Cấu trúc dữ liệu và giải thuật

TS. Phạm Tuấn Minh

Khoa Công nghệ Thông tin, Đại học Phenikaa

minh.phamtuan@phenikaa-uni.edu.vn

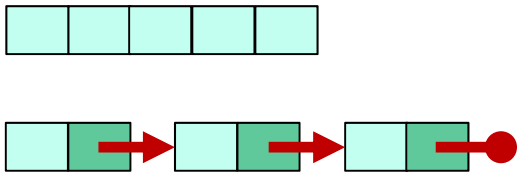
https://sites.google.com/site/phamtuanminh/

Chương 5: Đồ thị

- **Khái niệm và biểu diễn đồ thị**
- Duyệt đồ thị
- Tìm đường đi
- Ứng dụng

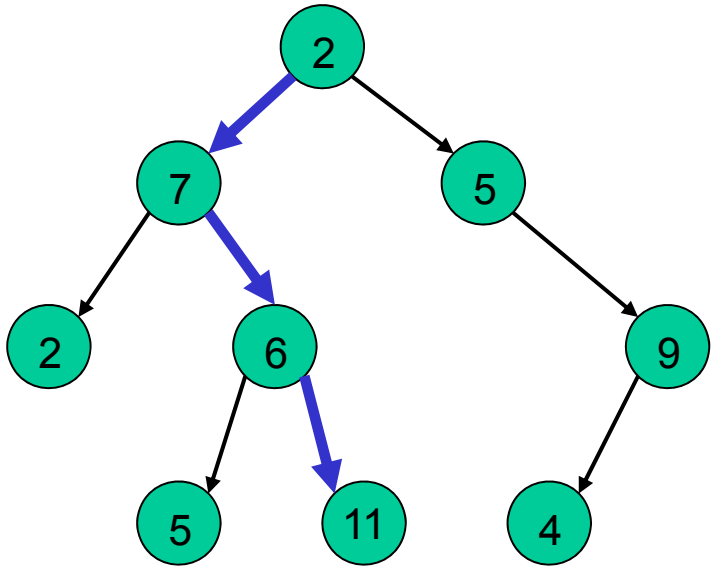
Cấu trúc dữ liệu đã học

- Tuyến tính:
  - Tất cả các phần tử được sắp xếp có thứ tự
  - Truy cập ngẫu nhiên
    - Mảng
  - Truy cập tuần tự
    - Danh sách liên kết
  - Truy cập tuần tự có hạn chế
    - Hàng đợi
    - Ngăn xếp
- Không tuyến tính
  - Cây
  - Đồ thị



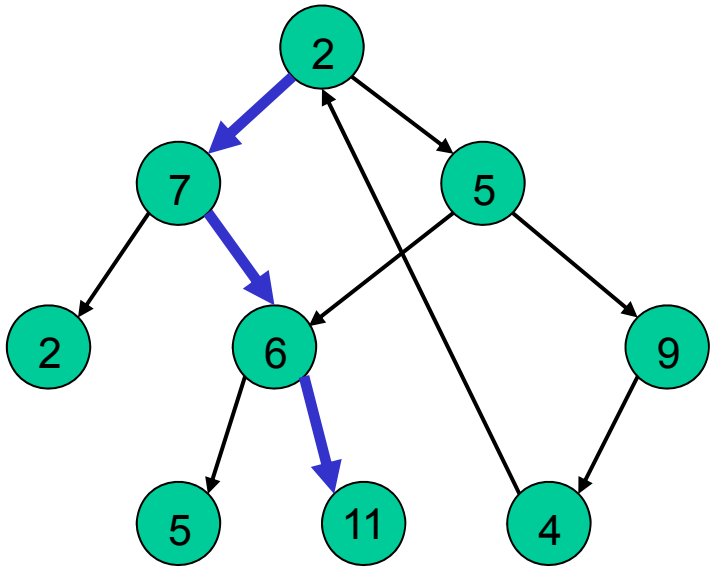
Nhắc lại cấu trúc dữ liệu cây

- Vẫn sử dụng các biểu diễn nút và liên kết
- Đặc điểm mới:
  - Mỗi nút có liên kết tới nhiều hơn một nút khác
  - Không có lặp



Cấu trúc dữ liệu đồ thị

- Vẫn sử dụng các biểu diễn nút và liên kết
- Đặc điểm mới:
  - Mỗi nút có liên kết tới nhiều hơn một nút khác
  - Cho phép có chu trình, các liên kết có thể kết nối giữa bất kì hai nút nào
  - Liên kết có thể có hướng (một chiều) hoặc vô hướng (hai chiều)



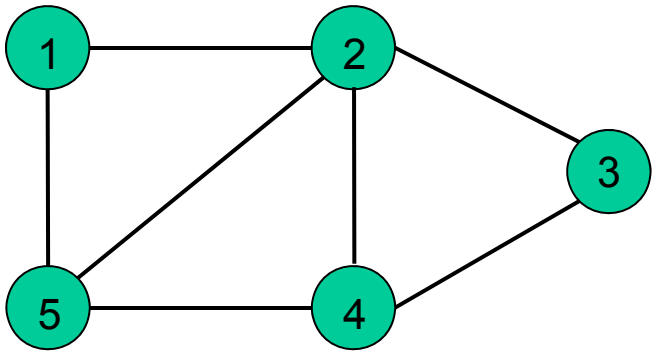
Cây là một trường hợp đặc biệt của đồ thị

Khái niệm đồ thị

- Đồ thị là một cấu trúc dữ liệu bao gồm một tập nút (đỉnh - vertice) và tập cạnh (liên kết - link) mô tả quan hệ giữa các nút
- G = (V, E), trong đó V là tập nút, E là tập các cặp có thứ tự các phần tử của V

Thuật ngữ trong đồ thị

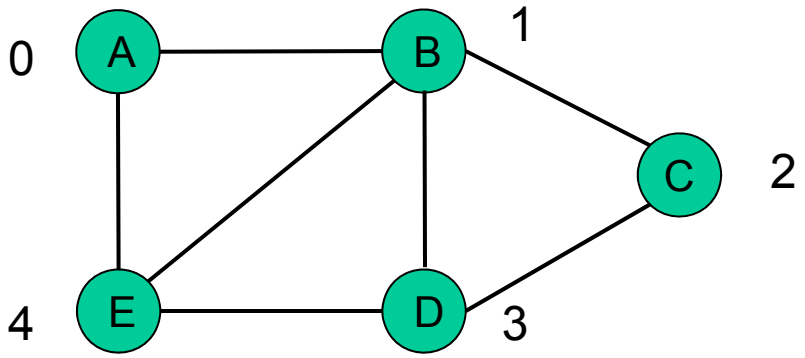
- Nút kề (Adjacent node): Hai nút là kề nhau nếu chúng nối với nhau bởi một cạnh
- Bậc của nút i (Degree): Số nút kề với nút i
- Đường đi (Path): Chuỗi tiếp nối các cạnh nối hai nút trên đồ thị
- Đồ thị liên thông (Connected graph): Đồ thị trong đó có đường đi giữa hai nút bất kì trên đồ thị
- Đồ thị đầy đủ: Đồ thị mà mọi đỉnh có cạnh nối tới mọi đỉnh khác



- Nút 1 kề với nút 2 và 5
- Bậc của nút 1 là 2
- Một đường đi từ nút 1 tới nút 4 là: 1->2->3->4
- Đây là đồ thị liên thông
- Đây không phải là đồ thị đầy đủ

Biểu diễn đồ thị

- Biểu diễn bằng mảng:
  - Một mảng 1 chiều để biểu diễn đỉnh
  - Một mảng 2 chiều (ma trận kề - adjacency matrix) để biểu diễn cạnh



[0]	A
[1]	B
[2]	C
[3]	D
[4]	E

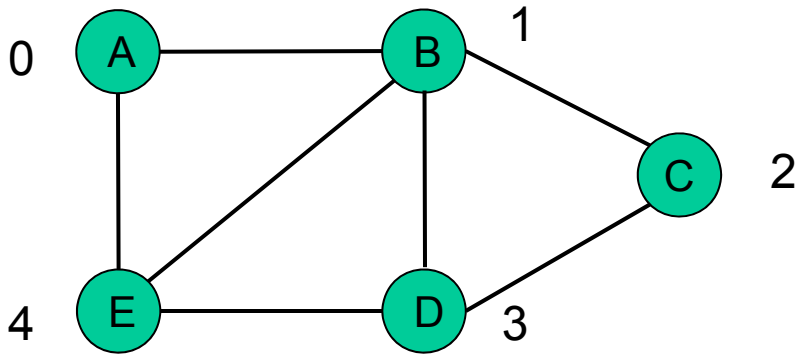
Chứa dữ liệu của đỉnh

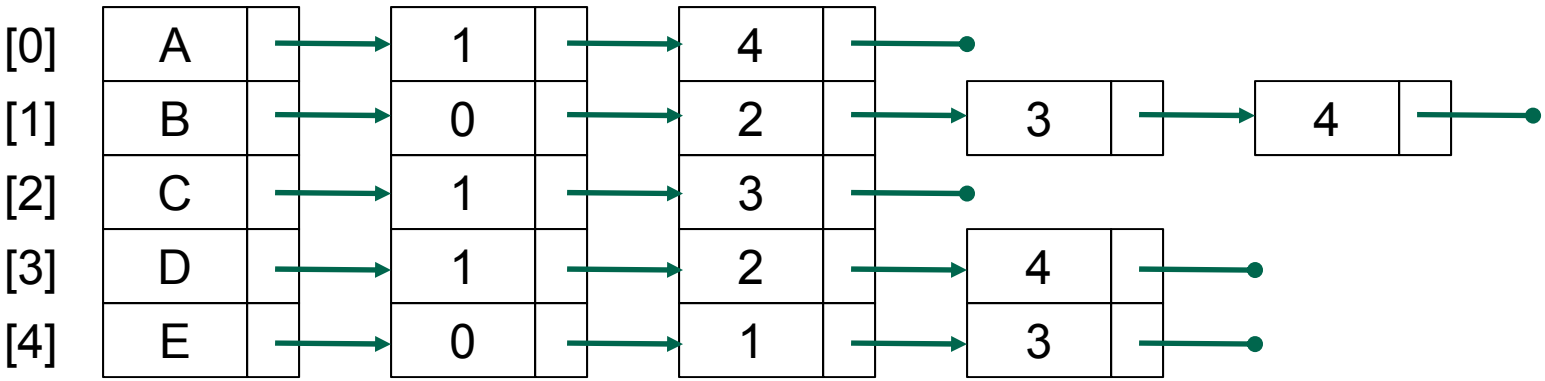
	[0]	[1]	[2]	[3]	[4]
[0]	0	1	0	0	1
[1]	1	0	1	1	1
[2]	0	1	0	1	0
[3]	0	1	1	0	1
[4]	1	1	0	1	0

Ma trận kề

Biểu diễn đồ thị

- Biểu diễn bằng danh sách liên kết:
  - Một mảng 1 chiều để biểu diễn đỉnh
  - Một danh sách cho mỗi đỉnh v, danh sách chứa các đỉnh kề với đỉnh v (danh sách kề)





Chứa dữ liệu của đỉnh

Danh sách kề

Ma trận kề và danh sách kề

- Ma trận kề
  - Tốt với đồ thị dày: |E| ~ O(|V|^2)
  - Yêu cầu bộ nhớ: O(|V| + |E|) = O(|V|^2)
  - Có thể kiểm tra nhanh chóng kết nối giữa hai đỉnh
- Danh sách kề
  - Tốt với đồ thị thưa: |E| ~ O(V)
  - Yêu cầu bộ nhớ: O(|V| + |E|) = O(|V|)
  - Có thể tìm nhanh chóng các đỉnh kề với một đỉnh

Chương 5: Đồ thị

- Khái niệm và biểu diễn đồ thị
- **Duyệt đồ thị**
- Tìm đường đi
- Ứng dụng

Duyệt đồ thị

- Duyệt danh sách: dễ
- Duyệt cây nhị phân: Thứ tự giữa, thứ tự trước, thứ tự sau
- Duyệt đồ thị?

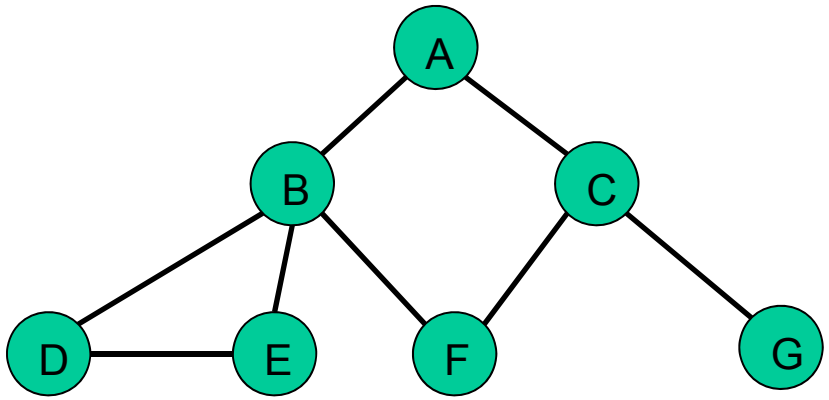
Duyệt theo chiều rộng (BFS)

- Duyệt theo chiều rộng (BFS - Breadth-first Search):
  - Xem tất cả các đường đi có thể với cùng một độ sâu trước khi đi tiếp với độ sâu lớn hơn
- Nếu đồ thị liên thông
  - Chọn một đỉnh xuất phát
  - Tìm mọi đỉnh kề
  - Lần lượt thăm từng đỉnh kề, sau đó quay lại thăm tất cả các đỉnh kề của nó

breadthFirstSearch()

```
breadthFirstSearch(gNode cur, graph g) {
    queue q;
    initialize q;
    mark cur as visited;
    enqueue(&q, cur);
    while (!emptyQueue(&q)) {
        cur= dequeue(&q);
        for all <đỉnh j chưa thăm và kề với cur> {
            mark j as visited;
            enqueue(&q, j);
        }
}
```

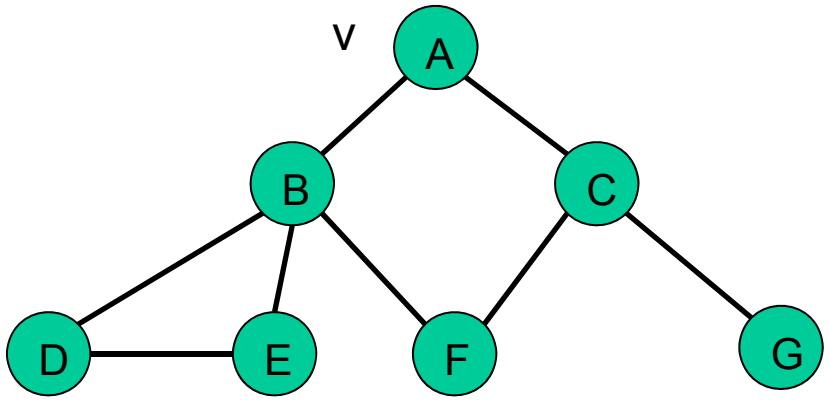
Duyệt theo chiều rộng



Hàng đợi:

cur:

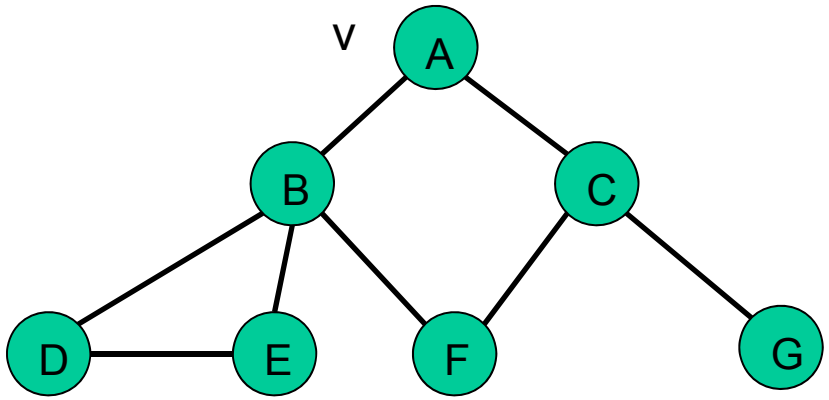
Duyệt theo chiều rộng



Hàng đợi: A

cur:

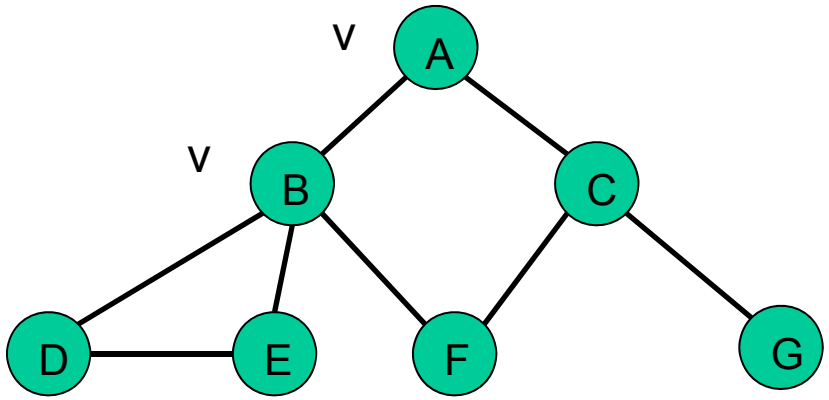
Duyệt theo chiều rộng



Hàng đợi:
cur: A

Kết quả duyệt: A

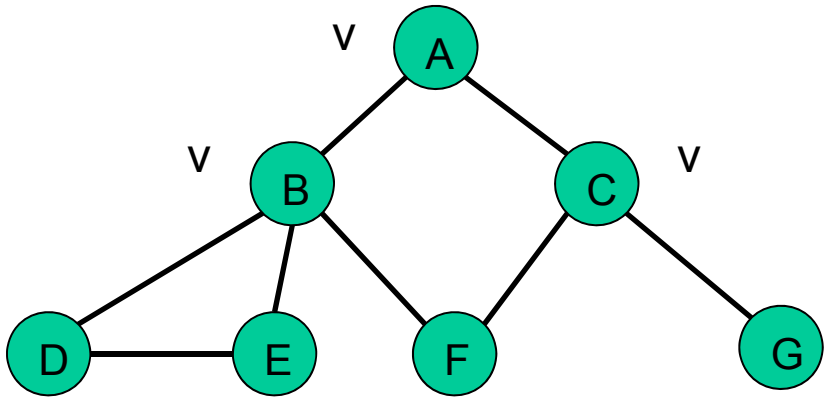
Duyệt theo chiều rộng



Hàng đợi: B
cur: A

Kết quả duyệt: A

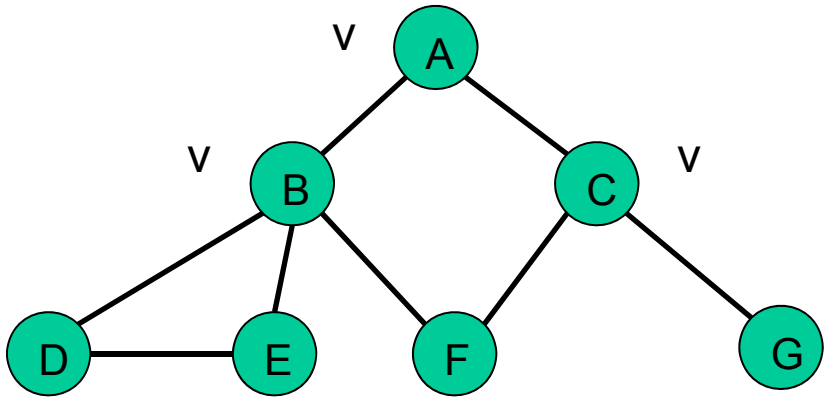
Duyệt theo chiều rộng



Hàng đợi: BC
cur: A

Kết quả duyệt: A

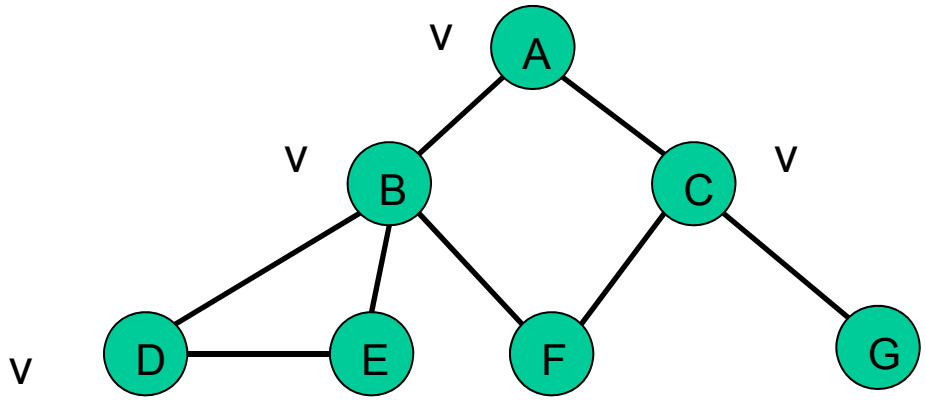
Duyệt theo chiều rộng



Hàng đợi: C
cur: B

Kết quả duyệt: A B

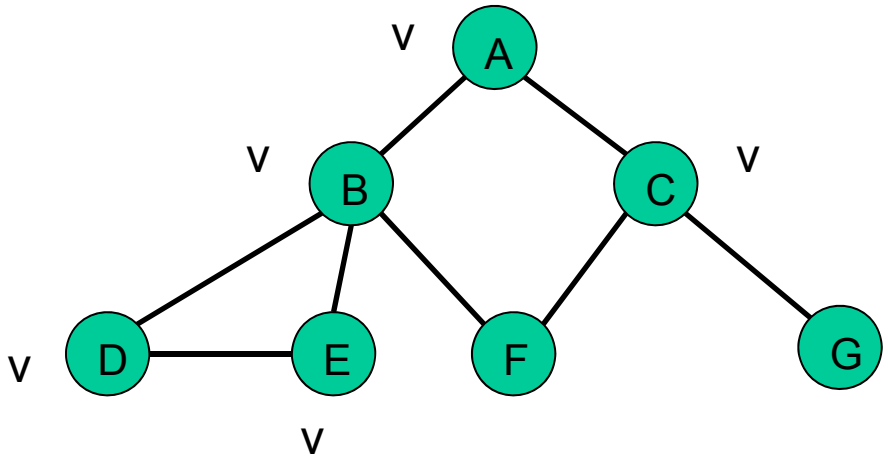
Duyệt theo chiều rộng



Hàng đợi: CD
cur: B

Kết quả duyệt: A B

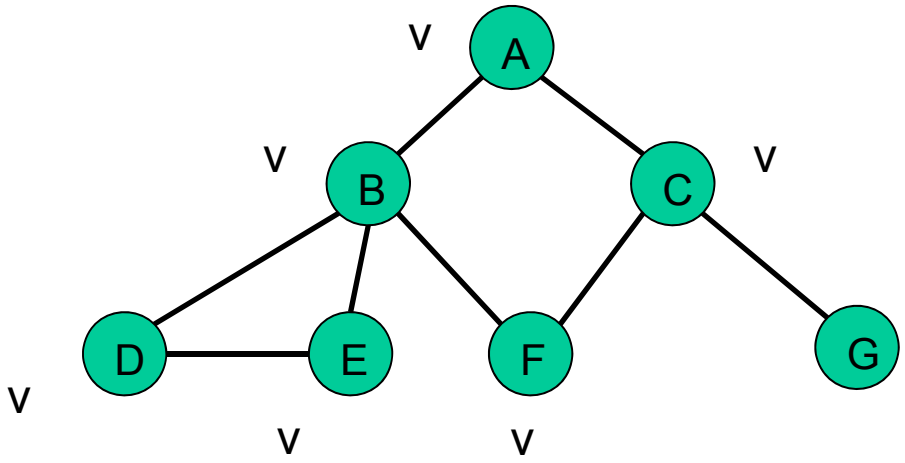
Duyệt theo chiều rộng



Hàng đợi: CDE
cur: B

Kết quả duyệt: A B

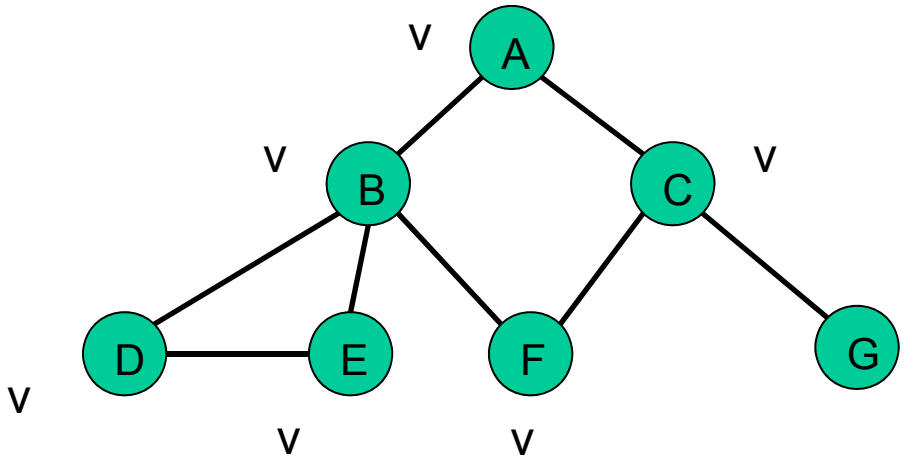
Duyệt theo chiều rộng



Hàng đợi: CDEF
cur: B

Kết quả duyệt: A B

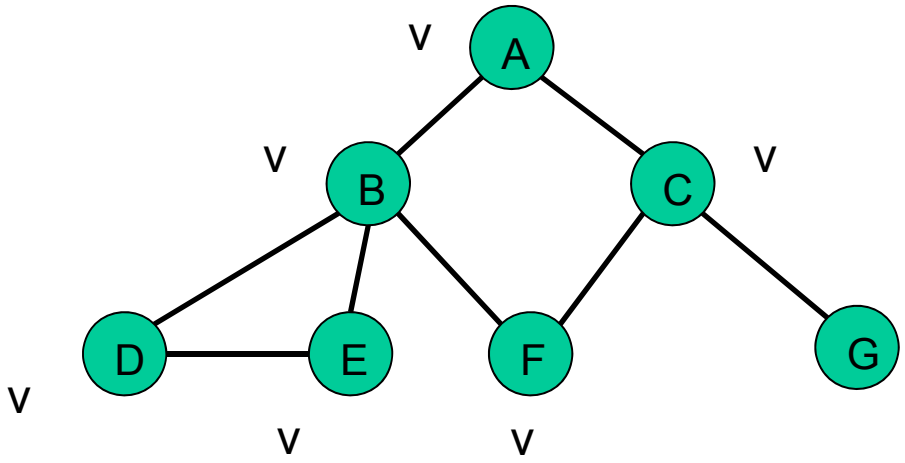
Duyệt theo chiều rộng



Hàng đợi: CDEF
cur: C

Kết quả duyệt: A B

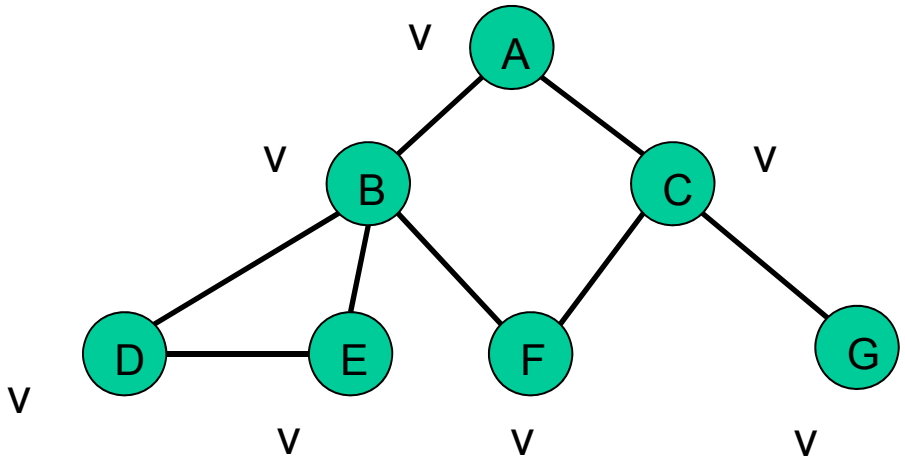
Duyệt theo chiều rộng



Hàng đợi: DEF
cur: C

Kết quả duyệt: A B C

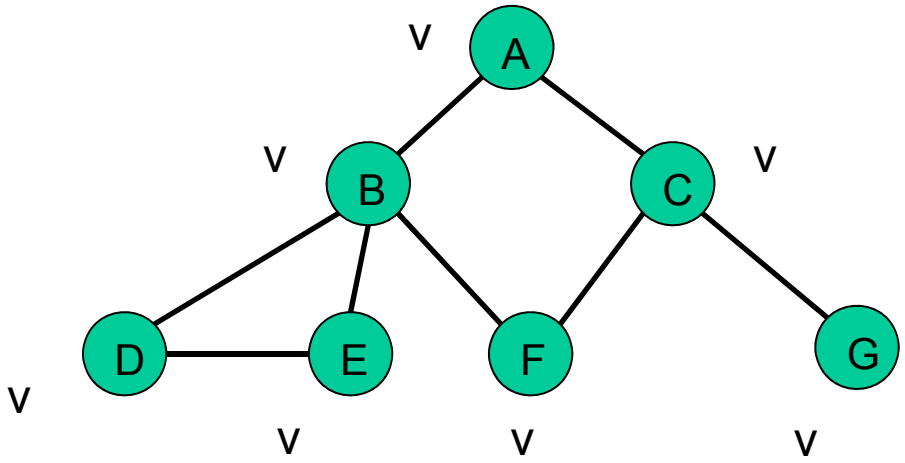
Duyệt theo chiều rộng



Hàng đợi: DEFG
cur: C

Kết quả duyệt: A B C

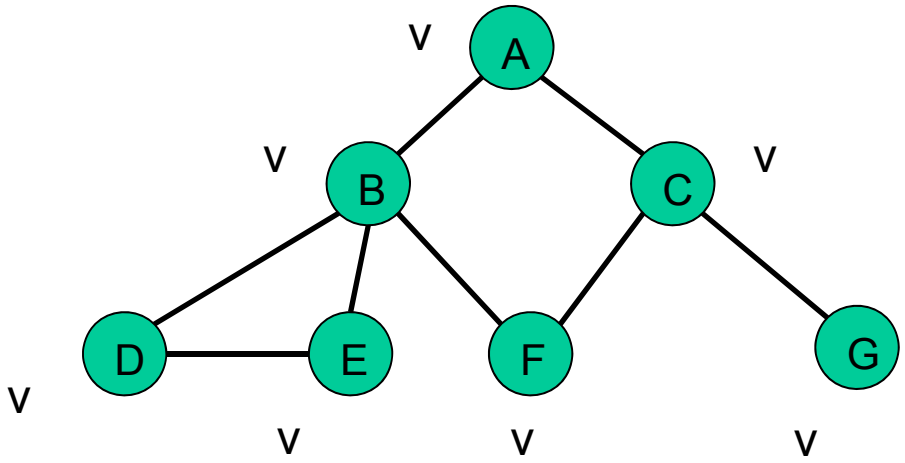
Duyệt theo chiều rộng



Hàng đợi: DEFG
cur: D

Kết quả duyệt: A B C

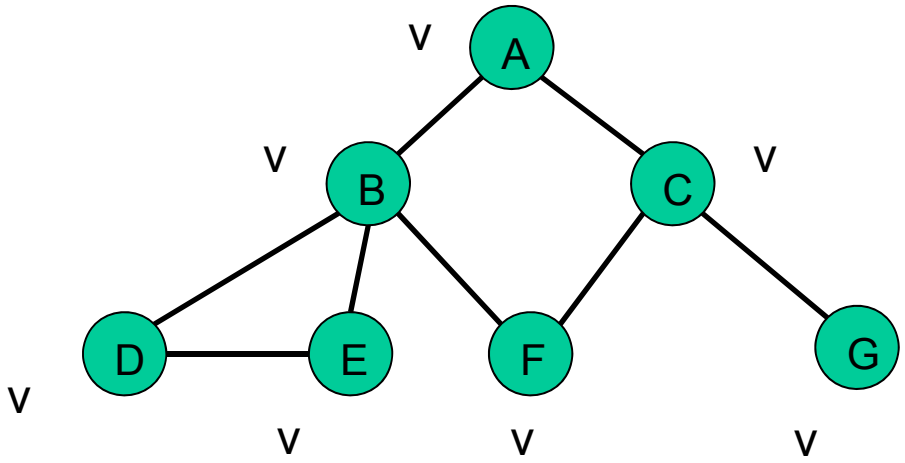
Duyệt theo chiều rộng



Hàng đợi: EFG
cur: D

Kết quả duyệt: A B C D

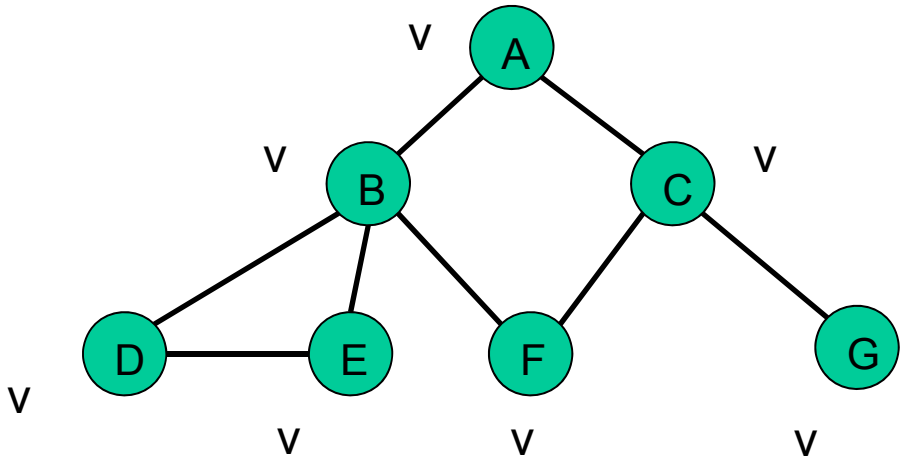
Duyệt theo chiều rộng



Hàng đợi: EFG
cur: E

Kết quả duyệt: A B C D

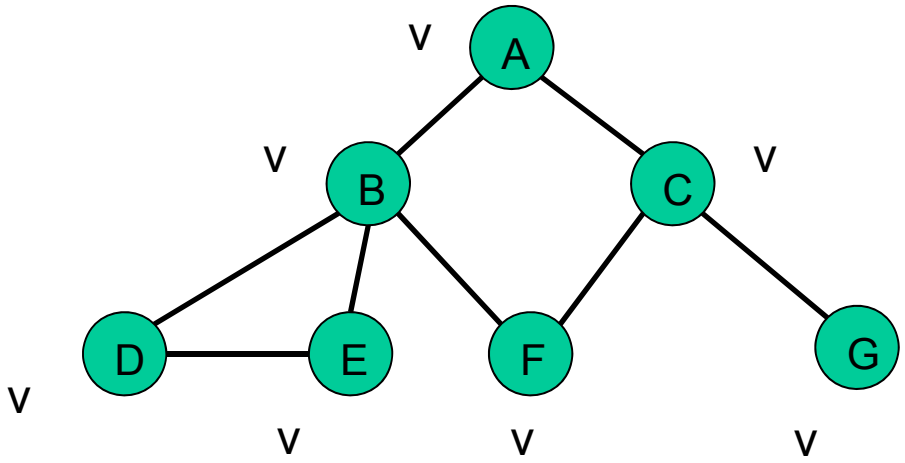
Duyệt theo chiều rộng



Hàng đợi: FG
cur: E

Kết quả duyệt: A B C D E

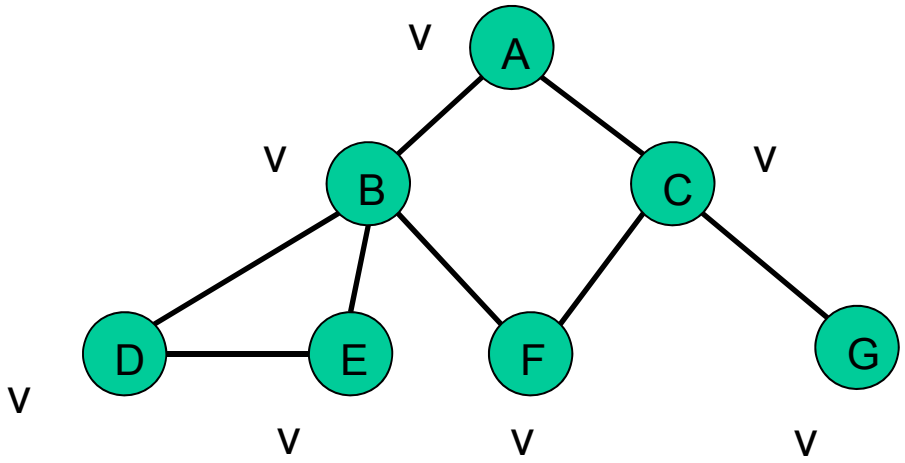
Duyệt theo chiều rộng



Hàng đợi: FG
cur: F

Kết quả duyệt: A B C D E

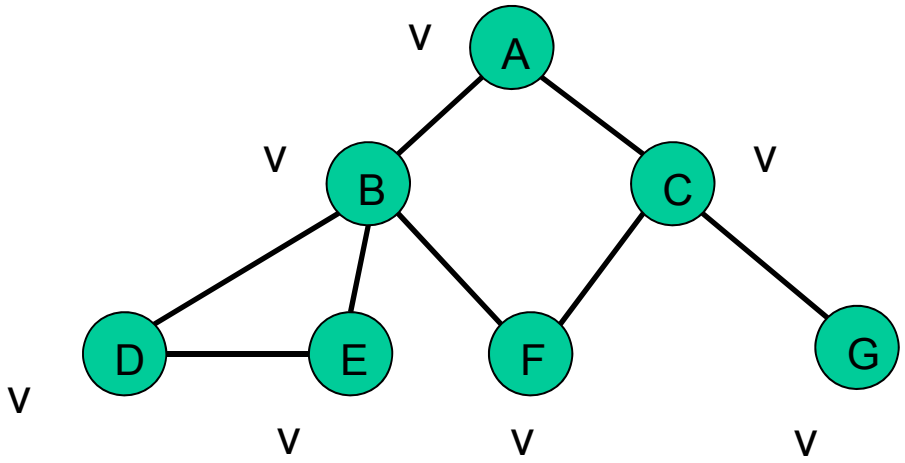
Duyệt theo chiều rộng



Hàng đợi: G
cur: F

Kết quả duyệt: A B C D E F

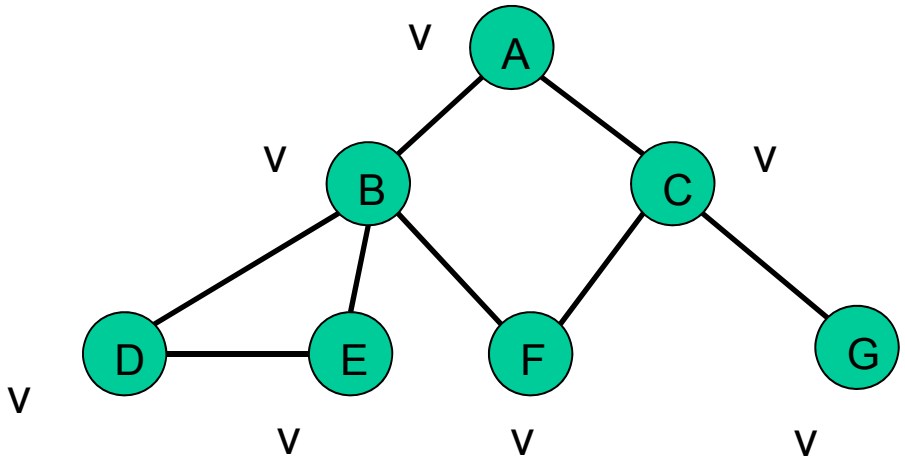
Duyệt theo chiều rộng



Hàng đợi: G
cur: G

Kết quả duyệt: A B C D E F G

Duyệt theo chiều rộng



Hàng đợi:
cur: G

Kết quả duyệt: A B C D E F G

Duyệt theo chiều sâu (DFS)

- Duyệt theo chiều sâu (DFS - Depth-first Search):
  - Thăm nút sâu hơn trên đồ thị khi còn có thể
- Nếu đồ thị liên thông
  - Chọn một đỉnh xuất phát
  - Theo cạnh của đỉnh mới thăm gần nhất v mà vẫn còn cạnh chưa thăm
  - Sau khi thăm hết mọi cạnh của v, quay ngược lại theo cạnh mà từ đó v được thăm

depthFirstSearch()

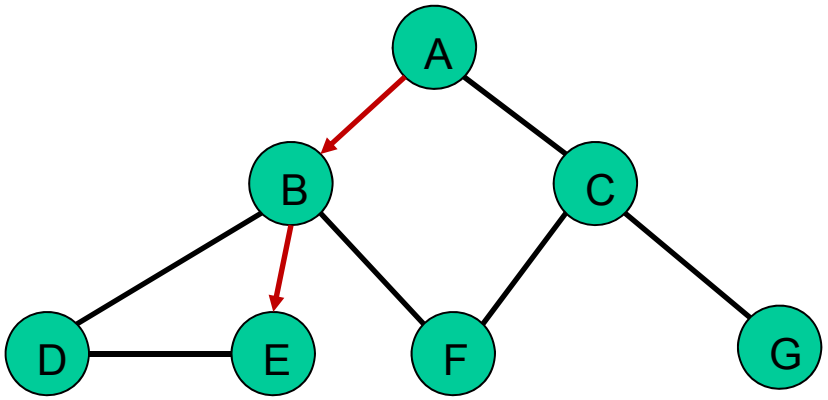
```
depthFirstSearch(v) {
    Đánh dấu v là đã thăm;
    for w chưa thăm và kề với v {
            depthFirstSearch(w}
}
```

Chương 5: Đồ thị

- Khái niệm và biểu diễn đồ thị
- Duyệt đồ thị
- **Tìm đường đi**
- Ứng dụng

Tìm đường

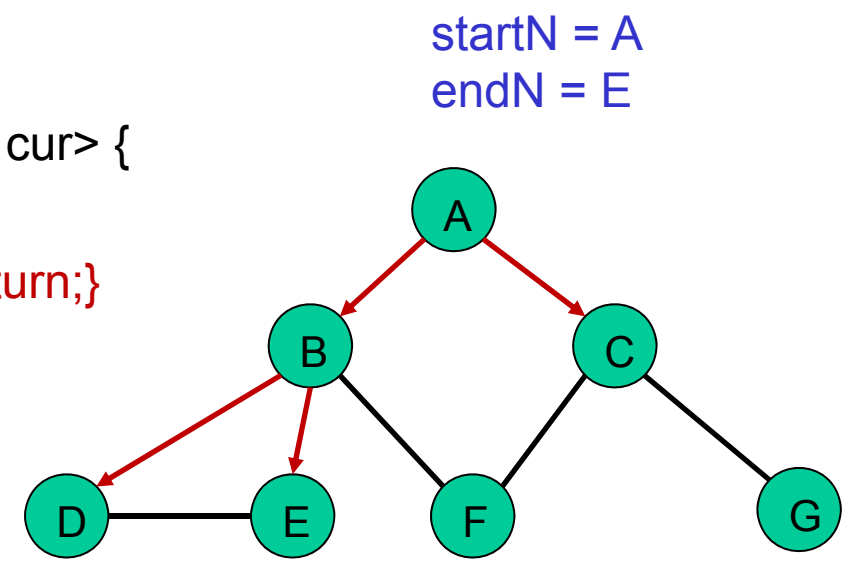
- Bài toán: Tìm một đường giữa hai nút của đồ thị (ví dụ từ A tới E), các nút trên đường đi là không trùng nhau
- Cách giải: Dùng BFS



Sử dụng BFS với đỉnh xuất phát là A, tới khi thăm E

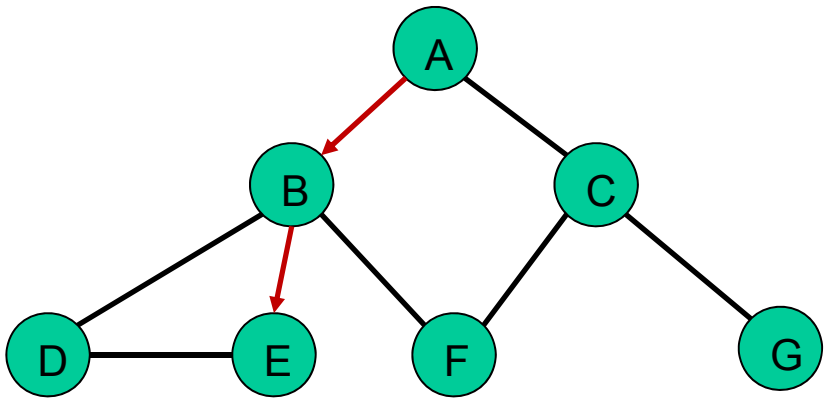
BFS_Path()

```
BFS_Path(gNode *startN, gNode *endN, graph g) {
    queue q;
    gNode *cur = startN;
    initialize q;
    mark cur as visited;
    enqueue(&q, cur);
    while (!emptyQueue(&q)) {
        cur = dequeue(&q);
        for all <đỉnh j chưa thăm và kề với cur> {
            if (j== endN)
                { printf “found the path!”; return;}
            mark j as visited;
            enqueue(&q, j);
        }
}
```



Đường đi ngắn nhất trên đồ thị không có trọng số

- Có nhiều đường đi từ một đỉnh nguồn tới một đỉnh đích
- Đường đi ngắn nhất đồ thị không có trọng số: Đường đi có số cạnh ít nhất
- Đường đi do BFS tìm là đường đi ngắn nhất



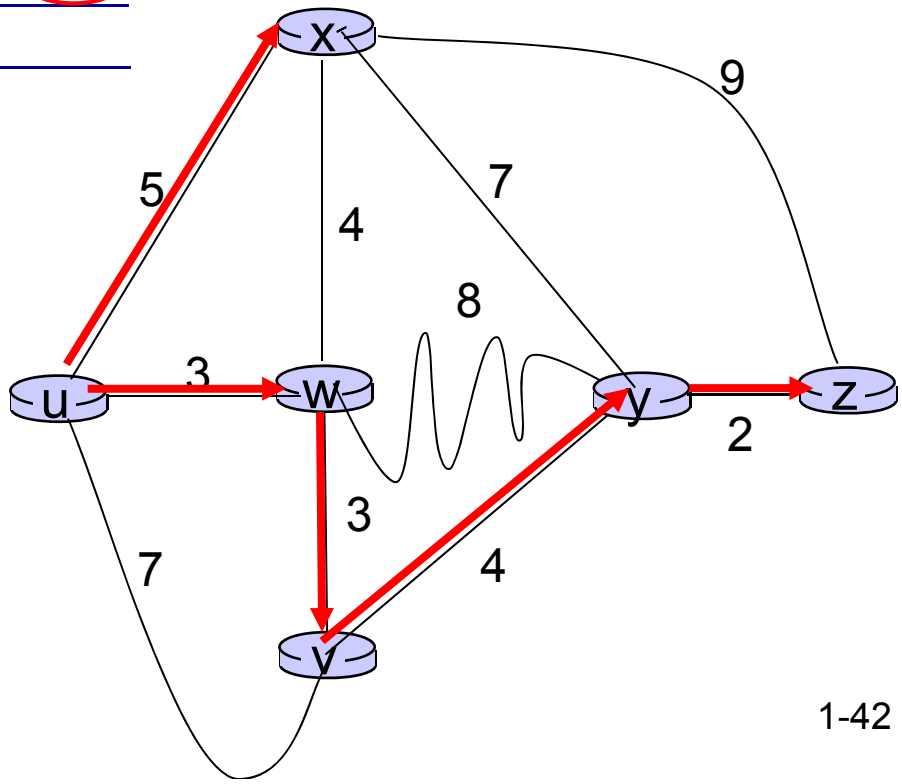
Sử dụng BFS với đỉnh xuất phát là A, tới khi thăm E

Giải thuật Dijsktra

```
1  Khởi tạo:
2    N' = {u}
3    for all nodes v
4      if v kề u
5          then D(v) = c(u,v)
6      else D(v) = ∞
7
8  Loop
9      tìm w không trong N' mà D(w) nhỏ nhất
10     thêm w vào N'
11     cập nhật D(v) cho mọi v kề với w và không trong N' :
12       D(v) = min( D(v), D(w) + c(w,v) )
13     /* chi phí mới tới v sẽ hoặc là chi phí cũ tới v hoặc là chi phí
          đường đi ngắn nhất tới w cộng với chi phí từ w tới v */
15 until mọi nút nằm trong N'
```

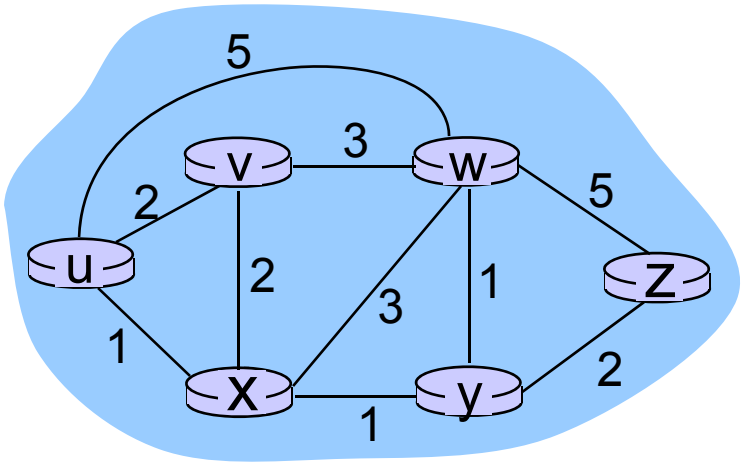
Giải thuật Dijkstra: Ví dụ 1

Bước	N'	D(**v**) p(v)	D(**w**) p(w)	D(**x**) p(x)	D(**y**) p(y)	D(**z**) p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					



Giải thuật Dijkstra: Ví dụ 2

Bước	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



Giải thuật Bellman-Ford

- Tìm đường đi ngắn nhất từ một nút nguồn, cạnh đồ thị có thể có trọng số âm
- Thông báo nếu tồn tại chu trình âm

Giải thuật Bellman-Ford: Khởi tạo

```
Initialize-Single-Source(G, s)
1       for each vertex v in G.V
2               d[v] = INFINITY
3               p[v] = NIL
4       d[s] = 0
```

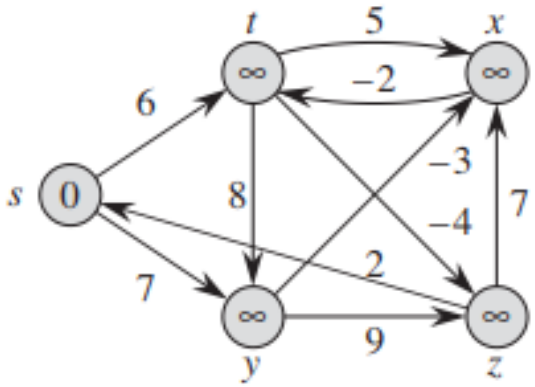
Giải thuật Bellman-Ford: Điều chỉnh

```
RELAX(u, v, c)
1       if d[v] > d[u] + c(u,v)
2               d[v] > d[u] + c(u,v)
3               p[v] = u
```

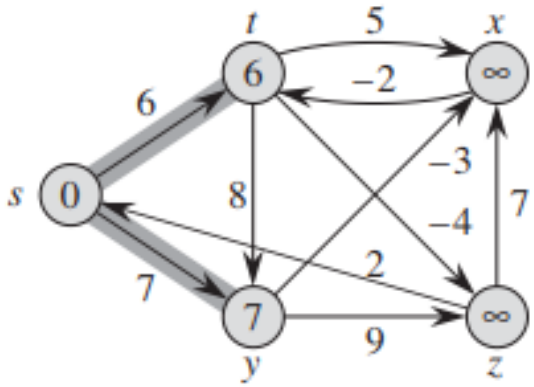
Giải thuật Bellman-Ford

```
BELLMAN-FORD(G, c, s)
1       Initialize-Single-Source(G, s)
2       for i = 1 to |G.V| - 1
3               for each edge (u,v) in G.E
4                       RELAX(u,v,c)
5       for each edge (u,v) in G.E
6               if d[v] > d[u] + c(u,v)
7                       return FALSE
8       return TRUE
```

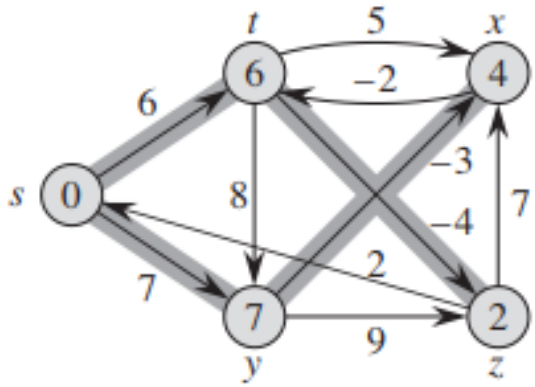
Ví dụ Bellman-Ford



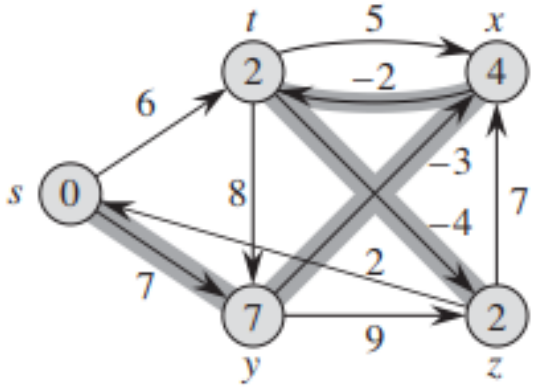
(a)



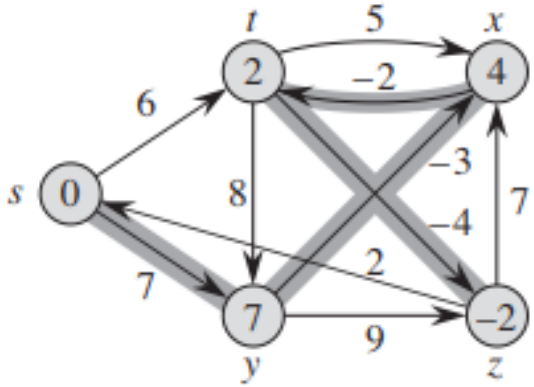
(b)



(c)



(d)



(e)

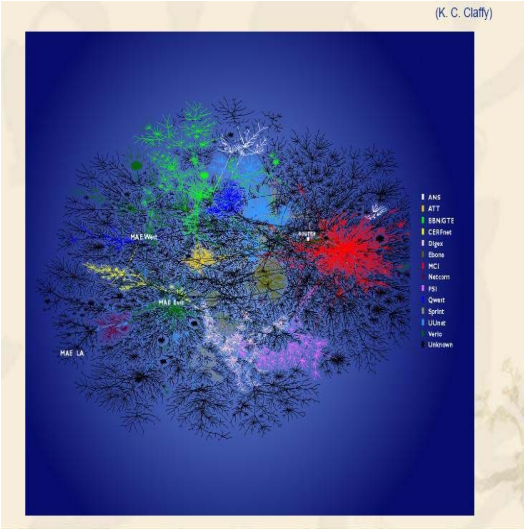
Thứ tự duyệt cạnh (t,x), (t,y), (t,z), (x,t), (y,x), (y,z), (z,x), (z,s), (s,t), (s,y)

Chương 5: Đồ thị

- Khái niệm và biểu diễn đồ thị
- Duyệt đồ thị
- Tìm đường đi
- **Ứng dụng**

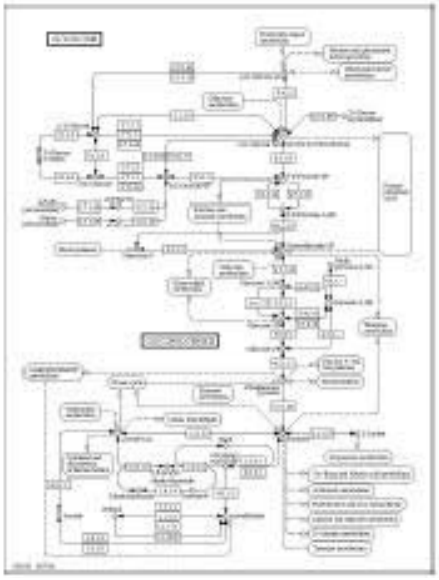
Đồ thị (mạng) xuất hiện trong mọi vấn đề của đời sống

Internet

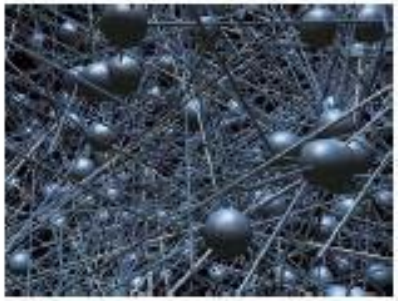


Neural network

US Airline network

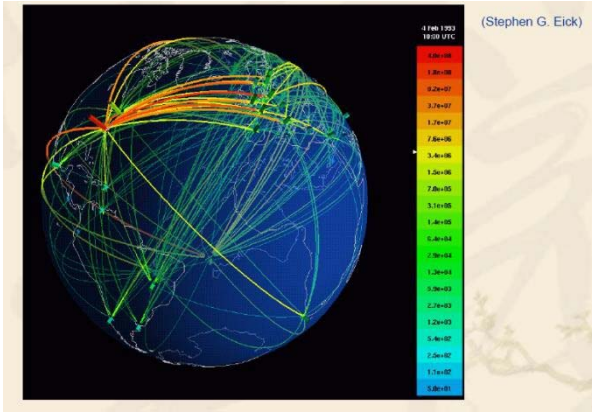


Metabolic network

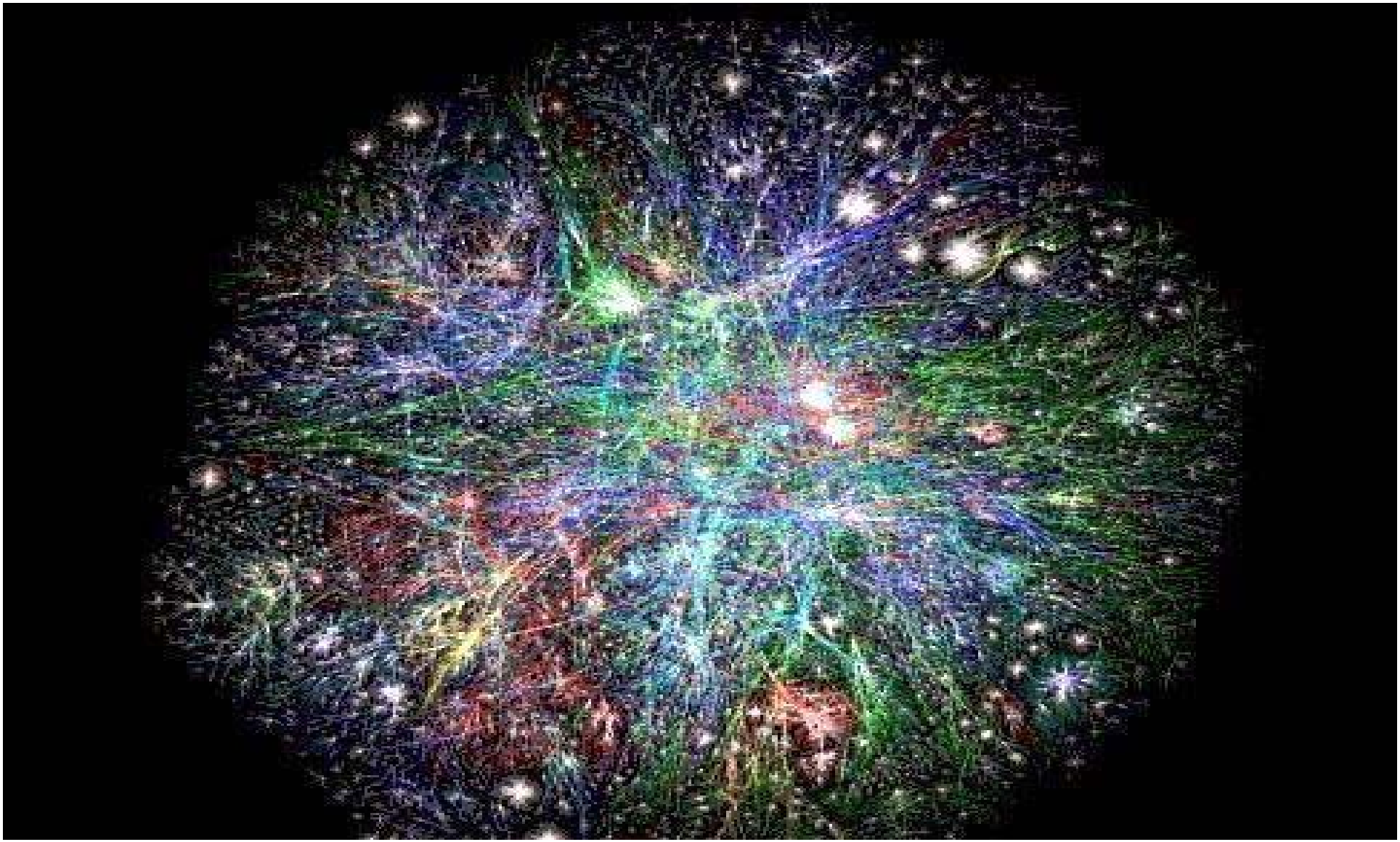


Social network

Telecommunication network



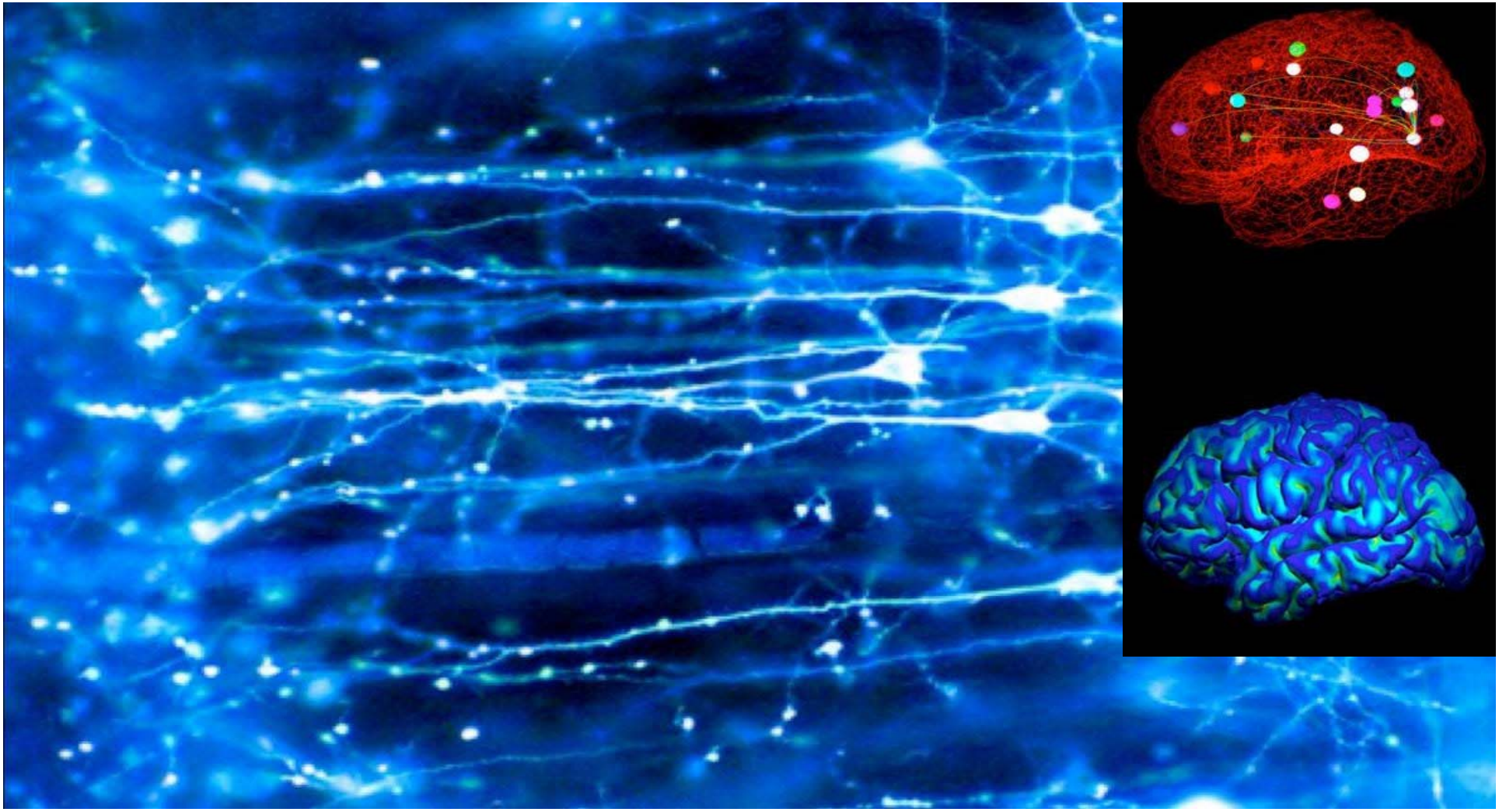
Đồ thị của WWW



Mạng xã hội Facebook

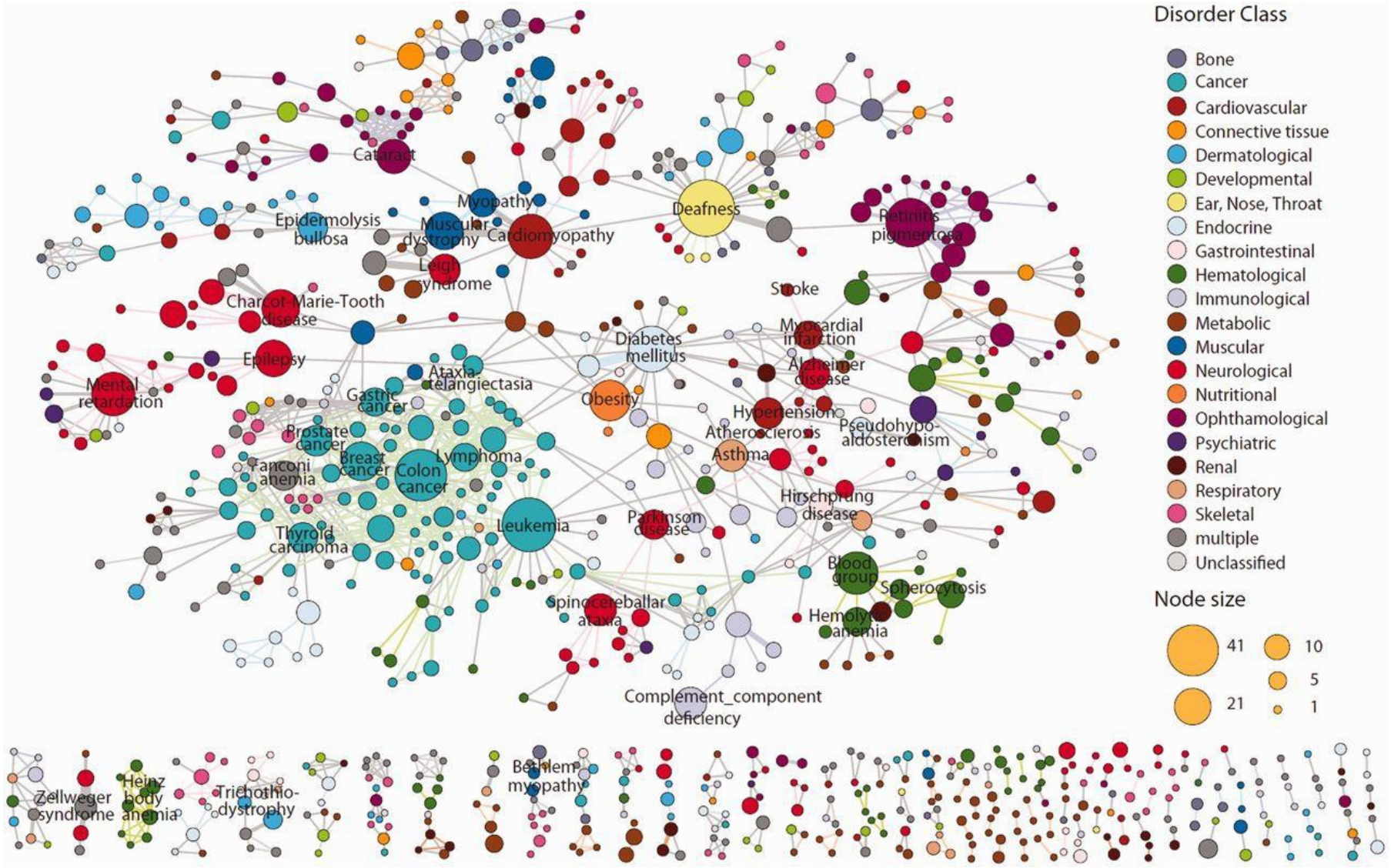
Mạng xã hội Facebook

Mạng nơ-ron bộ não con người: 10-100 tỉ nơ-ron

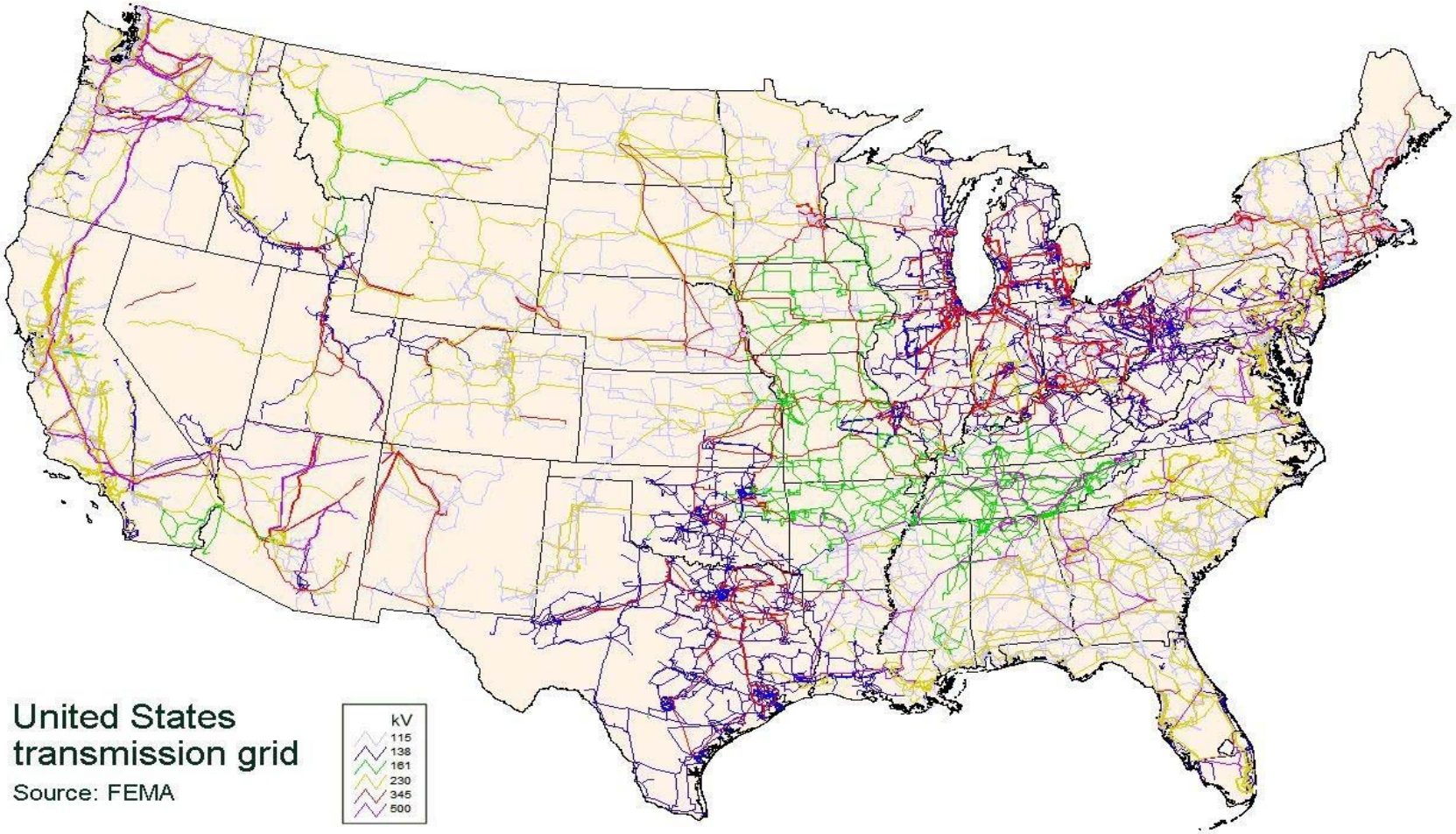


From Barabasi's ppt

Mạng gen và bệnh

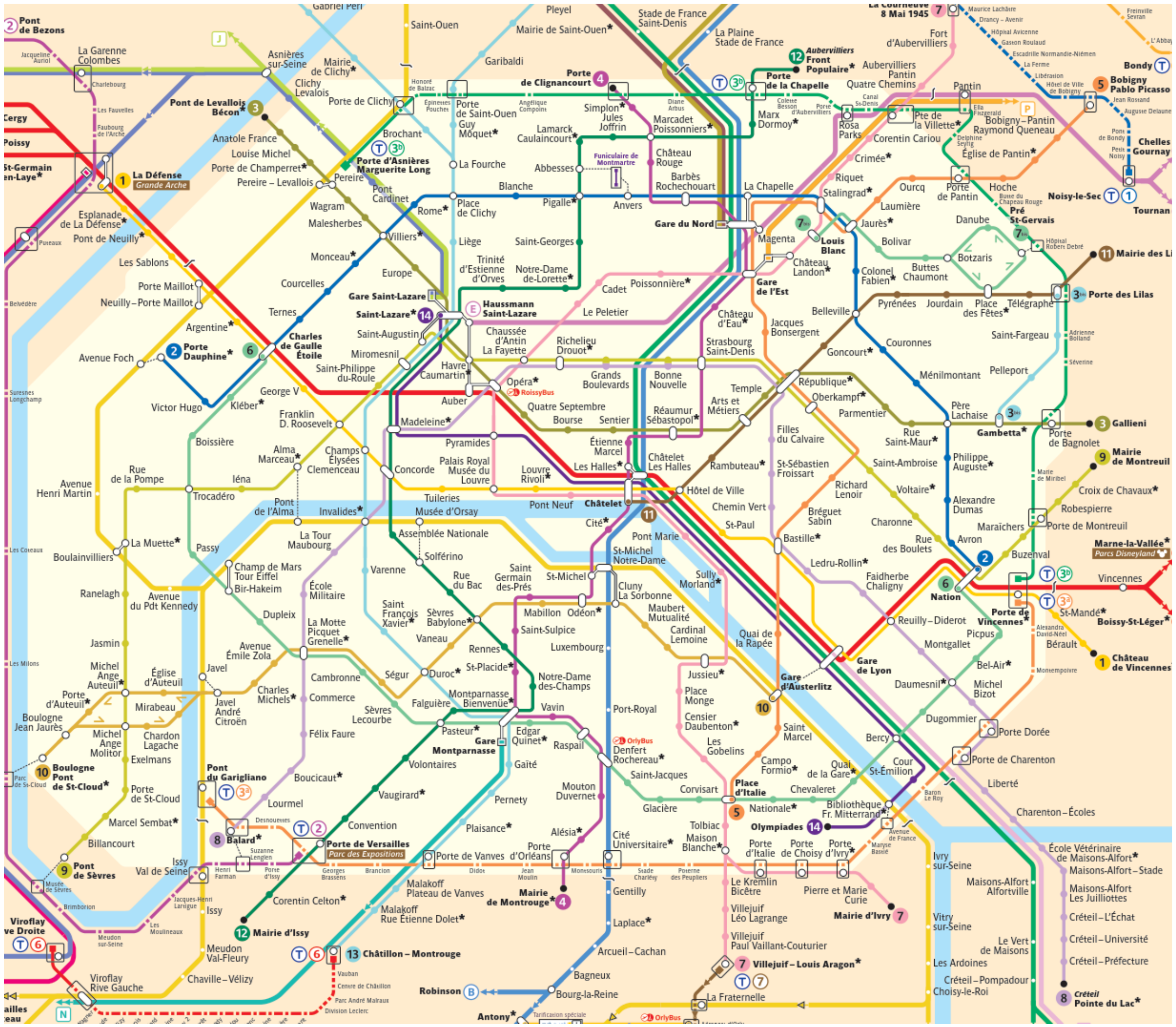


Mạng năng lượng



Lưới năng lượng của nước Mĩ có 300 000 km đường dây với 500 công ty

Mạng giao thông



Cấu trúc dữ liệu và giải thuật