WS 2007/2008

Fundamental Algorithms

Dmytro Chibisov, Jens Ernst

Fakultät für Informatik
TU München

http://www14.in.tum.de/lehre/2007WS/fa-cse/

Fall Semester 2007
1. Application of DFS: topological sort

1.1 Correctness of TopSort
For every vertex $v$ introduce the new variable $v.f\text{inished}$ as well as the second global variable $counter2$:

**Topological Sorting:**

```c
void TopSort(vertex v){
    foreach ($v \in V$) do $v.dfsnum := 0$; $v.f\text{inished} := 0$; od
    while $\exists v_0 \in V : v_0.dfsnum = 0$ do modified-DFS($v_0$) od
}
```

**Modified DFS:**

```c
void modified-DFS(vertex v){
    $v.dfsnum:= counter++;$
    foreach ($w|(v, w) \in E$) do
        if ($w.dfsnum=0$) then modified-DFS($w$); fi
    od
    $v.f\text{inished}:= counter2++;$
    push($v$) }
```
1. Application of DFS: topological sort

1.1 Correctness of TopSort

For every vertex \( v \) introduce the new variable \( v\).\textit{finished} as well as the second global variable \( \text{counter2} \):

**Topological Sorting:**

\[
\text{void TopSort(vertex } v)\{
\text{foreach } (v \in V ) \text{ do } v.dfsnum := 0; v\text{.finished} := 0; \text{ od}
\text{ while } \exists v_0 \in V : v_0\text{.dfsnum} = 0 \text{ do modified-DFS}(v_0) \text{ od}
\}
\]

**Modified DFS:**

\[
\text{void modified-DFS(vertex } v)\{
\text{ } v.dfsnum := \text{counter}++;
\text{ foreach } (w | (v, w) \in E) \text{ do}
\text{ if } (w\text{.dfsnum} = 0) \text{ then modified-DFS}(w); \text{ fi}
\text{ od}
\text{ } v\text{.finished} := \text{counter2}++;
\text{ push(v) } \}
\]
modified-DFS performs the partition of edges into four classes:

- **Tree edges** – edge \((u, v)\) is a tree edge if \(v\) was first discovered by exploring edge \((u, v)\) \((v.dfsnum = 0)\).
- **Back edges** – edge \((u, v)\) connecting a vertex \(u\) to an ancestor \(v\) in a depth-first tree \((v.dfsnum < u.dfsnum\), and DFS\((v)\) is not finished).
- **Forward edges** – non-tree edges \((u, v)\) connecting a vertex \(u\) to a descendant \(v\) in a depth-first tree \((v.dfsnum > u.dfsnum)\).
- **Cross edges** – are all other edges \((u.dfsnum > v.dfsnum, \text{ and } DFS(v) \text{ is finished})\).
Theorem 1

A $G = (V, E)$ is acyclic if and only if a depth-first search yields no back edges.

Proof.

$\Rightarrow$:

- suppose that there is a back edge $(u, v)$. Then, $v$ is an ancestor of $u$ in the depth-first forest (why? prove!). Thus, there is a path from $v$ to $u$, and $(u, v)$ finishes the cycle.

$\Leftarrow$:

- Suppose that $G$ contains a cycle $c$. We show that a depth-first search of $G$ yields a back edge. Let $v$ be the first vertex to be discovered in $c$, and let $(u, v)$ be the preceding edge in $c$. At step $v.dfsnum$, there is a path of unvisited vertices in $c$. Thus, $u$ becomes a descendant of $v$ in the depth-first forest. Therefore, $(u, v)$ is a back edge.
Theorem 2

TopSort(G) produces a topological sort of a directed acyclic graph G.

Proof.

- It suffices to show that for any pair of distinct vertices $u, v \in V$, if there is an edge $(u, v) \in E$, then $v.f\text{inished} < u.f\text{inished}$.
- When modified-DFS(G) explores $(u, v)$ three cases may occur (due to Theorem 1 $(u, v)$ may be cross, forward, or tree edge):
  - $(u, v)$ is a tree edge: $v.f\text{inished} < u.f\text{inished}$.
  - $(u, v)$ is a forward edge: $v.f\text{inished} < u.f\text{inished}$
  - $(u, v)$ is a cross edge: $v.f\text{inished} < u.f\text{inished}$
1.2 Application of Depth First Search: determining biconnected components

Definition 3
Let $G = (V, E)$ be a connected undirected graph. A vertex $a$ is said to be an *articulation point* of $G$ if there exist vertices $v$ and $w$, and every path between $v$ and $w$ contains the vertex $a$.

Stated another way, $a$ is an articulation point of $G$ if removing $a$ splits $G$ into two or more parts.

Definition 4
The graph $G = (V, E)$ is called biconnected if for every distinct triple of vertices $v$, $w$, and $a$ there exist a path between $v$ and $w$ not containing $a$. 

**Example 5**

Consider $G = (V, E)$ such that

$$V = (v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9)$$

and

$$E = (\{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3\}, \{v_2, v_5\}, \{v_4, v_5\}, \{v_4, v_6\}, \{v_6, v_9\}, \{v_6, v_8\}, \{v_6, v_7\}, \{v_7, v_8\}, \{v_8, v_9\})$$

**Articulation nodes:** $v_2$, $v_4$, $v_6$. **Biconnected components:**

$E_1 = (\{v_4, v_6\})$, $V_1 = (v_4, v_6)$; $V_2 = (v_1, v_2, v_3)$,

$E_2 = (\{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3\})$; $V_3 = (v_2, v_4, v_5)$,

$E_3 = (\{v_2, v_4\}, \{v_2, v_5\}, \{v_4, v_5\})$; $V_4 = (v_6, v_7, v_8, v_9)$

$E_4 = (\{v_6, v_7\}, \{v_6, v_8\}, \{v_6, v_9\}, \{v_9, v_8\}, \{v_7, v_8\})$. 
Example 6

Consider the electric power net supplying a city. The failure at the articulation point of the net leads to power blackout of some parts of the city. To locate the crash - find the articulated vertices of the power net graph. To design the safe power supply - check the biconnectivity of the power net graph.
Theorem 7

If \( \{u, v\} \) is a back edge, then in the DFS forest \( u \) is an ancestor of \( v \) or vice versa.

Proof.

Easy. Homework.