

22.3 Depth First Search

Depth-first search always visits a neighbour of the most recently visited vertex with an unvisited neighbour.

This means the search moves “forward” when possible, and only “backtracks” when moving forwards is impossible. (Sometimes DFS is called “backtrack search”.)

DFS can be written similarly to BFS, but using a **stack** rather than a **queue**, but usually it is written as a recursive procedure. This means there is an implicit stack implemented by the run-time system.

As with BFS, vertices are colored white, gray, or black during the search, with the same meanings.

22.3 Depth First Search (continued)

As well as $color[v]$ and $\pi[v]$, the search defines:

- $count$: a clock with values $0, 1, 2, \dots$
- $d[v]$: discovery time (when v is first visited)
- $f[v]$: finishing time (when all the neighbours of v have been examined)

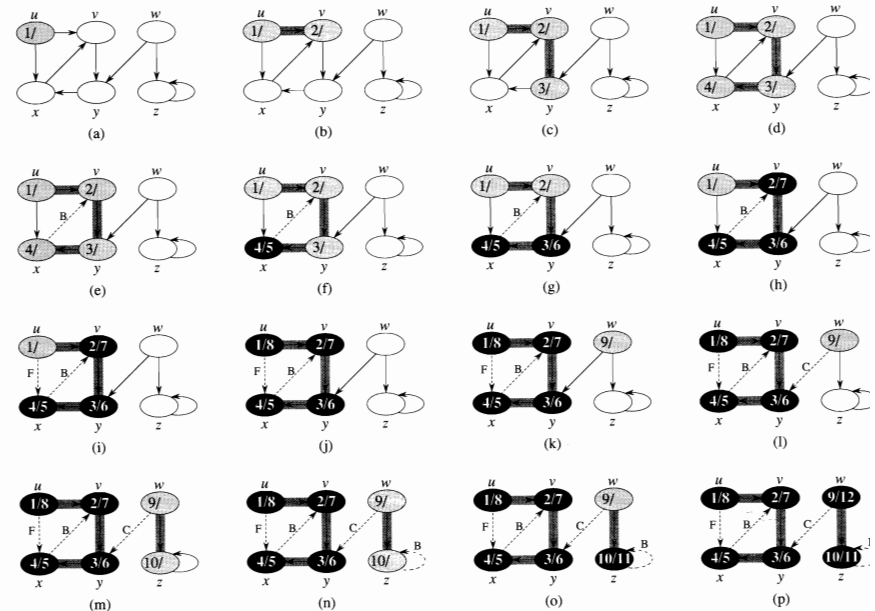
DFS(G)

- 1 for each vertex $u \in V[G]$ do
- 2 $color[u] \leftarrow white$
- 3 $\pi[u] \leftarrow NIL$
- 4 $count \leftarrow 0$
- 5 for each $u \in V[G]$ do
- 6 if $color[u] = white$ then
- 7 **DFS_Visit(u)**

22.3 Depth First Search (continued)

DFS_Visit(u)

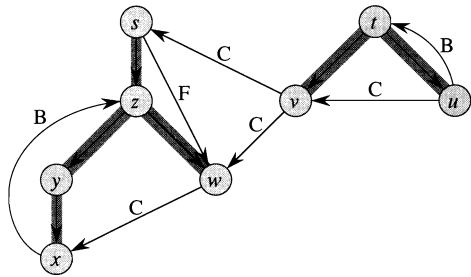
- 1 $color[u] \leftarrow gray$
- 2 $count \leftarrow count + 1$
- 3 $d[u] \leftarrow count$
- 4 for each $v \in Adj[u]$ do
- 5 if $color[v] = white$ then
- 6 $\pi[v] \leftarrow u$
- 7 **DFS_Visit(v)**
- 8 $color[u] \leftarrow black$
- 9 $count \leftarrow count + 1$
- 10 $f[u] \leftarrow count$



22.3 Edge classification in DFS

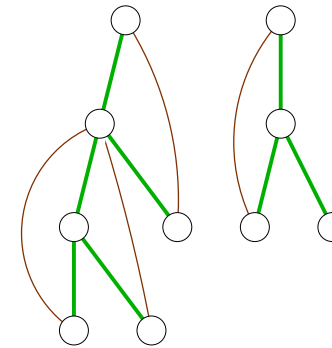
DFS classifies the edges in a directed graph into four types.

- **Tree edges** (edges along which a vertex is first discovered)
- **Back edges** (edges from a descendant to an ancestor)
- **Forward edges** (non-tree edges from an ancestor to a descendant)
- **Cross edges** (all other edges)



22.3 Edge classification in DFS (continued)

For an edge (u, v) in an undirected graph, (v, u) is the same edge. This gives some ambiguity, so we don't consider any edges to be forward edges and we don't consider the reverse of a tree edge to be a back edge.

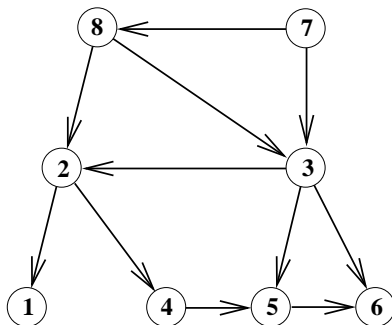


Theorem. In a depth-first search of an *undirected* graph G , each edge in G is either a tree edge or a back edge.

22.4 Topological Sorting

A directed graph $G(V, E)$ is called **acyclic** if it does not contain a directed cycle.

A **topological ordering** of a directed acyclic graph is a linear ordering of the vertices such that every edge is directed left to right.



7-8-3-2-4-5-6-1

22.4 Topological Sorting (continued)

Theorem. A directed graph G has a directed cycle if and only if a depth-first search of G yields a back edge.

Proof:

(\Leftarrow) If there is a back edge (u, v) , then v is an ancestor of u (by definition of back edge). Then the path of tree edges from v to u , together with (u, v) , forms a directed cycle.

(\Rightarrow) Let $v_1, v_2, \dots, v_k, v_1$ be a directed cycle in G , where v_1 is the first vertex discovered during the DFS. By induction, vertices v_2, \dots, v_k are descendants of v_1 . Therefore (v_k, v_1) is a back edge.

22.4 Topological Sorting (continued)

Theorem. Perform DFS on an acyclic directed graph G . Then the reverse order of finishing time is a topological ordering for G .

Proof:

Let (u, v) be an edge of G . We need to show that $f[v] < f[u]$.

Note that the finish time of a vertex x is when the call **DFS_Visit**(x) returns. The recursive structure of **DFS_Visit** implies that descendants always finish before their ancestors. Therefore, if (u, v) is a tree edge or forward edge, $f[v] < f[u]$.

Suppose that (u, v) is a cross-edge. It cannot be that u is discovered before v , since then v would be a descendant of u (since the call **DFS_Visit**(u) cannot return until v is visited). Therefore v is discovered before u . Since u is not a descendant of v , v must be finished before u is discovered. So $f[v] < f[u]$ again.