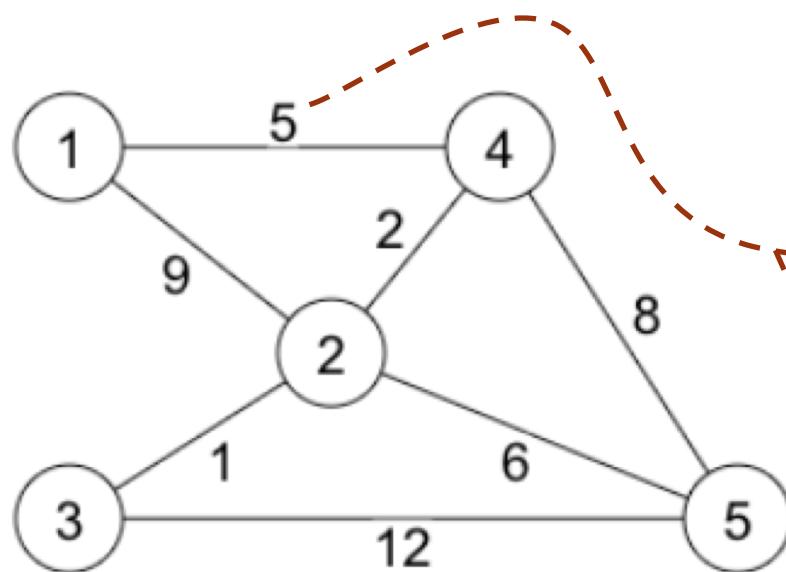
Biểu diễn đồ thị có trọng số

- Có thể mở rộng các phương pháp biểu diễn đồ thị trước đây để biểu diễn đồ thị có trọng số
 - Ma trận kề (đỉnh – đỉnh) => **ma trận trọng số**
 - Nếu có (u, v) thì $W[u][v] = \text{trọng số của cung } (u, v)$
 - Nếu không có (u, v) thì $W[u][v] = \text{NO_EDGE}$ (vd: -1)



Biểu diễn đồ thị có trọng số

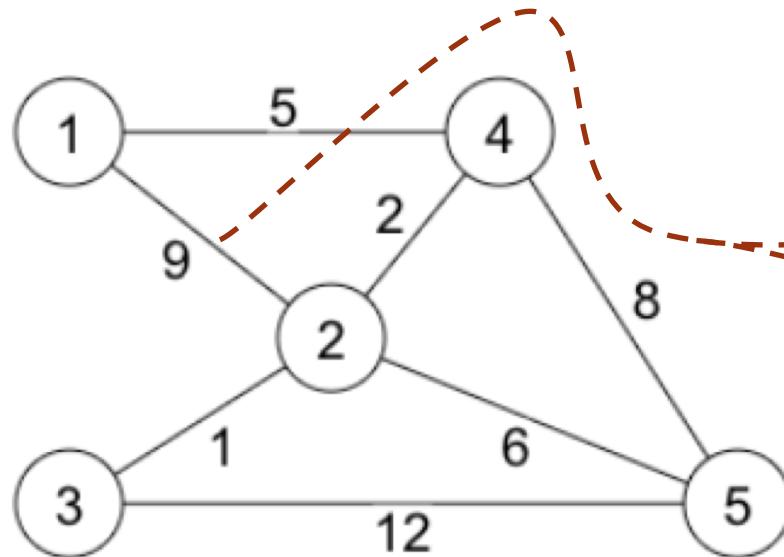
- Có thể mở rộng các phương pháp biểu diễn đồ thị trước đây để biểu diễn đồ thị có trọng số
 - Ma trận kề (đỉnh – đỉnh) => **ma trận trọng số**
 - Nếu có (u, v) thì $W[u][v] = \text{trọng số của cung } (u, v)$
 - Nếu không có (u, v) thì $W[u][v] = \text{NO_EDGE}$ (vd: -1)



	1	2	3	4	5
1					5
2					
3					
4				5	
5				5	

Biểu diễn đồ thị có trọng số

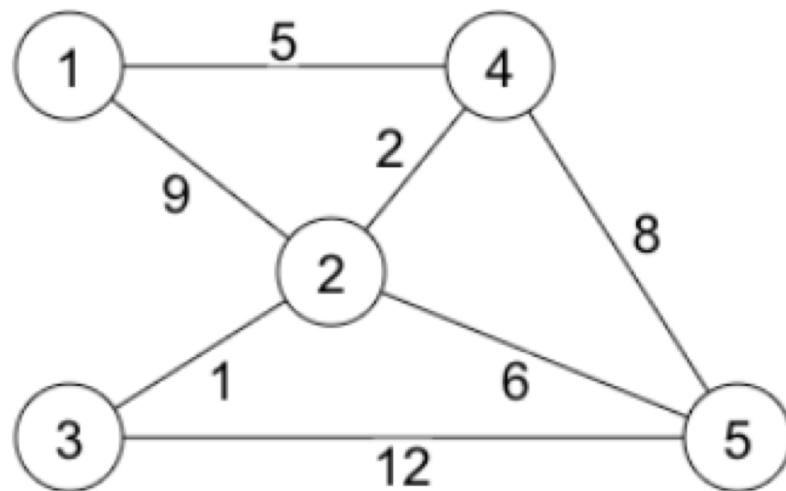
- Có thể mở rộng các phương pháp biểu diễn đồ thị trước đây để biểu diễn đồ thị có trọng số
 - Ma trận kề (đỉnh – đỉnh) => **ma trận trọng số**
 - Nếu có (u, v) thì $W[u][v] = \text{trọng số của cung } (u, v)$
 - Nếu không có (u, v) thì $W[u][v] = \text{NO_EDGE}$ (vd: -1)



	1	2	3	4	5
1					5
2				9	
3					
4	9				
5					

Biểu diễn đồ thị có trọng số

- Có thể mở rộng các phương pháp biểu diễn đồ thị trước đây để biểu diễn đồ thị có trọng số
 - Ma trận kề (đỉnh – đỉnh) => **ma trận trọng số**
 - Nếu có (u, v) thì $W[u][v] = \text{trọng số của cung } (u, v)$
 - Nếu không có (u, v) thì $W[u][v] = \text{NO_EDGE}$ (vd: -1)



Bài tập trên ELSE

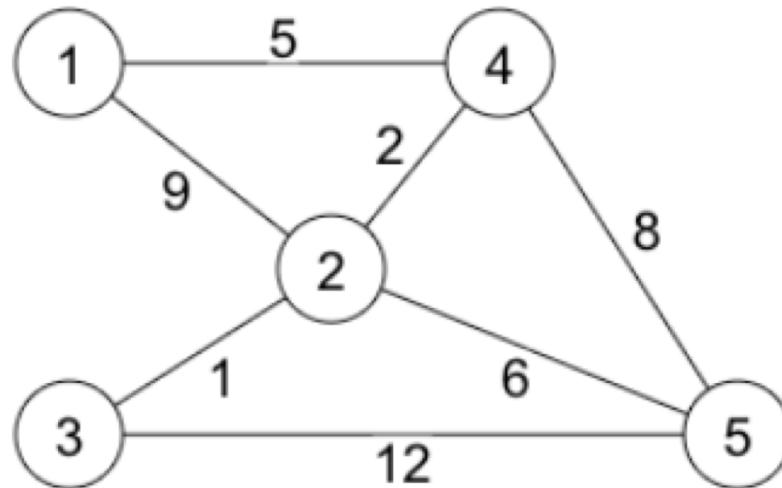
12



Biểu diễn đồ thị có trọng số bằng phương pháp ma trận trọng số (VÔ HƯỚNG)

Biểu diễn đồ thị có trọng số

- Có thể mở rộng các phương pháp biểu diễn đồ thị trước đây để biểu diễn đồ thị có trọng số
 - Ma trận kề (đỉnh – đỉnh) => **ma trận trọng số**
 - Nếu có (u, v) thì $W[u][v] = \text{trọng số của cung } (u, v)$
 - Nếu không có (u, v) thì $W[u][v] = \text{NO_EDGE}$ (vd: -1)



	1	2	3	4	5
1		9		5	
2	9		1	2	6
3		1			12
4	5	2			8
5		6	12	8	

Biểu diễn đồ thị có trọng số

- Có thể mở rộng các phương pháp biểu diễn đồ thị trước đây để biểu diễn đồ thị có trọng số
 - Ma trận kề (đỉnh – đỉnh) => **ma trận trọng số**
 - Nếu có (u, v) thì **$W[u][v] = \text{trọng số của cung } (u, v)$**
 - Nếu không có (u, v) thì **$W[u][v] = NO_EDGE$** (vd: -1)

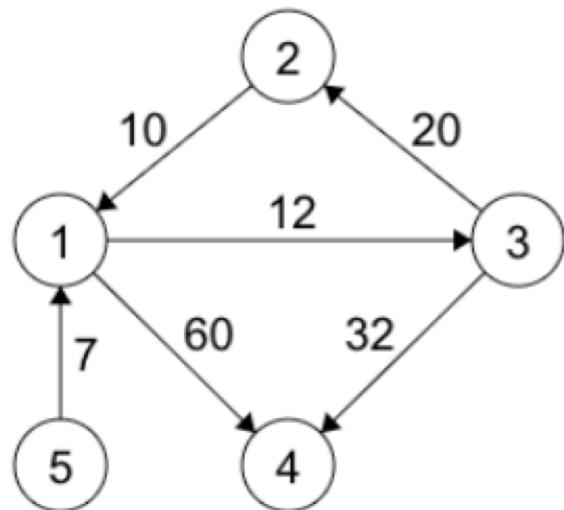
```
#define NO_EDGE -1
#define MAX_N 100

typedef struct {
    int n, m;
    int/double W[MAX_N][MAX_N];
} Graph;
```

	1	2	3	4	5
1		9		5	
2	9		1	2	6
3		1			12
4	5	2			8
5		6	12	8	

Biểu diễn đồ thị có trọng số

- Có thể mở rộng các phương pháp biểu diễn đồ thị trước đây để biểu diễn đồ thị có trọng số
 - Ma trận kề (đỉnh - đỉnh) => **ma trận trọng số**
 - Nếu có (u, v) thì $W[u][v] = \text{trọng số của cung } (u, v)$
 - Nếu không có (u, v) thì $W[u][v] = \text{NO_EDGE}$ (vd: -1)

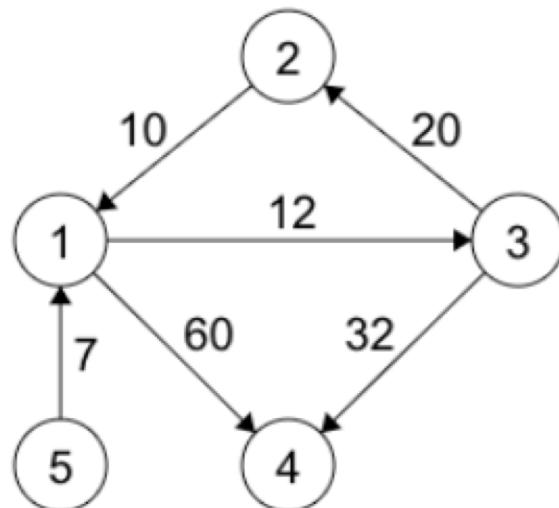


Bài tập trên ELSE



Biểu diễn đồ thị có trọng số

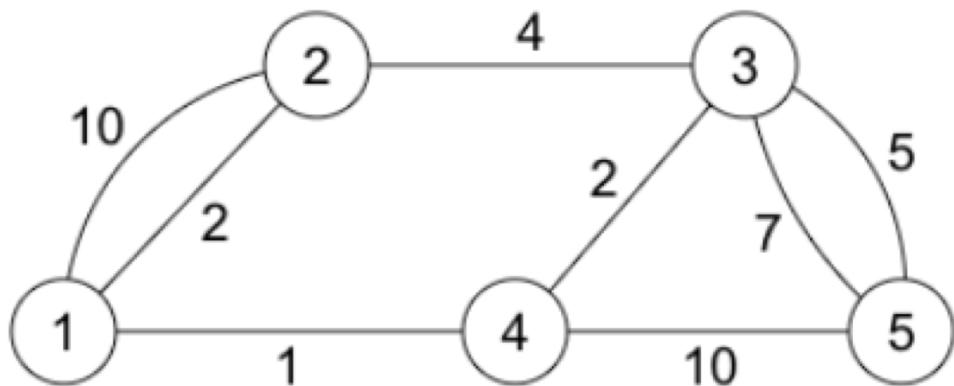
- Có thể mở rộng các phương pháp biểu diễn đồ thị trước đây để biểu diễn đồ thị có trọng số
 - Ma trận kề (đỉnh - đỉnh) => **ma trận trọng số**
 - Nếu có (u, v) thì $W[u][v] = \text{trọng số của cung } (u, v)$
 - Nếu không có (u, v) thì $W[u][v] = \text{NO_EDGE}$ (vd: -1)



	1	2	3	4	5
1			12	60	
2	10				
3		20		32	
4					
5	7				

Biểu diễn đồ thị có trọng số

- Có thể mở rộng các phương pháp biểu diễn đồ thị trước đây để biểu diễn đồ thị có trọng số
 - Ma trận kề (đỉnh - đỉnh) => **ma trận trọng số**
 - Nếu có (u, v) thì $W[u][v] = \text{trọng số của cung } (u, v)$
 - Nếu không có (u, v) thì $W[u][v] = \text{NO_EDGE}$ (vd: -1)

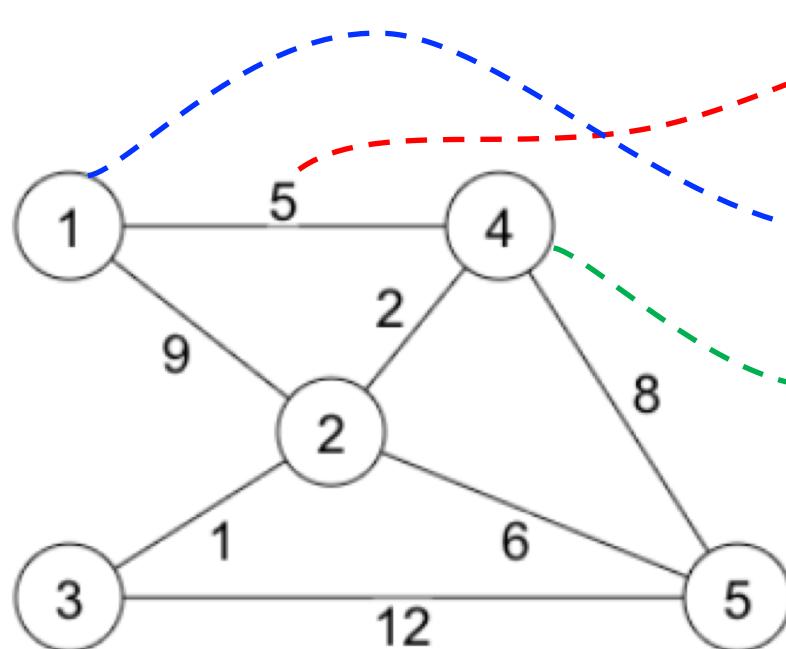


Đồ thị có đa cung

Có biểu diễn được không?

Biểu diễn đồ thị có trọng số

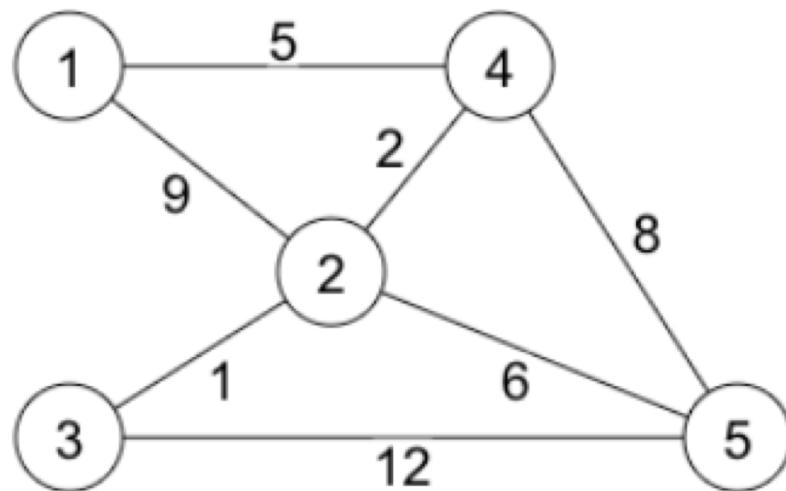
- Có thể mở rộng các phương pháp biểu diễn đồ thị trước đây để biểu diễn đồ thị có trọng số
 - Danh sách cung
 - Mỗi cung lưu trữ: đỉnh đầu, đỉnh cuối **và trọng số**



	u	v	w
1	1	4	5
2			
3			
4			
5			
6			
7			

Biểu diễn đồ thị có trọng số

- Có thể mở rộng các phương pháp biểu diễn đồ thị trước đây để biểu diễn đồ thị có trọng số
 - Danh sách cung
 - Mỗi cung lưu trữ: đỉnh đầu, đỉnh cuối **và trọng số**

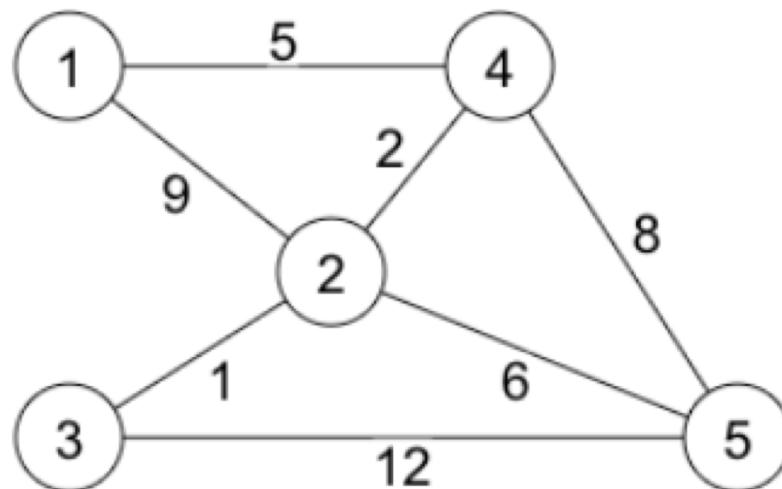


Bài tập trên ELSE



Biểu diễn đồ thị có trọng số

- Có thể mở rộng các phương pháp biểu diễn đồ thị trước đây để biểu diễn đồ thị có trọng số
 - Danh sách cung
 - Mỗi cung lưu trữ: đỉnh đầu, đỉnh cuối **và trọng số**



	u	v	w
1	1	4	5
2	1	2	9
3	2	3	1
4	2	4	2
5	2	5	6
6	3	5	12
7	4	5	8

Biểu diễn đồ thị có trọng số

- Có thể mở rộng các phương pháp biểu diễn đồ thị trước đây để biểu diễn đồ thị có trọng số
 - Danh sách cung
 - Mỗi cung lưu trữ: đỉnh đầu, đỉnh cuối **và trọng số**

```
#define NO_EDGE -1
#define MAX_N 100

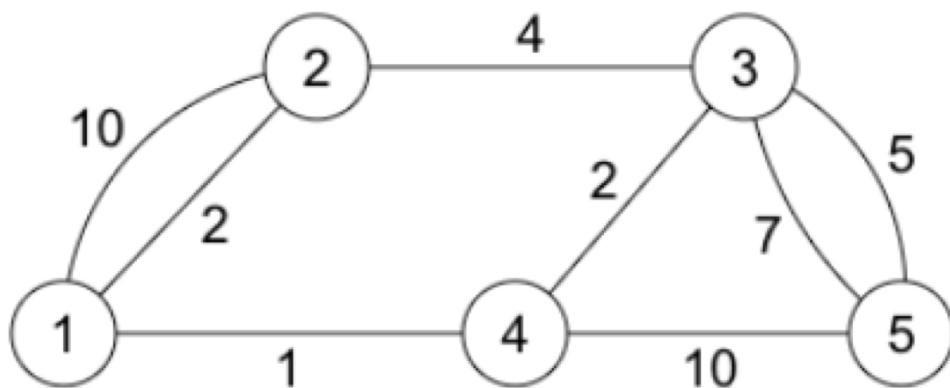
typedef struct {
    int u, v;
    int,double w;
} Edge;

Graph = danh sách Edge
```

	u	v	w
1	1	4	5
2			
3			
4			
5			
6			
7			

Biểu diễn đồ thị có trọng số

- Có thể mở rộng các phương pháp biểu diễn đồ thị trước đây để biểu diễn đồ thị có trọng số
 - Danh sách cung
 - Mỗi cung lưu trữ: đỉnh đầu, đỉnh cuối **và trọng số**

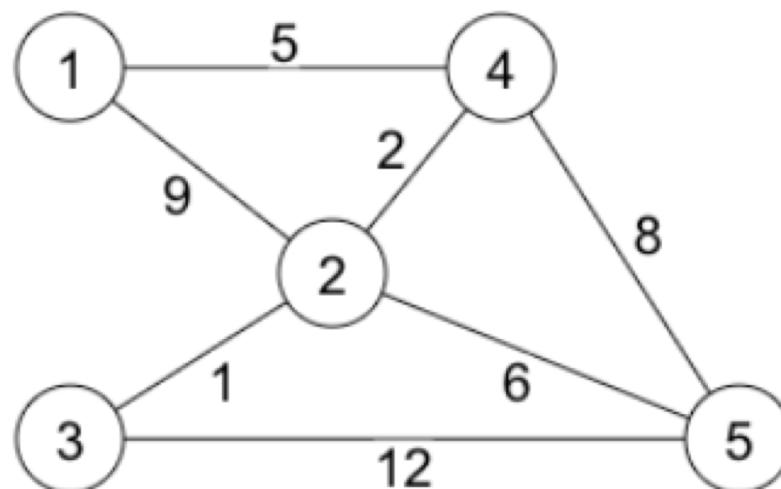


Đồ thị có đa cung

Có biểu diễn được không?

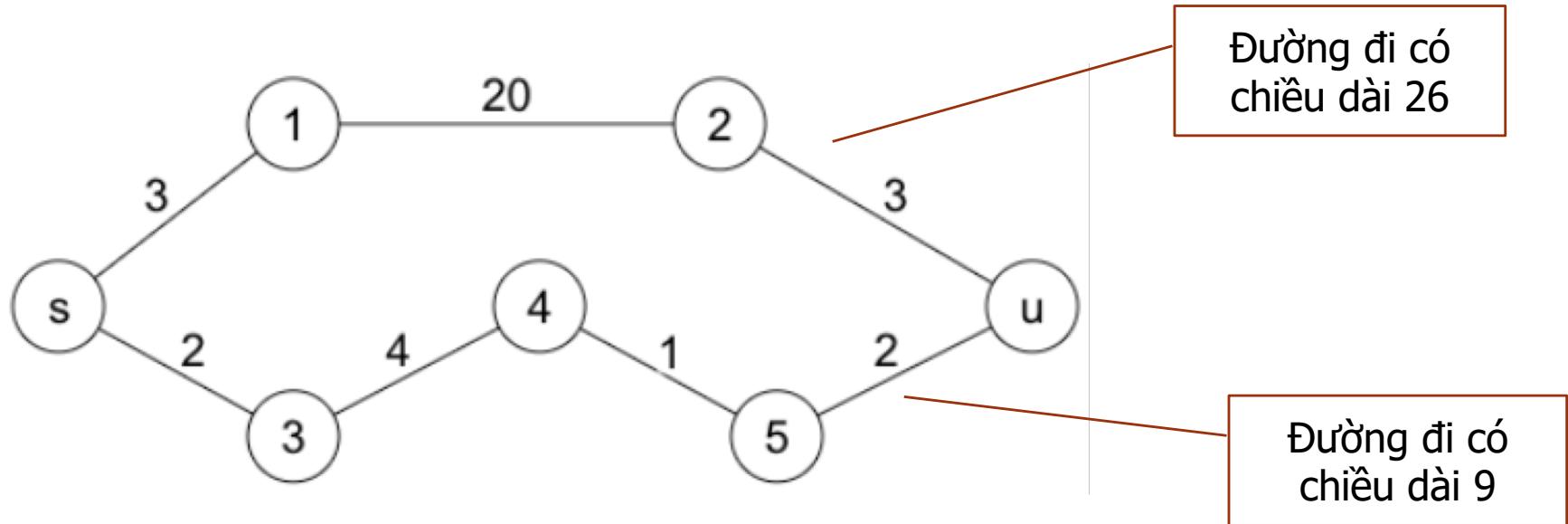
Biểu diễn đồ thị có trọng số

- Có thể mở rộng các phương pháp biểu diễn đồ thị trước đây để biểu diễn đồ thị có trọng số
 - Danh sách kê
 - Bài tập:
 - **Tìm cách mở rộng phương pháp danh sách kê để biểu diễn đồ thị có trọng số**



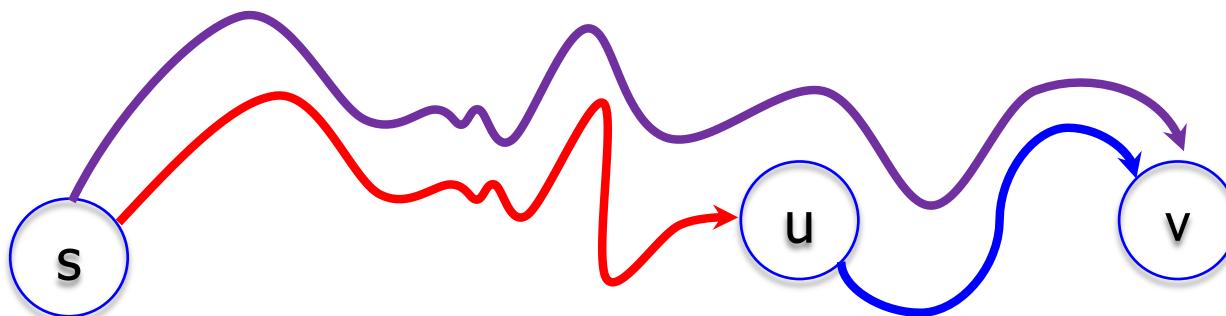
Bài toán đường đi ngắn nhất

- *Đường đi ngắn nhất* (shortest path) từ đỉnh u đến đỉnh v trong đồ thị có trọng số là gì ?
 - *Chiều dài/Chi phí* (cost) của đường đi (path) là tổng các trọng số của các cung trên đường đi.
 - *Đường đi ngắn nhất* (shortest path) là đường đi có chiều dài nhỏ nhất.



Đường đi ngắn nhất

- Một đoạn đường đi (sub-path) của đường đi ngắn nhất cũng là đường đi ngắn nhất.
 - Tính chất *cấu trúc con tối ưu* (optimal substructure)
=> có thể áp dụng kỹ thuật quy hoạch động để thiết kế các giải thuật tìm đường đi ngắn nhất.

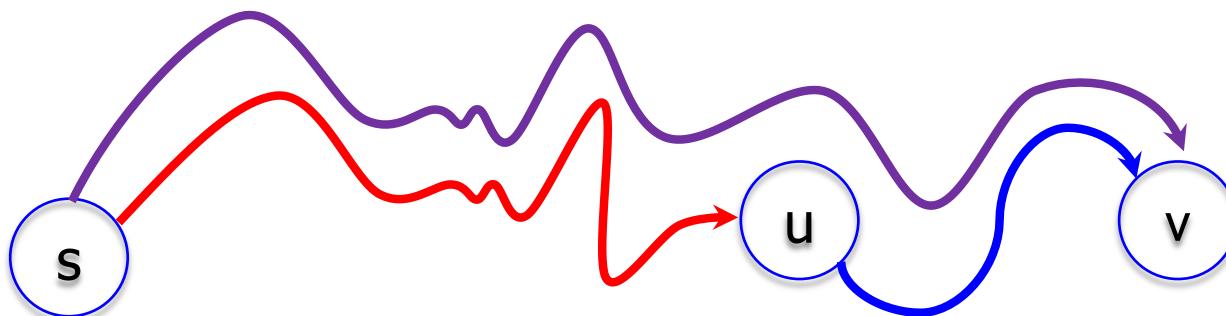


Nếu đường đi $[s, v]$ là **đường đi tối ưu** (từ s đến v) thì

- đoạn $[s, u]$ cũng là **đường đi tối ưu** (từ s đến u) và
- đoạn $[u, v]$ cũng là **đường đi tối ưu** (từ u đến v)

Đường đi ngắn nhất

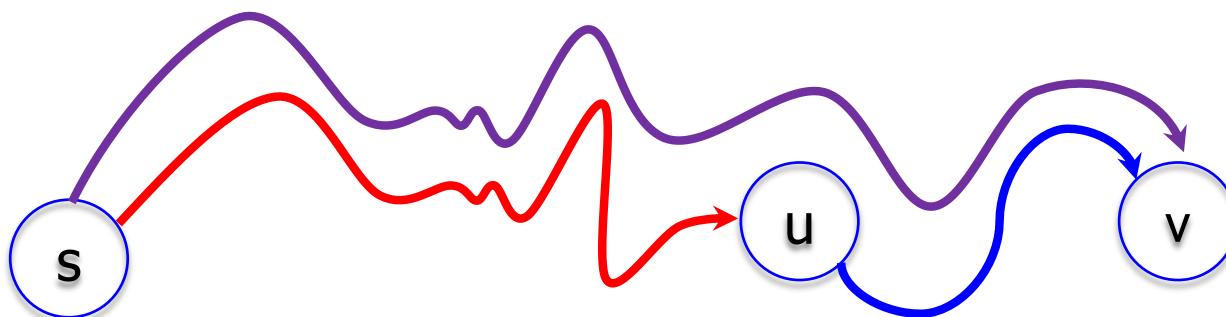
- Một đoạn đường đi (sub-path) của đường đi ngắn nhất cũng là đường đi ngắn nhất.
 - Tính chất *cấu trúc con tối ưu* (optimal substructure)
=> có thể áp dụng kỹ thuật quy hoạch động để thiết kế các giải thuật tìm đường đi ngắn nhất.



Vậy, để tìm đường đi tối ưu từ $s \rightarrow v$ ta cần tìm đường đi tối ưu từ $s \rightarrow u$ trước.

Đường đi ngắn nhất

- Một đoạn đường đi (sub-path) của đường đi ngắn nhất cũng là đường đi ngắn nhất.
 - Tính chất *cấu trúc con tối ưu* (optimal substructure)
=> có thể áp dụng kỹ thuật quy hoạch động để thiết kế các giải thuật tìm đường đi ngắn nhất.



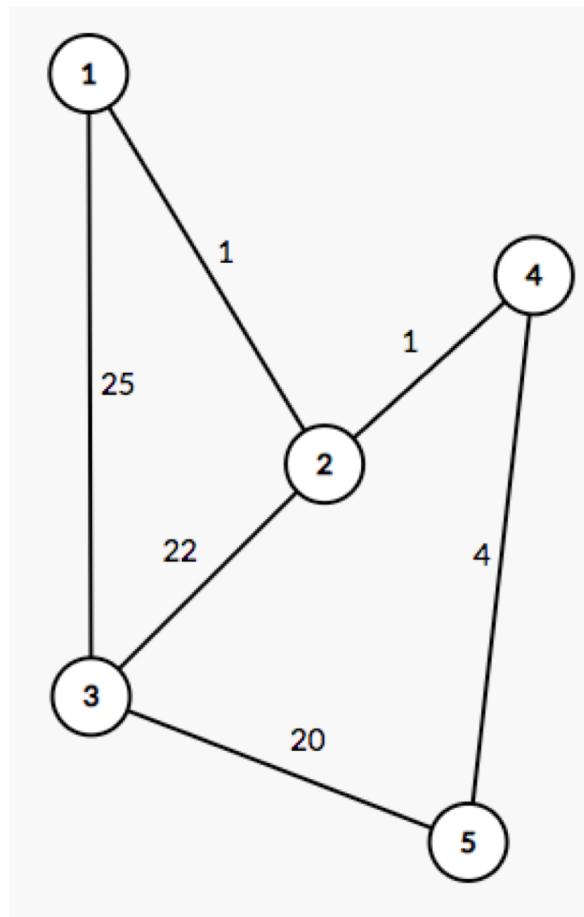
Tìm đường đi đến đỉnh gần trước (u) rồi mới tìm đường đi đến đỉnh xa hơn (v).

Thuật toán Moore – Dijkstra

- Tìm đường đi ngắn nhất từ *1 đỉnh đến các đỉnh khác* trên đồ thị có trọng số (single source shortest path problem – SSSP)
- Điều kiện áp dụng:
 - Đồ thị có trọng số không âm
 - Có hướng hoặc vô hướng đều được
- Ý tưởng chính:
 - Khởi tạo đường đi trực tiếp từ s đến các đỉnh
 - Lần lượt cập nhật lại đường đi nếu **tìm được đường đi mới tốt hơn đường đi cũ.**

Thuật toán Moore – Dijkstra

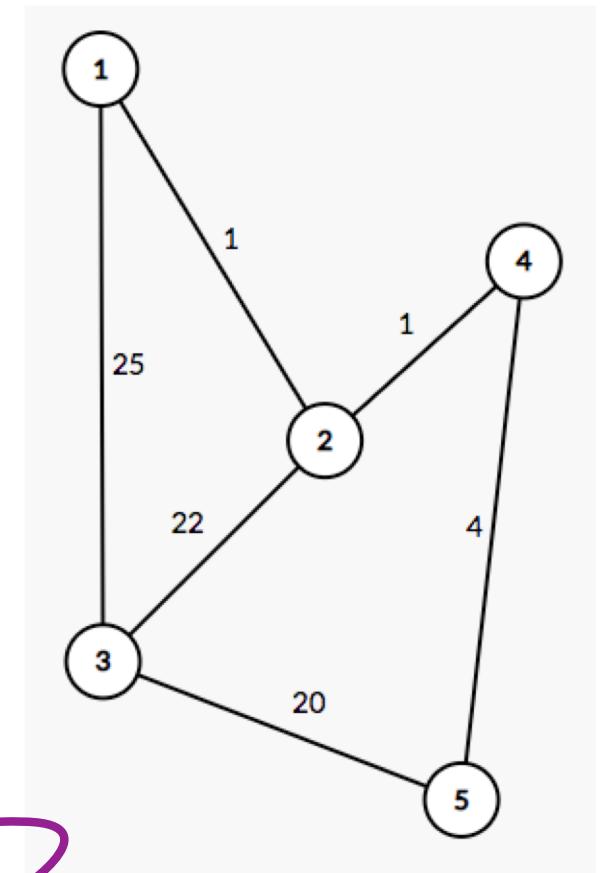
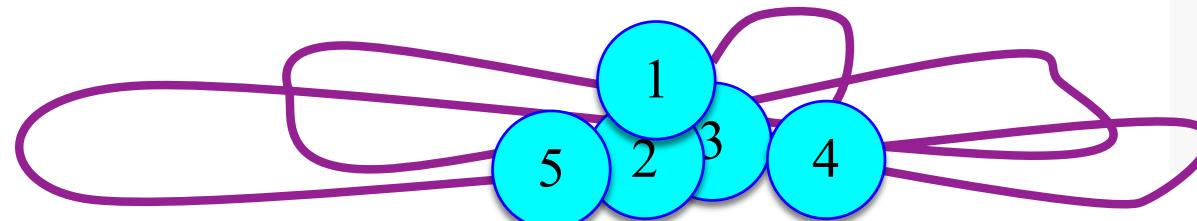
- Quan sát
 - Tìm đường đi ngắn nhất từ 1 đến các đỉnh khác



Thuật toán Moore-Dijkstra

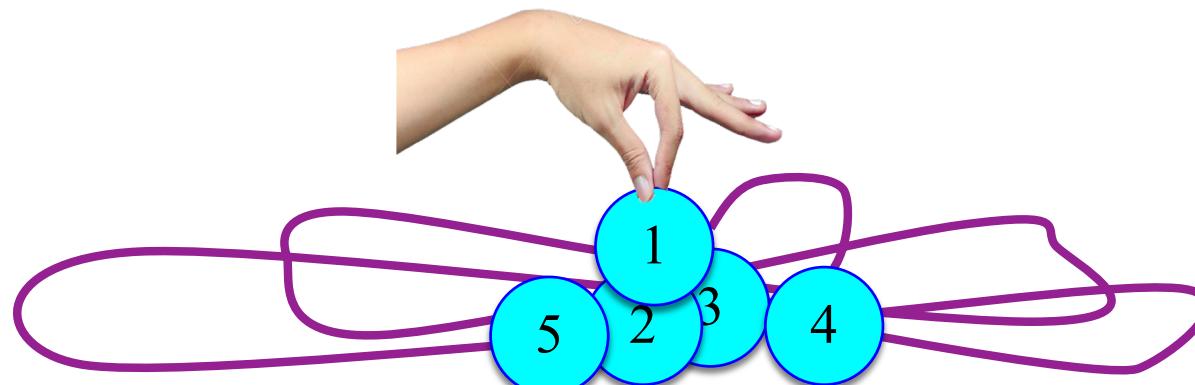
- Quan sát
 - Tìm đường đi ngắn nhất từ 1 đến các đỉnh khác

30



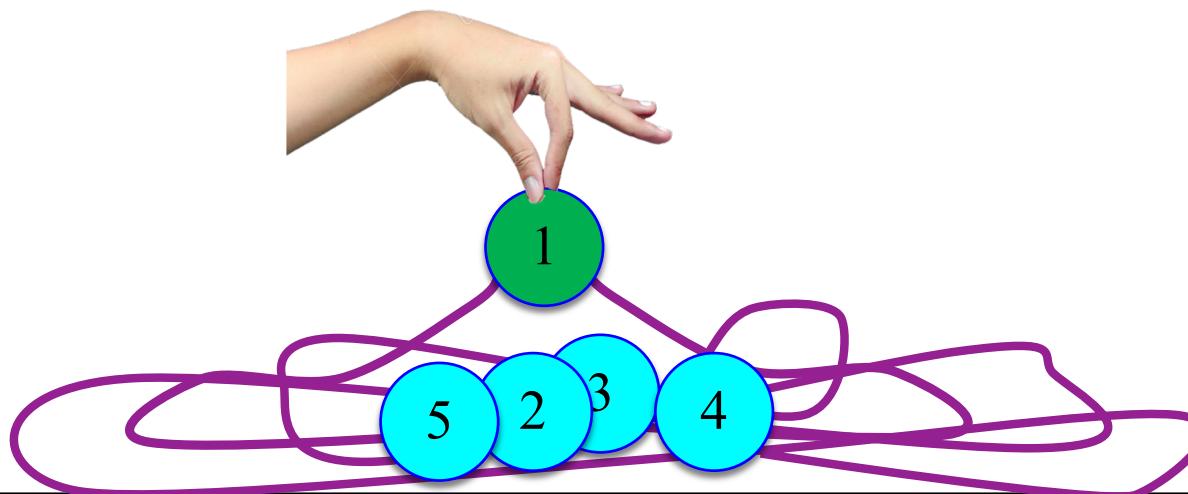
Thuật toán Moore – Dijkstra

- Quan sát
 - Tìm đường đi đến các đỉnh gần với 1 trước



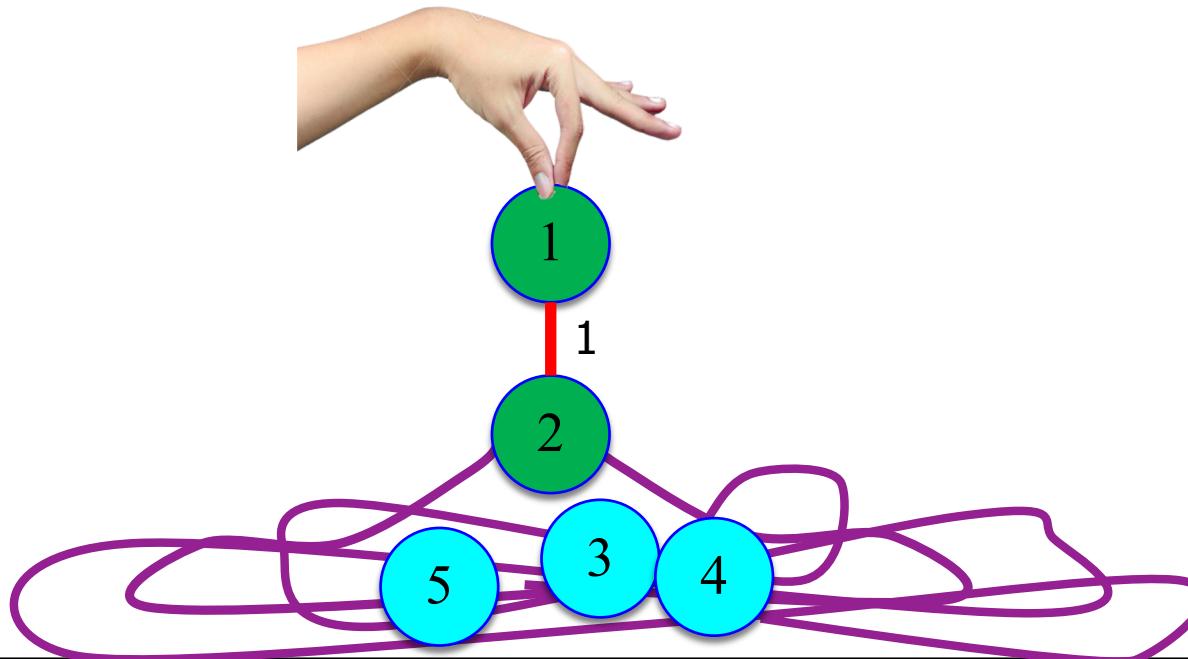
Thuật toán Moore – Dijkstra

- Quan sát
 - Gần với 1 nhất là chính nó



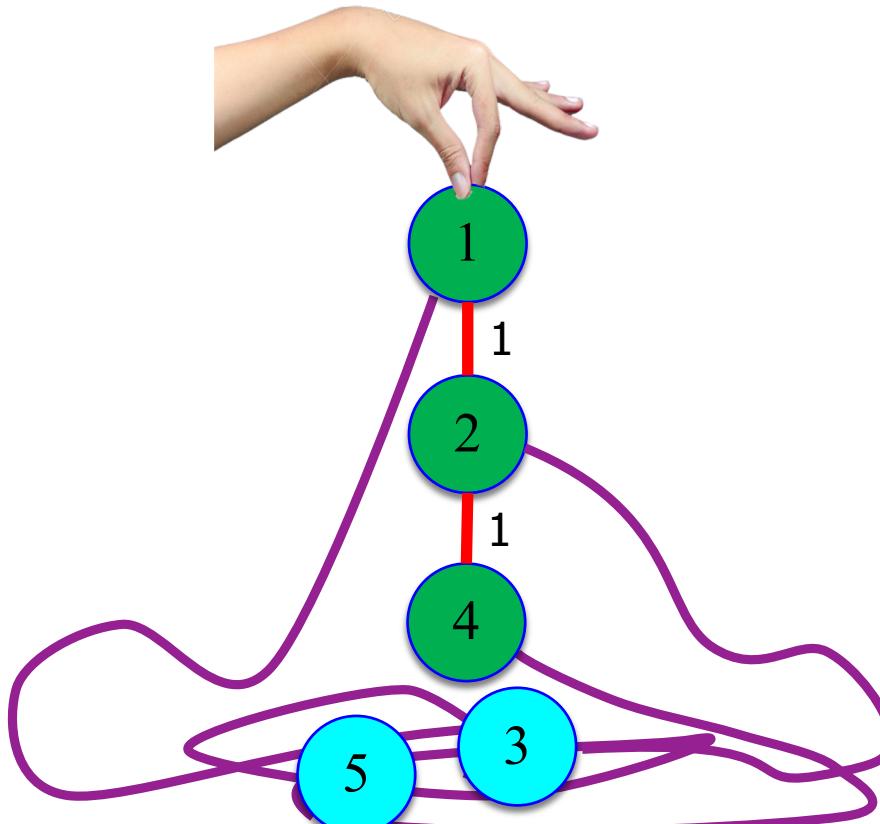
Thuật toán Moore – Dijkstra

- Quan sát
 - Đường thẳng là đường ngắn nhất
 - Đỉnh được kéo lên trước là đỉnh gần với 1 nhất
 - Vậy, đỉnh kế tiếp là 2



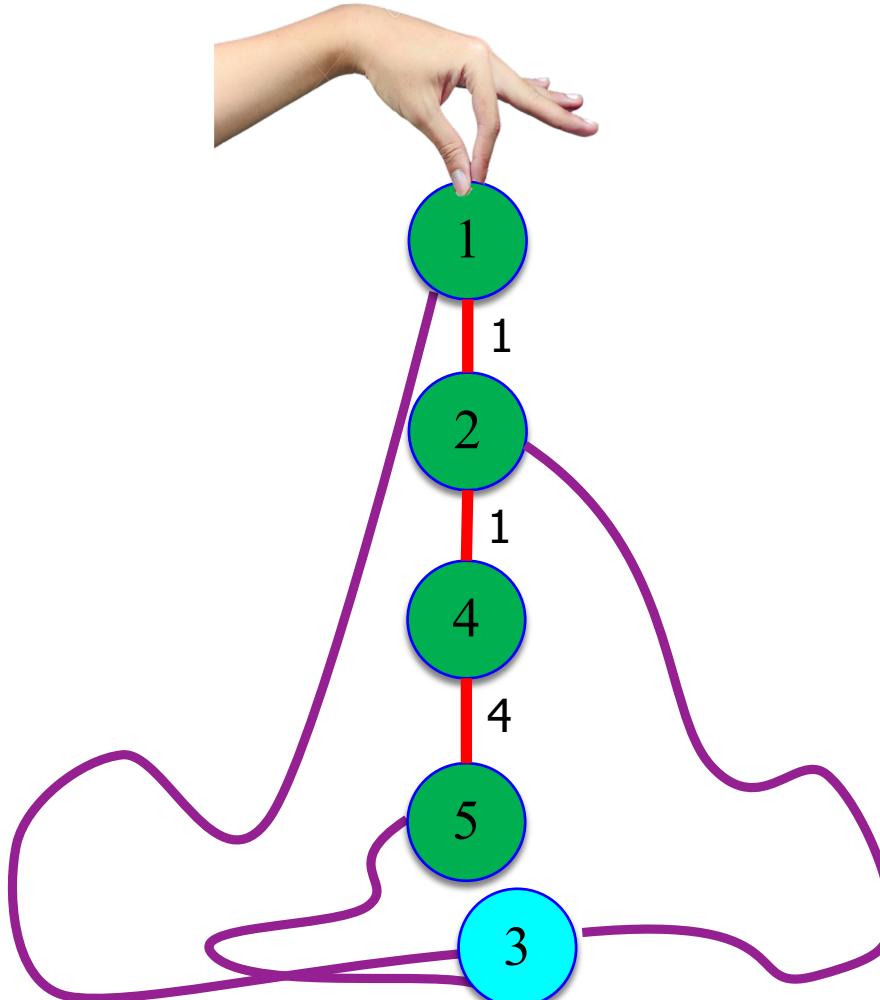
Thuật toán Moore – Dijkstra

- Quan sát
 - Tiếp đến là 4



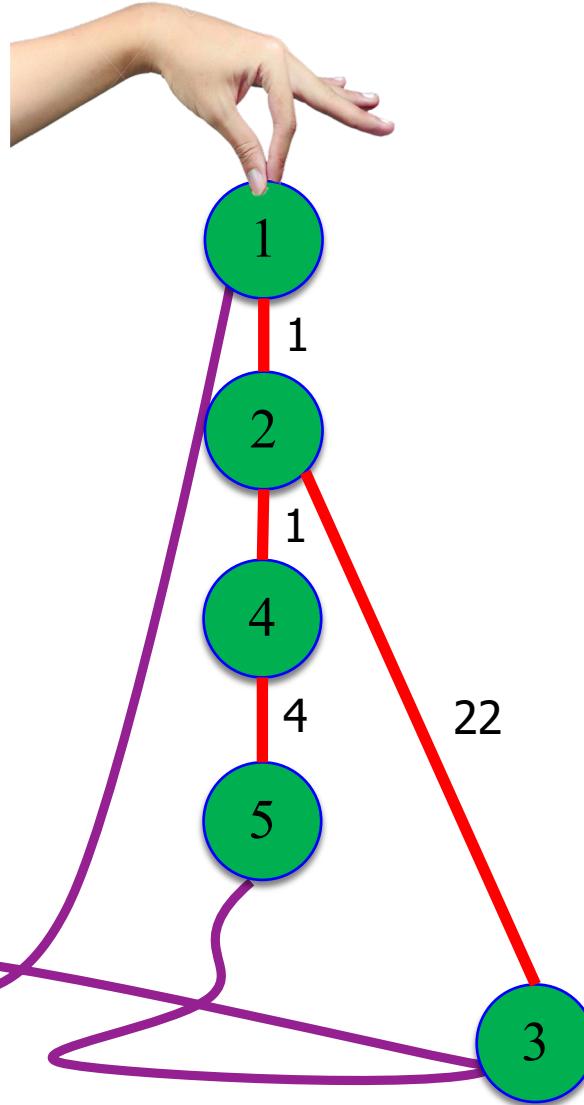
Thuật toán Moore – Dijkstra

- Quan sát



Thuật toán Moore – Dijkstra

- Quan sát

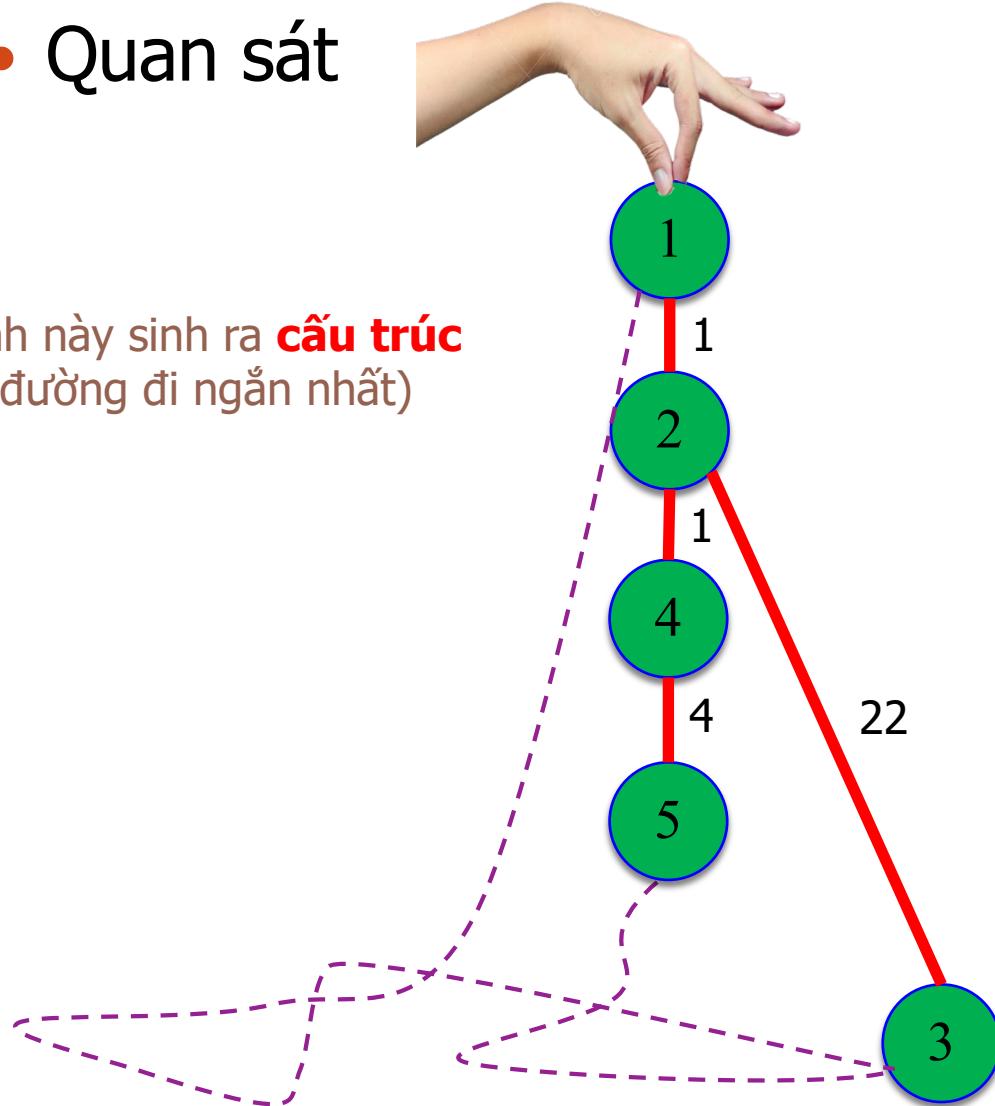


Thuật toán Moore – Dijkstra

- Quan sát

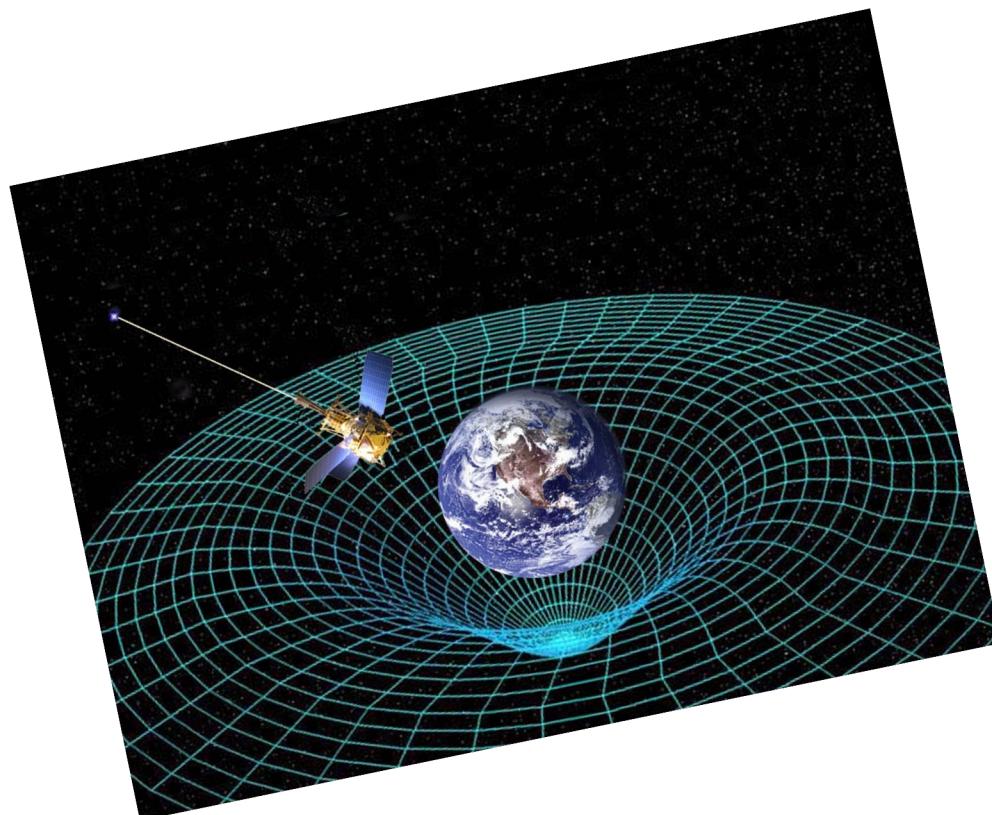
Quá trình này sinh ra **cấu trúc cây** (đường đi ngắn nhất)

Các đường đi ngắn nhất là
đường đi từ gốc đến các
đỉnh của cây này.

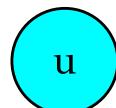


Cài đặt giải thuật như thế nào ?

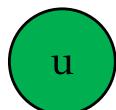
- **Không có** dây và trọng lực



Thuật toán Dijkstra qua ví dụ



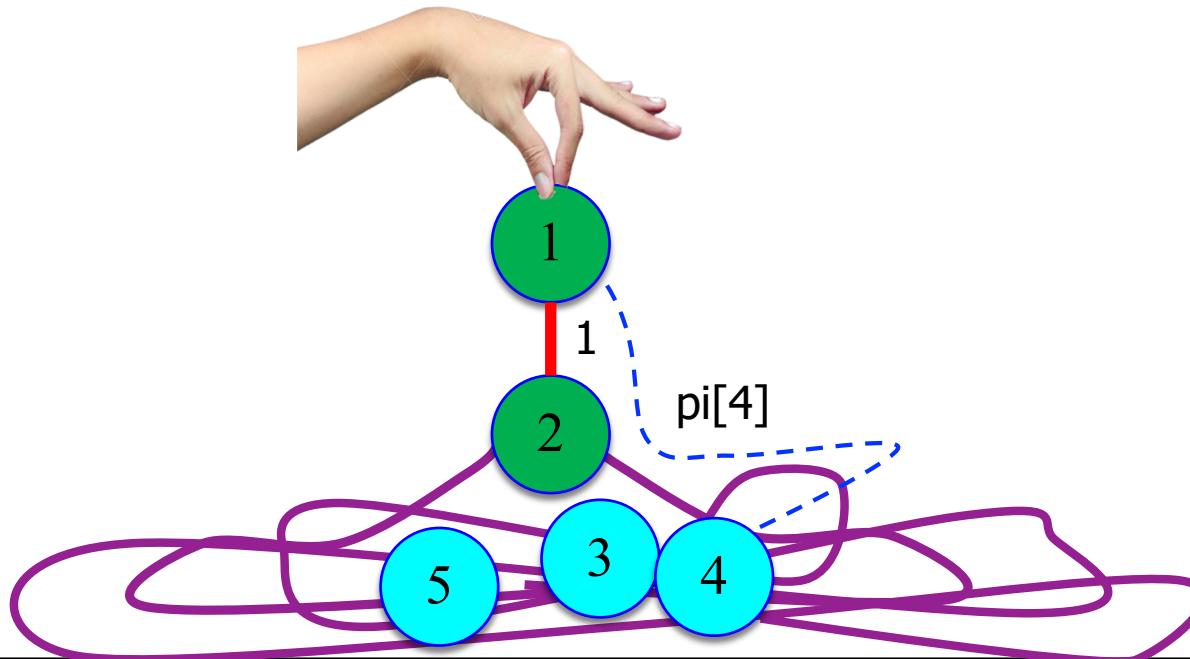
Đỉnh u chưa được kéo lên: chưa tìm được đường đi ngắn nhất đi đến nó
=> đỉnh chưa chắc chắn



Đỉnh u đã được kéo lên: đã tìm được đường đi ngắn nhất đi đến nó
=> đỉnh chắc chắn

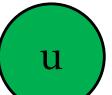


$x = \pi[u]$: chiều dài đường đi ngắn nhất từ s đến u, tính đến thời điểm hiện tại



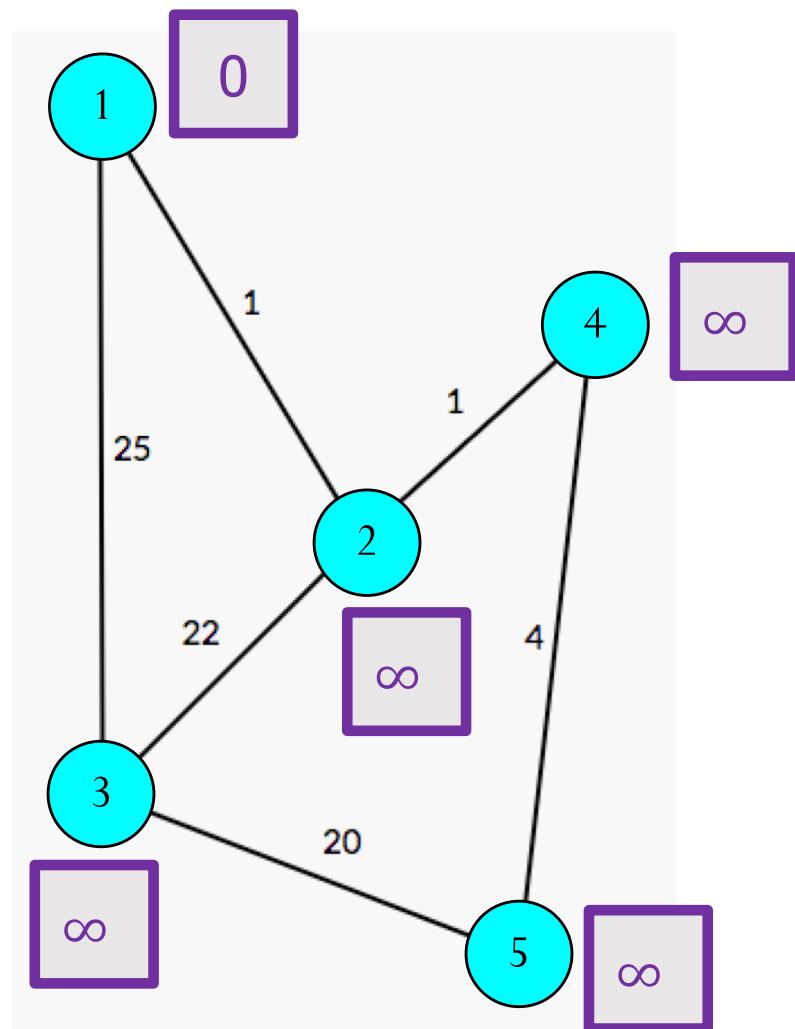
Thuật toán Dijkstra qua ví dụ

 Đỉnh chưa chắc chắn

 Đỉnh chắc chắn

 $x = \pi[u]$: chiều dài đường đi ngắn nhất tính đến thời điểm hiện tại.

- Đỉnh bắt đầu: s (vd: 1)
- Khởi tạo:
 - các đỉnh đều chưa chắc chắn
 - $\pi[s] = 0$,
 - Các đỉnh khác, $\pi[v] = \infty$.



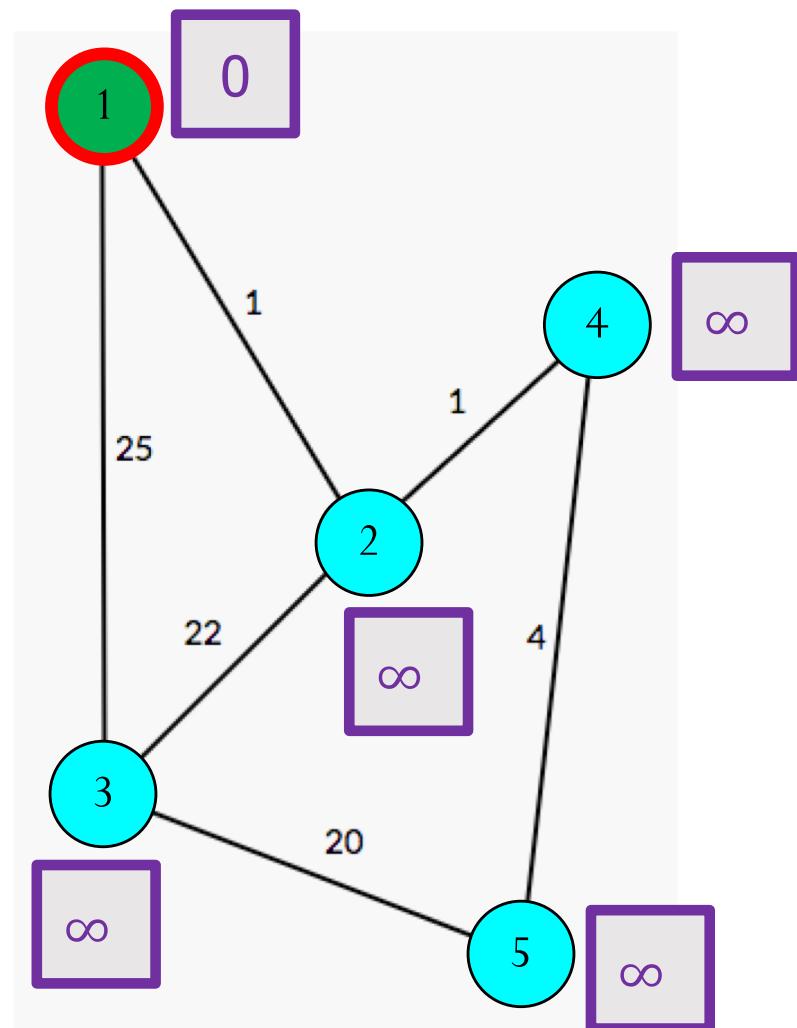
Thuật toán Dijkstra qua ví dụ

Đỉnh chưa chắc chắn

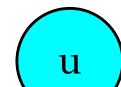
Đỉnh chắc chắn

$x = \pi[u]$: chiều dài đường đi ngắn nhất tính đến thời điểm hiện tại.

1. Chọn 1 đỉnh **chưa chắc chắn** u có $\pi[u]$ nhỏ nhất
2. Đánh dấu u là đỉnh **chắc chắn**
3. Cập nhật các đỉnh kề v của u
$$\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$$



Thuật toán Dijkstra qua ví dụ



Đỉnh chưa chắc chắn

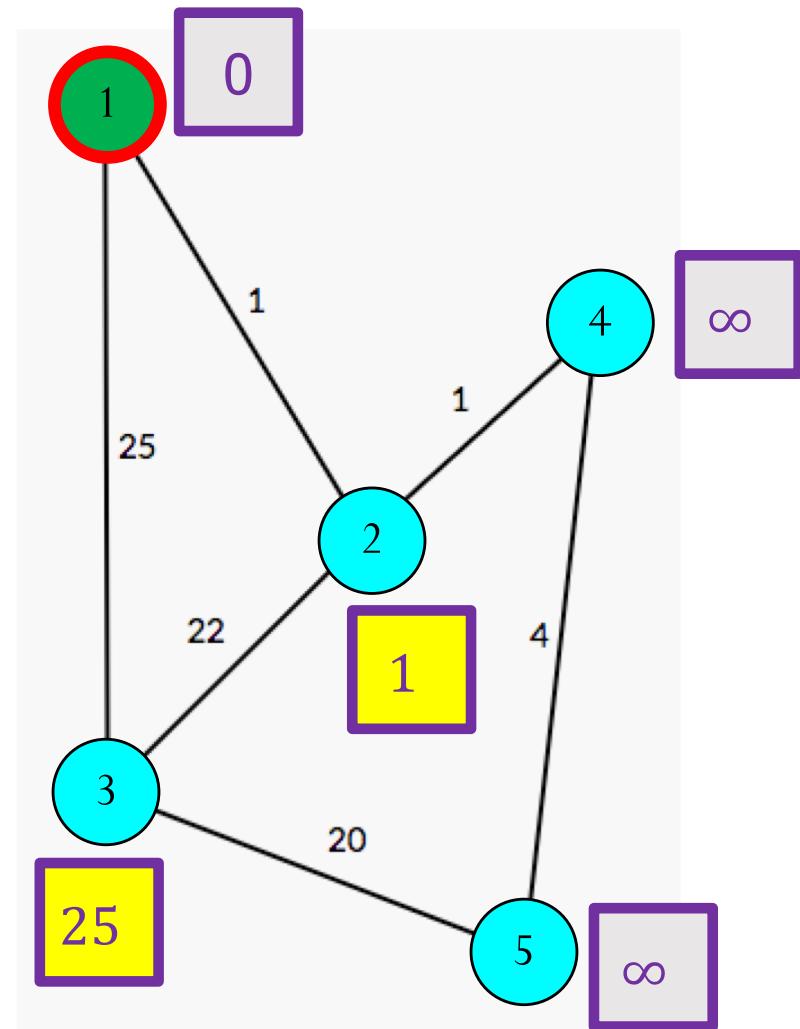


Đỉnh chắc chắn

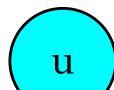


$x = \pi[u]$: chiều dài đường đi ngắn nhất tính đến thời điểm hiện tại.

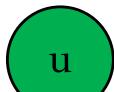
1. Chọn 1 đỉnh **chưa chắc chắn** u có $\pi[u]$ nhỏ nhất
2. Đánh dấu u là đỉnh **chắc chắn**
3. Cập nhật các đỉnh kề v của u
$$\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$$



Thuật toán Dijkstra qua ví dụ



Đỉnh chưa chắc chắn



Đỉnh chắc chắn

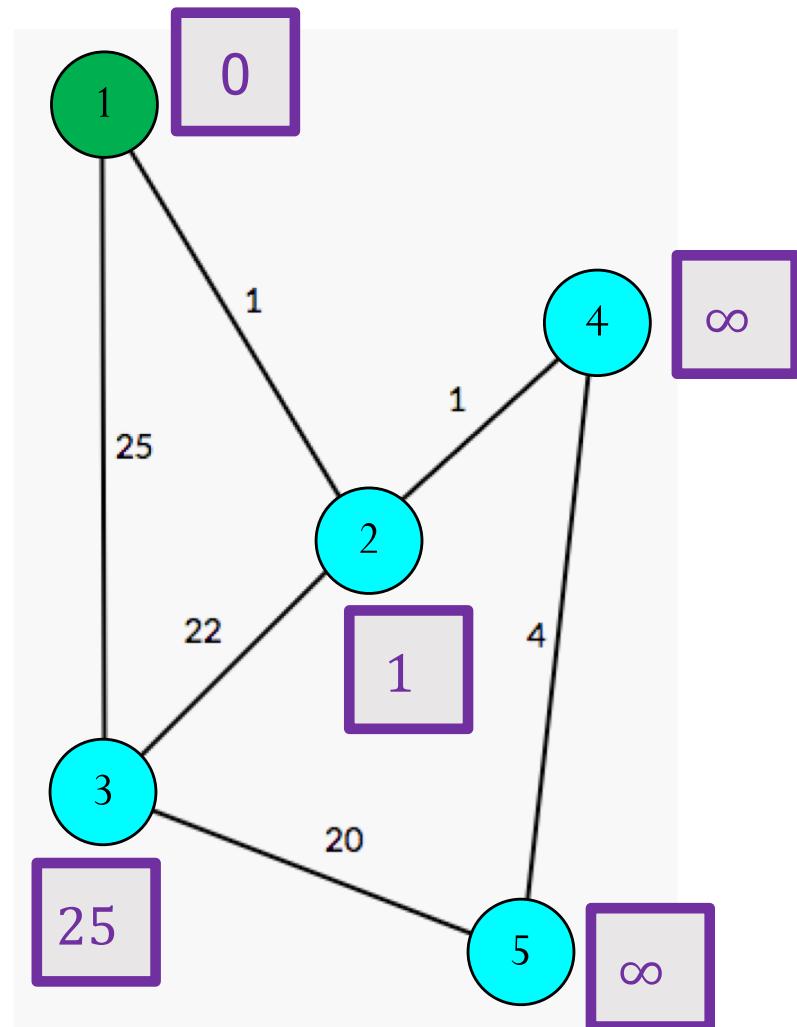


$x = \pi[u]$: chiều dài đường đi ngắn nhất tính đến thời điểm hiện tại.

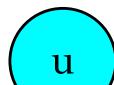
1. Chọn 1 đỉnh **chưa chắc chắn** u có $\pi[u]$ nhỏ nhất
2. Đánh dấu u là đỉnh **chắc chắn**
3. Cập nhật các đỉnh kề v của u
$$\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$$

Lặp lại 1, 2, 3

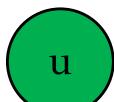
43



Thuật toán Dijkstra qua ví dụ



Đỉnh chưa chắc chắn



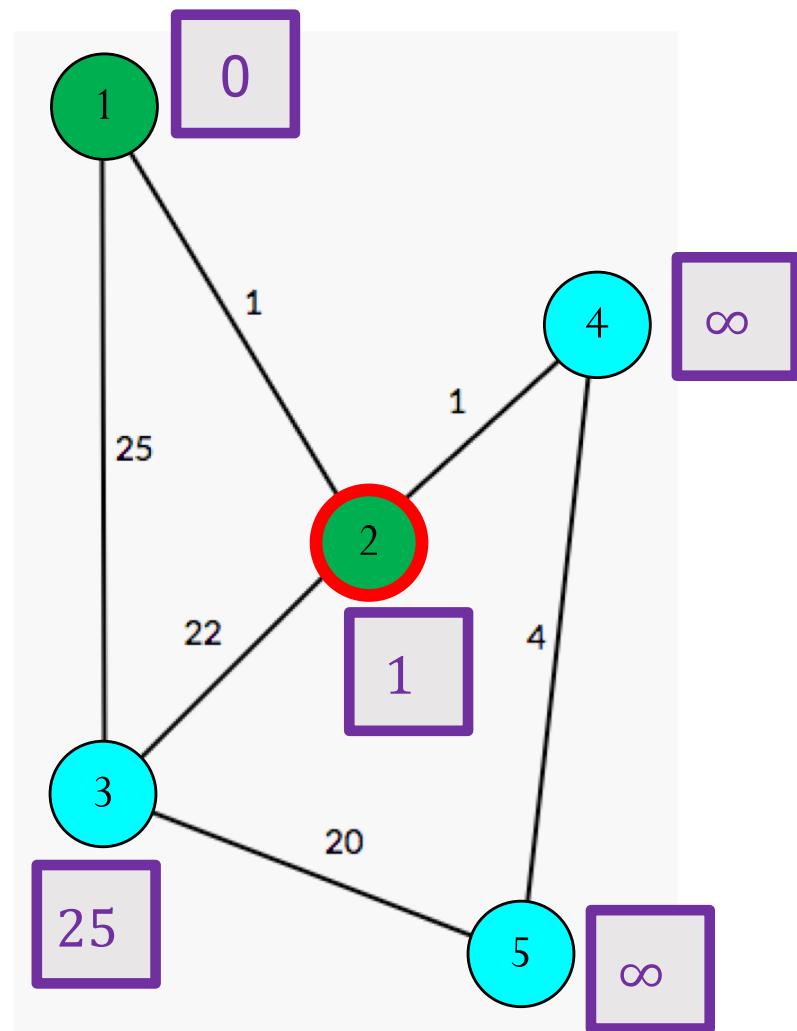
Đỉnh chắc chắn



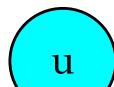
$x = \pi[u]$: chiều dài đường đi ngắn nhất tính đến thời điểm hiện tại.

1. Chọn 1 đỉnh **chưa chắc chắn** u có $\pi[u]$ nhỏ nhất
2. Đánh dấu u là đỉnh **chắc chắn**
3. Cập nhật các đỉnh kề v của u
$$\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$$

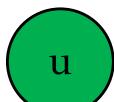
Lặp lại 1, 2, 3



Thuật toán Dijkstra qua ví dụ



Đỉnh chưa chắc chắn



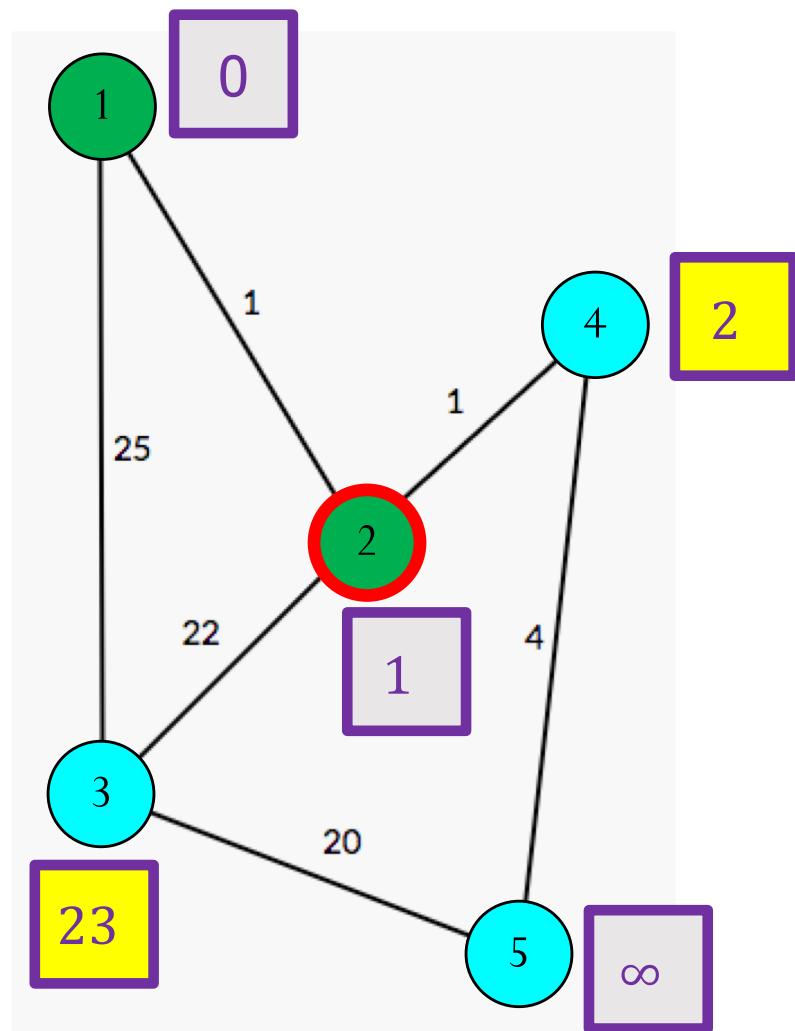
Đỉnh chắc chắn



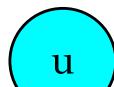
$x = \pi[u]$: chiều dài đường đi ngắn nhất tính đến thời điểm hiện tại.

1. Chọn 1 đỉnh **chưa chắc chắn** u có $\pi[u]$ nhỏ nhất
2. Đánh dấu u là đỉnh **chắc chắn**
3. Cập nhật các đỉnh kề v của u
$$\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$$

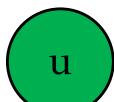
Lặp lại 1, 2, 3



Thuật toán Dijkstra qua ví dụ



Đỉnh chưa chắc chắn



Đỉnh chắc chắn

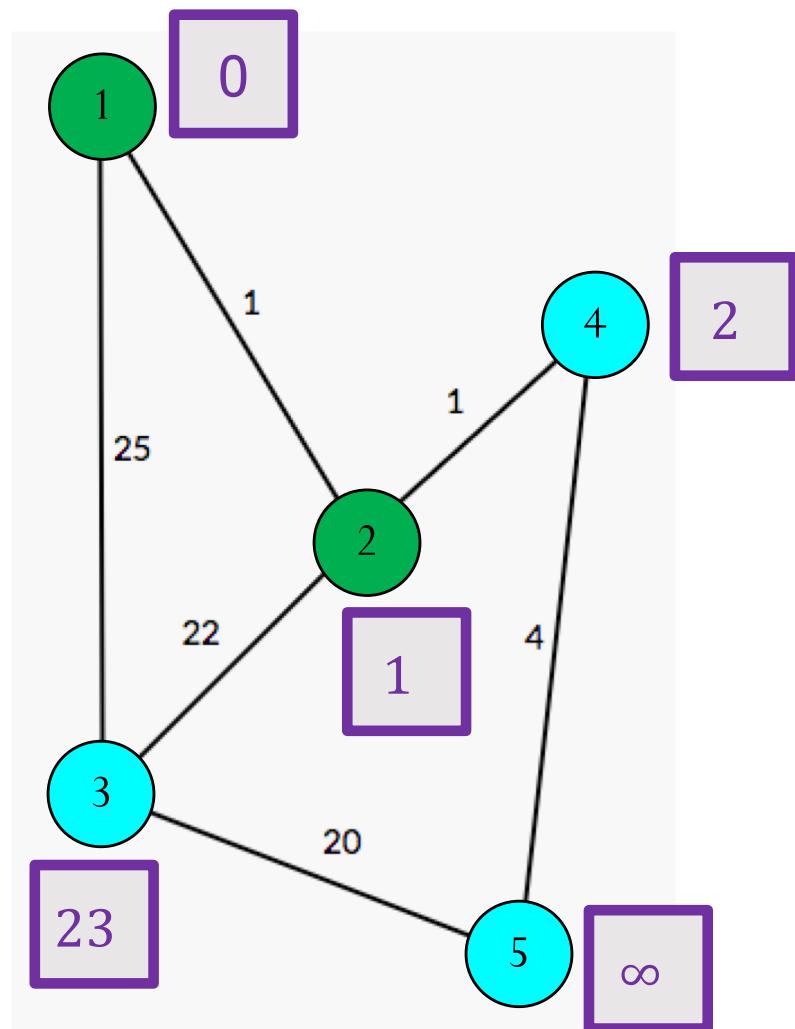


$x = \pi[u]$: chiều dài đường đi ngắn nhất tính đến thời điểm hiện tại.

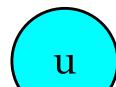
1. Chọn 1 đỉnh **chưa chắc chắn** u có $\pi[u]$ nhỏ nhất
2. Đánh dấu u là đỉnh **chắc chắn**
3. Cập nhật các đỉnh kề v của u
$$\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$$

Lặp lại 1, 2, 3

46



Thuật toán Dijkstra qua ví dụ



Đỉnh chưa chắc chắn



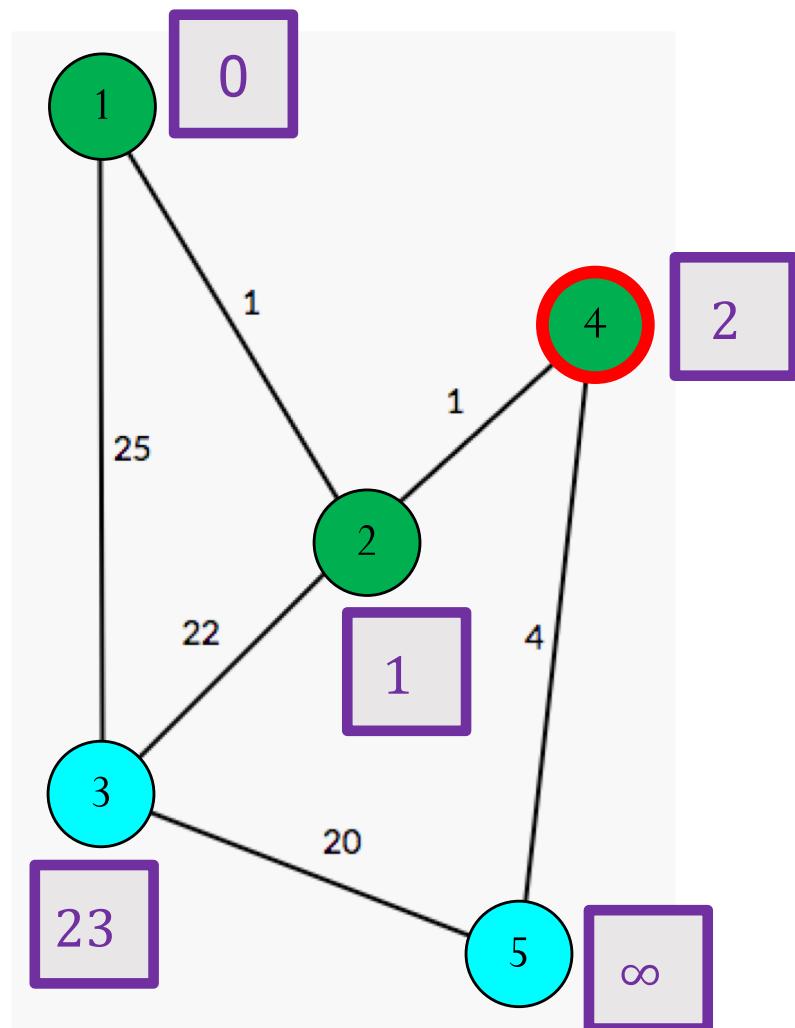
Đỉnh chắc chắn



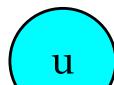
$x = \pi[u]$: chiều dài đường đi ngắn nhất tính đến thời điểm hiện tại.

1. Chọn 1 đỉnh **chưa chắc chắn** u có $\pi[u]$ nhỏ nhất
2. Đánh dấu u là đỉnh **chắc chắn**
3. Cập nhật các đỉnh kề v của u
$$\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$$

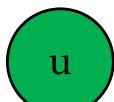
Lặp lại 1, 2, 3



Thuật toán Dijkstra qua ví dụ



Đỉnh chưa chắc chắn



Đỉnh chắc chắn

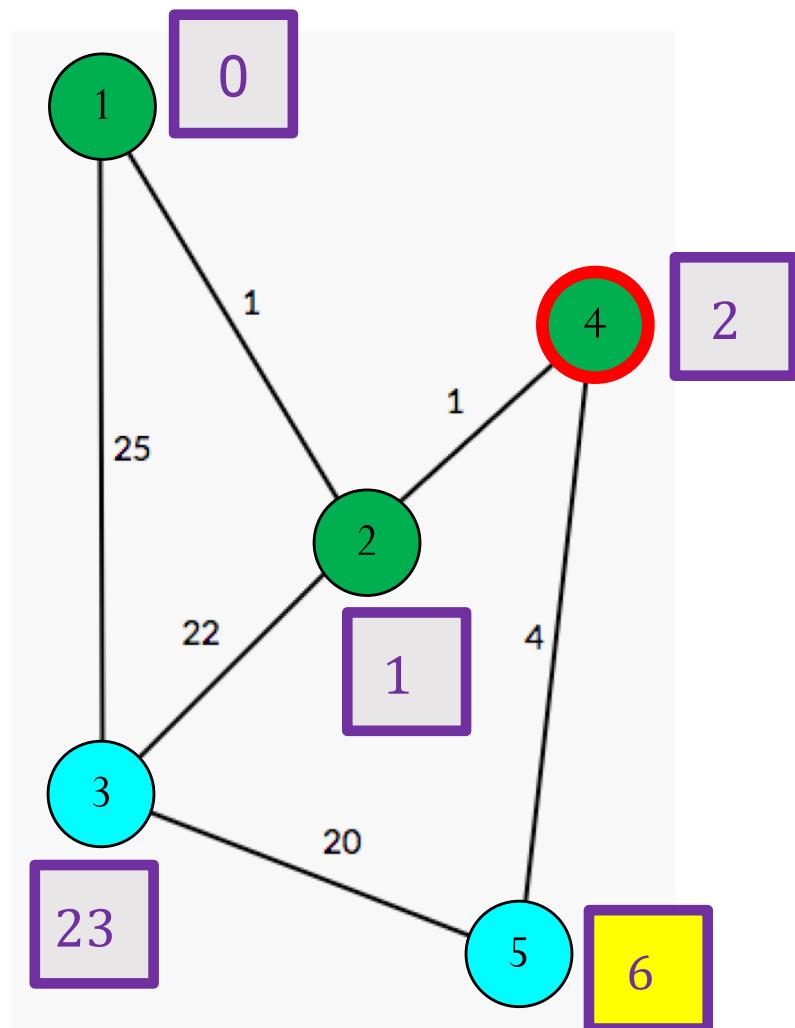


$x = \pi[u]$: chiều dài đường đi ngắn nhất tính đến thời điểm hiện tại.

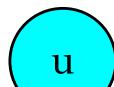
1. Chọn 1 đỉnh **chưa chắc chắn** u có $\pi[u]$ nhỏ nhất
2. Đánh dấu u là đỉnh **chắc chắn**
3. Cập nhật các đỉnh kề v của u
$$\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$$

Lặp lại 1, 2, 3

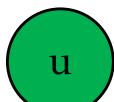
48



Thuật toán Dijkstra qua ví dụ



Đỉnh chưa chắc chắn



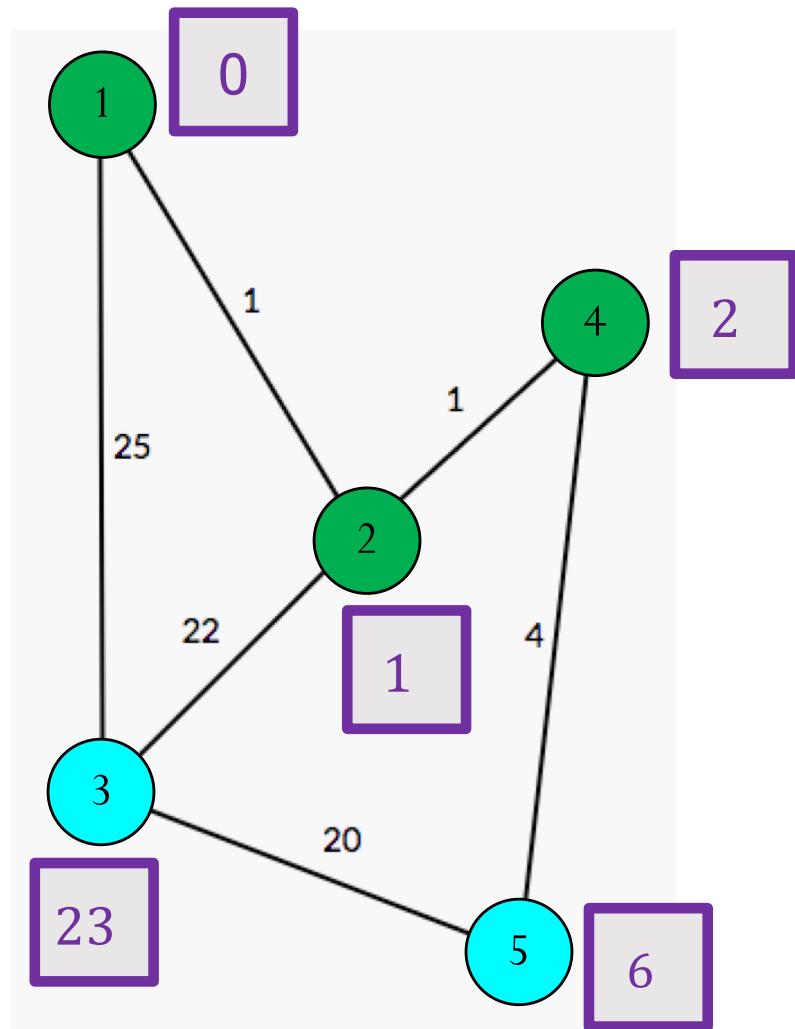
Đỉnh chắc chắn



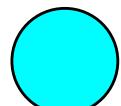
$x = \pi[u]$: chiều dài đường đi ngắn nhất tính đến thời điểm hiện tại.

1. Chọn 1 đỉnh **chưa chắc chắn** u có $\pi[u]$ nhỏ nhất
2. Đánh dấu u là đỉnh **chắc chắn**
3. Cập nhật các đỉnh kề v của u
 $\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$

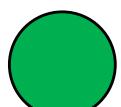
Lặp lại 1, 2, 3



Thuật toán Dijkstra qua ví dụ



Đỉnh chưa chắc chắn



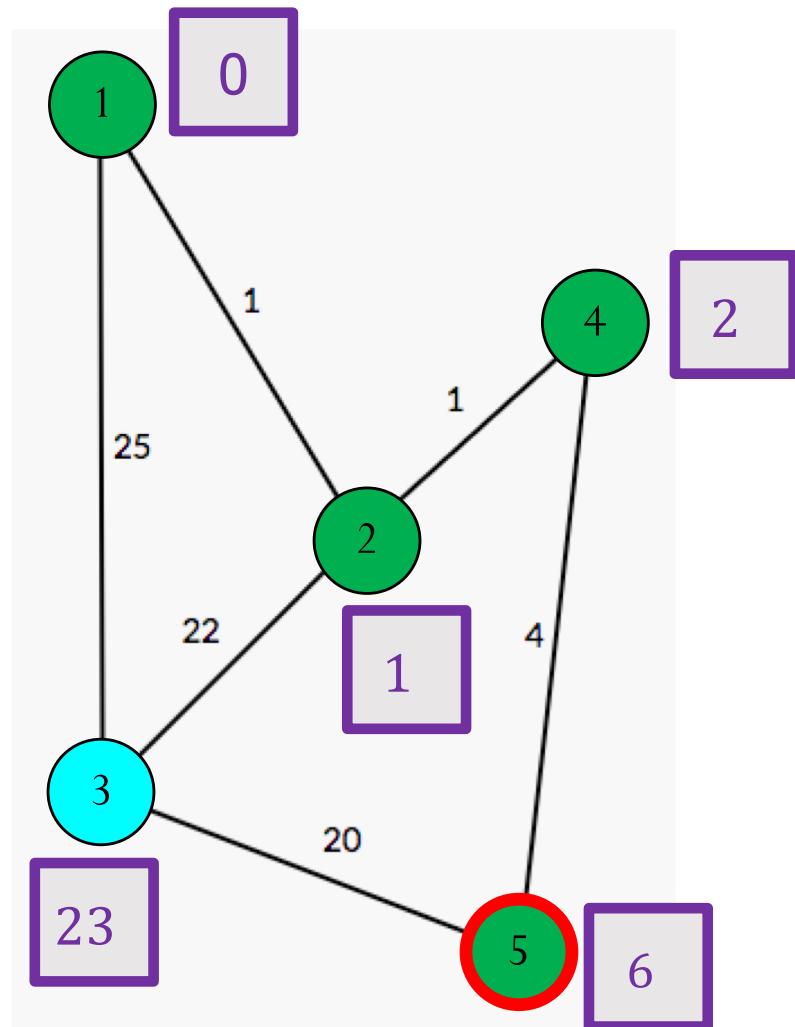
Đỉnh chắc chắn, đã tìm được
đường đi ngắn nhất từ s đến nó.



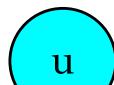
$x = \pi[u]$: chiều dài đường đi
ngắn nhất tính đến thời điểm này.

1. Chọn 1 đỉnh **chưa chắc chắn** u có $\pi[u]$ nhỏ nhất
2. Đánh dấu u là đỉnh **chắc chắn**
3. Cập nhật các đỉnh kề v của u
 $\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$

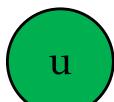
Lặp lại 1, 2, 3



Thuật toán Dijkstra qua ví dụ



Đỉnh chưa chắc chắn



Đỉnh chắc chắn

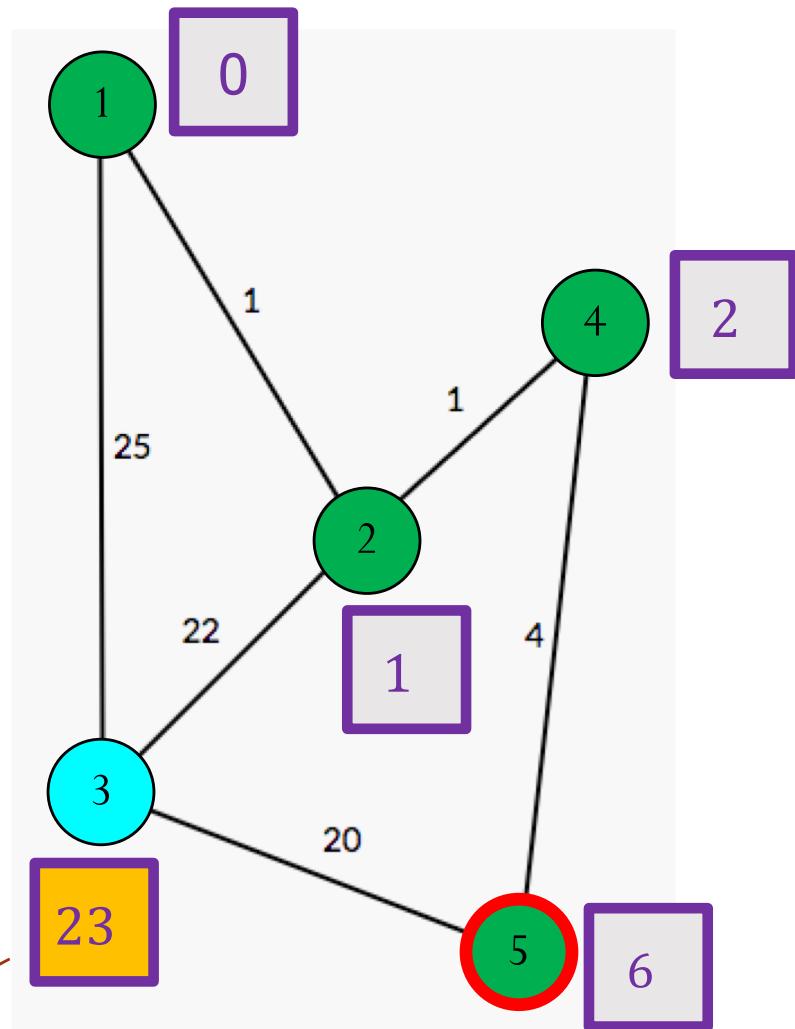


$x = \pi[u]$: chiều dài đường đi ngắn nhất tính đến thời điểm hiện tại.

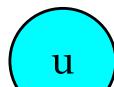
1. Chọn 1 đỉnh **chưa chắc chắn** u có $\pi[u]$ nhỏ nhất
2. Đánh dấu u là đỉnh **chắc chắn**
3. Cập nhật các đỉnh kề v của u
$$\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$$

Lặp lại 1, 2, 3

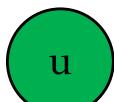
Không cập nhật



Thuật toán Dijkstra qua ví dụ



Đỉnh chưa chắc chắn



Đỉnh chắc chắn

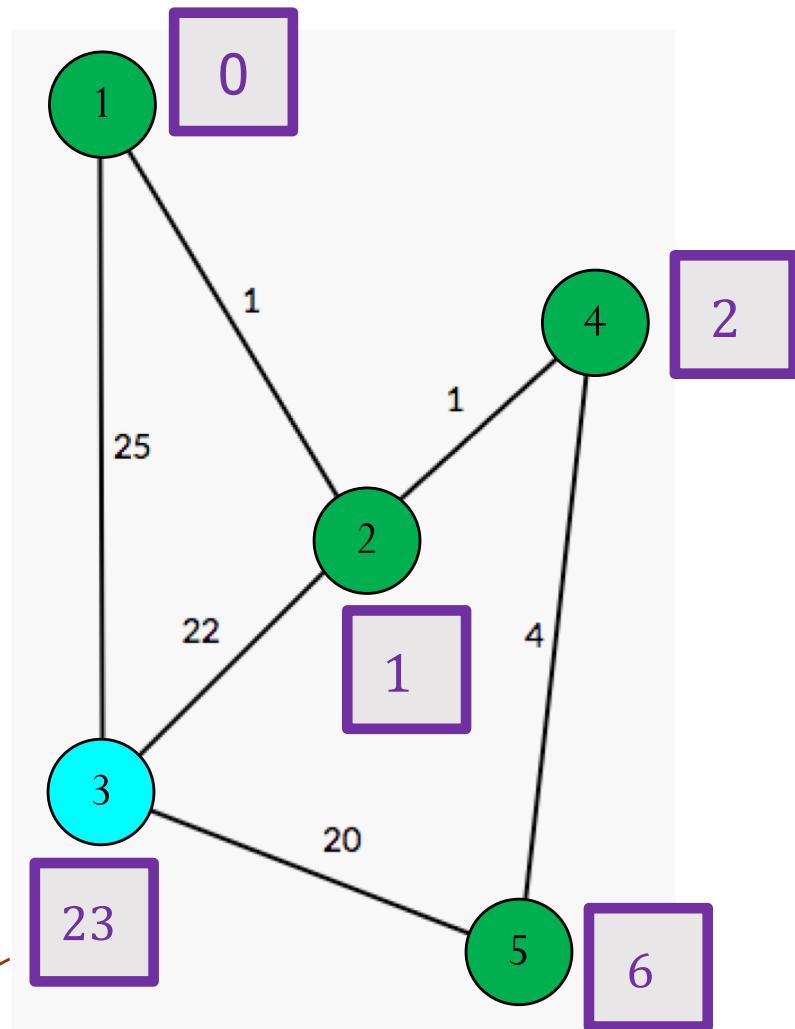


$x = \pi[u]$: chiều dài đường đi ngắn nhất tính đến thời điểm hiện tại.

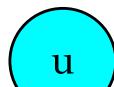
1. Chọn 1 đỉnh **chưa chắc chắn** u có $\pi[u]$ nhỏ nhất
2. Đánh dấu u là đỉnh **chắc chắn**
3. Cập nhật các đỉnh kề v của u
$$\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$$

Lặp lại 1, 2, 3

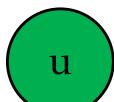
Không cập nhật



Thuật toán Dijkstra qua ví dụ



Đỉnh chưa chắc chắn



Đỉnh chắc chắn



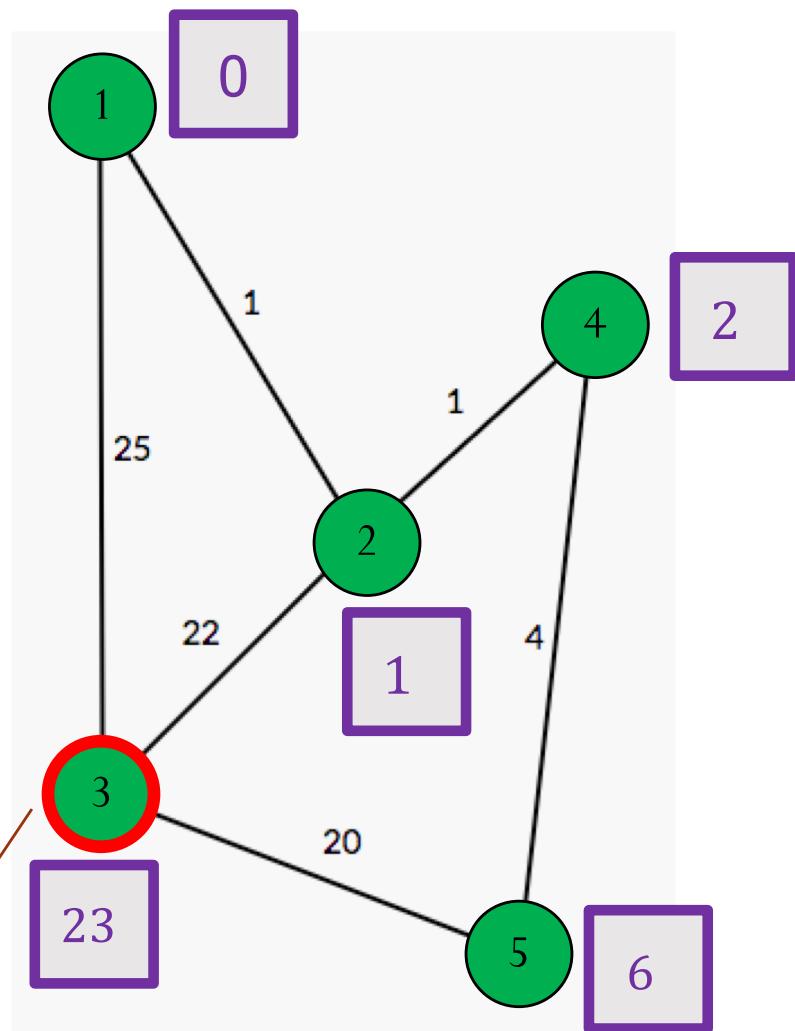
$x = \pi[u]$: chiều dài đường đi ngắn nhất tính đến thời điểm hiện tại.

1. Chọn 1 đỉnh **chưa chắc chắn** u có $\pi[u]$ nhỏ nhất
2. Đánh dấu u là đỉnh **chắc chắn**
3. Cập nhật các đỉnh kề v của u
$$\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$$

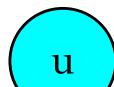
Lặp lại 1, 2, 3

Không còn đỉnh kề

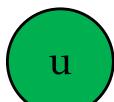
53



Thuật toán Dijkstra qua ví dụ



Đỉnh chưa chắc chắn



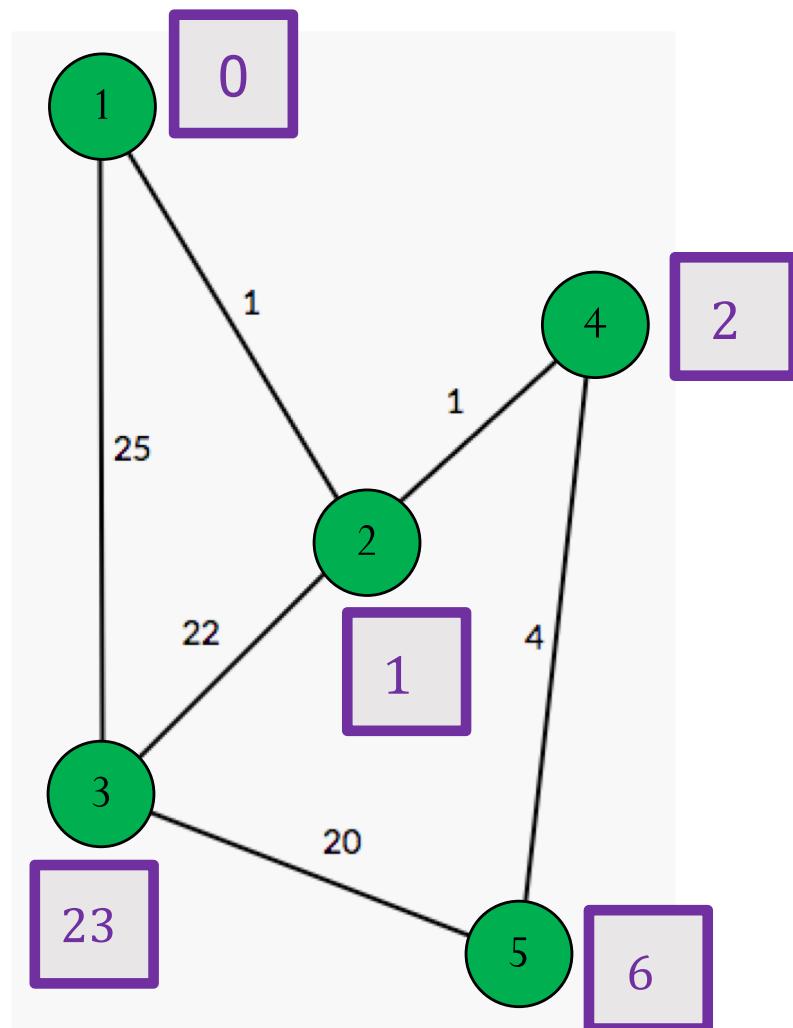
Đỉnh chắc chắn



$x = \pi[u]$: chiều dài đường đi ngắn nhất tính đến thời điểm hiện tại.

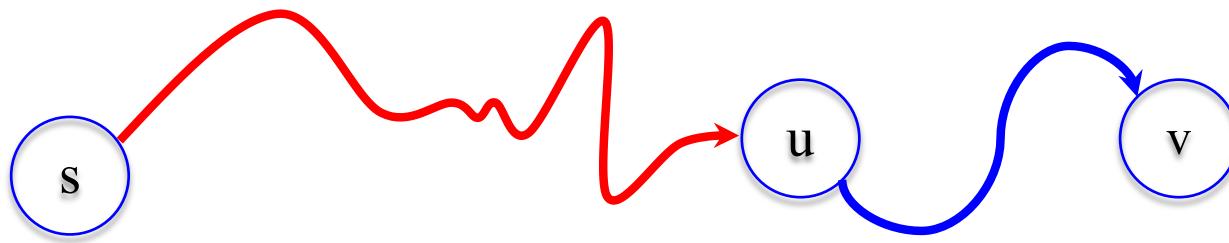
1. Chọn 1 đỉnh **chưa chắc chắn** u có $\pi[u]$ nhỏ nhất
2. Đánh dấu u là đỉnh **chắc chắn**
3. Cập nhật các đỉnh kề v của u
 $\pi[v] = \min(\pi[v], \pi[u] + L[u][v])$

Lặp lại 1, 2, 3



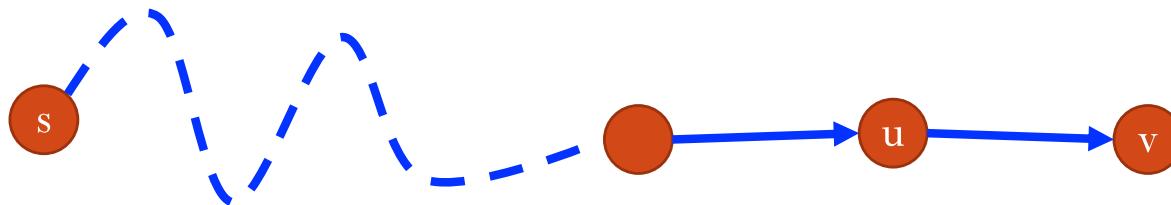
Biểu diễn đường đi ngắn nhất

- Một đoạn đường đi (sub-path) của đường đi ngắn nhất cũng là đường đi ngắn nhất.
 - Tính chất *cấu trúc con tối ưu* (optimal substructure)



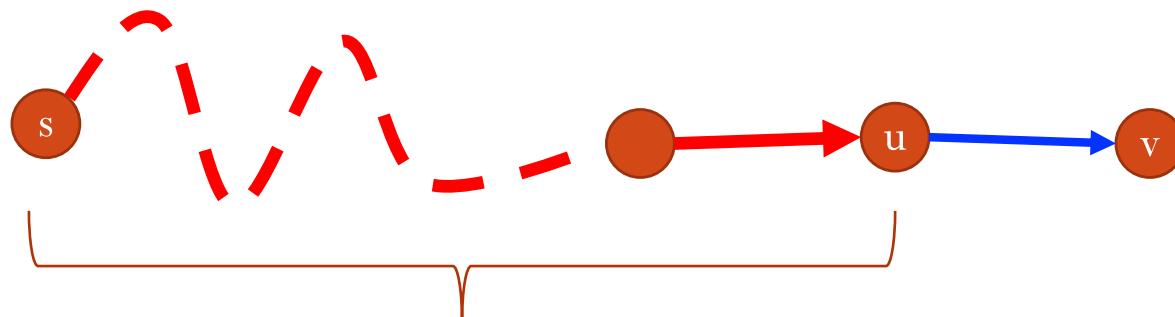
Biểu diễn đường đi ngắn nhất

- Một đoạn đường đi (sub-path) của đường đi ngắn nhất cũng là đường đi ngắn nhất.
 - Tính chất *cấu trúc con tối ưu* (optimal substructure)



Biểu diễn đường đi ngắn nhất

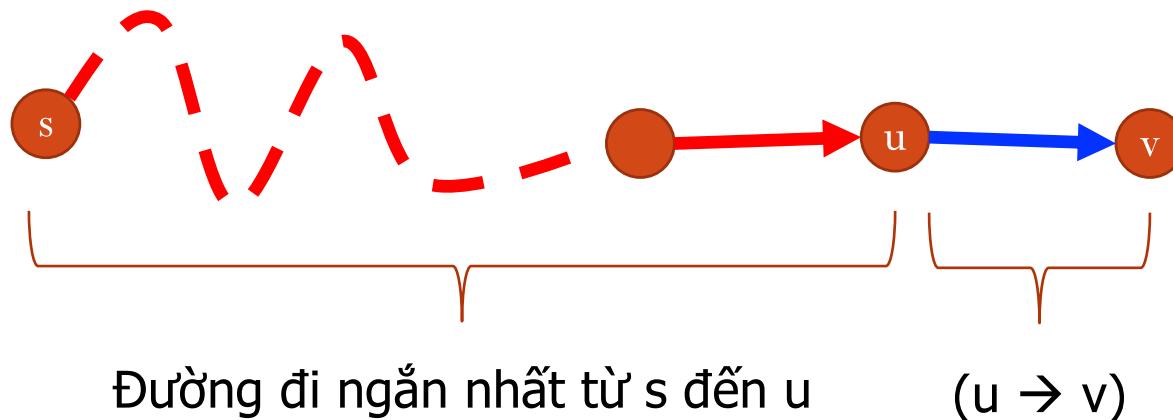
- Một đoạn đường đi (sub-path) của đường đi ngắn nhất cũng là đường đi ngắn nhất.
 - Tính chất *cấu trúc con tối ưu* (optimal substructure)



Đường đi ngắn nhất từ s đến u

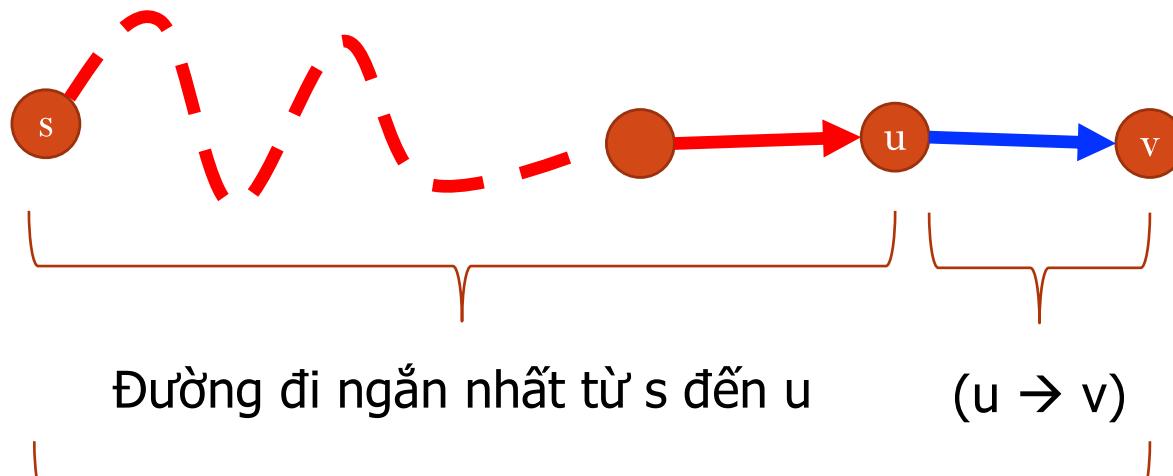
Biểu diễn đường đi ngắn nhất

- Một đoạn đường đi (sub-path) của đường đi ngắn nhất cũng là đường đi ngắn nhất.
 - Tính chất *cấu trúc con tối ưu* (optimal substructure)



Biểu diễn đường đi ngắn nhất

- Một đoạn đường đi (sub-path) của đường đi ngắn nhất cũng là đường đi ngắn nhất.
 - Tính chất *cấu trúc con tối ưu* (optimal substructure)

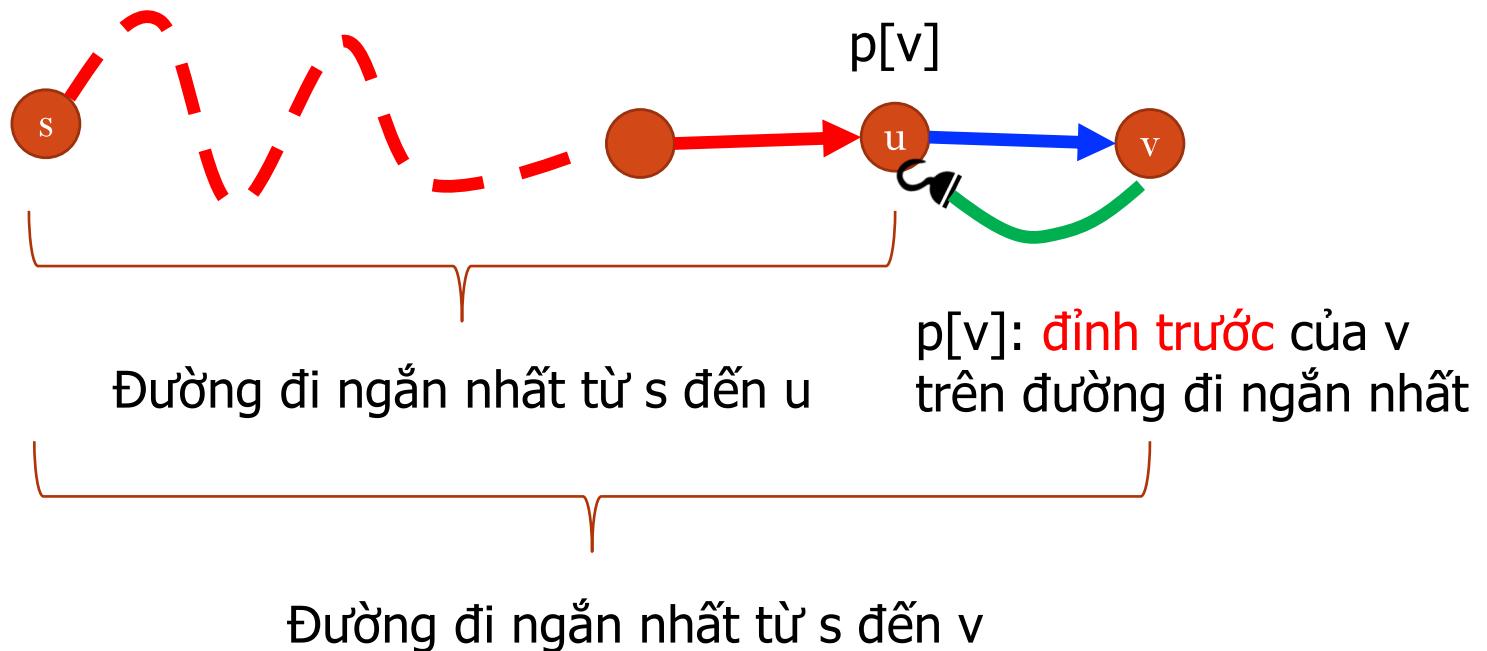


Đường đi ngắn nhất từ s đến v
(\mathbf{s} \rightarrow v)

Đường đi ngắn nhất từ s đến u

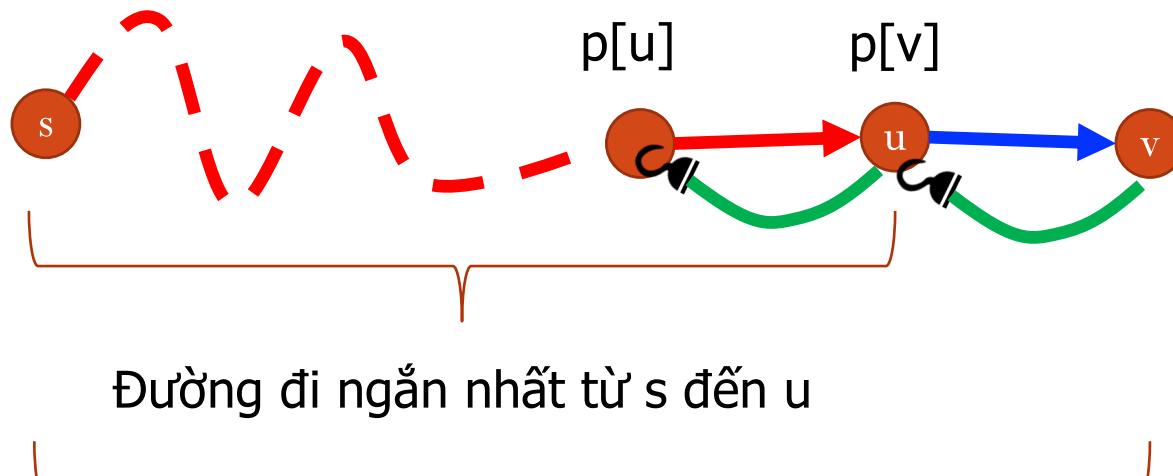
Biểu diễn đường đi ngắn nhất

- Một đoạn đường đi (sub-path) của đường đi ngắn nhất cũng là đường đi ngắn nhất.
 - Tính chất *cấu trúc con tối ưu* (optimal substructure)



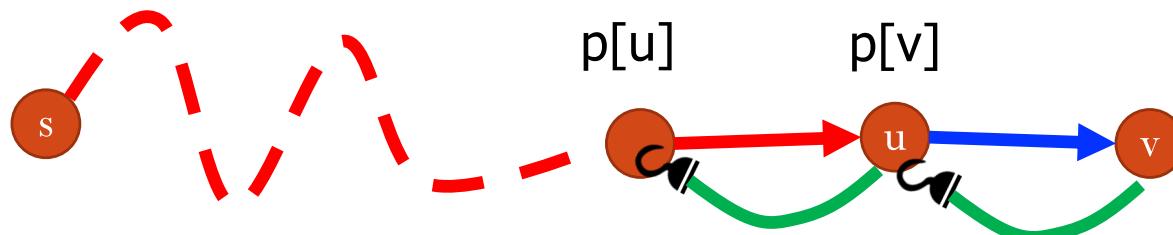
Biểu diễn đường đi ngắn nhất

- Một đoạn đường đi (sub-path) của đường đi ngắn nhất cũng là đường đi ngắn nhất.
 - Tính chất *cấu trúc con tối ưu* (optimal substructure)



Biểu diễn đường đi ngắn nhất

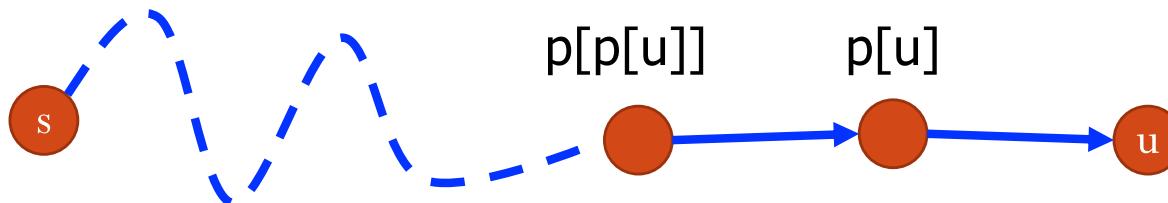
- Một đoạn đường đi (sub-path) của đường đi ngắn nhất cũng là đường đi ngắn nhất.
 - Tính chất *cấu trúc con tối ưu* (optimal substructure)



- Biểu diễn đường đi:
 - Lưu các $p[1], p[2], \dots, p[n]$

Thuật toán Moore-Dijkstra

- Gọi đỉnh bắt đầu là s
- Các biến hỗ trợ
 - $\text{pi}[u]$: chiều dài đường đi ngắn nhất (tạm thời) từ s đến u.
 - $\text{p}[u]$: đỉnh trước đỉnh u trên đường đi ngắn nhất (tạm thời) từ s đến u.
 - $\text{mark}[u]$: đánh dấu đỉnh u (đã tìm được đường đi ngắn nhất đến u, chắc chắn/chưa chắc chắn)
- Nếu $\text{mark}[u] == 1$ thì $\text{pi}[u]$ chứa chiều dài đường đi ngắn nhất từ s đến u và $\text{p}[u]$ là đỉnh trước của đỉnh.



Thuật toán Moore-Dijkstra

- Giải thuật

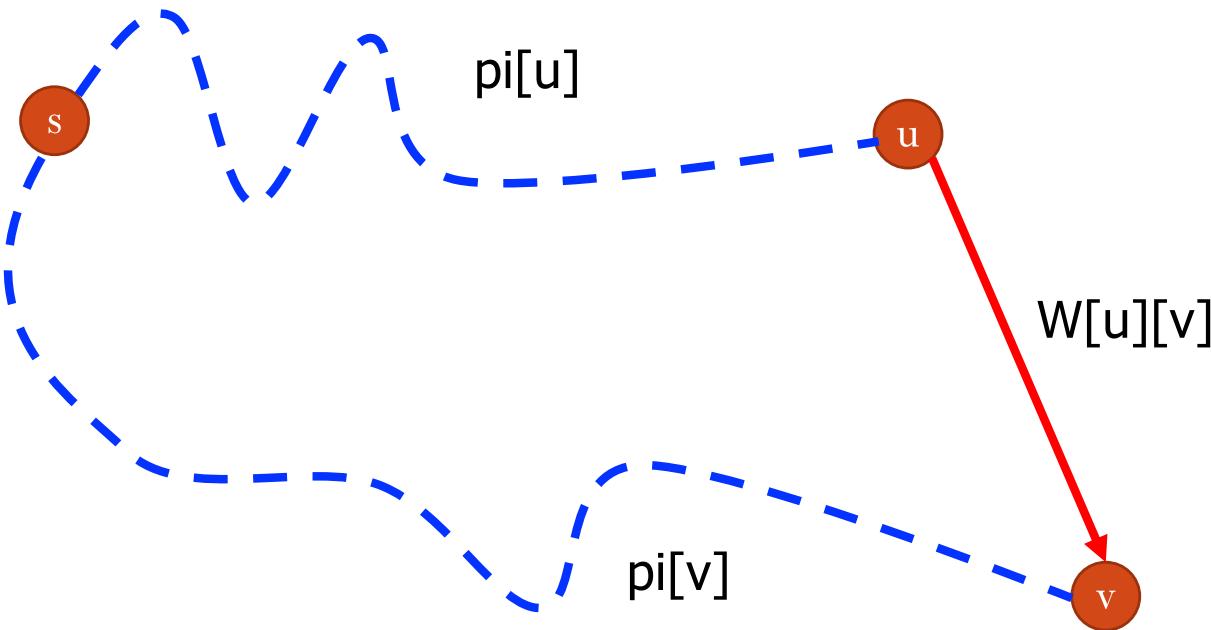
- Khởi tạo:

- $\text{mark}[u] = 0$ với mọi u (0 : chưa chắc chắn; 1 : chắc chắn)
 - $\text{pi}[u] = \infty$ với mọi u , $p[u] = -1$ với mọi u
 - $\text{pi}[s] = 0$; đường đi ngắn nhất từ s đến s bằng 0 .

- Lặp $n-1$ lần

- Chọn đỉnh **chưa đánh dấu** và có **$\text{pi}[u]$ nhỏ nhất**
 - Đánh dấu u
 - Xét các đỉnh kề v (chưa đánh dấu) của u để cập nhật đường đi nếu đường đi qua u rồi đến v tốt hơn đường đi cũ

Thuật toán Moore-Dijkstra



$$\pi[u] + w[u][v] < \pi[v]$$

Thuật toán Moore-Dijkstra

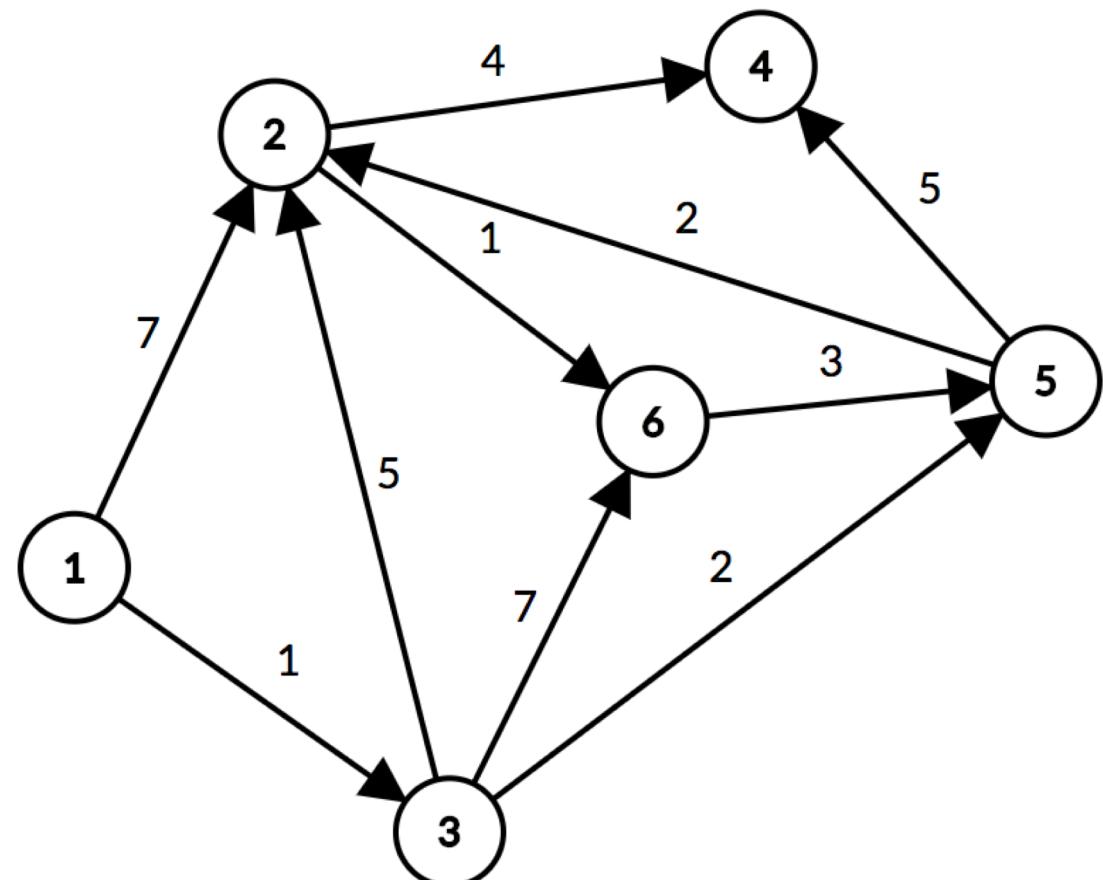
```
void Dijkstra(Graph *pG, int s) {
    Khởi tạo mark[u] = 0, với mọi u //Tất cả các đỉnh đều chưa
                                         //chắc chắn
    Khởi tạo pi[u] = oo, với mọi u
    pi[s] = 0;
    Lặp n - 1 lần:
        //1. Tìm u chưa chắc chắn và có pi[u] nhỏ nhất
        //2. Đánh dấu u là đỉnh chắc chắn: mark[u] = 1;
        //3. Xét các kề của v để cập nhật pi và p (nếu thỏa)
        for (các đỉnh kề chưa chắc chắn của u)
            if (pi[u] + trọng số cung (u, v) < pi[v]) {
                pi[v] = pi[u] + trọng số cung (u, v)
                p[v] = u; //cập nhật đỉnh trước của v
            }
        }
}
```

Thuật toán Moore-Dijkstra

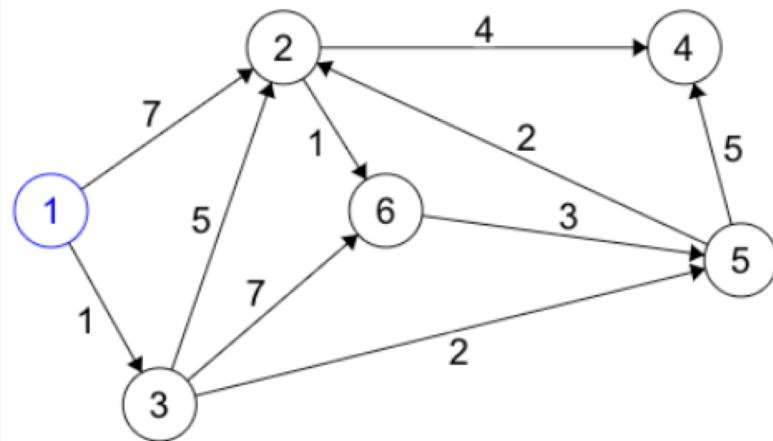
```
void Dijkstra(Graph *pG, int s) {
    int u, v, it;
    for (u = 1; u <= pG->n; u++) {
        mark[u] = 0;
        pi[u] = INFINITY;
    }
    pi[s] = 0;
    for (it = 1; it < pG->n; it++) {
        //1. Tìm u có mark[u] == 0 và có pi[u] nhỏ nhất
        //2. Đánh dấu u đã xét mark[u] = 1;
        //3. Cập nhật pi và p của các đỉnh kề của u (nếu thỏa)
        for (v = 1; v <= G->n; v++)
            if (pG->W[u][v] != NO_EDGE && mark[v] == 0) {
                if (pi[u] + pG->W[u][v] < pi[v]) {
                    pi[v] = pi[u] + pG->W[u][v];
                    p[v] = u;
                }
            }
    }
}
```

Thuật toán Moore-Dijkstra

Tìm đường đi ngắn nhất từ 1 đến các đỉnh khác trên đồ thị có hướng



Help Clear shift Delete Edit Undo



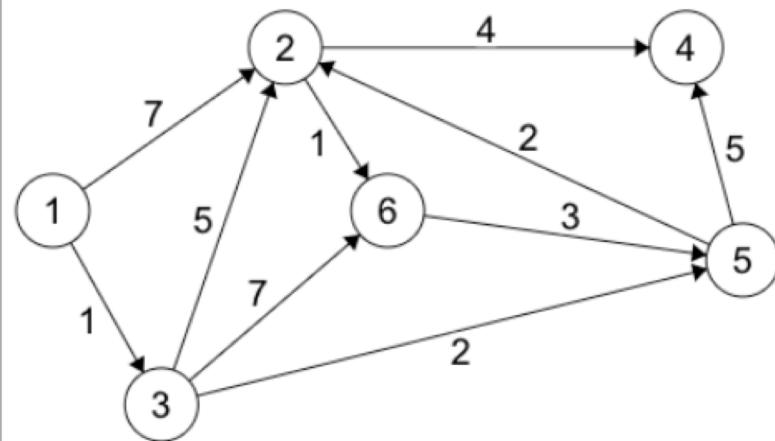
Khởi tạo

- ???
- ???
- ???

1. Áp dụng thuật toán Moore - Dijkstra và ghi kết quả vào bảng

	1	2	3	4	5	6	Công việc
Khởi tạo							
1							
2							
3							
4							
5							

Help Clear shift Delete Edit Undo



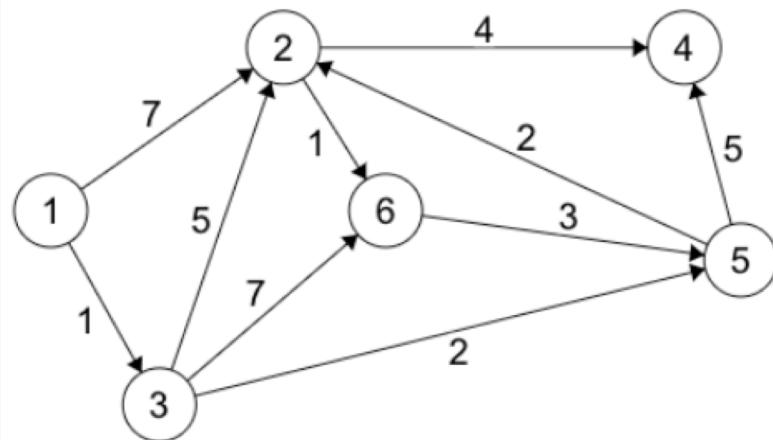
Khởi tạo

- $\text{mark}[u] = 0$
- $\text{pi}[u] = \text{oo}$ ($u \neq 1$)
- $\text{pi}[1] = 0$

1. Áp dụng thuật toán Moore - Dijkstra và ghi kết quả vào bảng

	1	2	3	4	5	6	Công việc
Khởi tạo	0	oo	oo	oo	oo	oo	Khởi tạo
1							
2							
3							
4							
5							

Help Clear shift Delete Edit Undo



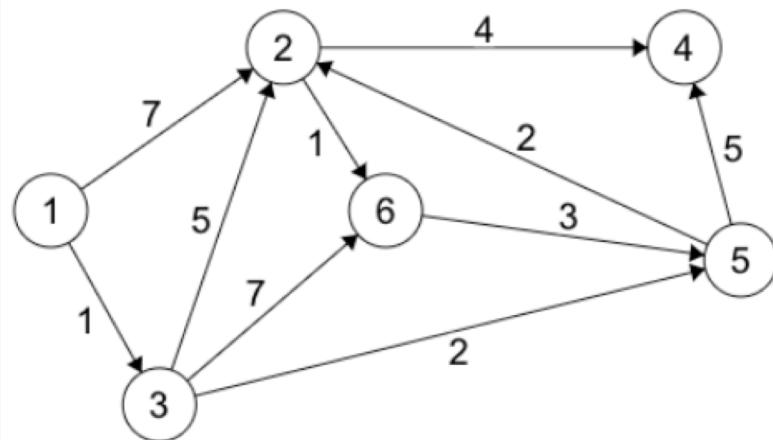
Lặp 1

- Chọn $u = ?$
- Đánh dấu nó
- Xét các đỉnh kề: ???

1. Áp dụng thuật toán Moore - Dijkstra và ghi kết quả vào bảng

	1	2	3	4	5	6	Công việc
Khởi tạo	0	oo	oo	oo	oo	oo	Khởi tạo
1							
2							
3							
4							
5							

Help Clear shift Delete Edit Undo



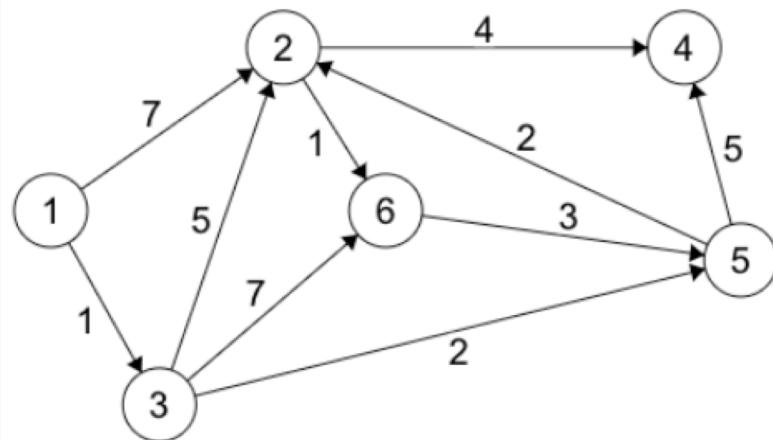
Lặp 1

- Chọn $u = 1$
- Đánh dấu nó
- Xét các đỉnh kề: 2, 3

1. Áp dụng thuật toán Moore - Dijkstra và ghi kết quả vào bảng

	1	2	3	4	5	6	Công việc
Khởi tạo	0	oo	oo	oo	oo	oo	Khởi tạo
1	*	7/1	1/1				Chọn $u = 1$ C/N: 2, 3
2							
3							
4							
5							

Help Clear shift Delete Edit Undo



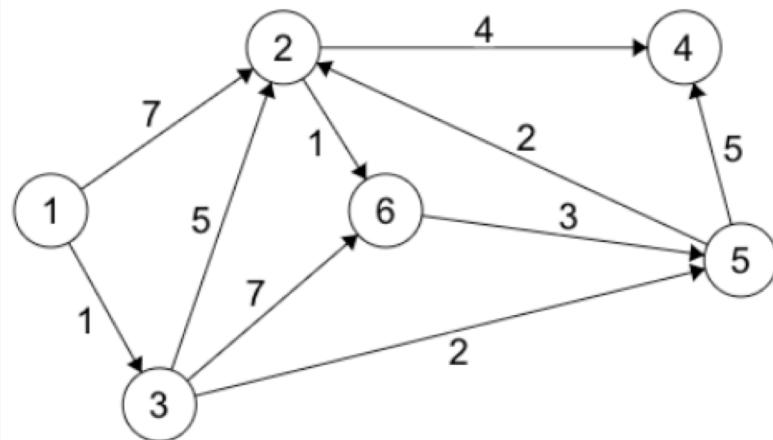
Lặp 2

- Chọn $u = ?$
- Đánh dấu nó
- Xét các đỉnh kề: ???

1. Áp dụng thuật toán Moore - Dijkstra và ghi kết quả vào bảng

	1	2	3	4	5	6	Công việc
Khởi tạo	0	oo	oo	oo	oo	oo	Khởi tạo
1	*	7/1	1/1				Chọn $u = 1$ C/N: 2, 3
2							
3							
4							
5							

Help Clear shift Delete Edit Undo



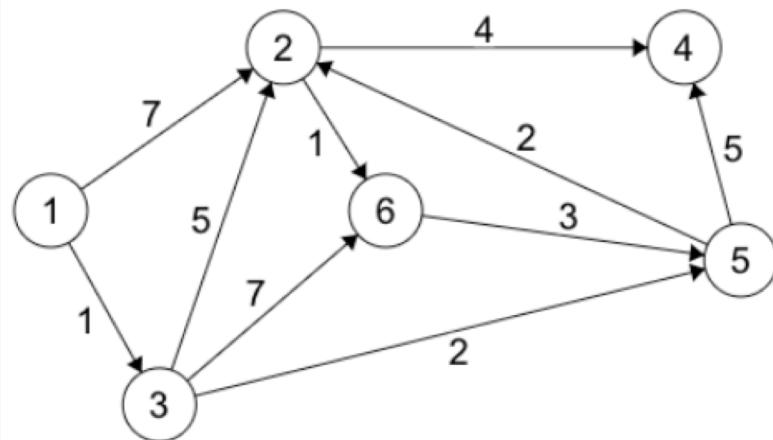
Lặp 2

- Chọn $u = 3$
- Đánh dấu nó
- Xét các đỉnh kề: 2, 5, 6

1. Áp dụng thuật toán Moore - Dijkstra và ghi kết quả vào bảng

	1	2	3	4	5	6	Công việc
Khởi tạo	0	oo	oo	oo	oo	oo	Khởi tạo
1	*	7/1	1/1				Chọn $u = 1$ C/N: 2, 3
2		6/3	*		3/3	8/3	Chọn $u = 3$ C/N: 2, 5, 6
3							
4							
5							

Help Clear shift Delete Edit Undo



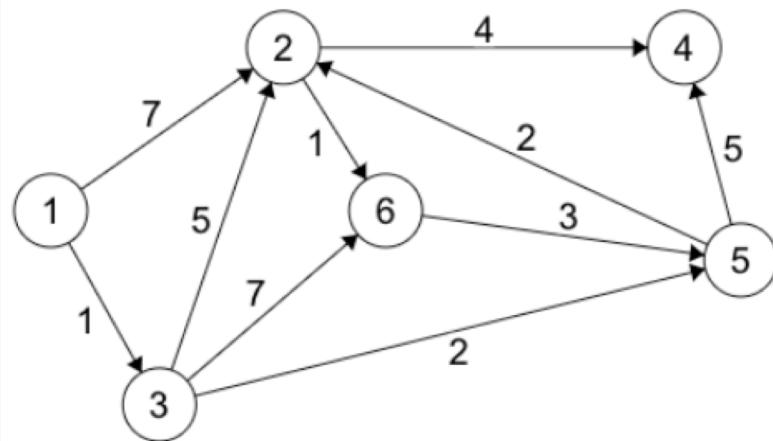
Lặp 3

- Chọn $u = ?$
- Đánh dấu nó
- Xét các đỉnh kề: ???

1. Áp dụng thuật toán Moore - Dijkstra và ghi kết quả vào bảng

	1	2	3	4	5	6	Công việc
Khởi tạo	0	oo	oo	oo	oo	oo	Khởi tạo
1	*	7/1	1/1				Chọn $u = 1$ C/N: 2, 3
2		6/3	*		3/3	8/3	Chọn $u = 3$ C/N: 2, 5, 6
3							
4							
5							

Help Clear shift Delete Edit Undo



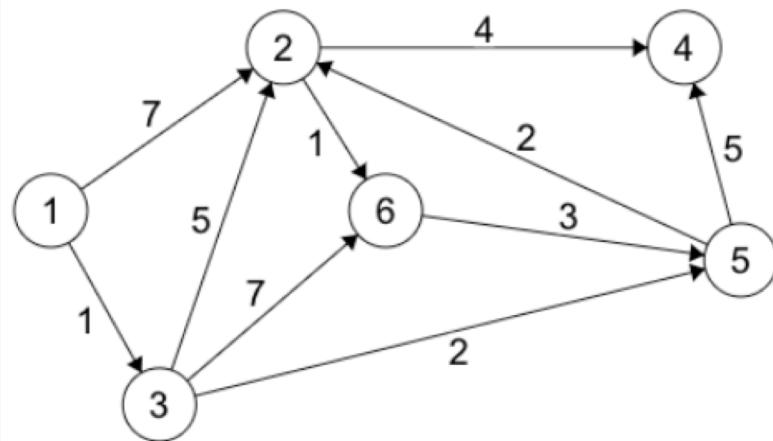
Lặp 2

- Chọn $u = 5$
- Đánh dấu nó
- Xét các đỉnh kề: 2, 4

1. Áp dụng thuật toán Moore - Dijkstra và ghi kết quả vào bảng

	1	2	3	4	5	6	Công việc
Khởi tạo	0	oo	oo	oo	oo	oo	Khởi tạo
1	*	7/1	1/1				Chọn $u = 1$ C/N: 2, 3
2		6/3	*		3/3	8/3	Chọn $u = 3$ C/N: 2, 5, 6
3		5/5		8/5	*		Chọn $u = 5$ C/N: 2, 4
4							
5							

Help Clear shift Delete Edit Undo



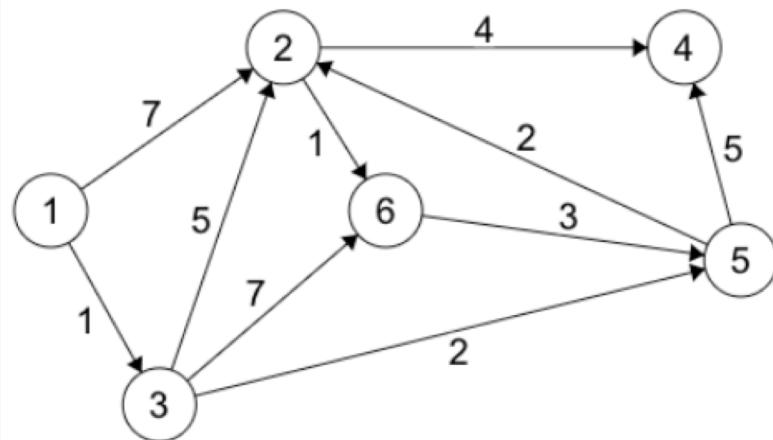
Lặp 4

- Chọn $u = ?$
- Đánh dấu nó
- Xét các đỉnh kề: ???

1. Áp dụng thuật toán Moore - Dijkstra và ghi kết quả vào bảng

	1	2	3	4	5	6	Công việc
Khởi tạo	0	oo	oo	oo	oo	oo	Khởi tạo
1	*	7/1	1/1				Chọn $u = 1$ C/N: 2, 3
2		6/3	*		3/3	8/3	Chọn $u = 3$ C/N: 2, 5, 6
3		5/5		8/5	*		Chọn $u = 5$ C/N: 2, 4
4							
5							

Help Clear shift Delete Edit Undo



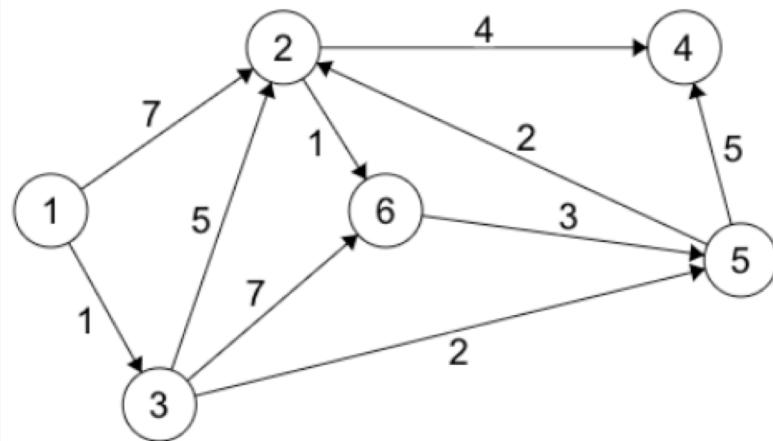
Lặp 4

- Chọn $u = 2$
- Đánh dấu nó
- Xét các đỉnh kề: 6

1. Áp dụng thuật toán Moore - Dijkstra và ghi kết quả vào bảng

	1	2	3	4	5	6	Công việc
Khởi tạo	0	oo	oo	oo	oo	oo	Khởi tạo
1	*	7/1	1/1				Chọn $u = 1$ C/N: 2, 3
2		6/3	*		3/3	8/3	Chọn $u = 3$ C/N: 2, 5, 6
3		5/5		8/5	*		Chọn $u = 5$ C/N: 2, 4
4		*				6/2	Chọn $u = 2$ C/N: 6
5							

Help Clear shift Delete Edit Undo



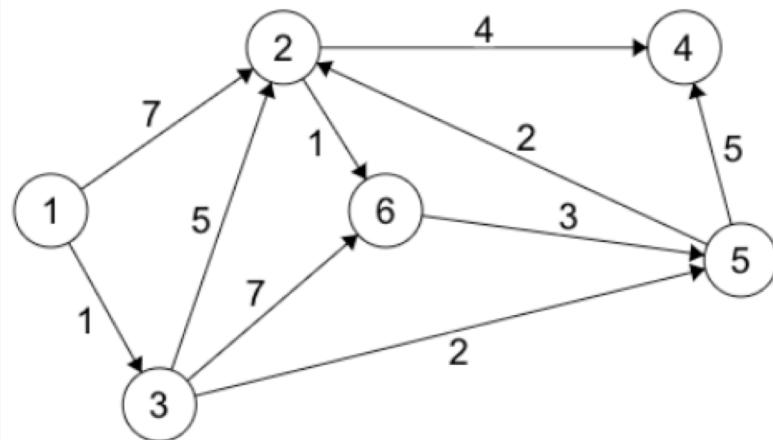
Lặp 5

- Chọn $u = ?$
- Đánh dấu nó
- Xét các đỉnh kề: ???

1. Áp dụng thuật toán Moore - Dijkstra và ghi kết quả vào bảng

	1	2	3	4	5	6	Công việc
Khởi tạo	0	oo	oo	oo	oo	oo	Khởi tạo
1	*	7/1	1/1				Chọn $u = 1$ C/N: 2, 3
2		6/3	*		3/3	8/3	Chọn $u = 3$ C/N: 2, 5, 6
3		5/5		8/5	*		Chọn $u = 5$ C/N: 2, 4
4		*				6/2	Chọn $u = 2$ C/N: 6
5							

Help Clear shift Delete Edit Undo



Lặp 5

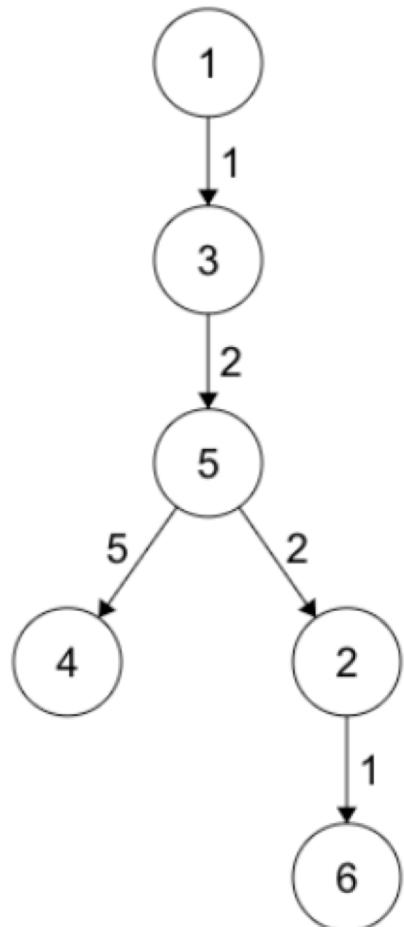
- Chọn $u = 6$
- Đánh dấu nó
- Xét các đỉnh kề: 5

1. Áp dụng thuật toán Moore - Dijkstra và ghi kết quả vào bảng

	1	2	3	4	5	6	Công việc
Khởi tạo	0	oo	oo	oo	oo	oo	Khởi tạo
1	*	7/1	1/1				Chọn $u = 1$ C/N: 2, 3
2		6/3	*		3/3	8/3	Chọn $u = 3$ C/N: 2, 5, 6
3		5/5		8/5	*		Chọn $u = 5$ C/N: 2, 4
4		*				6/2	Chọn $u = 2$ C/N: 6
5						*	Chọn $u = 6$ C/N:

2. Vẽ cây đường đi ngắn nhất

Help Clear shift Delete Edit Undo



Vẽ cây đường đi ngắn nhất

- Dựa vào $p[u]$ và đồ thị gốc
- Gồm các cung $(p[u], u)$

	Test	Got	
✓	1. Thuật toán Moore – Dijkstra (80%) 2. Cây đường đi ngắn nhất (20%)	1. Kiểm tra áp dụng thuật toán Moore – Dijkstra ✓ + Bước khởi tạo: - [I] Tất cả các bước con đều đúng. + Bước 1: - [I] Tất cả các bước con đều đúng. + Bước 2: - [I] Tất cả các bước con đều đúng. + Bước 3: - [I] Tất cả các bước con đều đúng. + Bước 4: - [I] Tất cả các bước con đều đúng. + Bước 5: - [I] Tất cả các bước con đều đúng. Tổng (1): 18/18. 2. Kiểm tra cây đường đi ngắn nhất - [I] Cây đường đi ngắn nhất okie. Tổng (2): 11/11.	

Passed all tests! ✓

Bài tập

Tìm đường đi ngắn nhất từ A đến các đỉnh khác trên đồ thị vô hướng có ma trận trọng số như bên cạnh.

	A	B	C	D	E	F	G	H	K
A		1	1						4
B							9	2	3
C				7					2
D					1	5		4	3
E						2			
F							9	3	
G								5	
H									7
K									