HOA SEN UNIVERSITY

www.hoasen.edu.vn

Cấu Trúc Dữ Liệu & Giải Thuật

Data Structures & Algorithms

GV: Phan Hồng Trung

HOA SEN UNIVERSITY

www.hoasen.edu.vn

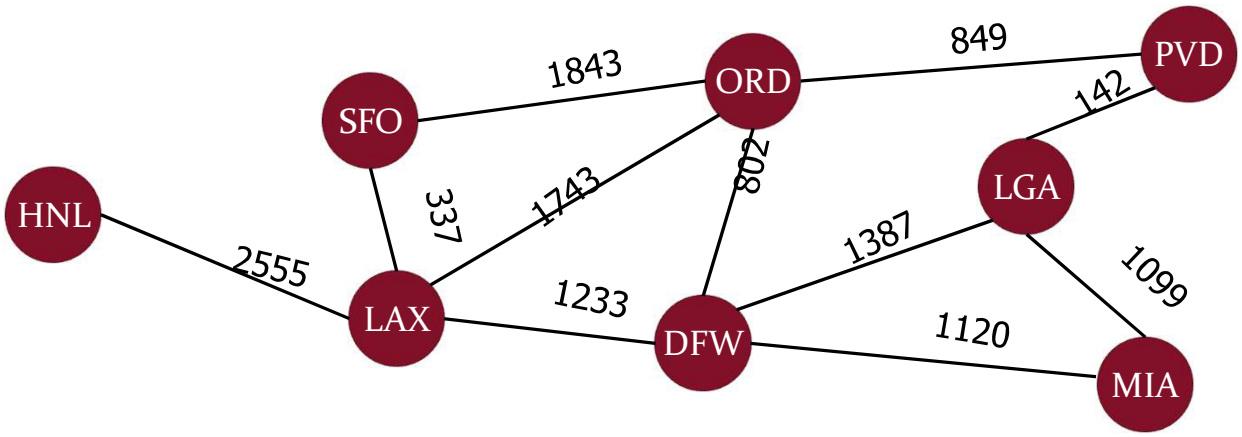
Bài 10 – Đồ Thị (Graph)

- Định nghĩa
- Thuật ngữ
- Ứng dụng
- Biểu diễn
- Duyệt đồ thị
- Tìm đường đi ngắn nhất

Định nghĩa đồ thị

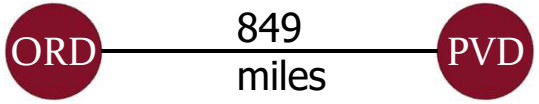
- Đồ thị là một cặp (V, E), trong đó:
  - V là một tập hợp các nút, được gọi là các đỉnh
  - E là một tập hợp các cặp đỉnh, được gọi là các cạnh
- Có thể lưu trữ dữ liệu ở đỉnh và cạnh

- Ví dụ:
  - Một đỉnh biểu diễn một sân bay và lưu trữ mã sân bay gồm ba chữ cái
  - Một cạnh biểu diễn một tuyến bay giữa hai sân bay và lưu trữ số dặm của tuyến bay đó



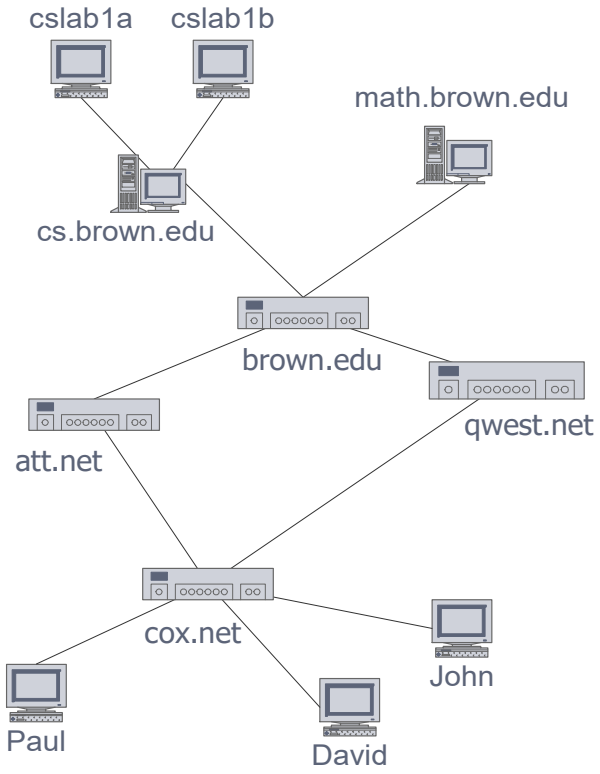
Loại cạnh

- Cạnh có hướng:
  - Cặp đỉnh có thứ tự (u,v)
  - Đỉnh đầu tiên u là đỉnh nguồn
  - Đỉnh thứ hai v là đỉnh đích
  - Ví dụ: chuyến bay
- Cạnh không có hướng:
  - Cặp đỉnh không có thứ tự (u,v)
  - Ví dụ: tuyến bay
- Đồ thị có hướng:
  - Tất cả các cạnh đều có hướng
  - Ví dụ: mạng lưới tuyến đường
- Đồ thị vô hướng:
  - Tất cả các cạnh đều vô hướng
  - Ví dụ: mạng lưới chuyến bay



Ứng dụng

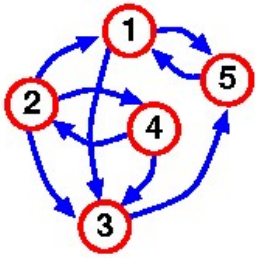
- Mạch điện tử:
  - Bảng mạch in
  - Mạch tích hợp
- Mạng lưới giao thông:
  - Mạng lưới đường cao tốc
  - Mạng lưới hàng không
- Mạng lưới máy tính:
  - Mạng cục bộ
  - Internet
  - Web
- Cơ sở dữ liệu:
  - Sơ đồ quan hệ - thực thể (ER)



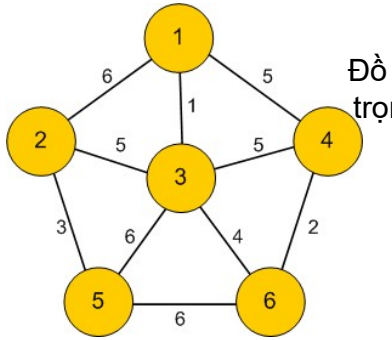
Ví dụ

Đồ thị đơn vô hướng

Đồ thị đơn giản vì nó không có nhiều cạnh và không có vòng lặp

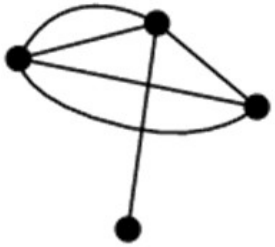


Đồ thị đơn có hướng

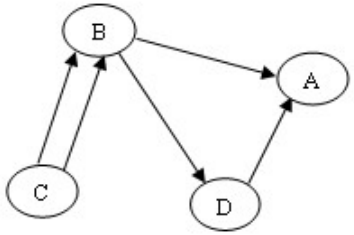


Đồ thị có trọng số

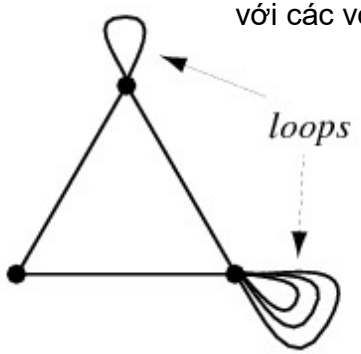
Một đồ thị vô hướng có nhiều cạnh



Một đồ thị có hướng có nhiều cạnh

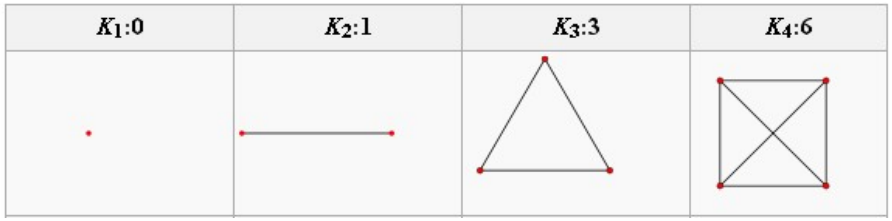


Một đồ thị vô hướng với các vòng lặp



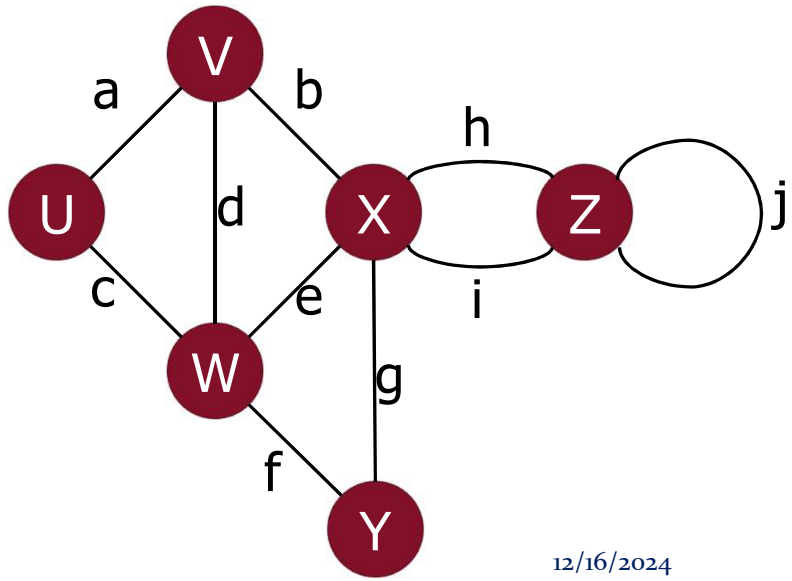
Thuật ngữ

- Một đồ thị đầy đủ (A complete graph) là một đồ thị mà mọi cặp đỉnh đều được kết nối bằng một cạnh.
- Một đồ thị đầy đủ đơn giản trên n đỉnh có n đỉnh và n(n-1)/2 cạnh, và được ký hiệu là Kn.
- Một đồ thị G'=(V', E') là một đồ thị con của một đồ thị G=(V, E) nếu $V' \subseteq V$ và $E' \subseteq E$.



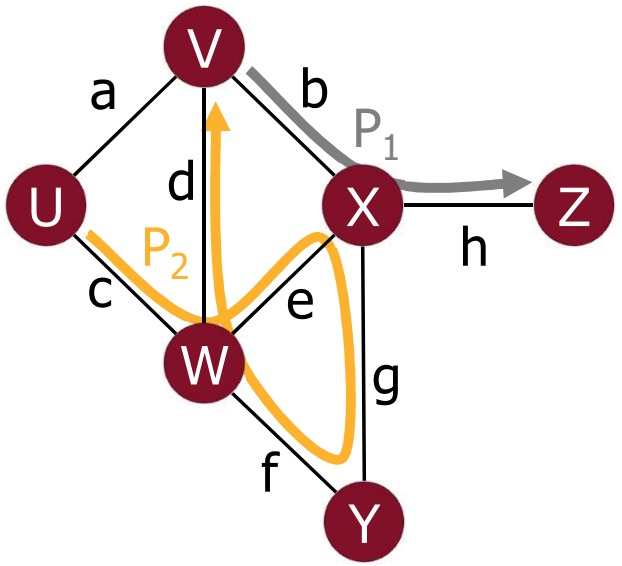
Thuật ngữ

- U và V là các đỉnh cuối (end vertices) của cạnh a.
- a, d và b là các cạnh tiếp giáp (incident edges) với V.
- U và V là các đỉnh kề nhau (adjacent vertices)
- X có bậc (degree) là 5
- h và i là các cạnh song song (parallel edges)
- j là một vòng lặp (self-loop)



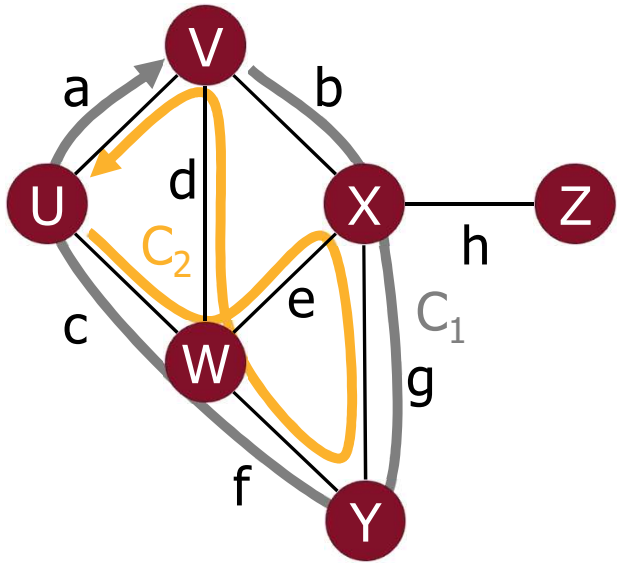
Thuật ngữ

- Đường dẫn (path)
  - Chuỗi các đỉnh và cạnh xen kẽ
  - Cắt đầu bằng một đỉnh
  - Kết thúc bằng một đỉnh
  - Mỗi cạnh bắt đầu và kết thúc bởi các đỉnh cuối của nó
- Đường dẫn đơn giản
  - Đường dẫn sao cho tất cả các đỉnh và cạnh của nó là phân biệt
- Ví dụ
  - P1=(V,b,X,h,Z) là một đường dẫn đơn giản
  - P2=(U,c,W,e,X,g,Y,f,W,d,V) là một đường dẫn không đơn giản



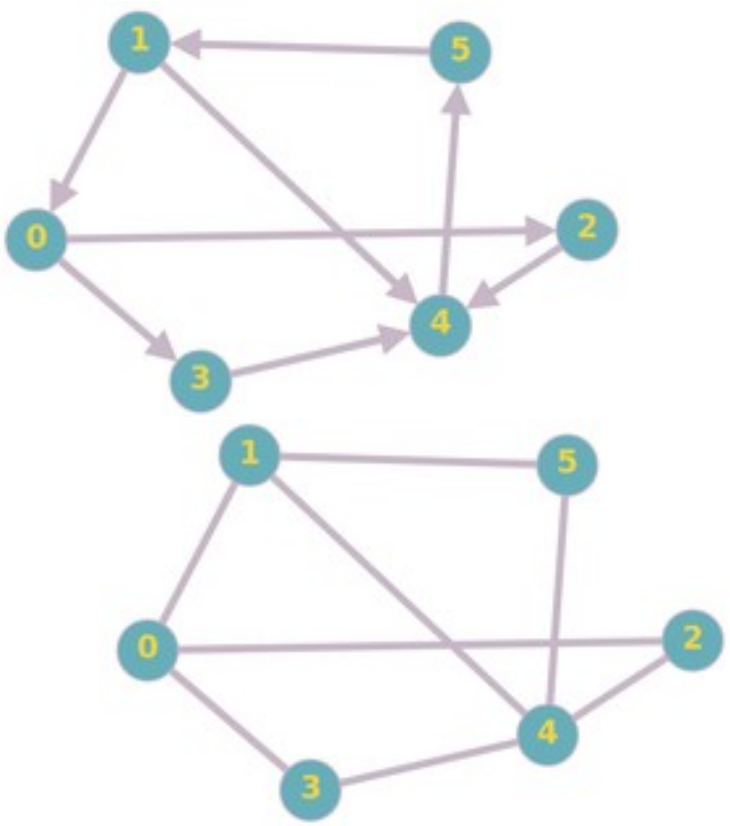
Thuật ngữ

- Chu trình
  - Chuỗi khép kín các đỉnh và cạnh xen kẽ
  - Mỗi cạnh bắt đầu và kết thúc bởi các đỉnh cuối của nó
- Chu trình đơn giản
  - Chu trình mà tất cả các đỉnh và cạnh của nó là phân biệt
- Ví dụ
  - C1=(V,b,X,g,Y,f,W,c,U,a) là một chu trình đơn giản
  - C2=(U,c,W,e,X,g,Y,f,W,d,V,a) là một chu trình không đơn giản



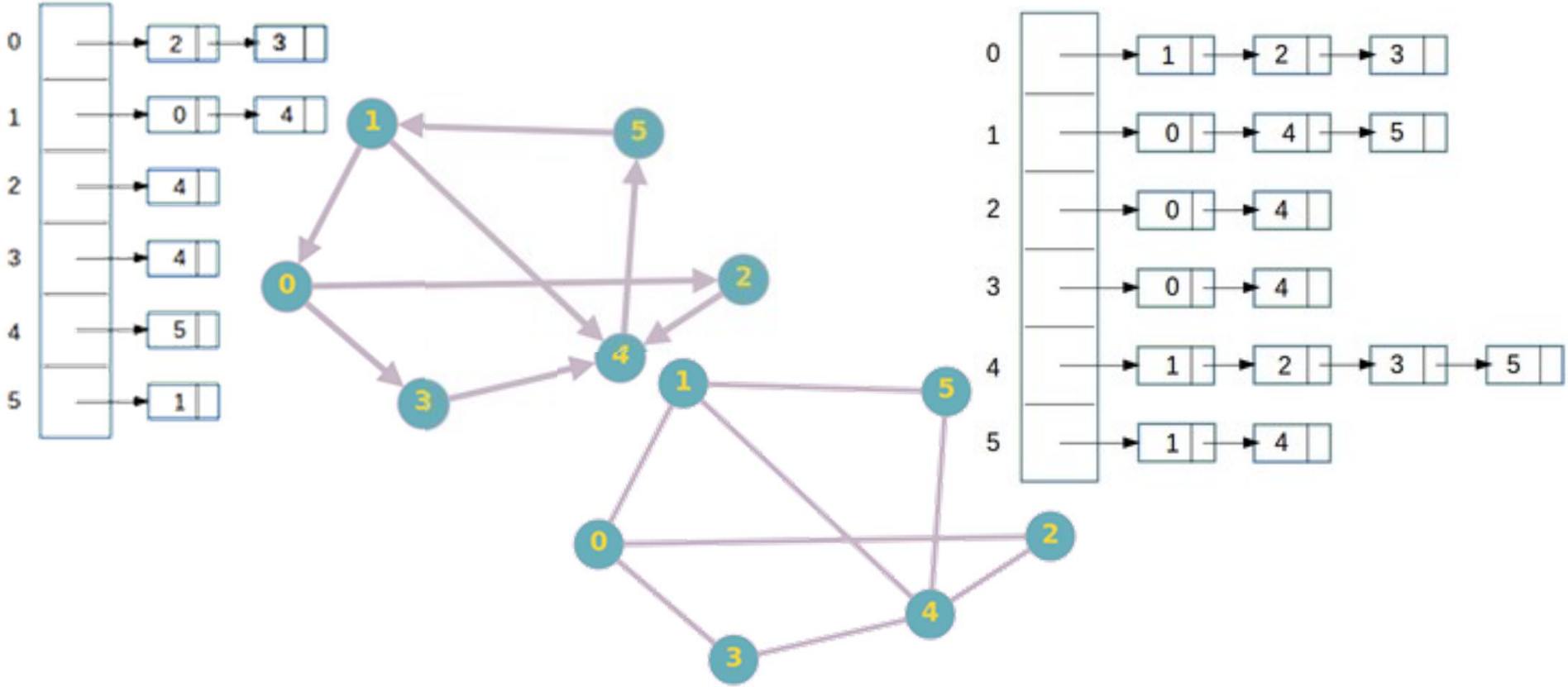
Ma trận kề (Adjacency matrix)

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



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

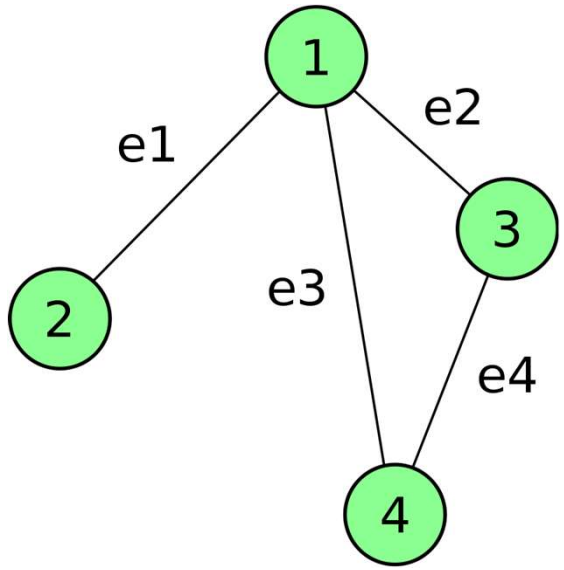
Danh sách kề (Adjacency list)



Ma trận tiếp xúc (Incidence matrix)

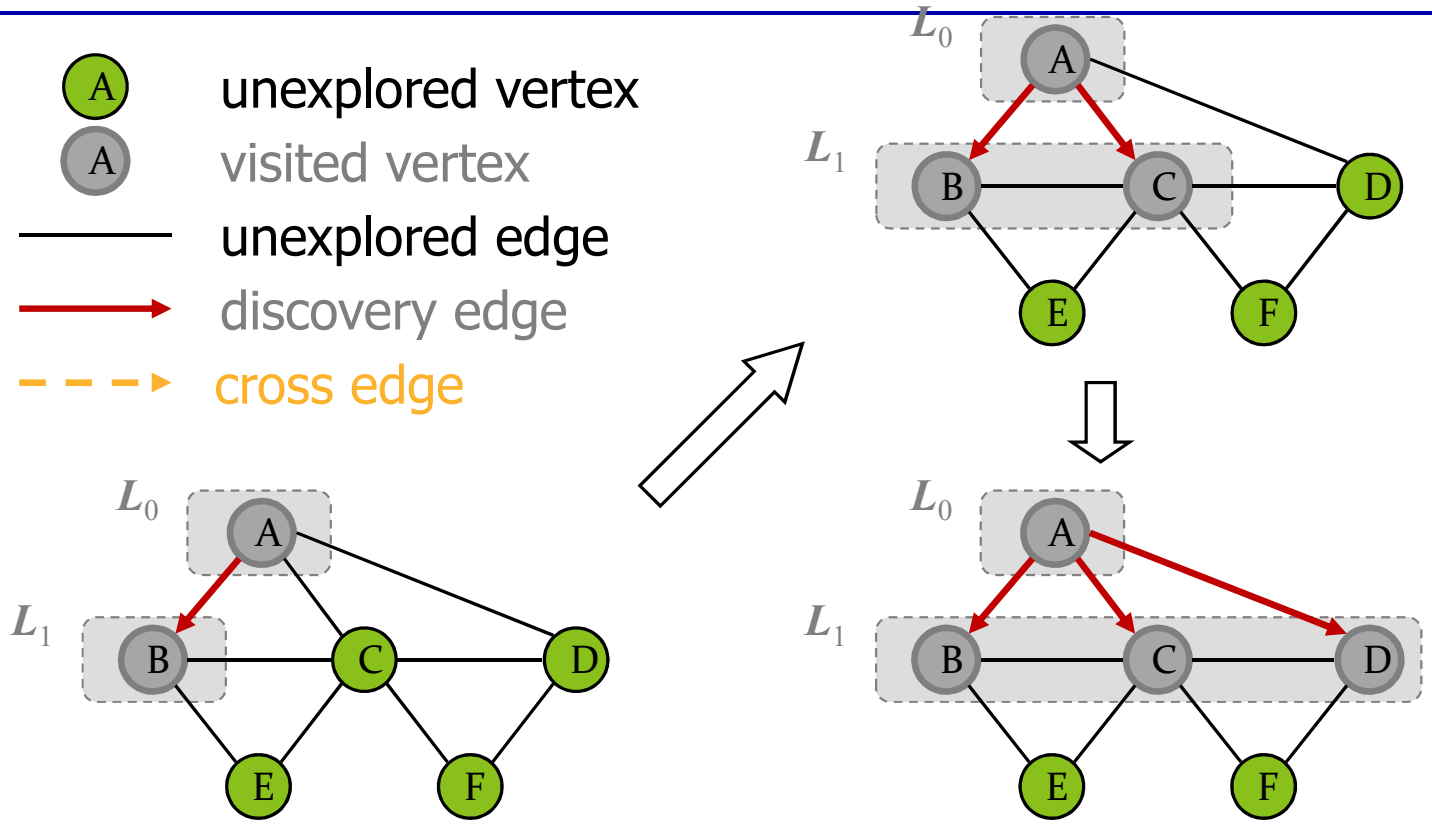
	e_1	e_2	e_3	e_4
1	1	1	1	0
2	1	0	0	0
3	0	1	0	1
4	0	0	1	1

$$= \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

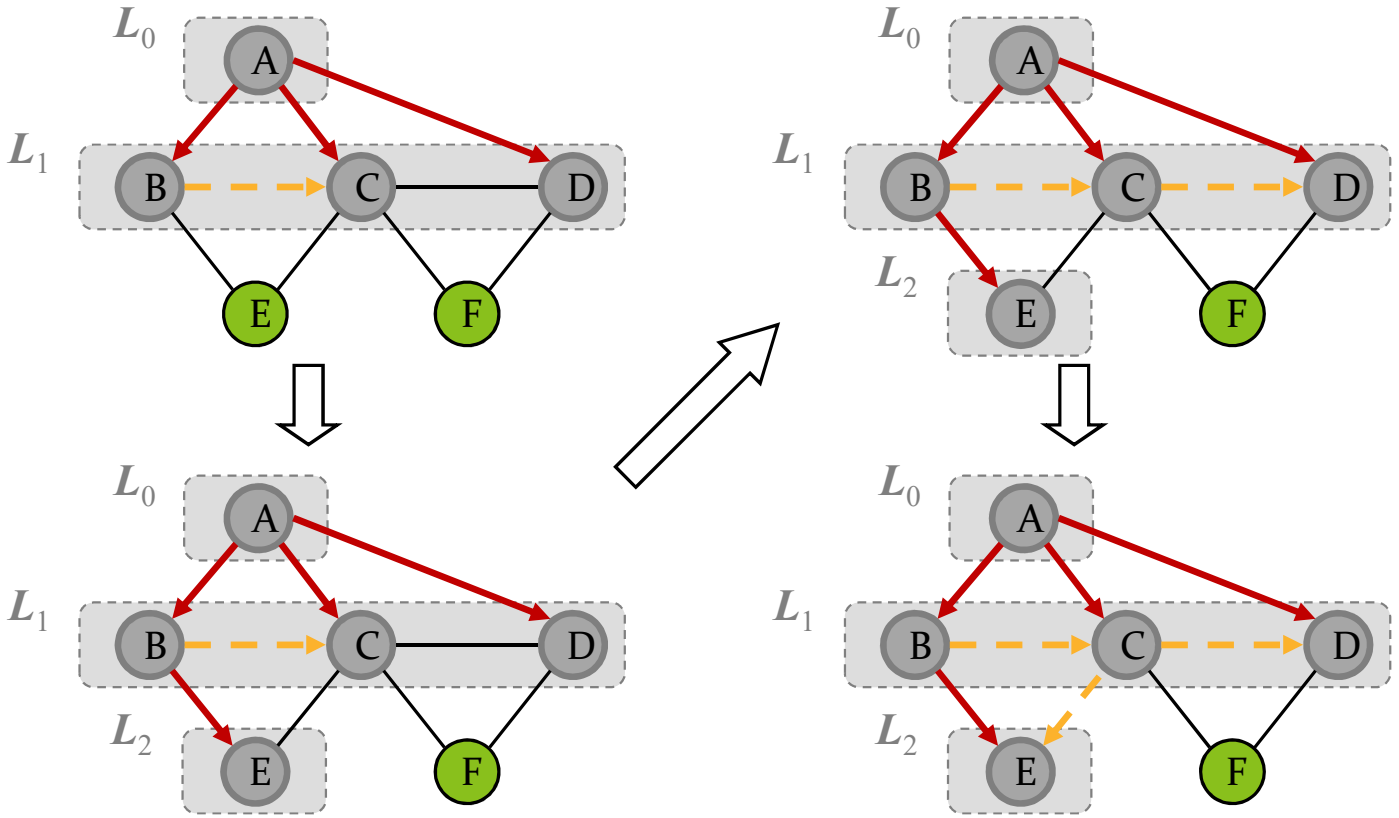


- Một đỉnh được gọi là tiếp xúc với một cạnh nếu nó là đỉnh cuối của cạnh đó.

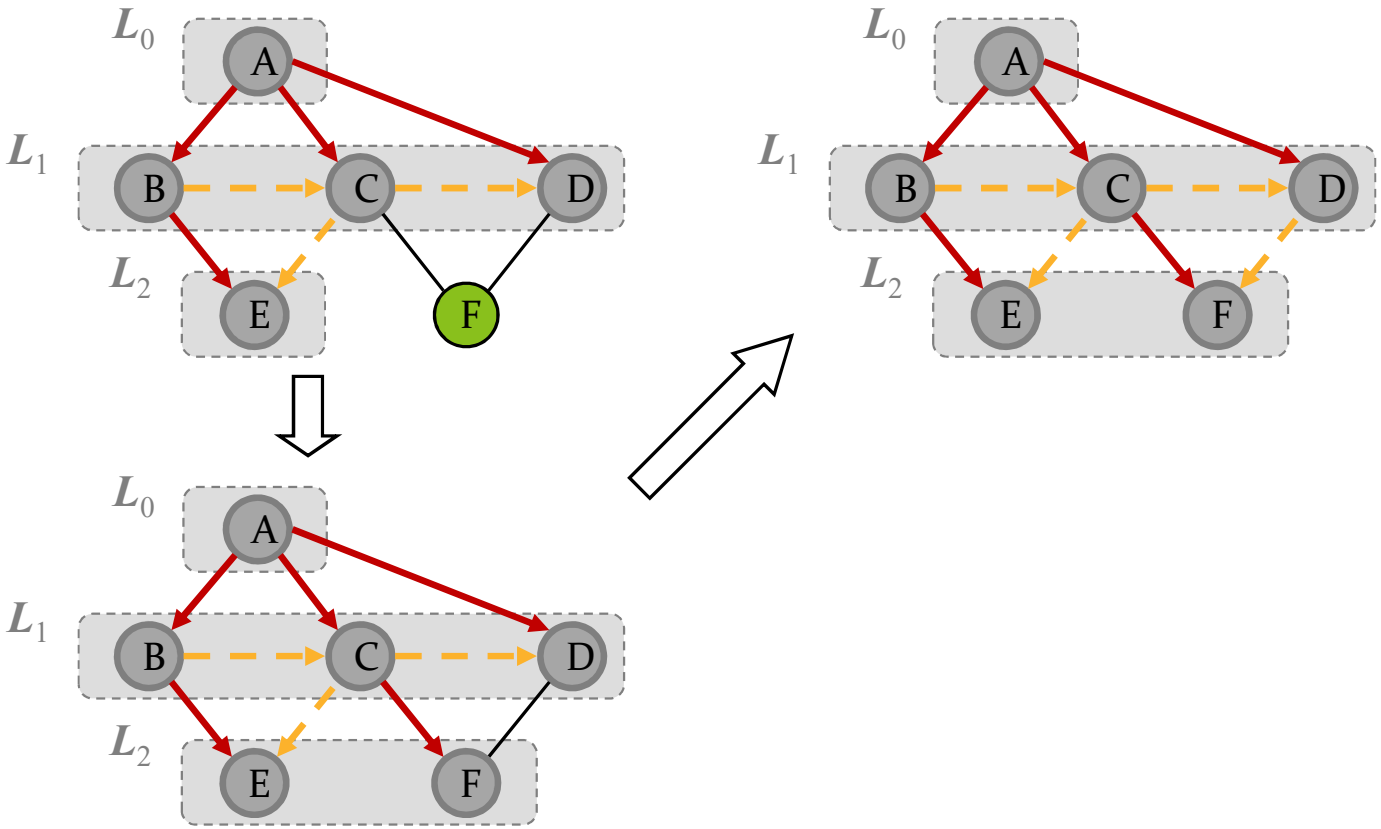
Breadth-first Search



Breadth-first Search

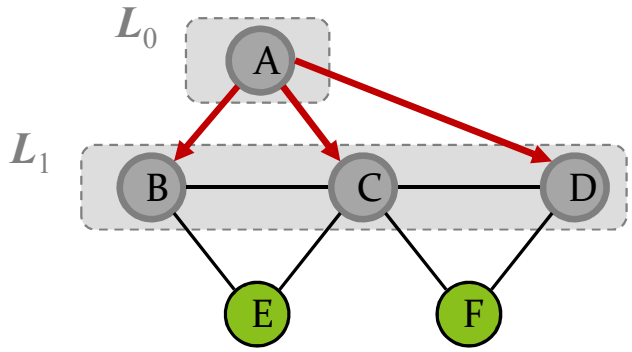


Breadth-first Search

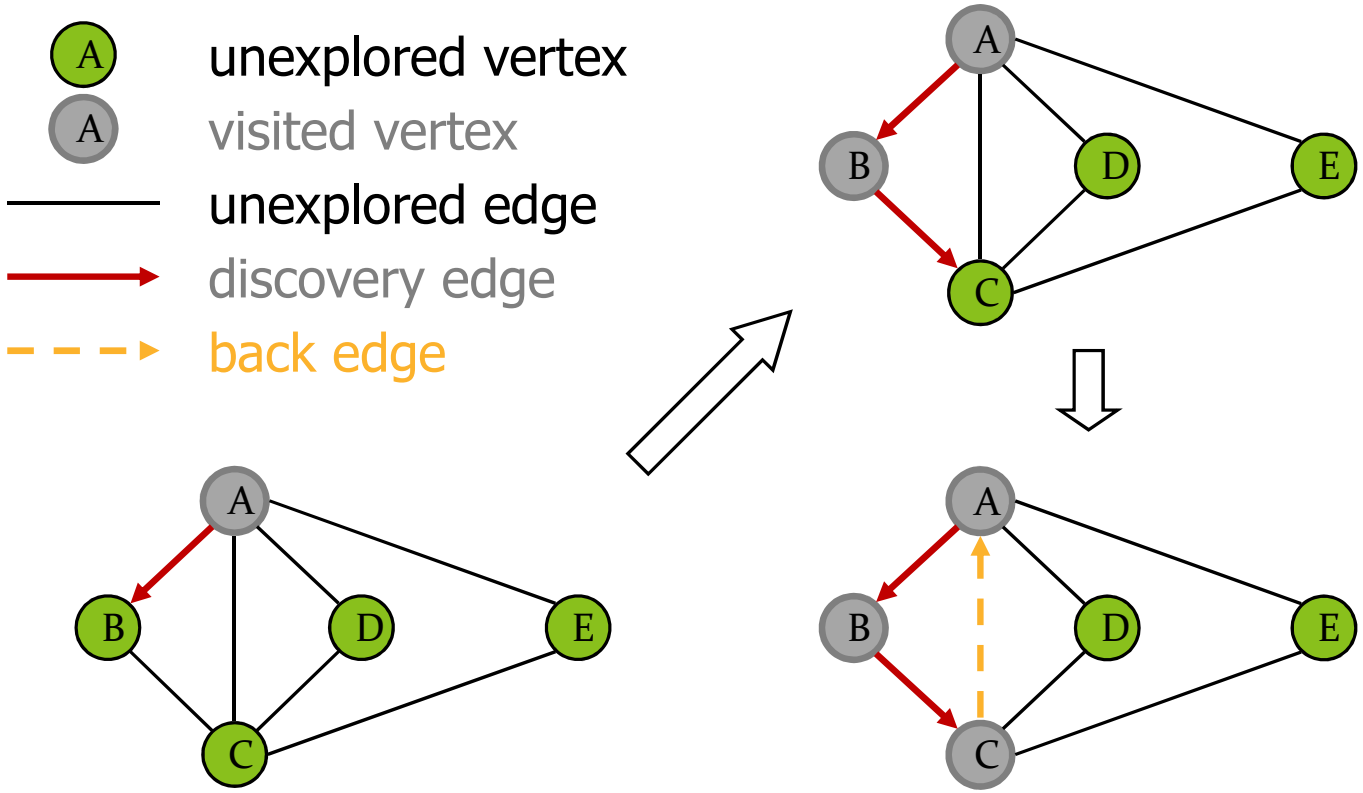


Breadth-first Search

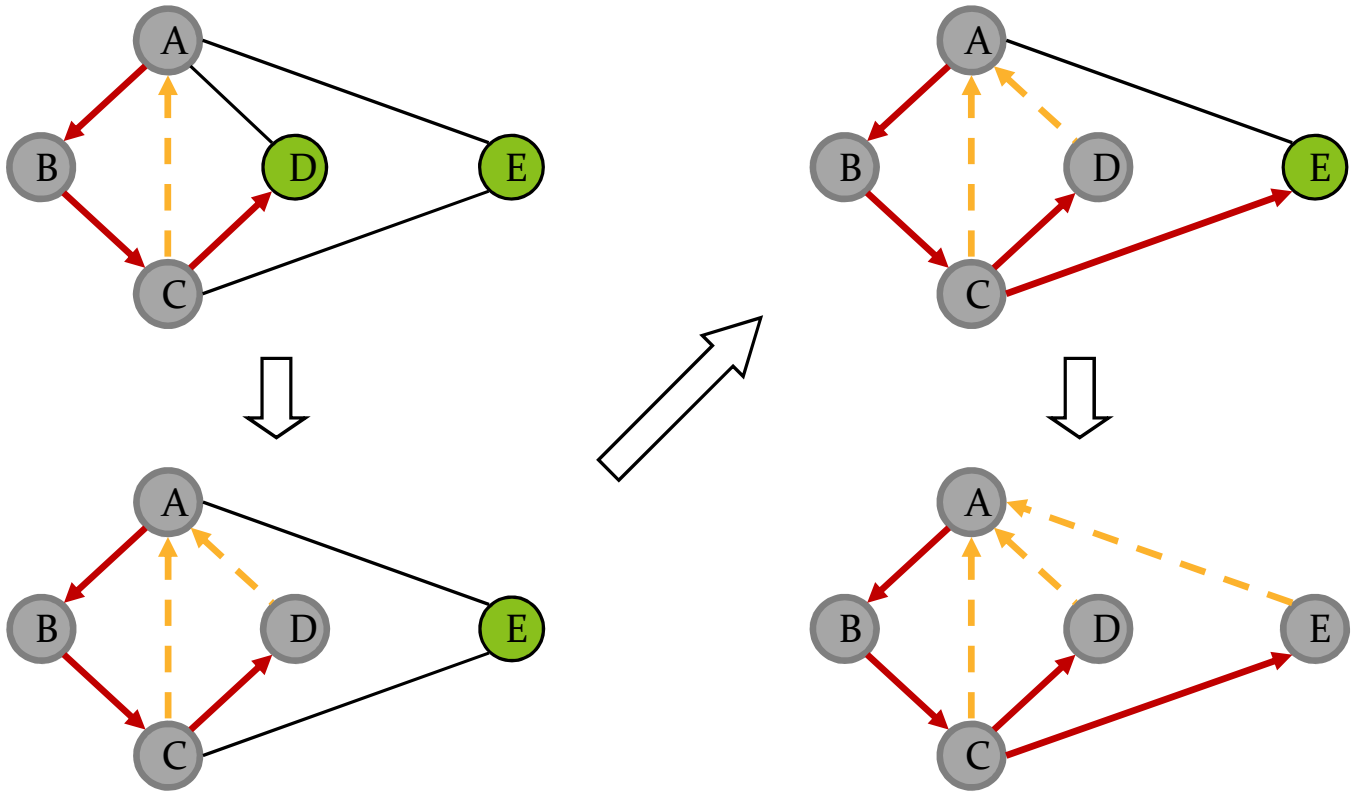
```
1  function List<String> BFS(Graph graph, String root)
2      List<String> visitedList ← new List<>()
3      Queue<String> queue ← new Queue<>()
4       visitedList .add(root)
5      queue.enqueue(root)
6      while (queue is not empty) do
7          String vertex ← queue.dequeue()
8          for (Vertex v in graph. getAdjVertices (vertex)) do
9              if (v. label not in visitedList ) then
10                 visitedList .add(v. label )
11                 queue.enqueue(v. label )
12             end if
13         end for
14     end while
15     return visitedList
16 end function
```



Depth-first Search

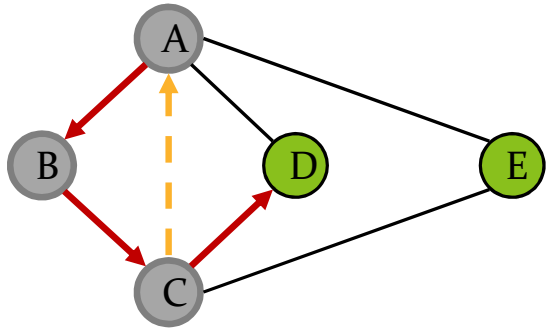


Depth-first Search



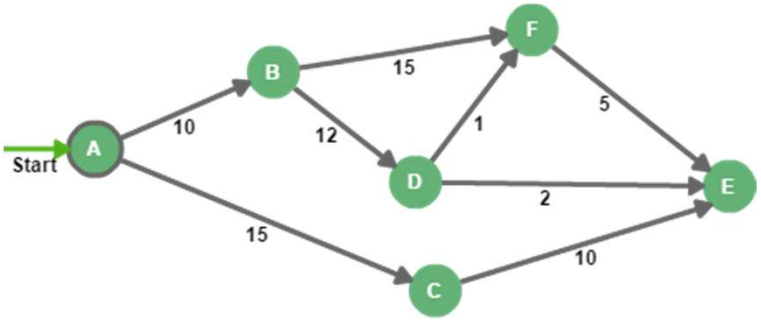
Depth-first Search

```
1  function List<String> DFS(Graph graph, String root)
2      List<String> visitedList ← new List<>()
3      Stack<String> stack ← new Stack<>()
4      stack.push(root)
5      while (stack is not empty) do
6          String vertex ← stack.pop()
7          if (vertex not in visitedList ) then
8              visitedList .add(v. label )
9              for (Vertex v in graph. getAdjVertices (vertex)) do
10                 stack.push(v. label )
11             end for
12         end if
13     end while
14     return visitedList
15 end function
```



Dijkstra Algorithm

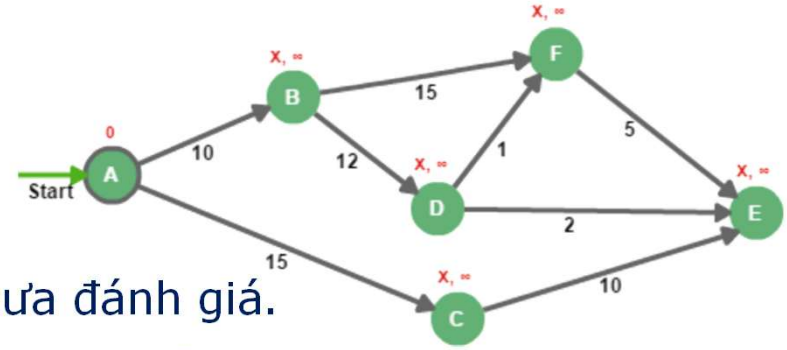
- Với một đồ thị có trọng số dương và một đỉnh bắt đầu A, Dijkstra xác định đường đi ngắn nhất và khoảng cách từ A đến tất cả các đỉnh đích trong đồ thị.



- Ý tưởng cốt lõi của thuật toán Dijkstra là liên tục loại bỏ các đường đi dài hơn giữa nút bắt đầu và tất cả các đỉnh đích có thể có.
- Để theo dõi, chúng ta dùng hai tập hợp đỉnh riêng biệt, đã đánh giá và chưa đánh giá.
- Các đỉnh đã đánh giá là các đỉnh đã tính khoảng cách tối thiểu từ đỉnh bắt đầu.
- Các đỉnh chưa đánh giá là các đỉnh có thể đi đến từ đỉnh bắt đầu, nhưng chưa tính khoảng cách tối thiểu từ đỉnh bắt đầu.

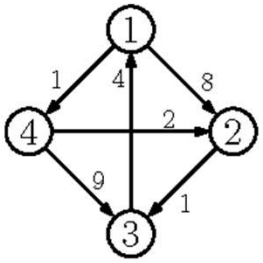
Dijkstra Algorithm

- Thuật toán Dijkstra:
  - Đặt khoảng cách đến startNode bằng 0.
  - Đặt các khoảng cách khác bằng vô cực.
  - Thêm startNode vào tập hợp các đỉnh chưa đánh giá.
  - Trong khi tập hợp các đỉnh chưa đánh giá khác rỗng:
    - Từ tập hợp các đỉnh chưa đánh giá, lấy ra đỉnh có khoảng cách nhỏ nhất tính từ đỉnh bắt đầu để đánh giá.
    - Tính khoảng cách mới đến các đỉnh kề của đỉnh đang đánh giá và giữ lại khoảng cách nhỏ nhất.
    - Thêm các đỉnh kề của đỉnh đang đánh giá vào tập hợp các nút chưa đánh giá.

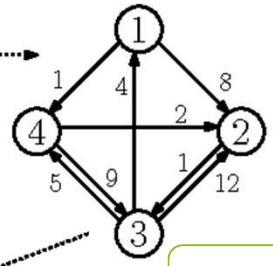


Floyd-Warshall Algorithm

```
1  function int[,] STP(G(V,E))
2      foreach (v,u) ∈ V × V do
3          d[v,u] ← weight(v,u)
4      n ← cardinality(V)
5      for k ← 1 to n do
6          for i ← 1 to n do
7              for j ← 1 to n do
8                  if d[i,j] > d[i,k] + d[k,j] then
9                      d[i,j] ← d[i,k] + d[k,j]
10     return d
11 end function
```

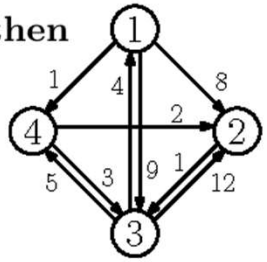


$$d^{(0)} = \begin{bmatrix} 0 & 8 & \infty & 1 \\ \infty & 0 & 1 & \infty \\ 4 & \infty & 0 & \infty \\ \infty & 2 & 9 & 0 \end{bmatrix}$$

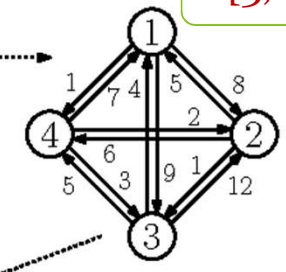


$$d^{(1)} = \begin{bmatrix} 0 & 8 & \infty & 1 \\ \infty & 0 & 1 & \infty \\ 4 & 12 & 0 & 5 \\ \infty & 2 & 9 & 0 \end{bmatrix}$$

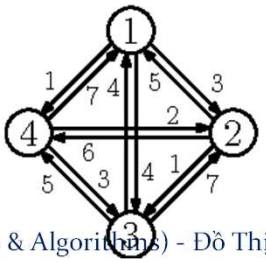
d[3,2] > d[3,1] + d[1,2]



$$d^{(2)} = \begin{bmatrix} 0 & 8 & 9 & 1 \\ \infty & 0 & 1 & \infty \\ 4 & 12 & 0 & 5 \\ \infty & 2 & 3 & 0 \end{bmatrix}$$



$$d^{(3)} = \begin{bmatrix} 0 & 8 & 9 & 1 \\ 5 & 0 & 1 & 6 \\ 4 & 12 & 0 & 5 \\ 7 & 2 & 3 & 0 \end{bmatrix}$$



$$d^{(4)} = \begin{bmatrix} 0 & 3 & 4 & 1 \\ 5 & 0 & 1 & 6 \\ 4 & 7 & 0 & 5 \\ 7 & 2 & 3 & 0 \end{bmatrix}$$

$$\text{final} = \begin{bmatrix} 0 & 3 & 4 & 1 \\ 5 & 0 & 1 & 6 \\ 4 & 7 & 0 & 5 \\ 7 & 2 & 3 & 0 \end{bmatrix}$$

Dijkstra Algorithm vs. Floyd-Warshall Algorithm

- Tìm đường đi ngắn nhất và khoảng cách từ một đỉnh nguồn đến tất cả các đỉnh đích trong đồ thị.
- Không đủ tổng quát vì yêu cầu trọng số dương.
- Độ phức tạp: $O(|V|^2)$

- Tìm đường đi ngắn nhất giữa mọi cặp đỉnh trong đồ thị.
- Đồ thị có thể chứa các cạnh âm nhưng không có chu trình âm.
- Độ phức tạp: $O(|V|^3)$


Q
A