Properties of Depth-First Search (7/9)

- **Theorem 9 (White-Path Theorem).** In a depth-first forest of a (directed or undirected) graph $G = (V, E)$, vertex v is a descendant of vertex u if and only if at the time $u.d$ that the search discovers u, there is a path from u to v consisting entirely of white vertices.

  Proof. $\Rightarrow$: If $v = u$, then the path from u to v contains just vertex u, which is still white when we set the value of $u.d$.

  Now, suppose that v is a proper descendant of u in the depth-first forest.

Properties of Depth-First Search (8/9)

By Corollary 8, $u.d < v.d$, and so v is white at time $u.d$. Since v can be any descendant of u, all vertices on the unique simple path from u to v in the depth-first forest are white at time $u.d$.

$\Leftarrow$: Suppose that there is a path of white vertices from u to v at time $u.d$, but v does not become a descendant of u in the depth-first tree.

Assume that every vertex other than v along the path becomes a descendant of u. (Otherwise, let v be the closest vertex to u along the path that doesn't become a descendant of u.)

Properties of Depth-First Search (9/9)

Let w be the predecessor of v in the path, so that w is a descendant of u (w and u may in fact be the same vertex).

By Corollary 8, $w.f \leq u.f$. Because v must be discovered after u is discovered, but before w is finished, we have $u.d < v.d < w.f \leq u.f$.

Theorem 7 then implies that the interval $[v.d, v.f]$ is contained entirely within the interval $[u.d, u.f]$.

By Corollary 8, v must after all be a descendant of u.

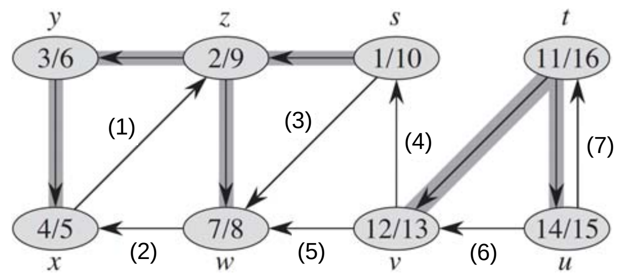
Classification of Edges (1/5)

- We can define four edge types in terms of the depth-first forest G_π produced by a depth-first search on G:

1. **Tree edges** are edges in the depth-first forest G_π. Edge (u, v) is a tree edge if v was first discovered by exploring edge (u, v).
2. **Back edges** are those edges (u, v) connecting a vertex u to an ancestor v in a depth-first tree. We consider self-loops, which may occur in directed graphs, to be back edges.

Classification of Edges (2/5)

3. **Forward** edges are those nontree edges (u, v) connecting a vertex u to a descendant v in a depth-first tree.
4. **Cross edges** are all other edges. They can go between vertices in the same depth-first tree, as long as one vertex is not an ancestor of the other, or they can go between vertices in different depth-first trees.



Classification of Edges (3/5)

- The DFS algorithm has enough information to classify some edges as it encounters them.
- The key idea is that when we first explore an edge (u, v), the color of vertex v tells us something about the edge:
  1. WHITE indicates a tree edge,
  2. GRAY indicates a back edge, and
  3. BLACK indicates a forward or cross edge.

Classification of Edges (4/5)

- **Theorem 10.** In a depth-first search of an undirected graph G, every edge of G is either a tree edge or a back edge.

  Proof. Let (u, v) be an arbitrary edge of G, and suppose without loss of generality that $u.d < v.d$. Then the search must discover and finish v before it finishes u (while u is gray), since v is on u's adjacency list.

Classification of Edges (5/5)

If the first time that the search explores edge (u, v), it is in the direction from u to v, then v is undiscovered (white) until that time.

➡ (u, v) becomes a tree edge.

If the search explores (u, v) first in the direction from v to u, then (u, v) is a back edge, since u is still gray at the time the edge is first explored.

Outline

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

Topological Sort

- A topological sort of a directed acyclic graph (dag) $G = (V, E)$ is a linear ordering of all its vertices such that if G contains an edge (u, v), then u appears before v in the ordering. (If the graph contains a cycle, then no linear ordering is possible.)
- We can view a topological sort of a graph as an ordering of its vertices along a horizontal line so that all directed edges go from left to right.
- Many applications use directed acyclic graphs to indicate precedence among events.

Example

- Professor Bumstead topologically sorts his clothing when getting dressed.
- A directed edge (u, v) in the dag indicates that garment u must be donned before garment v.