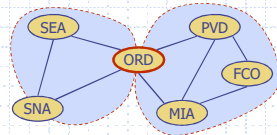


Presentation for use with the textbook, *Algorithm Design and Applications*, by M. T. Goodrich and R. Tamassia, Wiley, 2015

## Biconnected Components



© 2015 Goodrich and Tamassia

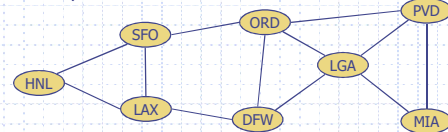
Biconnectivity

1

## Biconnected Graph

- ◆ Equivalent definitions of a biconnected graph  $G$ 
  - Graph  $G$  has no separation edges and no separation vertices
  - For any two vertices  $u$  and  $v$  of  $G$ , there are two disjoint simple paths between  $u$  and  $v$  (i.e., two simple paths between  $u$  and  $v$  that share no other vertices or edges)
  - For any two vertices  $u$  and  $v$  of  $G$ , there is a simple cycle containing  $u$  and  $v$

### ◆ Example



© 2015 Goodrich and Tamassia

Biconnectivity

4

## Application: Networking

- ◆ A computer network can be modeled as a graph, where vertices are routers and edges are network connections between edges.
- ◆ A router can be considered **critical** if it can disconnect the network for that router to fail.
- ◆ It would be nice to identify which routers are critical.
- ◆ We can do such an identification by solving the biconnected components problem.

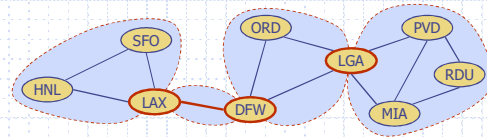
© 2015 Goodrich and Tamassia

Biconnectivity

2

## Biconnected Components

- ◆ Biconnected component of a graph  $G$ 
  - A maximal biconnected subgraph of  $G$ , or
  - A subgraph consisting of a separation edge of  $G$  and its end vertices
- ◆ Interaction of biconnected components
  - An edge belongs to exactly one biconnected component
  - A nonseparation vertex belongs to exactly one biconnected component
  - A separation vertex belongs to two or more biconnected components
- ◆ Example of a graph with four biconnected components



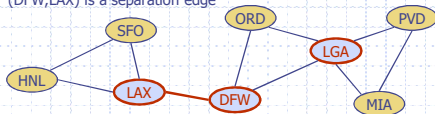
© 2015 Goodrich and Tamassia

Biconnectivity

5

## Separation Edges and Vertices

- ◆ Definitions
  - Let  $G$  be a connected graph
  - A separation edge of  $G$  is an edge whose removal disconnects  $G$
  - A separation vertex of  $G$  is a vertex whose removal disconnects  $G$
- ◆ Applications
  - Separation edges and vertices represent single points of failure in a network and are critical to the operation of the network
- ◆ Example
  - DFW, LGA and LAX are separation vertices
  - (DFW, LAX) is a separation edge



© 2015 Goodrich and Tamassia

Biconnectivity

3

## Equivalence Classes

- ◆ Given a set  $S$ , a relation  $R$  on  $S$  is a set of ordered pairs of elements of  $S$ , i.e.,  $R$  is a subset of  $S \times S$
- ◆ An equivalence relation  $R$  on  $S$  satisfies the following properties
  - Reflexive:  $(x, x) \in R$
  - Symmetric:  $(x, y) \in R \Rightarrow (y, x) \in R$
  - Transitive:  $(x, y) \in R \wedge (y, z) \in R \Rightarrow (x, z) \in R$
- ◆ An equivalence relation  $R$  on  $S$  induces a partition of the elements of  $S$  into equivalence classes
- ◆ Example (connectivity relation among the vertices of a graph):
  - Let  $V$  be the set of vertices of a graph  $G$
  - Define the relation  $C = \{(v, w) \in V \times V \text{ such that } G \text{ has a path from } v \text{ to } w\}$
  - Relation  $C$  is an equivalence relation
  - The equivalence classes of relation  $C$  are the vertices in each connected component of graph  $G$

© 2015 Goodrich and Tamassia

Biconnectivity

6

## Link Relation

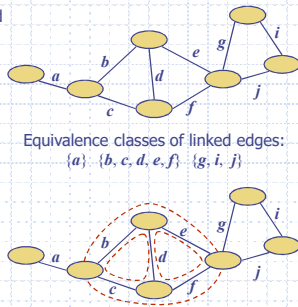
- Edges  $e$  and  $f$  of connected graph  $G$  are linked if
  - $e = f$ , or
  - $G$  has a simple cycle containing  $e$  and  $f$

### Theorem:

The link relation on the edges of a graph is an equivalence relation

### Proof Sketch:

- The reflexive and symmetric properties follow from the definition
- For the transitive property, consider two simple cycles sharing an edge



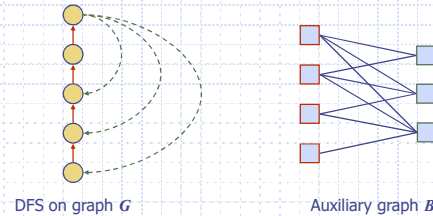
© 2015 Goodrich and Tamassia

Biconnectivity

7

## Auxiliary Graph (cont.)

- In the worst case, the number of edges of the auxiliary graph is proportional to  $nm$



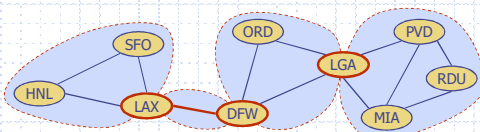
© 2015 Goodrich and Tamassia

Biconnectivity

10

## Link Components

- The link components of a connected graph  $G$  are the equivalence classes of edges with respect to the link relation
- A biconnected component of  $G$  is the subgraph of  $G$  induced by an equivalence class of linked edges
- A separation edge is a single-element equivalence class of linked edges
- A separation vertex has incident edges in at least two distinct equivalence classes of linked edge



© 2015 Goodrich and Tamassia

Biconnectivity

8

## An $O(nm)$ -Time Algorithm

- Lemma: The connected components of the auxiliary graph  $B$  correspond to the link components of the graph  $G$  that induced  $B$ .
- This lemma yields the following  $O(nm)$ -time algorithm for computing all the link components of a graph  $G$  with  $n$  vertices and  $m$  edges:

1. Perform a DFS traversal  $T$  on  $G$ .
2. Compute the auxiliary graph  $B$  by identifying the cycles of  $G$  induced by each back edge with respect to  $T$ .
3. Compute the connected components of  $B$ , for example, by performing a DFS traversal of the auxiliary graph  $B$ .
4. For each connected component of  $B$ , output the vertices of  $B$  (which are edges of  $G$ ) as a link component of  $G$ .

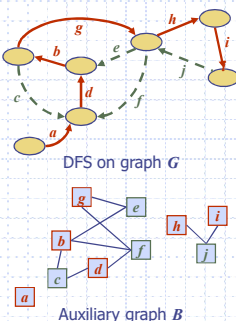
© 2015 Goodrich and Tamassia

Biconnectivity

11

## Auxiliary Graph

- Auxiliary graph  $B$  for a connected graph  $G$ 
  - Associated with a DFS traversal of  $G$
  - The vertices of  $B$  are the edges of  $G$
- For each back edge  $e$  of  $G$ ,  $B$  has edges  $(ef_1), (ef_2), \dots, (ef_k)$ , where  $f_1, f_2, \dots, f_k$  are the discovery edges of  $G$  that form a simple cycle with  $e$
- Its connected components correspond to the link components of  $G$



© 2015 Goodrich and Tamassia

Biconnectivity

9

## A Linear-Time Algorithm

### Algorithm LinkComponents( $G$ ):

**Input:** A connected graph  $G$

**Output:** The link components of  $G$

Let  $F$  be an initially empty auxiliary graph.

Perform a DFS traversal of  $G$  starting at an arbitrary vertex  $s$ .

Add each DFS discovery edge  $f$  as a vertex in  $F$  and mark  $f$  "unlinked."

For each vertex  $v$  of  $G$ , let  $p(v)$  be the parent of  $v$  in the DFS spanning tree.

**for** each vertex  $v$ , in increasing rank order as visited in the DFS traversal **do**

**for** each back edge  $e = (u, v)$  with destination  $v$  **do**

        Add  $e$  as a vertex of the graph  $F$ .

        // Match up from  $u$  to  $s$  adding edges to  $F$  only as necessary.

**while**  $u \neq v$  **do**

            Let  $f$  be the vertex in  $F$  corresponding to the discovery edge  $(u, p(u))$ .

            Add the edge  $(e, f)$  to  $F$ .

**if**  $f$  is marked "unlinked" **then**

                Mark  $f$  as "linked."

$u \leftarrow p(u)$

**else**

$u \leftarrow v$  // shortcut to the end of the while loop

    Compute the connected components of the graph  $F$ .

© 2015 Goodrich and Tamassia

Biconnectivity

## Analysis with the Proxy Graph, $F$

- ◆ Proxy graph  $F$  for a connected graph  $G$ 
  - Spanning forest of the auxiliary graph  $B$
  - Has  $m$  vertices and  $O(m)$  edges
  - Can be constructed in  $O(n + m)$  time
  - Its connected components (trees) correspond to the link components of  $G$
- ◆ Given a graph  $G$  with  $n$  vertices and  $m$  edges, we can compute the following in  $O(n + m)$  time:
  - The biconnected components of  $G$
  - The separation vertices of  $G$
  - The separation edges of  $G$

DFS on graph  $G$

Proxy graph  $F$

© 2015 Goodrich and Tamassia      Biconnectivity      13