Correctness of Breadth-First Search (2/3)

Proof. For the purpose of contradiction, let v be the vertex with $v.d > \delta(s, v)$.

- Vertex v must be reachable from s, for if it is not, then $\delta(s, v) = \infty \geq v.d$.
- Let u be the vertex immediately preceding v on a shortest path from s to v, so that $\delta(s, v) = \delta(s, u) + 1$.
- Because $\delta(s, u) < \delta(s, v)$, and because of how we chose v, we have $u.d = \delta(s, u)$.
- Putting these properties together, we have $v.d > \delta(s, v) = \delta(s, u) + 1 = u.d + 1$ (1)

Correctness of Breadth-First Search (3/3)

Now consider the time when BFS chooses to dequeue vertex u from Q in line 11.

- If v is white, then line 15 sets $v.d = u.d + 1$, contradicting inequality (1).
- If v is black, then it was already removed from the queue and, by **Corollary 4**, we have $v.d \leq u.d$, again contradicting inequality (1).
- If v is gray, then it was painted gray upon dequeuing some vertex w and $v.d = w.d + 1$. By **Corollary 4**, however, $w.d \leq u.d$, and so we have $v.d = w.d + 1 \leq u.d + 1$, once again contradicting inequality (1).

➡ Thus we conclude that $v.d = \delta(s, v)$ for all $v \in V$.

Breadth-First Trees (1/4)

- The procedure BFS builds a breadth-first tree as it searches the graph.
  - The tree corresponds to the π attributes.
- For a graph $G = (V, E)$ with source s, we define the predecessor subgraph of G as $G_\pi = (V_\pi, E_\pi)$, where

$$V_\pi = \{v \in V : v.\pi \neq \text{NIL}\} \cup \{s\}$$

and

$$E_\pi = \{(v.\pi, v) : v \in V_\pi - \{s\}\} .$$

Breadth-First Trees (2/4)

- The predecessor subgraph G_π is a breadth-first tree if V_π consists of vertices reachable from s and, for all $v \in V_\pi$, the subgraph G_π contains a unique simple path from s to v that is also a shortest path from s to v in G.

Breadth-First Trees (3/4)

- **Lemma 6.** When applied to a directed or undirected graph $G = (V, E)$, procedure BFS constructs π so that the predecessor subgraph $G_\pi = (V_\pi, E_\pi)$ is a breadth-first tree.

  Proof. Line 16 of BFS sets $v.\pi = u$ if and only if $(u, v) \in E$ and $\delta(s, v) < \infty$, thus V_π consists of the vertices in V reachable from s.

  Since G_π forms a tree, it contains a unique simple path from s to each vertex in V_π.

  By applying **Theorem 5** inductively, we conclude that every such path is a shortest path in G.

Breadth-First Trees (4/4)

- The following procedure prints out the vertices on a shortest path from s to v, assuming that BFS has already computed a breadth-first tree:

```
PRINT-PATH(G, s, v)
if v == s
    print s
elseif v.π == NIL
    print "no path from" s "to" v "exists"
else PRINT-PATH(G, s, v.π)
    print v
```


Outline

- Representations of Graphs
- Breadth-First Search
- ***Depth-First Search***
- Topological Sort
- Strongly Connected Components

Depth-First Search (1/2)

- The strategy followed by depth-first search is to search "deeper" in the graph whenever possible.
  - Explore edges out of the most recently discovered vertex v that still has unexplored edges leaving it.
  - Once all of v's edges have been explored, the search "backtracks" to explore edges leaving the vertex from which v was discovered.
  - This process continues until we have discovered all the vertices that are reachable from the original source vertex.

Depth-First Search (2/2)

- If any undiscovered vertices remain, then depth-first search selects one of them as a new source, and it repeats the search from that source.
- The algorithm repeats this entire process until it has discovered every vertex.

Predecessor Subgraph

- We define the predecessor subgraph of a depth-first search slightly differently from that of a breadth-first search: we let $G_\pi = (V, E_\pi)$, where $E_\pi = \{(v.\pi, v) : v \in V \text{ and } v.\pi \neq \text{NIL}\}$.
- The predecessor subgraph of a depth-first search forms a depth-first forest comprising several depth-first trees. The edges in E_π are tree edges.

Timestamps

- Each vertex v has two timestamps: the first timestamp $v.d$ records when v is first discovered (and grayed), and the second timestamp $v.f$ records when the search finishes examining v's adjacency list (and blackens v).
- ➡ Vertex u is WHITE before time $u.d$, GRAY between time $u.d$ and time $u.f$, and BLACK thereafter.

DFS(G)

- The input graph G may be undirected or directed.
- The variable time is a global variable that we use for timestamping.

```
DFS(G)
for each vertex u ∈ G.V
    u.color = WHITE
    u.π = NIL
time = 0
for each vertex u ∈ G.V
    if u.color == WHITE
        DFS-VISIT(G, u)
```

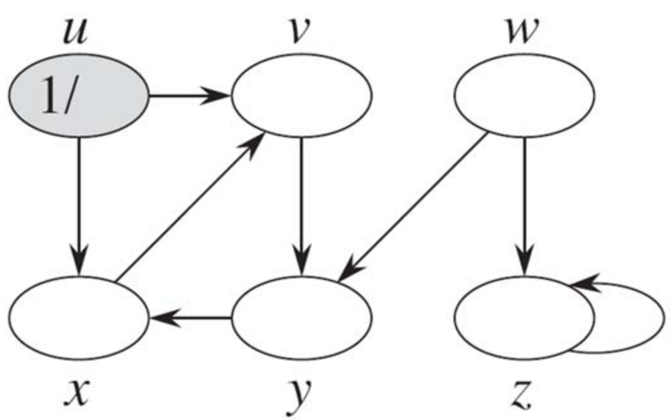
DFS-VISIT(G, u)

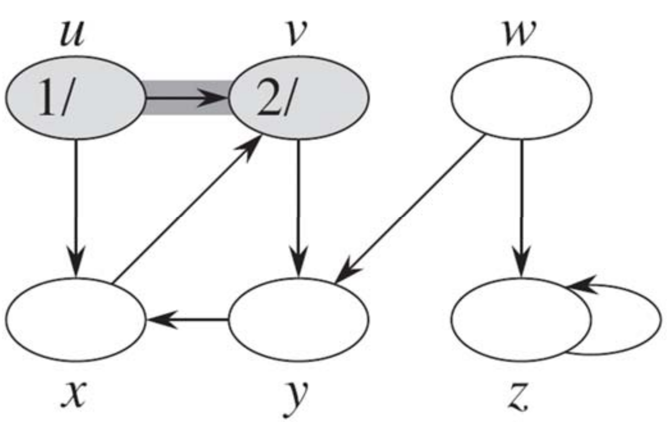
```
DFS-VISIT(G, u)
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
    if v.color == WHITE
        v.π = u
        DFS-VISIT(G, v)
u.color = BLACK
time = time + 1
u.f = time
```

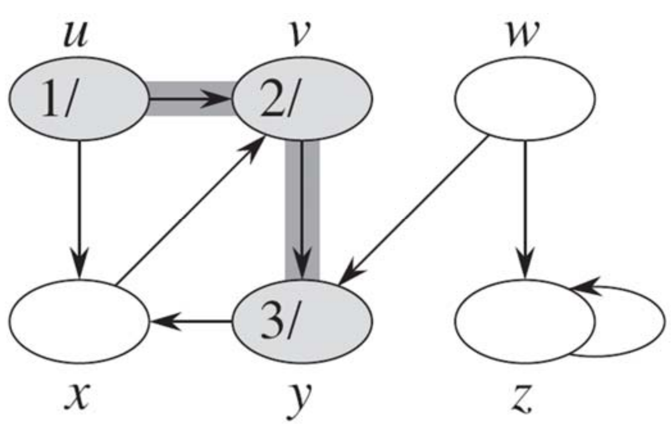
white vertex u has just been discovered.

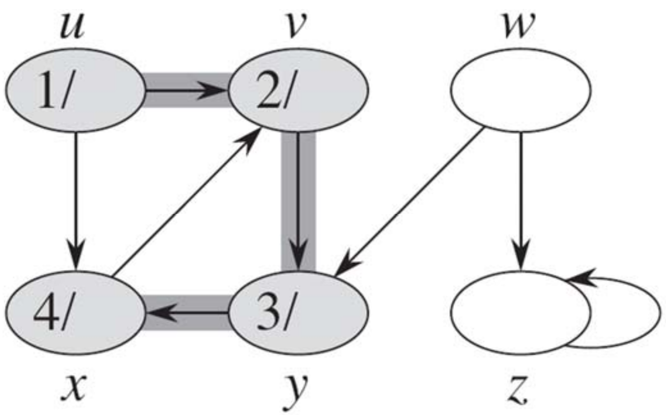
explore edge (u, v)

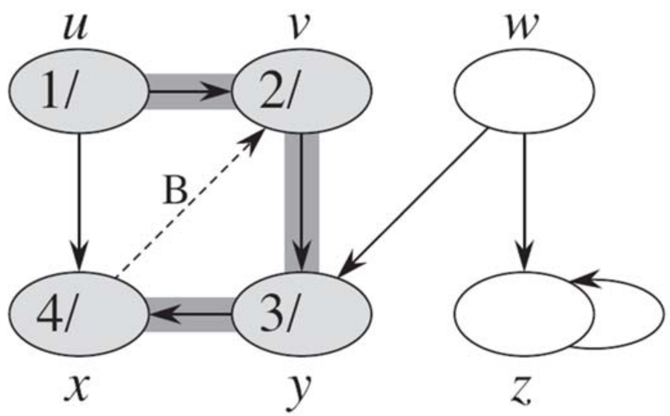
blacken u; it is finished

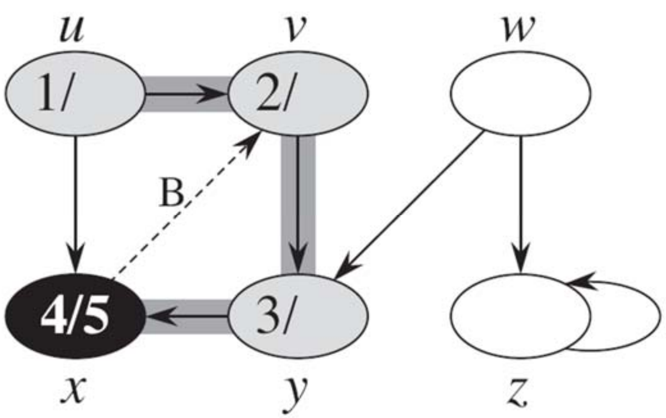


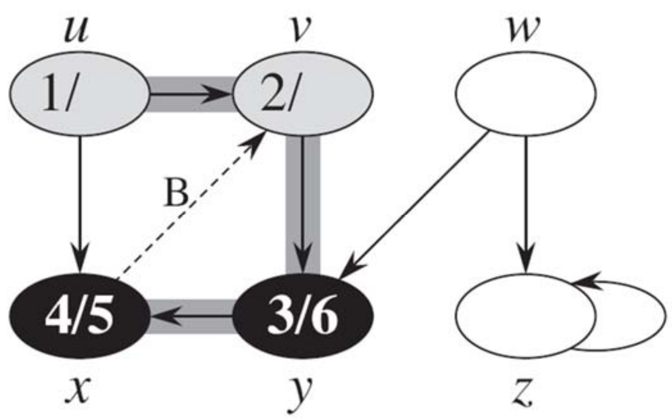


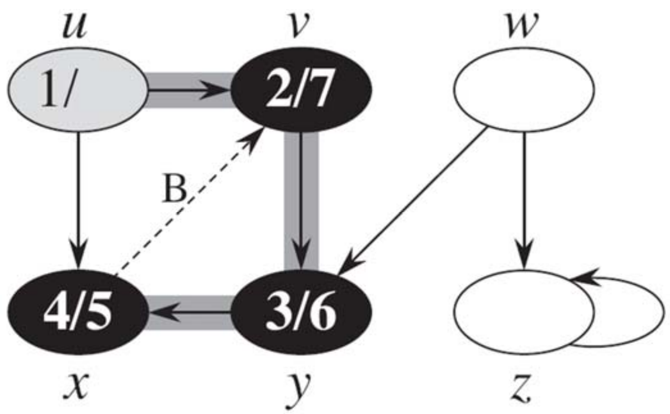


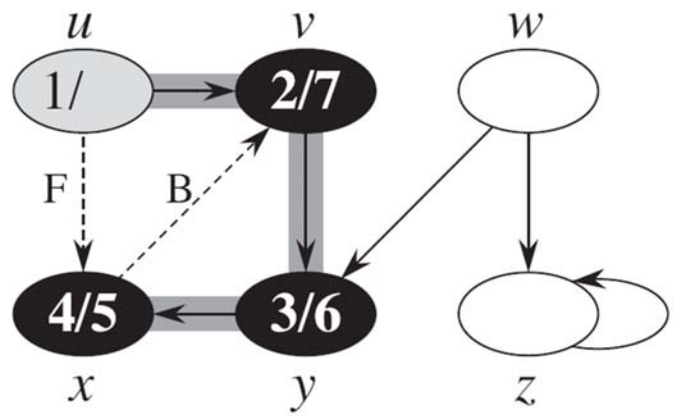

u
v
w
1/
2/
4/
3/
x
y
z

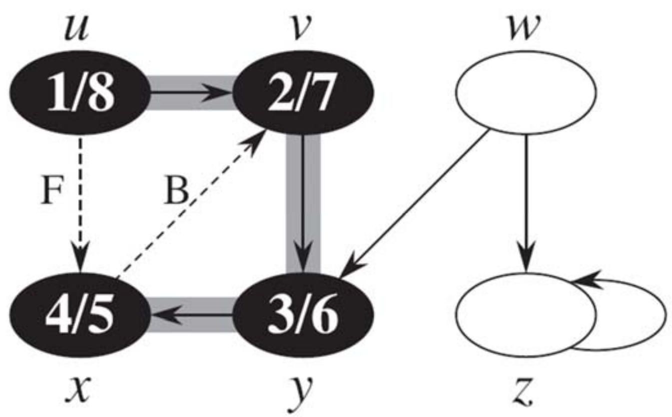

u
v
w
1/
2/
B
4/
3/
x
y
z

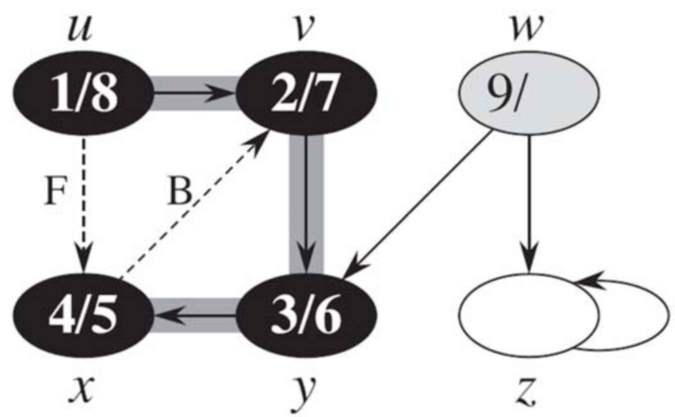

u
v
w
1/
2/
B
4/5
3/
x
y
z

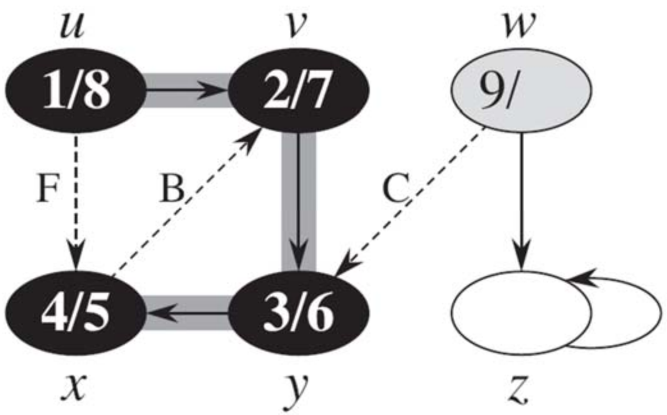

u
v
w
1/
2/
B
4/5
3/6
x
y
z

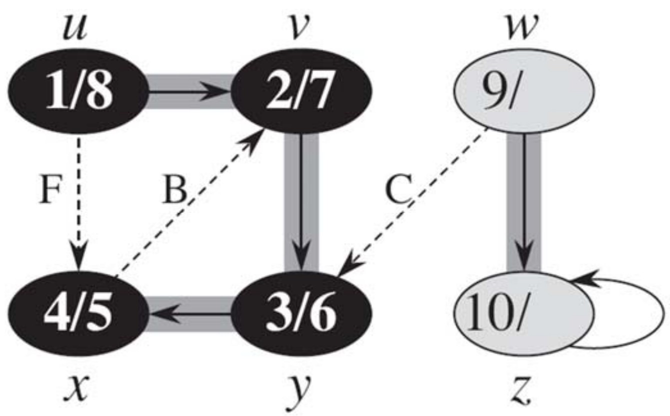

u
v
w
1/
2/7
B
4/5
3/6
x
y
z

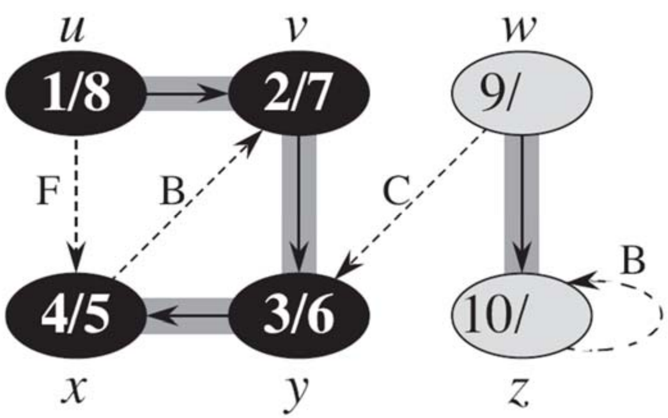

u
v
w
1/
2/7
F
B
4/5
3/6
x
y
z

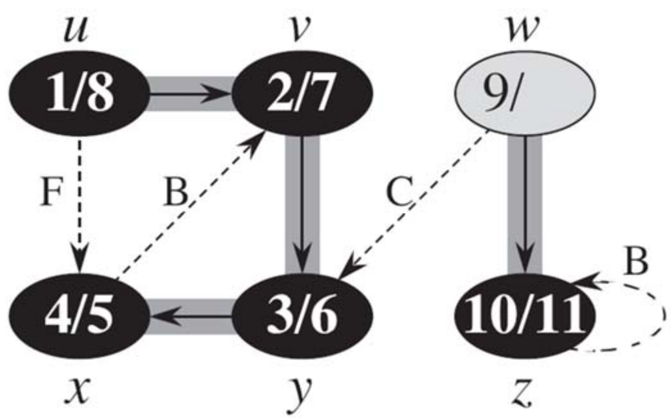

u
v
w
1/8
2/7
F
B
4/5
3/6
x
y
z

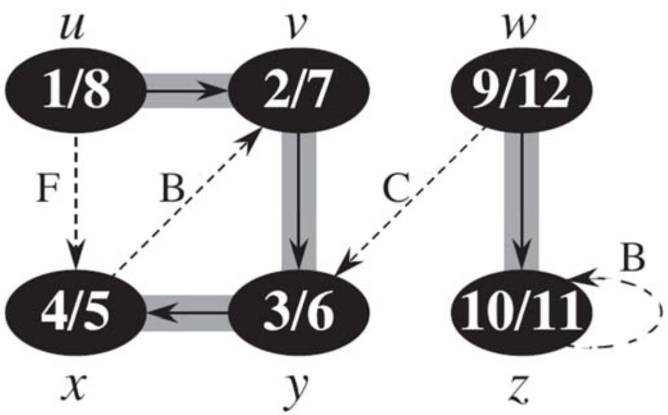

u
v
w
1/8
2/7
9/
F
B
4/5
3/6
x
y
z


u
v
w
1/8
2/7
9/
F
B
C
4/5
3/6
x
y
z


u
v
w
1/8
2/7
9/
F
B
C
4/5
3/6
10/
x
y
z

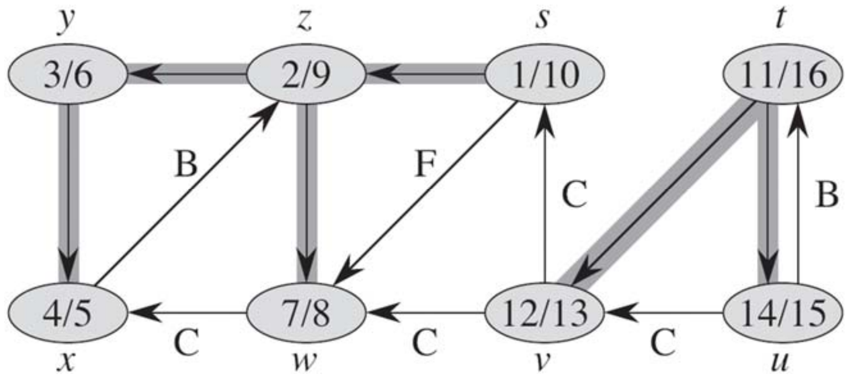

u
v
w
1/8
2/7
9/
F
B
C
B
4/5
3/6
10/
x
y
z


u
v
w
1/8
2/7
9/
F
B
C
B
4/5
3/6
10/11
x
y
z



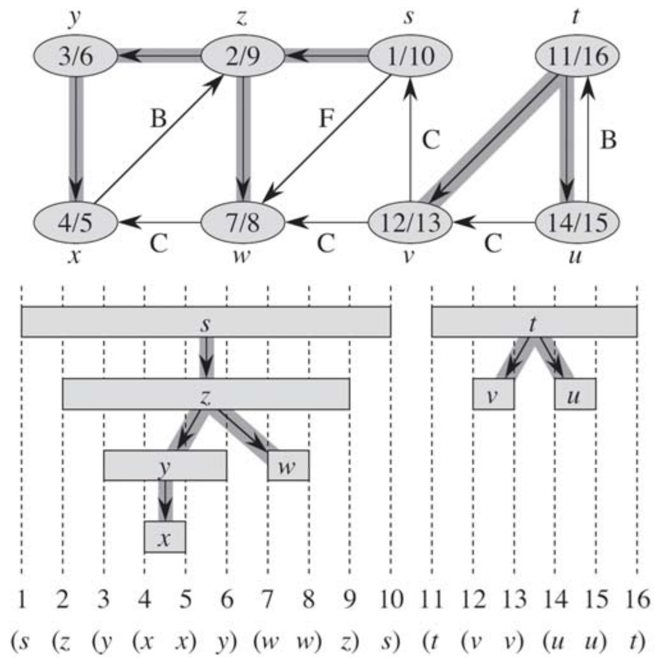
Properties of Depth-First Search (1/9)

- The most basic property of depth-first search is that the predecessor subgraph G_π does indeed form a forest of trees.



Properties of Depth-First Search (2/9)

- Another important property of depth-first search is that discovery and finishing times have parenthesis structure.
- If we represent the discovery of vertex u with a left parenthesis "(u" and represent its finishing by a right parenthesis "u)", then the history of discoveries and finishings makes a well-formed expression in the sense that the parentheses are properly nested.



Properties of Depth-First Search (3/9)

- **Theorem 7 (Parenthesis Theorem).** In any depth-first search of a (directed or undirected) graph $G = (V, E)$, for any two vertices u and v, exactly one of the following three conditions holds:
  - the intervals $[u.d, u.f]$ and $[v.d, v.f]$ are entirely disjoint, and neither u nor v is a descendant of the other in the depth-first forest,
  - the interval $[u.d, u.f]$ is contained entirely within the interval $[v.d, v.f]$, and u is a descendant of v in a depth-first tree, or

Properties of Depth-First Search (4/9)

- the interval $[v.d, v.f]$ is contained entirely within the interval $[u.d, u.f]$, and v is a descendant of u in a depth-first tree.

Proof. For the case $u.d < v.d$, we consider two subcases, according to whether $v.d < u.f$ or not.

When $v.d < u.f$, v was discovered while u was still gray ➡ v is a descendant of u.

Since v was discovered more recently than u, all of its outgoing edges are explored, and v is finished, before the search returns to and finishes u.

Properties of Depth-First Search (5/9)

➡ The interval $[v.d, v.f]$ is entirely contained within the interval $[u.d, u.f]$.

When $u.f < v.d$, we have $u.d < u.f < v.d < v.f$.

➡ The intervals $[u.d, u.f]$ and $[v.d, v.f]$ are disjoint.

Because the intervals are disjoint, neither vertex was discovered while the other was gray, and so neither vertex is a descendant of the other.

The case in which $v.d < u.d$ is similar, with the roles of u and v reversed in the above argument.

Properties of Depth-First Search (6/9)

- **Corollary 8 (Nesting of Descendants' Intervals).** Vertex v is a proper descendant of vertex u in the depth-first forest for a (directed or undirected) graph G if and only if $u.d < v.d < v.f < u.f$.

Proof. Immediate from Theorem 7.