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

Depth-first Search (DFS)

- Explore edges out of the most recently discovered vertex v.
- When all edges of v have been explored, backtrack to explore other edges leaving the vertex from which v was discovered (its *predecessor*).
- "Search as deep as possible first."
- Continue until all vertices reachable from the original source are discovered.
- If any undiscovered vertices remain, then one of them is chosen as a new source and search is repeated from that source.

Depth-first Search

- **Input:** $G = (V, E)$, directed or undirected. No source vertex given!
- **Output:**
  - 2 **timestamps** on each vertex. Integers between 1 and 2|V|.
    - $d[v]$ = ***discovery time*** (v turns from white to gray)
    - $f[v]$ = ***finishing time*** (v turns from gray to black)
  - $\pi[v]$: predecessor of $v = u$, such that v was discovered during the scan of u's adjacency list.
- Uses the same coloring scheme for vertices as BFS.

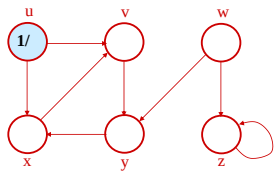
Pseudo-code

```
DFS(G)
1. for each vertex u ∈ V[G]
2.     do color[u] ← white
3.        π[u] ← NIL
4. time ← 0
5. for each vertex u ∈ V[G]
6.     do if color[u] = white
7.           then DFS-Visit(u)
```

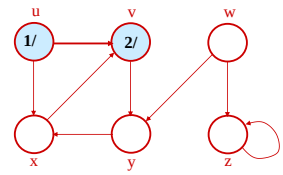
Uses a global timestamp **time**.

```
DFS-Visit(u)
1.  color[u] ← GRAY  ∇ White vertex u has been discovered
2.  time ← time + 1
3.  d[u] ← time
4.  for each v ∈ Adj[u]
5.      do if color[v] = WHITE
6.            then π[v] ← u
7.                 DFS-Visit(v)
8.  color[u] ← BLACK   ∇ Blacken u; it is finished.
9.  f[u] ← time ← time + 1
```

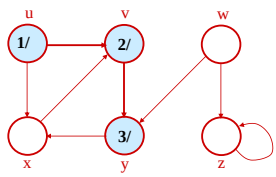
Example (DFS)



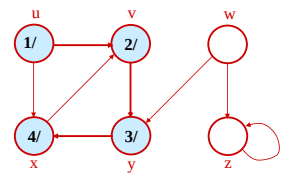
Example (DFS)



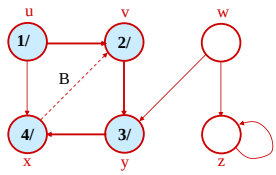
Example (DFS)



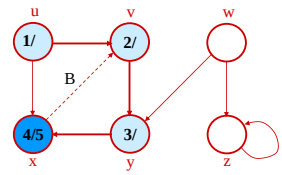
Example (DFS)



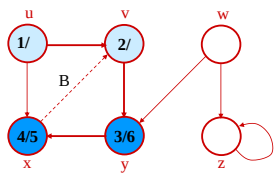
Example (DFS)



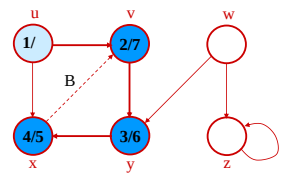
Example (DFS)



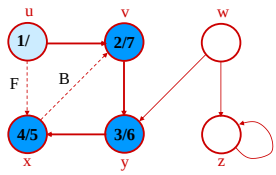
Example (DFS)



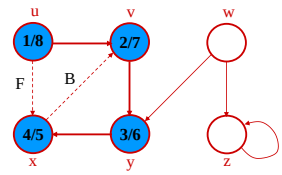
Example (DFS)



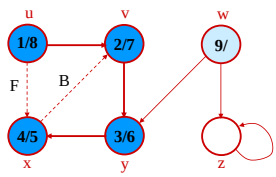
Example (DFS)



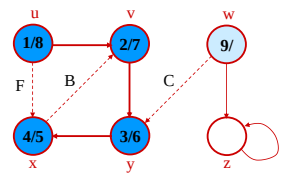
Example (DFS)



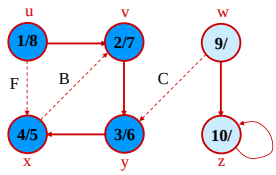
Example (DFS)



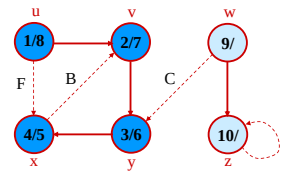
Example (DFS)



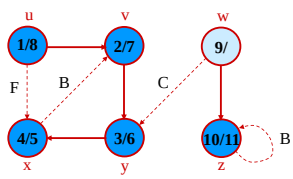
Example (DFS)



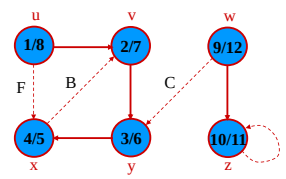
Example (DFS)



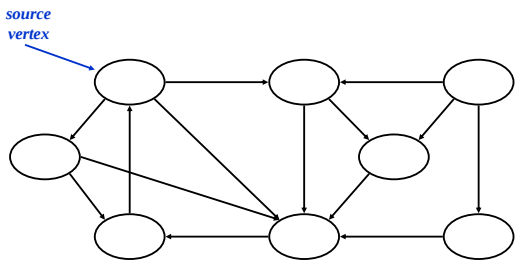
Example (DFS)



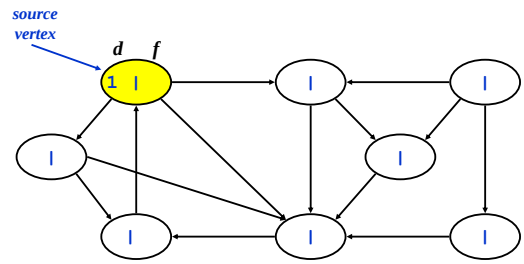
Example (DFS)



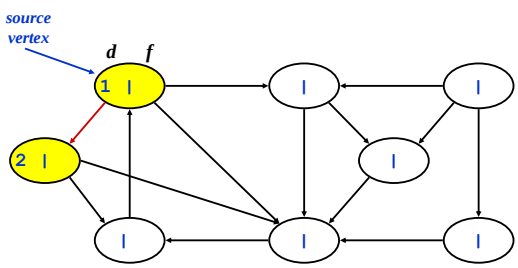
Another DFS Example



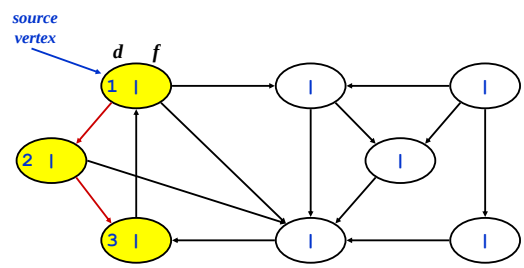
DFS Example



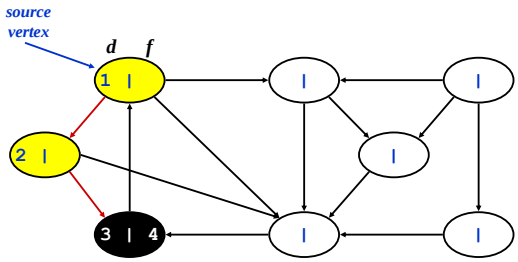
DFS Example



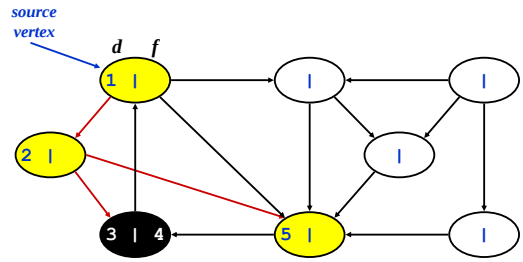
DFS Example



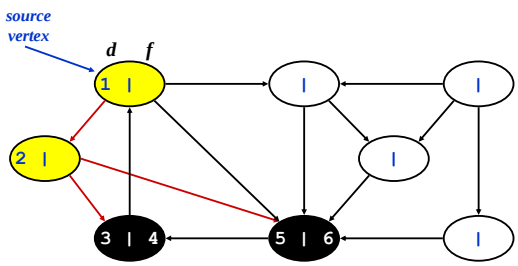
DFS Example



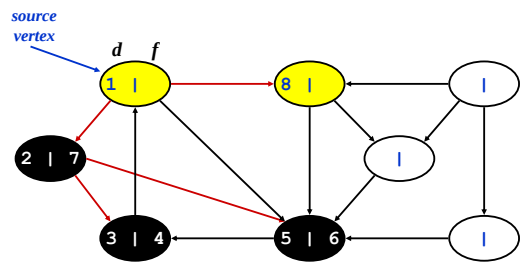
DFS Example



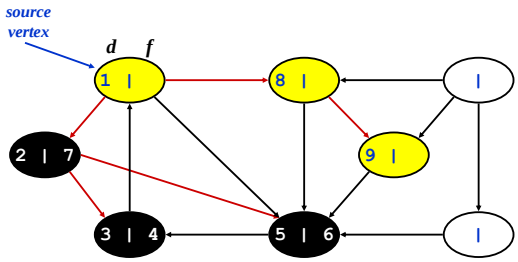
DFS Example



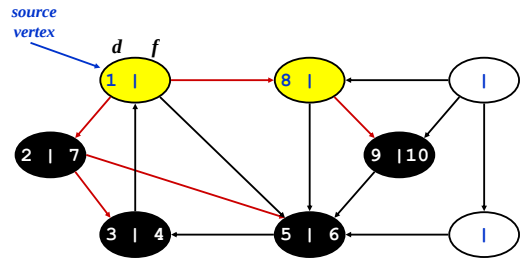
DFS Example



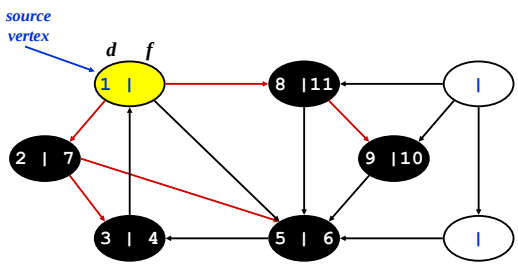
DFS Example



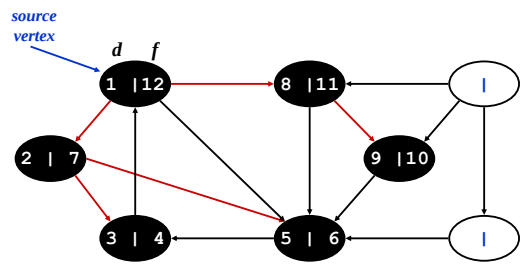
DFS Example



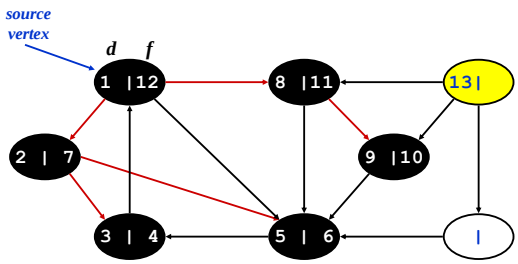
DFS Example



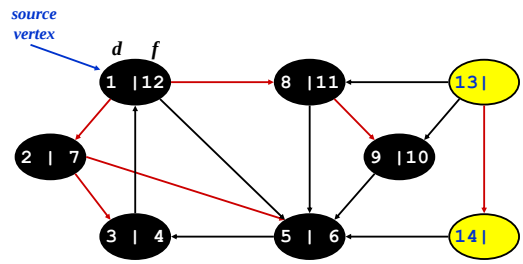
DFS Example



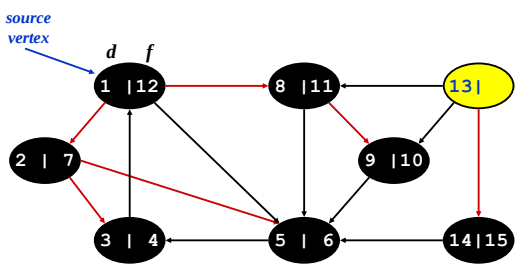
DFS Example



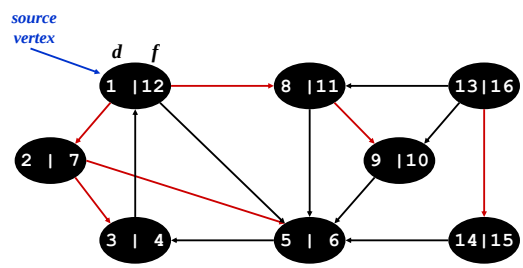
DFS Example



DFS Example



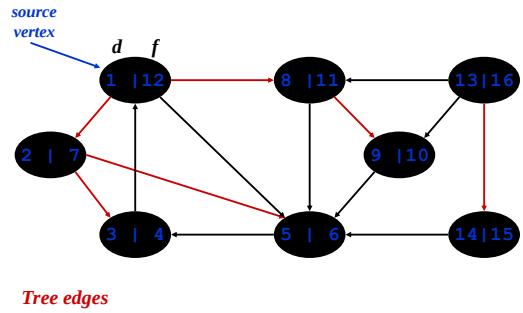
DFS Example



DFS: Kinds of edges

- DFS introduces an important distinction among edges in the original graph:
  - *Tree edge*: encounter new (white) vertex
    - The tree edges form a spanning forest
    - *Can tree edges form cycles? Why or why not?*

DFS Example

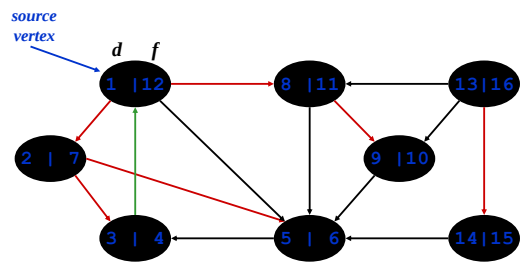


Tree edges

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
    - Encounter a yellow vertex (yellow to yellow)

DFS Example

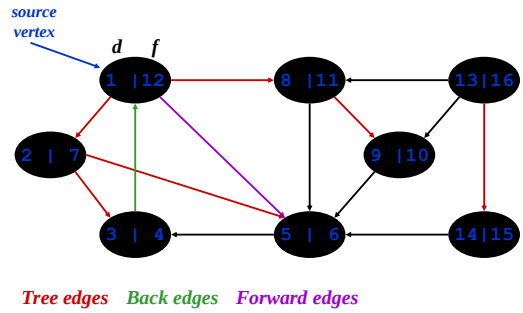


Tree edges *Back edges*

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 yellow node to black node

DFS Example

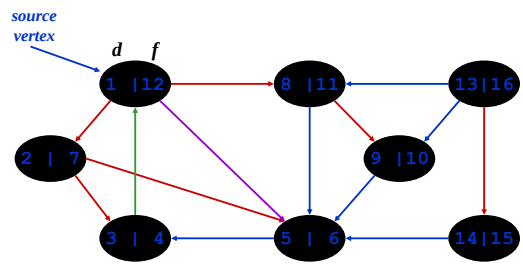


Tree edges *Back edges* *Forward edges*

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 yellow node to a black node

DFS Example



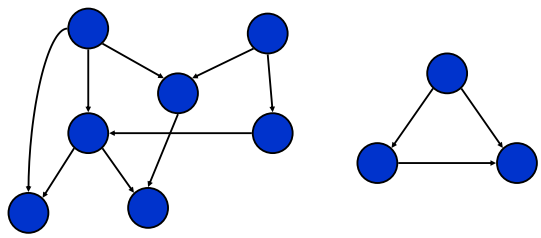
Tree edges *Back edges* *Forward edges* *Cross edges*

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 & back edges are important; most algorithms don't distinguish forward & cross

Directed Acyclic Graphs

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



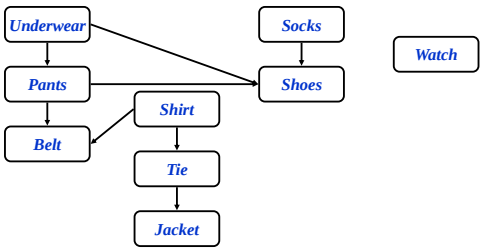
DFS and DAGs

- Argue that a directed graph G is acyclic iff a DFS of G yields no back edges:
  - Forward: if G is acyclic, will be no back edges
    - Trivial: a back edge implies a cycle
  - Backward: if no back edges, G is acyclic
    - Argue contrapositive: G has a cycle $\Rightarrow \exists$ a back edge
      - Let v be the vertex on the cycle first discovered, and u be the predecessor of v on the cycle
      - When v discovered, whole cycle is white
      - Must visit everything reachable from v before returning from DFS-Visit()
      - So path from u→v is yellow→yellow, thus (u, v) is a back edge

Topological Sort

- *Topological sort* of a DAG:
  - Linear ordering of all vertices in graph G such that vertex u comes before vertex v if edge $(u, v) \in$ G
- Real-world example: getting dressed

Getting Dressed



Getting Dressed