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

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

- Can be implemented using stack or recursion

Depth-First Search: The Code

```
DFS(G)
{
    for each vertex u ∈ G->V
    {
        u->color = WHITE;
    }
    time = 0;
    for each vertex u ∈ G->V
    {
        if (u->color == WHITE)
            DFS_Visit(u);
    }
}
```

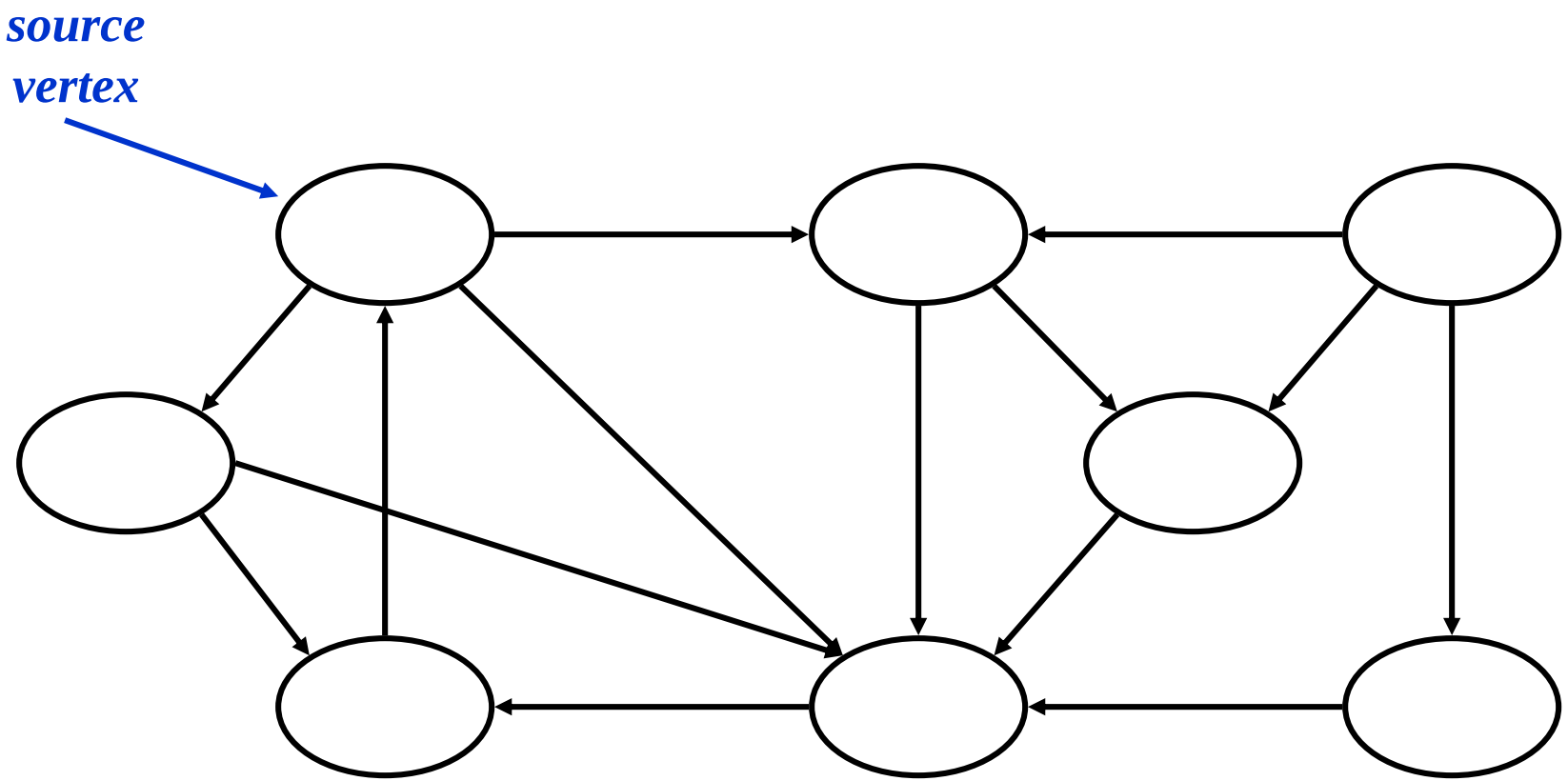
```
DFS_Visit(u)
{
    u->color = GREY;
    time = time+1;
    u->d = time;
    for each v ∈ u->Adj[]
    {
        if (v->color == WHITE)
            DFS_Visit(v);
    }
    u->color = BLACK;
    time = time+1;
    u->f = time;
}
```

Depth-First Search: The Code

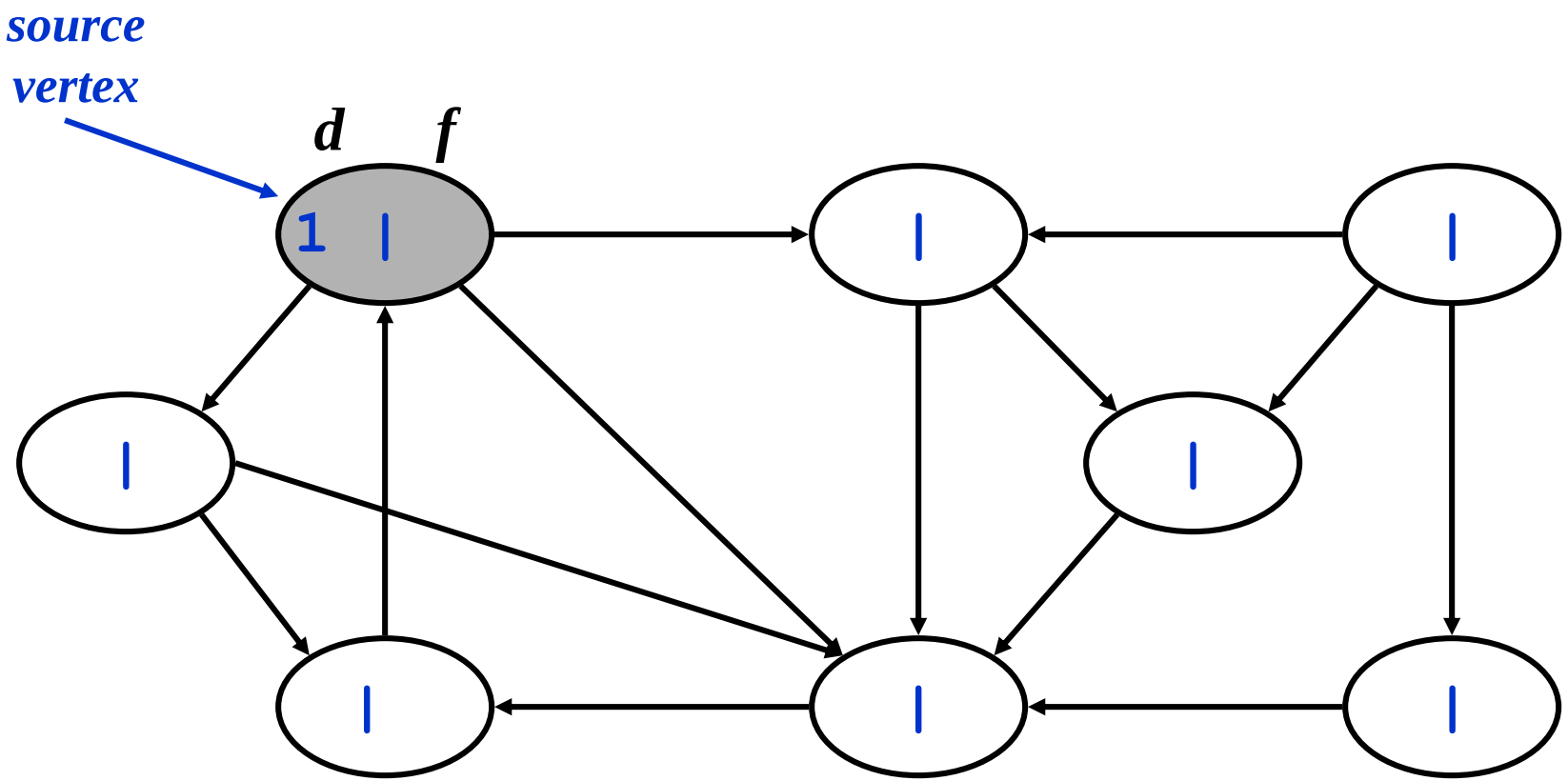
```
DFS(G)
{
    for each vertex u ∈ G->V
    {
        u->color = WHITE;
    }
    time = 0;
    for each vertex u ∈ G->V
    {
        if (u->color == WHITE)
            DFS_Visit(u);
    }
}
```

```
DFS_Visit(u)
{
    u->color = GREY;
    time = time+1;
    u->d = time;
    for each v ∈ u->Adj[]
    {
        if (v->color == WHITE)
            DFS_Visit(v);
    }
    u->color = BLACK;
    time = time+1;
    u->f = time;
}
```

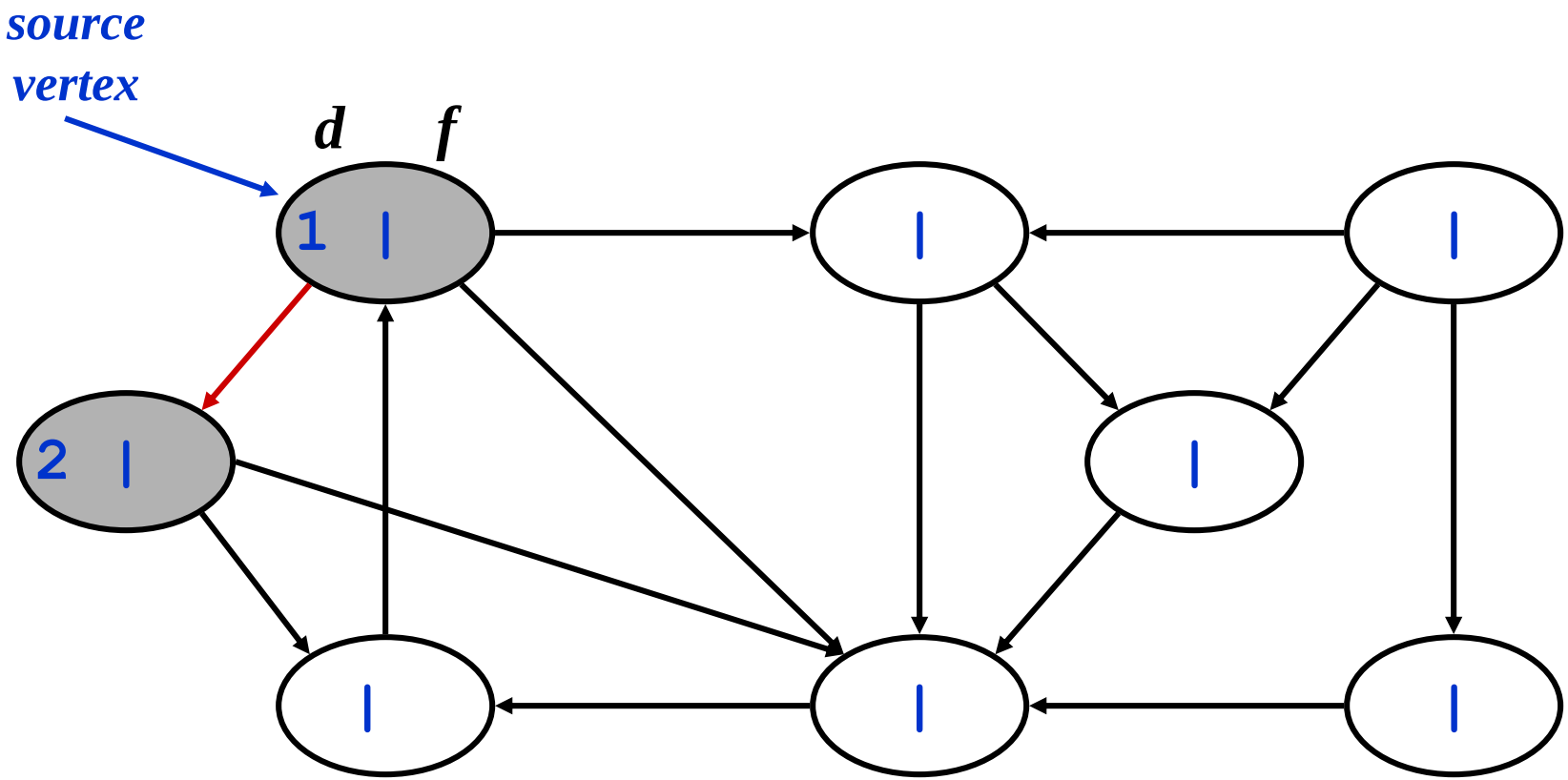
DFS Example



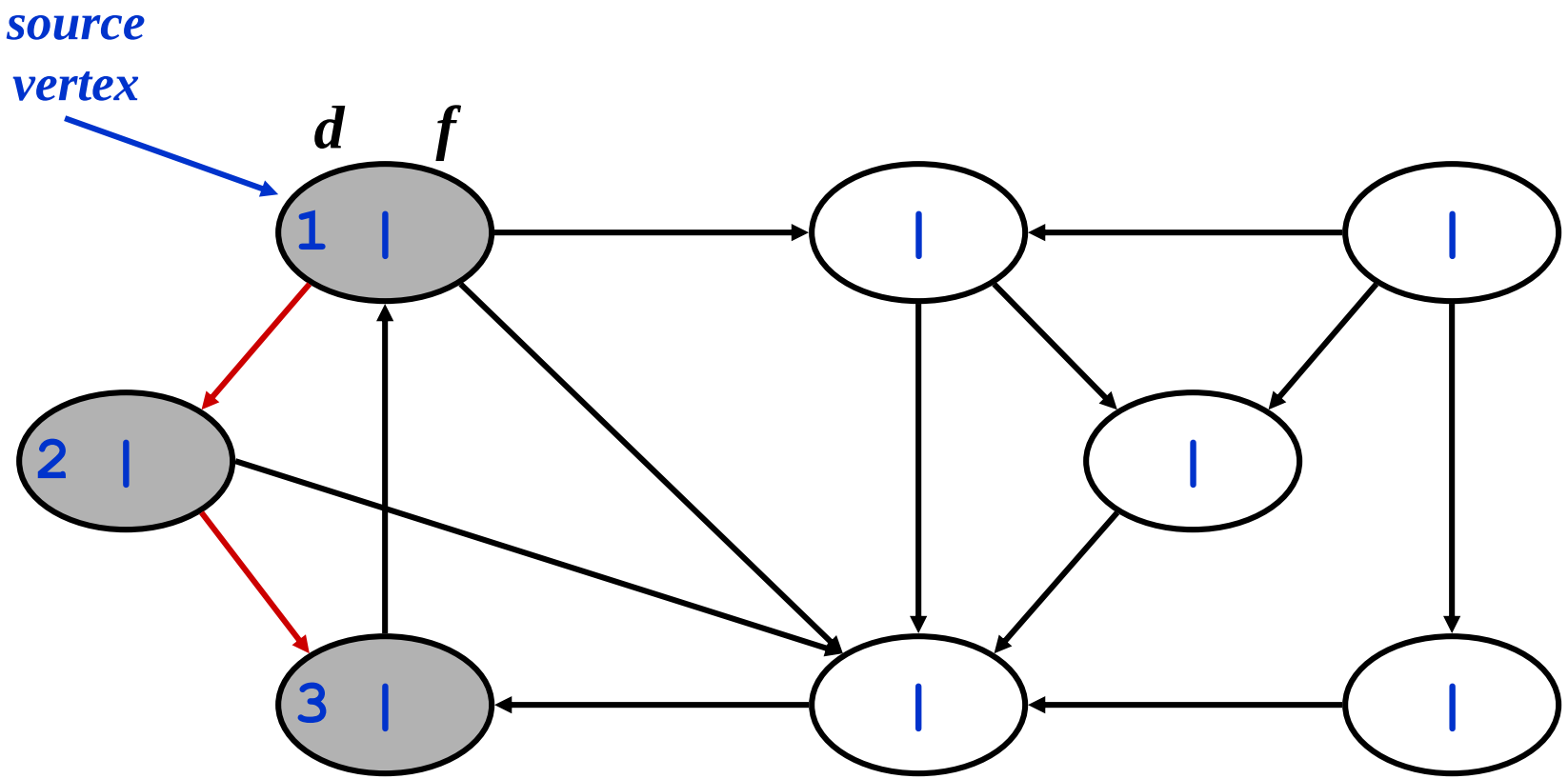
DFS Example



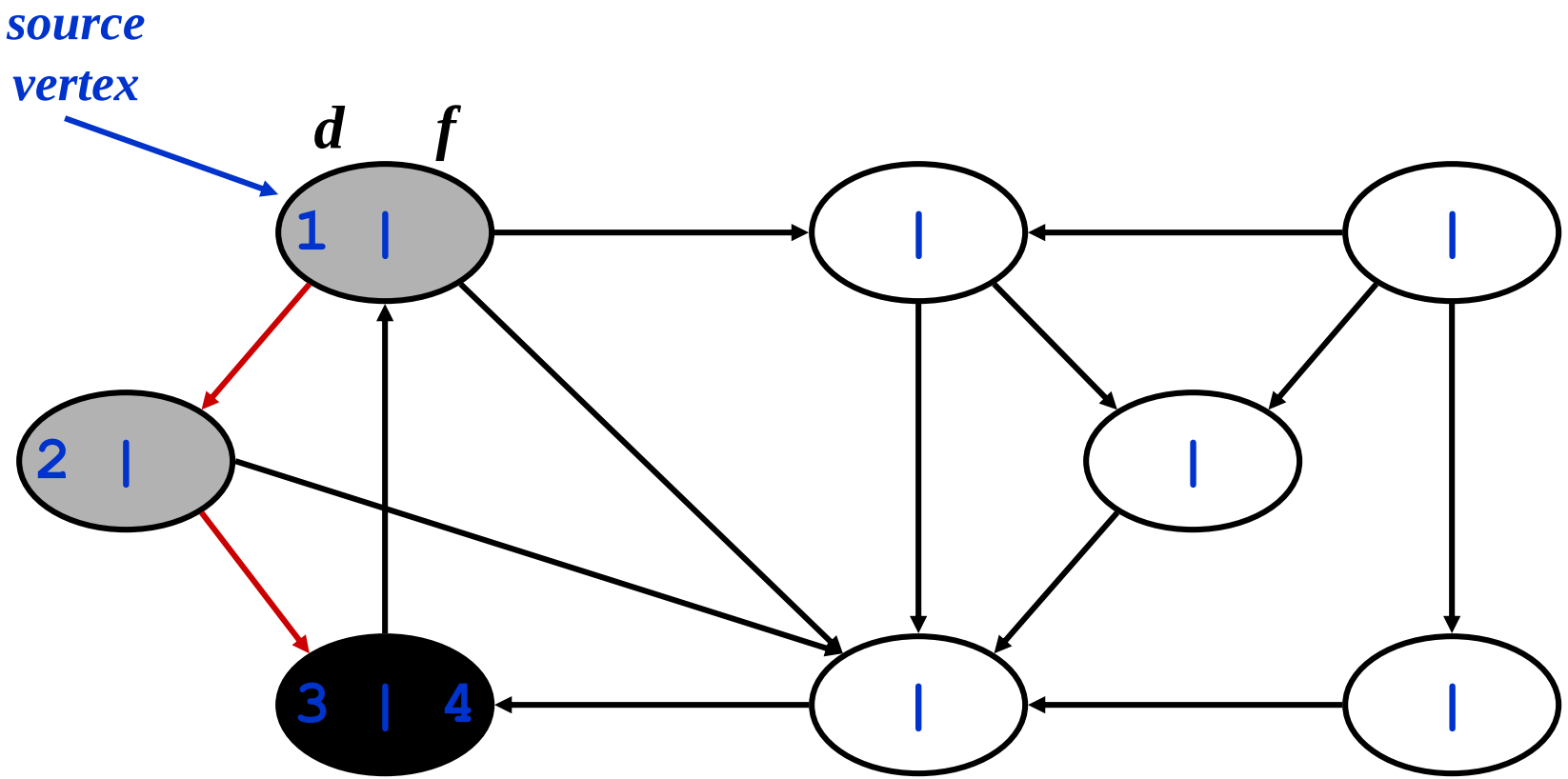
DFS Example



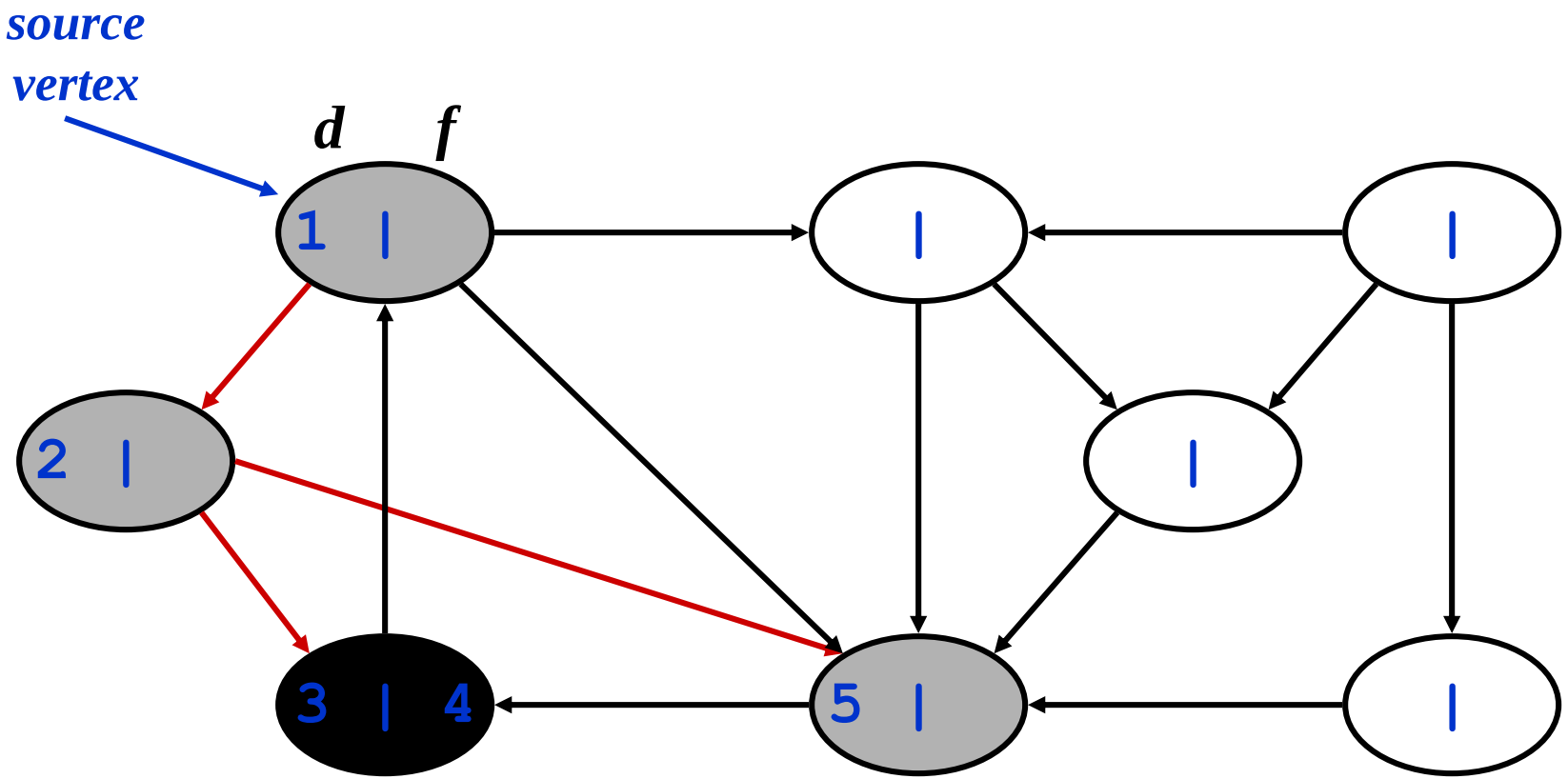
DFS Example



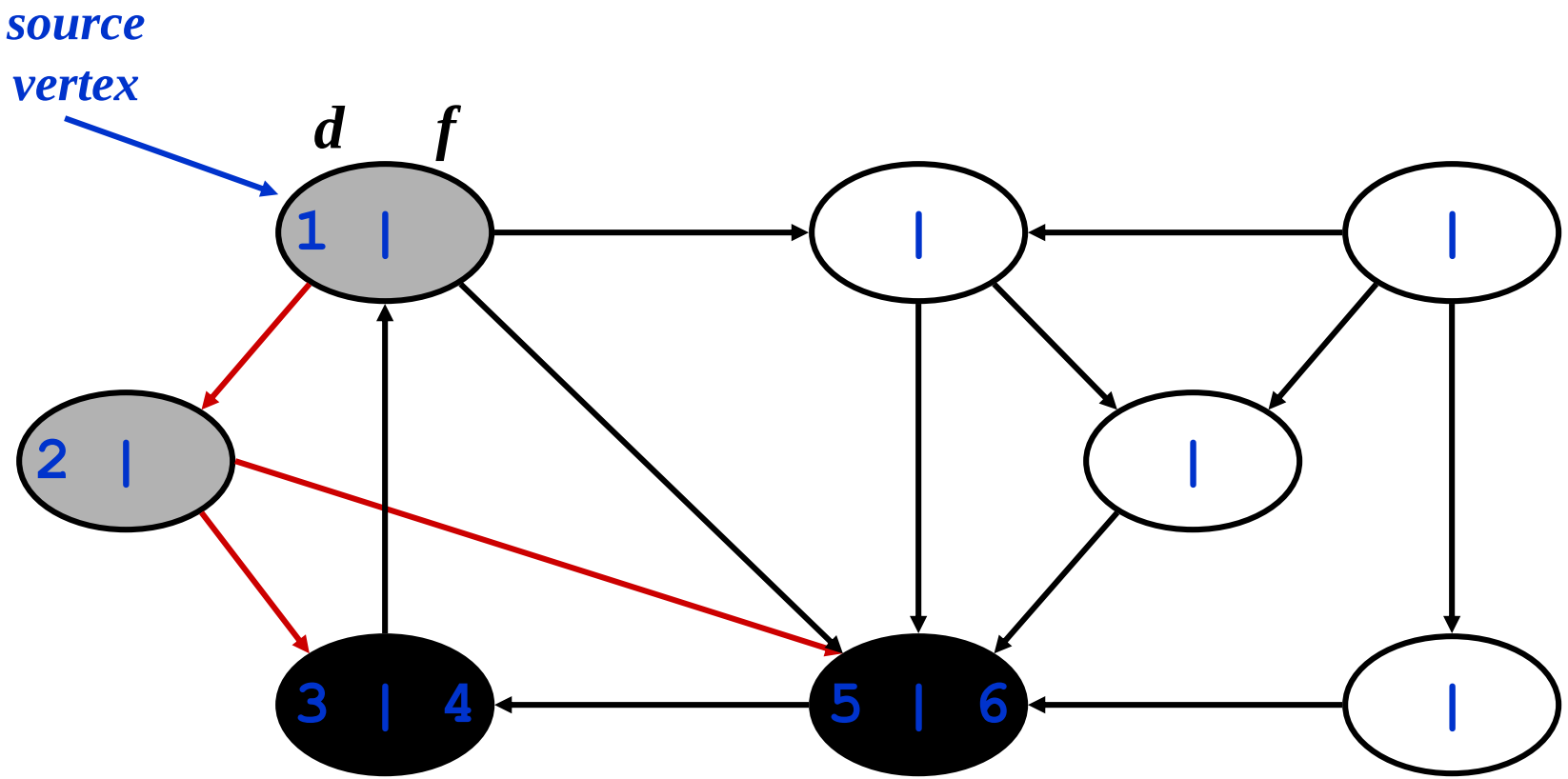
DFS Example



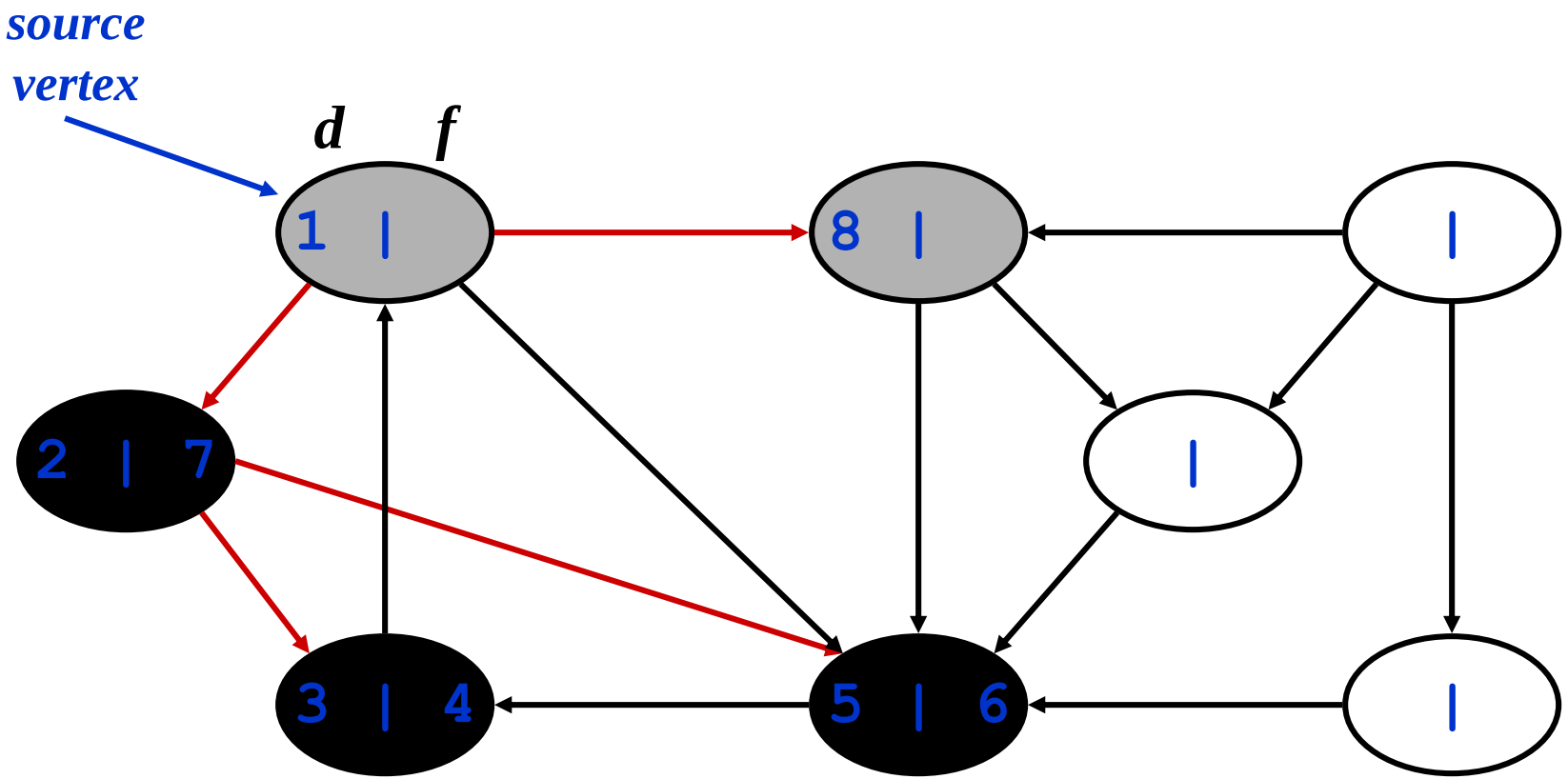
DFS Example



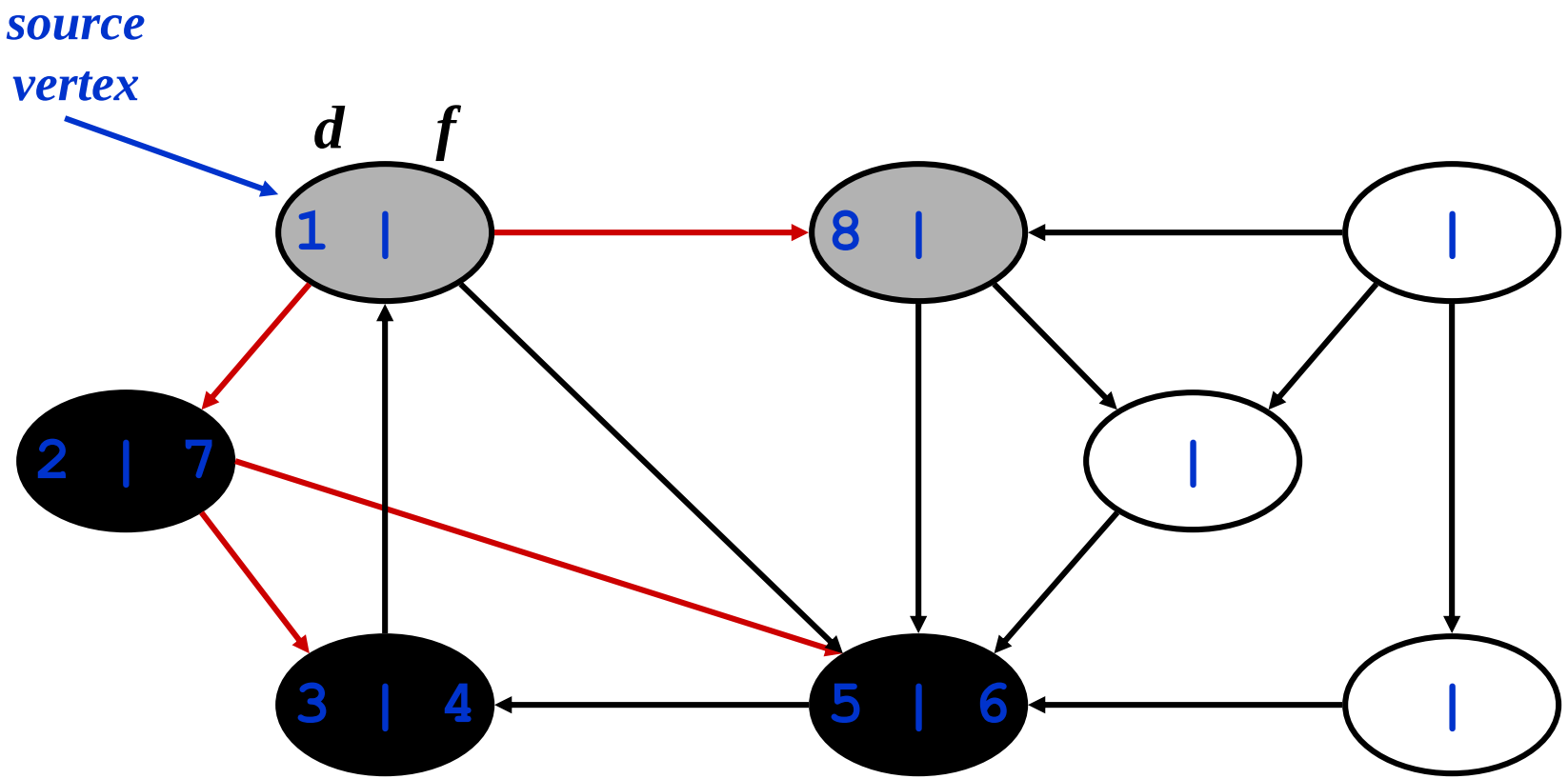
DFS Example



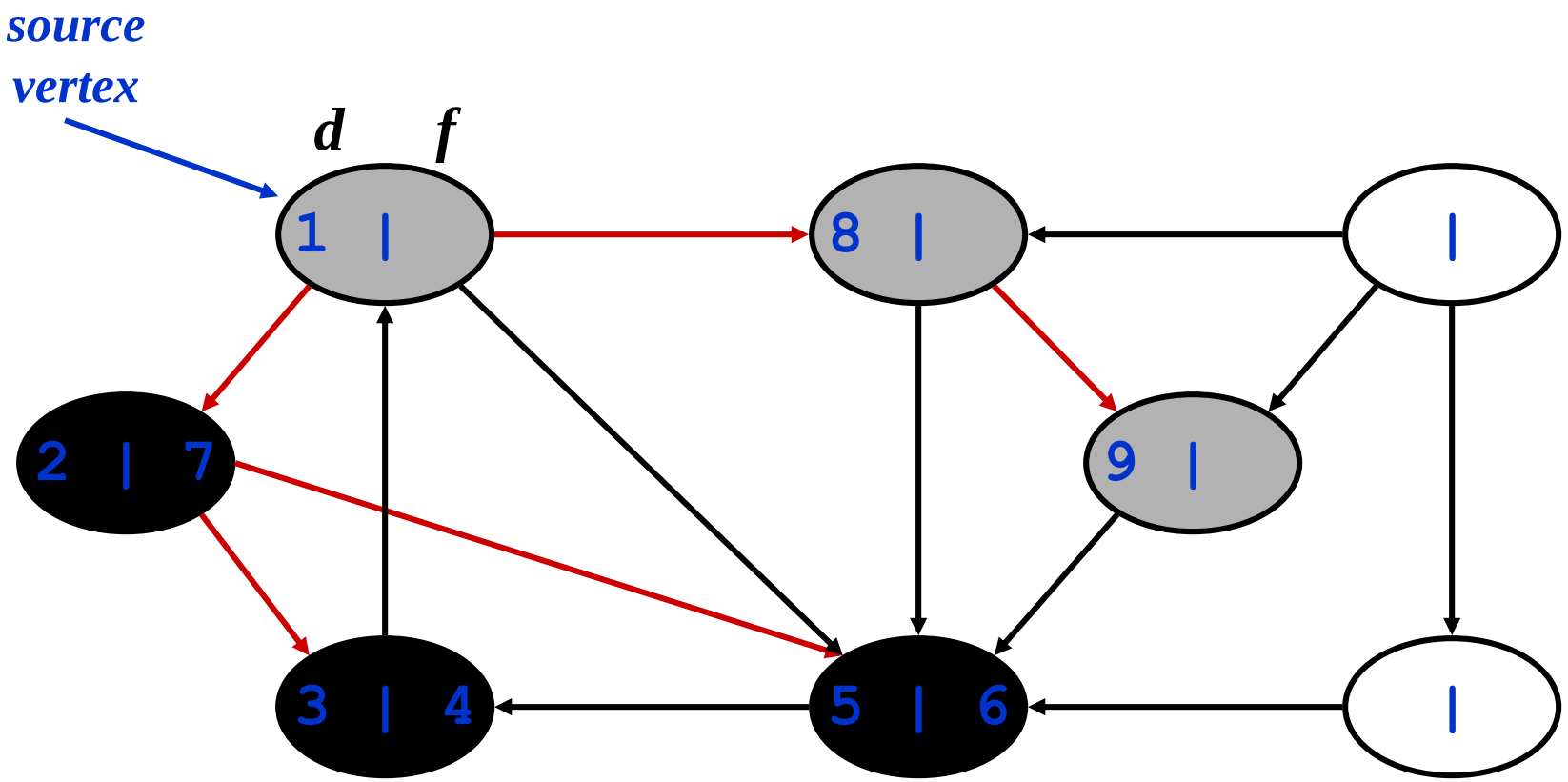
DFS Example



DFS Example

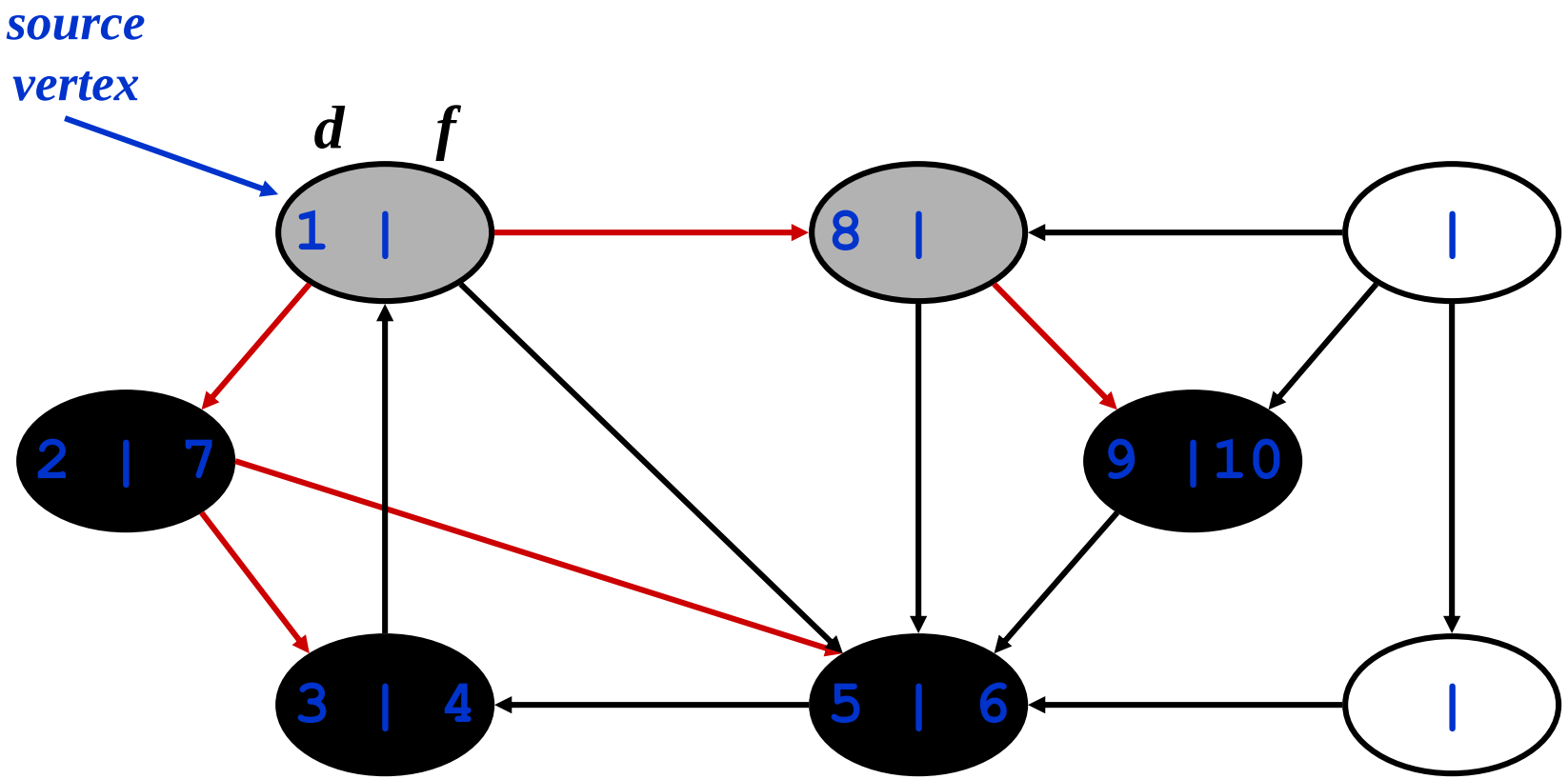


DFS Example

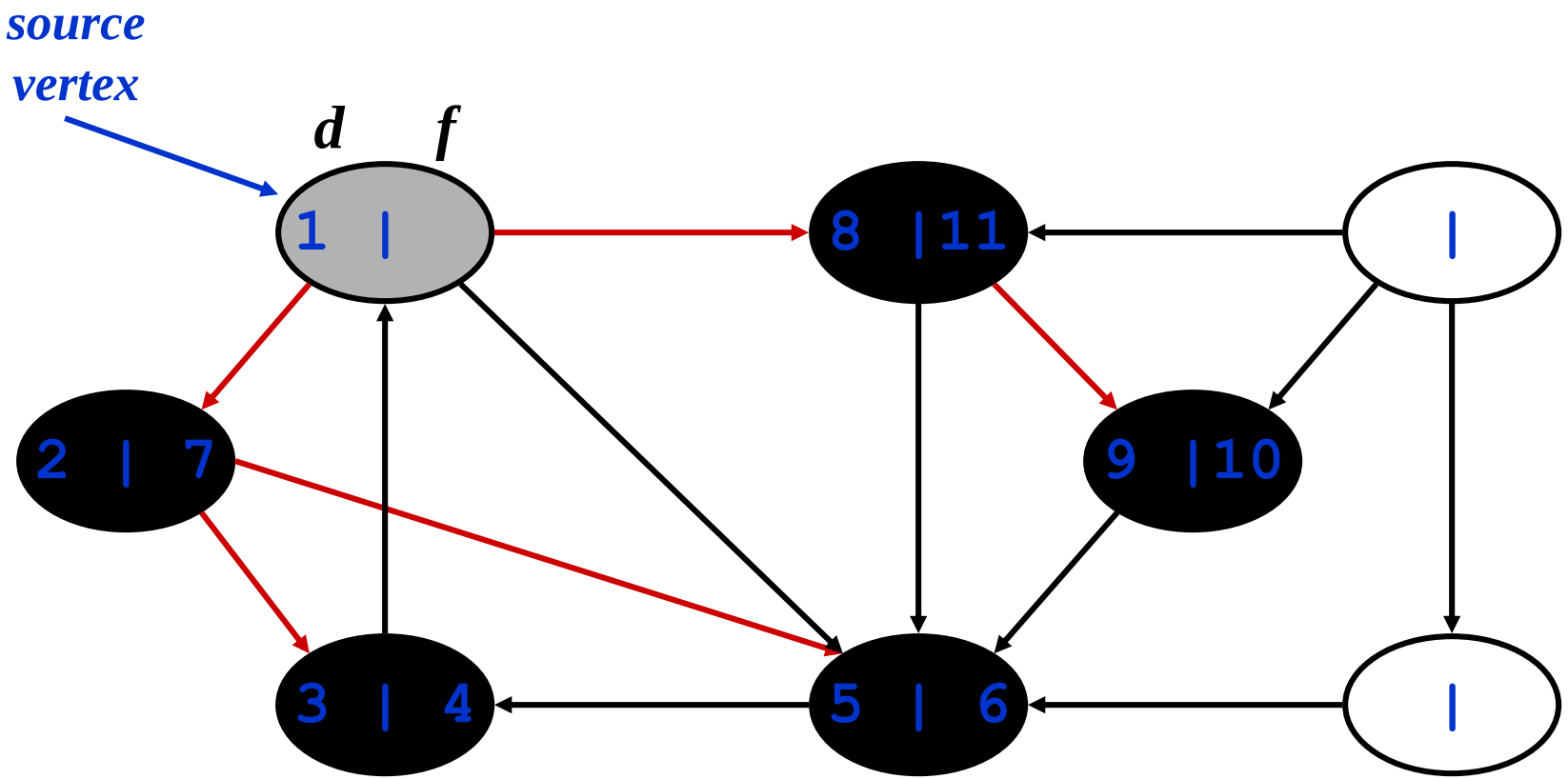


What is the structure of the grey vertices?
What do they represent?

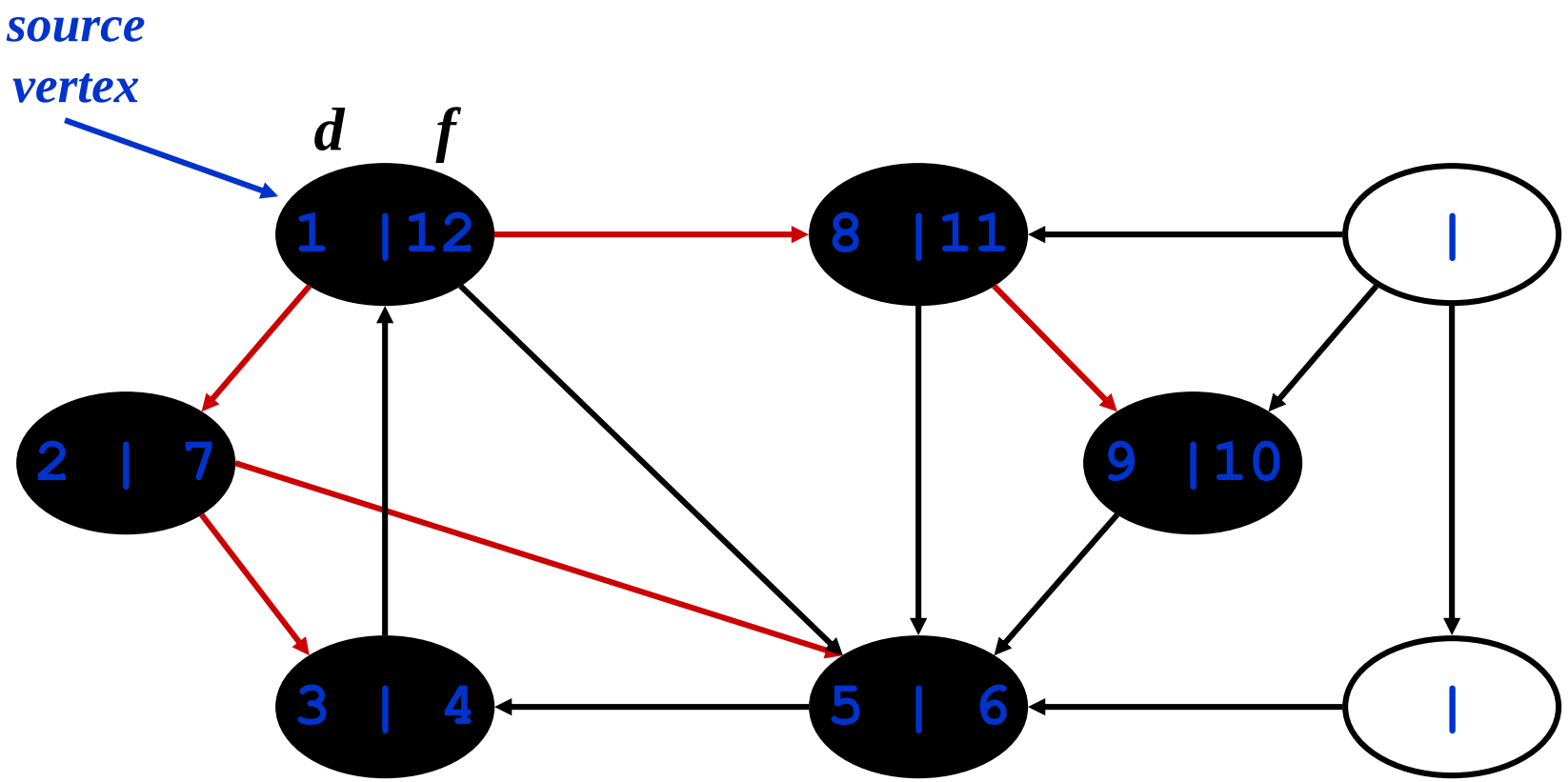
DFS Example



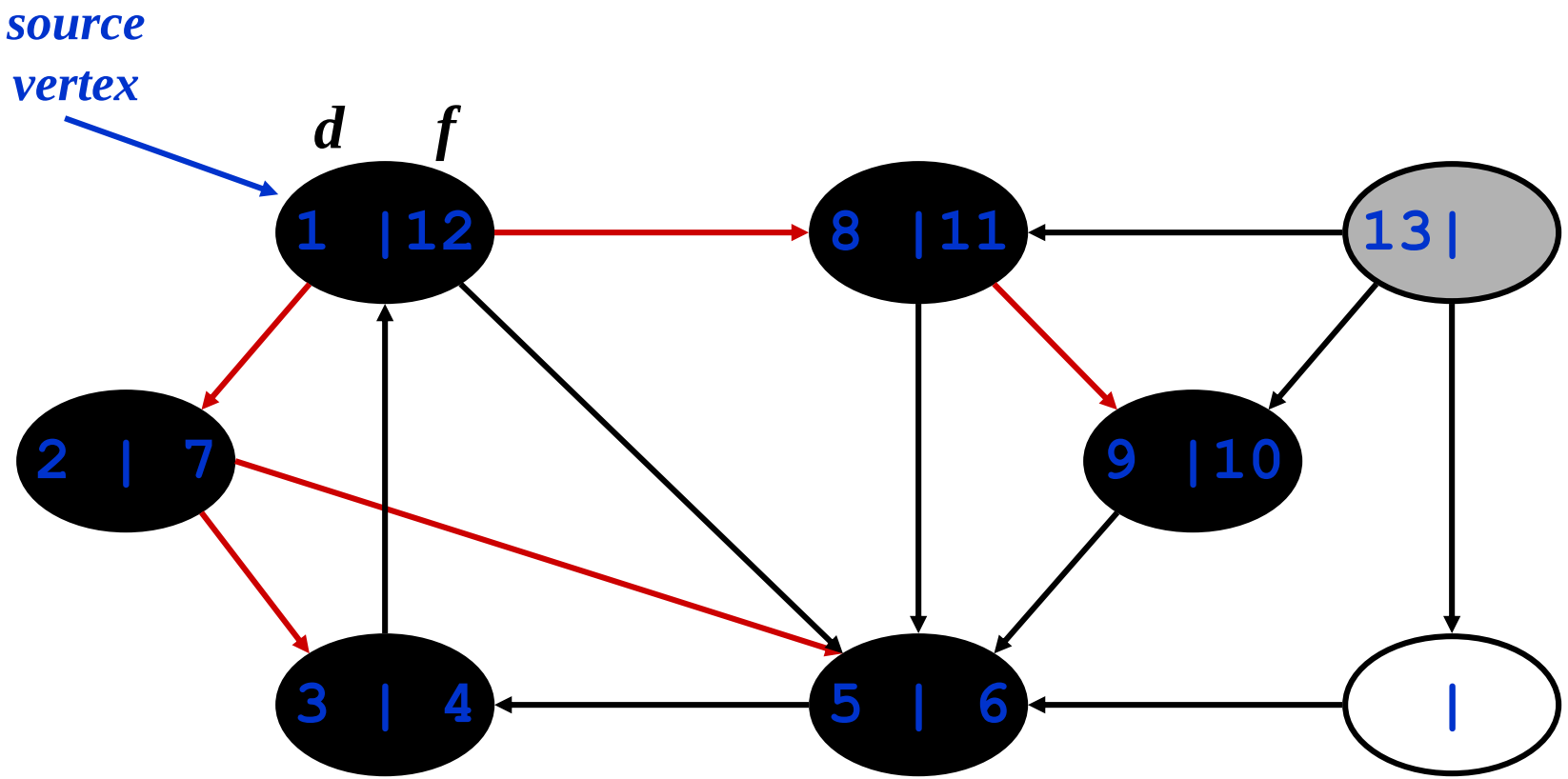
DFS Example



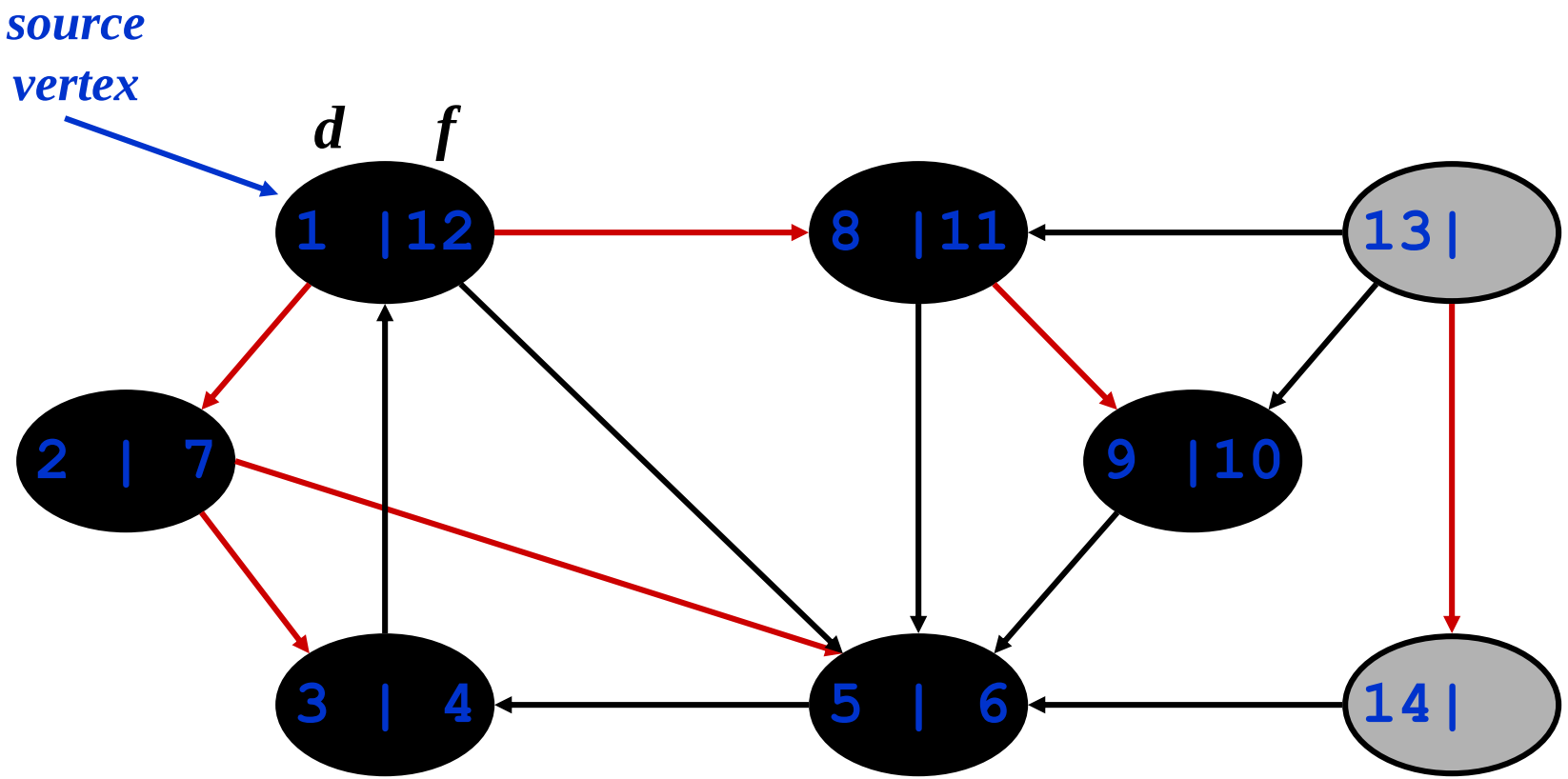
DFS Example



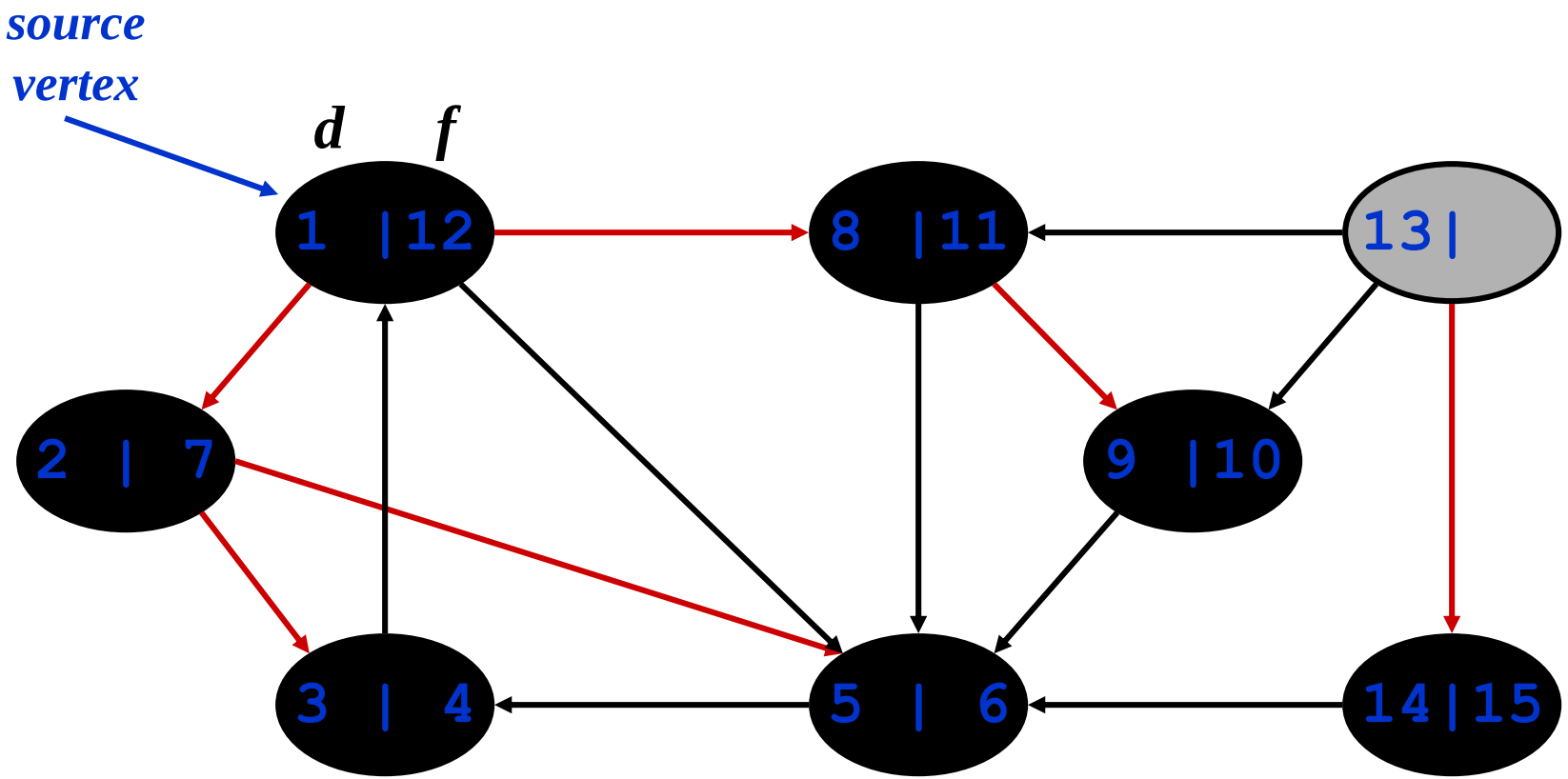
DFS Example



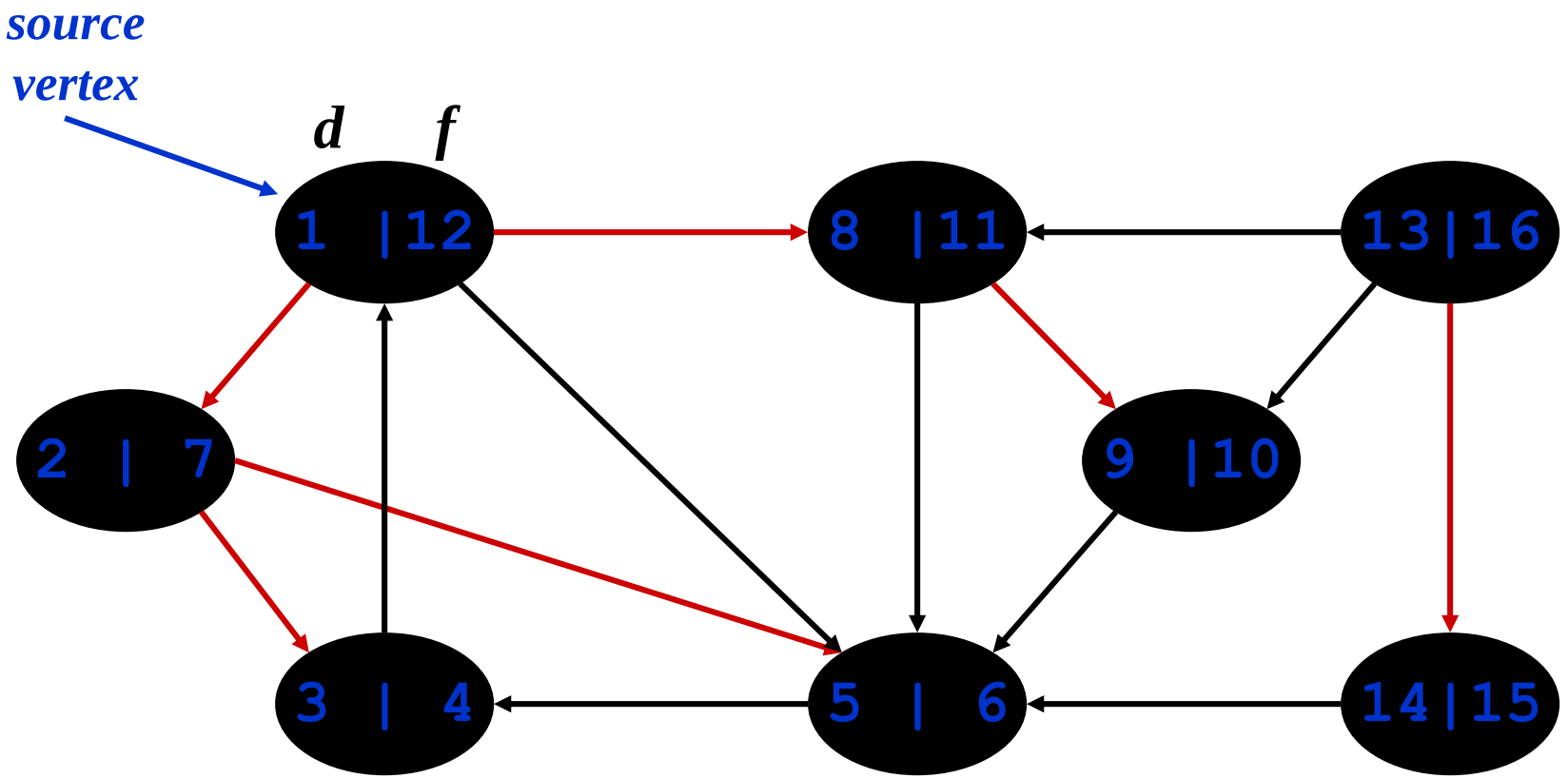
DFS Example



DFS Example



DFS Example



Paranthesis theorem

For any two vertices u and v in the graph only one of the below hold true

D[u] < f[u] <d[v] < f[v]

D[v] < f[v] <d[u] < f[u]

D[u] < d[v] <f[v] < f[u]

D[v] < d[u] <f[u] < f[v]

- Corollary: Vertex v is a proper descendant of vertex u in the depth-first forest for a (directed or undirected) graph G iff d[u] < d[v] <f[v] < f[u]

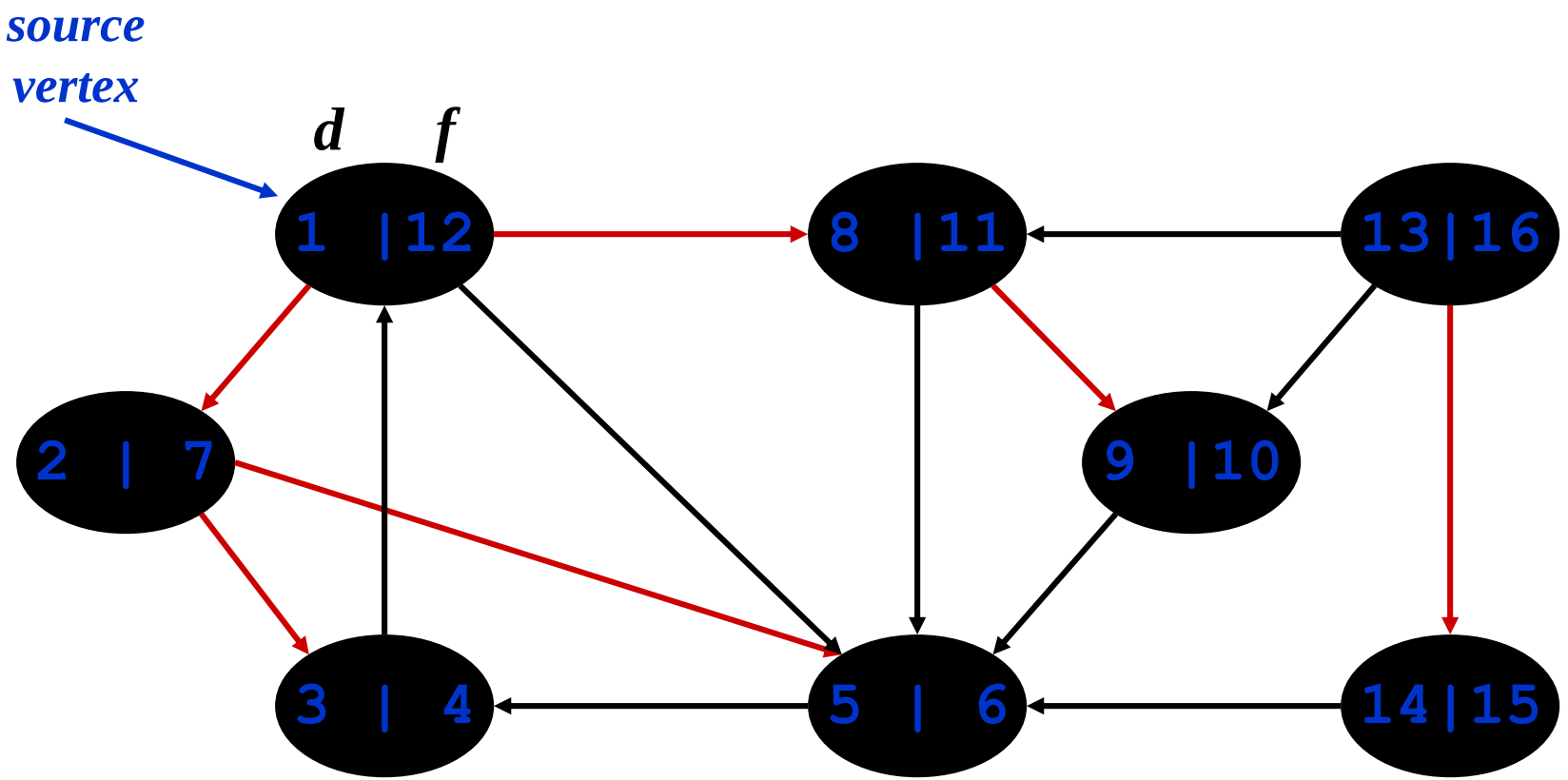
DFS: Kinds of edges

- 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 across subtrees where the two nodes are not related by descendant ancestor property.

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

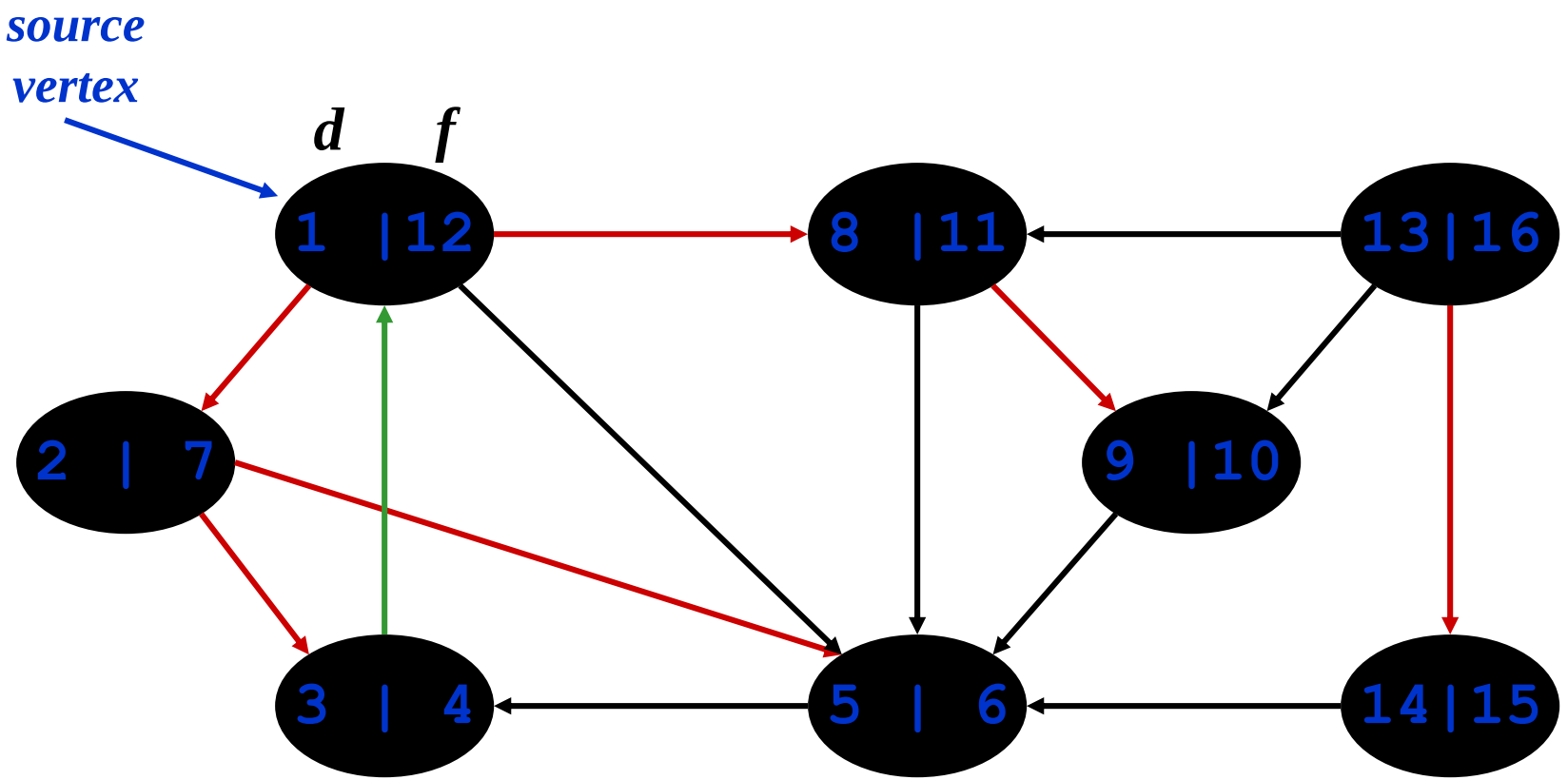


Tree edges

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 grey vertex (grey to grey)

DFS Example

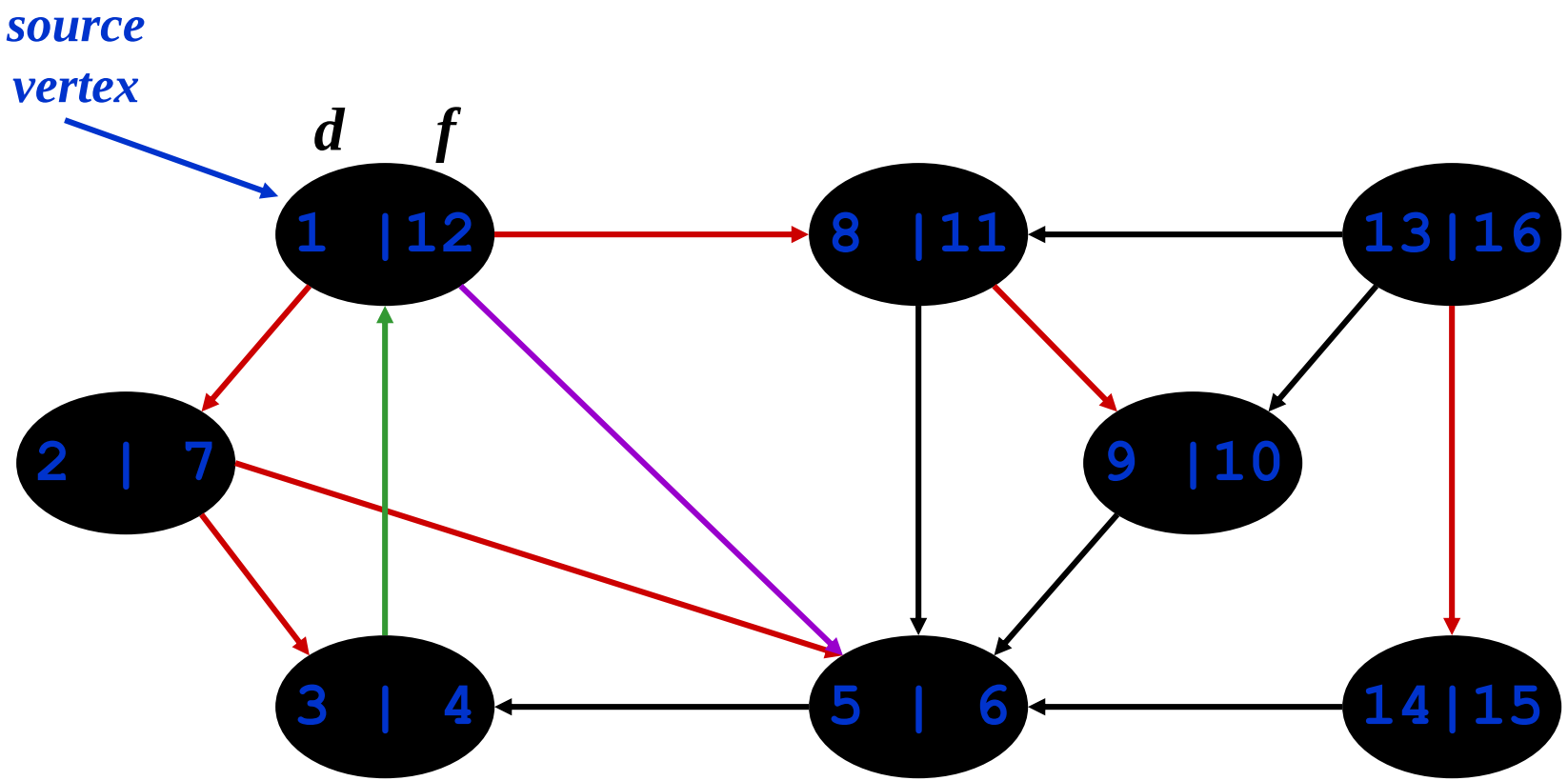


Tree edges *Back edges*

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 grey node to black node

DFS Example

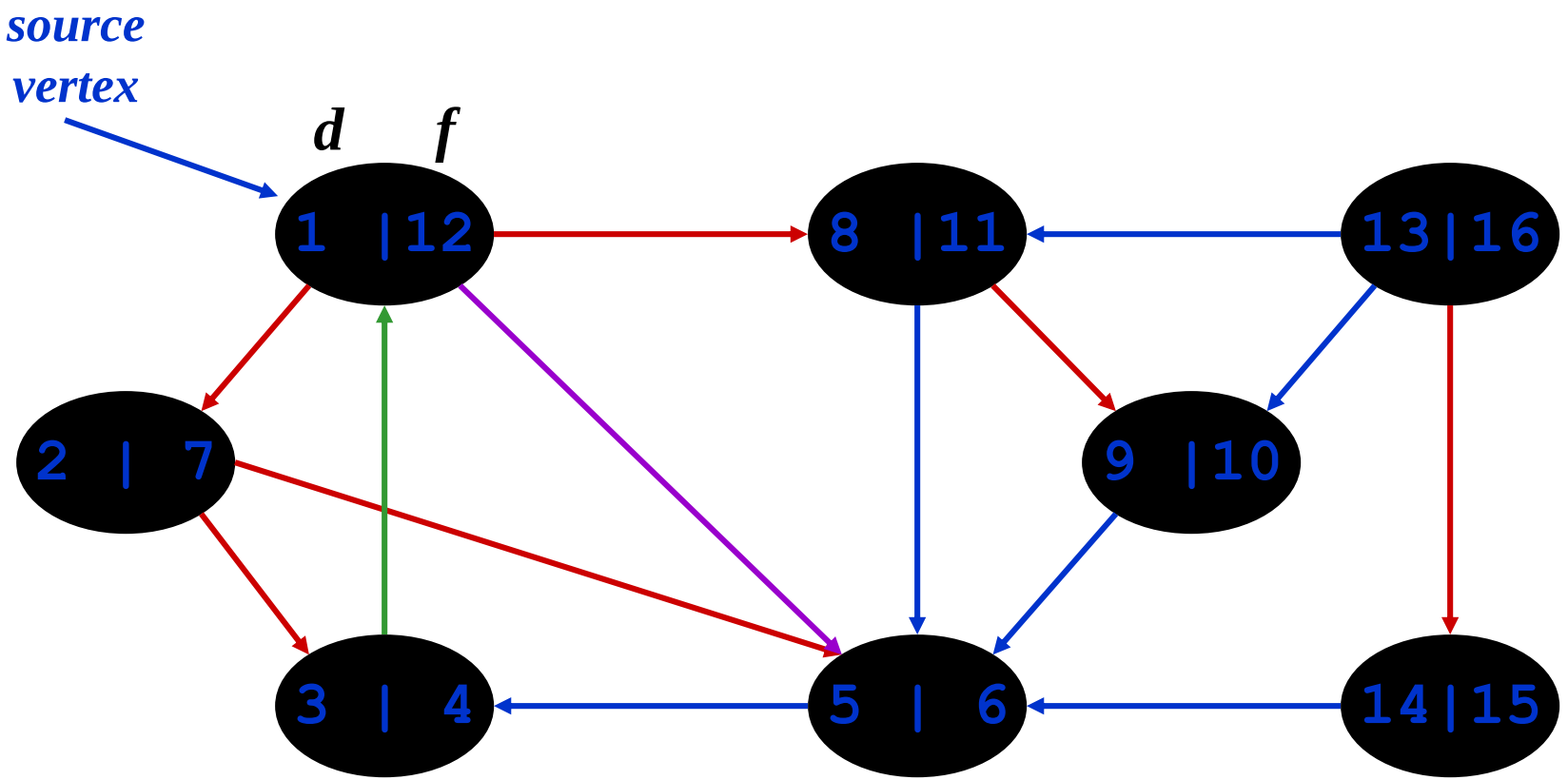


Tree edges *Back edges* *Forward edges*

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 across subtrees where the two nodes are not related by descendant ancestor property.
    - From a grey 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

True or False

- if a directed graph G contains a path from u to v, and if u.d < v.d in a depth-first search of G ,then v is a descendant of u in the depth-first forest produced.
- if a directed graph G contains a path from u to v, then any depth-first search must result in v.d <= u.f

True or False conti....

- if a directed graph G contains a path from u to v, and if u.d < v.d in a depth-first search of G ,then v is a descendant of u in the depth-first forest produced.
- if a directed graph G contains a path from u to v, then any depth-first search must result in v.d <= u.f

  Take Graph edges a->u u->a a->v

So path fom u to v but if we start dfs at a then both conditions dont hold.

DFS: Kinds Of Edges

- Thm 23.9: If G is undirected, a DFS produces only tree and back edges

DFS: Kinds Of Edges

- Thm 23.9: If G is undirected, a DFS produces only tree and back edges
- Proof by contradiction:
  - Cross Edges(u,v) or forward edges(u,v) are edges from grey to black. In an undirected graph can there be an edge from grey to black?? Black would imply all connections are explored. But then in an undirected graph u becomes a connection of v. Node v should have explored ‘u’ before it turned black. Contradiction!!