



## ĐỒ THỊ- GRAPH



Bản vẽ hay sơ đồ biểu diễn dữ liệu nhờ sử dụng hệ thống tọa độ !

## CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

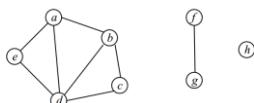
Data Structures & Algorithms

### ĐỒ THỊ- GRAPH



### ĐỒ THỊ- GRAPH

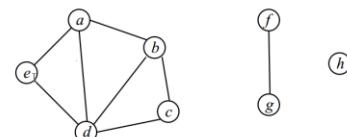
- Trong toán rời rạc: Là cấu trúc rời rạc **có tính trực quan cao**, được sử dụng để **biểu diễn một tập đối tượng có quan hệ với nhau** theo một cách nào đó.



- Định nghĩa hình thức: Đồ thị G được xác định bởi một cặp ( $V, E$ ), trong đó
  - $V$  là tập đỉnh
  - $E$  là tập các cạnh nối cặp đỉnh  $E \subseteq \{(u,v) | u, v \in V\}$

### Đồ thị -GRAPH

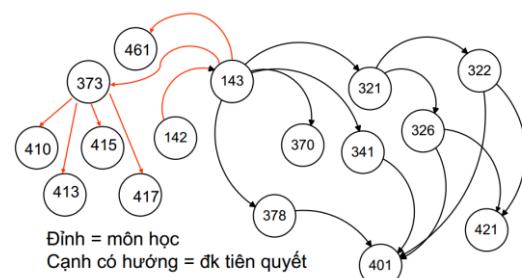
- Ví dụ:** Đơn đồ thị  $G_1 = (V_1, E_1)$ , trong đó  
 $V_1 = \{a, b, c, d, e, f, g, h\}$ ,  
 $E_1 = \{(a,b), (b,c), (c,d), (a,d), (d,e), (a,e), (d,b), (f,g)\}$ .



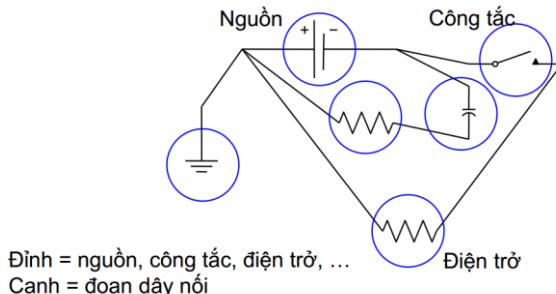
### Ứng dụng thực tế GRAPH

- Có tiềm năng ứng dụng trong nhiều lĩnh vực (Đồ thị có thể dùng để biểu diễn các quan hệ. Nghiên cứu quan hệ giữa các đối tượng là mục tiêu của nhiều lĩnh vực khác nhau).
- Ứng dụng trong mạng máy tính, mạng giao thông, mạng cung cấp nước, mạng điện,...) lập lịch, tối ưu hoá luồng, thiết kế mạch, quy hoạch phát triển...
- Các ứng dụng khác: Phân tích gen, trò chơi máy tính, chương trình dịch, thiết kế hướng đối tượng, ...

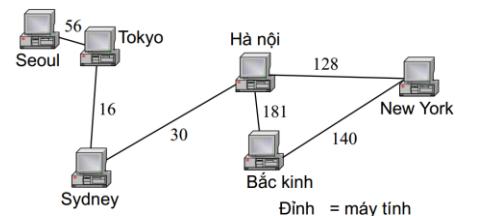
### Mối quan hệ giữa các môn học



### Biểu diễn mạch điện



### Truyền thông trong mạng máy tính

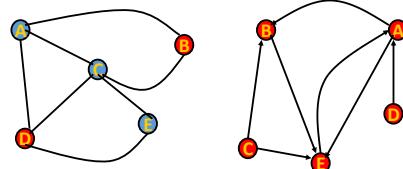


### Mô hình mạng xã hội



### Cung – Cạnh trong đồ thị

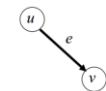
- Tập các cung nối các cặp nút, có thể có nhiều cung trên một cặp nút



### Cung – Cạnh trong đồ thị



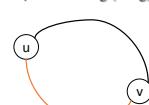
Cạnh vô hướng  $e=(u,v)$



Cạnh có hướng (cung)  $e=(u,v)$



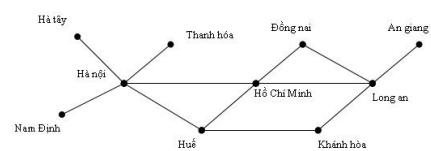
Khuyên



Cạnh song song

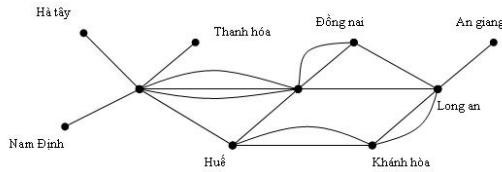
### Đơn đồ thị - Simple Graph

- Đơn đồ thị**: là đồ thị mà **không có khuyên** và **không có cạnh song song**



## Đa đồ thị - Multi Graphs

• **Đa đồ thị:** là đồ thị không thỏa mãn đơn đồ thị !



13

## Đồ thị vô hướng (Undirected graphs)

**Định nghĩa.** Đơn (đa) đồ thị vô hướng  $G = (V, E)$  là cặp gồm:

- Tập đỉnh  $V$  là tập hữu hạn phần tử, các phần tử gọi là các **dinh**
- Tập cạnh  $E$  là tập (hợp) các bộ không có thứ tự dạng  $(u, v), u, v \in V, u \neq v$

➔ Cạnh – cung **không** có thứ tự

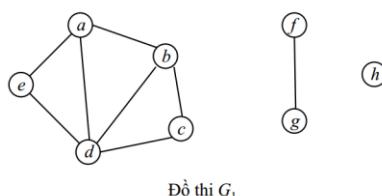
14

## Đồ thị vô hướng (Undirected graphs)

**Ví dụ:** Đơn đồ thị  $G_1 = (V_1, E_1)$ , trong đó

$$V_1 = \{a, b, c, d, e, f, g, h\},$$

$$E_1 = \{(a, b), (b, c), (c, d), (a, d), (d, e), (a, e), (d, b), (f, g)\}.$$

Đồ thị  $G_1$ 

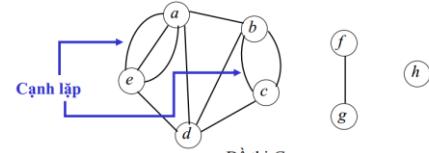
15

## Đồ thị vô hướng (Undirected graphs)

**Ví dụ:** Đa đồ thị  $G_2 = (V_2, E_2)$ , trong đó

$$V_2 = \{a, b, c, d, e, f, g, h\},$$

$$E_2 = \{(a, b), (b, c), (b, c), (c, d), (c, d), (a, d), (d, e), (a, e), (a, e), (d, b), (f, g)\}.$$

Đồ thị  $G_2$ 

16

## Đồ thị có hướng (Directed graphs)

**Định nghĩa.** Đơn (đa) đồ thị có hướng  $G = (V, E)$  là cặp gồm:

- Tập đỉnh  $V$  là tập hữu hạn phần tử, các phần tử gọi là các **dinh**
- Tập cung  $E$  là tập (hợp) các bộ có thứ tự dạng  $(u, v), u, v \in V, u \neq v$

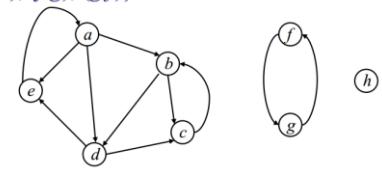
➔ Cạnh – cung **có** thứ tự

## Đồ thị có hướng (Directed graphs)

**Ví dụ:** Đơn đồ thị có hướng  $G_3 = (V_3, E_3)$ , trong đó

$$V_3 = \{a, b, c, d, e, f, g, h\},$$

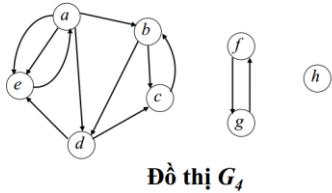
$$E_3 = \{(a, b), (b, c), (c, b), (d, c), (a, d), (b, d), (a, e), (d, e), (e, a), (f, g), (g, f)\}$$

Đồ thị  $G_3$ 

17

### Đồ thị có hướng (Directed graphs)

**Ví dụ:** Đồ thị có hướng  $G_4 = (V_4, E_4)$ , trong đó  
 $V_4 = \{a, b, c, d, e, f, g, h\}$ ,  
 $E_4 = \{(a,b), (b,c), (c,b), (d,c), (a,d), (b, d), (a,e), (a,e), (d,e), (e,a), (f,g), (g,f)\}$



Đồ thị  $G_4$

### Kề (Adjacency) – đồ thị vô hướng

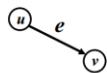


Cho  $G$  là đồ thị vô hướng với tập cạnh  $E$ . Giả sử  $e \in E$  là cạnh  $(u,v)$ . Khi đó ta nói:

- $u, v$  là **kề nhau/lân cận/nối với nhau** (*adjacent / neighbors / connected*).
- Cạnh  $e$  là **liên thuộc** với hai đỉnh  $u$  và  $v$ .
- Cạnh  $e$  **nối** (*connect*)  $u$  và  $v$ .
- Các đỉnh  $u$  và  $v$  là các **đầu mút** (*endpoints*) của cạnh  $e$ .

20

### Kề (Adjacency) – đồ thị có hướng



- Cho  $G$  là đồ thị có hướng (có thể là đơn hoặc đa) và giả sử  $e = (u,v)$  là cạnh của  $G$ . Ta nói:
  - $u$  và  $v$  là kè nhau,  $u$  là kè tới  $v$ ,  $v$  là kè từ  $u$
  - $e$  đi ra khỏi  $u$ ,  $e$  đi vào  $v$ .
  - $e$  nối  $u$  với  $v$ ,  $e$  đi từ  $u$  tới  $v$
  - Đỉnh đầu (*initial vertex*) của  $e$  là  $u$
  - Đỉnh cuối (*terminal vertex*) của  $e$  là  $v$

21

### Đường đi (Path)

- **Định nghĩa.** Đường đi  $P$  độ dài  $n$  từ đỉnh  $u$  đến đỉnh  $v$ , trong đó  $n$  là số nguyên dương, trên đồ thị  $G=(V,E)$  là **dãy**
  - $P: x_0, x_1, \dots, x_{n-1}, x_n$
  - trong đó  $u = x_0, v = x_n, (x_i, x_{i+1}) \in E, i = 0, 1, 2, \dots, n-1$ .
  - Đường đi nói trên còn có thể biểu diễn dưới dạng **dãy** các cạnh:
  - $(x_0, x_1), (x_1, x_2), \dots, (x_{n-1}, x_n)$ .
  - **Đỉnh u gọi là đỉnh đầu, còn đỉnh v gọi là đỉnh cuối** của đường đi.

22

### Chu trình (cycle)

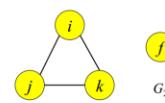
- Đường đi có đỉnh đầu trùng với đỉnh cuối (tức là  $u = v$ ) được gọi là **chu trình**.
- Chu trình được gọi là **sơ cấp** nếu như ngoại trừ đỉnh đầu trùng với đỉnh cuối, không có đỉnh nào bị lặp lại.

23

### Tính liên thông (connectedness)

- Đồ thị vô hướng được gọi là **liên thông** nếu luôn tìm được đường đi nối hai đỉnh bất kỳ của nó.
- **Ví dụ**
  - $G_1$  và  $G_2$  là các đồ thị liên thông
  - Đồ thị  $G$  bao gồm  $G_1$  và  $G_2$  không là đồ thị liên thông
  - **Mệnh đề:** Luôn tìm được đường đi đơn nối hai đỉnh bất kỳ của đồ thị vô hướng liên thông.

24

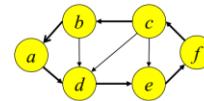


### Tính liên thông (connectedness)

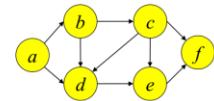
- Đồ thị có hướng được gọi là *liên thông mạnh* (*strongly connected*) nếu như luôn tìm được đường đi nối hai đỉnh bất kỳ của nó.
- Đồ thị có hướng được gọi là *liên thông yếu* (*weakly connected*) nếu như đồ thị vô hướng thu được từ nó bởi việc bỏ qua hướng của tất cả các cạnh của nó là đồ thị vô hướng liên thông.
- Dễ thấy là nếu  $G$  là liên thông mạnh thì nó cũng là liên thông yếu, nhưng điều ngược lại không luôn đúng.

25

### Tính liên thông (connectedness)



Đồ thị liên thông mạnh



Đồ thị liên thông yếu

### Biểu diễn đồ thị trên máy tính

#### (Representation of Graphs)

- Có nhiều cách biểu diễn. Việc lựa chọn cách biểu diễn phụ thuộc vào từng bài toán cụ thể cần xét, thuật toán cụ thể cần cài đặt.
- Có hai vấn đề chính cần quan tâm khi lựa chọn cách biểu diễn:
  - Bộ nhớ mà cách biểu diễn đó đòi hỏi
  - Thời gian cần thiết để trả lời các truy vấn thường xuyên đổi với đồ thị trong quá trình xử lý đồ thị:
    - Chẳng hạn:**
      - Có cạnh nối hai đỉnh  $u, v$ ?
      - Liệt kê các đỉnh kề của đỉnh  $v$ ?

27

### Biểu diễn bằng ma trận kề(Adjacency Matrix)

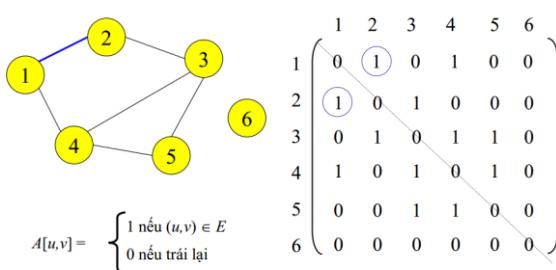
- $|V| \times |V|$  ma trận  $A$ .
- Các đỉnh được đánh số từ 1 đến  $|V|$  theo 1 thứ tự nào đó.
- $A$  xác định bởi:

$$A[i, j] = a_{ij} = \begin{cases} 1 & \text{nếu } (i, j) \in E \\ 0 & \text{nếu trái lại} \end{cases}$$

- $n = |V|$ ;  $m = |E|$

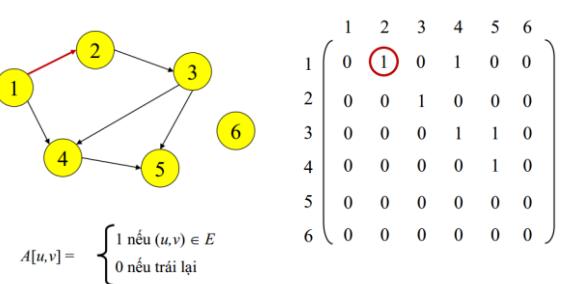
28

### Biểu diễn bằng ma trận kề(Adjacency Matrix)



29

### Biểu diễn bằng ma trận kề(Adjacency Matrix)



30

### Biểu diễn ma trận liên thuộc đỉnh cạnh

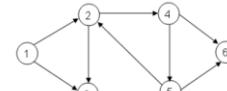
- Xét  $G = (V, E)$ , ( $V = \{1, 2, \dots, n\}$ ,  $E = \{e_1, e_2, \dots, e_m\}$ ), là đơn đồ thị có hóng.
- Ma trận liên thuộc đỉnh cạnh  $A = (a_{ij}; i = 1, 2, \dots, n; j = 1, 2, \dots, m)$ , với

$$a_{ij} = \begin{cases} 1, & \text{nếu đỉnh } i \text{ là đỉnh đầu của cung } e_j \\ -1, & \text{nếu đỉnh } i \text{ là đỉnh cuối của cung } e_j \\ 0, & \text{nếu đỉnh } i \text{ không là đầu/mút của cung } e_j \end{cases}$$

- Ma trận liên thuộc đỉnh-cạnh là một trong những cách biểu diễn rất hay được sử dụng trong các bài toán liên quan đến đồ thị có hóng mà trong đó phải xử lý các cung của đồ thị.

31

### Biểu diễn ma trận liên thuộc đỉnh cạnh



(1,2) (1,3) (2,3) (2,4) (3,5) (4,5) (4,6) (5,2) (5,6)

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & -1 & 0 & 1 & 1 & 0 & 0 & 0 & -1 & 0 \\ 3 & 0 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & -1 & 0 & 1 & 1 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 1 \\ 6 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 \end{bmatrix}$$

32

### Biểu diễn ma trận trọng số

- Trong trường hợp đồ thị có trọng số trên cạnh, thay vì ma trận kề, để biểu diễn đồ thị ta sử dụng **ma trận trọng số**

$$C = c[i, j], \quad i, j = 1, 2, \dots, n,$$

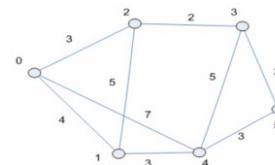
với

$$c[i, j] = \begin{cases} c(i, j), & \text{nếu } (i, j) \in E \\ \theta, & \text{nếu } (i, j) \notin E, \end{cases}$$

trong đó  $\theta$  là giá trị đặc biệt để chỉ ra một cặp  $(i, j)$  không là cạnh, tùy từng trường hợp cụ thể, có thể đọc đặt bằng một trong các giá trị sau:  $0, +\infty, -\infty$ .

33

### Biểu diễn ma trận trọng số



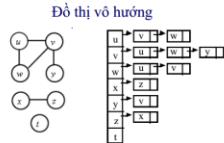
	0	1	2	3	4	5
0	0	4	3	0	7	0
1	4	0	5	0	3	0
2	3	5	0	2	0	0
3	0	0	2	0	5	2
4	7	3	0	5	0	3
5	0	0	0	2	3	0

34

### Biểu diễn danh sách kề (Adjacency List)

- Danh sách kề (Adjacency Lists):** Với mỗi đỉnh  $v$  cất giữ danh sách các đỉnh kề của nó.
  - Là mảng  $Ke$  gồm  $|V|$  danh sách.
  - Mỗi đỉnh có một danh sách.
  - Với mỗi  $u \in V$ ,  $Ke[u]$  bao gồm tất cả các đỉnh kề của  $u$ .

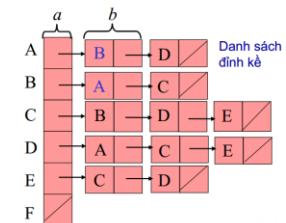
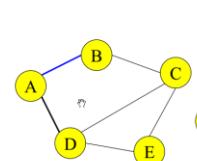
#### Ví dụ:



35

### Biểu diễn danh sách kề (Adjacency List)

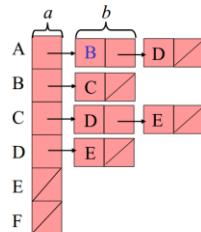
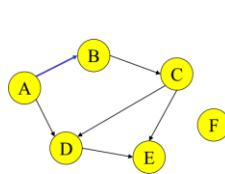
Với mỗi  $v \in V$ ,  $Ke(v) =$  danh sách các đỉnh  $u: (v, u) \in E$



36

## Biểu diễn danh sách kề (Adjacency List)

Với mỗi  $v \in V$ ,  $\text{Ke}(v) = \{ u : (v, u) \in E \}$



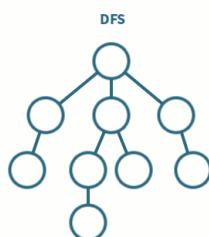
37

## Các thuật toán duyệt đồ thị

- Duyệt đồ thị: Graph Searching hoặc Graph Traversal
  - Duyệt qua mỗi đỉnh và mỗi cạnh của đồ thị
- Ứng dụng:
  - Cần để khảo sát các tính chất của đồ thị
  - Là thành phần cơ bản của nhiều thuật toán trên đồ thị
- Hai thuật toán duyệt cơ bản:
  - Tìm kiếm theo chiều rộng (Breadth First Search – BFS)
  - Tìm kiếm theo chiều sâu (Depth First Search – DFS)

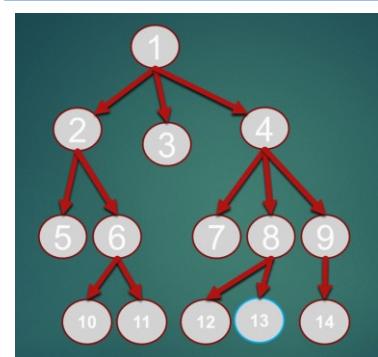
38

## DFS VS BFS



39

## Tìm kiếm theo chiều rộng (BFS)



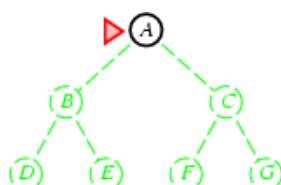
40

## Tìm kiếm theo chiều rộng (BFS)

- Tìm kiếm theo từng tầng. Expand đỉnh gần nút nhất.

### Cài đặt:

- $L$  (danh sách các node đã được sinh ra và chờ được duyệt) **được cài đặt dưới dạng danh sách FIFO**, i.e., Các node con được sinh ra (bởi EXPAND) sẽ được đặt ở dưới cùng của  $L$ .



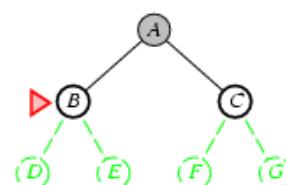
41

## Tìm kiếm theo chiều rộng (BFS)

- Tìm kiếm theo từng tầng. **Expand node gần nút nhất.**

### Cài đặt:

- $L$  (danh sách các node đã được sinh ra và chờ được duyệt) **được cài đặt dưới dạng danh sách FIFO**, i.e., Các node con được sinh ra (bởi EXPAND) sẽ được đặt ở dưới cùng của  $L$ .



Chương 3: Tìm kiếm mứ

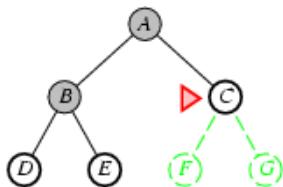
42

### Tìm kiếm theo chiều rộng (BFS)

- Tìm kiếm theo từng tầng. Expand node gần nút nhất.

- Cài đặt:

- $L$  (danh sách các node đã được sinh ra và chờ được duyệt) được cài đặt dưới dạng danh sách FIFO, i.e., Các node con được sinh ra (bởi EXPAND) sẽ được đặt ở dưới cùng của  $L$ .



Chương 3: Tìm kiếm mờ

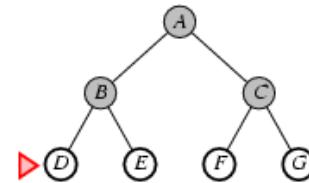
43

### Tìm kiếm theo chiều rộng (BFS)

- Tìm kiếm theo từng tầng. Expand node gần nút nhất.

- Cài đặt:

- $L$  (danh sách các node đã được sinh ra và chờ được duyệt) được cài đặt dưới dạng danh sách FIFO, i.e., Các node con được sinh ra (bởi EXPAND) sẽ được đặt ở dưới cùng của  $L$ .



44

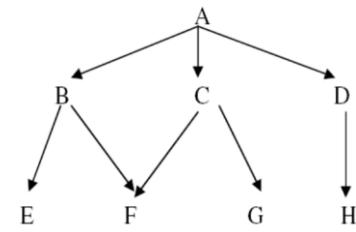
### Tìm kiếm theo chiều rộng (BFS)

Thuật toán tìm kiếm theo bề rộng được mô tả bởi thủ tục sau:

```
procedure Breadth_First_Search;
begin
1. Khởi tạo danh sách  $L$  chỉ chứa trạng thái ban đầu;
2. loop do
  2.1 if  $L$  rỗng then {thông báo tìm kiếm thất bại; stop};
  2.2 Loại trạng thái  $u$  ở đầu danh sách  $L$ ;
  2.3 if  $u$  là trạng thái kết thúc then {thông báo tìm kiếm thành công; stop};
  2.4 for mỗi trạng thái  $v$  kề  $u$  do {
    Đặt  $v$  vào cuối danh sách  $L$ ;
    father( $v$ )  $\leftarrow u$ }
end;
```

Trong thuật toán, sử dụng hàm father( $v$ ) để lưu lại cha của mỗi đỉnh trên đường đi, father( $v$ )=u nếu  $u$  là cha của đỉnh  $v$ .

45



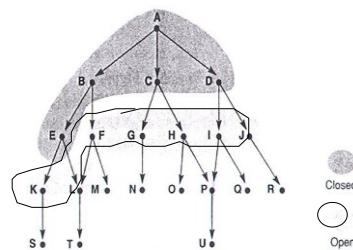
Các step duyệt theo chiều rộng theo hai tập:

Open[] các đỉnh đang xét  
Close[] Đỉnh đã duyệt qua

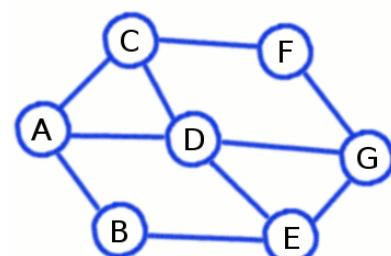
46

### Tìm kiếm theo chiều rộng (BFS)

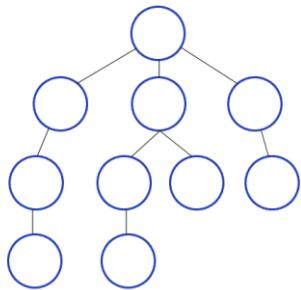
- Khởi tạo Open = [A];  
Closed = []
- Open = [B,C,D]  
Closed = [A]
- Open = [C,D,E,F]  
Closed = [B,A]
- Open = [D,E,F,G,H];  
Closed = [C,B,A]
- Open = [E,F,G,H,I,J];  
Closed = [D,C,B,A]
- Open = [F,G,H,I,J,K,L];  
Closed = [E,D,C,B,A]
- Open = [G,H,I,J,K,L,M];  
Closed = [F,E,D,C,B,A]
- ...



### Tìm kiếm theo chiều rộng (BFS)



### Tìm kiếm theo chiều sâu (DFS)



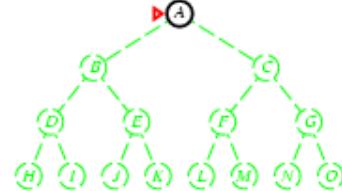
49

### Tìm kiếm theo chiều sâu (DFS)

- EXPAND node chưa xét ở sâu nhất.

**Cài đặt:**

- $L$  = danh sách kiểu LIFO, i.e., Đẩy các nodes con sinh bởi EXPAND vào đầu  $L$



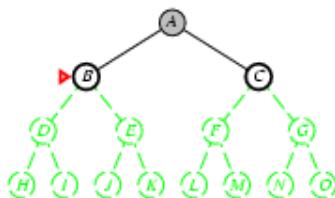
50

### Tìm kiếm theo chiều sâu (DFS)

- EXPAND node chưa xét ở sâu nhất.

**Cài đặt:**

- $L$  = danh sách kiểu LIFO, i.e., Đẩy các nodes con sinh bởi EXPAND vào đầu  $L$



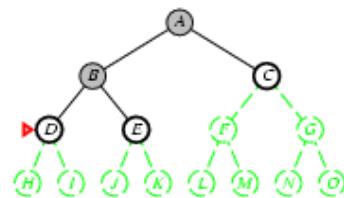
51

### Tìm kiếm theo chiều sâu (DFS)

- EXPAND node chưa xét ở sâu nhất.

**Cài đặt:**

- $L$  = danh sách kiểu LIFO, i.e., Đẩy các nodes con sinh bởi EXPAND vào đầu  $L$



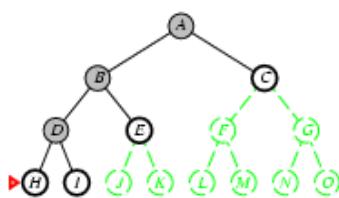
52

### Tìm kiếm theo chiều sâu (DFS)

- EXPAND node chưa xét ở sâu nhất.

**Cài đặt:**

- $L$  = danh sách kiểu LIFO, i.e., Đẩy các nodes con sinh bởi EXPAND vào đầu  $L$



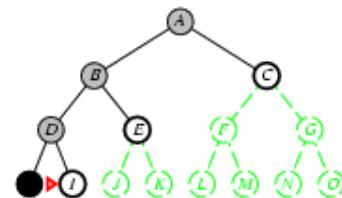
53

### Tìm kiếm theo chiều sâu (DFS)

- EXPAND node chưa xét ở sâu nhất.

**Cài đặt:**

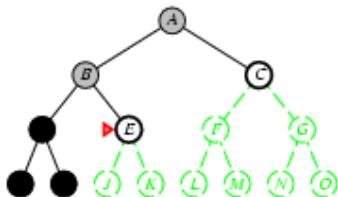
- $L$  = danh sách kiểu LIFO, i.e., Đẩy các nodes con sinh bởi EXPAND vào đầu  $L$



54

### Tìm kiếm theo chiều sâu (DFS)

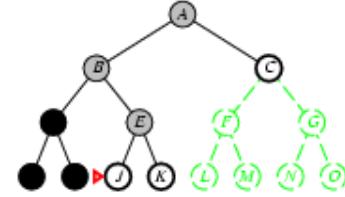
- EXPAND node chưa xét ở sâu nhất.
- **Cài đặt:**
  - $L$  = danh sách kiểu LIFO, i.e., Đẩy các nodes con sinh bởi EXPAND vào đầu  $L$



55

### Tìm kiếm theo chiều sâu (DFS)

- EXPAND node chưa xét ở sâu nhất.
- **Cài đặt:**
  - $L$  = danh sách kiểu LIFO, i.e., Đẩy các nodes con sinh bởi EXPAND vào đầu  $L$

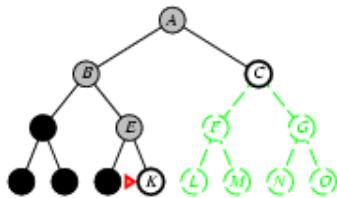


Chương 3: Tìm kiếm mứ

56

### Tìm kiếm theo chiều sâu (DFS)

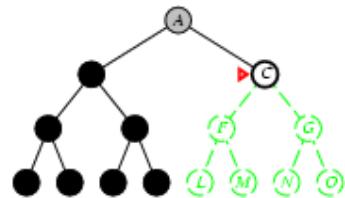
- EXPAND node chưa xét ở sâu nhất.
- **Cài đặt:**
  - $L$  = danh sách kiểu LIFO, i.e., Đẩy các nodes con sinh bởi EXPAND vào đầu  $L$



57

### Tìm kiếm theo chiều sâu (DFS)

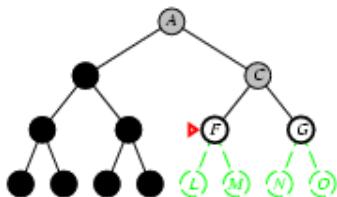
- EXPAND node chưa xét ở sâu nhất.
- **Cài đặt:**
  - $L$  = danh sách kiểu LIFO, i.e., Đẩy các nodes con sinh bởi EXPAND vào đầu  $L$



58

### Tìm kiếm theo chiều sâu (DFS)

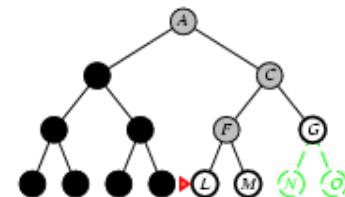
- EXPAND node chưa xét ở sâu nhất.
- **Cài đặt:**
  - $L$  = danh sách kiểu LIFO, i.e., Đẩy các nodes con sinh bởi EXPAND vào đầu  $L$



59

### Tìm kiếm theo chiều sâu (DFS)

- EXPAND node chưa xét ở sâu nhất.
- **Cài đặt:**
  - $L$  = danh sách kiểu LIFO, i.e., Đẩy các nodes con sinh bởi EXPAND vào đầu  $L$



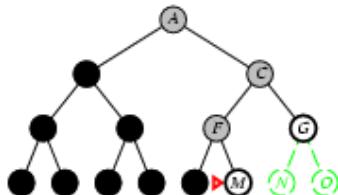
60

### Tìm kiếm theo chiều sâu (DFS)

- EXPAND node chưa xét ở sâu nhất.

**Cài đặt:**

- $L$  = danh sách kiểu LIFO, i.e., Đẩy các nodes con sinh bởi EXPAND vào đầu  $L$



61

### Tìm kiếm theo chiều sâu (DFS)

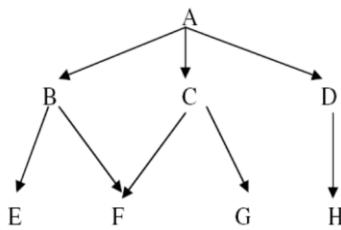
```

Procedure Depth_First_Search;
begin
1. Khởi tạo danh sách  $L$  chỉ chứa trạng thái ban đầu  $u_0$ ;
2. loop do
2.1. if  $L$  rỗng then
    {thông báo thất bại; stop};
2.2. Loại trạng thái  $u$  ở đầu danh sách  $L$ ;
2.3. if  $u$  là trạng thái kết thúc then
    {thông báo thành công; stop};
2.4. for mỗi trạng thái  $v$  kề  $u$  do
    {Đặt  $v$  vào đầu danh sách  $L$ ;};

end;

```

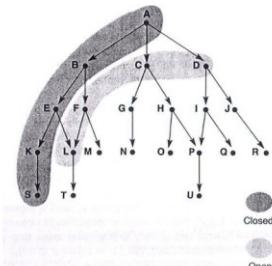
62



Open[] các tính đang xét  
Close[] Đỉnh đã duyệt qua

63

1. Open = [A]; closed = []
2. Open = [B,C,D]; closed = [A]
3. Open = [E,F,C,D]; closed = [B,A]
4. Open = [K,L,F,C,D];
 closed = [E,B,A]
5. Open = [S,L,F,C,D];
 closed = [K,E,B,A]
6. Open = [L,F,C,D];
 closed = [S,K,E,B,A]
7. Open = [T,F,C,D];
 closed = [L,S,K,E,B,A]
8. Open = [F,C,D];
 closed = [T,L,S,K,E,B,A]
9. ...



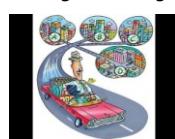
64

### BÀI TOÁN TÌM ĐƯỜNG ĐI NGẮN NHẤT

Nguyên lý tham lam (Greedy Best First Search)

Travelling Salesman Problem – TSP

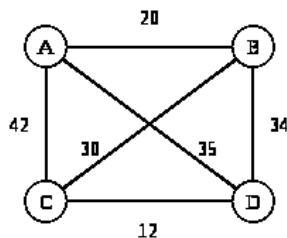
- Cho trước một **danh sách các điểm giao hàng** và **khoảng cách giữa chúng**.
- Nhân viên giao hàng **xuất phát** từ **một điểm** cho trước
- Tìm đường đi ngắn nhất** sao cho **tất cả** các điểm phải **được giao hàng** và **mỗi điểm** chỉ **viếng thăm** đúng một lần sau đó **trở về** **điểm xuất phát**.



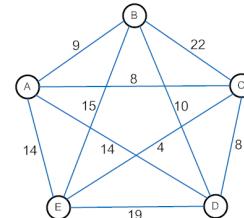
### Ý tưởng

- Từ điểm xuất phát, **liệt kê tất cả đường đi** từ **điểm xuất phát** cho **đến n điểm** → **chọn đi theo con đường ngắn nhất**
- Khi đã đi đến một điểm, **chọn đi đến điểm kế tiếp cũng theo nguyên tắc trên**.
- Lặp lại quá trình cho **đến lúc không còn điểm nào để đi và trở về điểm ban đầu**

Xuất phát từ điểm A → đường đi?



**Bài tập:** Tìm vị trí xuất phát mà đi hết trong đoạn đường tốt nhất theo nguyên lý BFS ?



Định bắt đầu	Đường đi	Tổng chiều dài
A	ACEBDA	51
B	BACEDB	50
C	CEABDC	45
D	DCEABD	45
E	ECABDE	50
E	ECDBAE	45

## BÀI TOÁN TÔ MÀU ĐỒ THỊ

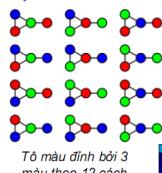
👉 Tô màu đồ thị (graph coloring) là một phép gán “màu sắc” đến các phần tử của đồ thị thỏa mãn ràng buộc cho trước.

- 👉 Một số bài toán tô màu phổ biến:
  - ❖ Tô màu sao cho 2 đỉnh kề nhau không cùng màu gọi là tô màu đỉnh (vertex coloring).
  - ❖ Tô màu sao cho 2 cạnh kề nhau không cùng màu gọi là tô màu cạnh (edge coloring).
  - ❖ Tô màu bề mặt (face coloring) của một đồ thị phẳng là phép tô mỗi mặt hay miền sao cho hai mặt có cùng đường biên không có cùng màu.

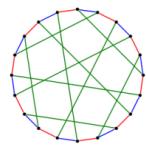
70

## BÀI TOÁN TÔ MÀU ĐỒ THỊ

💡 Sắc số là số màu tối thiểu để tô màu cho đồ thị thỏa mãn ràng buộc.



Tô màu đỉnh bởi 3 màu theo 12 cách



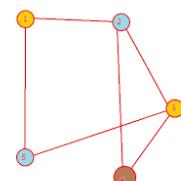
Tô màu cạnh bởi 3 màu



Tô màu bề mặt

## BÀI TOÁN TÔ MÀU ĐỒ THỊ

Cho đồ thị vô hướng, hãy tô màu tất cả các đỉnh với số màu ít nhất, sao cho 2 đỉnh nối với nhau được tô khác màu



72

## Bài toán tô màu đồ thị

Gọi  $k$  là số màu đã được dùng để tô màu

$k=1;$

Trong khi chưa tô hết các đỉnh

{

Chọn đỉnh  $p$  có bậc cao nhất

Lặp từ màu 1 đến màu  $k$

{

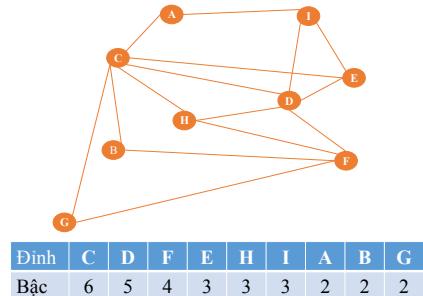
Nếu tồn tại màu  $i$  khác với màu các đỉnh kề với  $p$  thì chọn màu  $i$

}

Nếu đỉnh  $p$  chưa tô màu. Tô màu cho đỉnh  $p$  với màu mới  $k+1$

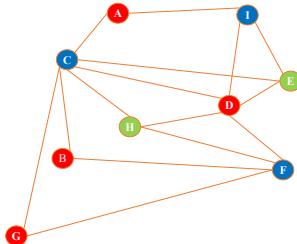
}

## Bài toán tô màu đồ thị - Ví dụ



## Bài toán tô màu đồ thị - Ví dụ

- [1] Tô màu 1 cho đỉnh C
- [2] Tô màu 2 cho đỉnh D
- [3] Tô màu 1 cho đỉnh F
- [4] Tô màu 3 cho đỉnh E
- [5] Tô màu 3 cho đỉnh H
- [6] Tô màu 1 cho đỉnh I
- [7] Tô màu 2 cho đỉnh A
- [8] Tô màu 2 cho đỉnh B
- [9] Tô màu 2 cho đỉnh G



## Bài toán tô màu đồ thị

Vấn đề cài đặt:

- Loại bỏ đỉnh đã tô

- Đánh dấu những màu bị cấm tô của mỗi đỉnh

## Bài toán tô màu đồ thị

Gọi  $k$  là số màu đã được dùng để tô màu  
 $k=1;$

Trong khi chưa tô hết các đỉnh

{

Chọn đỉnh  $p$  có bậc cao nhất

Lặp từ màu 1 đến màu  $k$

Nếu tồn tại màu  $i$  không bị cấm tô thì màu tô  $p = i$

Nếu đỉnh  $p$  chưa tô màu thì

$k=k+1$ , màu tô  $p = k$

Với những đỉnh  $q$  có nối với  $p$

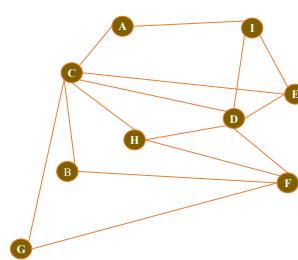
Hãy bậc  $q$ , thêm màu tô  $p$  vào danh sách màu cấm tô của  $q$

Gán bậc của đỉnh  $p = 0$

}

## Bài toán tô màu đồ thị - Ví dụ

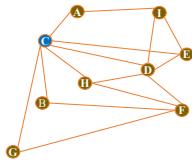
Áp dụng tô màu cho đồ thị sau



Dính	Bậc
C	6
D	5
F	4
E	3
H	3
I	3
A	2
B	2
G	2

### Bài toán tô màu đồ thị - Ví dụ

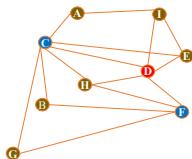
- **Bước 1:** Chọn đỉnh C
- Các đỉnh nối với C: A, B, D, E, G, H, G



Dính	Bậc	Bậc mới	Màu cám tô	Màu tô
C	6	<b>0</b>		1
D	5	<b>4</b>	1	
F	4	4		
E	3	<b>2</b>	1	
H	3	<b>2</b>	1	
I	3	3		
A	2	<b>I</b>	1	
B	2	<b>I</b>	1	
G	2	<b>I</b>	1	

### Bài toán tô màu đồ thị - Ví dụ

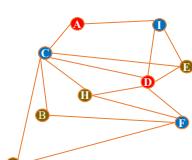
- **Bước 3:** Chọn đỉnh F
- Các đỉnh nối với F: B, H, G



Dính	Bậc	Bậc mới	Màu cám tô	Màu tô
F	3	<b>0</b>	2	1
E	1	1	1, 2	
H	1	<b>0</b>	1, 2	
I	2	2	2	
A	1	1	1	
B	1	<b>0</b>	1	
G	1	<b>0</b>	1	
C	0			1
D	0			2

### Bài toán tô màu đồ thị - Ví dụ

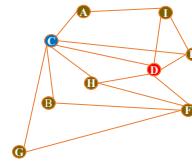
- **Bước 5:** Chọn đỉnh A



Dính	Bậc	Bậc mới	Màu cám tô	Màu tô
A	0	0	1	2
B	0	0	1	
E	0	0	1, 2	
H	0	0	1, 2	
G	0	0	1	
C	0			1
D	0			2
F	0			1
I	0			1

### Bài toán tô màu đồ thị - Ví dụ

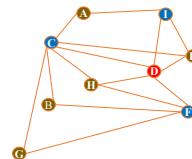
- **Bước 2:** Chọn đỉnh D
- Các đỉnh nối với D: E, F, H, I



Dính	Bậc	Bậc mới	Màu cám tô	Màu tô
D	4	<b>0</b>	1	2
F	4	<b>3</b>	2	
E	2	<b>I</b>	1, 2	
H	2	<b>I</b>	1, 2	
I	3	<b>2</b>	2	
A	1	<b>1</b>	1	
B	1	<b>1</b>	1	
G	1	<b>1</b>	1	
C	0	<b>0</b>		1

### Bài toán tô màu đồ thị - Ví dụ

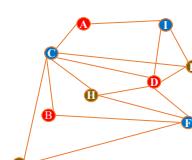
- **Bước 4:** Chọn đỉnh I
- Các đỉnh nối với I: A, E



Dính	Bậc	Bậc mới	Màu cám tô	Màu tô
I	2	0	2	1
A	1	<b>0</b>	1	
E	1	<b>0</b>	1, 2	
B	0	0	1	
H	0	0	1, 2	
G	0	0	1	
C	0			1
D	0			2
F	0			1

### Bài toán tô màu đồ thị - Ví dụ

- **Bước 6:** Chọn đỉnh B

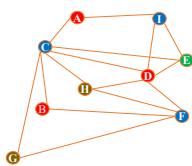


Dính	Bậc	Bậc mới	Màu cám tô	Màu tô
B	0	0	1	2
E	0	0	1, 2	
H	0	0	1, 2	
G	0	0	1	
C	0			1
D	0			2
F	0			1
I	0			1
A	0			2

### Bài toán tô màu đồ thị - Ví dụ

- Bước 7: Chọn đỉnh E

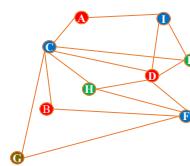
Định	Bậc	Bậc mới	Màu cấm tô	Màu tô
E	0	0	1, 2	3
H	0	0	1, 2	
G	0	0	1	
C	0			1
D	0			2
F	0			1
I	0			1
A	0			2
B	0			2



### Bài toán tô màu đồ thị - Ví dụ

- Bước 8: Chọn đỉnh H

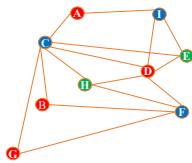
Định	BẬC	BẬC MỚI	MÀU CẤM TÔ	MÀU TÔ
H	0	0	1, 2	3
G	0	0	1	
C	0			1
D	0			2
F	0			1
I	0			1
A	0			2
B	0			2
E	0			3



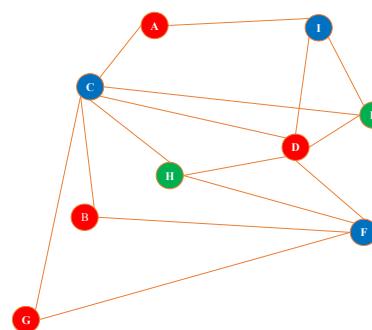
### Bài toán tô màu đồ thị - Ví dụ

- Bước 9: Chọn đỉnh G

Định	BẬC	BẬC MỚI	MÀU CẤM TÔ	MÀU TÔ
G	0	0	1	2
C	0			1
D	0			2
F	0			1
I	0			1
A	0			2
B	0			2
E	0			3
H	0			3



### Bài toán tô màu đồ thị - Bài tập



### Bài tập 1

Một công ty có 8 đài phát A, B, C, D, E, F, G, H có khoảng cách (km) được cho trong ma trận sau:

	A	B	C	D	E	F	G	H
A	0	100	50	30	200	150	40	120
B	0	30	80	120	50	200	150	
C	0	120	100	30	80	50		
D		0	50	120	150	30		
E			0	200	120	120		
F				0	180	150		
G					0	50		
H						0		

Các đài có khoảng cách  $\geq 100$  km không được dùng chung 1 trạm phát sóng. Hãy lắp đặt các trạm phát sóng sao cho số trạm ít nhất.

### Bài tập (gợi ý)

#### Đồ thị

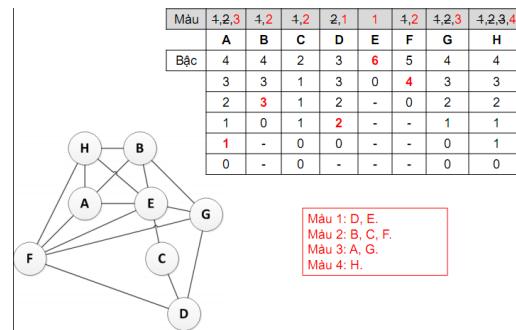
❖Định: các đài phát thanh.

❖Cung: nối giữa 2 đài có khoảng cách  $\geq 100$  km.

Ma trận quan hệ (hay đồ thị quan hệ như sau):

	A	B	C	D	E	F	G	H	BẬC
A	0	1	0	0	1	1	0	1	4
B	1	0	0	0	1	0	1	1	4
C	0	0	0	1	1	0	0	0	2
D	0	0	1	0	0	1	1	0	3
E	1	1	1	0	0	1	1	1	6
F	1	0	0	1	1	0	1	1	5
G	0	1	0	1	1	1	0	0	4
H	1	1	0	0	1	1	0	0	4

### Bài tập (gợi ý)



91

### Bài tập (gợi ý)

#### Đồ thị

- ❖ Định: các trận đấu.
- ❖ Cung: nối giữa hai trận đấu có cùng một đội bóng tham gia.

- 10 trận: AC, AD, AF, BC, BD, BE, CE, DE, DF, EF.

	AC	AD	AF	BC	BD	BE	CE	DE	DF	EF	Bắc
AC	0	1	1	1	0	0	1	0	0	0	4
AD	1	0	1	0	1	0	0	1	1	0	5
AF	1	1	0	0	0	0	0	0	1	1	4
BC	1	0	0	0	1	1	1	0	0	0	4
BD	0	1	0	1	0	1	0	1	1	0	5
BE	0	0	0	1	1	0	1	1	0	1	5
CE	1	0	0	1	0	1	0	1	0	1	5
DE	0	1	0	0	1	1	1	0	1	1	6
DF	0	1	1	0	1	0	0	1	0	1	5
EF	0	0	1	0	0	1	1	1	1	0	5

92

### Bài tập (gợi ý)

#### Đồ thị:

- ❖ Định: các chủ đề.
- ❖ Cung: nối giữa hai chủ đề không được diễn ra trong cùng một buổi.

	a	b	c	d	e	f	g	h	i	Bắc
a	0	0	1	1	0	0	1	0	0	3
b	0	0	0	1	1	0	0	0	0	2
c	1	0	0	1	0	1	0	0	0	3
d	1	1	1	0	1	1	1	0	0	6
e	0	1	0	1	0	0	1	1	0	4
f	0	0	1	1	0	0	1	0	0	3
g	1	0	0	1	1	1	0	1	1	6
h	0	0	0	0	1	0	1	0	1	3
i	0	0	0	0	0	0	1	1	0	2

93

### Bài tập 2

♣ Có 6 đội bóng A, B, C, D, E, F thi đấu vòng tròn (1 lượt) biết rằng các trận đấu sau đã xảy ra:

- ❖ A đã đấu với B,E.
- ❖ B đã đấu với A,F.
- ❖ C đã đấu với D,F.

♣ Mỗi đội chỉ được thi đấu 1 trận trong 1 tuần. Hãy sắp xếp các trận đấu vào các tuần sao cho số tuần diễn ra là ít nhất.

92

### Bài tập 3

♣ Một cuộc hội thảo có 9 chủ đề a, b, c, d, e, f, g, h, i biết rằng các chủ đề sau không được phép diễn ra trong cùng một buổi: ac, bde, adg, cdf, dfg, egh, ghi.

♣ Hãy sắp xếp các chủ đề vào các buổi sao cho số buổi diễn ra là ít nhất.

94

### Slide được tham khảo từ

#### • Slide được tham khảo từ:

- Slide CTDL GT, Khoa Khoa Học Máy Tính, ĐHCNTT
- Congdongcviet.com
- Cplusplus.com

95



97