CS590/CPE590

Graph Algorithms | DFS

Kazi Lutful Kabir

Spring 2023

Graph Searching

- Given: a graph G = (V, E), directed or undirected
- Goal: methodically explore every vertex and every edge
- Ultimately: build a tree on the graph
  - Pick a vertex as the root
  - Choose certain edges to produce a tree

- There are two standard graph traversal techniques:
  - Breadth-First Search (BFS)
  - **Depth-First Search (DFS)**

Depth-First Search

- *Depth-first search* is another strategy for exploring a graph
  - Explore "deeper" in the graph whenever possible
  - Edges are explored out of the most recently discovered vertex *v* that still has unexplored edges
  - When all of *v*'s edges have been explored, backtrack to the vertex from which *v* was discovered

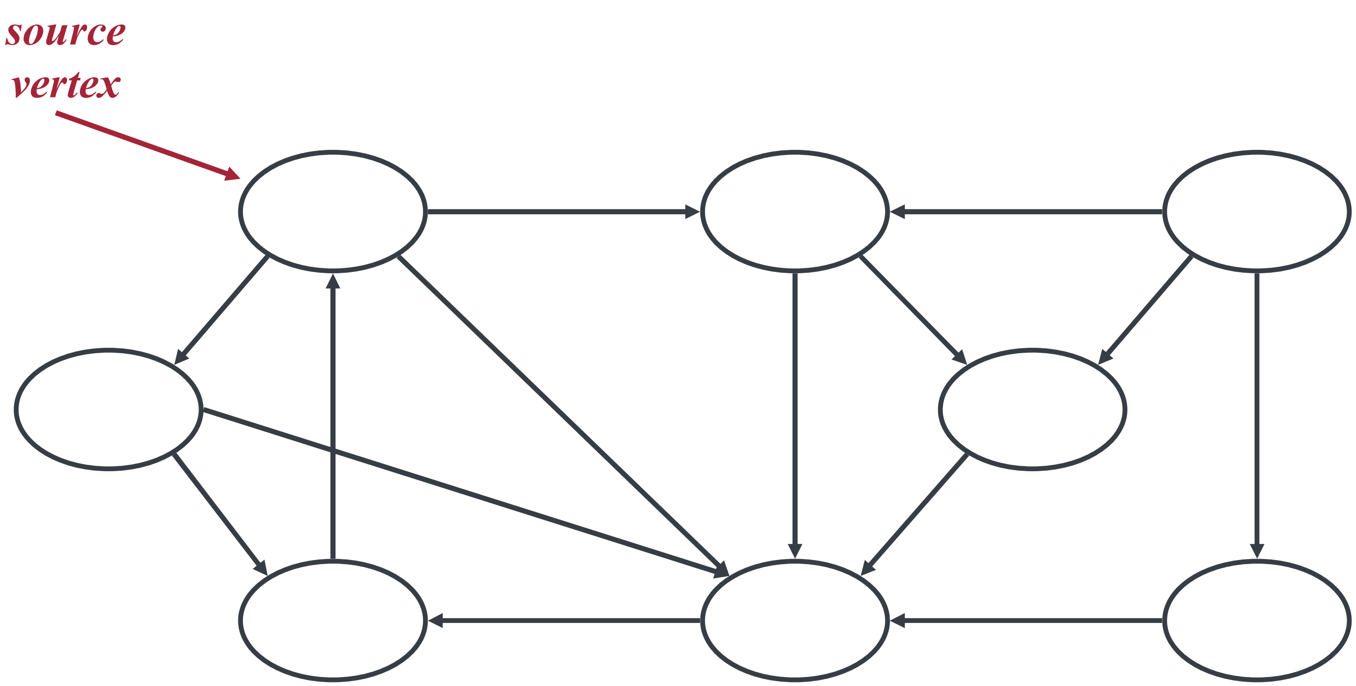
- Vertices initially colored white
- Then colored grey when discovered
- Then black when finished

Depth-First Search: The Code

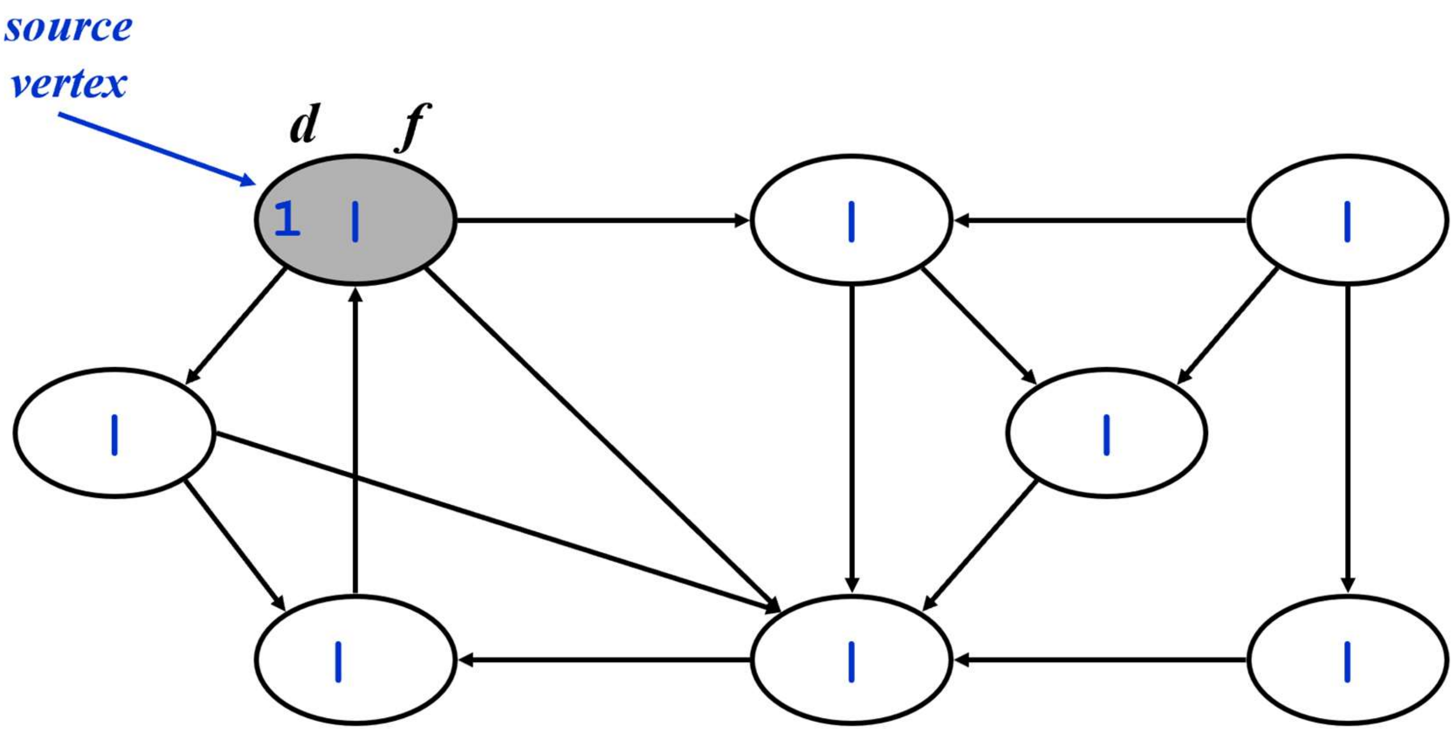
```
DFS(G)
for each vertex u ∈ G.V
    u.color = WHITE
    u.π = NIL
time = 0
for each vertex u ∈ G.V
    if u.color == WHITE
        DFS-VISIT(G, u)
```

```
DFS-VISIT(G, u)
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
```

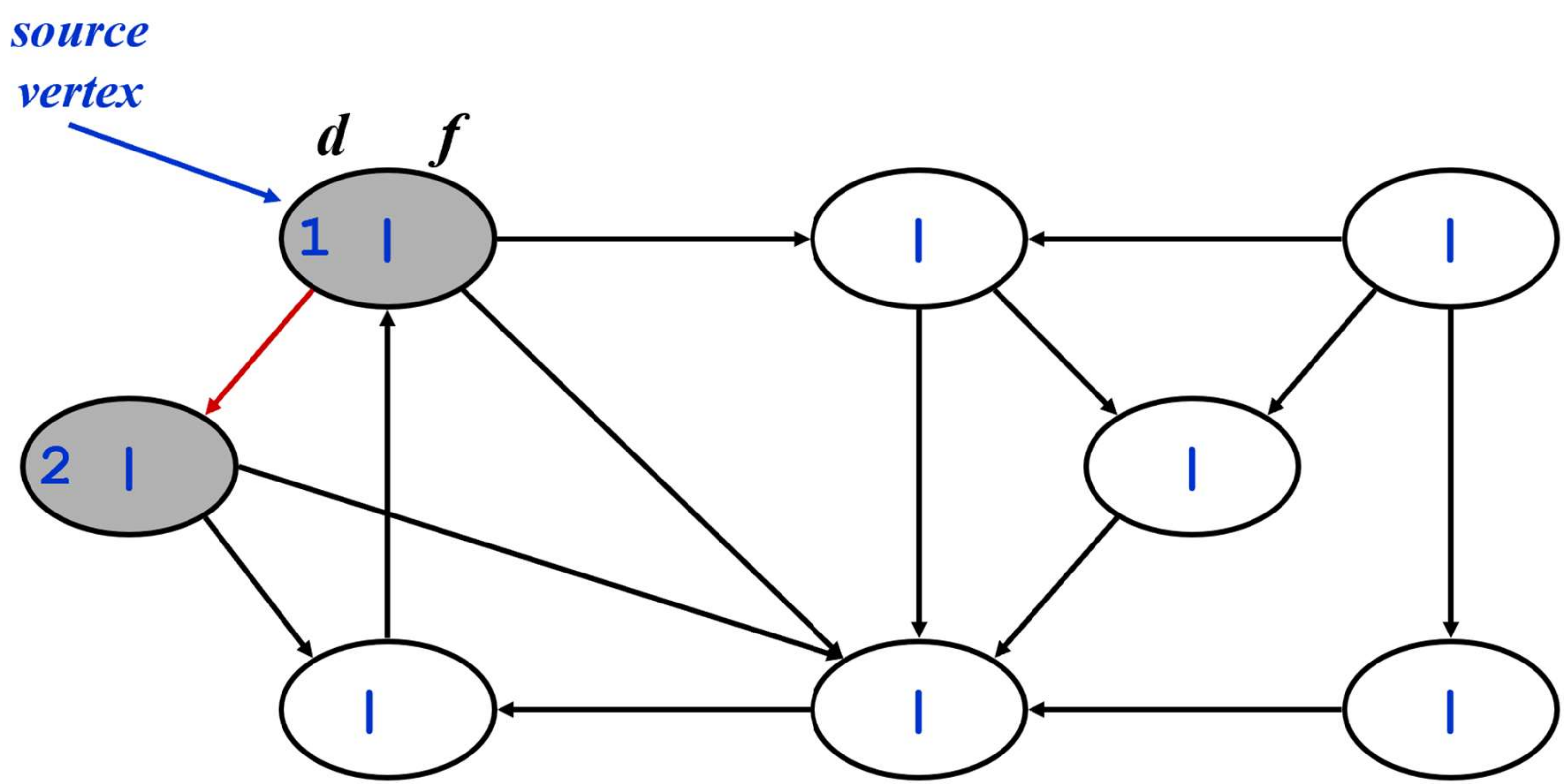
DFS Example



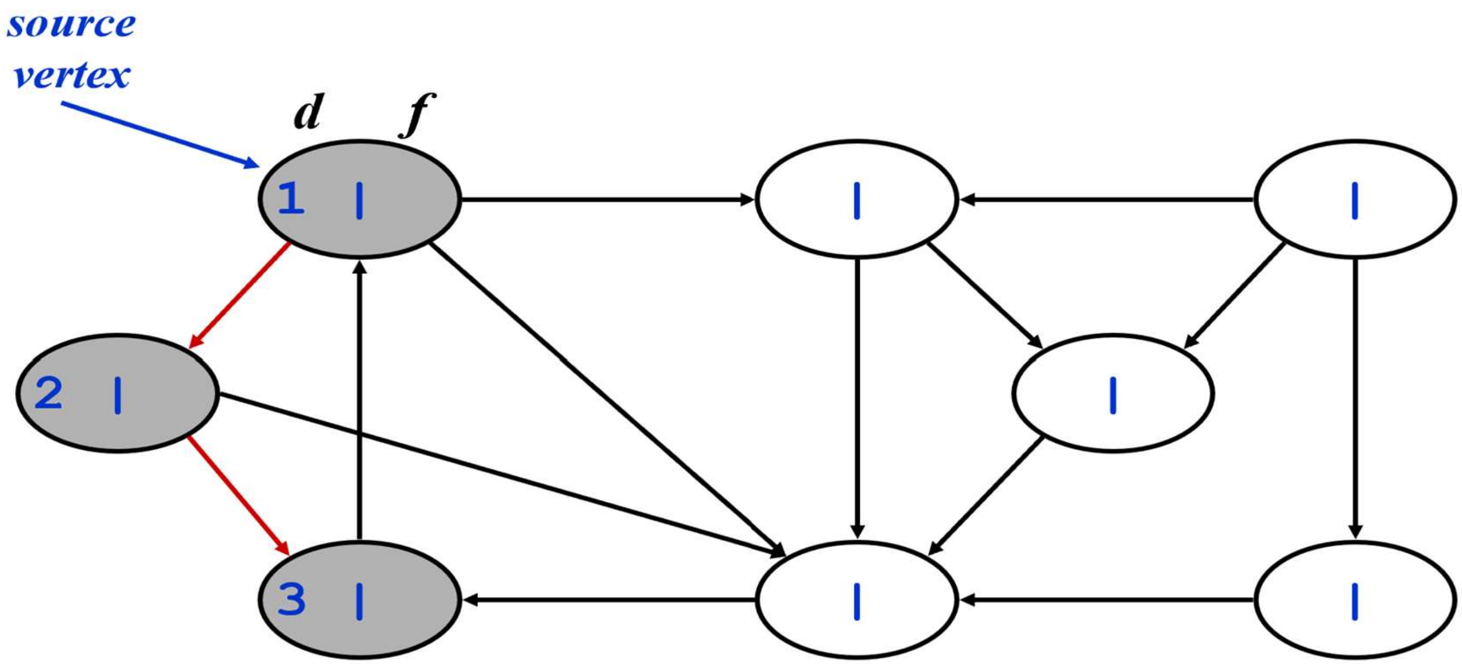
DFS Example



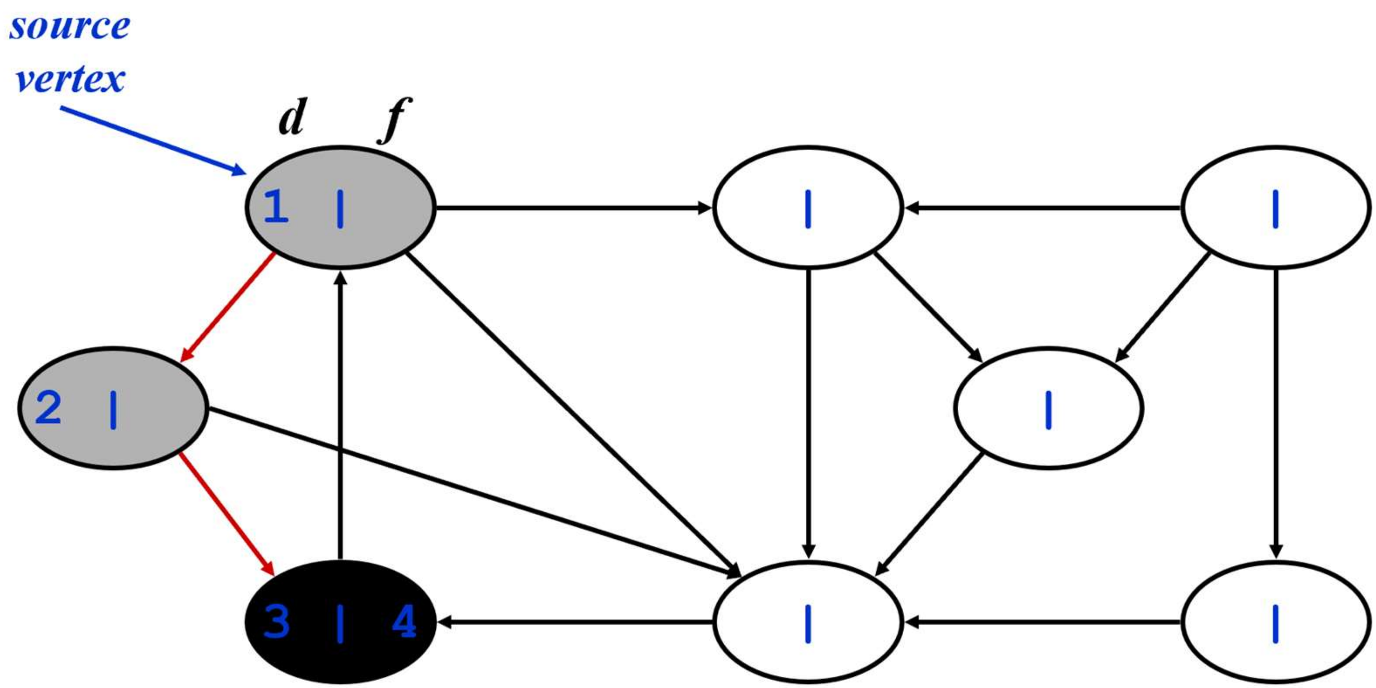
DFS Example



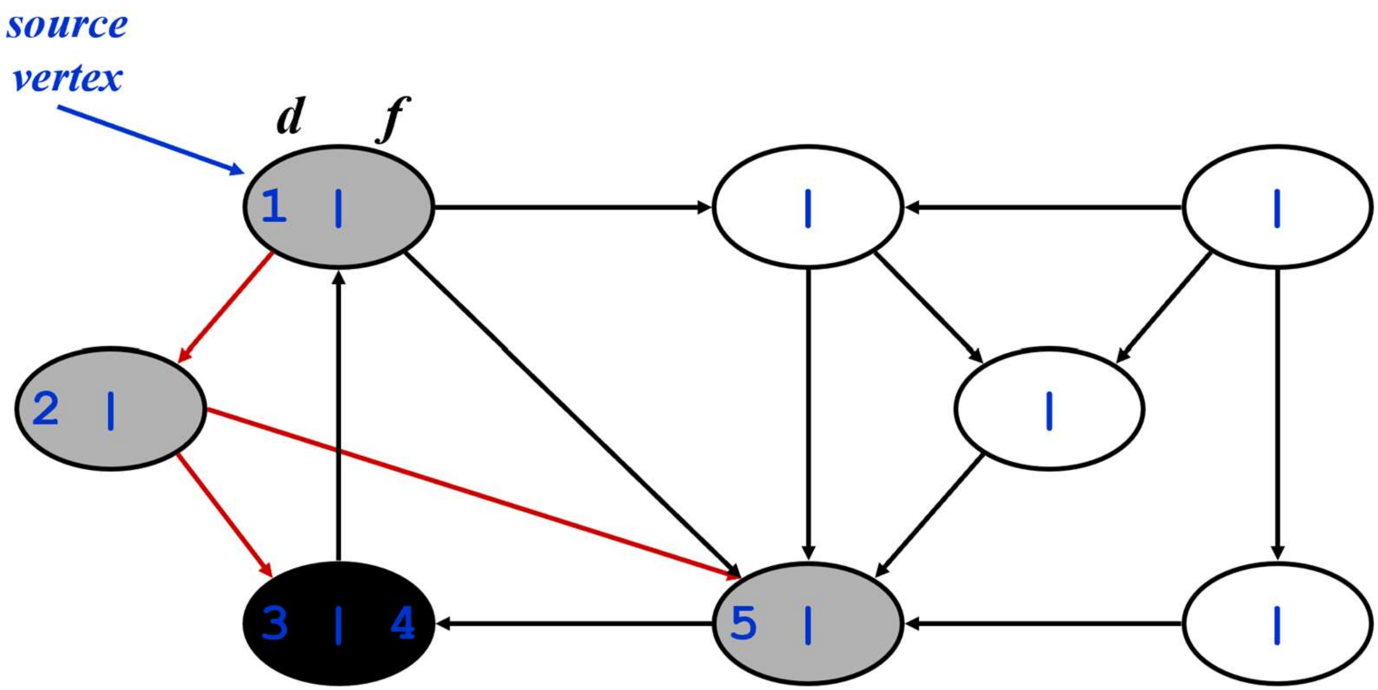
DFS Example



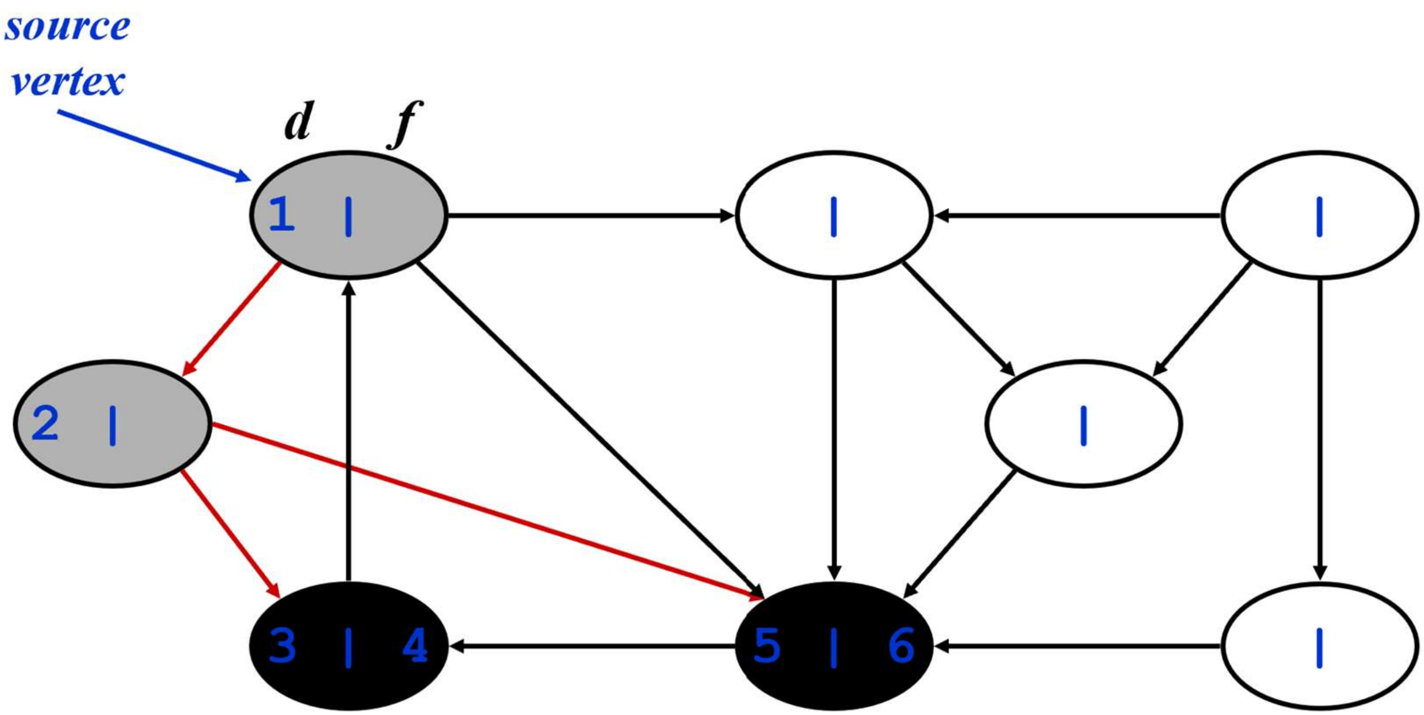
DFS Example



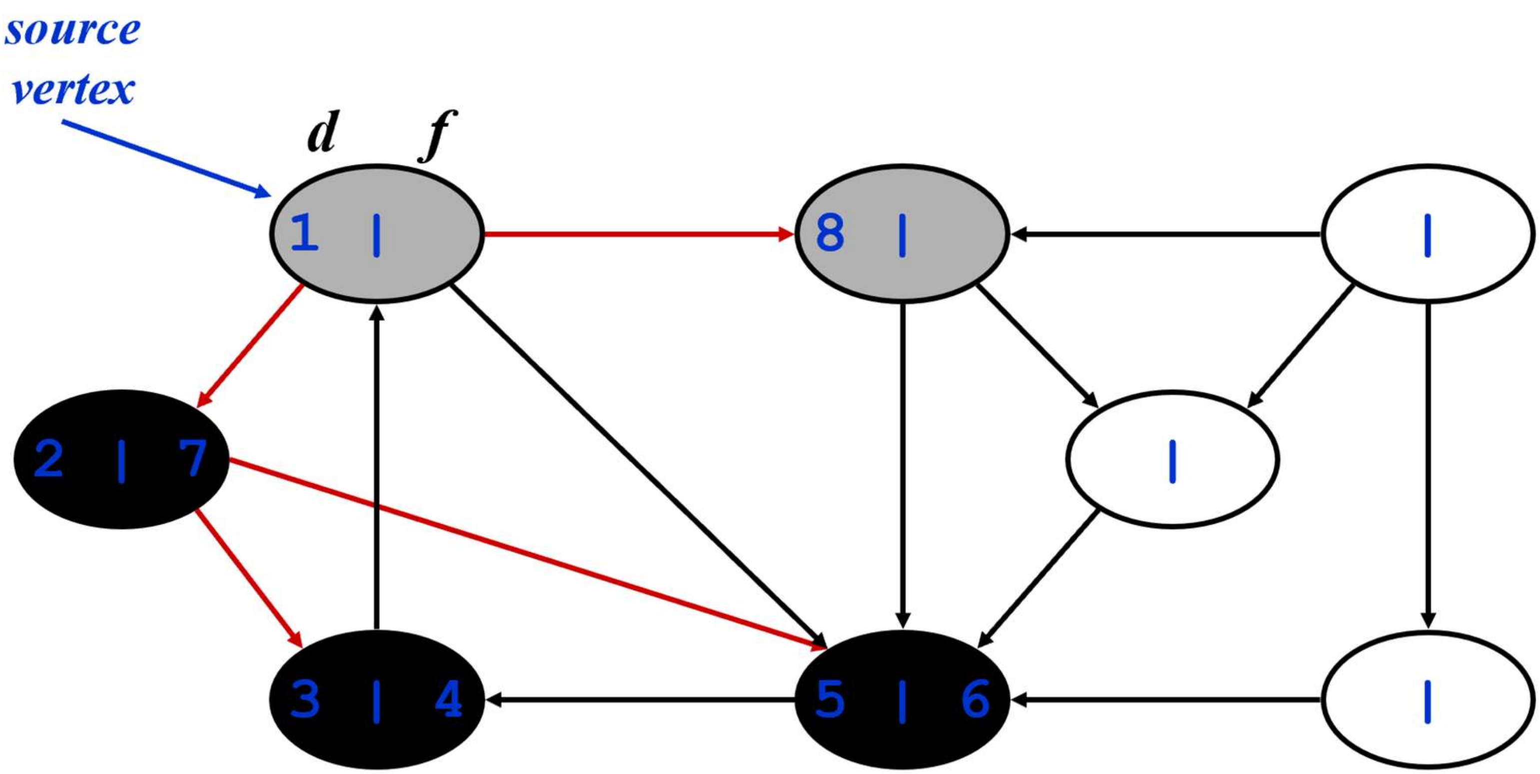
DFS Example



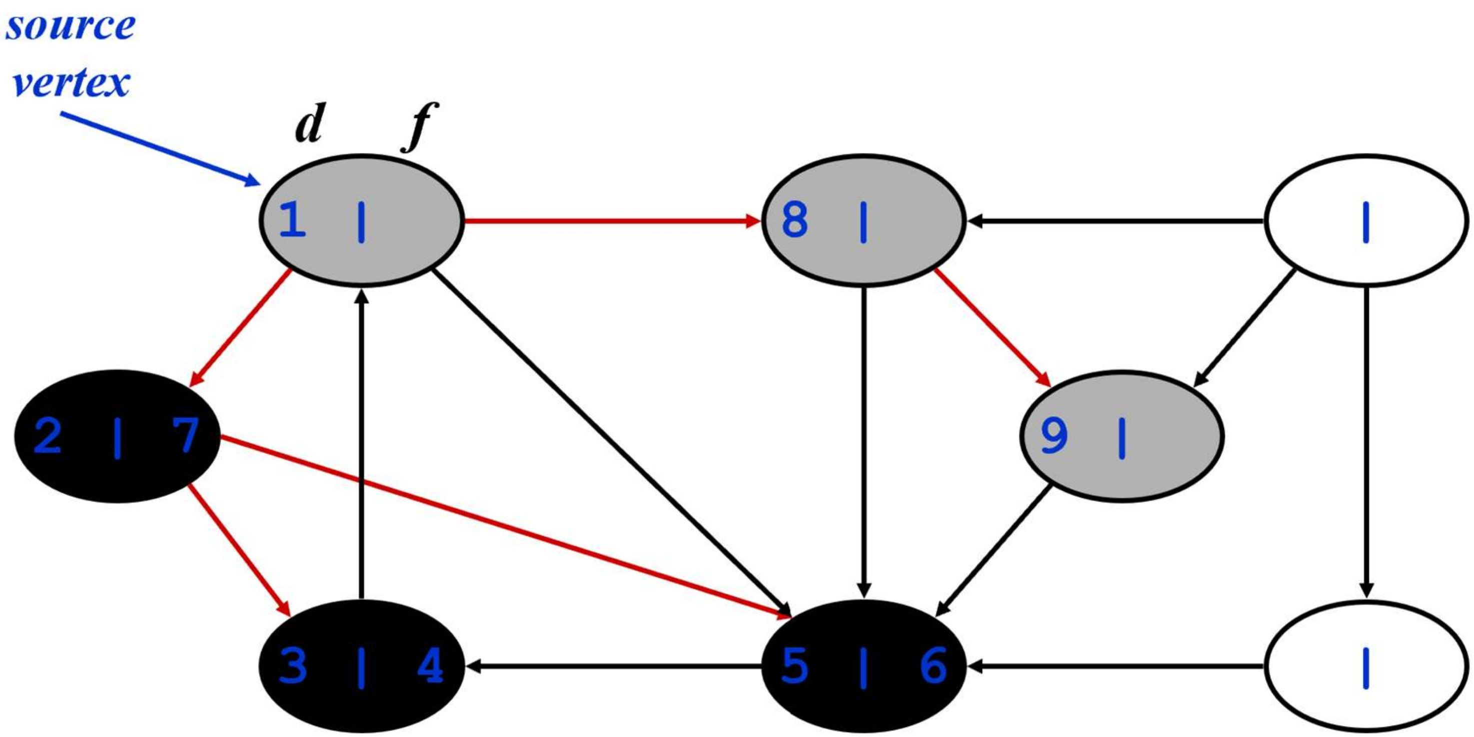
DFS Example



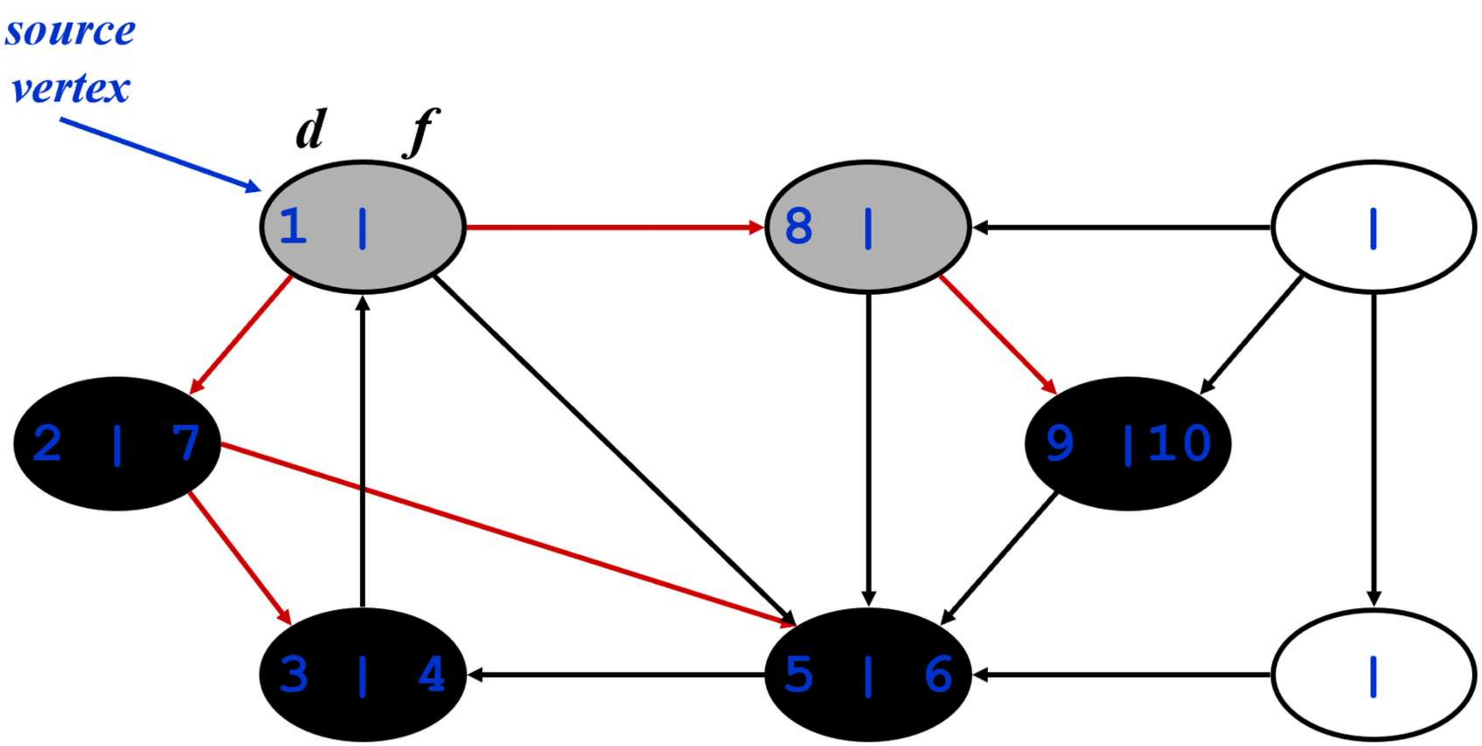
DFS Example



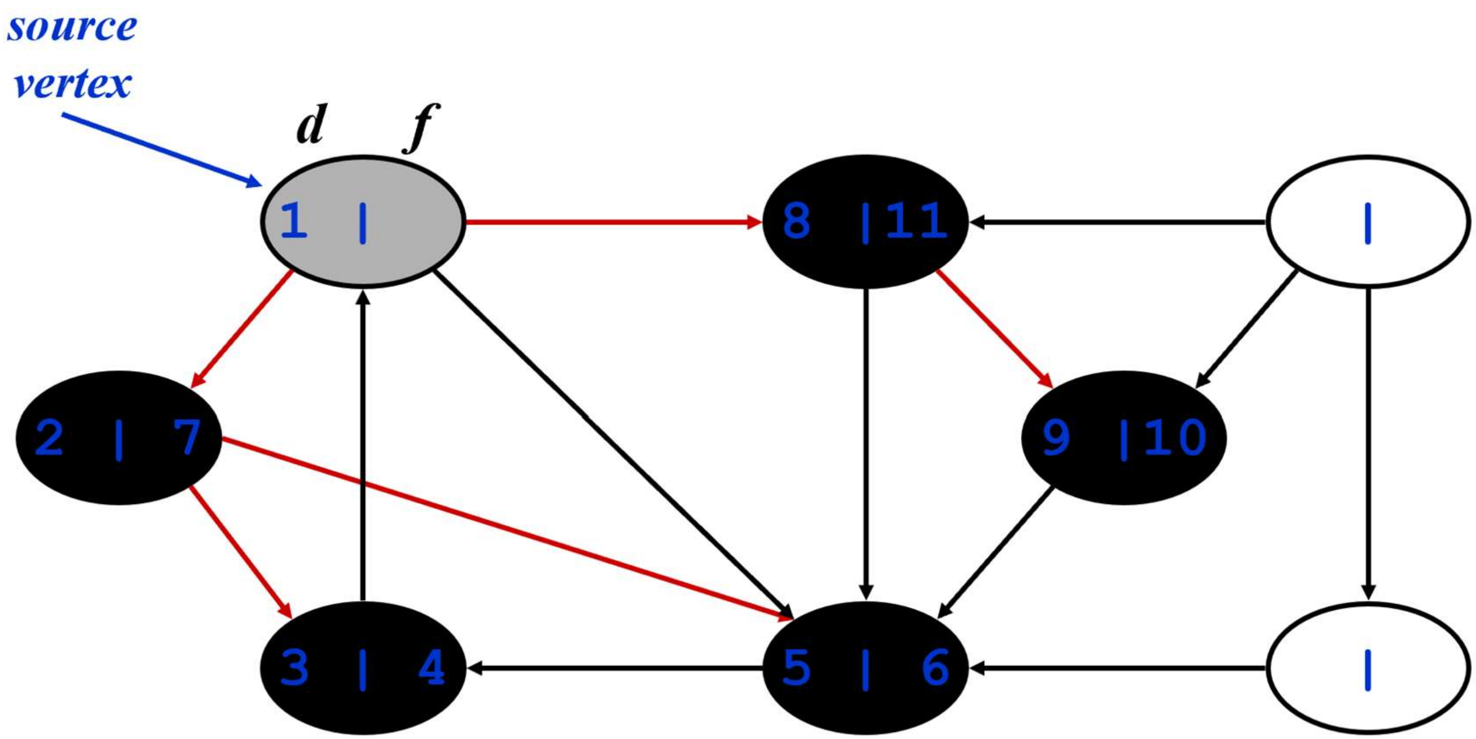
DFS Example



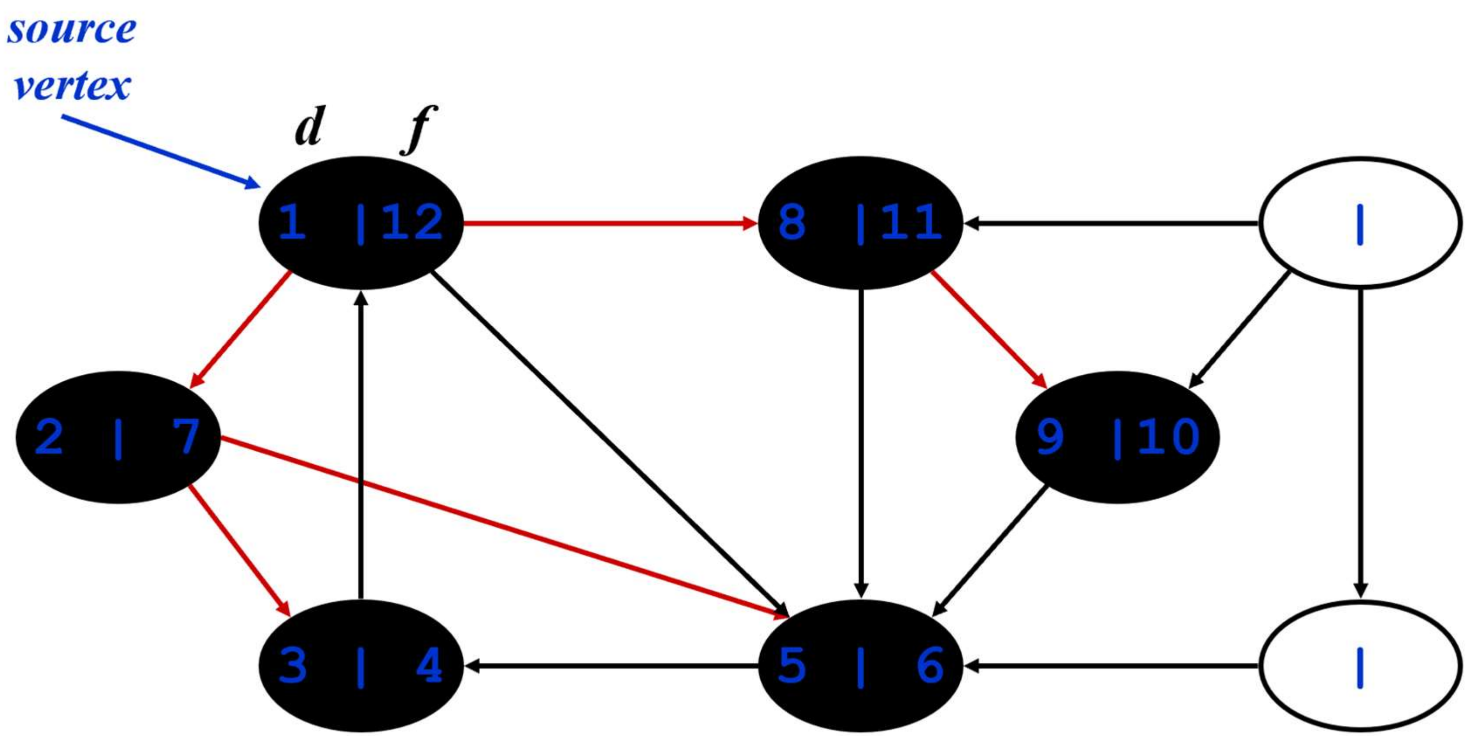
DFS Example



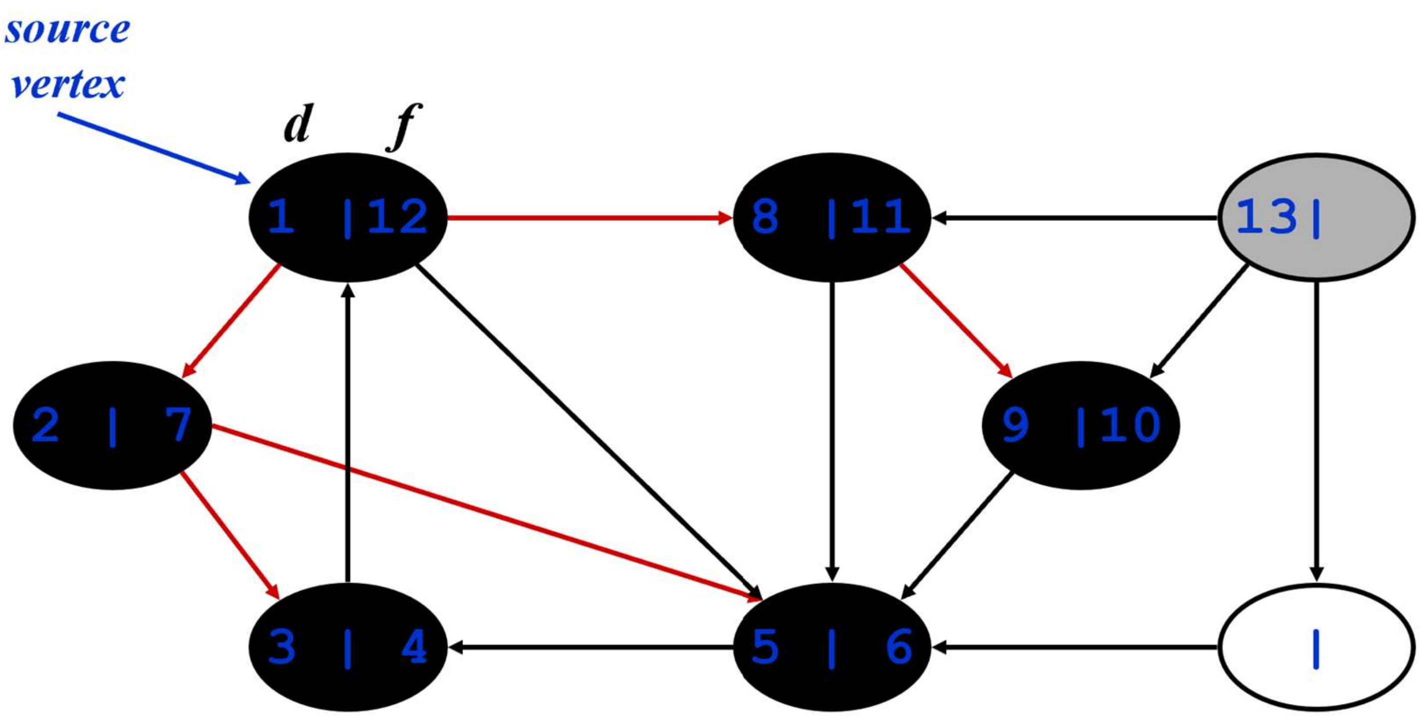
DFS Example



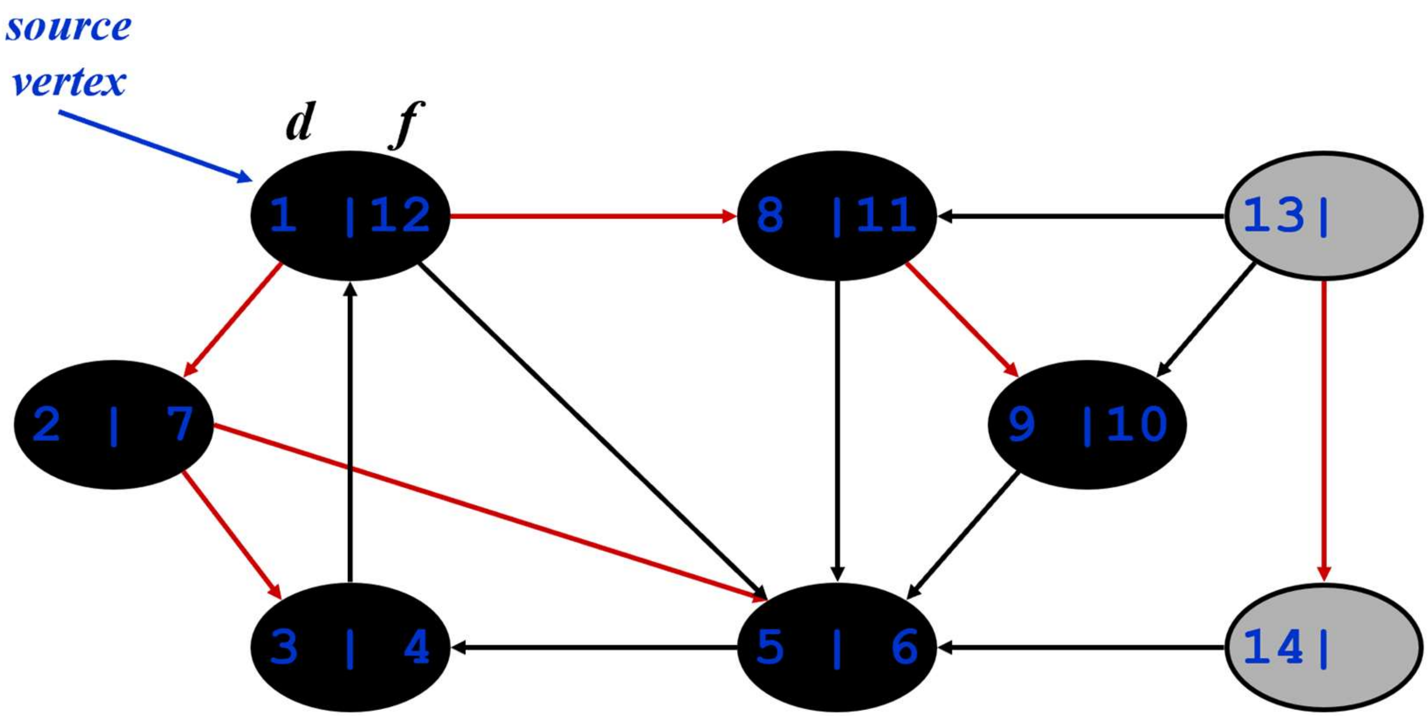
DFS Example



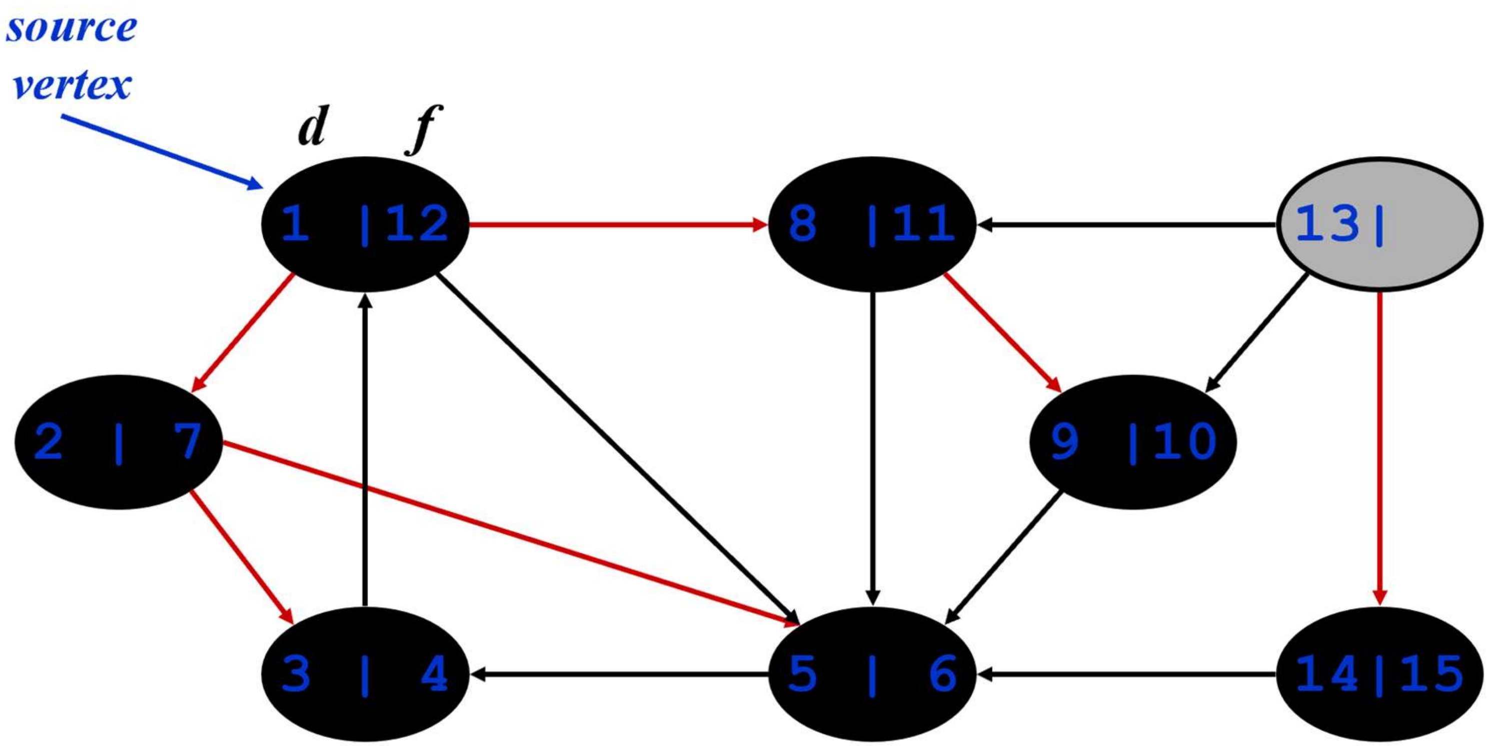
DFS Example



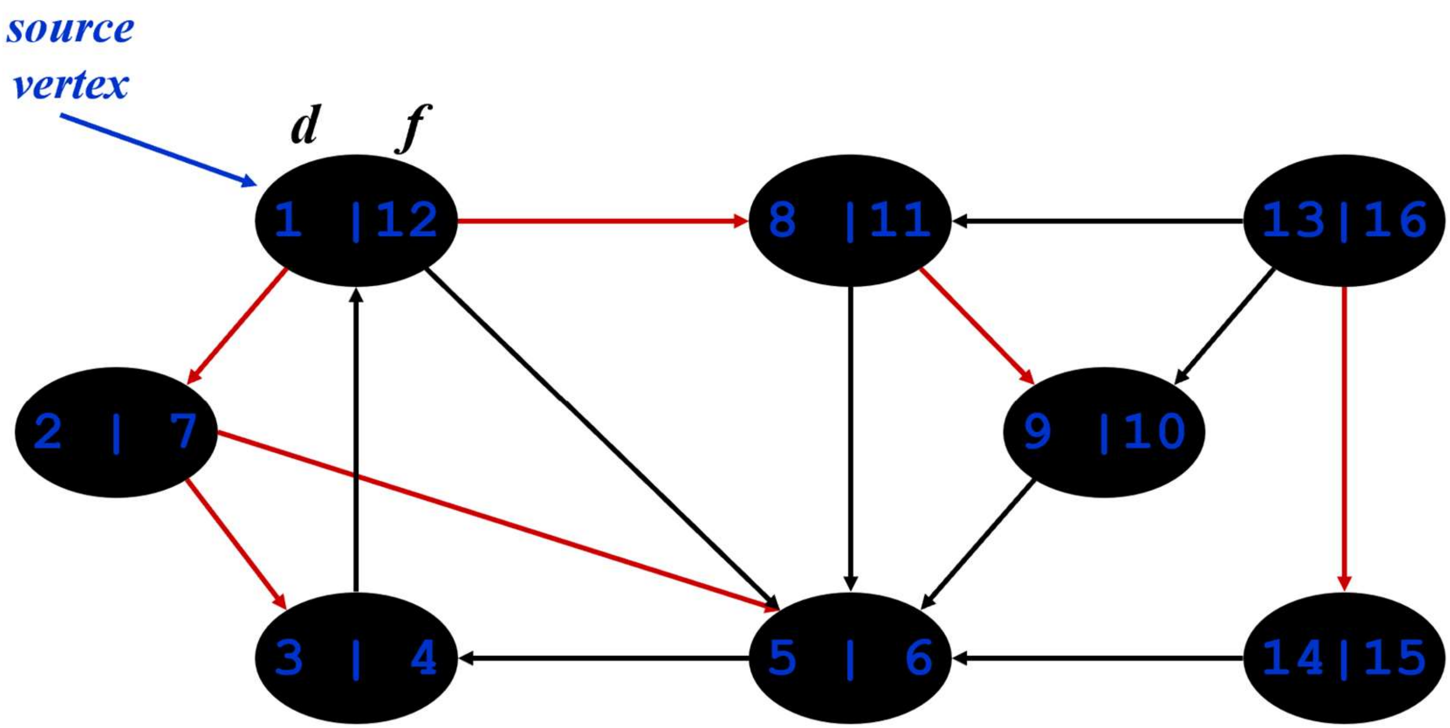
DFS Example



DFS Example



DFS Example



Depth-First Search Analysis

- This running time argument is an example of informal analysis
  - Consider the exploration of edge to the edge:
    - Each loop in **DFS_Visit** can be attributed to an edge in the graph
    - Runs once/edge if directed graph, twice if undirected
    - Thus, loop will run in $\theta(E)$ time, algorithm $\theta(V + E)$

DFS: Kinds of edges

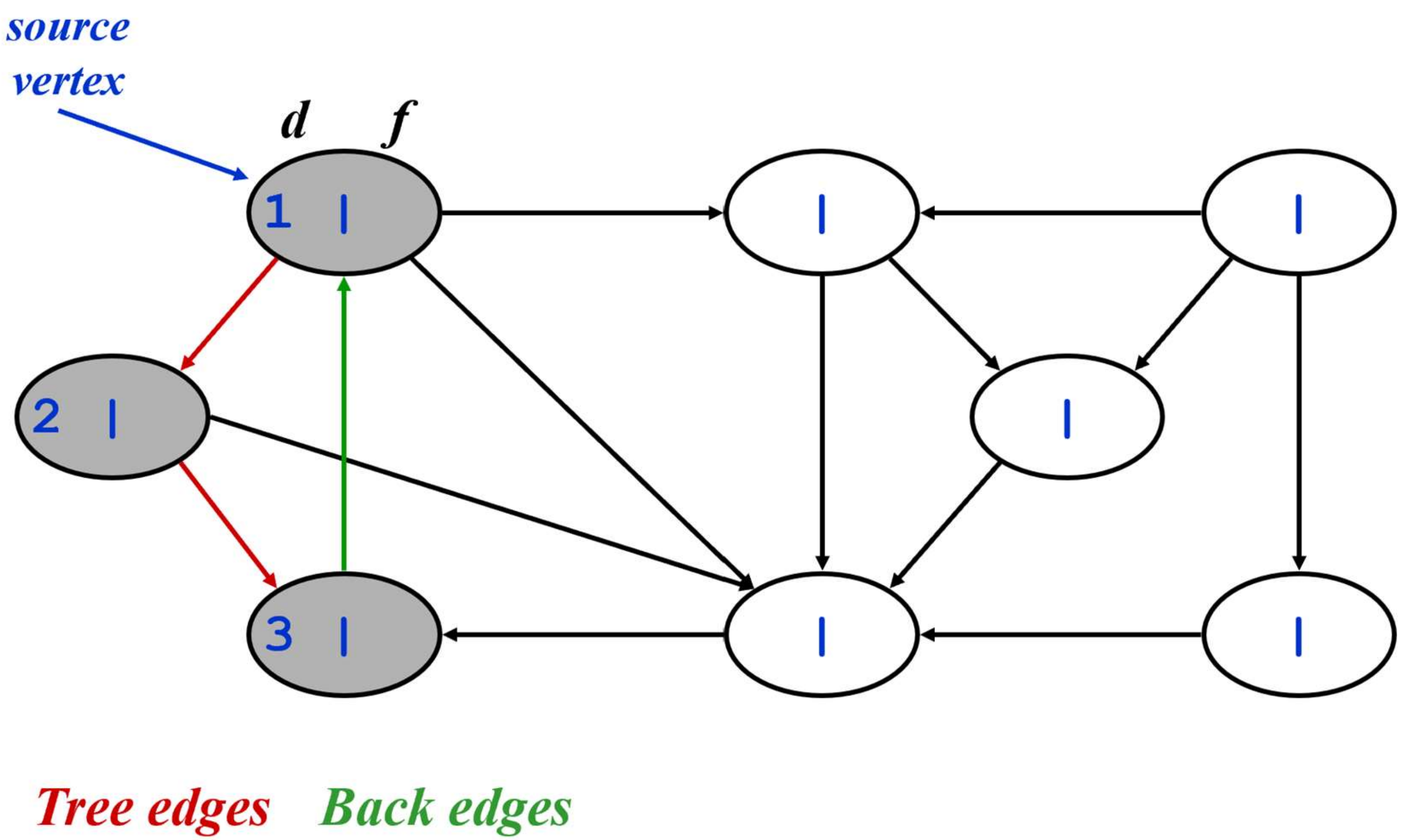
- DFS introduces an important distinction among edges in the original graph:
  - *Tree edge*: encounter new (white) vertex

    -- Can tree edges form cycles? Why or why not?

DFS: Kinds of edges

- DFS introduces an important distinction among edges in the original graph:
  - *Tree edge*: encounter new (white) vertex
  - *Back edge*: from descendent to ancestor
    - Encounter a grey vertex (grey to grey)

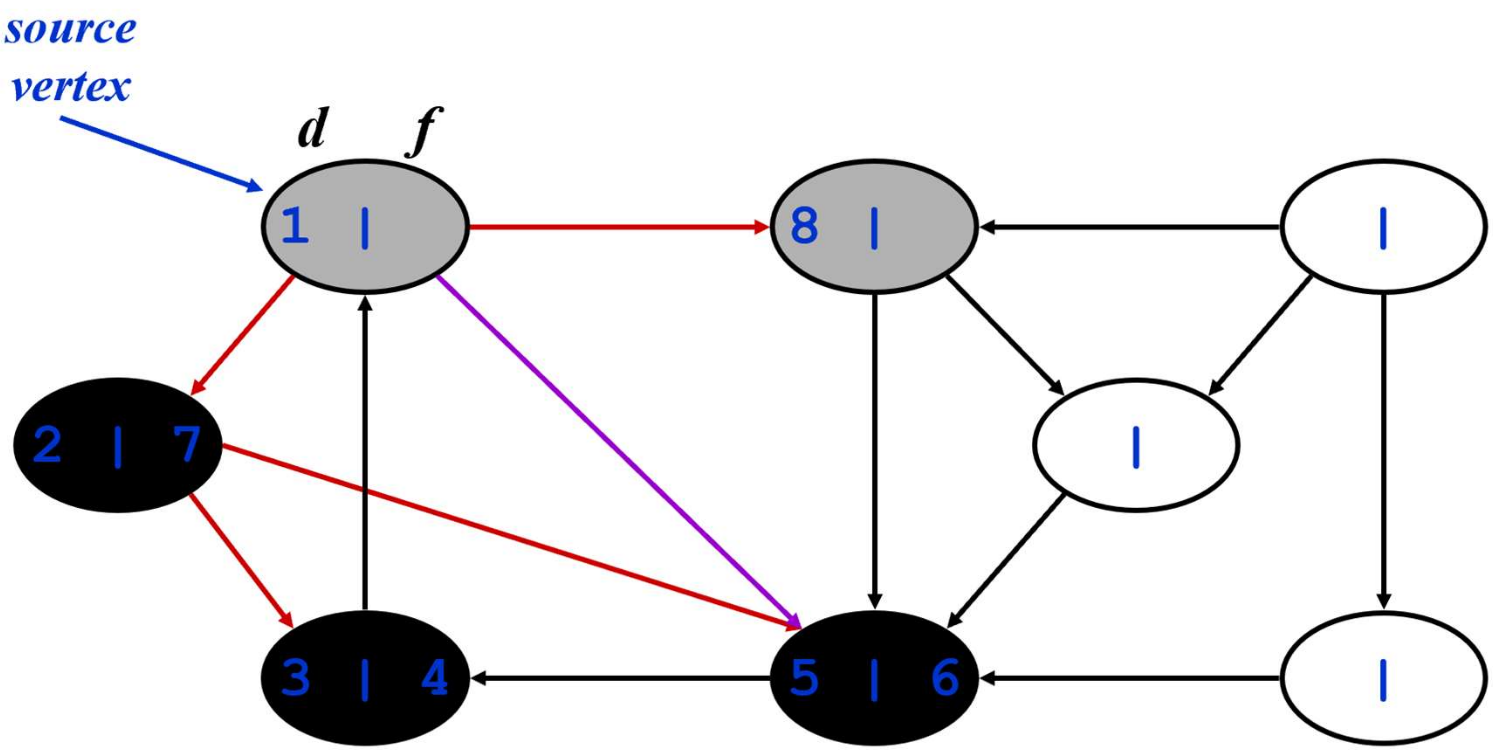
DFS: Kinds of edges



DFS: Kinds of edges

- DFS introduces an important distinction among edges in the original graph:
  - *Tree edge*: encounter new (white) vertex
  - *Back edge*: from descendent to ancestor
  - *Forward edge*: from ancestor to descendent
    - Not a tree edge, though
    - From grey node to black node

DFS: Kinds of edges

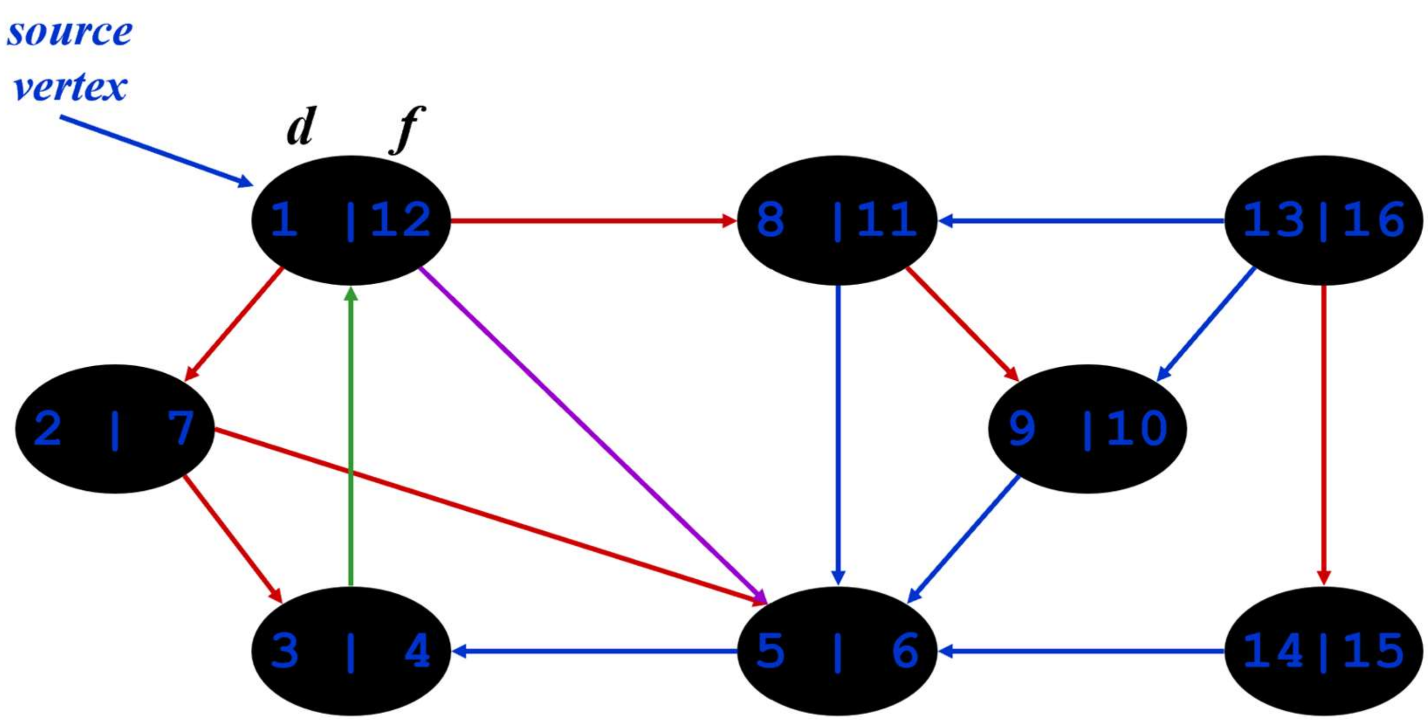


Tree edges *Back edges* *Forward edges*

DFS: Kinds of edges

- DFS introduces an important distinction among edges in the original graph:
  - *Tree edge*: encounter new (white) vertex
  - *Back edge*: from descendent to ancestor
  - *Forward edge*: from ancestor to descendent
  - *Cross edge*: between a tree or subtrees
    - From a grey node to a black node

DFS: Kinds of edges



Tree edges *Back edges* *Forward edges* *Cross edges*

DFS: Kinds of edges

- DFS introduces an important distinction among edges in the original graph:
  - *Tree edge*: encounter new (white) vertex
  - *Back edge*: from descendent to ancestor
  - *Forward edge*: from ancestor to descendent
  - *Cross edge*: between a tree or subtrees

- Note: tree and back edges are very important
  -- some algorithms use forward and cross edges

DFS: Kinds of edges

- In a DFS of an undirected graph G, every edge is either a tree edge or a back edge.

  Proof: Theorem 22.10

- A directed graph G is *acyclic* if and only if a DFS of G yields no back edges.

  Proof: Theorem 22.11

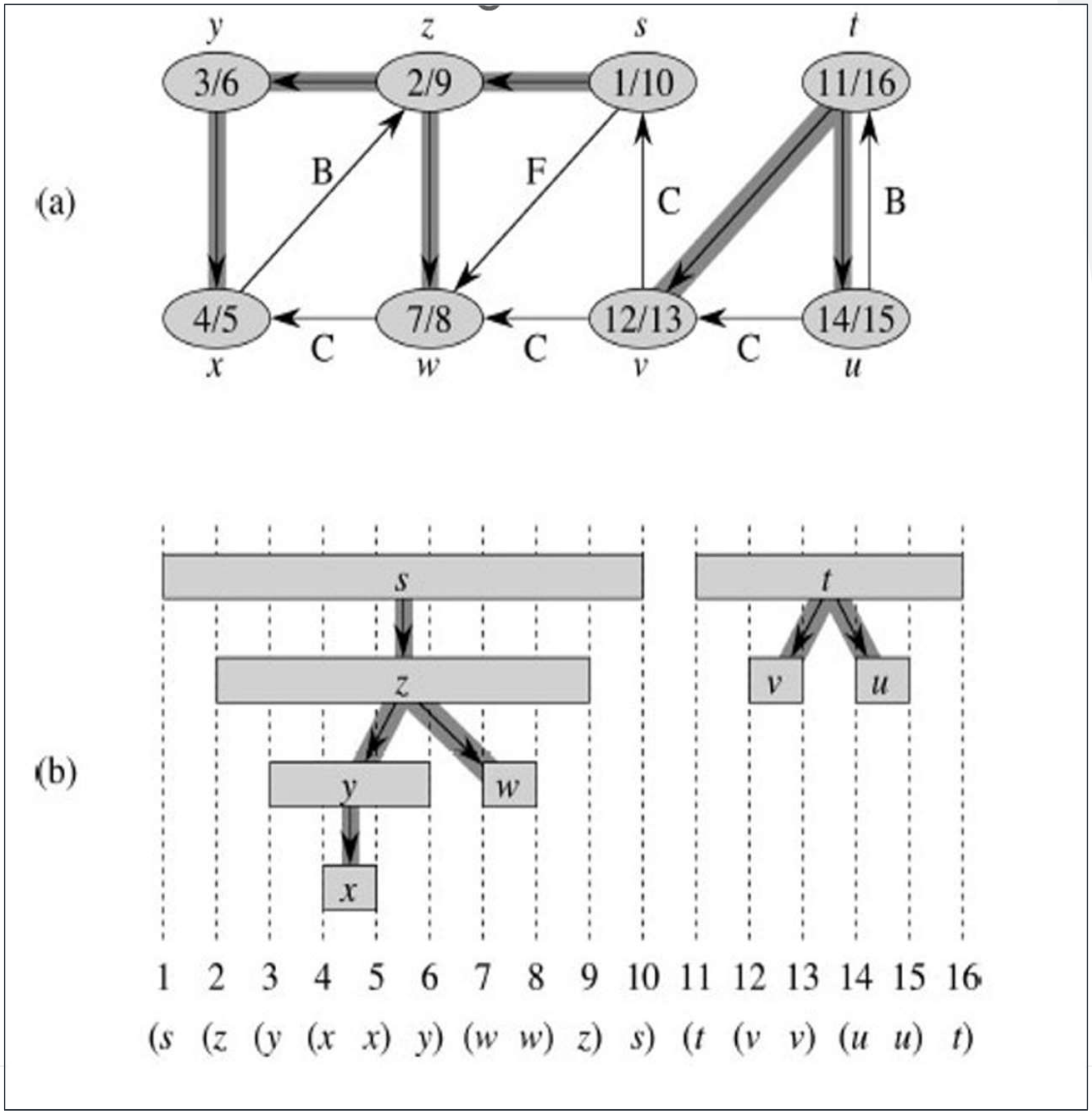
- DFS can be utilized to find cycles

Properties of DFS: Parenthesis Structure

- The discovery and finishing times of vertices have parenthesis structure.

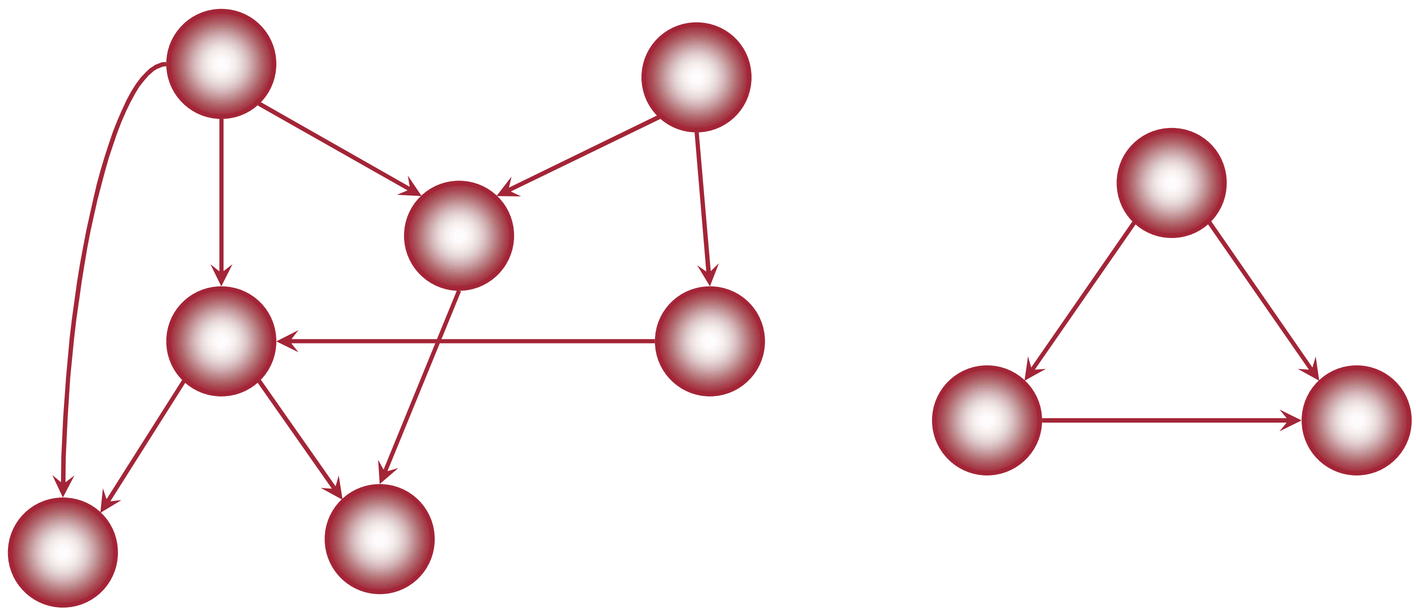
- In any DFS of a graph G, for any two vertices u and v, one of the following three conditions holds: [Theorem 22.7]
  - The intervals $[d(u), f(u)]$ and $[d(v), f(v)]$ are entirely disjoint, and neither u nor v is a descendent of the other in the DFS forest,
  - The interval $[d(u), f(u)]$ is contained entirely within the interval $[d(v), f(v)]$, and u is a descendent of v in a DFS tree, or
  - The interval $[d(v), f(v)]$ is contained entirely within the interval $[d(u), f(u)]$, and v is a descendent of u in a DFS tree.

Properties of DFS: Parenthesis Structure



Directed Acyclic Graphs

- A *directed acyclic graph* or *DAG* is a directed graph with no directed cycles



Topological Sort

- A *topological sort* of a DAG is
  - A linear ordering of all vertices of the graph G such that vertex u comes before vertex v if (u, v) is an edge in G.

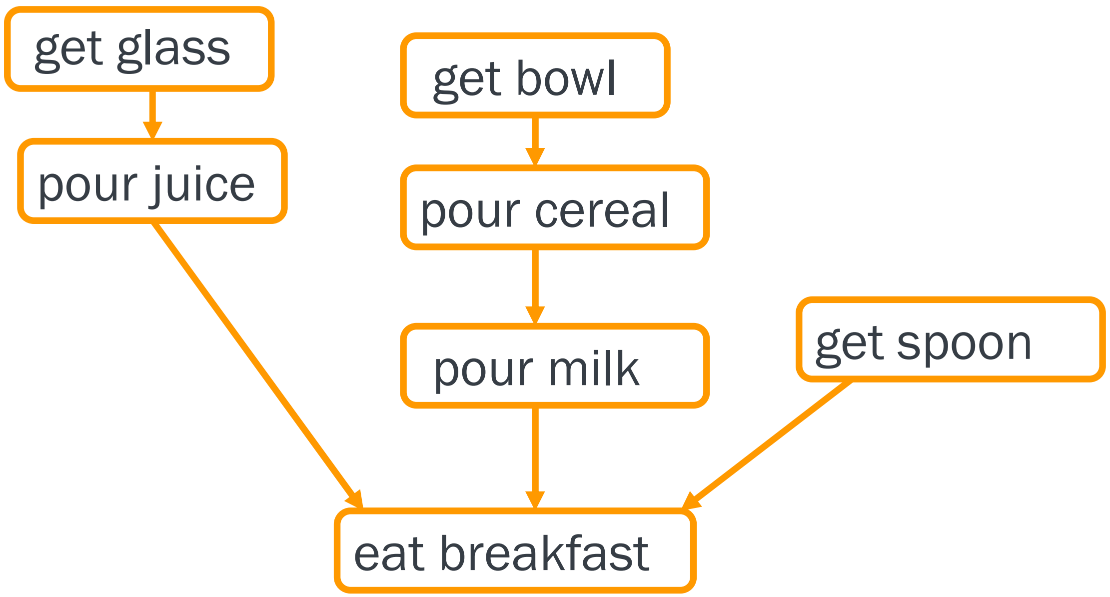
- DAG indicates precedence among events:
  - Events are graph vertices, edge from u to v means event u has precedence over event v

- Real-world example:
  - Getting dressed
  - Course registration
  - Tasks for eating meal

Precedence Example

- Tasks that have to be done to eat breakfast:
  - get glass, pour juice, get bowl, pour cereal, pour milk, get spoon, eat.
- Certain events must happen in a certain order (ex: get bowl before pouring milk)
- For other events, it doesn't matter (ex: get bowl and get spoon)

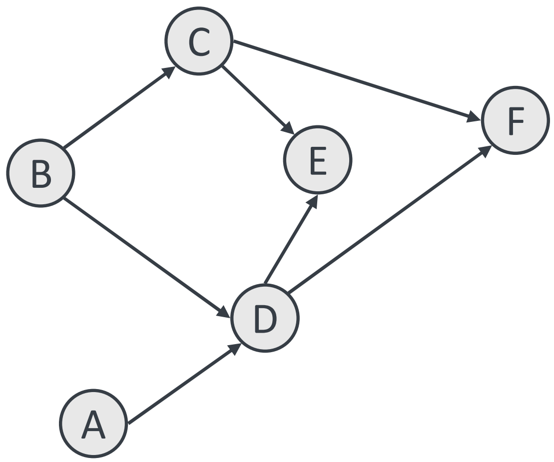
Precedence Example



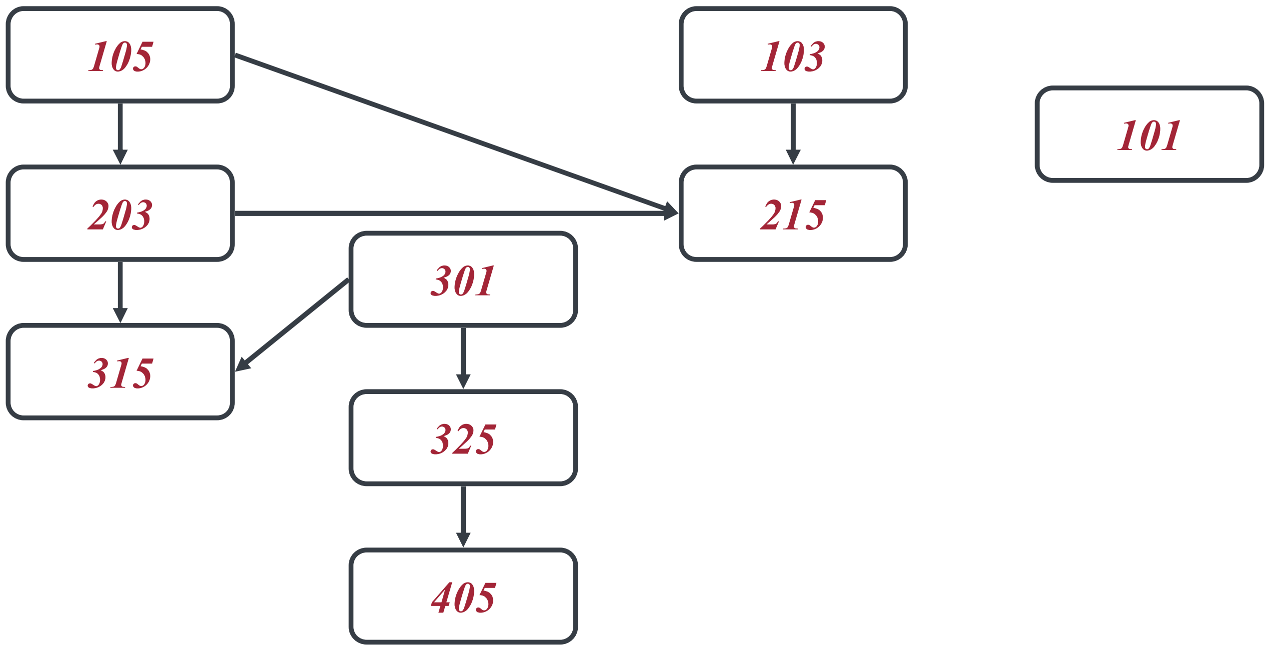
Order: glass, juice, bowl, cereal, milk, spoon, eat.

More Example …

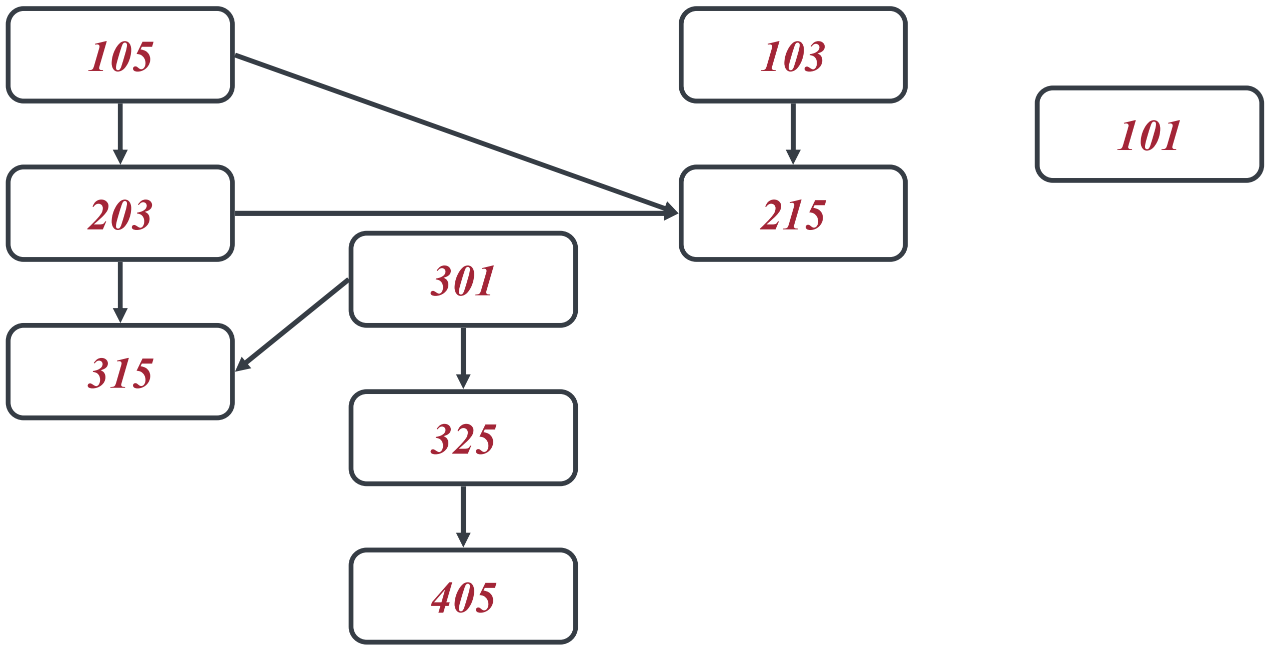
- How many valid topological sort orderings can you find for the vertices in the graph below?
  - [A, B, C, D, E, F]
  - [A, B, C, D, F, E]
  - [A, B, D, C, E, F]
  - [A, B, D, C, F, E]
  - [B, A, C, D, E, F]
  - [B, A, C, D, F, E]
  - [B, A, D, C, E, F]
  - [B, A, D, C, F, E]
  - [B, C, A, D, E, F]
  - [B, C, A, D, F, E]
  - ....



Another Example: Course Registration



Another Example: Course Registration



Why Acyclic?

- Why must directed graph be acyclic for the topological sort problem?
- Otherwise, no way to order events linearly without violating a precedence constraint.

Topological Sort Algorithm

TOPOLOGICAL-SORT(G)

1 call DFS(G) to compute finishing times $v.f$ for each vertex v
2 as each vertex is finished, insert it onto the front of a linked list
3 **return** the linked list of vertices

- Time Complexity: $\theta(V + E)$

Contents of this presentation are based on
Book Chapter- 22, Introduction to Algorithms by *Cormen, Leiserson, Rivest,* & *Stein*

STEVENS
INSTITUTE OF TECHNOLOGY
1870

THANK **YOU**

Stevens Institute of Technology