Graph Connectivity
3rd Recitation for CS1501

Huichao Xue

Department of Computer Science
Universities of Pittsburgh

Algorithm Implementation, Spring, 2009
Outline

1. Basic Concepts
   - Connectivity
   - More definitions about connectivity
   - An example

2. Detecting articulation nodes
   - Graph Representation
   - DFS with back-edges
   - “min” values
Outline

1. **Basic Concepts**
   - Connectivity
     - More definitions about connectivity
     - An example

2. **Detecting articulation nodes**
   - Graph Representation
   - DFS with back-edges
   - “min” values
Connectivity

Definition

A graph is **connected** if and only if for every pair of vertices $u$ and $v$, there is a path from $u$ to $v$. 
How do we determine whether or not a graph is connected?
An algorithm

Count the number of calls made to visit() from the search() method described in lecture:

```c
count = 0;
void search(){
    int k;
    //initialize array val[]
    for(k = 1; k <= V; k++) val[k] = unseen;
    for(k = 1; k <= V; k++){
        if(val[k] == unseen){
            count++;
            //added step
            visit(k);
        }
    }
}
```
1 Basic Concepts
- Connectivity
- More definitions about connectivity
- An example

2 Detecting articulation nodes
- Graph Representation
- DFS with back-edges
- “min” values
Graph’s “robustness”

Definition
A graph is biconnected if and only if there are at least two different paths connecting each pair of vertices.

Definition
An articulation point in a connected graph is a vertex that if deleted would break the graph into two or more pieces.
Graph’s “robustness”

**Definition**

A graph is biconnected if and only if there are at least two different paths connecting each pair of vertices.

**Definition**

An articulation point in a connected graph is a vertex that if deleted would break the graph into two or more pieces.
Outline

1. Basic Concepts
   - Connectivity
   - More definitions about connectivity
   - An example

2. Detecting articulation nodes
   - Graph Representation
   - DFS with back-edges
   - “min” values
What are the articulation points in this graph?

An example

Graph Connectivity

Huichao Xue
What are the articulation points in this graph?

An example
Outline

1. Basic Concepts
   - Connectivity
   - More definitions about connectivity
   - An example

2. Detecting articulation nodes
   - Graph Representation
   - DFS with back-edges
   - “min” values
### Adjacent List Representation

<table>
<thead>
<tr>
<th>Node</th>
<th>Neighbor list</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B → E → F</td>
</tr>
<tr>
<td>B</td>
<td>A → F</td>
</tr>
<tr>
<td>C</td>
<td>D → F</td>
</tr>
<tr>
<td>D</td>
<td>C → F → I</td>
</tr>
<tr>
<td>E</td>
<td>A → G → H</td>
</tr>
<tr>
<td>F</td>
<td>A → B → C → D</td>
</tr>
<tr>
<td>G</td>
<td>E → H</td>
</tr>
<tr>
<td>H</td>
<td>E → G</td>
</tr>
<tr>
<td>I</td>
<td>D</td>
</tr>
</tbody>
</table>

![Graph Diagram](image-url)
Outline

1. Basic Concepts
   - Connectivity
   - More definitions about connectivity
   - An example

2. Detecting articulation nodes
   - Graph Representation
   - DFS with back-edges
   - “min” values
Basic Concepts
Detecting articulation nodes

DFS

Graph Representation
DFS with back-edges
“min” values

Huichao Xue
Graph Connectivity
Basic Concepts
Detecting articulation nodes

DFS

Graph Representation
DFS with back-edges
“min” values

Huichao Xue
Graph Connectivity
Basic Concepts
Detecting articulation nodes

Graph Representation
DFS with back-edges
“min” values

DFS

<table>
<thead>
<tr>
<th>Node</th>
<th>Neighbor list</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B → E → F</td>
</tr>
<tr>
<td>B</td>
<td>A → F</td>
</tr>
<tr>
<td>C</td>
<td>D → F</td>
</tr>
<tr>
<td>D</td>
<td>C → F → I</td>
</tr>
<tr>
<td>E</td>
<td>A → G → H</td>
</tr>
<tr>
<td>F</td>
<td>A → B → C → D</td>
</tr>
<tr>
<td>G</td>
<td>E → H</td>
</tr>
<tr>
<td>H</td>
<td>E → G</td>
</tr>
<tr>
<td>I</td>
<td>D</td>
</tr>
</tbody>
</table>
DFS

**Basic Concepts**
- Detecting articulation nodes

**Graph Representation**
- DFS with back-edges
- “min” values

### DFS

**Diagram**

- **Node and Neighbor List**

<table>
<thead>
<tr>
<th>Node</th>
<th>Neighbor list</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B → E → F</td>
</tr>
<tr>
<td>B</td>
<td>A → F</td>
</tr>
<tr>
<td>C</td>
<td>D → F</td>
</tr>
<tr>
<td>D</td>
<td>C → F → I</td>
</tr>
<tr>
<td>E</td>
<td>A → G → H</td>
</tr>
<tr>
<td>F</td>
<td>A → B → C → D</td>
</tr>
<tr>
<td>G</td>
<td>E → H</td>
</tr>
<tr>
<td>H</td>
<td>E → G</td>
</tr>
<tr>
<td>I</td>
<td>D</td>
</tr>
</tbody>
</table>
Detecting articulation nodes

DFS

Graph Connectivity

<table>
<thead>
<tr>
<th>Node</th>
<th>Neighbor list</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B → E → F</td>
</tr>
<tr>
<td>B</td>
<td>A → F</td>
</tr>
<tr>
<td>C</td>
<td>D → F</td>
</tr>
<tr>
<td>D</td>
<td>C → F → I</td>
</tr>
<tr>
<td>E</td>
<td>A → G → H</td>
</tr>
<tr>
<td>F</td>
<td>A → B → C → D</td>
</tr>
<tr>
<td>G</td>
<td>E → H</td>
</tr>
<tr>
<td>H</td>
<td>E → G</td>
</tr>
<tr>
<td>I</td>
<td>D</td>
</tr>
</tbody>
</table>
Basic Concepts
Detecting articulation nodes

Graph Representation
DFS with back-edges
“min” values

DFS

Node | Neighbor list
---|---
A | B → E → F
B | A → F
C | D → F
D | C → F → I
E | A → G → H
F | A → B → C → D
G | E → H
H | E → G
I | D

Huichao Xue
Graph Connectivity
Basic Concepts
Detecting articulation nodes

DFS

Graph Representation
DFS with back-edges
“min” values

Huichao Xue
Graph Connectivity
Basic Concepts
Detecting articulation nodes

Graph Representation
DFS with back-edges
“min” values

DFS

Node | Neighbor list
--- | ---
A | B → E → F
B | A → F
C | D → F
D | C → F → I
E | A → G → H
F | A → B → C → D
G | E → H
H | E → G
I | D
Basic Concepts
Detecting articulation nodes

Graph Representation
DFS with back-edges
“min” values

DFS

Node | Neighbor list
--- | ---
A | B → E → F
B | A → F
C | D → F
D | C → F → I
E | A → G → H
F | A → B → C → D
G | E → H
H | E → G
I | D

Huichao Xue  Graph Connectivity
Basic Concepts
Detecting articulation nodes

Graph Representation
DFS with back-edges
“min” values

DFS

Node | Neighbor list
--- | ---
A | B → E → F
B | A → F
C | D → F
D | C → F → I
E | A → G → H
F | A → B → C → D
G | E → H
H | E → G
I | D
**DFS**

- **A** (1) → **B** (2) → **E** (7) → **F** (3)
- **C** (4) → **D** (5) → **I** (6)
- **F** (3) → **G** (8)
- **H** (9)

**Neighbor list**

<table>
<thead>
<tr>
<th>Node</th>
<th>Neighbor list</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B → E → F</td>
</tr>
<tr>
<td>B</td>
<td>A → F</td>
</tr>
<tr>
<td>C</td>
<td>D → F</td>
</tr>
<tr>
<td>D</td>
<td>C → F → I</td>
</tr>
<tr>
<td>E</td>
<td>A → G → H</td>
</tr>
<tr>
<td>F</td>
<td>A → B → C → D</td>
</tr>
<tr>
<td>G</td>
<td>E → H</td>
</tr>
<tr>
<td>H</td>
<td>E → G</td>
</tr>
<tr>
<td>I</td>
<td>D</td>
</tr>
</tbody>
</table>
Basic Concepts
Detecting articulation nodes

DFS

Node | Neighbor list
---|---
A | B → E → F
B | A → F
C | D → F
D | C → F → I
E | A → G → H
F | A → B → C → D
G | E → H
H | E → G
I | D

Graph Connectivity
A vertex $x$ is **NOT** an articulation point if every child $y$ of $x$ has some node lower in the tree connected to a node higher in the tree than $x$.

**Example**

- $B(2)$ is not an articulation point because child $F(3)$ connected to $A(1)$ in original graph.
- $C(4)$ is not an articulation point because child $D(5)$ connected to $F(3)$ in original graph.
A property of non-articulation points

A vertex $x$ is **NOT** an articulation point if every child $y$ of $x$ has some node lower in the tree connected to a node higher in the tree than $x$.

**Example**

- $B(2)$ is not an articulation point because child $F(3)$ connected to $A(1)$ in original graph.
- $C(4)$ is not an articulation point because child $D(5)$ connected to $F(3)$ in original graph.
Outline

1. Basic Concepts
   - Connectivity
   - More definitions about connectivity
   - An example

2. Detecting articulation nodes
   - Graph Representation
   - DFS with back-edges
   - “min” values
The algorithm calculating “min” values

```java
int visit(int k) { //DFS, adjacency list
    struct node *t;
    int m, min;

    val[k] = ++id;
    for(t = adj[k]; t != z; t = t->next){
        if(val[t->v] == unseen){
            m = visit(t->v);
            if(m<min) min = m;
            // Displays D, F, A, and E in this order
            if(m>=val[k]) System.out.println(name(k));
        }
        else{
            if(val[t->v] < min) min = val[t->v];
        }
    }
    return min;
}
```
The articulation points are those vertices whose DFS numbers are less or equal to the min values of ANY of their descendents in the DFS tree.

Example

- **E** is an articulation point since its number \(7 \leq \min(G) \leq \min(H) = 7\).
The articulation points are those vertices whose DFS numbers are less or equal to the min values of ANY of their descendents in the DFS tree.

Example

- $F$ is an articulation point since its number $3 \leq \min(C) = \min(D) = \min(I) = 5$. 
The articulation points are those vertices whose DFS numbers are less or equal to the min values of ANY of their descendents in the DFS tree.

Example

- D is an articulation point since its number $5 \leq \min(I) = 5$. 
The articulation points are those vertices whose DFS numbers are less or equal to the min values of ANY of their descendents in the DFS tree.

Example
- A is the root and is an articulation point. Because it has branches.