

## Lecture 26: Graph Algorithms

### Agenda:

- Graph traversal — Depth-first search
- DFS application: finding biconnected components

### Reading:

- Textbook pages 540 – 549, 558 – 559

## Depth First Search (DFS):

- Input: simple undirected graph  $G = (V, E)$
- Output: all vertices discovered (pick one vertex from each component as the start vertex)
- Idea: to search deeper in the graph whenever possible ...
- Pseudocode (recursive version):

```

procedure DFS( $G$ )           ** $G = (V, E)$ 

  for each  $v \in V$  do
     $c[v] \leftarrow \text{WHITE}$       **unknown yet
     $p[v] \leftarrow \text{NIL}$        **predecessor
  time  $\leftarrow 0$ 
  for each  $v \in V$  do
    if  $c[v] = \text{WHITE}$  then
      DFS-visit( $v$ )

procedure DFS-visit( $v$ )    **any  $v \in V$ 

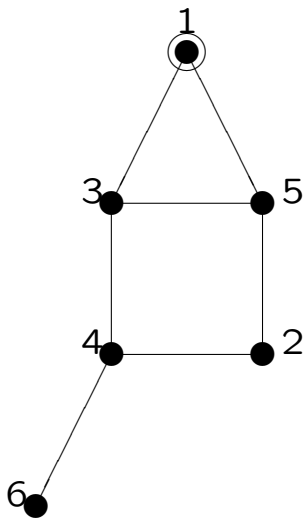
   $c[v] \leftarrow \text{GRAY}$       **start discovering  $v$ 
  time  $\leftarrow \text{time} + 1$ 
  dtime[ $v$ ]  $\leftarrow \text{time}$ 
  for each  $u \in \text{Adj}[v]$  do
    if  $c[u] = \text{WHITE}$  then
       $p[u] \leftarrow v$ 
      DFS-visit( $u$ )
   $c[v] \leftarrow \text{BLACK}$       **finished discovering
  time  $\leftarrow \text{time} + 1$ 
  ftime[ $v$ ]  $\leftarrow \text{time}$ 

```

## Lecture 26: Graph Algorithms

DFS example:

- $V = \{1, 2, 3, 4, 5, 6\}$   
 $E = \{\{1, 3\}, \{1, 5\}, \{2, 4\}, \{2, 5\}, \{3, 4\}, \{3, 5\}, \{4, 6\}\}$   
 $s = 2$



Adjacency lists:

|    |   |   |   |
|----|---|---|---|
| 1: | 3 | 5 |   |
| 2: | 4 | 5 |   |
| 3: | 1 | 4 | 5 |
| 4: | 2 | 3 | 6 |
| 5: | 1 | 2 | 3 |
| 6: | 4 |   |   |

|        | 1        | 2        | 3        | 4        | 5        | 6        | DFS-visit path       |
|--------|----------|----------|----------|----------|----------|----------|----------------------|
| color  | W        | W        | W        | W        | W        | W        | initialization       |
| parent | NIL      | NIL      | NIL      | NIL      | NIL      | NIL      |                      |
| dtime  | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ |                      |
| ftime  | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ |                      |
| color  | G        | W        | W        | W        | W        | W        | DFS-visit(1)         |
| parent | NIL      | NIL      | NIL      | NIL      | NIL      | NIL      |                      |
| dtime  | 1        | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ |                      |
| ftime  | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ |                      |
| color  | G        | W        | G        | W        | W        | W        | DFS-visit(1-3)       |
| parent | NIL      | NIL      | 1        | NIL      | NIL      | NIL      |                      |
| dtime  | 1        | $\infty$ | 2        | $\infty$ | $\infty$ | $\infty$ |                      |
| ftime  | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ |                      |
| color  | G        | W        | G        | G        | W        | W        | DFS-visit(1-3-4)     |
| parent | NIL      | NIL      | 1        | 3        | NIL      | NIL      |                      |
| dtime  | 1        | $\infty$ | 2        | 3        | $\infty$ | $\infty$ |                      |
| ftime  | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ |                      |
| color  | G        | G        | G        | G        | W        | W        | DFS-visit(1-3-4-2)   |
| parent | NIL      | 4        | 1        | 3        | NIL      | NIL      |                      |
| dtime  | 1        | 4        | 2        | 3        | $\infty$ | $\infty$ |                      |
| ftime  | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ |                      |
| color  | G        | G        | G        | G        | G        | W        | DFS-visit(1-3-4-2-5) |
| parent | NIL      | 4        | 1        | 3        | 2        | NIL      |                      |
| dtime  | 1        | 4        | 2        | 3        | 5        | $\infty$ |                      |
| ftime  | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ |                      |
| color  | G        | G        | G        | G        | B        | W        | DFS-visit(1-3-4-2-5) |
| parent | NIL      | 4        | 1        | 3        | 2        | NIL      |                      |
| dtime  | 1        | 4        | 2        | 3        | 5        | $\infty$ |                      |
| ftime  | $\infty$ | $\infty$ | $\infty$ | $\infty$ | 6        | $\infty$ |                      |
| color  | G        | B        | G        | G        | B        | W        | DFS-visit(1-3-4-2)   |
| parent | NIL      | 4        | 1        | 3        | 2        | NIL      |                      |
| dtime  | 1        | 4        | 2        | 3        | 5        | $\infty$ |                      |
| ftime  | $\infty$ | 7        | $\infty$ | $\infty$ | 6        | $\infty$ |                      |
| color  | G        | B        | G        | G        | B        | G        | DFS-visit(1-3-4-6)   |
| parent | NIL      | 4        | 1        | 3        | 2        | 4        |                      |
| dtime  | 1        | 4        | 2        | 3        | 5        | 8        |                      |
| ftime  | $\infty$ | 7        | $\infty$ | $\infty$ | 6        | $\infty$ |                      |
| color  | G        | B        | G        | G        | B        | B        | DFS-visit(1-3-4-6)   |
| parent | NIL      | 4        | 1        | 3        | 2        | 4        |                      |
| dtime  | 1        | 4        | 2        | 3        | 5        | 8        |                      |
| ftime  | $\infty$ | 7        | $\infty$ | $\infty$ | 6        | 9        |                      |
| color  | G        | B        | G        | B        | B        | B        | DFS-visit(1-3-4)     |
| parent | NIL      | 4        | 1        | 3        | 2        | 4        |                      |
| dtime  | 1        | 4        | 2        | 3        | 5        | 8        |                      |
| ftime  | $\infty$ | 7        | $\infty$ | 10       | 6        | 9        |                      |
| color  | G        | B        | B        | B        | B        | B        | DFS-visit(1-3)       |
| parent | NIL      | 4        | 1        | 3        | 2        | 4        |                      |
| dtime  | 1        | 4        | 2        | 3        | 5        | 8        |                      |
| ftime  | $\infty$ | 7        | 11       | 10       | 6        | 9        |                      |
| color  | B        | B        | B        | B        | B        | B        | DFS-visit(1)         |
| parent | NIL      | 4        | 1        | 3        | 2        | 4        |                      |
| dtime  | 1        | 4        | 2        | 3        | 5        | 8        |                      |
| ftime  | 12       | 7        | 11       | 10       | 6        | 9        |                      |

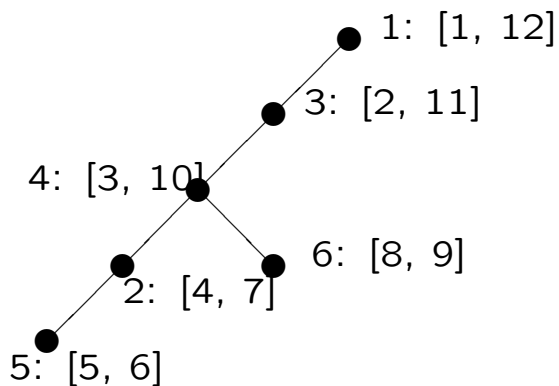
## DFS example:

- Adjacency lists:

```

1:  3  5
2:  4  5
3:  1  4  5
4:  2  3  6
5:  1  2  3
6:  4
    
```

- DFS tree: [dtime, ftime]



Notes:

- the result would be a forest of rooted trees
- the root of each tree is up to the selection (ordering of the vertices)
- parent of  $x$  is predecessor  $p[x]$
- different orderings of adjacency lists might result in different trees
- nested structure of [dtime, ftime]
  - they don't intersect each other

## DFS analysis:

- $n = |V|, m = |E|$
- Handshaking Lemma:  $\sum_{v \in V} \text{degree}(v) = 2m$
- Analysis:
  - each vertex is discovered exactly once (WHITE  $\rightarrow$  GRAY  $\rightarrow$  BLACK)  
each edge is examined exactly twice
  - running time:
    1. adjacency list representation:  
 $\Theta(n + 2m) = \Theta(n + m)$
    2. adjacency matrix representation:  
 $\Theta(n + n^2) = \Theta(n^2)$
  - space complexity:
    1. adjacency list representation:  
 $\Theta(n + m)$
    2. adjacency matrix representation:  
 $\Theta(n^2)$

## Classifying graph edges with BFS/DFS:

- During the traversal, all vertices and edges are examined
- Given a BFS/DFS traversal forest:
  - tree root — start vertex for that component
  - tree edge — child discovered while processing the parent
  - each edge in the original graph is examined twice

- Question:

Where are the other possible edges, besides tree edges ???

- Answer:

With respect to the traversal forest, categorize graph edges by their first time encounter:

- tree edges
- back edges: to ancestor
- forward edges: to descendant
- cross edges: to non-ancestor, non-descendant

Note: in undirected graphs, “back” = “forward”

- Examples:

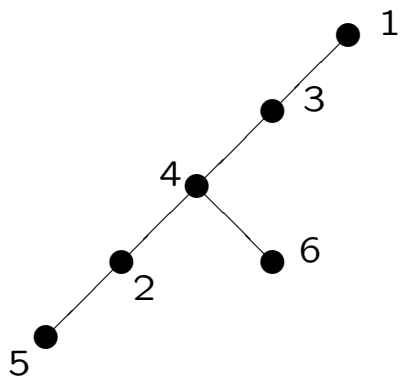
## Lecture 26: Graph Algorithms

An example:

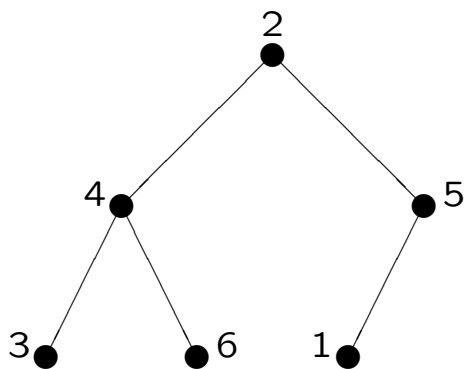
- Adjacency lists:

```
1:  3  5
2:  4  5
3:  1  4  5
4:  2  3  6
5:  1  2  3
6:  4
```

- DFS tree (start vertex 1):



- BFS Tree (start vertex 2):

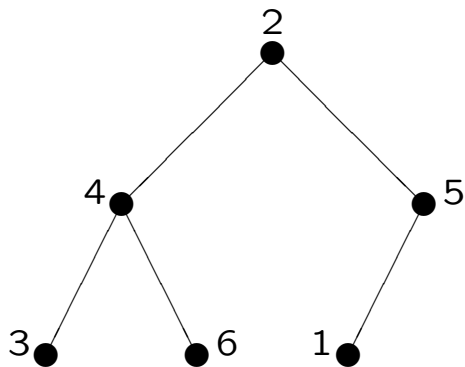


## Properties of BFS/DFS:

- BFS:
  - each graph edge connects two vertices with level-difference  $\leq 1$   
*Proof.*
  - no back / forward edges
- DFS:
  - each non-tree edge is a back edge  
*Proof.*
  - no forward edges (if  $G$  is undirected)
  - no cross edges
  - vertex processing time intervals  $[\text{dtime}[v], \text{ftime}[v]]$  and  $[\text{dtime}[w], \text{ftime}[w]]$ :
    - $[\text{dtime}[v], \text{ftime}[v]] \subset [\text{dtime}[w], \text{ftime}[w]]$  —  $v$  is a descendant of  $w$  in the DFS forest
    - $[\text{dtime}[v], \text{ftime}[v]] \cap [\text{dtime}[w], \text{ftime}[w]] = \emptyset$  — no ancestor-descendant relationship between  $v$  and  $w$
- BFS vertex order:  
level-order of each tree in the BFS forest
- DFS vertex order:  
pre-order of each tree in the DFS forest
- Some other vertex order associated with rooted trees:
  - in-order (for binary trees only)
  - post-order

Vertex order with respect to a binary rooted tree:

- Tree:



- Vertex orders:
  - level-order: level by level (each level: left to right)  
(2, 4, 5, 3, 6, 1)
  - pre-order: parent - child one - child two - ... - last child  
(2, 4, 3, 6, 5, 1)
  - in-order: left child - parent - right child  
(3, 4, 6, 2, 1, 5)
  - post-order: child one - child two - ... - last child - parent  
(3, 6, 4, 1, 5, 2)

## Biconnected component:

- Definition — every pair of vertices are connected by two vertex-disjoint paths
- Cut vertex — its removal increases the number of connected components
- Fact: biconnected  $\iff$  no cut vertices
- Biconnected component  $\iff$  maximal connected subgraph containing no cut vertex
- In a DFS tree:
  - root is a cut vertex **iff** it has  $\geq 2$  child vertices
  - any other vertex is a cut vertex **iff** vertices in the child subtrees have **no** back edges to its proper ancestors
- One simplest implementation (assuming connected):
  1. try every vertex  $v$  as the start vertex and do the DFS
  2. in the DFS tree, if  $\text{degree}_{DFS}(v) > 1$ , decompose the graph accordingly into  $\text{degree}_{DFS}(v)$  subgraphs with only one common vertex  $v$
  3. repeat on subgraphs until for every subgraph the DFS tree with every possible start vertex has a root degree 1

**Problem:** too time consuming  $\Theta(n(n + m))$  ...

- Idea in finding biconnected components via DFS tree
  - $(\Theta(n + m))$ :  
for each vertex  $v$ , and each of its child  $w$ , keep track of furthest back edge from the  $w$ -subtree (detail next lecture)

## Lecture 26: Graph Algorithms

Have you understood the lecture contents?

| well                     | ok                       | not-at-all               | topic                              |
|--------------------------|--------------------------|--------------------------|------------------------------------|
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | depth first search execution       |
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | depth first search analysis        |
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | graph edge categorization          |
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | BFS/DFS vertex order               |
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | biconnected component & cut vertex |
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | one simplest implementation        |