

MÔN LÝ THUYẾT ĐỒ THỊ

Giảng viên: ThS. Mai Chiém Tuấn

1

CHƯƠNG III

CÁC THUẬT TOÁN TÌM KIẾM TRÊN ĐỒ THỊ VÀ ỨNG DỤNG

2

Chương III: CÁC THUẬT TOÁN TRÊN ĐỒ THỊ

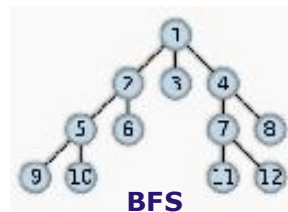
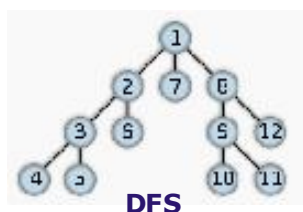


- 1 III.1 Tìm kiếm theo chiều sâu - DFS
- 2 III.2 Tìm kiếm theo chiều rộng - BFS
- 3 III.3 Tìm đường đi, kiểm tra liên thông
- 4 III.4 Bài tập

Giới thiệu



- ❖ Thuật toán tìm kiếm trên đồ thị được xây dựng để duyệt tất cả các đỉnh của đồ thị sao cho mỗi đỉnh được viếng thăm đúng một lần.
 - Thuật toán tìm kiếm theo chiều sâu trên đồ thị (DFS - Depth First Search).
 - Thuật toán tìm kiếm theo chiều rộng trên đồ thị (Breadth First Search).



III.1 Tìm kiếm theo chiều sâu - DFS



III.1.1 Ý tưởng của thuật toán

- ❖ B1. Xuất phát từ 1 đỉnh cho trước nào đó.
- ❖ B2. Xử lý đỉnh này và đánh dấu để không xử lý lần sau.
- ❖ B3. Đưa tất cả các đỉnh kề với nó vào danh sách xử lý và chọn 1 đỉnh để xử lý tiếp theo.
- ❖ B4. Quay lại B2 cho đến khi không còn đỉnh trong danh sách.
- ❖ Để kiểm tra việc duyệt mỗi đỉnh đúng một lần:
 - Sử dụng một mảng `chuaxet[]` gồm n phần tử (n đỉnh).
 - `chuaxet[u]=1` (u chưa xét), `chuaxet[u]=0` (u đã xét).

III.1 Tìm kiếm theo chiều sâu - DFS



III.1.2 Cài đặt mã giả đệ quy - DFS(u)

Thuật toán DFS(u) có thể được mô tả bằng thủ tục đệ qui như sau:

Thuật toán DFS (u): //u là đỉnh bắt đầu duyệt

Begin

```

<Thăm đỉnh u>; //Duyệt đỉnh u
chuaxet[u] := FALSE; //Xác nhận đỉnh u đã duyệt
for each v ∈ ke(u) do //Lấy mỗi đỉnh v ∈ ke(u).
  if (chuaxet[v]) then //Nếu đỉnh v chưa duyệt
    DFS(v); //Duyệt theo chiều sâu từ đỉnh v
  EndIf;
EndFor;

```

End.

III.1 Tìm kiếm theo chiều sâu - DFS



III.1.3 Cài đặt mã giả Stack - DFS(u)

Thuật toán DFS(u) có thể được mô tả bằng cấu trúc Stack như sau:

Thuật toán DFS (u): //u là đỉnh bắt đầu duyệt
Begin

Bước 1 (Khởi tạo):

```
stack =  $\emptyset$ ; //Khởi tạo stack là  $\emptyset$ 
Push(stack, u); //Đưa đỉnh u vào ngăn xếp
<Thăm đỉnh u>; //Duyệt đỉnh u
chuaxet[u] = False; //Xác nhận đỉnh u đã duyệt
```

III.1 Tìm kiếm theo chiều sâu - DFS



III.1.3 Cài đặt mã giả Stack - DFS(u)

Bước 2 (Lặp):

```
while (stack  $\neq \emptyset$ ) do
    s = Pop(stack); //Loại đỉnh ở đầu ngăn xếp
    for each t  $\in$  Ke(s) do //Lấy mỗi đỉnh t  $\in$  Ke(s)
        if ( chuaxet[t] ) then //Nếu t đúng là chưa duyệt
            <Thăm đỉnh t>; //Duyệt đỉnh t
            chuaxet[t] = False; //đỉnh t đã duyệt
            Push(stack, s); //Đưa s vào stack
            Push(stack, t); //Đưa t vào stack
            break; //Chỉ lấy một đỉnh t
        EndIf;
    EndFor;
EndWhile;
```

Bước 3 (Trả lại kết quả):

```
Return(<Tập đỉnh đã duyệt>);
End.
```

III.1 Tìm kiếm theo chiều sâu - DFS



III.1.4 Độ phức tạp của thuật toán DFS(u)

- ❖ Độ phức tạp thuật toán DFS(u) phụ thuộc vào phương pháp biểu diễn đồ thị.
 - Độ phức tạp thuật toán là $O(n^2)$ trong trường hợp đồ thị biểu diễn dưới dạng ma trận kề, với n là số đỉnh của đồ thị.
 - Độ phức tạp thuật toán là $O(n.m)$ trong trường hợp đồ thị biểu diễn dưới dạng danh sách kề, với n là số đỉnh của đồ thị, m là số cạnh của đồ thị.
 - Độ phức tạp thuật toán là $O(\max(n, m))$ trong trường hợp đồ thị biểu diễn dưới dạng danh sách kề, với n là số đỉnh của đồ thị, m là số cạnh của đồ thị.

III.1 Tìm kiếm theo chiều sâu - DFS

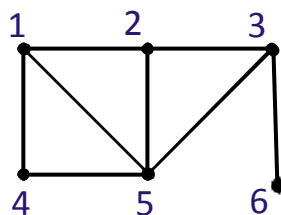


III.1.5 Kiểm nghiệm thuật toán - DFS

- ❖ Ví dụ 3.1: Xét đồ thị gồm 6 đỉnh như sau:

- ❖ Bắt đầu DFS(1):

- DS các đỉnh kề với 1: 2, 4, 5
- Đỉnh đã duyệt: 1.
- Chọn đỉnh 2: DFS(2)
- DS các đỉnh kề với 2: 1, 3, 5
- Đỉnh đã duyệt: 1, 2.
- Chọn đỉnh 3: DFS(3)
- DS các đỉnh kề với 3: 2, 5, 6
- Đỉnh đã duyệt: 1, 2, 3.
- Chọn đỉnh 5: DFS(5)
- DS các đỉnh kề với 5: 1, 2, 4
- Đỉnh đã duyệt: 1, 2, 3, 5.
- Chọn đỉnh 4: DFS(4)
- DS các đỉnh kề với 4: 1, 5
- Đỉnh đã duyệt: 1, 2, 3, 5, 4.
- Quay lại xét các đỉnh kề với 5: 1, 2, 4 (đã duyệt)
- Quay lại xét các đỉnh kề với 3: 2, 5, 6
- Chọn đỉnh 6: DFS(6)
- DS các đỉnh kề với 6: 3 (đã duyệt) => Kết quả đỉnh đã duyệt: 1, 2, 3, 5, 4, 6.



III.1 Tìm kiếm theo chiều sâu - DFS

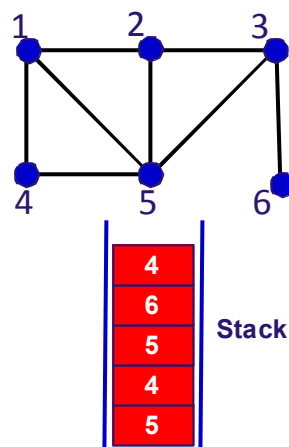


III.1.5 Kiểm nghiệm thuật toán - DFS

❖ Ví dụ 3.2: Xét đồ thị gồm 6 đỉnh như sau:

Bắt đầu DFS(1):

- Đưa 1 vào Stack
- Lấy 1 ra xử lý, đưa 5, 4, **2** vào Stack
- Lấy **2** ra xử lý, đưa 5, **3** vào Stack
- Lấy **3** ra xử lý, đưa 6, **5** vào Stack
- Lấy **5** ra xử lý, đưa **4** vào Stack
- Lấy **4** ra xử lý. Không đưa gì vào Stack
- Lấy **6** ra xử lý. Không đưa gì vào Stack
- Lấy **5** ra. Không xử lý (*vì đã xử lý rồi*)
- Lấy **4** ra. Không xử lý
- Lấy **5** ra. Không xử lý



Thứ tự duyệt:



Lý thuyết đồ thị - Chương III: Các Thuật Toán Tìm Kiếm Trên Đồ Thị

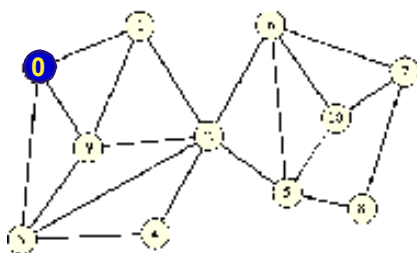
11

III.1 Tìm kiếm theo chiều sâu - DFS

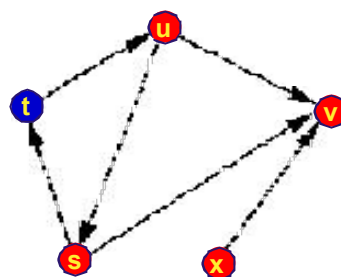


III.1.5 Kiểm nghiệm thuật toán - DFS

❖ Ví dụ 3.3: Áp dụng DFS, hãy thể hiện thứ tự duyệt các đỉnh trong đồ thị sau:



Đáp án: 0 1 2 3 4 9 5 6 7 8 10



Đáp án: t u s v

Đỉnh x không được duyệt

Lý thuyết đồ thị - Chương III: Các Thuật Toán Tìm Kiếm Trên Đồ Thị

12

III.2 Tìm kiếm theo chiều rộng - BFS



III.2.1 Ý tưởng của thuật toán

- ❖ **B1.** Xuất phát từ 1 đỉnh cho trước nào đó.
- ❖ **B2.** Xử lý đỉnh này và đánh dấu để không xử lý lần sau.
- ❖ **B3.** Đưa tất cả các đỉnh kề với nó vào danh sách xử lý và lần lượt xử lý các đỉnh kề với đỉnh đang xét
- ❖ **B4.** Quay lại **B2** cho đến khi không còn đỉnh trong danh sách.
- ❖ Để kiểm tra việc duyệt mỗi đỉnh đúng một lần:
 - Sử dụng một mảng **chuaxet[]** gồm **n** phần tử (**n** đỉnh).
 - **chuaxet[u]=1** (**u** chưa xét), **chuaxet[u]=0** (**u** đã xét).

III.2 Tìm kiếm theo chiều rộng - BFS



III.2.2 Cài đặt mã giả - BFS(u)

Begin

Bước 1 (Khởi tạo):

Queue = $\emptyset$; Push(Queue, u); chuaxet[u] = False;

Bước 2 (Lặp):

```
while (Queue  $\neq \emptyset$ ) do
  s = Pop(Queue); <Thăm đỉnh s>;
  for each t  $\in$  Ke(s) do
    if ( chuaxet[t] ) then
      Push(Queue, t);
      chuaxet[t] = False;
    EndIf ;
  EndFor ;
EndWhile ;
```

Bước 3 (Trả lại kết quả):

Return(<Tập đỉnh được duyệt>);

End.

III.2 Tìm kiếm theo chiều rộng - BFS

III.2.3 Độ phức tạp của thuật toán BFS(u)

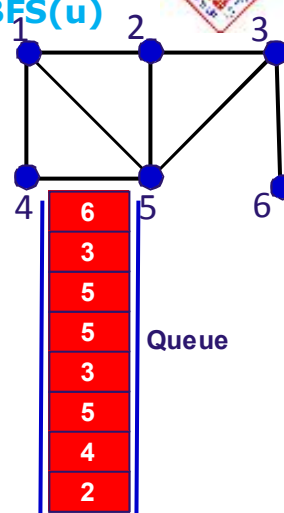
- ❖ Độ phức tạp thuật toán BFS(u) phụ thuộc vào phương pháp biểu diễn đồ thị.
 - Độ phức tạp thuật toán là $O(n^2)$ trong trường hợp đồ thị biểu diễn dưới dạng ma trận kề, với n là số đỉnh của đồ thị.
 - Độ phức tạp thuật toán là $O(n.m)$ trong trường hợp đồ thị biểu diễn dưới dạng danh sách kề, với n là số đỉnh của đồ thị, m là số cạnh của đồ thị.
 - Độ phức tạp thuật toán là $O(\max(n, m))$ trong trường hợp đồ thị biểu diễn dưới dạng danh sách kề, với n là số đỉnh của đồ thị, m là số cạnh của đồ thị.

III.2 Tìm kiếm theo chiều rộng - BFS

III.2.4 Kiểm nghiệm thuật toán BFS(u)

Ví dụ: 3.4: Bắt đầu BFS(1):

- ❖ Đưa **1** vào Queue
- ❖ Lấy **1** ra xử lý, đưa **2**, 4, 5 vào Queue
- ❖ Lấy **2** ra xử lý, đưa **3**, 5 vào Queue
- ❖ Lấy **4** ra xử lý, đưa 5 vào Queue
- ❖ Lấy **5** ra xử lý, đưa 3 vào Queue
- ❖ Lấy **3** ra xử lý. Đưa 6 vào Queue
- ❖ Lấy 5 ra. *Không xử lý (vì đã xử lý rồi)*
- ❖ Lấy 5 ra. *Không xử lý*
- ❖ Lấy 3 ra. *Không xử lý*
- ❖ Lấy 6 ra xử lý. *Không đưa gì vào Queue*



Thứ tự duyệt:

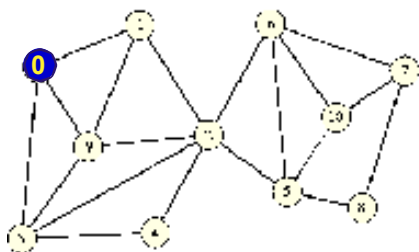
1	2	4	5	3	6
---	---	---	---	---	---

III.2 Tìm kiếm theo chiều rộng - BFS

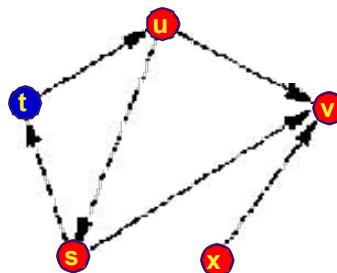


III.2.4 Kiểm nghiệm thuật toán BFS(u)

- ❖ Ví dụ 3.5: Áp dụng BFS, hãy thể hiện thứ tự duyệt các đỉnh trong đồ thị sau:



Đáp án: 0 1 3 9 2 4 5 6 8 10 7



Đáp án: t u s v

Đỉnh x không được duyệt

III.3 Tìm đường đi, kiểm tra liên thông



Ứng dụng của thuật toán DFS và BFS

- ❖ Những ứng dụng cơ bản của thuật toán DFS và BFS được đề cập bao gồm:
- Duyệt tất cả các đỉnh của đồ thị.
 - Duyệt tất cả các thành phần liên thông của đồ thị.
 - Tìm đường đi từ đỉnh s đến đỉnh t trên đồ thị.
 - Duyệt các đỉnh trự trên đồ thị vô hướng.
 - Duyệt các đỉnh trụ trên đồ thị vô hướng.
 - Duyệt các cạnh cầu trên đồ thị vô hướng.
 - Định chiều đồ thị vô hướng.
 - Duyệt các đỉnh rẽ nhánh của cặp đỉnh s, t.
 - Xác định tính liên thông mạnh trên đồ thị có hướng.
 - Xác định tính liên thông yếu trên đồ thị có hướng.
 - Thuật toán tìm kiếm theo chiều rộng trên đồ thị.
 - Xây dựng cây khung của đồ thị vô hướng liên thông

III.3 Tìm đường đi, kiểm tra liên thông



III.3.1 Tìm đường đi giữa 2 đỉnh

❖ Bài toán

- Cho đồ thị $G = \langle V, E \rangle$ (vô hướng hoặc có hướng), trong đó V là tập đỉnh, E là tập cạnh. Bài toán đặt ra là hãy **tìm đường đi từ đỉnh $s \in V$ đến đỉnh $t \in V$** ?

❖ Phương pháp

- Bắt đầu từ đỉnh s , Sử dụng **DFS** hoặc **BFS** để duyệt đồ thị.
 - Nếu $t \in \text{DFS}(s)$ hoặc $t \in \text{BFS}(s)$ thì ta có thể kết luận có đường đi từ s đến t trên đồ thị.
 - Nếu $t \notin \text{DFS}(s)$ hoặc $t \notin \text{BFS}(s)$ thì ta có thể kết luận không có đường đi từ s đến t trên đồ thị.
- Sử dụng thêm mảng **Truoc[]** để ghi nhận đường đi từ s đến t .

III.3 Tìm đường đi, kiểm tra liên thông



III.3.1 Tìm đường đi giữa 2 đỉnh

❖ Thuật toán DFS đệ quy:

Thuật toán DFS (u): // u là đỉnh bắt đầu duyệt
Begin

```

<Thăm đỉnh u>; //Duyệt đỉnh u
chuaxet[u] := FALSE; //Xác nhận đỉnh u đã duyệt
for each v eke(u) do //Lấy mỗi đỉnh v eKe(u).
    if (chuaxet[v]) then //Nếu đỉnh v chưa duyệt
        Truoc[u]=v; Ghi nhận truoc[v] là u
        DFS(v); //Duyệt theo chiều sâu từ đỉnh v
    EndIf;
EndFor;
End.
```

III.3 Tìm đường đi, kiểm tra liên thông



III.3.1 Tìm đường đi giữa 2 đỉnh

❖ Thuật toán DFS (s) dùng Stack:

Bước 2 (Lặp):

```

while (stack  $\neq \emptyset$ ) do
    u = Pop(stack); //Loại đỉnh ở đầu ngăn xếp
    for each  $v \in Ke(u)$  do //Lấy mỗi đỉnh  $t \in Ke(s)$ 
        if ( chuaxet[v] ) then //Nếu t đúng là chưa duyệt
            chuaxet[v] = False; //đỉnh t đã duyệt
            Push(stack, u); //Đưa s vào stack
            Push(stack, v); //Đưa t vào stack
            truoc[v] = u; //Ghi nhận truoc[v] là u
            break; //Chỉ lấy một đỉnh t
        EndIf;
    EndFor;
EndWhile;

```

III.3 Tìm đường đi, kiểm tra liên thông



III.3.1 Tìm đường đi giữa 2 đỉnh

❖ Thuật toán BFS(s):

Bước 1 (Khởi tạo):

Queue = $\emptyset$; Push(Queue, s); chuaxet[s] = False;

Bước 2 (Lặp):

```

while (Queue  $\neq \emptyset$ ) do
    u = Pop(Queue);
    for each  $v \in Ke(u)$  do
        if ( chuaxet[v] ) then
            Push(Queue, v); chuaxet[v] = False;
            truoc[v] = u; //Ghi nhận truoc[v] là u
        EndIf;
    EndFor;
EndWhile;

```

Bước 3 (Trả lại kết quả) :

Return(<Tập đỉnh được duyệt>);

End.

III.3 Tìm đường đi, kiểm tra liên thông



III.3.1 Tìm đường đi giữa 2 đỉnh

```

Thuật toán Ghi-Nhan-Duong-Di (s, t) {
  if ( truoc[t] == 0 ) {
    <Không có đường đi từ s đến t>;
  }
  else {
    <Đưa ra đỉnh t>; //Đưa ra trước đỉnh t
    u = truoc[t]; //u là đỉnh trước khi đến được t
    while (u ≠ s) { //Lặp nếu u chưa phải là s
      <Đưa ra đỉnh u>; //Đưa ra đỉnh u
      u = truoc[u]; // Lăn ngược lại đỉnh truoc[u].
    }
    <Đưa ra đỉnh s>;
  }
}.

```

III.3 Tìm đường đi, kiểm tra liên thông



III.3.2 Tìm các thành phần liên thông

❖ Bài toán

- Cho đồ thị đồ thị vô hướng $G = \langle V, E \rangle$, trong đó V là tập đỉnh, E là tập cạnh. Bài toán đặt ra là **xác định các thành phần liên thông** của $G = \langle V, E \rangle$?

❖ Phương pháp

- Đồ thị liên thông** thì phép duyệt theo thủ tục **DFS(u)** hoặc **BFS(u)** được gọi đến đúng một lần và khi đó **DFS(u)=V** và **BFS(u)=V**.
- Nếu đồ thị không liên thông số thành phần liên thông của đồ thị bằng đúng số lần gọi tới thủ tục **DFS()** hoặc **BFS()**.
- Sử dụng thêm biến **solt** để nghi nhận thành phần liên thông.

III.3 Tìm đường đi, kiểm tra liên thông

III.3.2 Tìm các thành phần liên thông

Thuật toán Duyệt-TPLT:

Begin

Bước 1(Khởi tạo):

$solt = 0$; //Khởi tạo số thành phần liên thông

Bước 2(Lặp):

for ($u = 1$; $u \leq n$; $u++$) do //lặp trên tập đỉnh

if (chuaxet[u]) then

$solt = solt + 1$; //Ghi nhận số TPLT

<Ghi nhận các đỉnh thuộc TPLT>;

BFS (u); //DFS(u);

endif;

endfor;

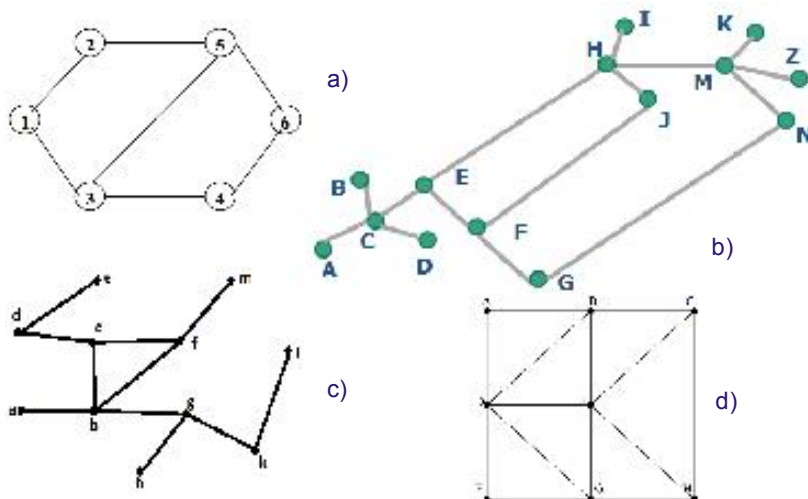
Bước 3(Trả lại kết quả):

Return(solt);

end.

III.4 Bài tập

1. Cho đồ thị vô hướng G như hình vẽ sau.



III.4 Bài tập



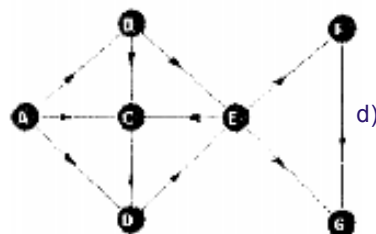
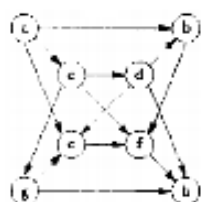
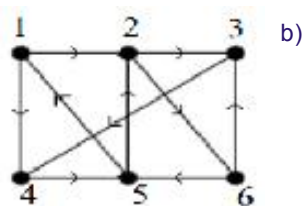
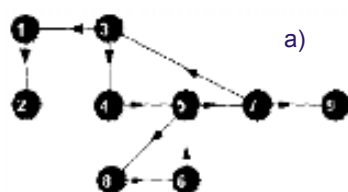
❖ Hãy thực hiện:

- Hình a) Tìm $DFS(1)=?$, $DFS(5)=?$, $BFS(1)=?$, $BFS(5)=?$, Tìm đường đi từ 1 đến 6 bằng thuật toán DFS, BFS?
- Hình b) Tìm $DFS(A)=?$, $DFS(I)=?$, $BFS(A)=?$, $BFS(I)=?$, Tìm đường đi từ A đến Z bằng thuật toán DFS, BFS?
- Hình c) Tìm $DFS(a)=?$, $DFS(c)=?$, $BFS(a)=?$, $BFS(c)=?$, Tìm đường đi từ c đến h bằng thuật toán DFS, BFS?
- Hình d) Tìm $DFS(B)=?$, $BFS(B)=?$, $BFS(c)=?$, Tìm đường đi từ A đến H bằng thuật toán DFS, BFS?

III.4 Bài tập



2. Cho đồ thị có hướng G như hình vẽ sau.



III.4 Bài tập



❖ Hãy thực hiện:

- Hình a) Tìm $DFS(3)=?$, $DFS(7)=?$, $BFS(3)=?$, $BFS(7)=?$, Tìm đường đi từ 1 đến 6 bằng thuật toán DFS, BFS?
- Hình b) Tìm $DFS(1)=?$, $DFS(2)=?$, $BFS(1)=?$, $BFS(2)=?$, Tìm đường đi từ 1 đến 6 bằng thuật toán DFS, BFS?
- Hình c) Tìm $DFS(a)=?$, $DFS(c)=?$, $BFS(a)=?$, $BFS(c)=?$, Tìm đường đi từ a đến i bằng thuật toán DFS, BFS?
- Hình d) Tìm $DFS(B)=?$, $BFS(B)=?$, Tìm đường đi từ A đến G bằng thuật toán DFS, BFS?

III.4 Bài tập



3. Cho đồ thị vô hướng liên thông $G = \langle V, E \rangle$ như dưới đây:

- $Ke(1) = \{2, 3, 4\}$. $Ke(2) = \{1, 3, 4, 6\}$.
- $Ke(3) = \{1, 2, 4, 5\}$. $Ke(4) = \{1, 2, 3, 7\}$.
- $Ke(5) = \{3, 6, 7, 8, 12\}$. $Ke(6) = \{2, 5, 7, 12\}$.
- $Ke(7) = \{4, 5, 6, 8\}$. $Ke(8) = \{5, 7, 12\}$.
- $Ke(9) = \{10, 11, 13\}$. $Ke(10) = \{9, 11, 12, 13\}$.
- $Ke(11) = \{9, 10, 13\}$. $Ke(12) = \{5, 6, 8, 10\}$.
- $Ke(13) = \{9, 10, 11\}$.

Hãy thực hiện:

- Tìm $BFS(1)=?$
- Tìm $BFS(5)=?$
- Tìm $DFS(1)=?$
- Tìm $DFS(5)=?$
- Tìm đường đi từ 1 đến 13 bằng thuật toán BFS?
- Tìm đường đi từ 1 đến 13 bằng thuật toán DFS?

Thank You !

31