

# TOÁN RỜI RẠC

## Biểu Diễn Đồ Thị Trên Máy Tính

NGUYỄN HẢI TRIỀU<sup>1</sup>

<sup>1</sup>Bộ môn Kỹ thuật phần mềm,  
Khoa Công nghệ thông tin, Trường ĐH Nha Trang

NhaTrang, February 2022

# Tổng quan

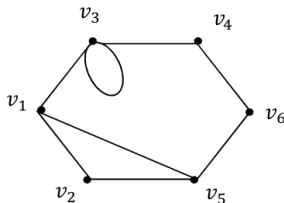
- 1 Ma trận kề, ma trận trọng số
- 2 Các thuật toán tìm kiếm trên đồ thị

# Ma trận kề

## Định nghĩa 1.1

Xét đồ thị  $G = (V, E)$  với tập đỉnh  $V$  và tập cạnh  $E$ . Ma trận kề của đồ thị là ma trận vuông  $A_{n \times n}$  xác định như sau:

$$a_{ij} = \begin{cases} 0 & \text{nếu } v_i v_j \notin E, \\ 1 & \text{nếu } v_i v_j \in E. \end{cases}$$



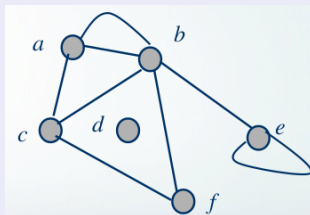
$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

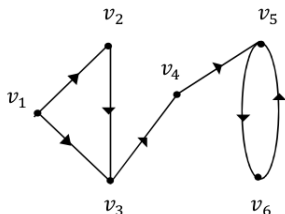
Hình 1: Ví dụ đồ thị vô hướng và ma trận kề tương ứng

## Tính chất của ma trận kề của đồ thị vô hướng

- Là ma trận đối xứng (qua đường chéo chính)
- Các phần tử trên đường chéo chính bằng 0 (trừ đồ thị có khuyên)
- Tổng các phần tử trên một hàng (cột) bằng bậc của đỉnh tương ứng.
- Giá trị của phần tử ở dòng  $i$ , cột  $j$  của ma trận  $A_m$  là số đường đi từ đỉnh  $i$  đến đỉnh  $j$  có độ dài  $m$ .

### Ví dụ 1.1 (Tìm ma trận kề cho đồ thị vô hướng sau)





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

Hình 2: Ví dụ đồ thị có hướng và ma trận kề tương ứng

## Tính chất của ma trận kề của đồ thị có hướng

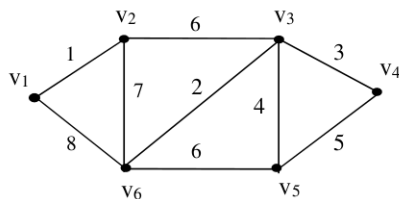
- Không phải là ma trận đối xứng
- Các phần tử trên đường chéo chính bằng 0
- Tổng các phần tử trên một hàng (cột) bằng bán bậc ra/vào của các đỉnh tương ứng.
- Giá trị của phần tử ở dòng  $i$ , cột  $j$  của ma trận  $A_m$  là số đường đi từ đỉnh  $i$  đến đỉnh  $j$  có độ dài  $m$ .

# Ma trận trọng số

## Ma trận trọng số

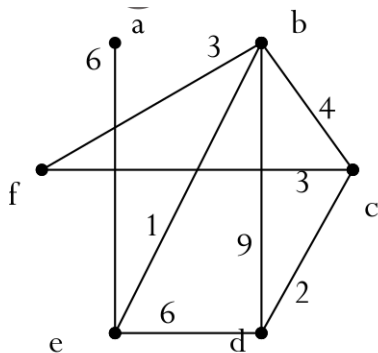
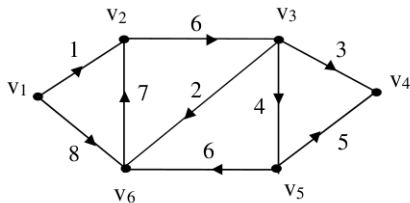
Ma trận của đồ thị có trọng số được xây dựng bằng cách thay giá trị 1 bằng trọng số của cạnh tương ứng. Ký hiệu  $c(u, v)$  là trọng số của cạnh/cung  $(u, v)$ , trong đó:

- $c = \{0, +\infty, -\infty\}$  nếu 2 đỉnh  $u, v$  không kề nhau
- $c =$  giá trị trọng số của cạnh/cung  $(u, v)$  nếu có



$$C = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 8 \\ 1 & 0 & 6 & 0 & 0 & 7 \\ 0 & 6 & 0 & 4 & 3 & 2 \\ 0 & 0 & 4 & 0 & 5 & 6 \\ 0 & 0 & 3 & 5 & 0 & 0 \\ 8 & 7 & 2 & 0 & 6 & 0 \end{bmatrix}$$

**Hình 3:** Đồ thị vô hướng và ma trận trọng số tương ứng

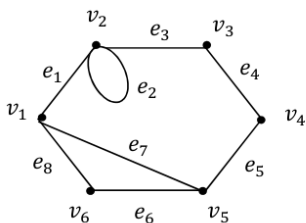


Hình 4: Tìm ma trận trọng số cho các đồ thị có hướng trên?

# Ma trận liên thuộc đỉnh-cạnh

## Định nghĩa 1.2

Xét đồ thị  $G = (V, E)$  với tập đỉnh  $V = \{v_1, v_2, \dots, v_n\}$  và tập cạnh  $E = \{e_1, e_2, \dots, e_m\}$ . Gọi  $M = m_{ij}$  là ma trận liên thuộc của  $G$ , trong đó  $m_{ij} = 1$  nếu đỉnh  $v_i$  thuộc cạnh  $e_j$ ,  $m_{ij} = 0$  nếu đỉnh  $v_i$  không thuộc cạnh  $e_j$ .

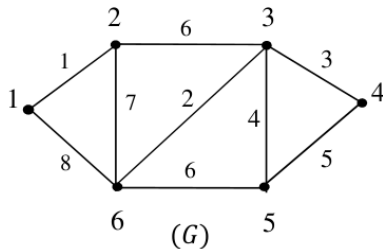


$$M = \begin{matrix} & \begin{matrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 \end{matrix} \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{matrix} & \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \end{matrix}$$

**Hình 5:** Giả đồ thị  $G$  và ma trận liên thuộc  $M$  tương ứng



Đồ thị thưa—đồ thị có số cạnh nhỏ hơn  $n(n-1)/4$  thường được lưu trữ dưới dạng danh sách cạnh. Các cạnh của đồ thị được lưu thành 2 dãy, 1 dãy chứa đỉnh đầu, 1 dãy chứa đỉnh cuối của cạnh  $e$  tương ứng:  $Dau[e]$  và  $Cuoi[e]$ . Bổ sung thêm một dãy nữa nếu cạnh có trọng số.



| <i>Đỉnh đầu</i> | <i>Đỉnh cuối</i> | <i>Trọng số</i> |
|-----------------|------------------|-----------------|
| 1               | 2                | 1               |
| 1               | 6                | 8               |
| 2               | 3                | 6               |
| 2               | 6                | 7               |
| 3               | 4                | 3               |
| 3               | 5                | 4               |
| 3               | 6                | 2               |
| 4               | 5                | 5               |
| 5               | 6                | 6               |

# Bài tập ôn tập

## Ví dụ 1.2

*Tự vẽ ra 2 đồ thị, 1 có hướng, 1 vô hướng, mỗi đồ thị có  $n > 5$  đỉnh, có lớn  $\geq n$  cạnh. Biểu diễn các đồ thị vừa vẽ bằng tất cả các cách có thể.*

## Ví dụ 1.3

*Vẽ một đồ thị liên thông  $\leq 10$  đỉnh có ít nhất 1 chu trình đơn ứng với mỗi độ dài từ 5 đến 9 nhưng không chứa chu trình đơn có độ dài khác.*

# Các thuật toán tìm kiếm trên đồ thị

- Tìm kiếm trên đồ thị là các bài toán xác định các yếu tố cơ bản của đồ thị như **xác định đỉnh, đường đi, chu trình, cây khung, tính liên thông**.
- Một số ứng dụng điển hình của bài toán này trong cuộc sống thường nhật như tìm kiếm tập tin trong máy tính hay điện thoại, tìm kiếm các website trong mạng máy tính theo các từ khóa, ...
- Hai thuật toán cơ bản về tìm kiếm trên đồ thị, đó là *thuật toán tìm kiếm ưu tiên theo chiều sâu – DFS (Depth First Search)* và *tìm kiếm ưu tiên theo chiều rộng – BFS (Breadth First Search)*

# Tìm kiếm ưu tiên theo chiều sâu

```
1      DFS(G,v)    ( v is the vertex where the search starts )
2          Stack S := {};    ( start with an empty stack )
3          for each vertex u, set visited[u] := false;
4          push S, v;
5          while (S is not empty) do
6              u := pop S;
7              if (not visited[u]) then
8                  visited[u] := true;
9                  for each unvisited neighbour w of u
10                     push S, w;
11             end if
12         end while
13     END DFS()
```

SV tham khảo chi tiết thuật toán tại đây!

# Tìm kiếm ưu tiên theo chiều sâu

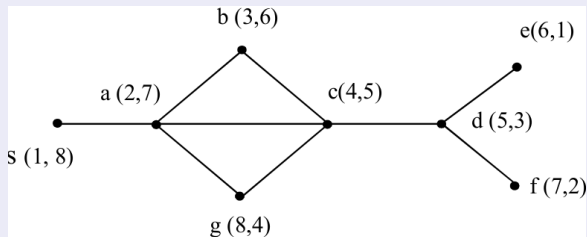
## Giải thích thuật toán DFS

- Giả sử thuật toán DFS bắt đầu từ đỉnh  $v_1$  (đỉnh xuất phát), hiển nhiên  $v_1$  là đỉnh đã được thăm (tìm thấy).
- chọn  $v_2$  là một trong những đỉnh kề với  $v_1 \Rightarrow v_2$  đã được thăm. Tổng quát cho đỉnh  $v_k$  và  $v_{k+1}$
- nếu không còn đỉnh nào kề với  $v_k$  ( $v_k$  là đỉnh treo hoặc các đỉnh kề với nó đã được thăm) thì ta nói rằng đỉnh  $v_k$  đã **được duyệt**
- quay lui về đỉnh  $v_{k-1}$  để tiếp tục tìm kiếm từ đỉnh này.
- *thuật toán kết thúc khi mọi đỉnh đã được thăm-duyet*
- DFS có độ phức tạp thời gian đa thức  $O(n^2)$

# Tìm kiếm ưu tiên theo chiều sâu

## Ví dụ 2.1

Minh họa một thứ tự thăm-duyet các đỉnh của đồ thị  $G$  bởi thuật toán DFS xuất phát từ đỉnh  $s$ .



| ... | ...  | ...   |
|-----|------|-------|
| 8   | $g$  | $s$   |
| 7   | $f$  | $a$   |
| 6   | $e$  | $b$   |
| 5   | $d$  | $c$   |
| 4   | $c$  | $g$   |
| 3   | $b$  | $d$   |
| 2   | $a$  | $f$   |
| 1   | $s$  | $e$   |
| TT  | Thăm | Duyệt |

Đỉnh được thăm càng muộn sẽ càng sớm trở thành đã duyệt xong-Last Out First In (LOFI), là hệ quả tất yếu với các đỉnh được thăm sẽ được kết nạp vào ngăn xếp Stack.

# Tìm kiếm ưu tiên theo chiều rộng

```
1 BFS (G, s) #Where G is the graph and s is the source node
2     let Q be queue. Inserting s in queue until all its
        neighbour vertices are marked.
3     Q.enqueue(s)
4     mark s as visited.
5     while ( Q is not empty)
6         #Removing that vertex from queue, whose neighbour will
            be visited now
7         v = Q.dequeue( )
8         #processing all the neighbours of v
9         for all neighbours w of v in Graph G
10            if w is not visited
11                #Stores w in Q to further visit its
                    neighbour
12                Q.enqueue( w )
13                mark w as visited.
```

SV tham khảo chi tiết thuật toán tại đây!

## Giải thích thuật toán BFS

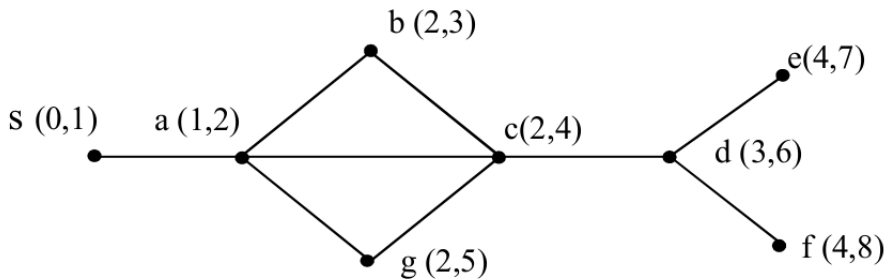
- BFS được xây dựng trên cơ sở hàng đợi (Queue) - vào trước ra trước=> **đỉnh được thăm càng sớm sẽ càng sớm trở thành đã duyệt xong và rời khỏi hàng đợi.**
- Giả sử bắt đầu từ đỉnh  $s$ . Đỉnh này xem như đã được thăm-duyet và gọi là **đỉnh mức 0**
- Xét tất cả các đỉnh kề với nó và được gọi là các **đỉnh mức 1**. Các đỉnh này xem như đã được thăm-duyet, sắp xếp chúng theo một thứ tự nào đó (tương ứng với một hàng đợi)
- Tập các đỉnh kề với các đỉnh mức 1 được gọi là đỉnh mức 2, các đỉnh này xem như đã được thăm-duyet
- Tiếp tục tương tự như vậy cho đến khi tất cả các đỉnh đã được thăm-duyet
- Mức của một đỉnh  $v$  cho biết độ dài của đường đi ngắn nhất (theo số cạnh) từ  $s$  đến  $v$  trong đồ thị  $G$ .



## Ví dụ 2.2

Minh họa một thứ tự thăm-duyet các đỉnh của đồ thị bởi thuật toán BFS xuất phát từ đỉnh  $s$  và một hàng đợi

$Queue = (s, a, b, c, g, d, e, f)$  tương ứng.



Mỗi đỉnh được gán số mức và thứ tự thăm-duyet. Chẳng hạn, đỉnh  $c$  có mức 2 và thứ tự thăm-duyet là 4.

# Ứng dụng cụ thể của DFS và BFS

Ví dụ 2.3 (Tìm đường đi từ đỉnh  $s$  đến đỉnh  $t$  của một đồ thị  $G$  cho trước. )

- ➊ Xuất phát từ đỉnh  $s$  của  $G$ , lần lượt thăm-duyet đỉnh bằng thuật toán  $DFS(s)$  hoặc  $BFS(s)$  cho đến khi gặp đỉnh  $t$ .
- ➋ không có đường đi từ  $s$  đến  $t \Rightarrow$  Đồ thị không liên thông
- ➌ Vết của đường đi từ đỉnh  $s$  đến đỉnh  $t$ : bắt đầu từ đỉnh  $t$ , đi theo hướng ngược lại, chọn đỉnh  $w$  kề với  $t$  mà từ đó dẫn đến đỉnh  $t$ . Thuật toán  $BFS$  luôn cho kết quả là đường đi ngắn nhất theo số cạnh của  $G$ .

# Tài liệu tham khảo



Đ.N. An

Giáo Trình Toán Rời Rạc. *Trường ĐH Nha Trang, (2021).*



Giáo trình Toán rời rạc

Giáo trình Toán Rời Rạc. *Trường DHSP Huế. (2003), 22-35.*



N.T. Nhật

Bài giảng Toán Rời Rạc. *Trường ĐH KHTN Tp.HCM. (2011).*