Học viện Công nghệ Bưu chính Viễn thông
Khoa Công nghệ thông tin 1

Toán rời rạc 2

Tìm kiếm trên đồ thị

Bộ môn KHMT, Khoa CNTT1

Nội dung

- Thuật toán tìm kiếm theo chiều sâu (Depth-First Search) - DFS)
- Thuật toán tìm kiếm theo chiều rộng (Breadth-First Search - BFS)
- Một số ứng dụng của DFS và BFS

Tìm kiếm theo chiều sâu - DFS

- Tư tưởng
  - Trong quá trình tìm kiếm, ưu tiên "chiều sâu" hơn "chiều rộng"
  - Đi xuống sâu nhất có thể trước khi quay lại

- Thuật toán

```
DFS(u){ //u là đỉnh bắt đầu duyệt
        <Thăm đỉnh u>; //duyệt đỉnh u
        chuaxet[u] = false;  //xác nhận đỉnh u đã duyệt
        for(v ∈ Ke(u)){
                if( chuaxet[v])      //nếu v chưa được duyệt
                        DFS(v);  //duyệt theo chiều sâu từ v
        }
}
```

DFS sử dụng ngăn xếp

```
DFS(u){
        Bước 1: Khởi tạo
        stack = ∅; //khởi tạo stack là ∅
        push(stack, u);  //đưa đỉnh u vào ngăn xếp
        <Thăm đỉnh u>; //duyệt đỉnh u
        chuaxet[u] = false; //xác nhận đỉnh u đã duyệt
        Bước 2: Lặp
        while(stack ≠ ∅){
                s = pop(stack);  //lấy đỉnh ở đầu ngăn xếp
                for(t ∈ Ke(s)){
                        if( chuaxet[t]){  //nếu t chưa được duyệt
                                <Thăm đỉnh t>; //duyệt đỉnh t
                                chuaxet[t] = false; //t đã duyệt
                                push(stack, s); //đưa s vào ngăn xếp
                                push(stack, t); //đưa t vào ngăn xếp
                                break;  //chỉ lấy một đỉnh t
                        }
                }
        }
        Bước 3: Trả lại kết quả
        return <tập đỉnh đã duyệt>;
}
```

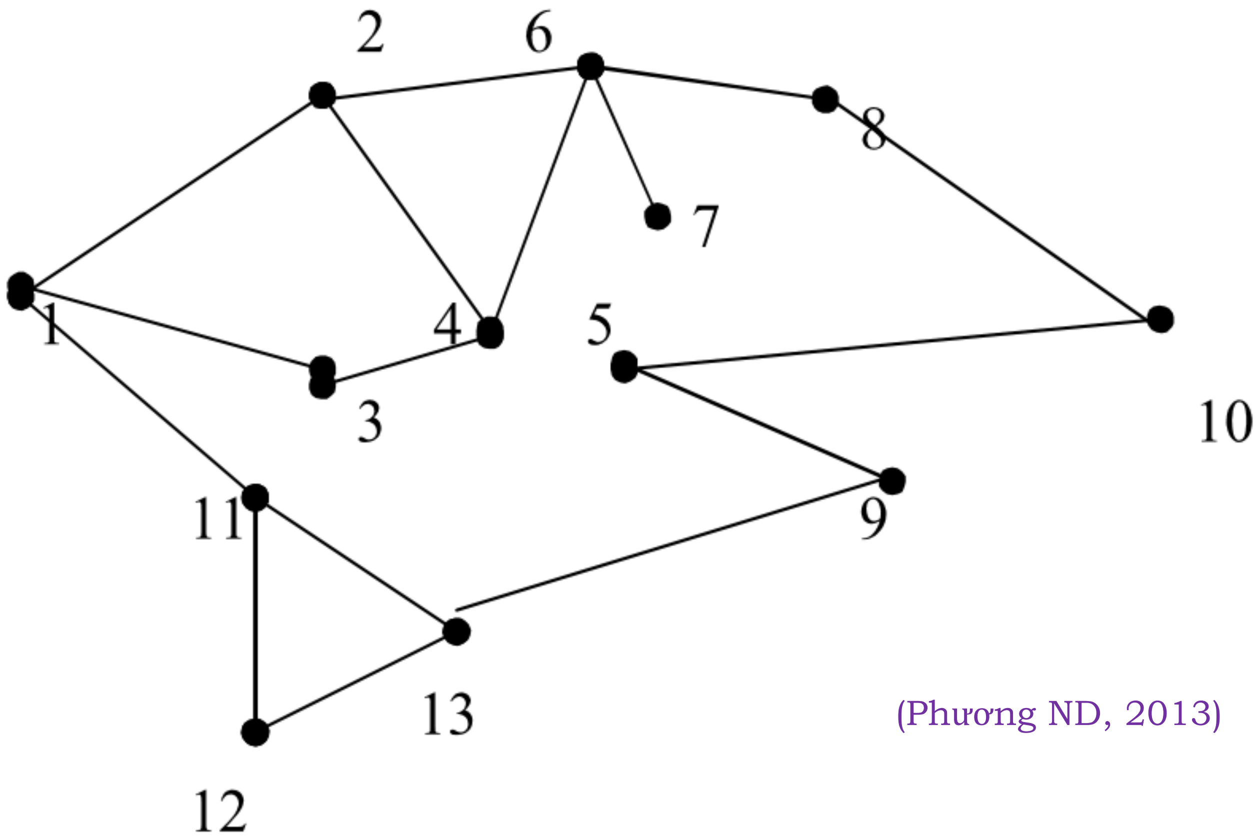
Độ phức tạp thuật toán DFS

- Độ phức tạp thuật toán DFS(u) phụ thuộc vào phương pháp biểu diễn đồ thị

- Biểu diễn đồ thị bằng ma trận kề
  - Độ phức tạp thuật toán là $O(n^2)$, n là số đỉnh
- Biểu diễn đồ thị bằng danh sách cạnh
  - Độ phức tạp thuật toán là $O(n.m)$, n là số đỉnh, m là số cạnh
- Biểu diễn đồ thị bằng danh sách kề
  - Độ phức tạp thuật toán là $O(\max(n, m))$, n là số đỉnh, m là số cạnh

Kiểm nghiệm thuật toán DFS (1/2)

- **Ví dụ 1**: Cho đồ thị gồm 13 đỉnh như hình vẽ. Hãy kiểm nghiệm thuật toán DFS(1).



(Phương ND, 2013)

Kiểm nghiệm thuật toán DFS (2/2)

STT	Trạng thái ngăn xếp	Danh sách đỉnh được duyệt
1	1	1
2	1, 2	1, 2
3	1, 2, 4	1, 2, 4
4	1, 2, 4, 3	1, 2, 4, 3
5	1, 2, 4	1, 2, 4, 3
6	1, 2, 4, 6	1, 2, 4, 3, 6
7	1, 2, 4, 6, 7	1, 2, 4, 3, 6, 7
8	1, 2, 4, 6	1, 2, 4, 3, 6, 7
9	1, 2, 4, 6, 8	1, 2, 4, 3, 6, 7, 8
10	1, 2, 4, 6, 8, 10	1, 2, 4, 3, 6, 7, 8, 10
11	1, 2, 4, 6, 8, 10, 5	1, 2, 4, 3, 6, 7, 8, 10, 5
12	1, 2, 4, 6, 8, 10, 5, 9	1, 2, 4, 3, 6, 7, 8, 10, 5, 9
13	1, 2, 4, 6, 8, 10, 5, 9, 13	1, 2, 4, 3, 6, 7, 8, 10, 5, 9, 13
14	1, 2, 4, 6, 8, 10, 5, 9, 13, 11	1, 2, 4, 3, 6, 7, 8, 10, 5, 9, 13, 11
15	1, 2, 4, 6, 8, 10, 5, 9, 13, 11, 12	1, 2, 4, 3, 6, 7, 8, 10, 5, 9, 13, 11, 12
16-	Lần lượt bỏ các đỉnh ra khỏi ngăn xếp	

Kết quả duyệt: 1, 2, 4, 3, 6, 7, 8, 10, 5, 9, 13, 11, 12

Bài tập 1

- Cho đồ thị gồm 13 đỉnh được biểu diễn dưới dạng ma trận kề như hình vẽ. Hãy cho biết kết quả thực hiện thuật toán DFS(1). Chỉ rõ trạng thái của ngăn xếp và tập đỉnh được duyệt theo mỗi bước thực hiện của thuật toán.

$$\begin{vmatrix}
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0
\end{vmatrix}$$

(Phương ND, 2013)

Nội dung

- Thuật toán tìm kiếm theo chiều sâu (Depth-First Search) - DFS)
- Thuật toán tìm kiếm theo chiều rộng (Breadth-First Search - BFS)
- Một số ứng dụng của DFS và BFS

Tìm kiếm theo chiều rộng - BFS

- Tư tưởng
  - Trong quá trình tìm kiếm, ưu tiên "chiều rộng" hơn "chiều sâu"
  - Tìm kiếm xung quanh trước khi đi xuống sâu hơn
- Thuật toán

BFS(u){

Bước 1: Khởi tạo

$queue = \emptyset$; $push(queue, u)$; $chuaxet[u] = false$;

Bước 2: Lặp

while($queue \neq \emptyset$){

$s = pop(queue)$; <Thăm đỉnh s>;

for($t \in Ke(s)$){

if($chuaxet[t]$){

$push(queue, t)$; $chuaxet[t] = false$;

}

}

}

Bước 3: Trả lại kết quả

return <tập đỉnh đã duyệt>;

}

Độ phức tạp thuật toán BFS

- Độ phức tạp thuật toán BFS(u) phụ thuộc vào phương pháp biểu diễn đồ thị

- Biểu diễn đồ thị bằng ma trận kề
  - Độ phức tạp thuật toán là $O(n^2)$, n là số đỉnh
- Biểu diễn đồ thị bằng danh sách cạnh
  - Độ phức tạp thuật toán là $O(n.m)$, n là số đỉnh, m là số cạnh
- Biểu diễn đồ thị bằng danh sách kề
  - Độ phức tạp thuật toán là $O(\max(n, m))$, n là số đỉnh, m là số cạnh

Kiểm nghiệm thuật toán BFS (1/2)

- Cho đồ thị gồm 13 đỉnh được biểu diễn dưới dạng ma trận kề như hình vẽ. Hãy cho biết kết quả thực hiện thuật toán BFS(1). Chỉ rõ trạng thái của hàng đợi và tập đỉnh được duyệt theo mỗi bước thực hiện của thuật toán.

$$\begin{vmatrix}
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0
\end{vmatrix}$$

(Phương ND, 2013)

Kiểm nghiệm thuật toán BFS (2/2)

STT	Trạng thái hàng đợi	Danh sách đỉnh được duyệt
1	1	∅
2	2, 3, 4	1
3	3, 4, 6	1, 2
4	4, 6, 5	1, 2, 3
5	6, 5, 7	1, 2, 3, 4
6	5, 7, 12	1, 2, 3, 4, 6
7	7, 12, 8	1, 2, 3, 4, 6, 5
8	12, 8	1, 2, 3, 4, 6, 5, 7
9	8, 10	1, 2, 3, 4, 6, 5, 7, 12
10	10	1, 2, 3, 4, 6, 5, 7, 12, 8
11	9, 11, 13	1, 2, 3, 4, 6, 5, 7, 12, 8, 10
12	11, 13	1, 2, 3, 4, 6, 5, 7, 12, 8, 10, 9
13	13	1, 2, 3, 4, 6, 5, 7, 12, 8, 10, 9, 11
14	∅	1, 2, 3, 4, 6, 5, 7, 12, 8, 10, 9, 11, 13

Kết quả duyệt: 1, 2, 3, 4, 6, 5, 7, 12, 8, 10, 9, 11, 13

Chú ý

- Với đồ thị vô hướng
  - Nếu $DFS(u) = V$ hoặc $BFS(u) = V$, ta có thể kết luận đồ thị liên thông
- Với đồ thị có hướng
  - Nếu $DFS(u) = V$ hoặc $BFS(u) = V$, ta có thể kết luận đồ thị liên thông yếu

Nội dung

- Thuật toán tìm kiếm theo chiều sâu (Depth-First Search) - DFS)
- Thuật toán tìm kiếm theo chiều rộng (Breadth-First Search - BFS)
- Một số ứng dụng của DFS và BFS

Xác định thành phần liên thông của đồ thị

- Phát biểu bài toán
  - Cho đồ thị vô hướng $G =< V, E >$, trong đó V là tập đỉnh, E là tập cạnh. Xác định các thành phần liên thông của G?
- Thuật toán

```
Duyet-TPLT(){ //duyệt thành phần liên thông
        Bước 1: Khởi tạo
        soTPLT = 0; //khởi tạo số thành phần liên thông bằng 0
        Bước 2: Lặp
        for(u ∈ V){ //lặp trên tập đỉnh
                if( chuaxet[u]){
                        soTPLT = soTPLT + 1;//ghi nhận số TPLT
                        BFS(u); // có thể gọi DFS(u)
                        <Ghi nhận các đỉnh thuộc TPLT>;
                }
        }
        Bước 3: Trả lại kết quả
        return <các TPLT>;
}
```

Bài tập 2

- Cho đồ thị vô hướng được biểu diễn dưới dạng ma trận kề như hình vẽ. Xác định các thành phần liên thông của đồ thị?

$$\begin{vmatrix}
0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0
\end{vmatrix}$$

(Phương ND, 2013)

Tìm đường đi giữa các đỉnh trên đồ thị (1/4)

- Phát biểu bài toán
  - Cho đồ thị $G =< V, E >$ (vô hướng hoặc có hướng), trong đó V là tập đỉnh, E là tập cạnh. Hãy tìm đường đi từ $s \in V$ đến $t \in V$?

- Mô tả thuật toán
  - Nếu $t \in DFS(s)$ hoặc $t \in BFS(s)$ thì ta có thể kết luận có đường đi từ s đến t trên đồ thị, ngược lại sẽ không có đường đi
  - Để ghi nhận đường đi ta sử dụng mảng $truoc[]$ gồm n phần tử $(n = |V|)$
    - Khởi tạo ban đầu $truoc[u] = 0$ với mọi u
    - Mỗi khi đưa $v \in Ke(u)$ vào ngăn xếp (nếu sử dụng DFS) hoặc hàng đợi (nếu sử dụng BFS) ta ghi nhận $truoc[v] = u$
    - Nếu DFS và BFS không duyệt được đến đỉnh t, khi đó $truoc[t] = 0$ thì ta kết luận không có đường đi từ s đến t

Tìm đường đi giữa các đỉnh trên đồ thị (2/4)

Sử dụng thuật toán DFS

$\boldsymbol{DFS}(s)\{$

Bước 1: Khởi tạo

$stack = \emptyset$; $push(stack, s)$; $chuaxet[s] = false$;

Bước 2: Lặp

while$(stack \neq \emptyset)\{$

$u = pop(stack)$; //lấy đỉnh ở ngăn xếp

for$(v \in Ke(u))\{$

if$(\ chuaxet[v])\{$ //nếu v chưa được duyệt

$chuaxet[v] = false$; //v đã duyệt

$push(stack, u)$; //đưa u vào ngăn xếp

$push(stack, v)$; //đưa v vào ngăn xếp

$\boldsymbol{truoc[v] = u}$; //Ghi nhận $truoc[v]$ là u

break; //chỉ lấy một đỉnh

}

}

}

Bước 3: Trả lại kết quả

return <tập đỉnh đã duyệt>;

}

Tìm đường đi giữa các đỉnh trên đồ thị (3/4)

Sử dụng thuật toán BFS

BFS(s){

Bước 1: Khởi tạo

$queue = \emptyset$; $push(queue, s)$; $chuaxet[s] = false$;

Bước 2: Lặp

while($queue \neq \emptyset$){

$u = \text{pop}(queue)$;

for($v \in Ke(u)$){

if($chuaxet[v]$){

$push(queue, v)$;

$chuaxet[v] = false$;

$\boldsymbol{truoc[v] = u}$;

}

}

}

Bước 3: Trả lại kết quả

return <tập đỉnh đã duyệt>;

}

Tìm đường đi giữa các đỉnh trên đồ thị (4/4)

Ghi nhận đường đi

```
Ghi-Nhan-Duong-Di(s, t){
        if( truoc[t] == 0){
                <Không có đường đi từ s tới t>;
        }
        else{
                <Đưa ra đỉnh t>; // Đưa ra đỉnh t trước
                u = truoc[t]; // u là đỉnh trước khi đến được t
                while(u ≠ s){
                        <Đưa ra đỉnh u>;
                        u = truoc[u]; // lần ngược lại đỉnh trước u
                }
                <Đưa ra đỉnh s>;
        }
}
```

Bài tập 3

- Cho đồ thị gồm 13 đỉnh được biểu diễn dưới dạng ma trận kề như hình vẽ. Tìm **đường đi từ đỉnh 1 đến đỉnh 13** của đồ thị ?

$$
\begin{vmatrix}
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0
\end{vmatrix}
$$

(Phương ND, 2013)

Tính liên thông mạnh trên đồ thị có hướng

- Phát biểu bài toán
  - Đồ thị có hướng $G =< V, E >$ là liên thông mạnh nếu giữa hai đỉnh bất kỳ của nó đều tồn tại đường đi. Cho trước đồ thị có hướng $G \ =< V, E >$. Kiểm tra xem G có liên thông mạnh hay không?
- Thuật toán

bool Strong_Connected $(G =< V, E >)\{$ //kt tính liên thông mạnh của G
$\quad$ $ReInit(\,)$; // $\forall u \in V$: $chuaxet[u] = true$;
$\quad$ **for**$(u \in V)\{$ //lặp trên tập đỉnh
$\quad\quad$ **if**$(\boldsymbol{BFS}(\boldsymbol{u}) \neq V)$ // có thể kiểm tra $\boldsymbol{DFS}(\boldsymbol{u}) \neq V$
$\quad\quad\quad$ **return** false; // đồ thị không liên thông mạnh
$\quad\quad$ **else**
$\quad\quad\quad$ $ReInit()$; // khởi tạo lại mảng $chuaxet[]$
$\quad$ $\}$
$\quad$ **return** true; // đồ thị liên thông mạnh
$\}$

Bài tập 4

- Cho đồ thị có hướng $G = < V, E >$ được biểu diễn dưới dạng ma trận kề như hình bên. Xác định xem G có liên thông mạnh hay không?

$$
\begin{vmatrix}
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0
\end{vmatrix}
$$

(Phương ND, 2013)

Duyệt các đỉnh trụ

- Phát biểu bài toán
  - Đỉnh $u \in V$ của đồ thị vô hướng $G =< V, E >$ được gọi là trụ nếu loại bỏ đỉnh u cùng với các cạnh nối với u làm tăng thành phần liên thông của G. Cho trước đồ thị vô hướng (liên thông) $G =< V, E >$, tìm các đỉnh trụ của G?
- Thuật toán

Duyet_Tru $(G =< V, E >)$\{
    $ReInit(\,)$; // $\forall u \in V$: $chuaxet[u] = true$;
    for$(u \in V)$\{ //lấy mỗi đỉnh u
        $chuaxet[u] = false$; // cấm BFS hoặc DFS duyệt u
        if$(\boldsymbol{BFS}(\boldsymbol{v}) \neq V\backslash\{u\})$ // có thể kiểm tra $\boldsymbol{DFS}(\boldsymbol{v}) \neq V\backslash\{u\}$
            <Ghi nhận u là trụ>;
        $ReInit()$; // khởi tạo lại mảng $chuaxet[]$
    \}
\}

Bài tập 5

- Cho đồ thị vô hướng $G = < V, E >$ được biểu diễn dưới dạng ma trận kề như hình bên. Tìm các đỉnh trụ của G?

$$
\begin{vmatrix}
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0
\end{vmatrix}
$$

(Phương ND, 2013)

Duyệt các cạnh cầu

- Phát biểu bài toán
  - Cạnh $e \in E$ của đồ thị vô hướng $G =< V, E >$ được gọi là cạnh cầu nếu loại bỏ e làm tăng thành phần liên thông của G. Cho trước đồ thị vô hướng (liên thông) $G =< V, E >$, tìm các cạnh cầu của G?
- Thuật toán

Duyet_Cau $(G =< V, E >)$\{
  $ReInit()$; // $\forall u \in V$: $chuaxet[u] = true$;
  for$(e \in E)$\{ //lấy mỗi cạnh của đồ thị
    $E = E \setminus \{e\}$; // loại bỏ cạnh e ra khỏi đồ thị
    if$(\boldsymbol{BFS}(\mathbf{1}) \neq V)$ // có thể kiểm tra $\boldsymbol{DFS}(\mathbf{1}) \neq V$
      <Ghi nhận e là cầu>;
    $E = E \cup \{e\}$; // hoàn trả lại cạnh e
    $ReInit()$; // khởi tạo lại mảng $chuaxet[]$
  }
}

Bài tập 6

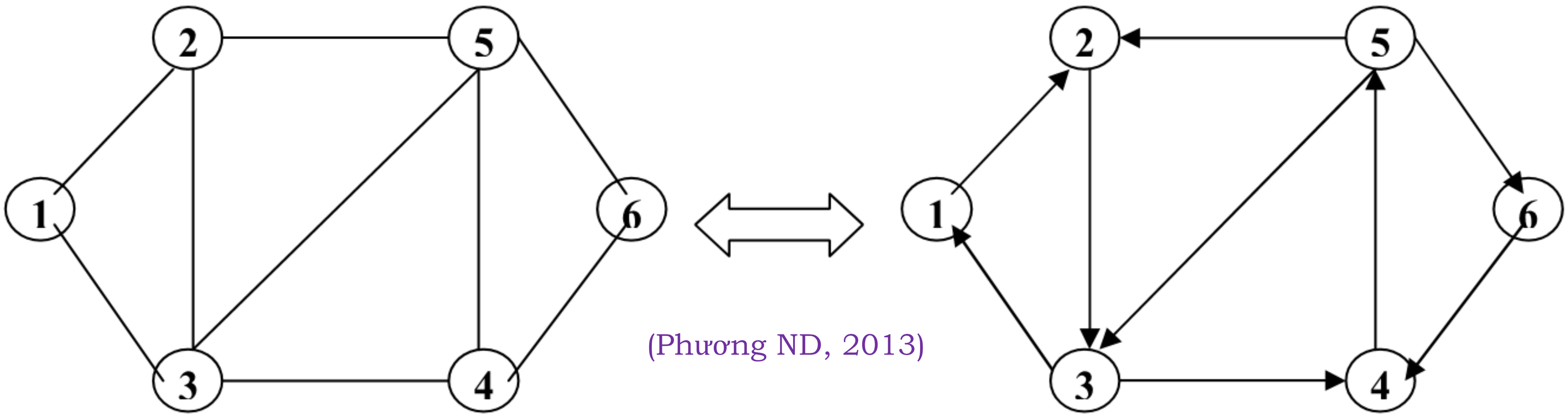
- Cho đồ thị vô hướng $G = < V, E >$ được biểu diễn dưới dạng ma trận kề như hình bên. Tìm các cạnh cầu của G?

$$\begin{vmatrix}
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0
\end{vmatrix}$$

(Phương ND, 2013)

Bài toán định chiều đồ thị (1/2)

- Định nghĩa
  - Phép định chiều đồ thị vô hướng liên thông là phép biến đổi đồ thị vô hướng liên thông thành đồ thị có hướng liên thông mạnh.
  - Đồ thị vô hướng $G = < V, E >$ được gọi là đồ thị định chiều được nếu có thể dịch chuyển được thành đồ thị có hướng liên thông mạnh bằng cách định chiều mỗi cạnh vô hướng thành một cung có hướng.
- Ví dụ



(Phương ND, 2013)

Bài toán định chiều đồ thị (2/2)

- Định lý
  - Đồ thị vô hướng liên thông $G =< V, E >$ định chiều được khi và chỉ khi tất cả các cạnh $e \in E$ của G đều không phải là cầu.

- Một số vấn đề cần quan tâm
  - Chứng minh một đồ thị vô hướng là định chiều được
  - Viết chương trình kiểm tra một đồ thị vô hướng có định chiều được hay không?
  - Chỉ ra một phép định chiều trên một đồ thị vô hướng

Tóm tắt

- Thuật toán duyệt theo chiều sâu bắt đầu tại đỉnh $u \in V$, $DFS(u)$
- Thuật toán duyệt theo chiều rộng bắt đầu tại đỉnh $u \in V$, $BFS(u)$
- Ứng dụng các thuật toán $DFS(u)$ và $BFS(u)$
  - 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
  - Kiểm tra tính liên thông mạnh của đồ thị
  - Duyệt các đỉnh trụ của đồ thị
  - Duyệt các cạnh cầu của đồ thị
  - Kiểm tra một đồ thị có định chiều được hay không

Bài tập

- Làm một số bài tập trong giáo trình