GRAPH SEARCH ALGORITHMS

➢DFS(Depth First Search)

Mashiwat Tabassum Waishy

Lecturer

Department *of*

CSE

STAMFORD UNIVERSITY BANGLADESH

Depth First Search

- DFS, go as far as possible along a single path until it reaches a dead end (that is a vertex with no edge out or no neighbor unexplored) then backtrack
- As the name implies the DFS search deeper in the graph whenever possible. DFS explores edges out of the most recently discovered vertex **v** that still has unexplored edges leaving it. Once all of v's edges have been explored, the search backtracks to explore edges leaving the vertex from which **v** was discovered.

Depth-First Search

- Graph G = (V, E) directed or undirected
- Adjacency list representation
- Goal: Systematically explore every vertex and every edge
- Idea: search deeper whenever possible
  - Using a Stack(LIFO)

Information Needed to Maintain in DFS Algorithm

- Input graph G = (V,E) is represented using adjacency list.

- Each vertex is colored **white, gray or black**.

    (1) **Vertex with White Color**: undiscovered vertex.

    (2) **Vertex with Gray Color**: Consider to be discovered.

    (3) **Vertex with Black Color:** Adjacency-list of Vertex has been discovered .

- Additional Data Structures

    (1) color[u] : Store the color of each vertex $u \varepsilon V$

    (2) Π[u]   : Store the predecessor of u.

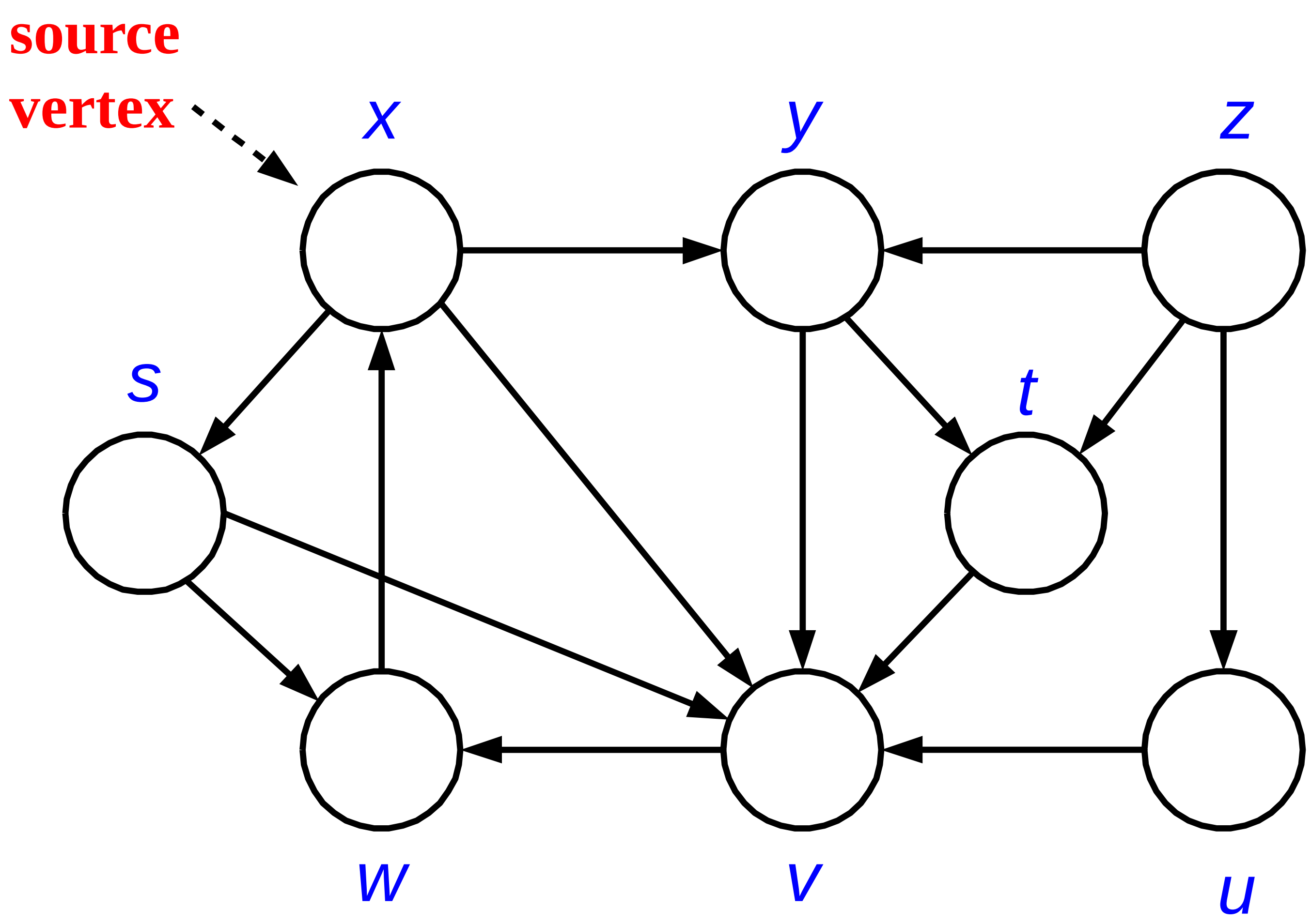
- Besides creating depth first forest DFS also timestamps each vertex. Each vertex goes through two time stamps:

    (3) d[u]   : Store when u is first discovered and grayed (discovery time).

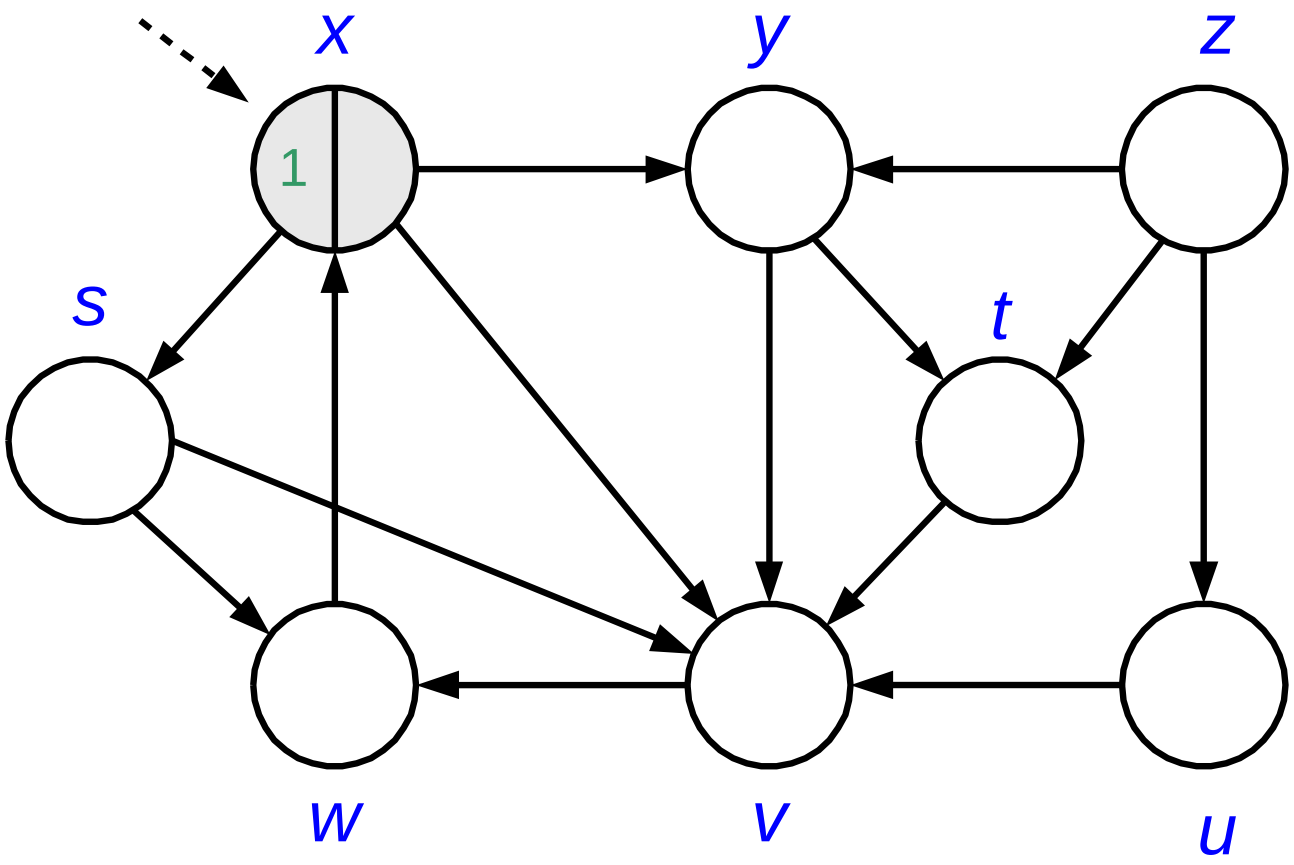
    (4) f[u]   : Store when the search finishes examining u's adjacency-list and blackens u (finishing time).

    - $f[u] > d[u]$

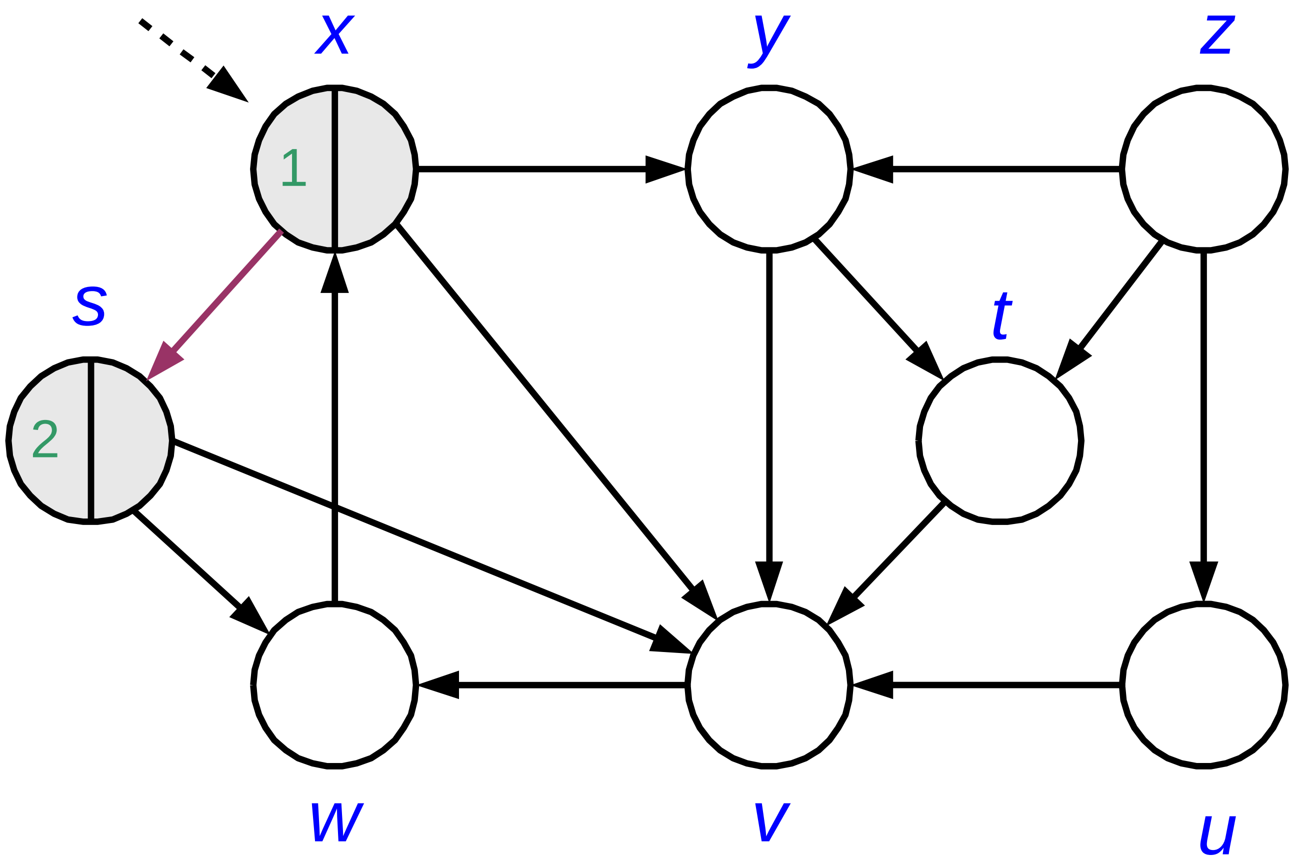
Depth-First Search: Example



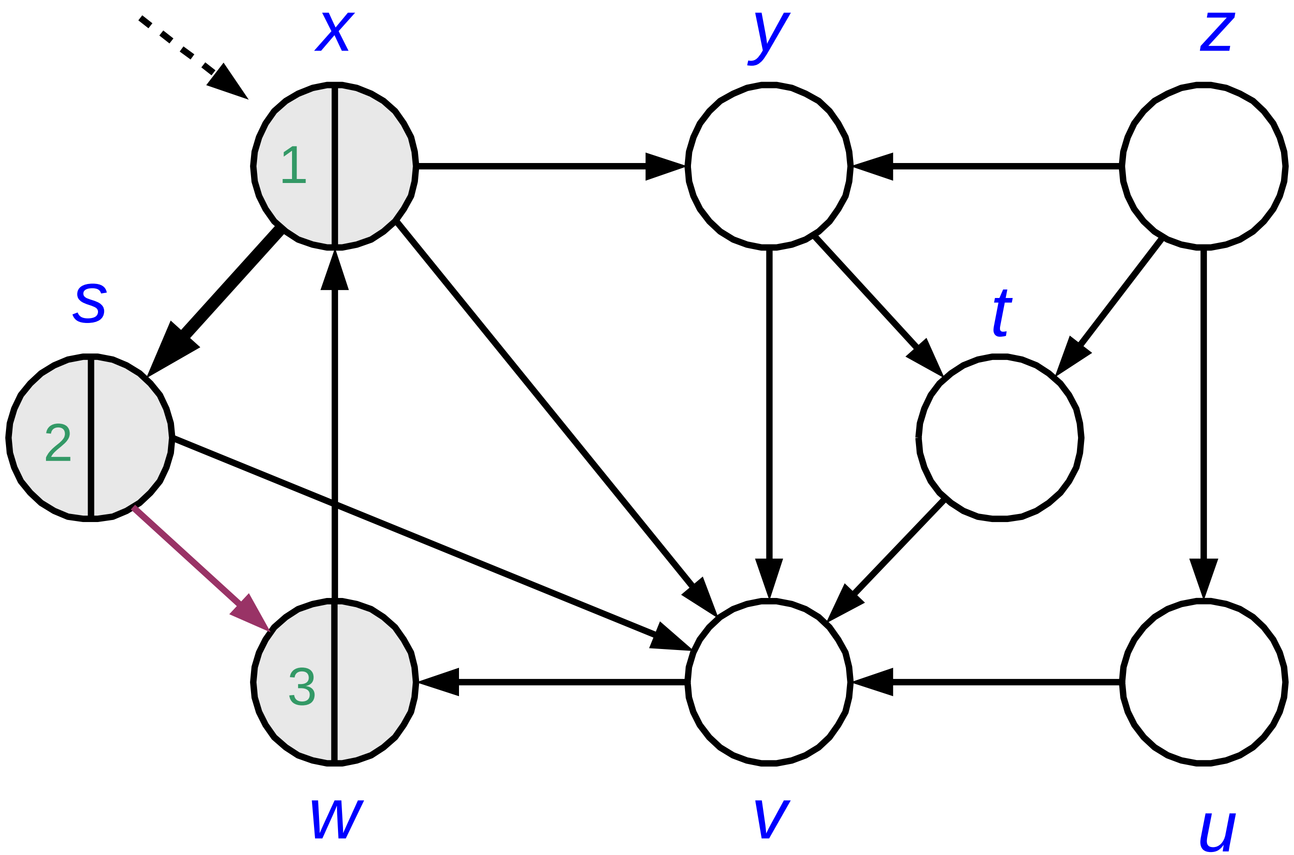
Depth-First Search: Example



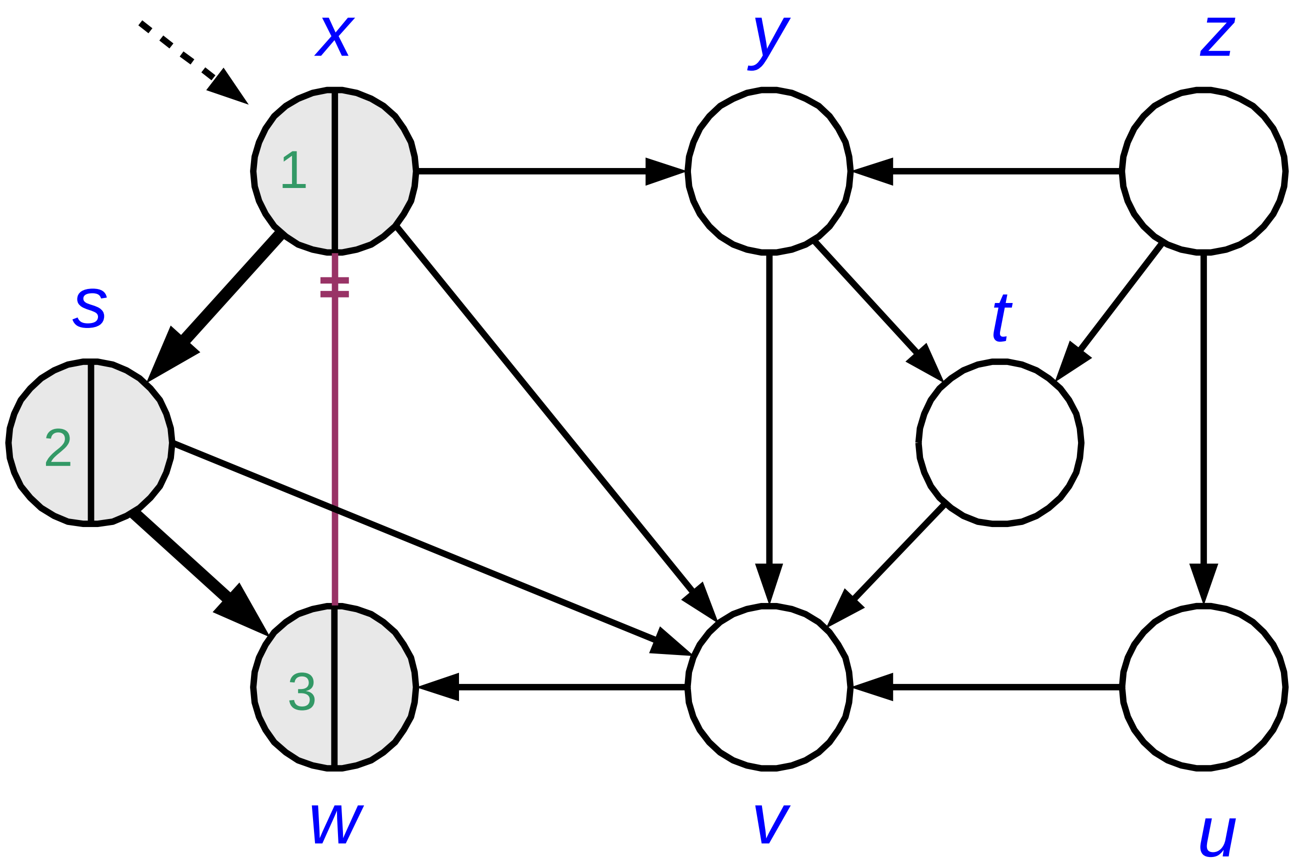
Depth-First Search: Example



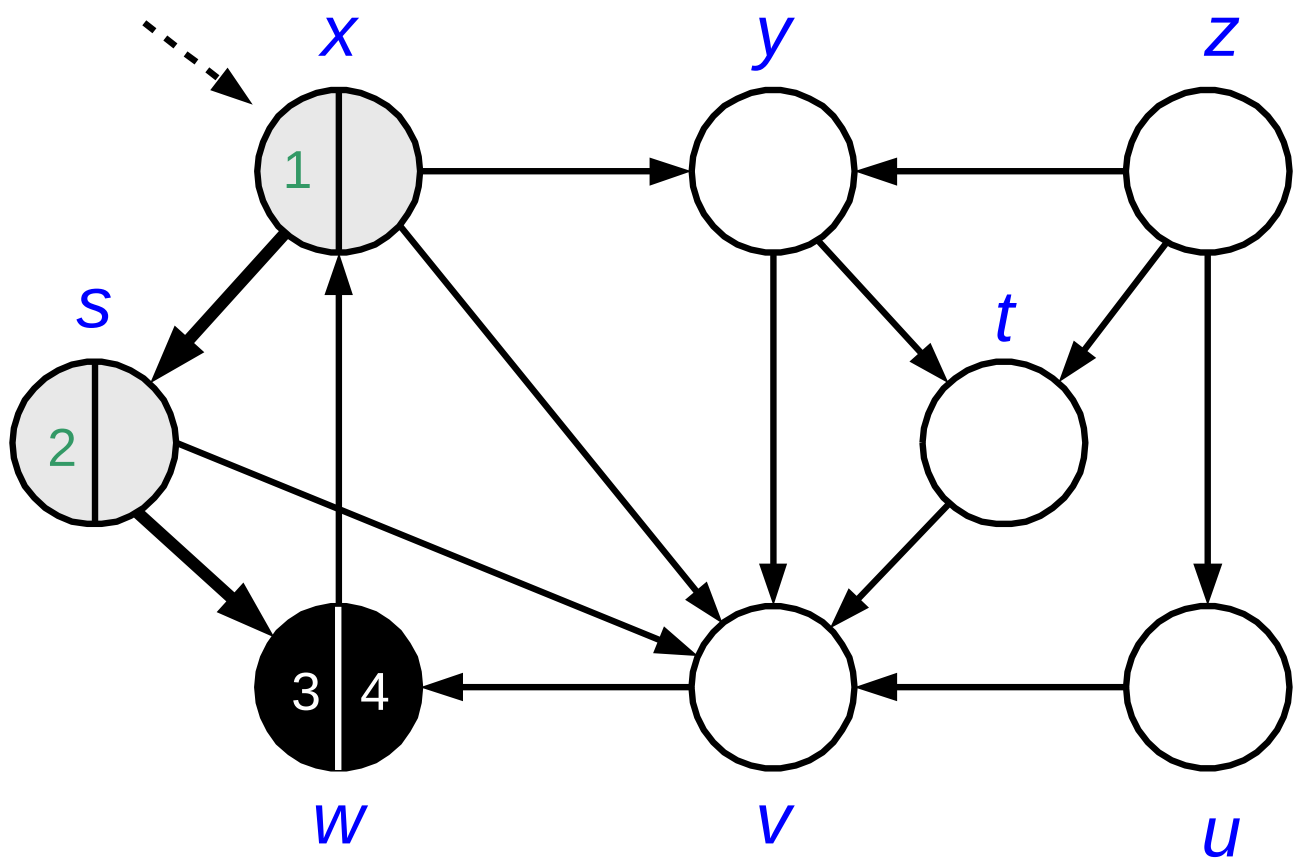
Depth-First Search: *Example*



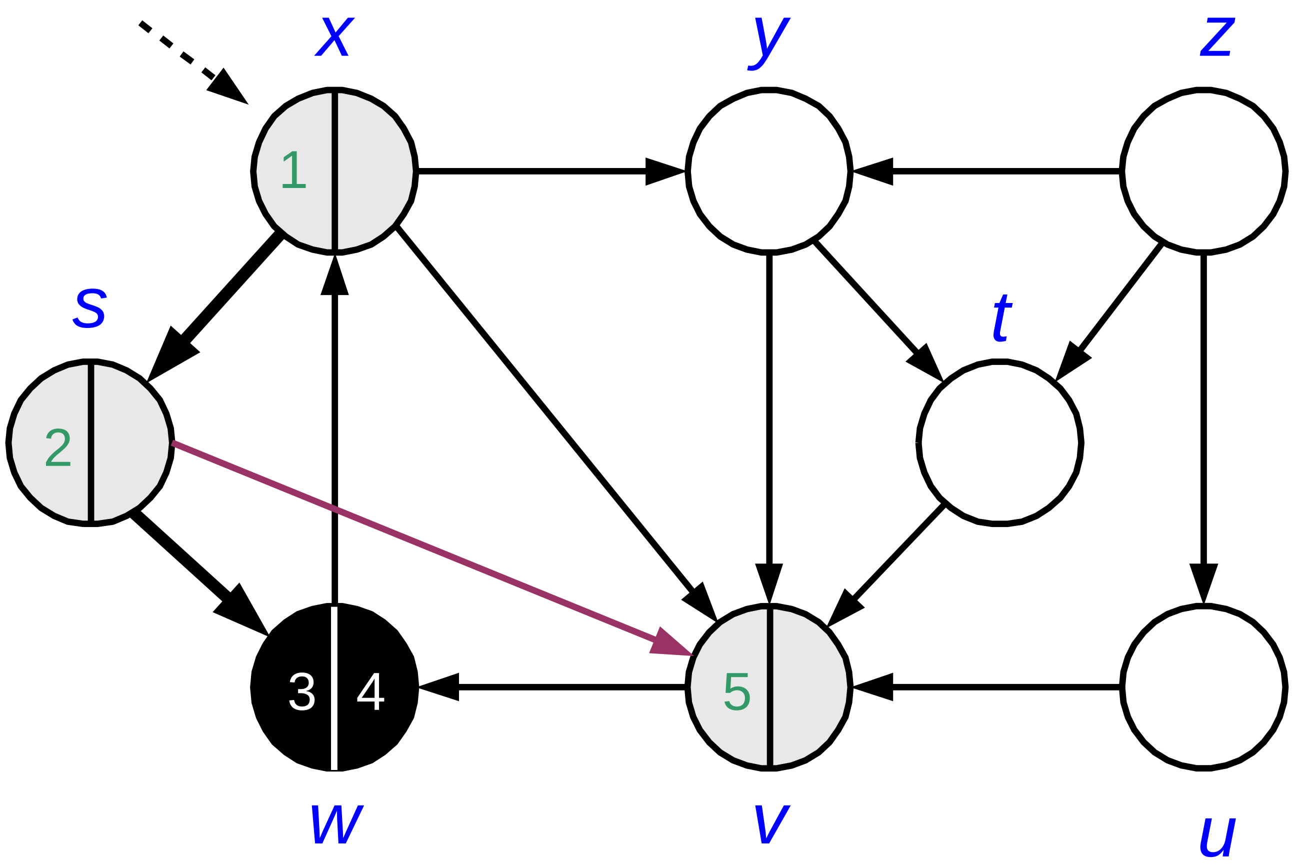
Depth-First Search: Example



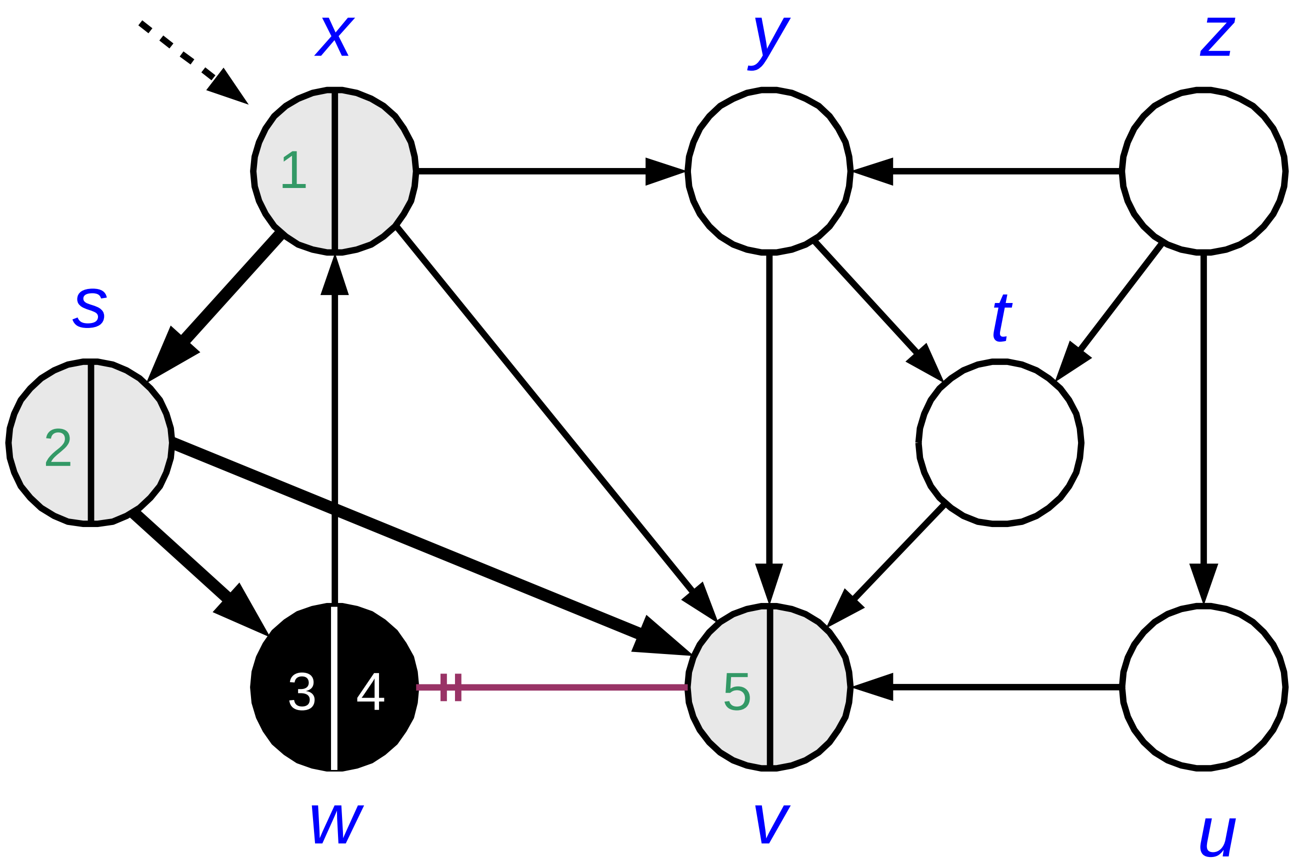
Depth-First Search: *Example*



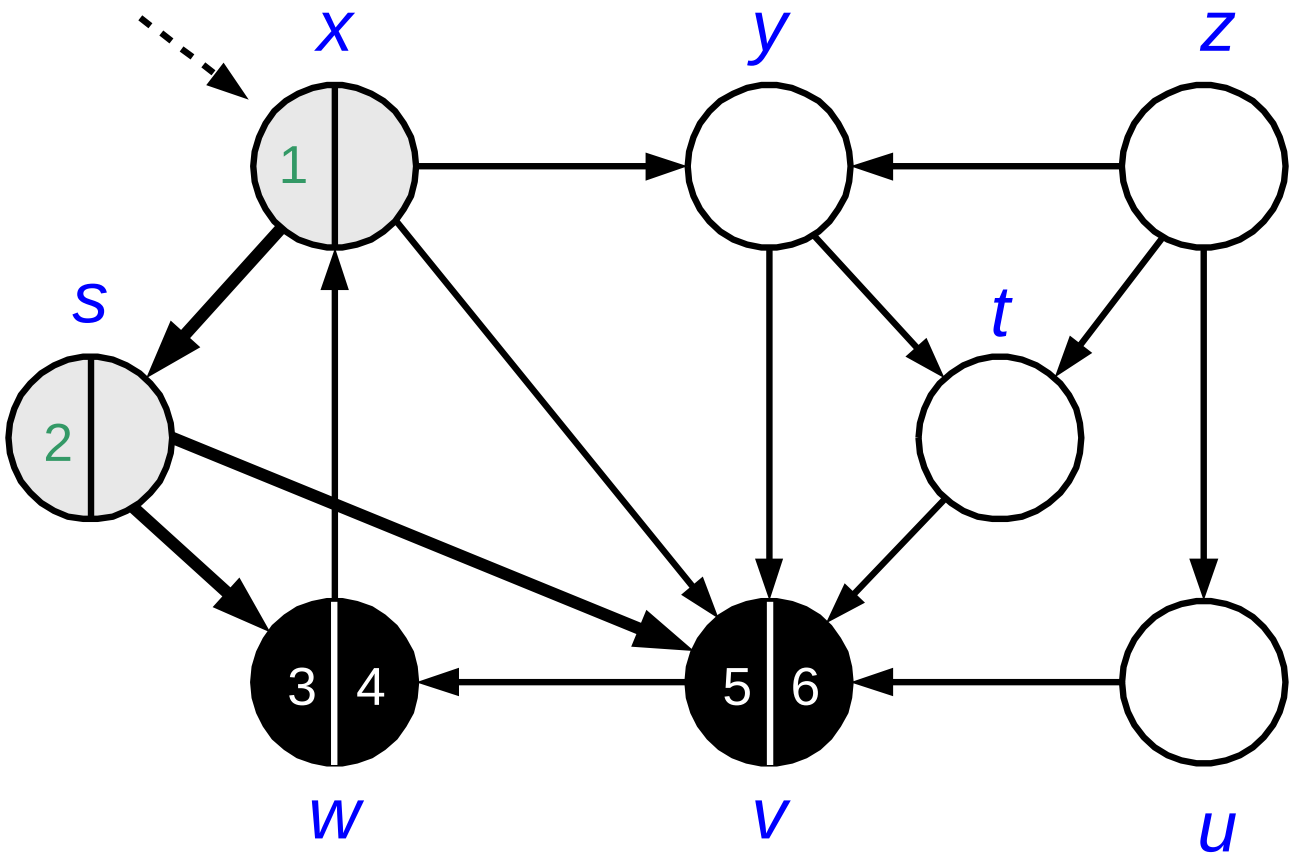
Depth-First Search: Example



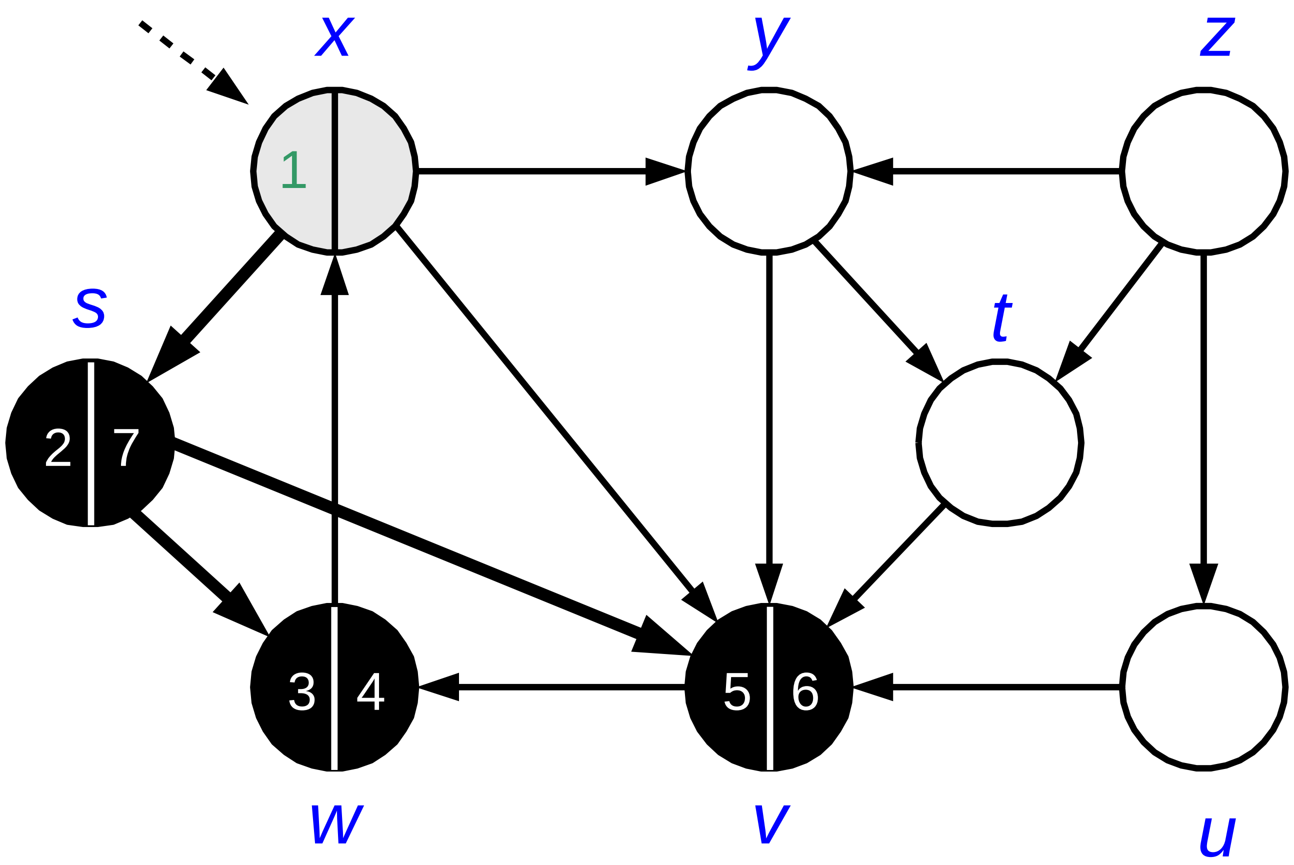
Depth-First Search: Example



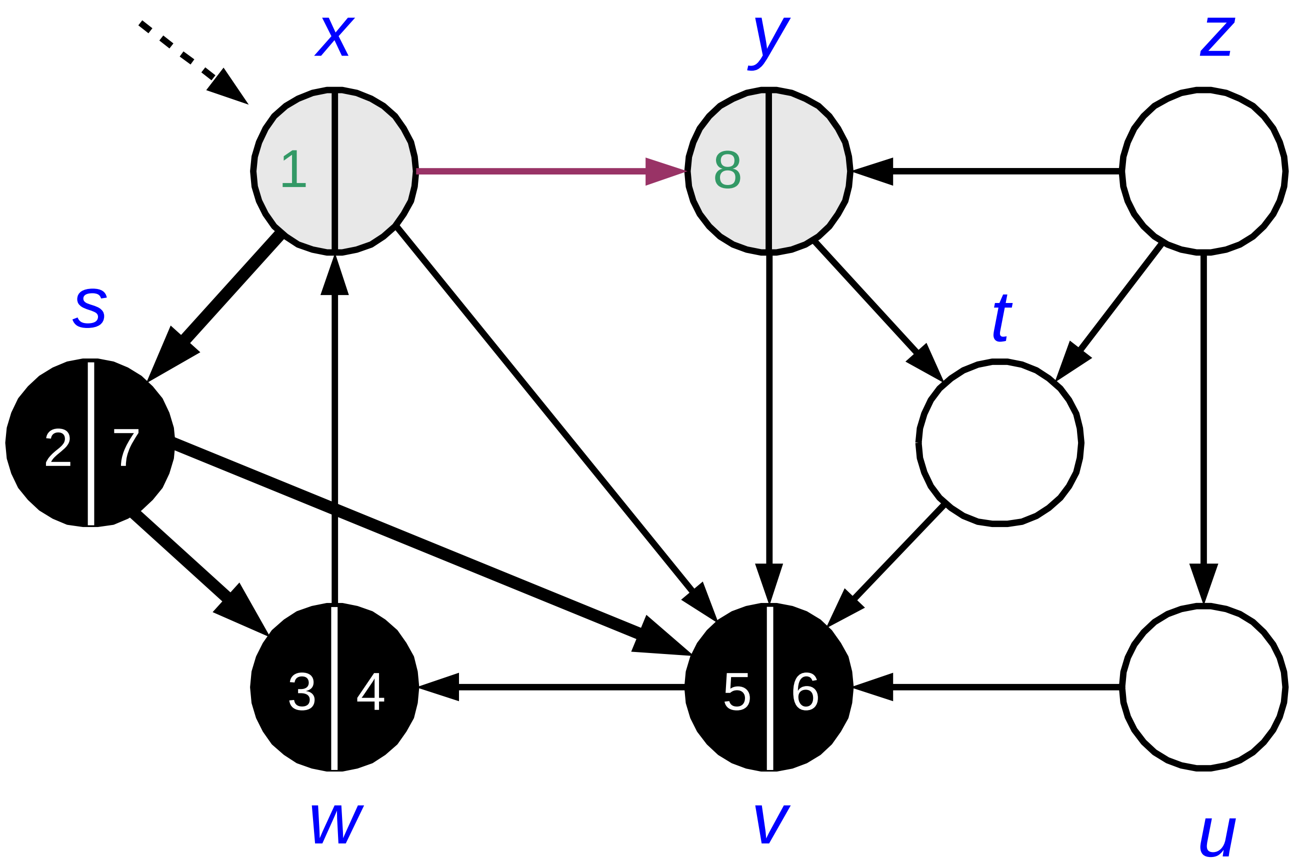
Depth-First Search: Example



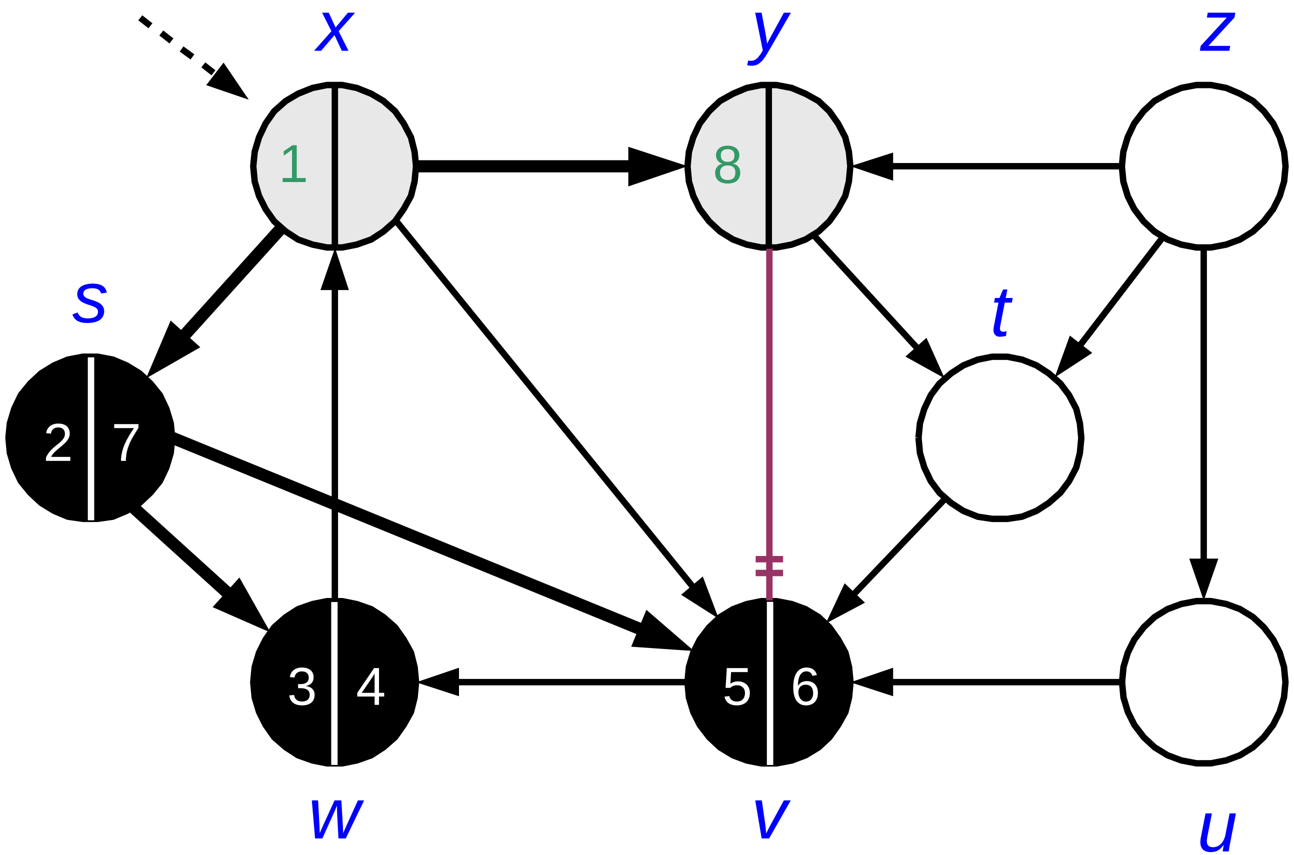
Depth-First Search: Example



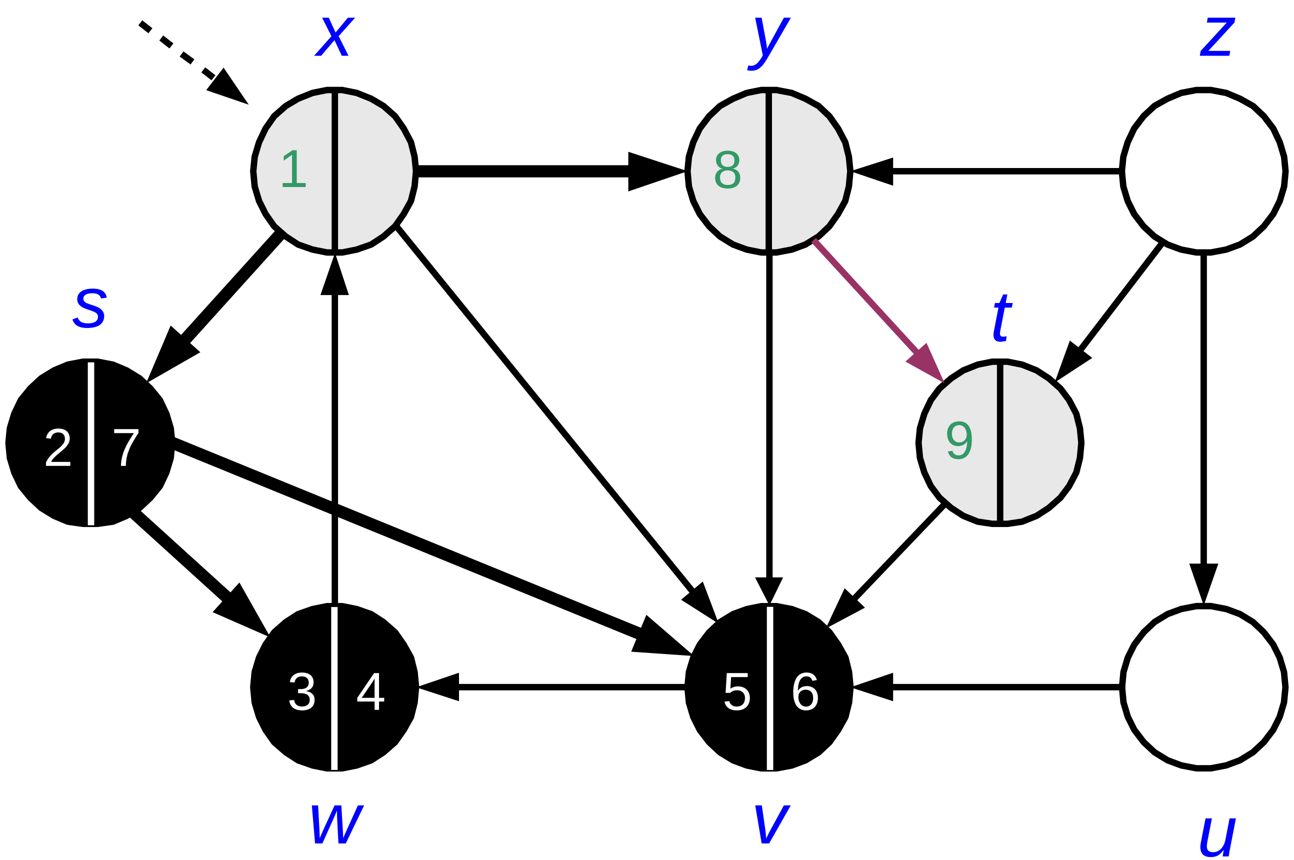
Depth-First Search: *Example*



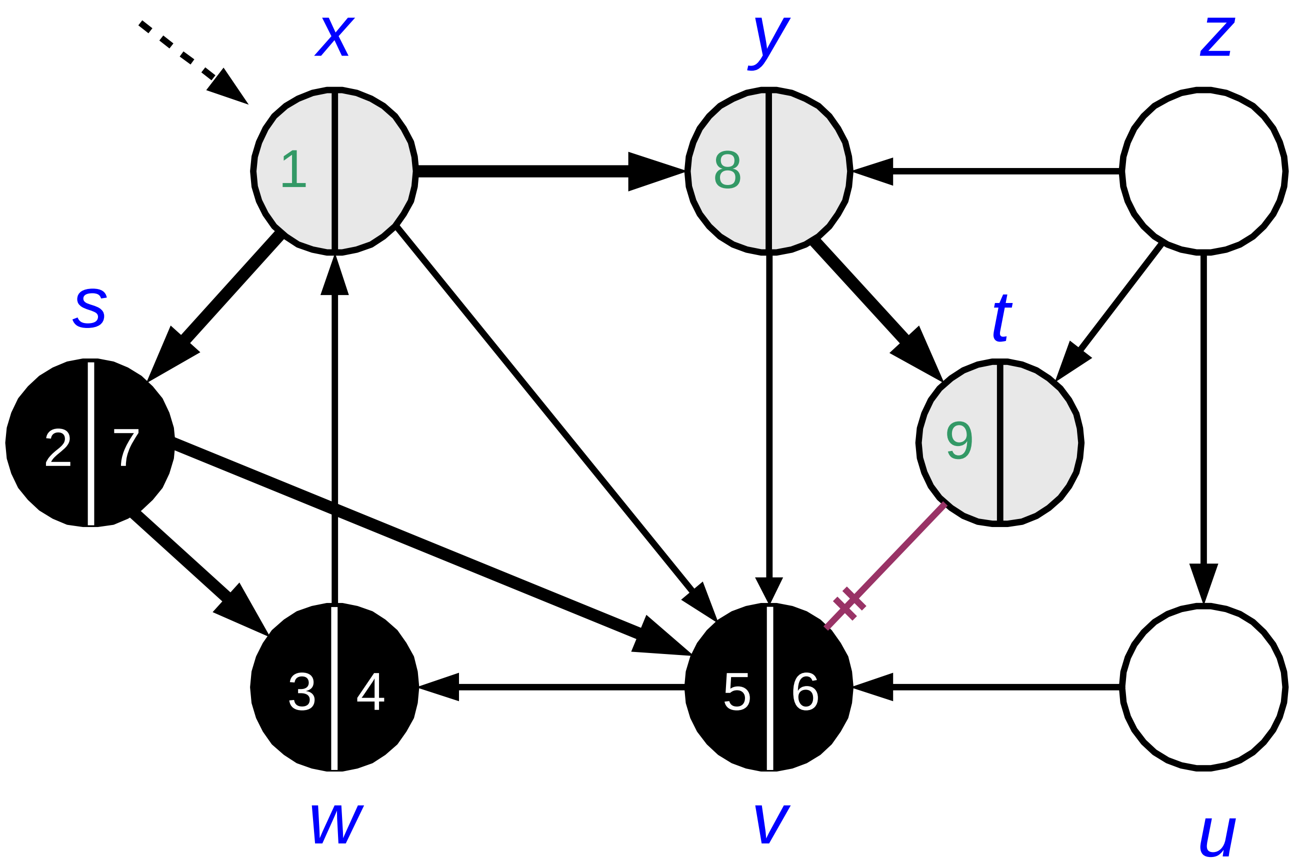
Depth-First Search: Example



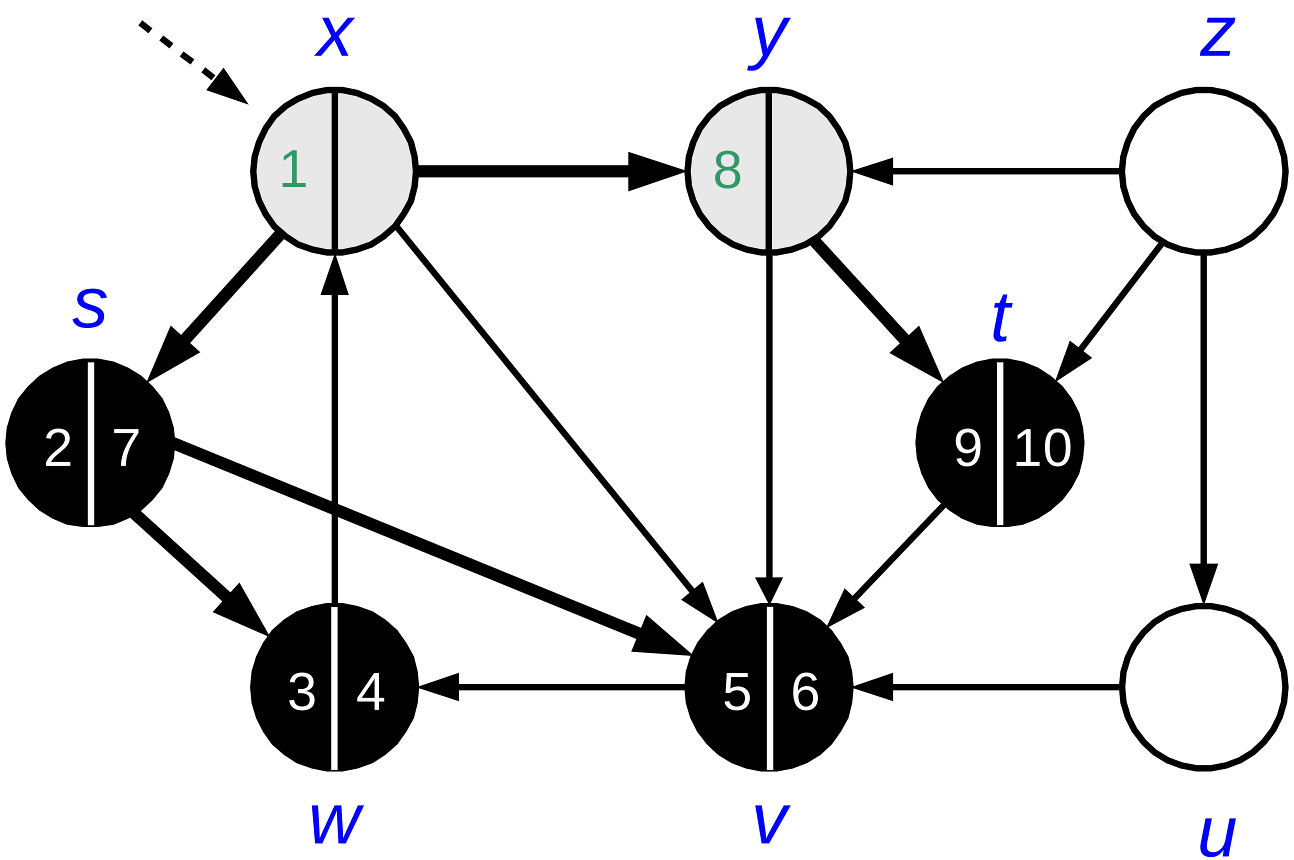
Depth-First Search: Example



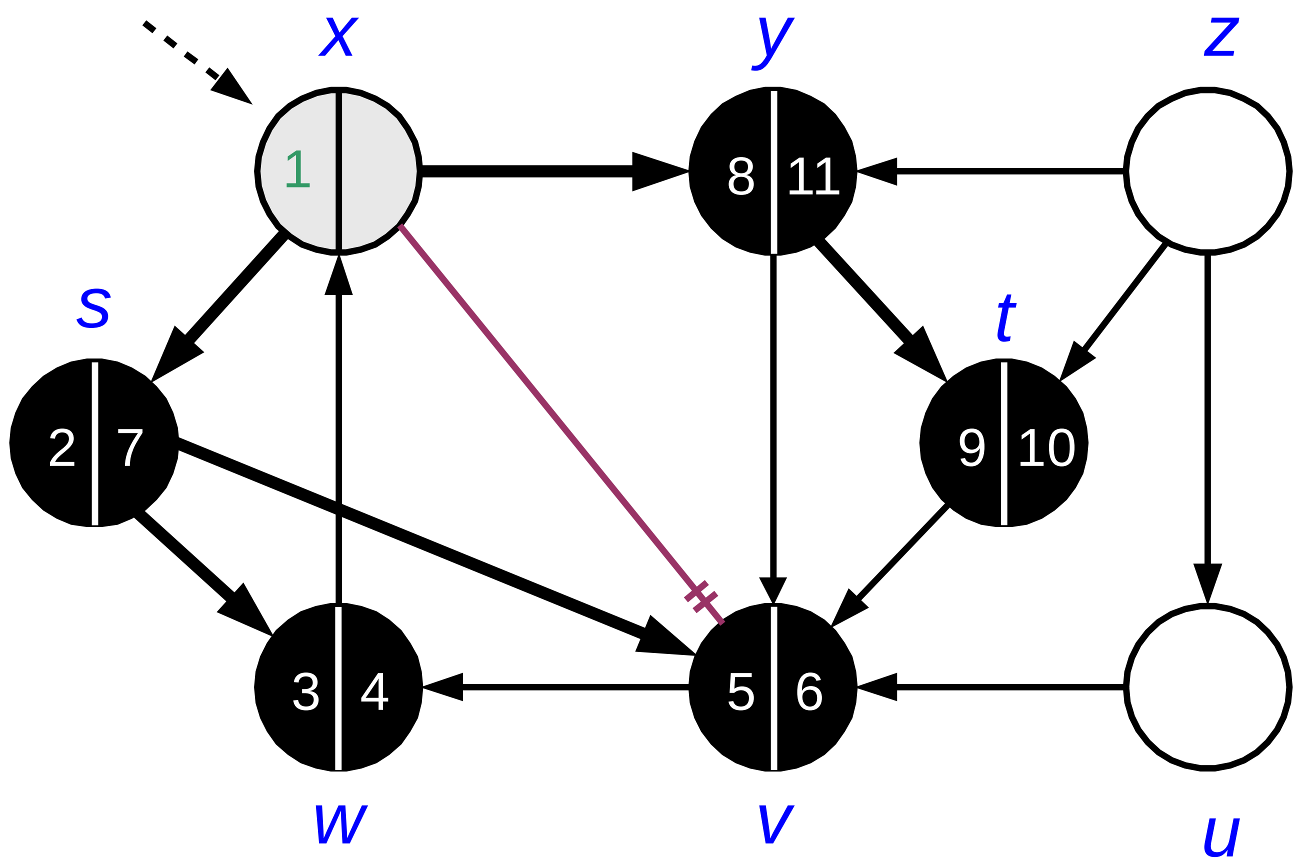
Depth-First Search: Example



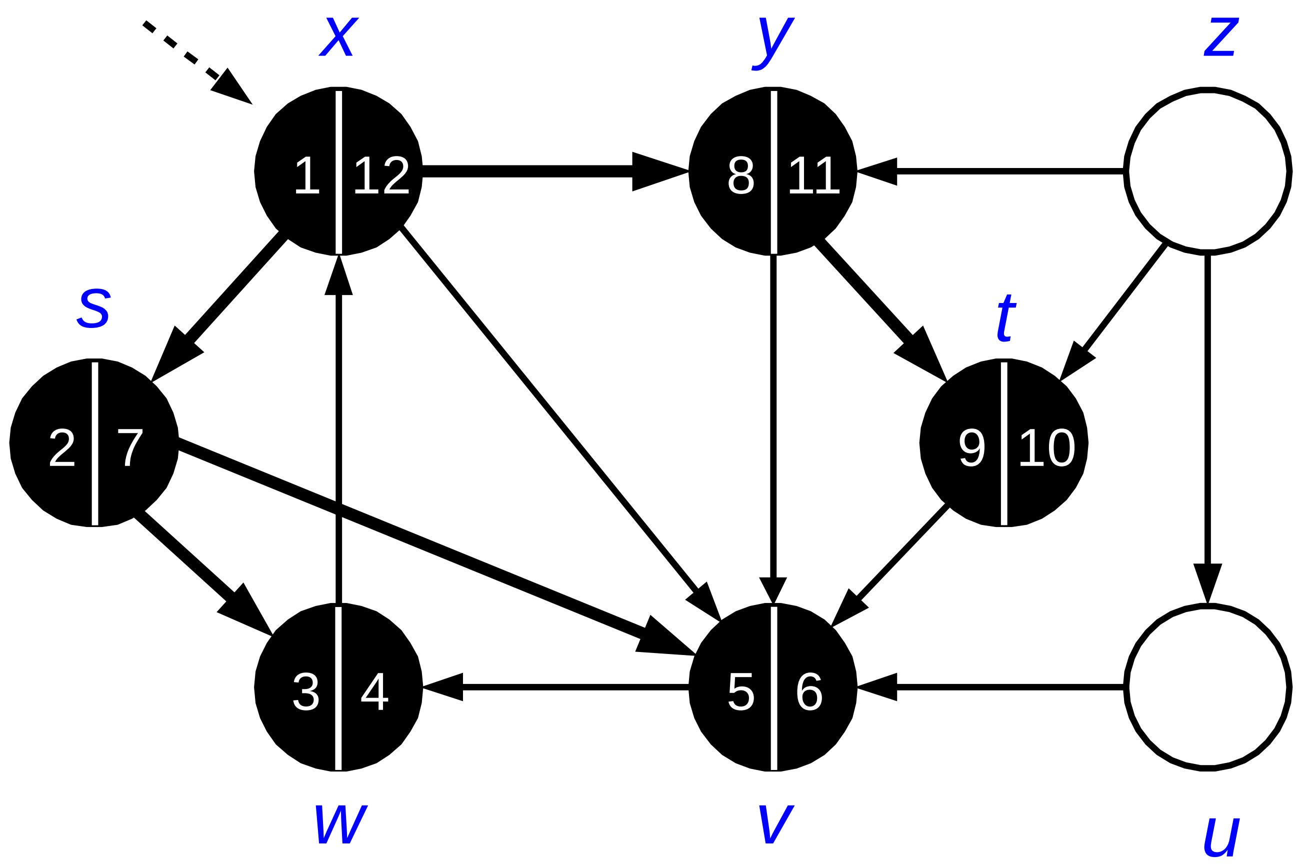
Depth-First Search: Example



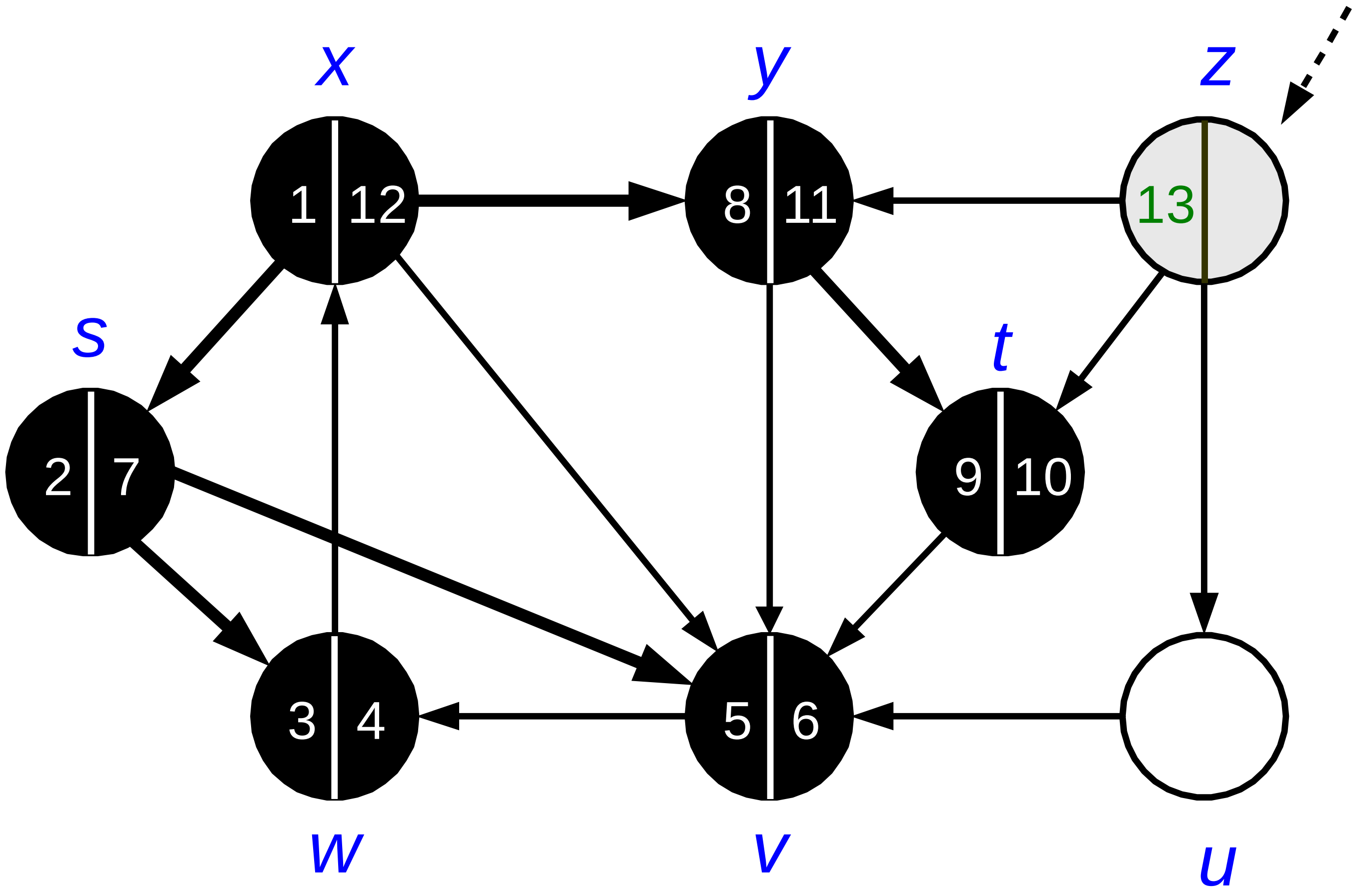
Depth-First Search: Example



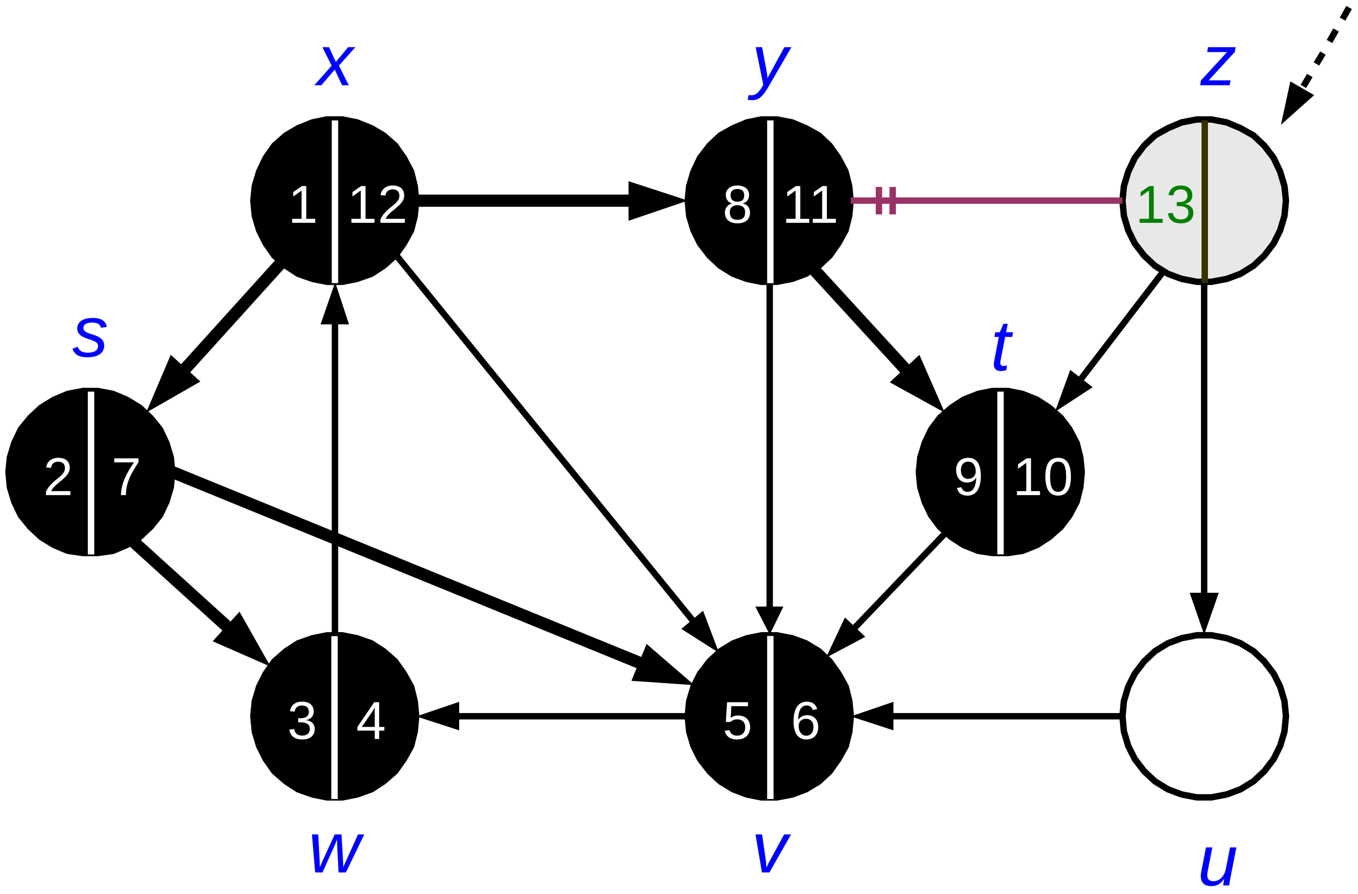
Depth-First Search: Example



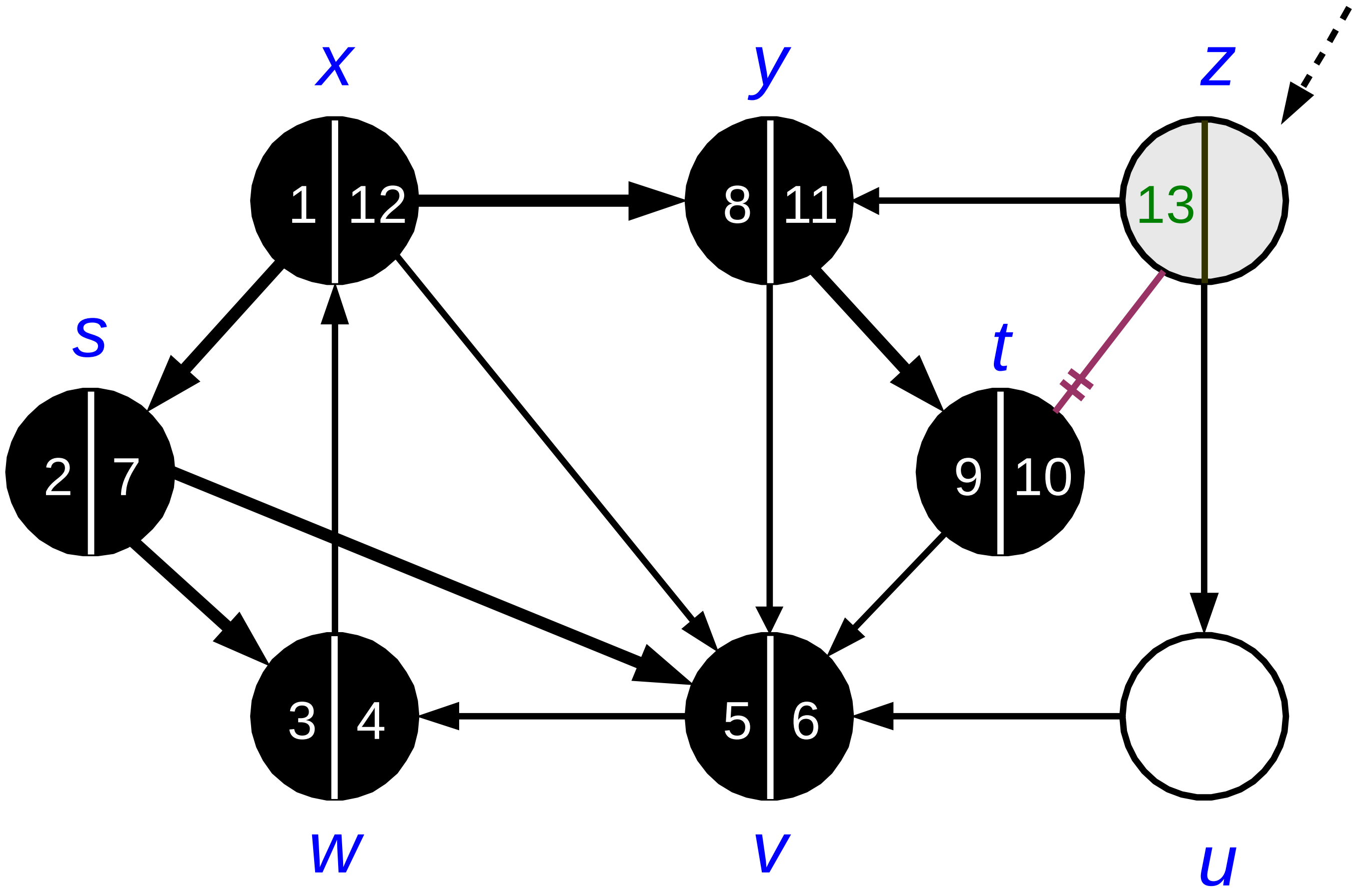
Depth-First Search: Example



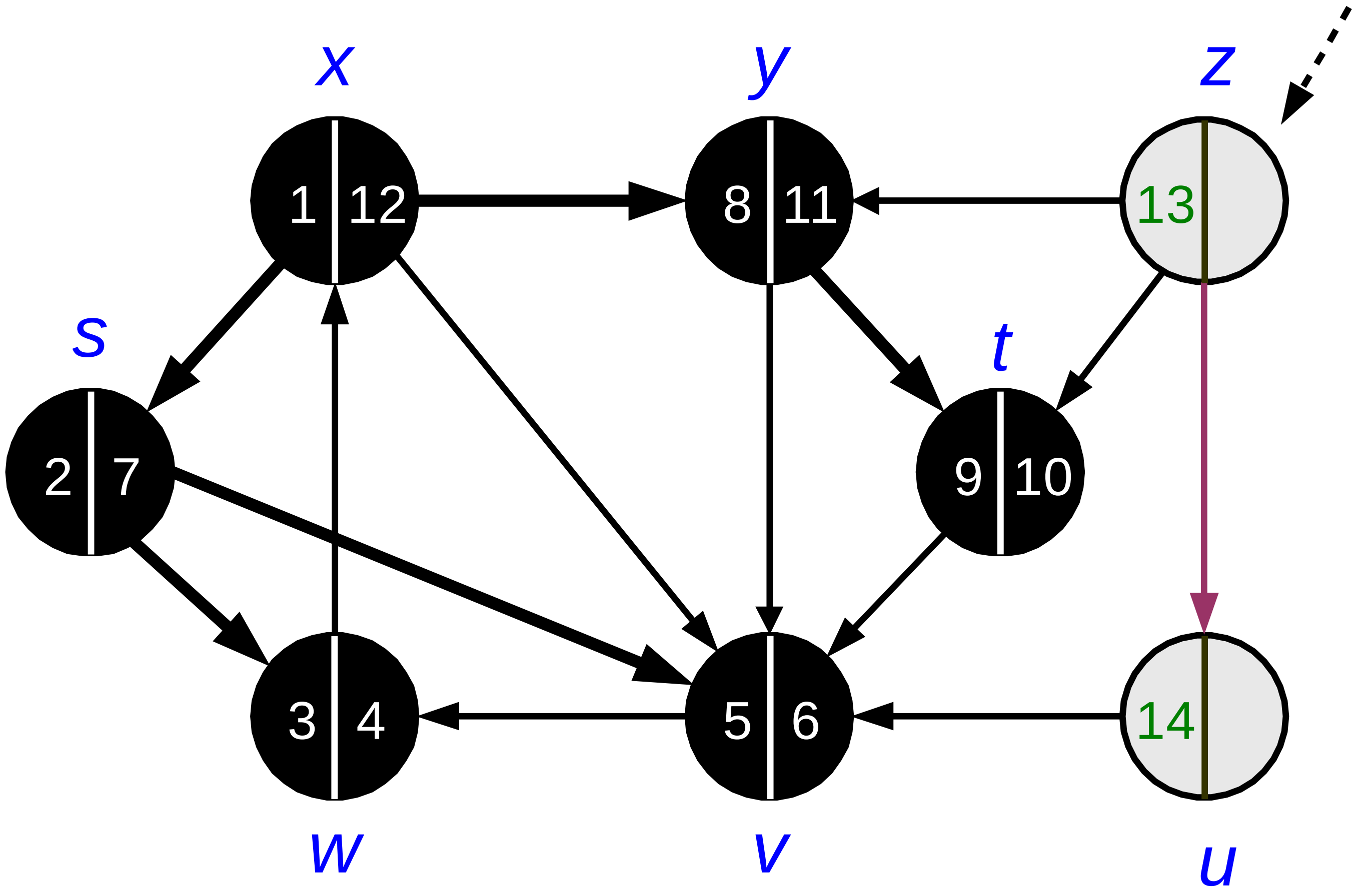
Depth-First Search: Example



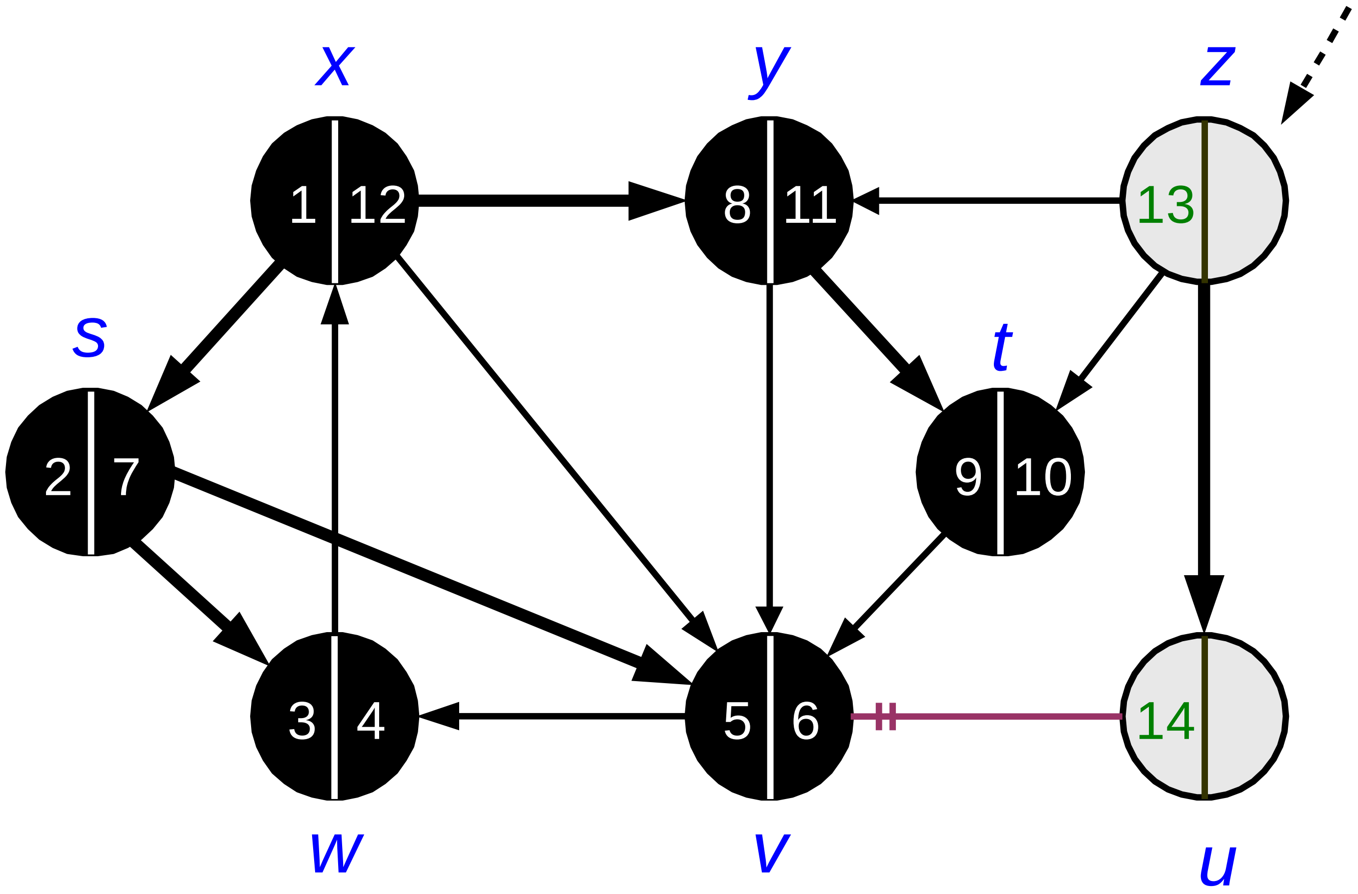
Depth-First Search: Example



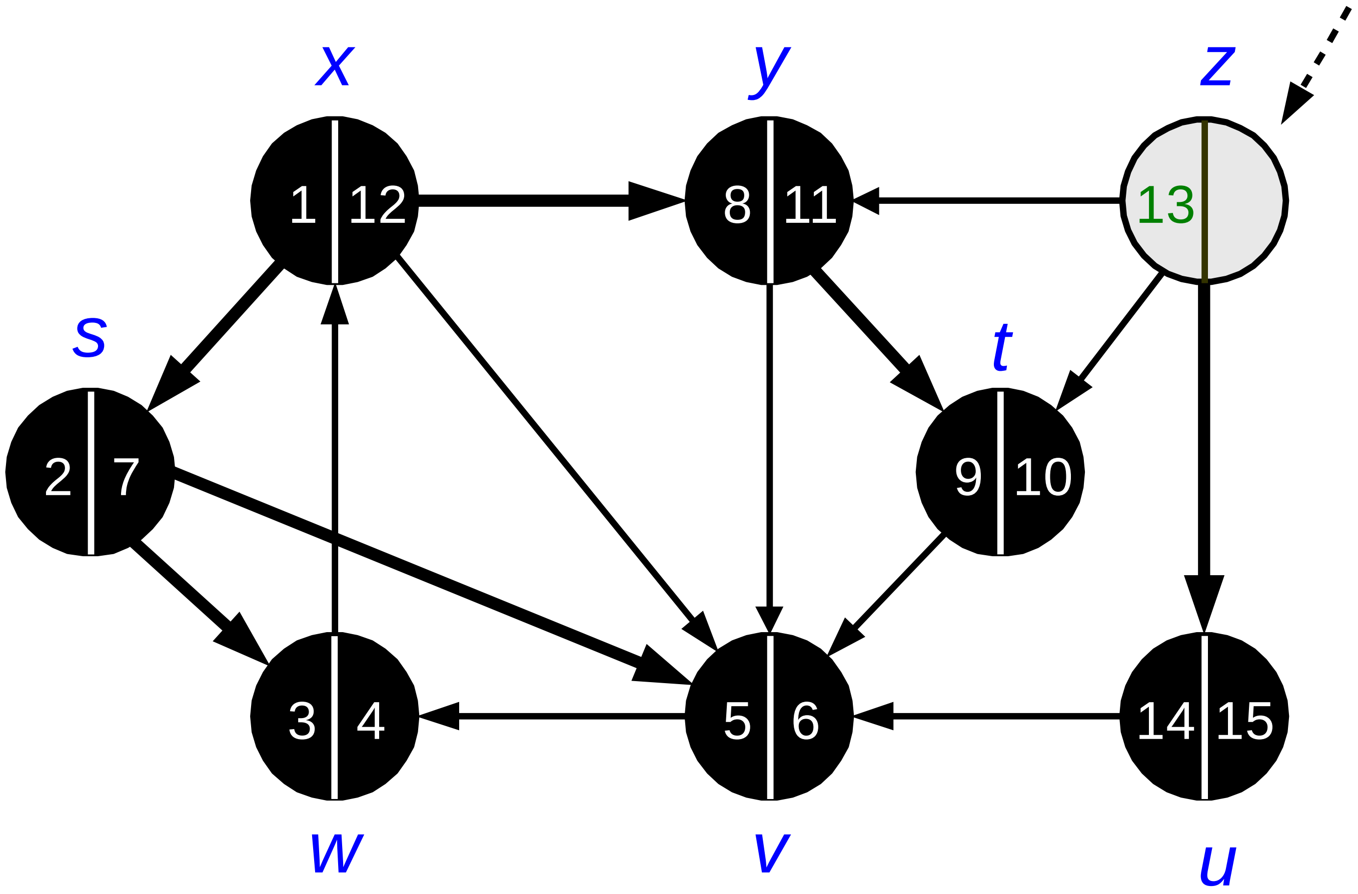
Depth-First Search: *Example*



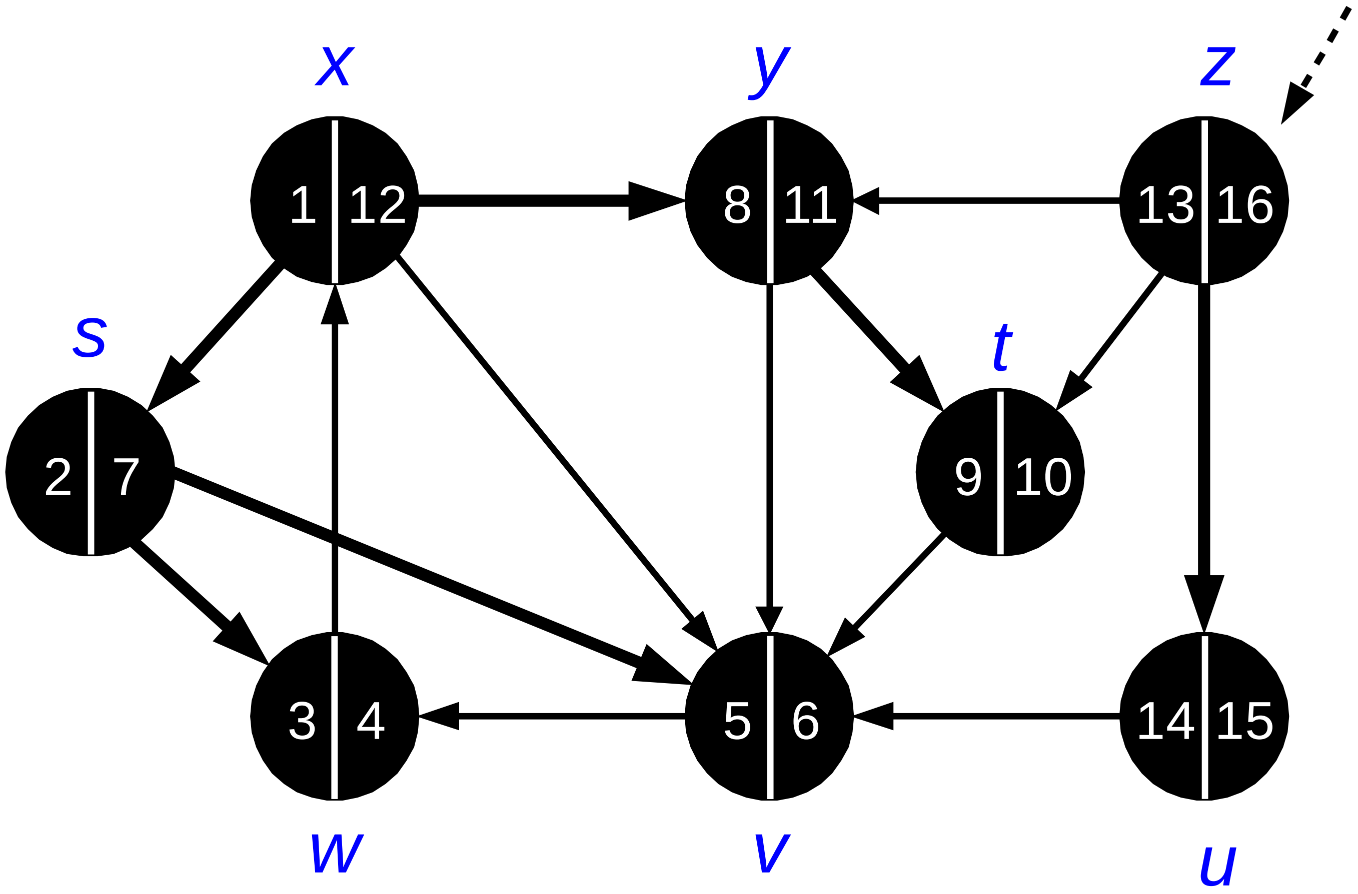
Depth-First Search: Example



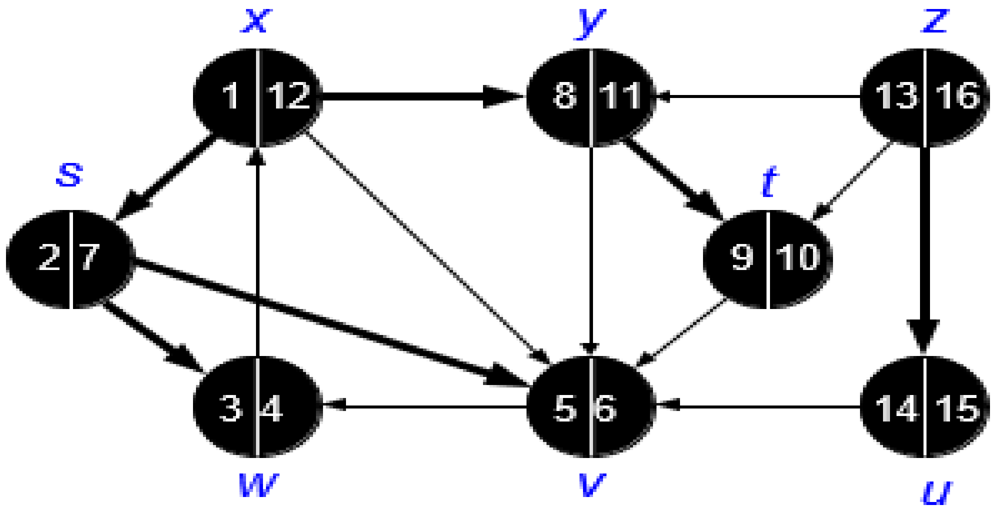
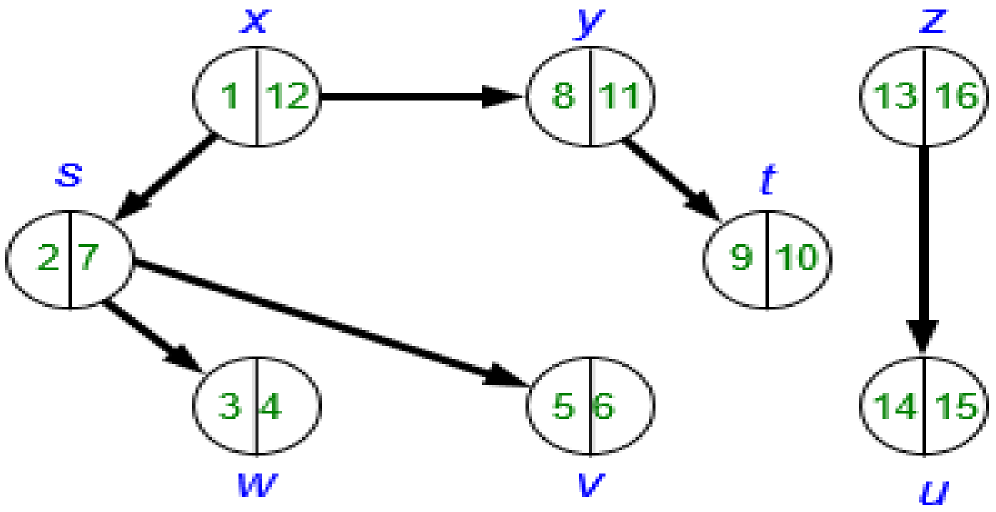
Depth-First Search: Example



Depth-First Search: Example



Depth-First Search: Example

DFS(G) terminated	Depth-first forest (DFF)
	

DFS: Algorithm

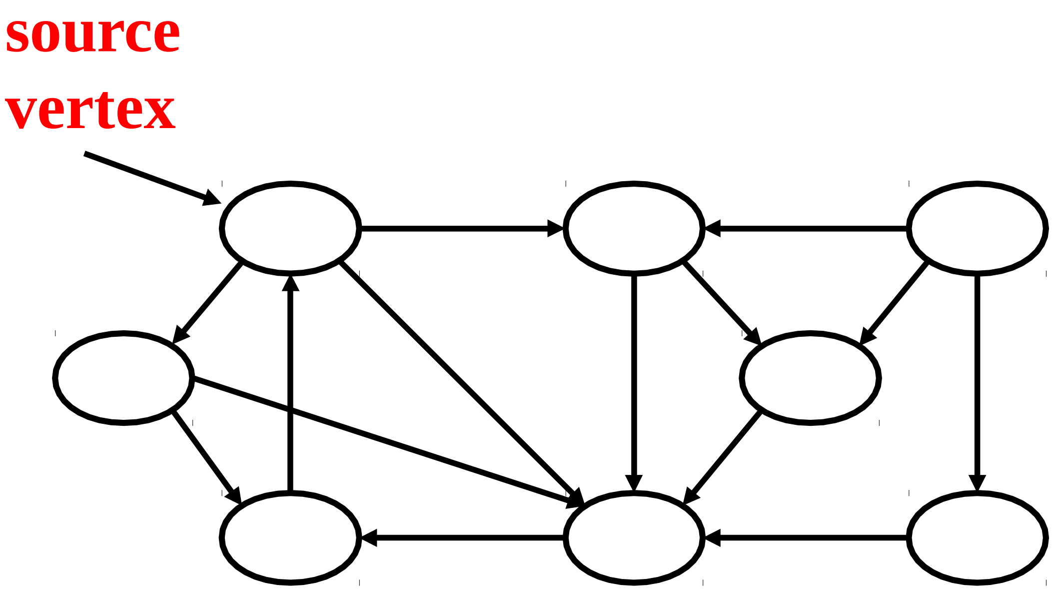
- DFS(G)

1. for each vertex u € V[G]
2. color[u]=white
3. π[u]=NIL
4. time = 0
5. for each vertex u € V[G]
6. if (color[u]= =white)
7. DFS-VISIT(G,u)

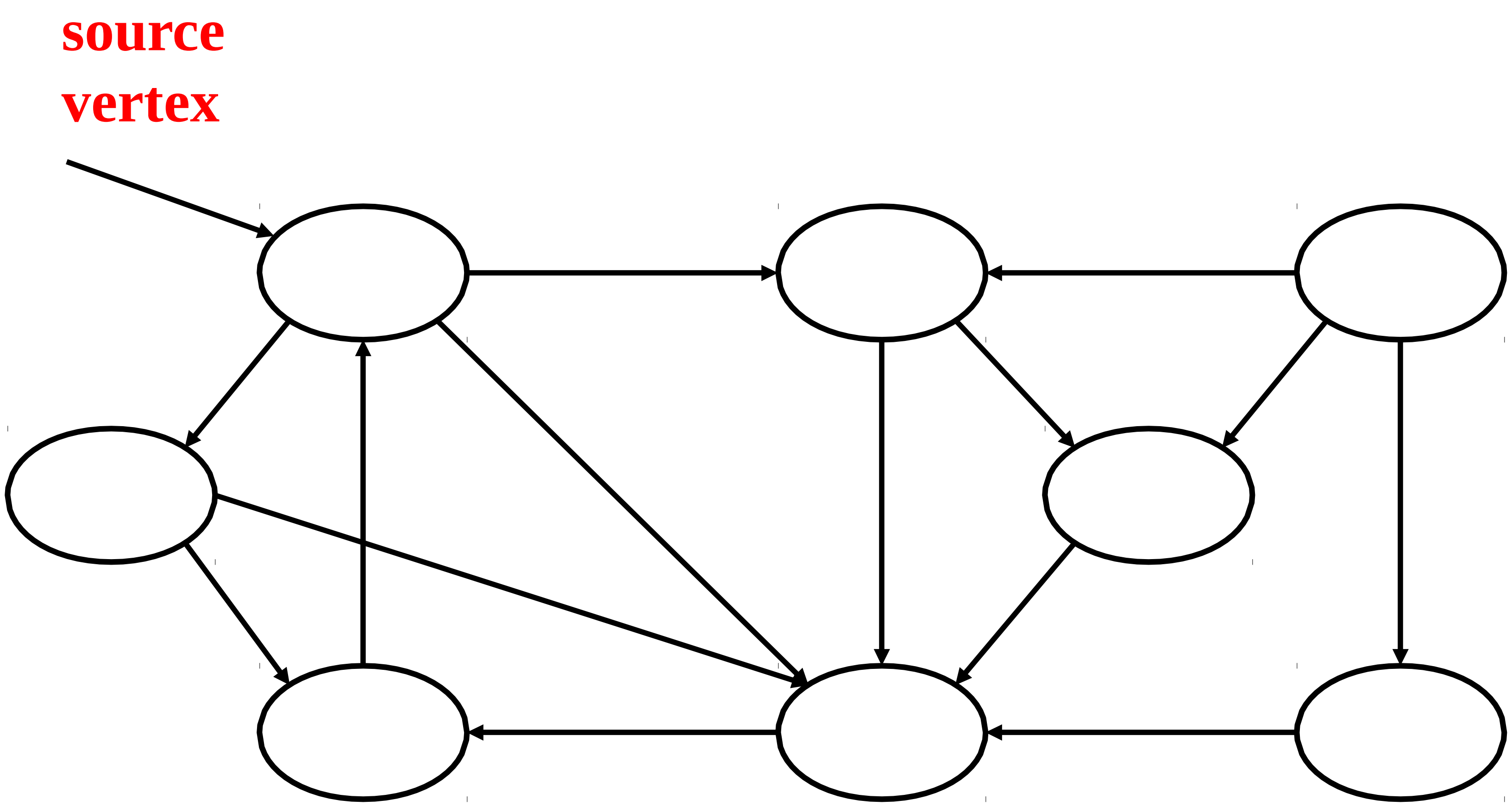
DFS: Algorithm (Cont.)

- DFS-VISIT(u)

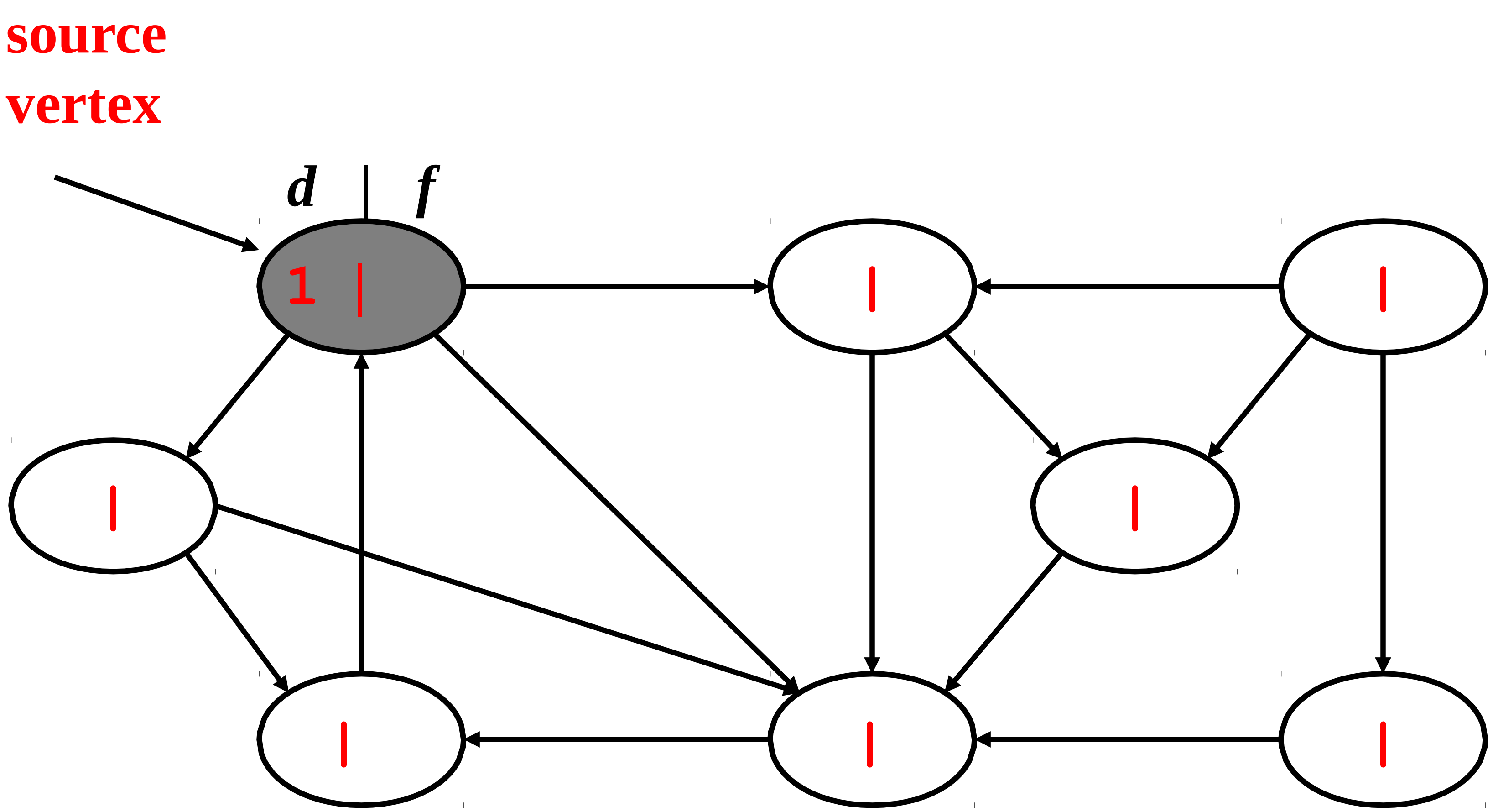
1. time = time + 1
2. d[u] = time
3. color[u] = gray
4. for each v € Adj(u) in **G** do
5. if (color[v] == white)
6. π [v] = u;
7. DFS-VISIT(G,v);
8. color[u] = black
9. time = time + 1;
10. f[u] = time;



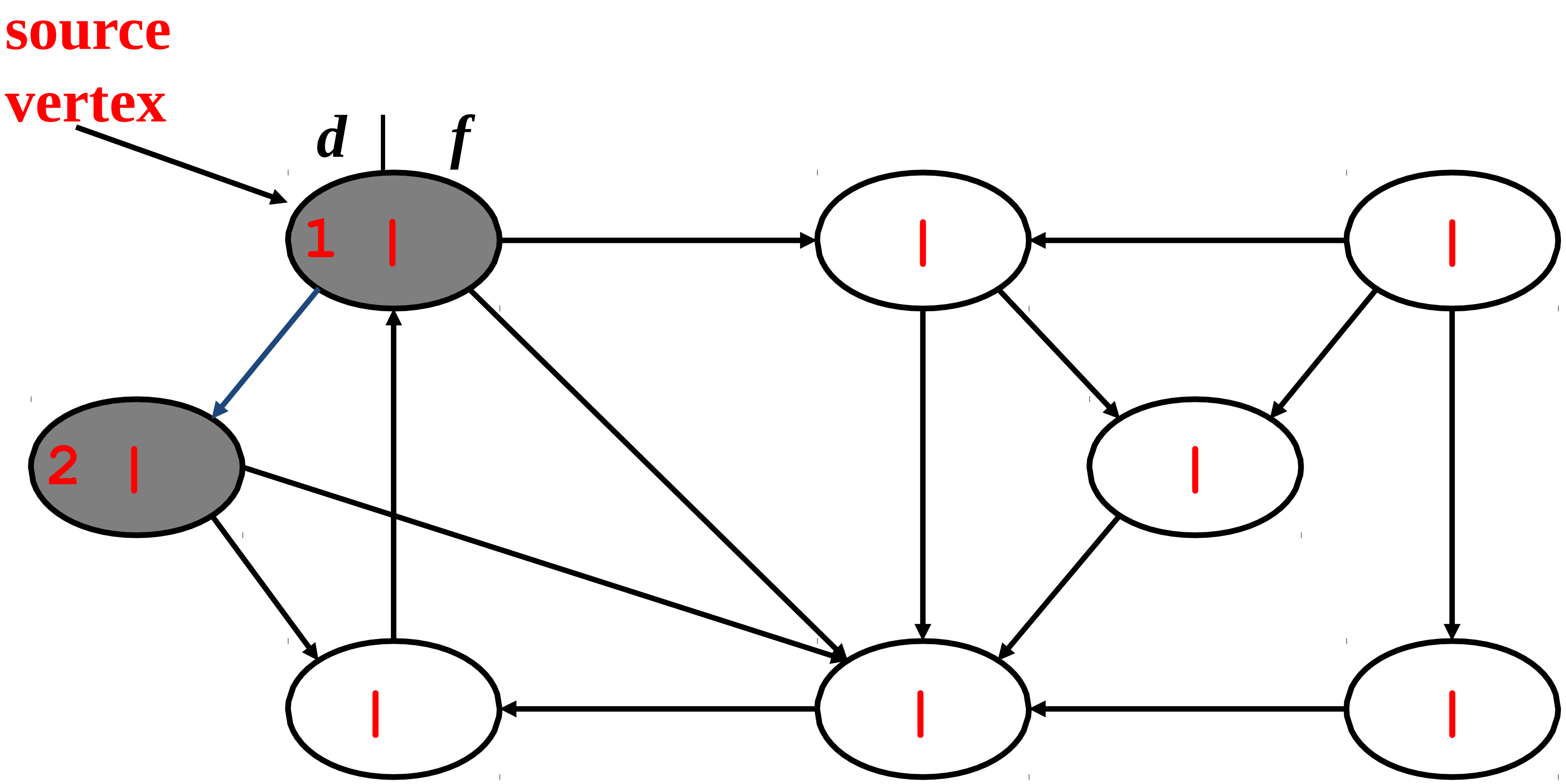
DFS Example



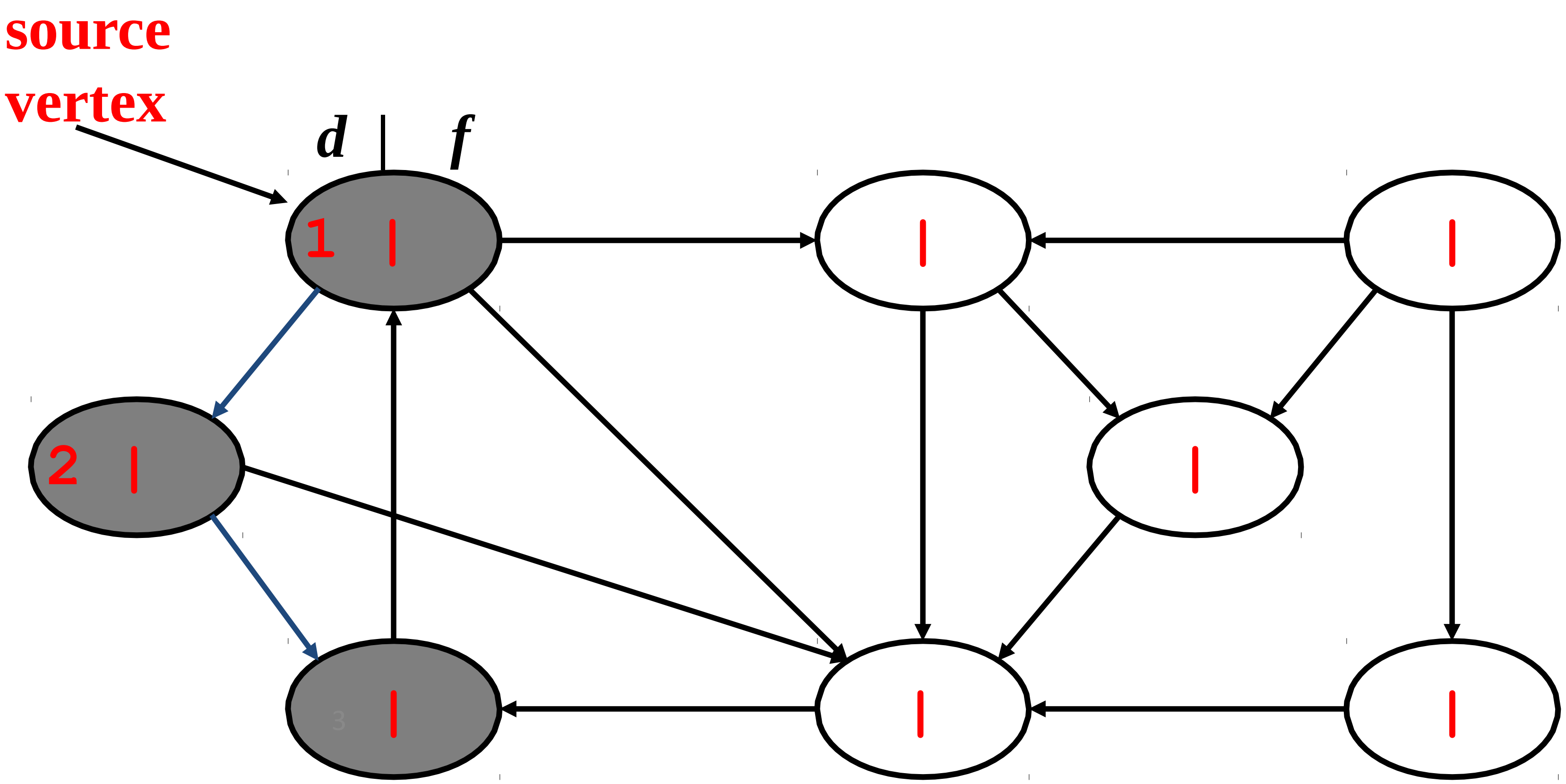
DFS Example



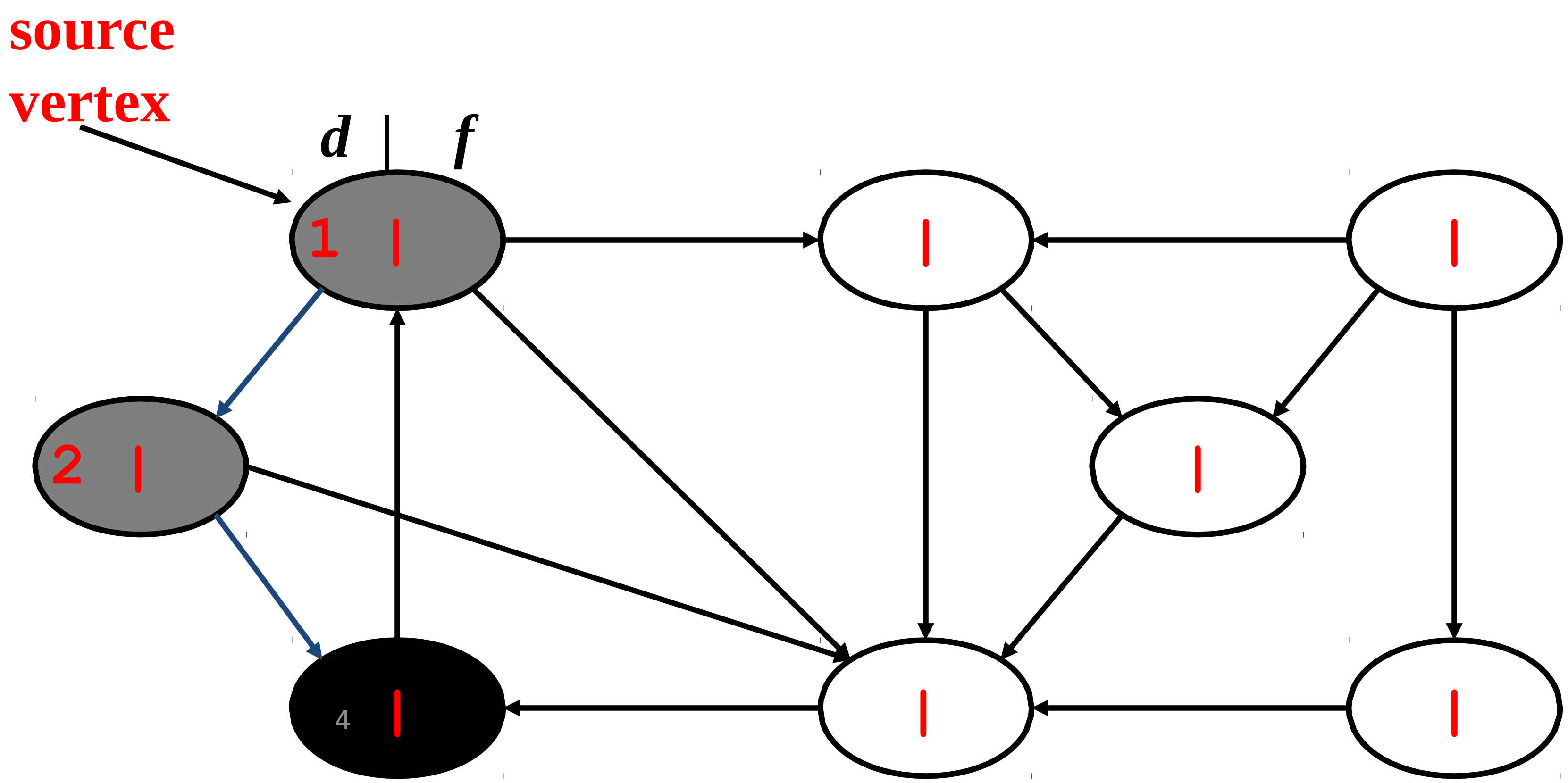
DFS Example



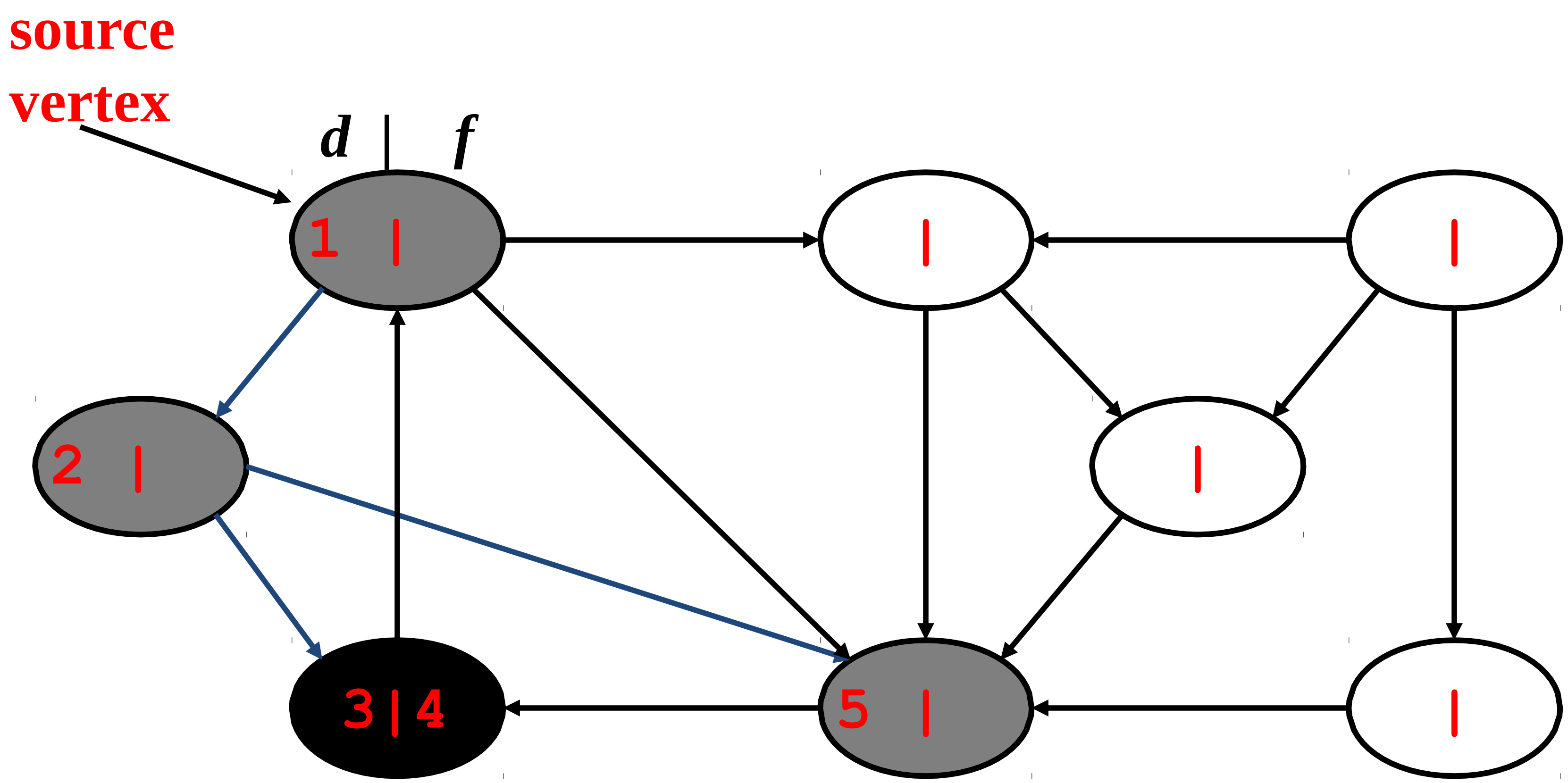
DFS Example



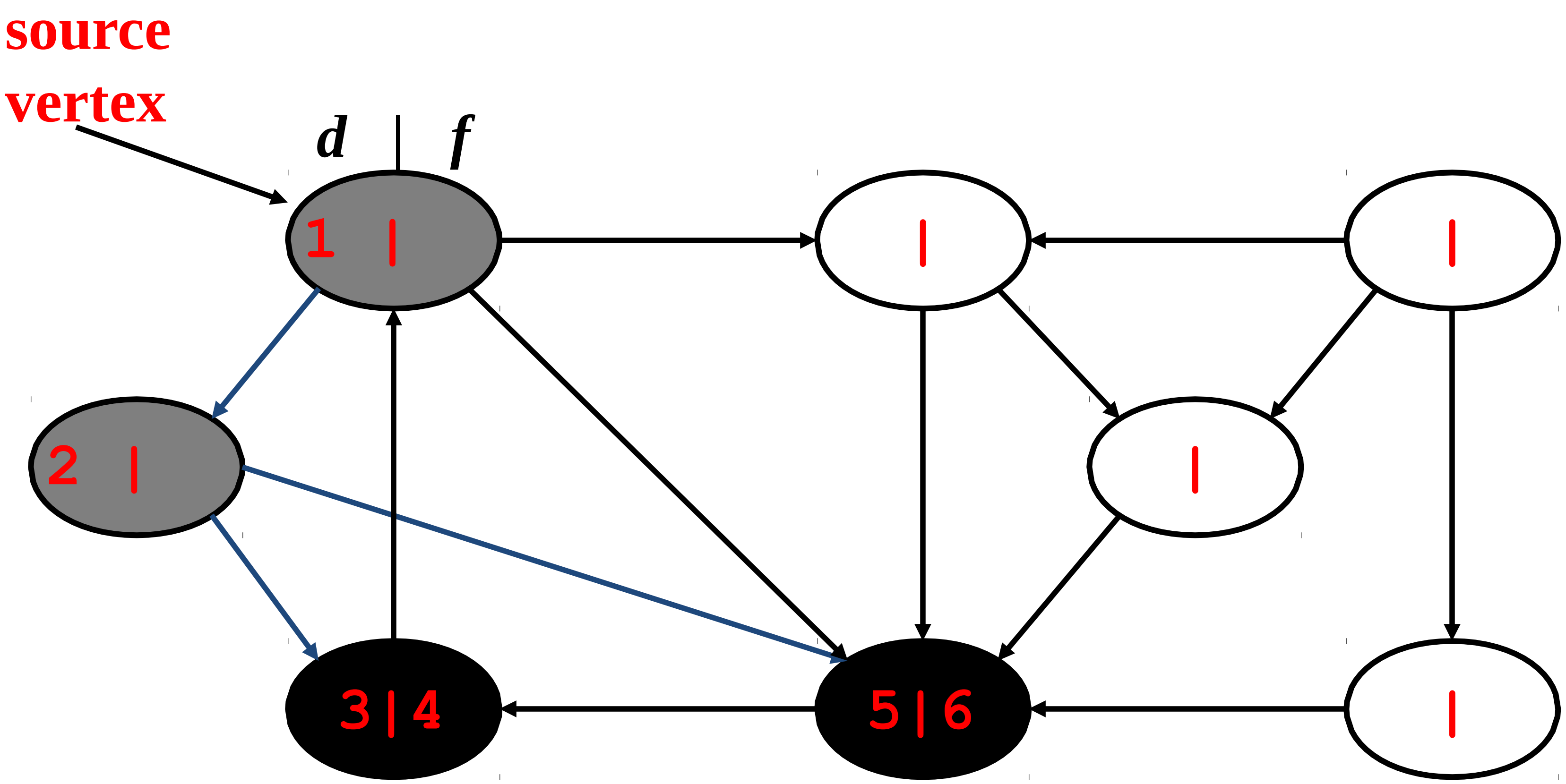
DFS Example



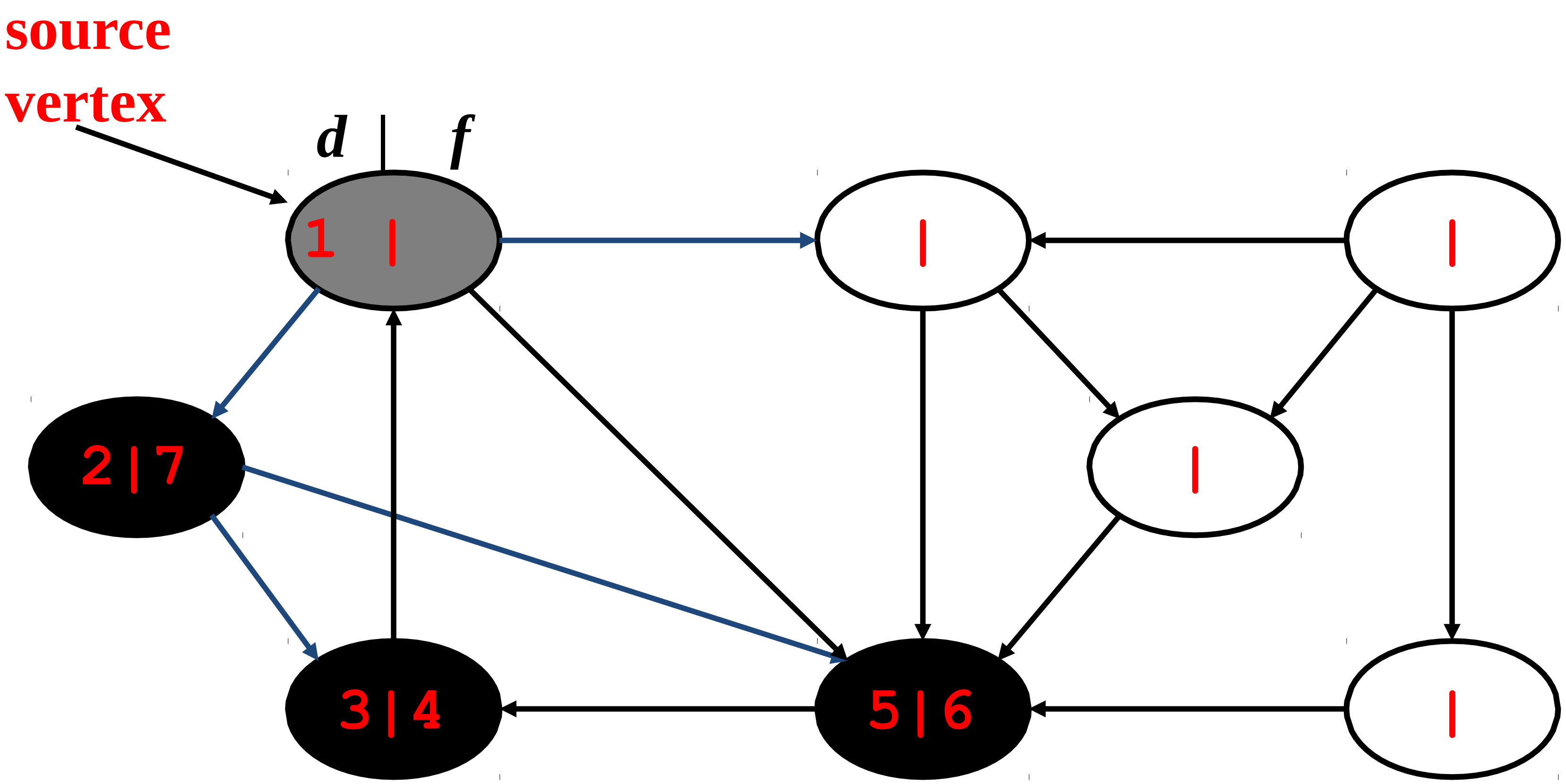
DFS Example



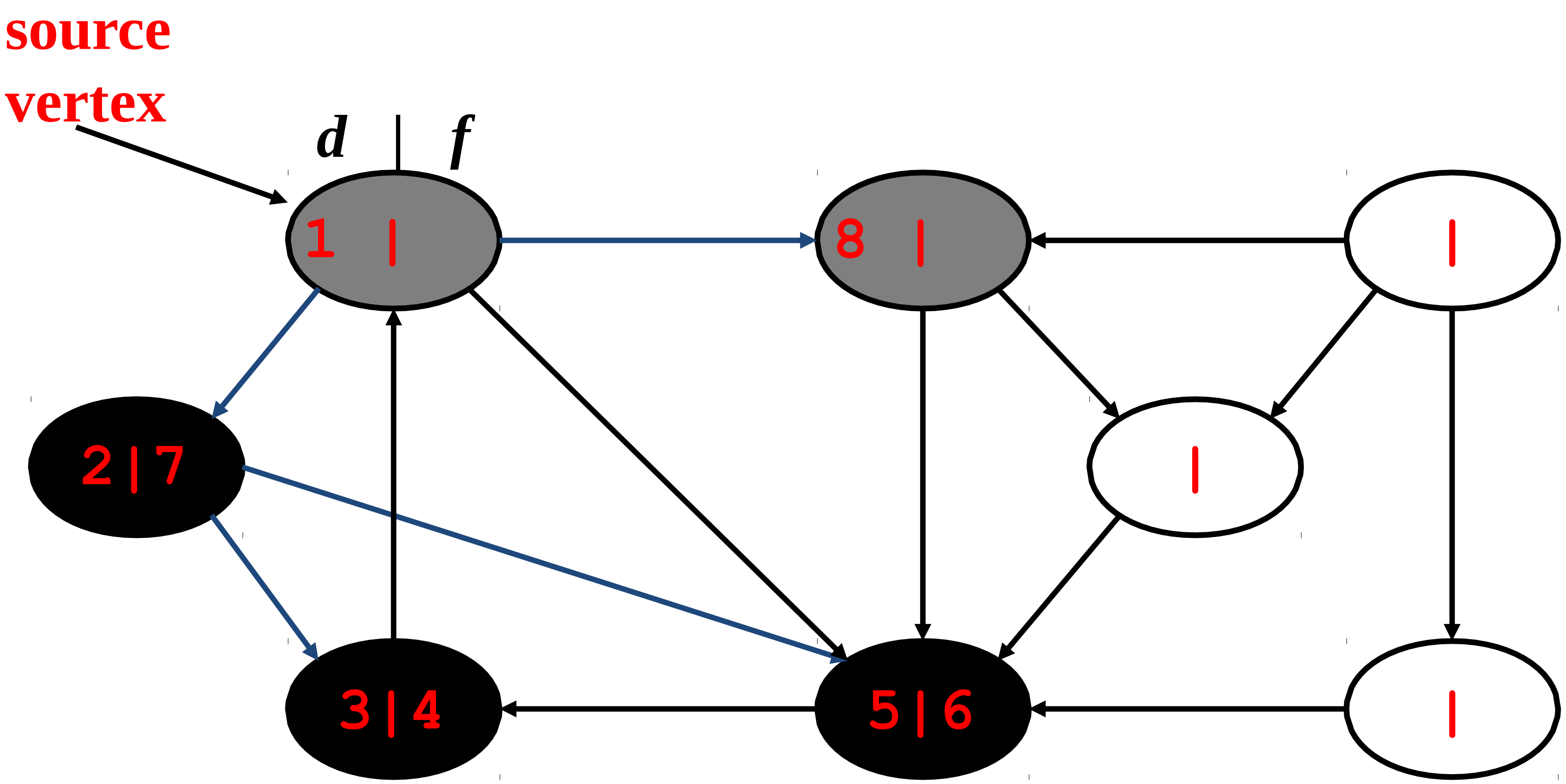
DFS Example



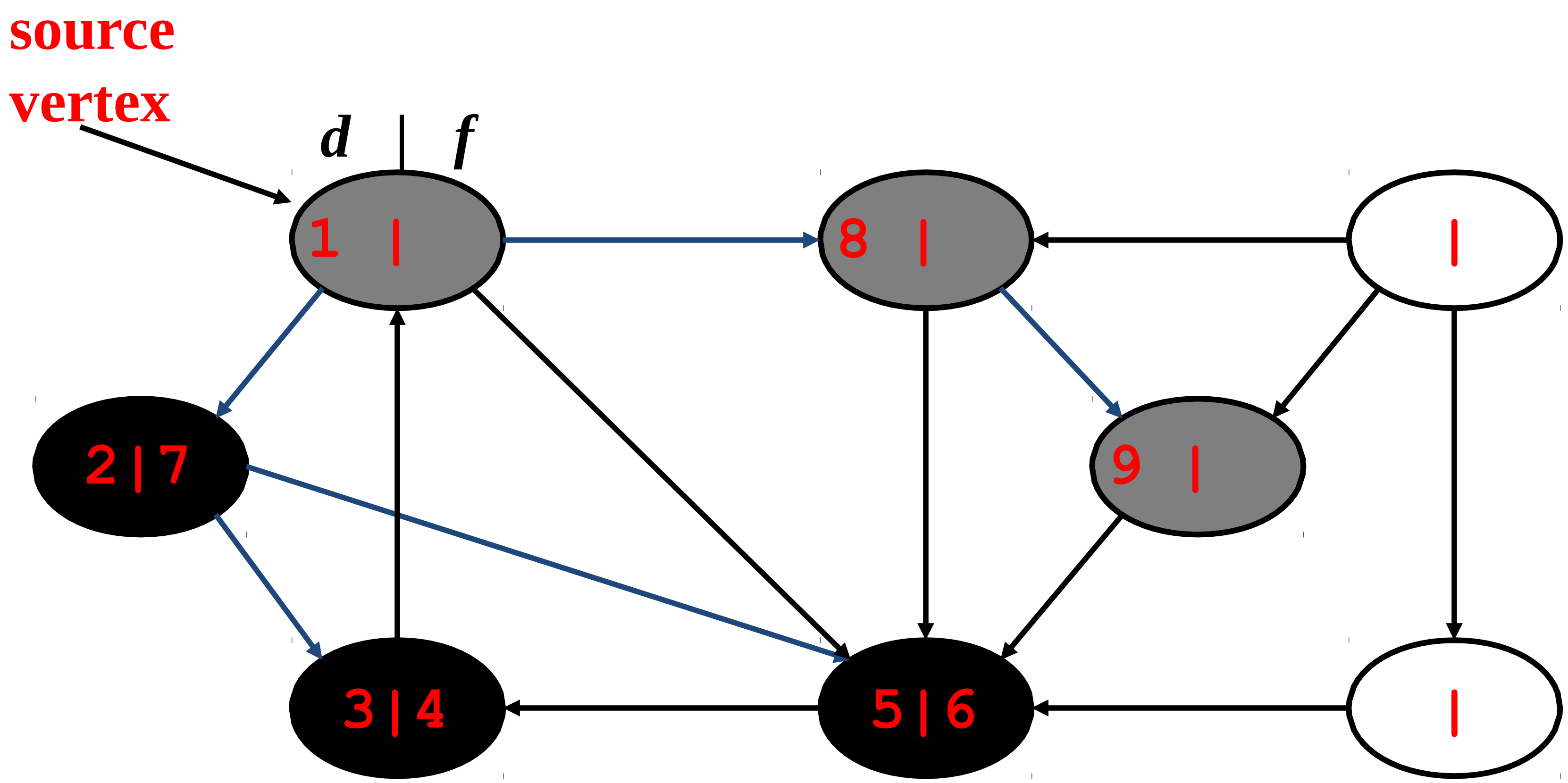
DFS Example



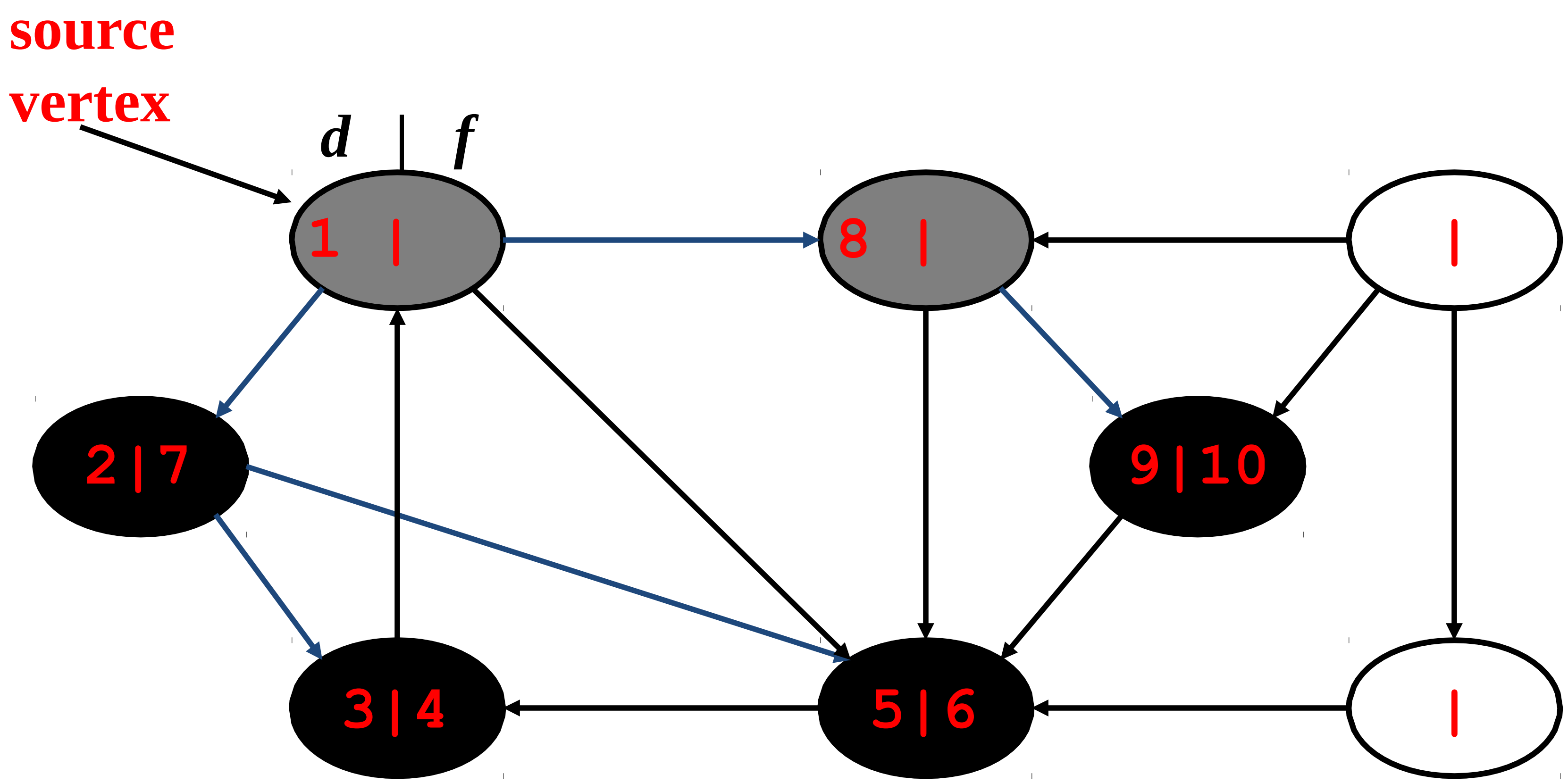
DFS Example



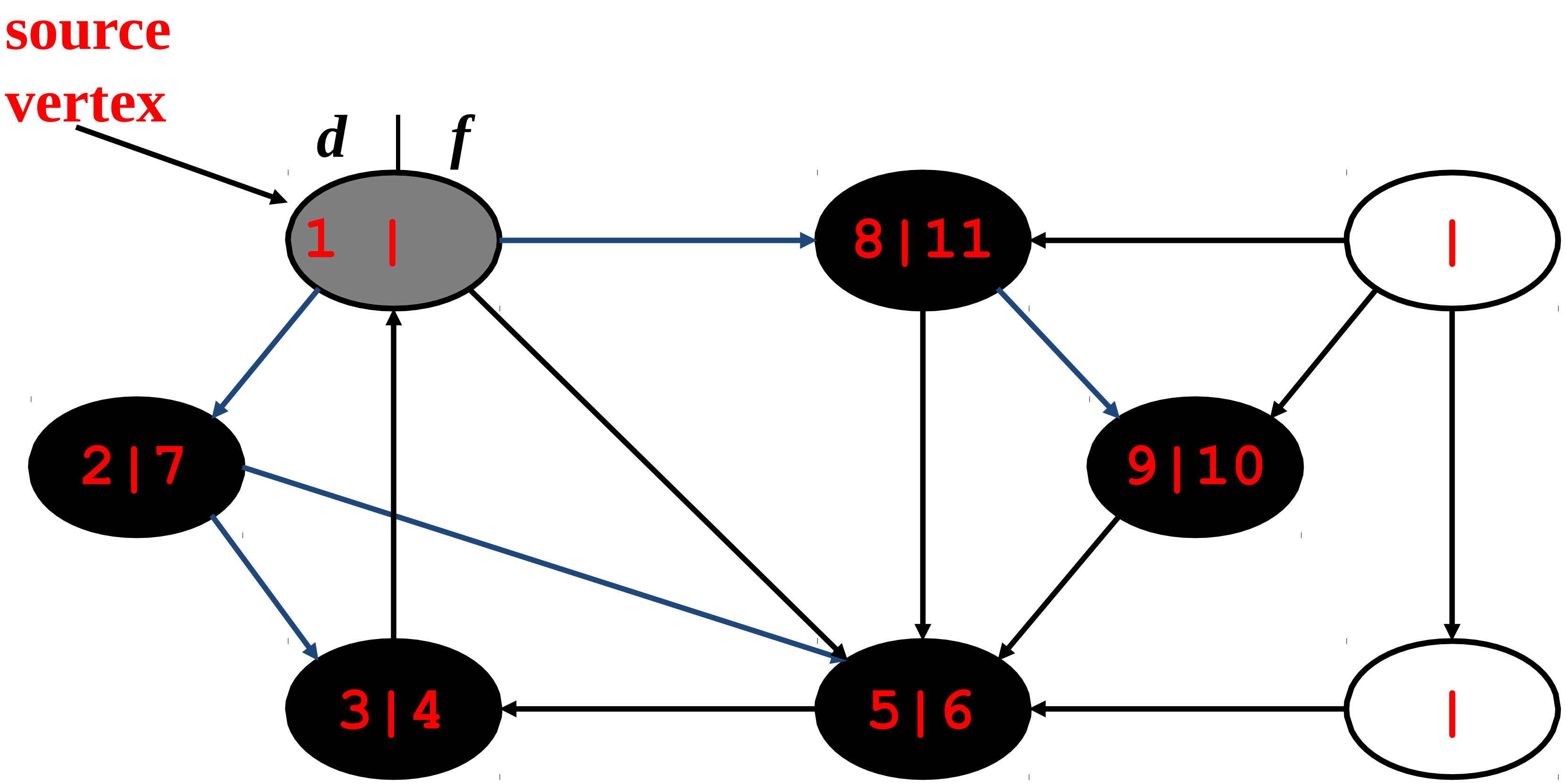
DFS Example



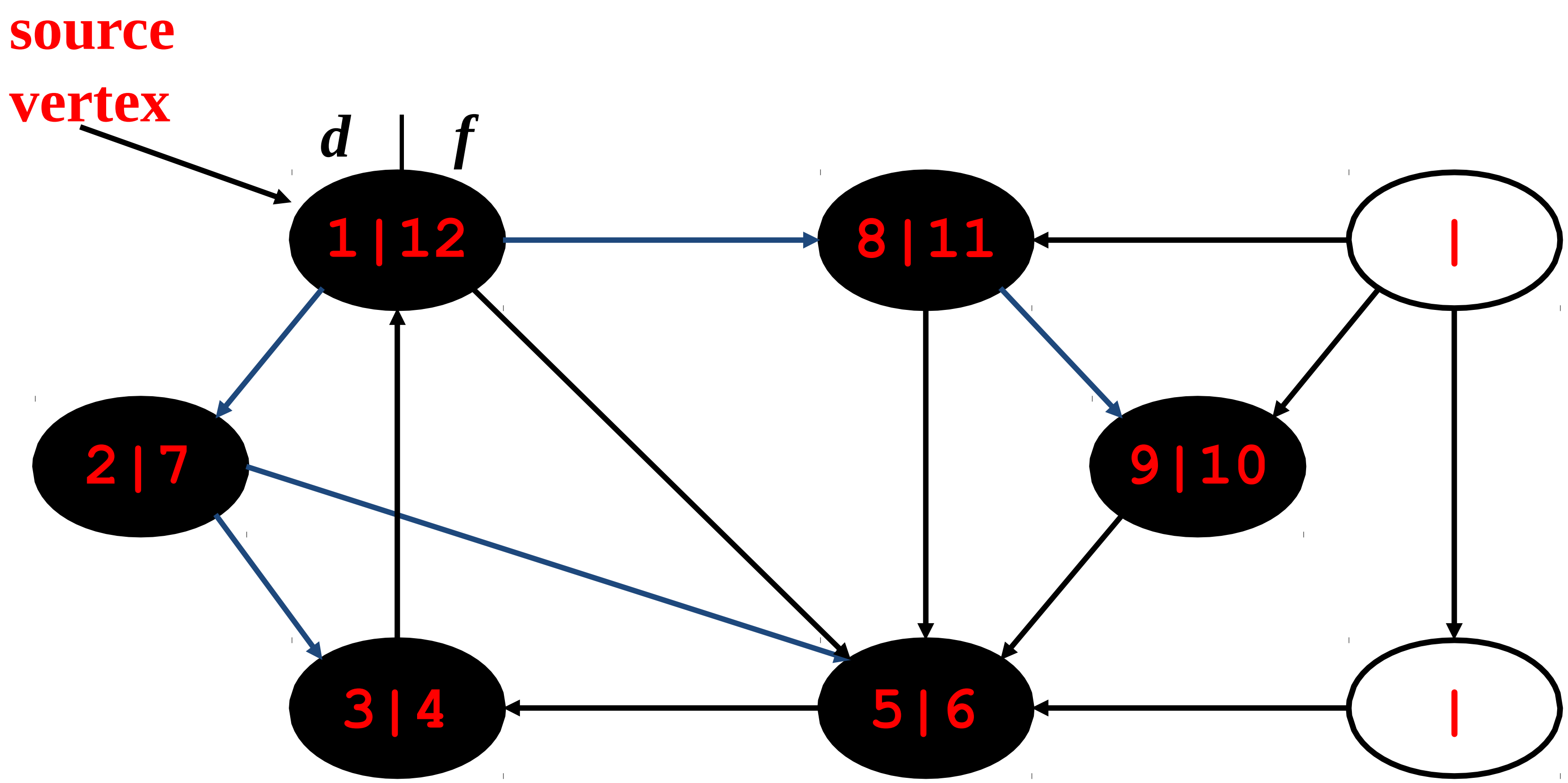
DFS Example



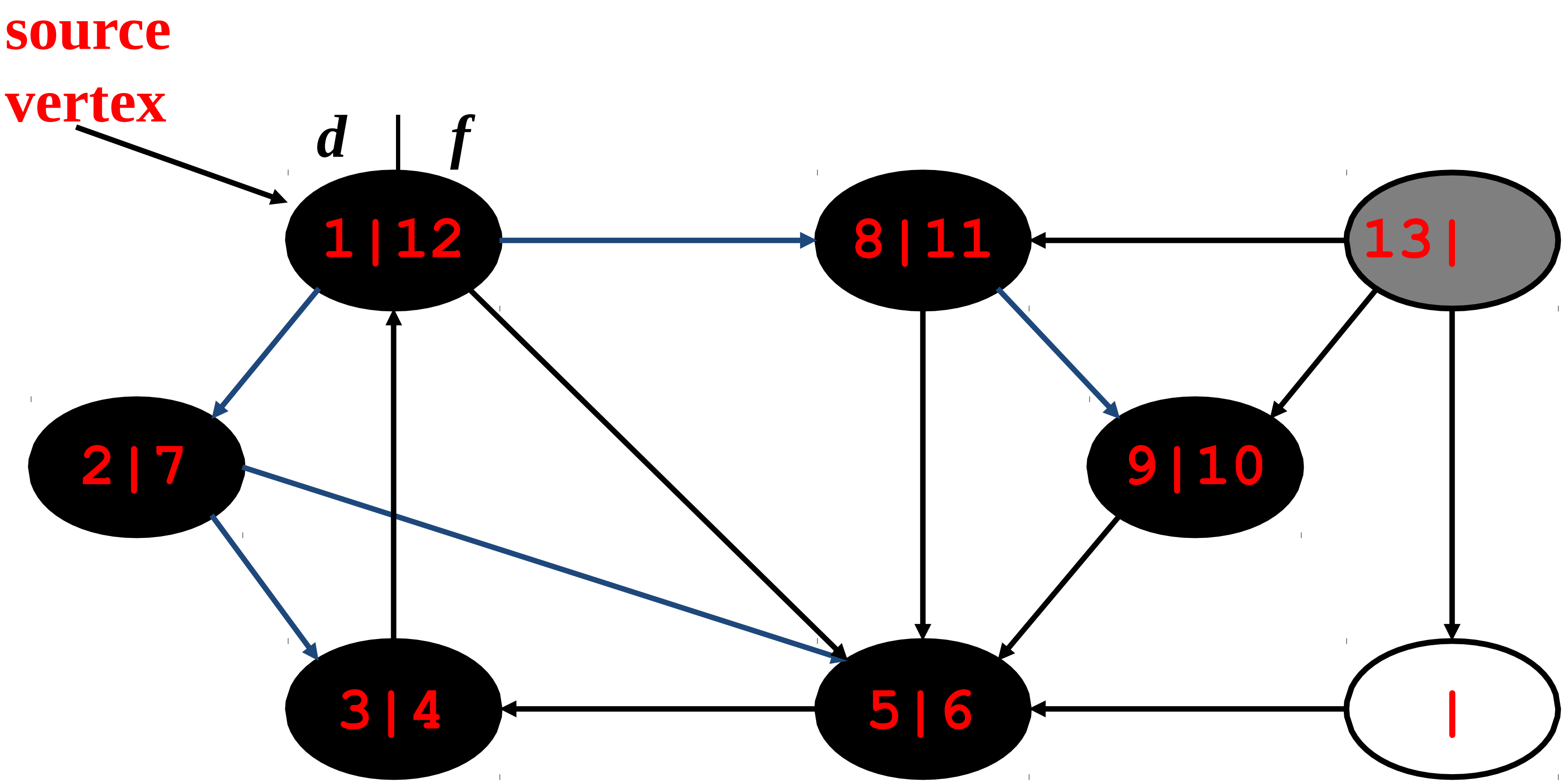
DFS Example



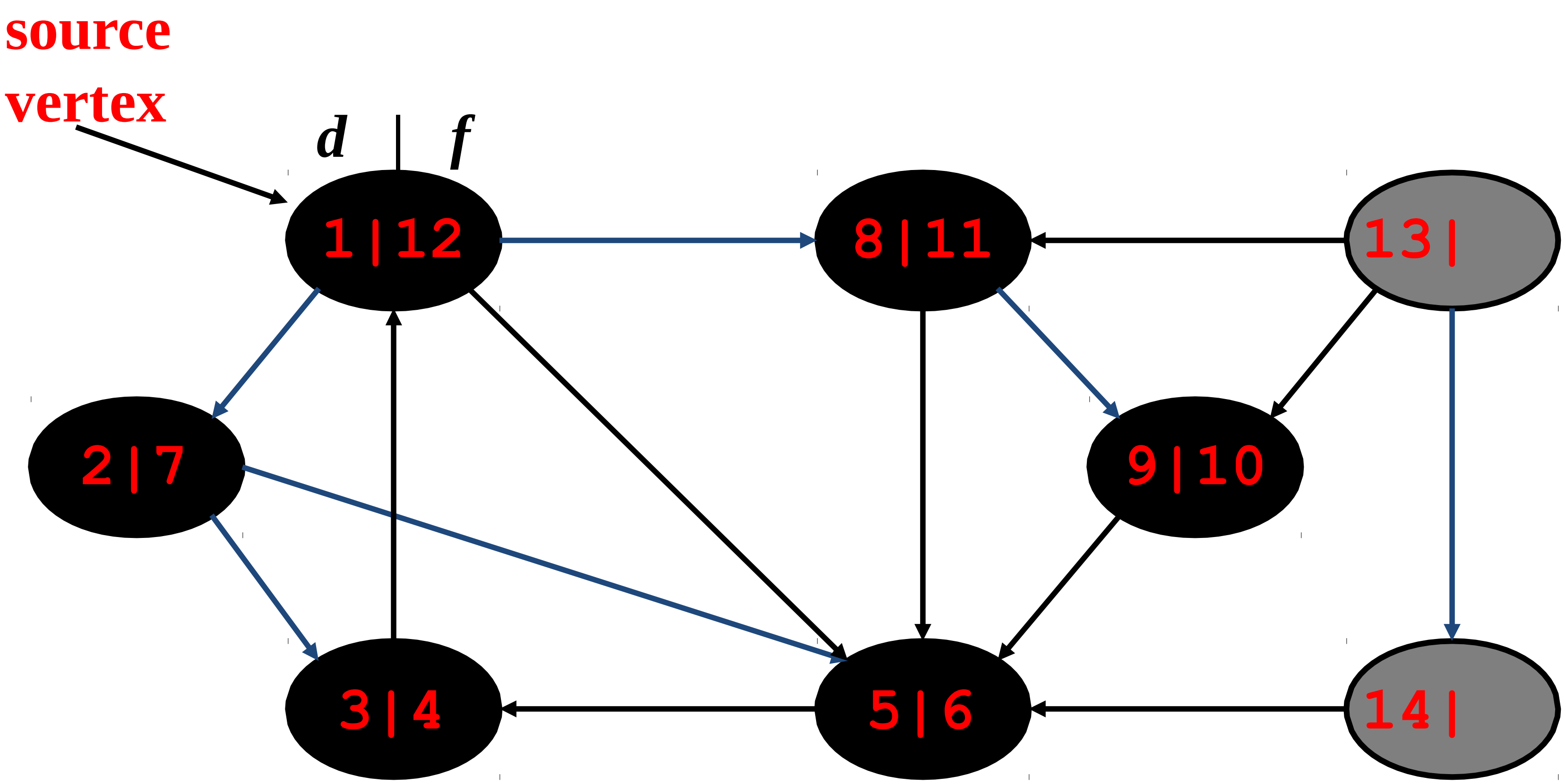
DFS Example



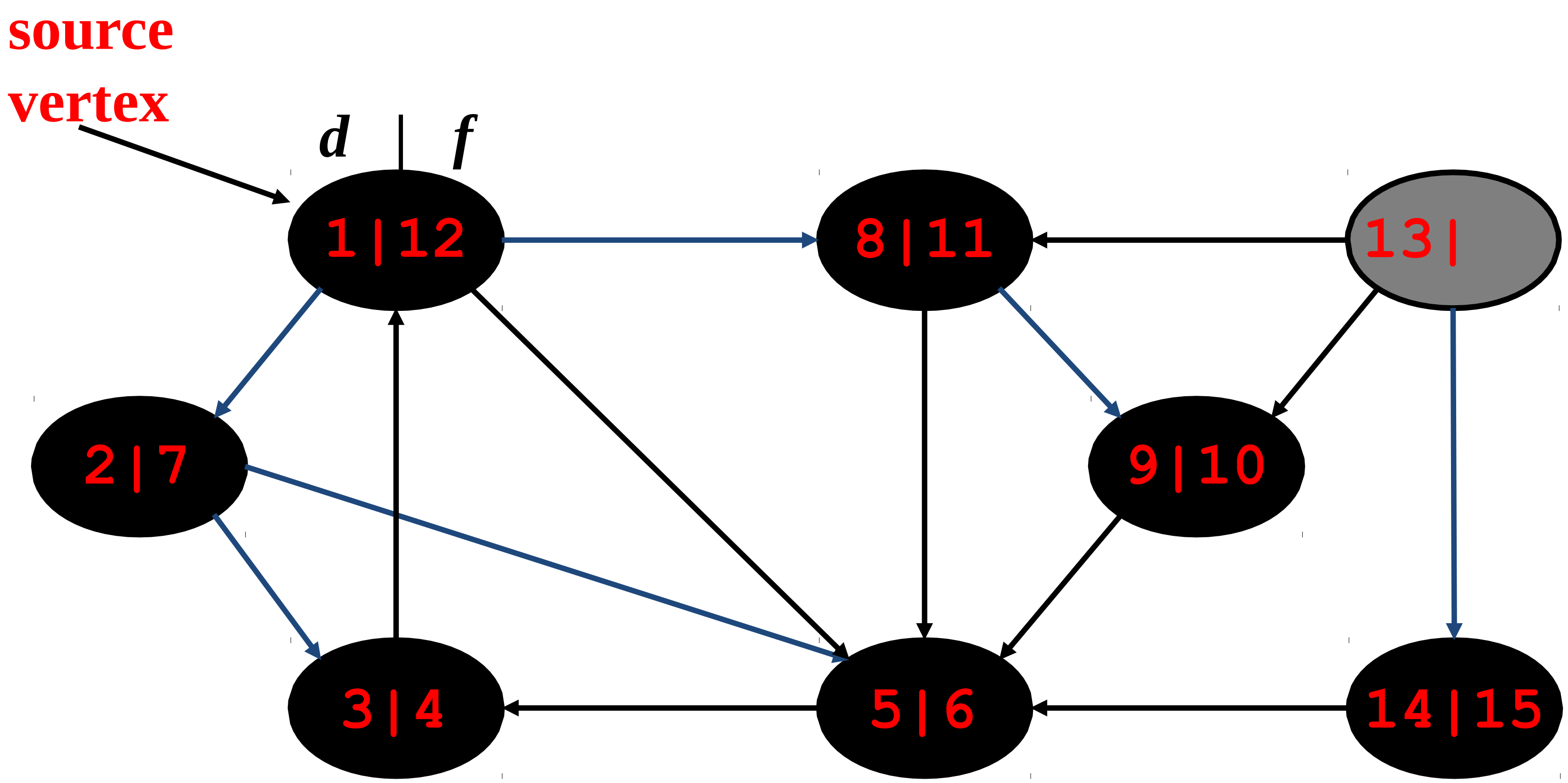
DFS Example



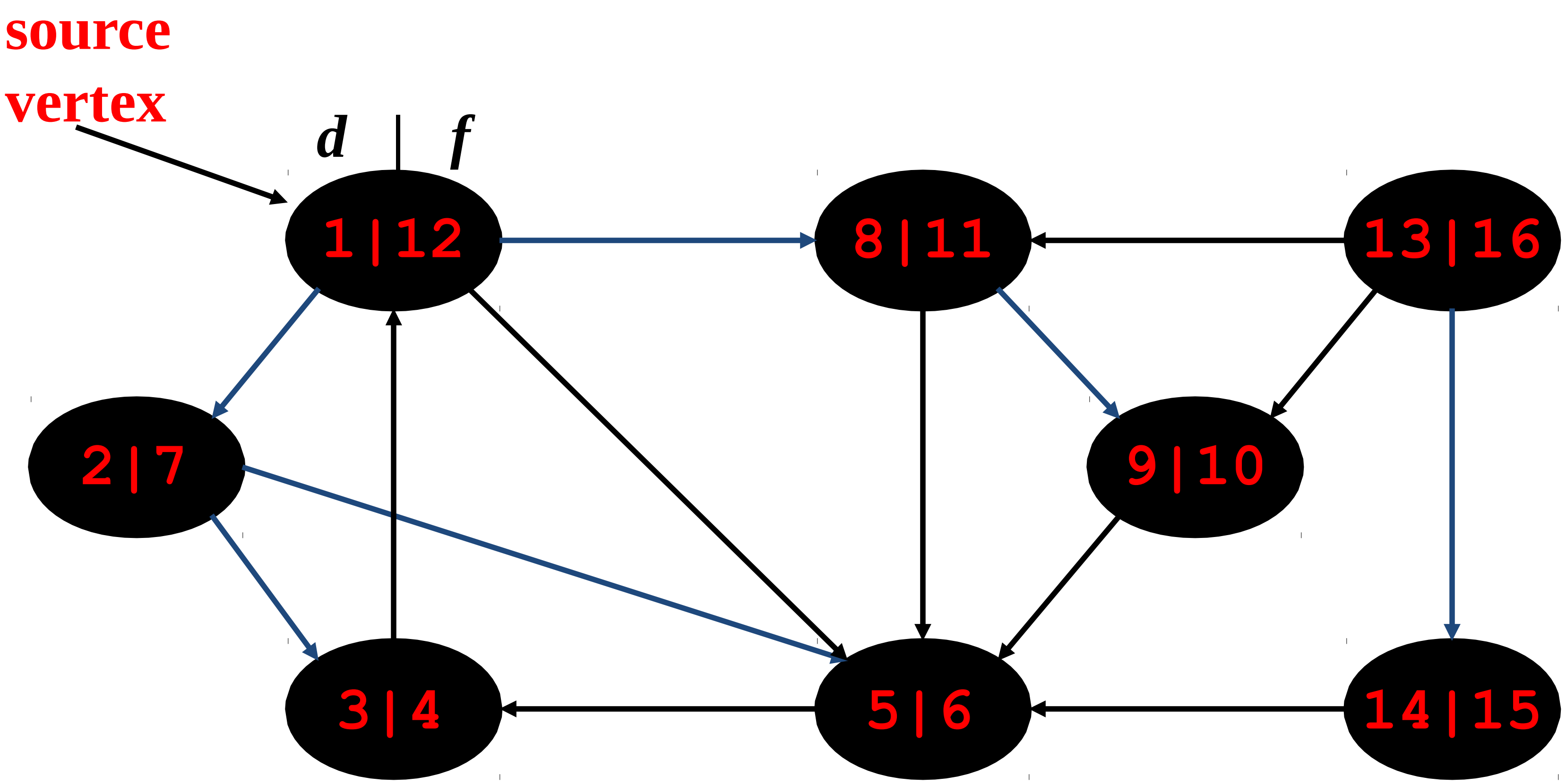
DFS Example



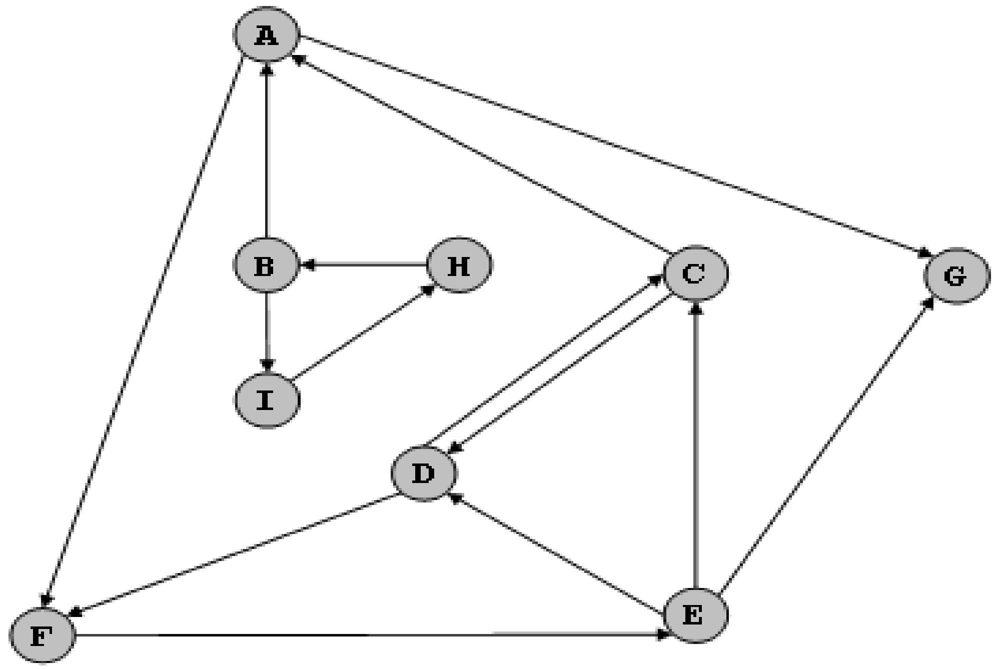
DFS Example



DFS Example



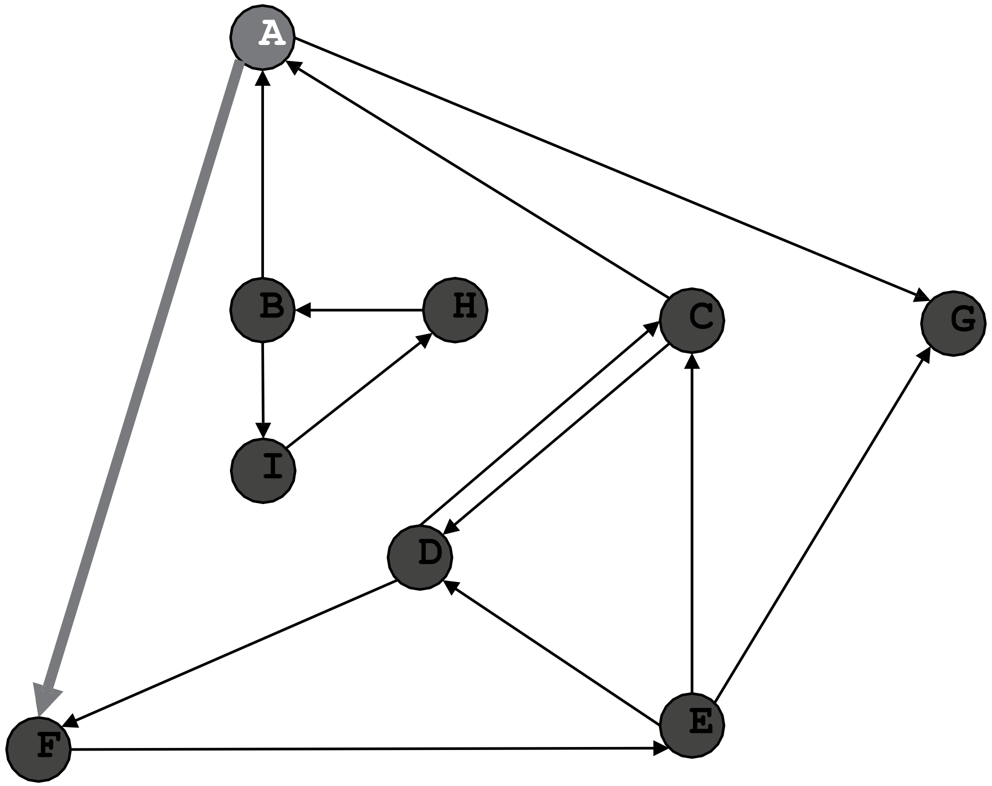
Directed Depth First Search



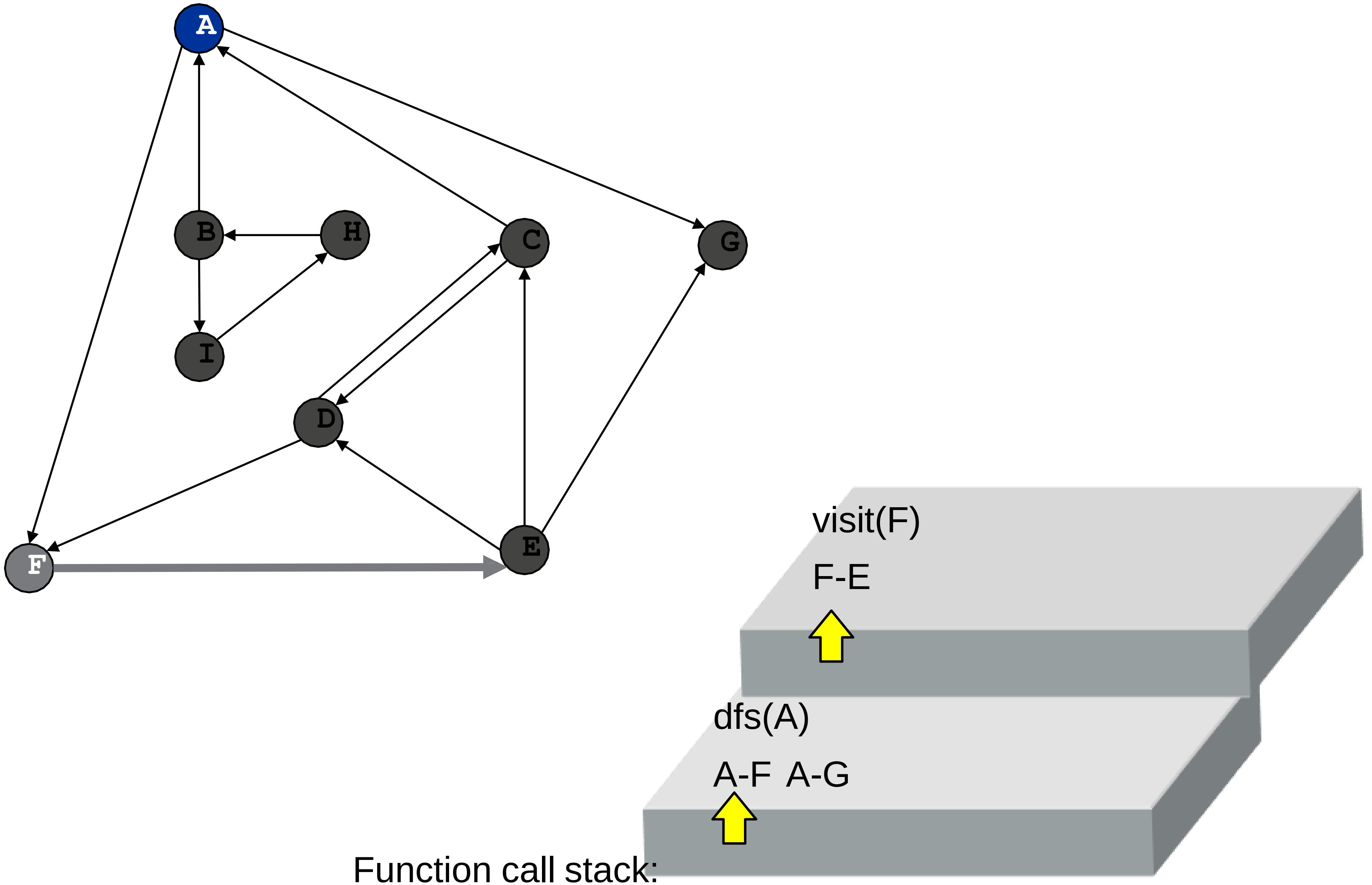
Adjacency Lists

A : F G
B : A H
C : A D
D : C F
E : C D G
F : E
G :
H : B
I : H

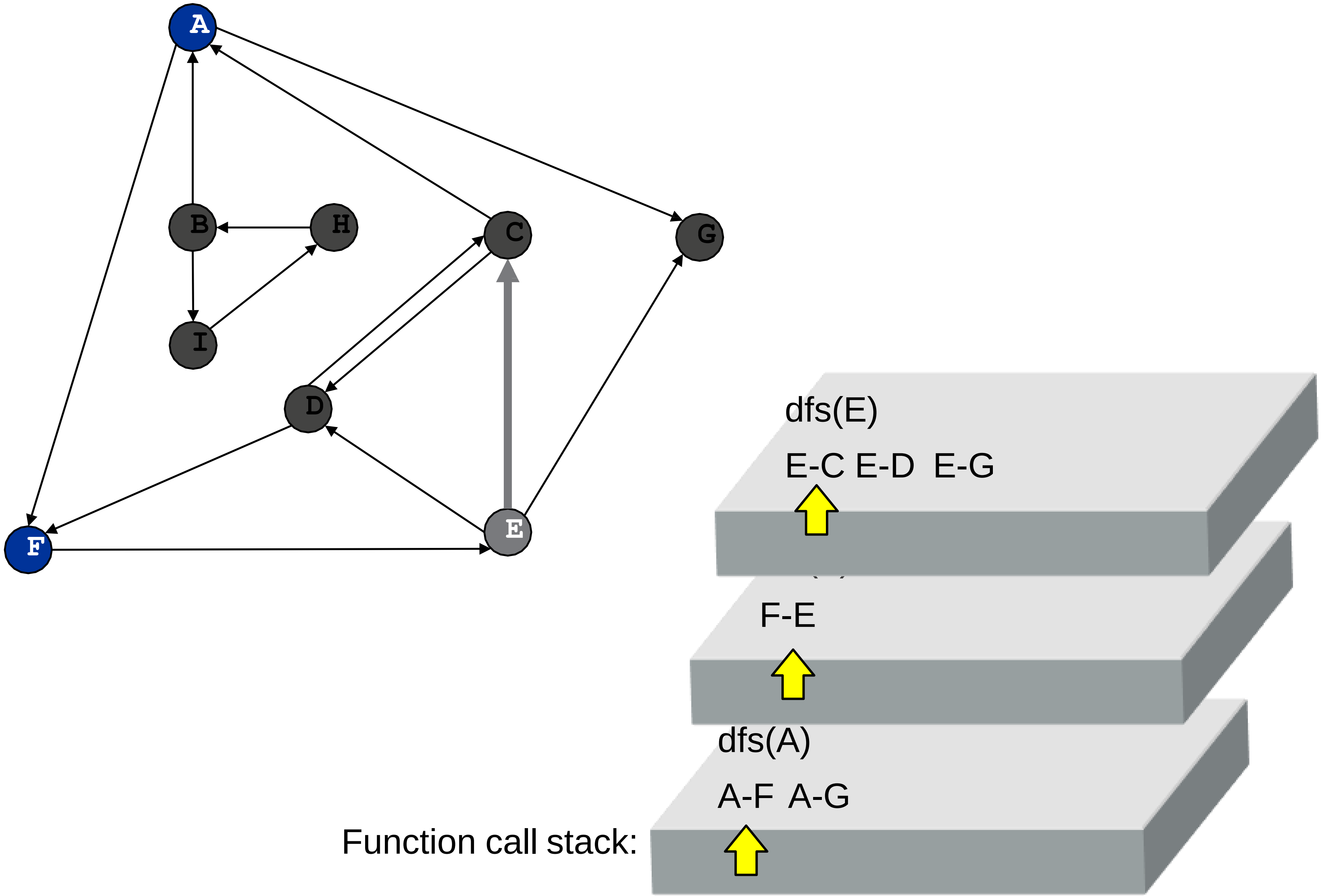
DIRECTED DEPTH FIRST SEARCH

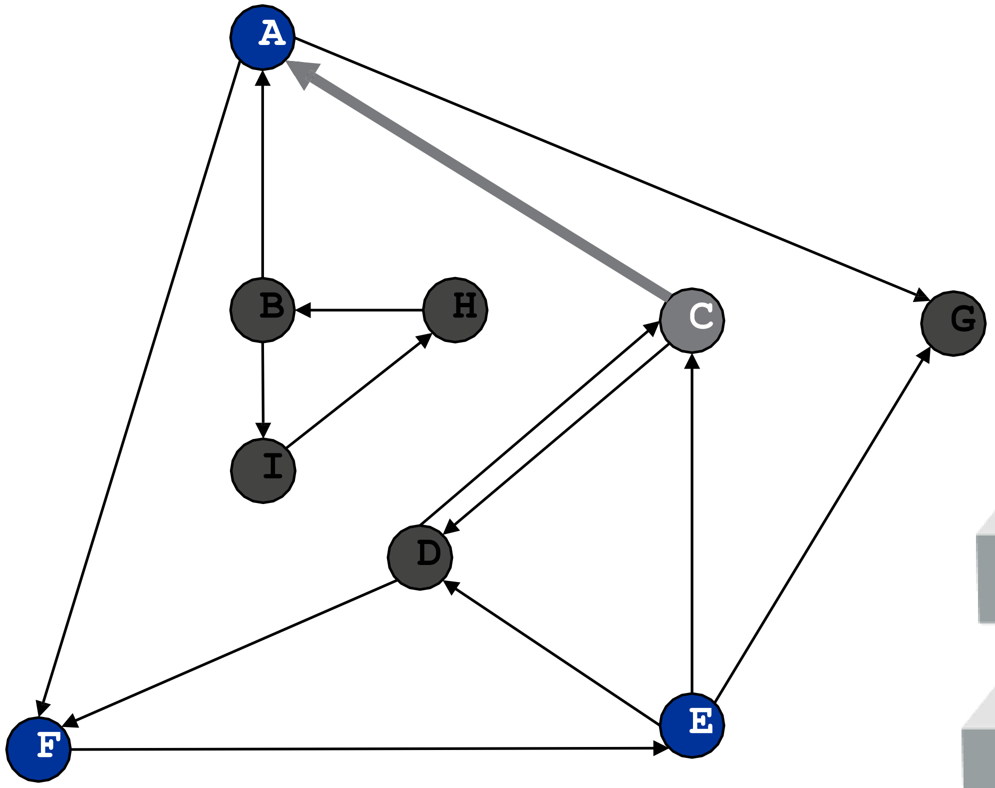


Function call stack:

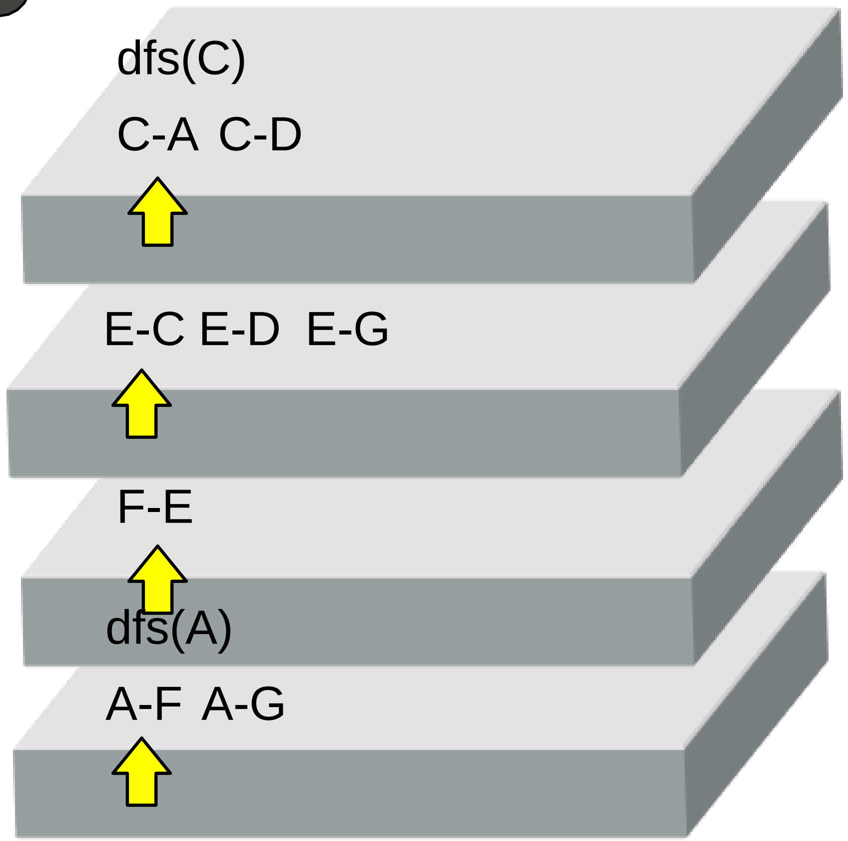


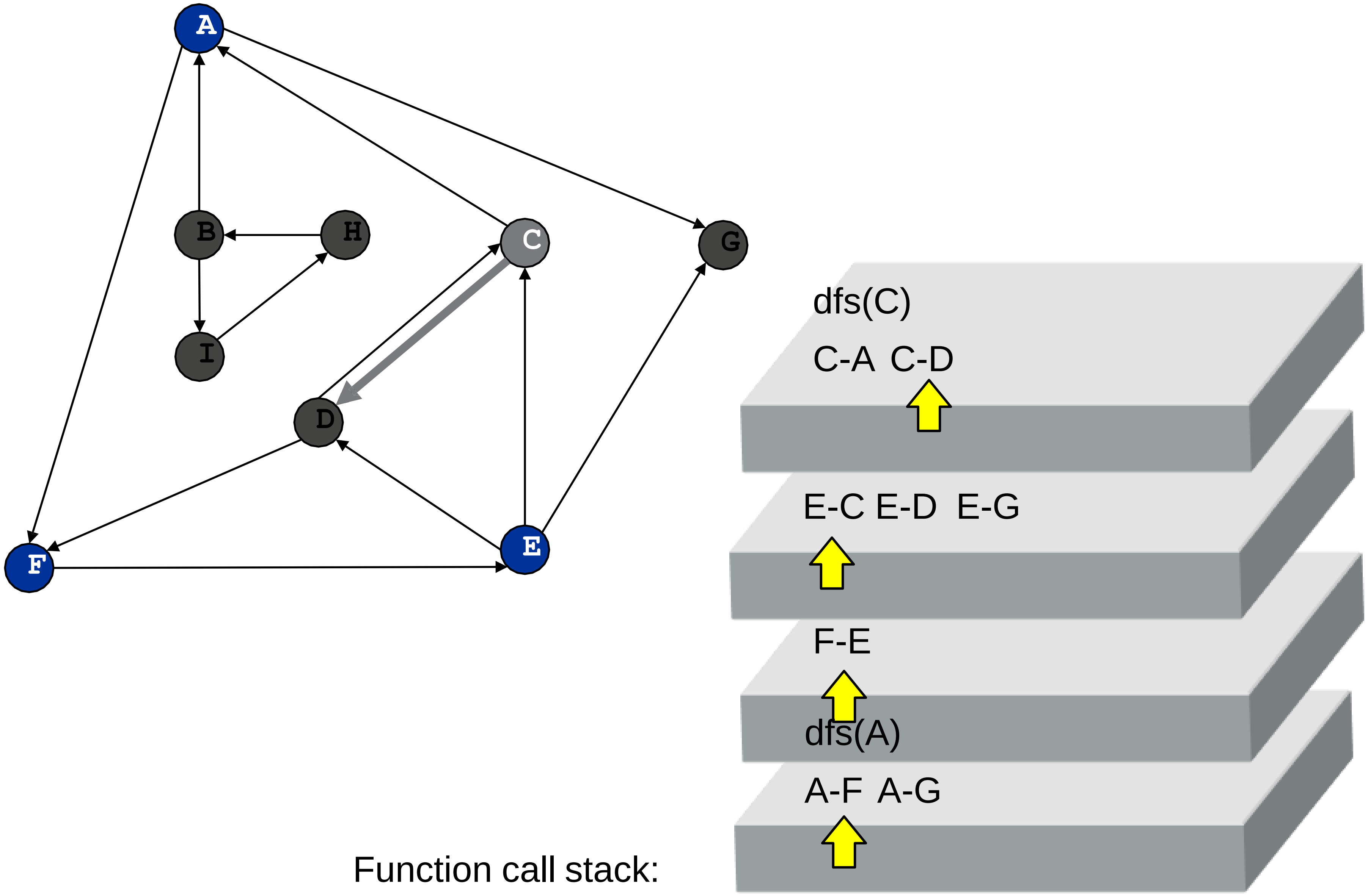
Function call stack:


A
B
H
C
G
I
D
E
F
dfs(E)
E-C E-D E-G
F-E
dfs(A)
A-F A-G
Function call stack:

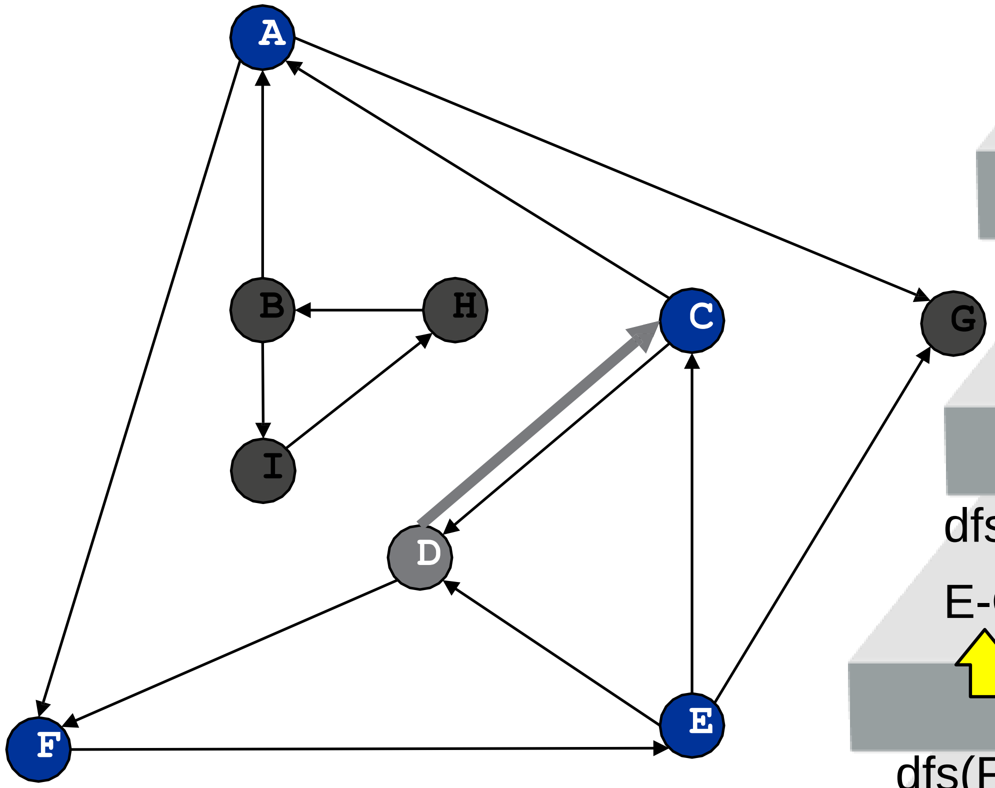


Function call stack:

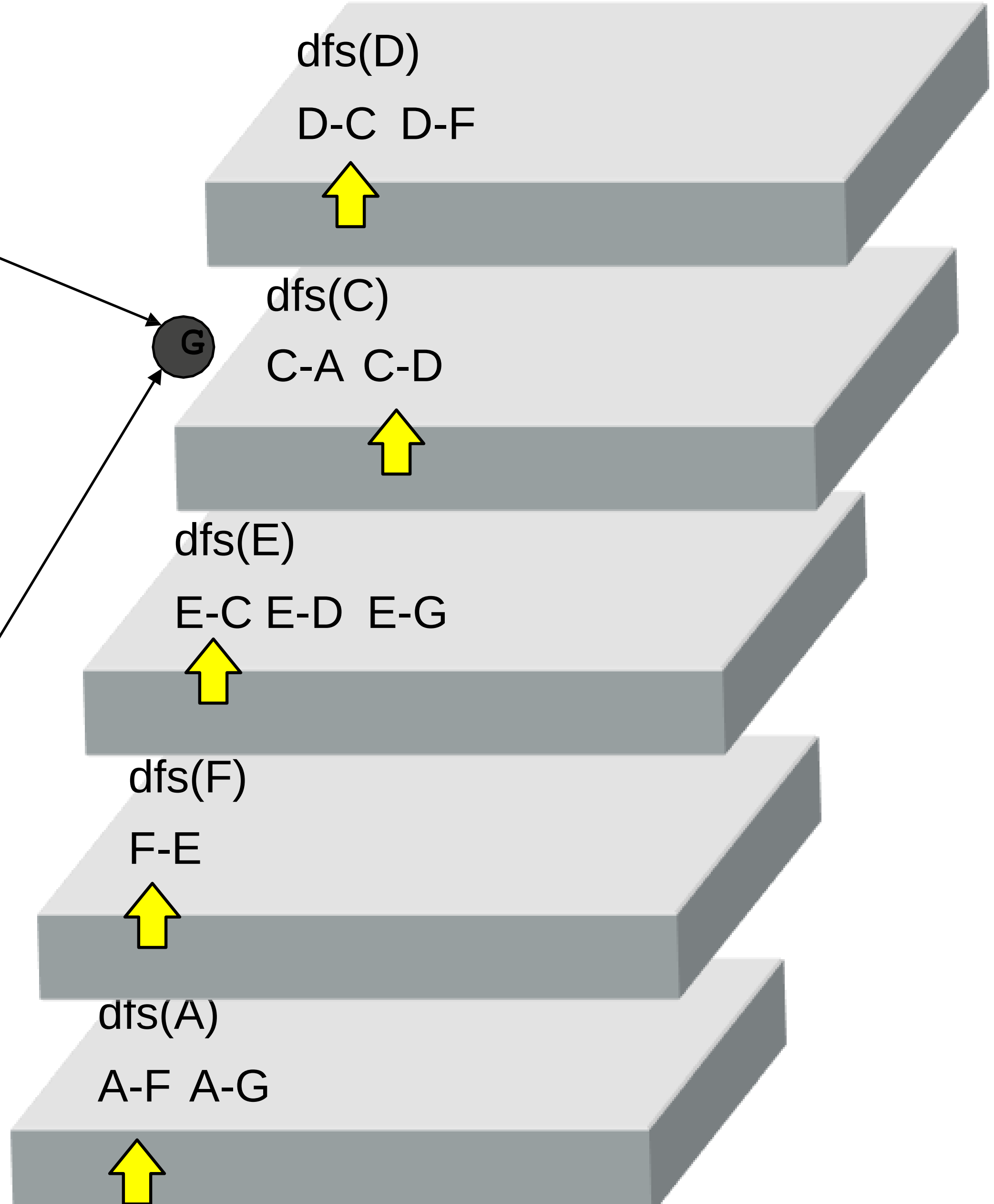


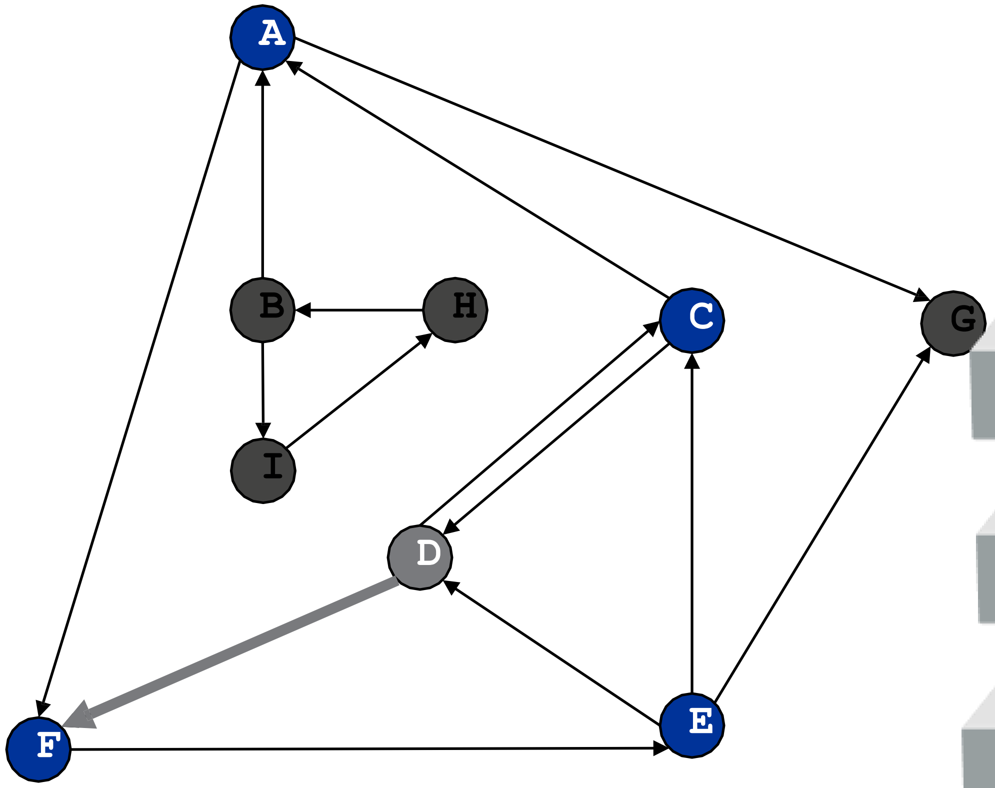


Function call stack:

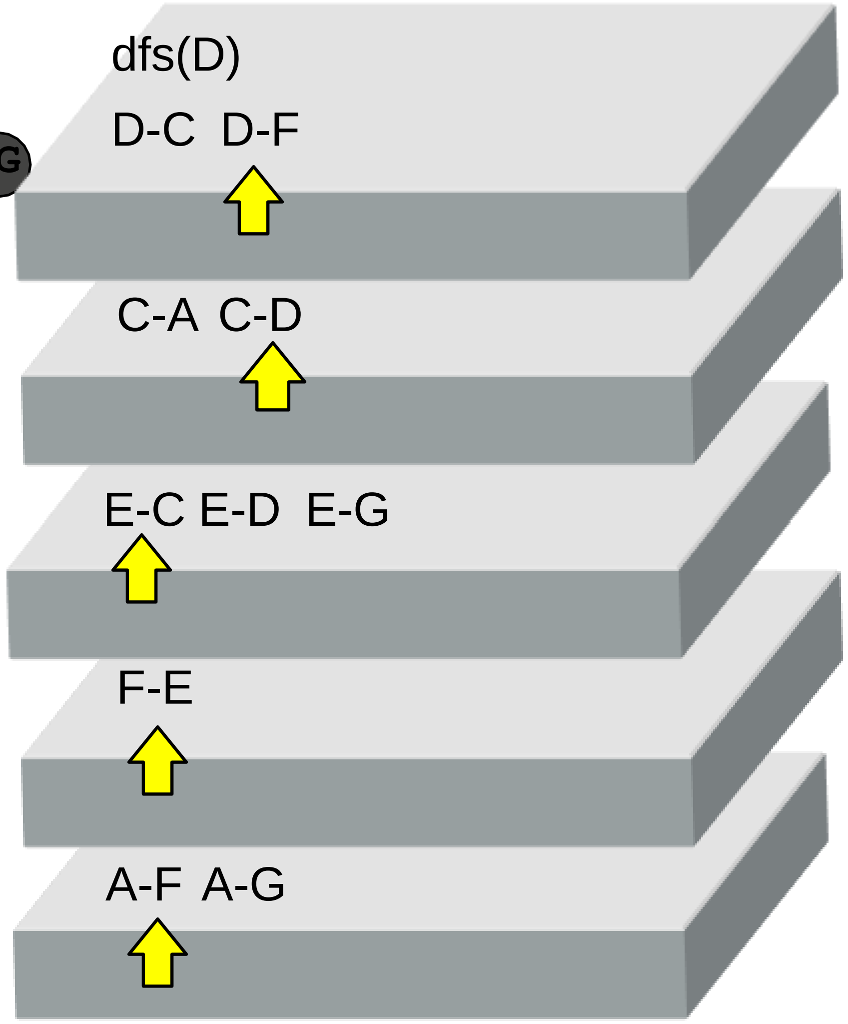


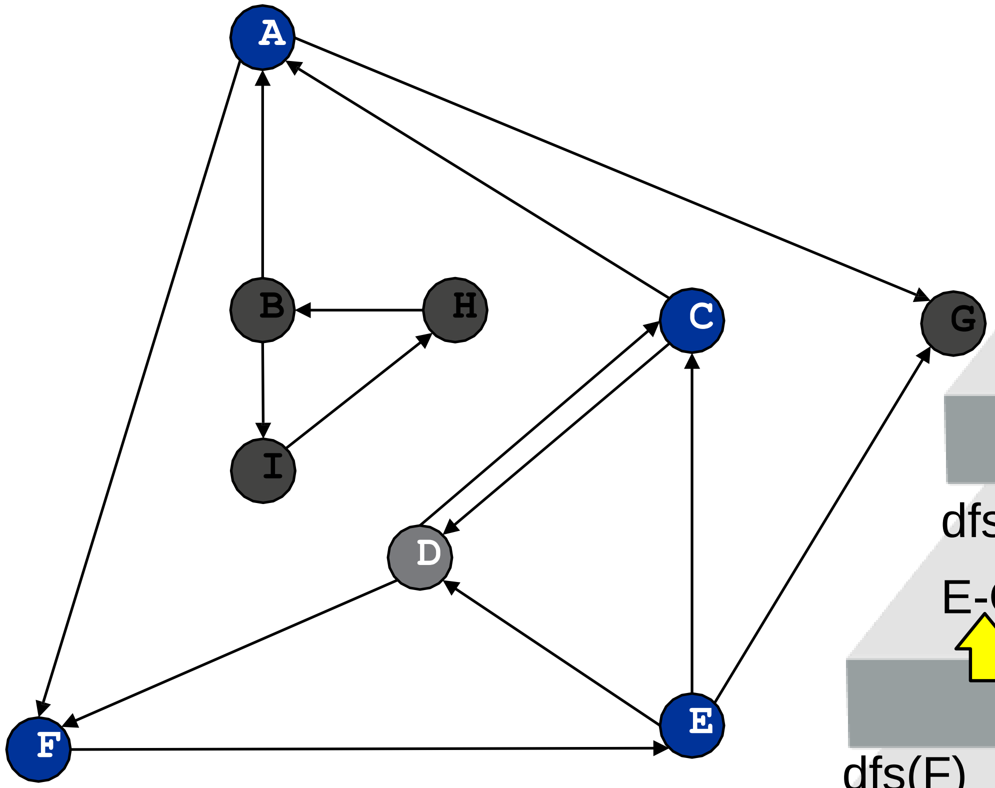
Function call stack:



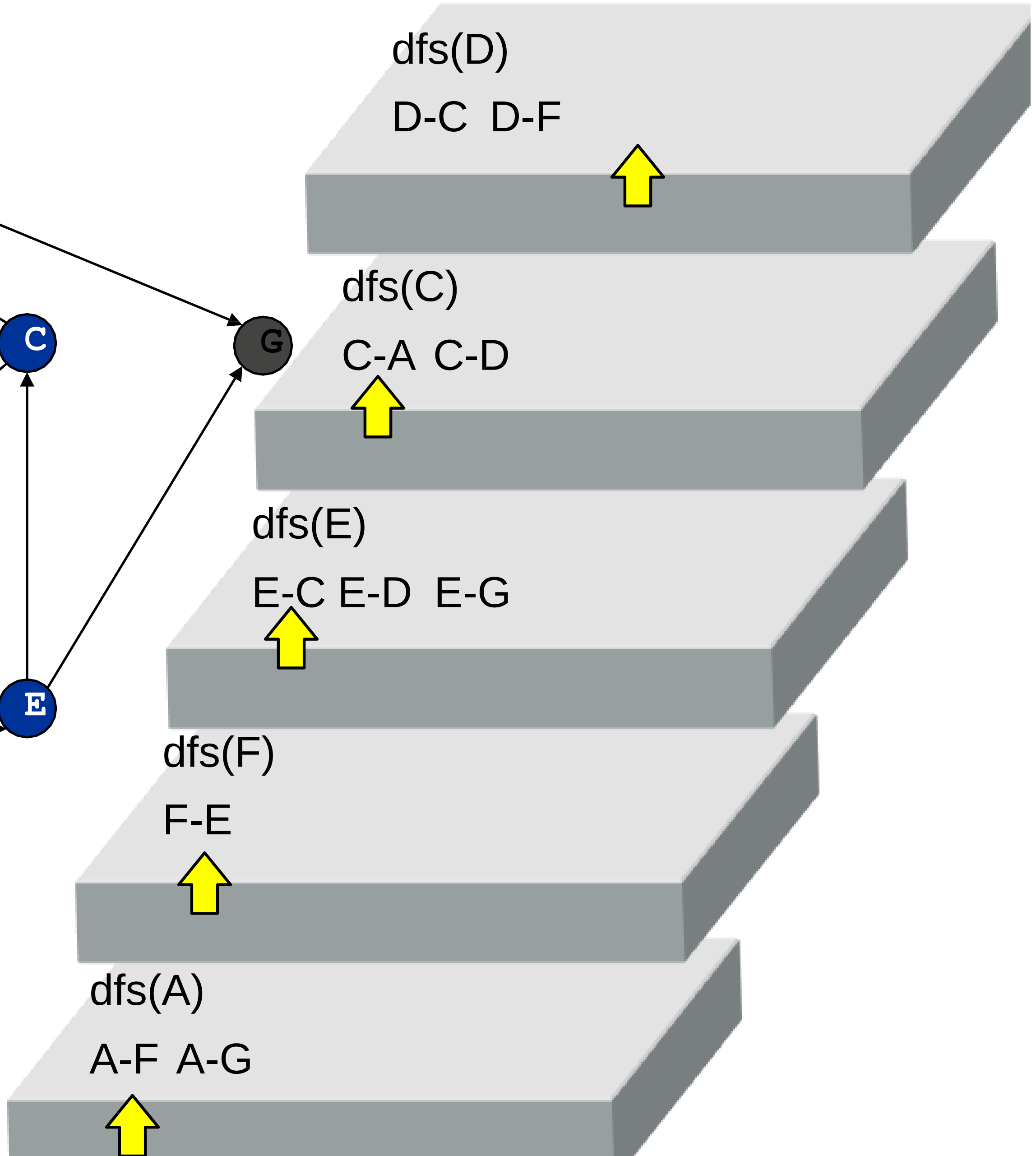


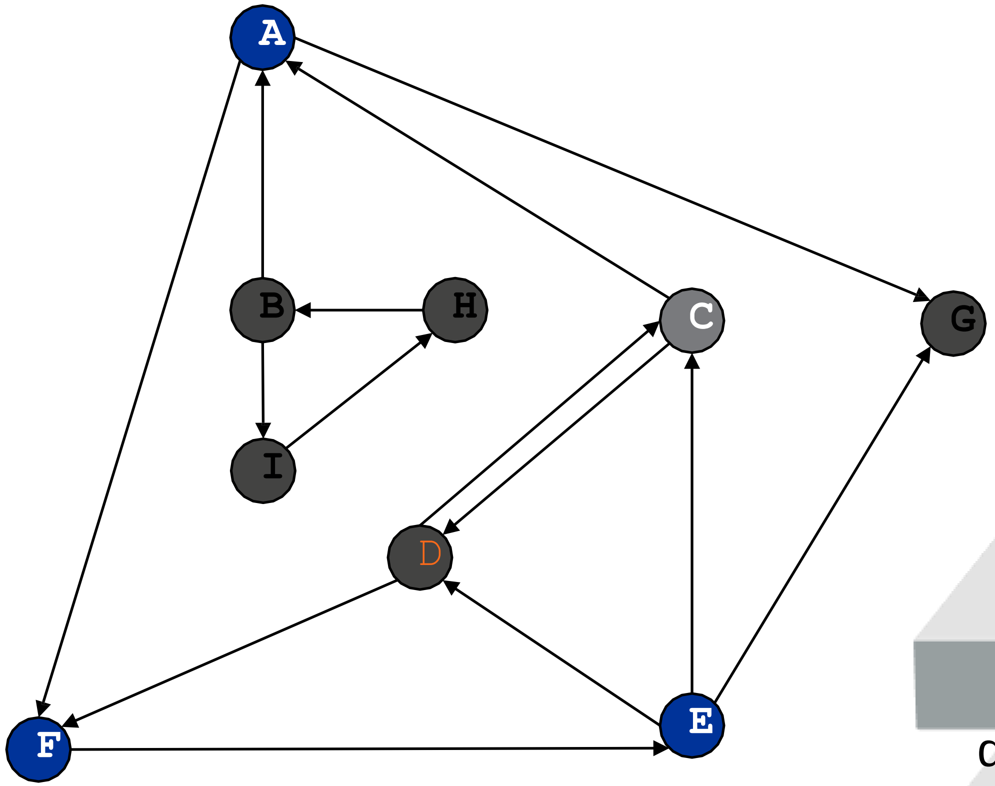
Function call stack:

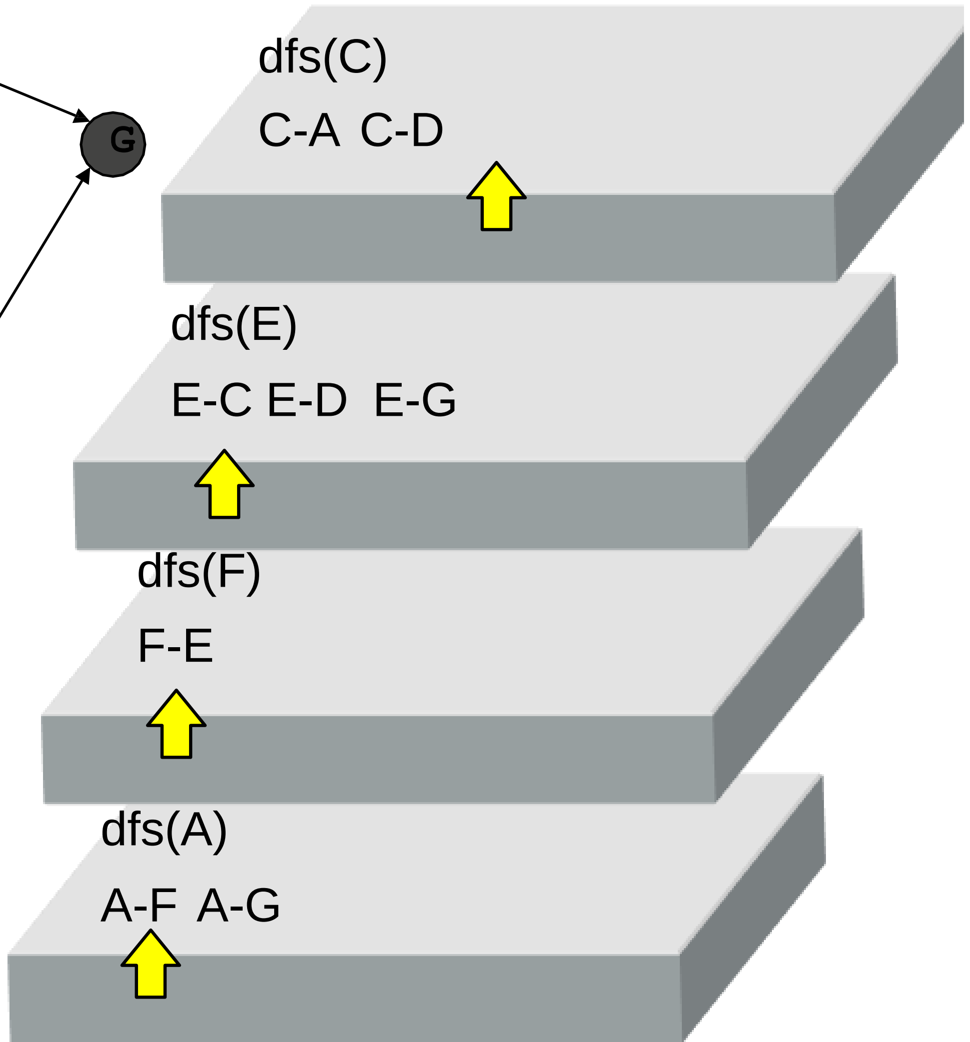




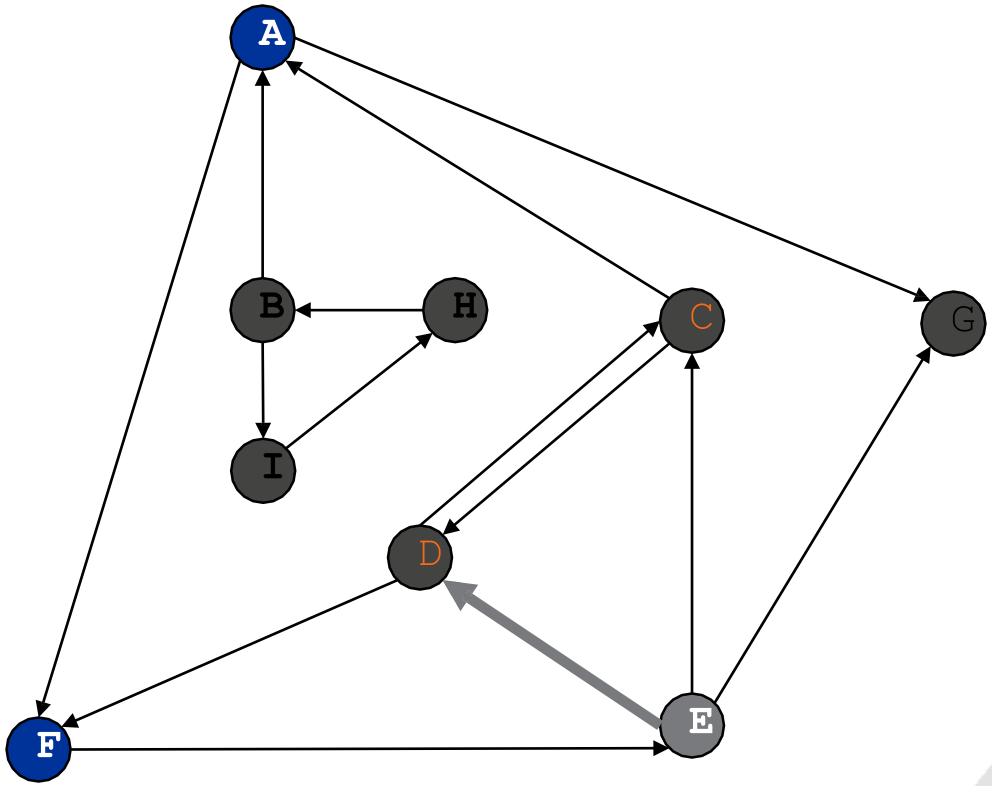
Function call stack:



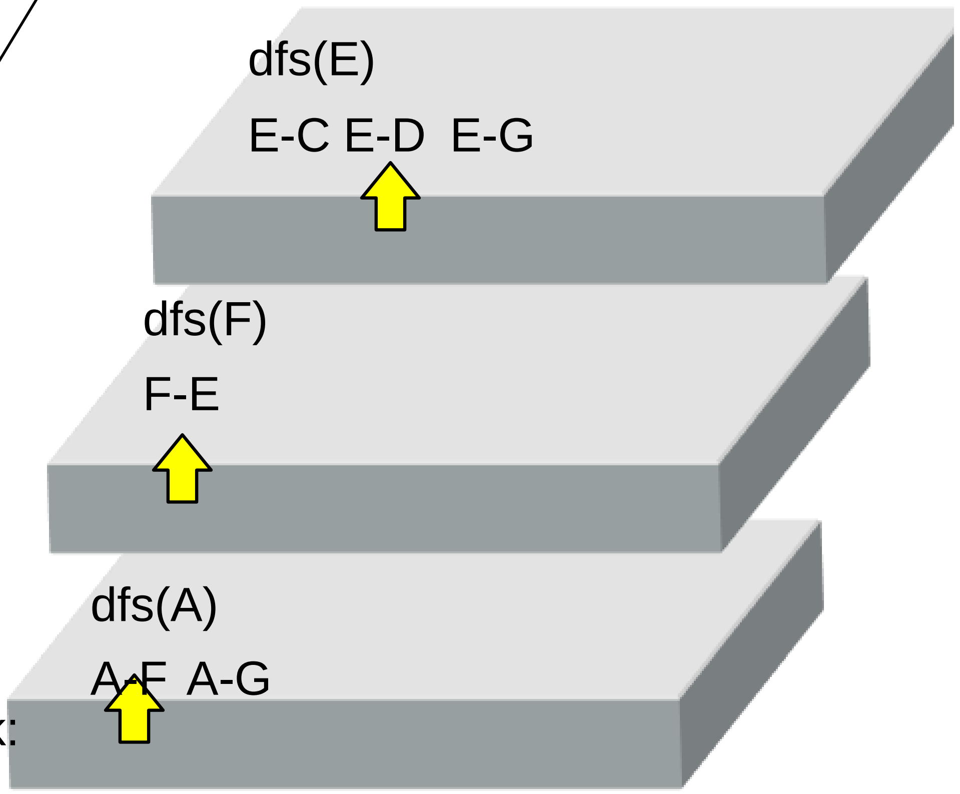


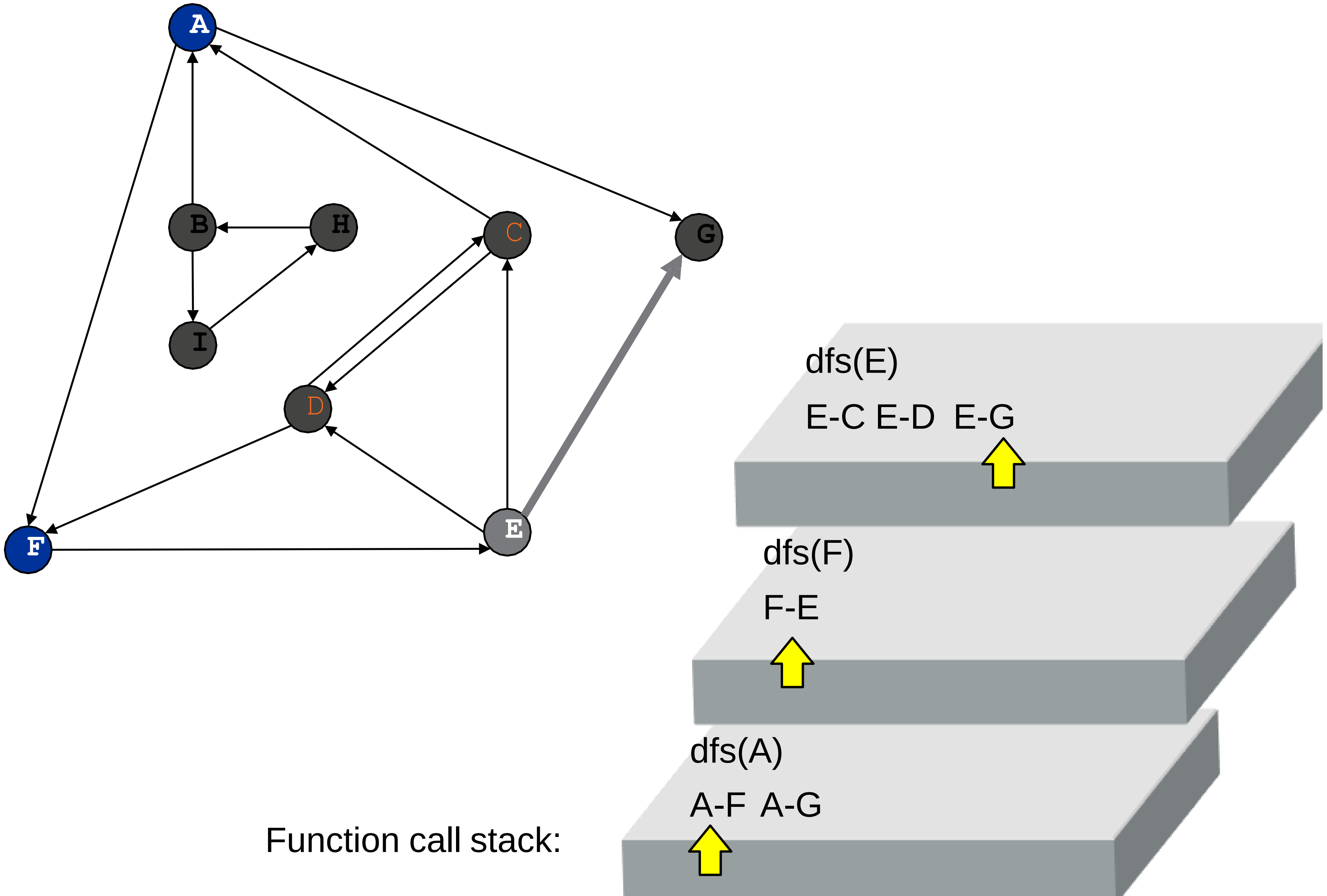


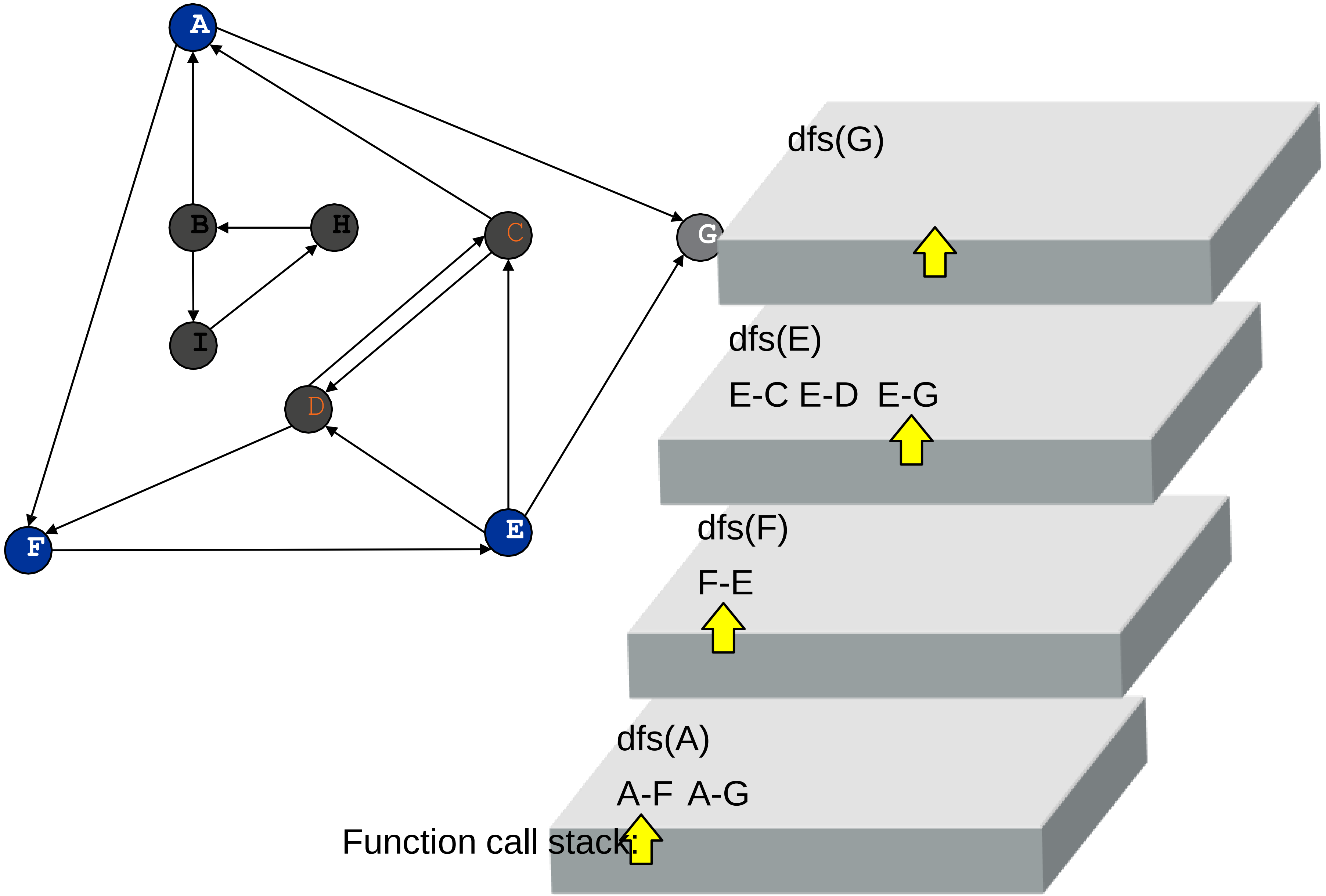
Function call stack:

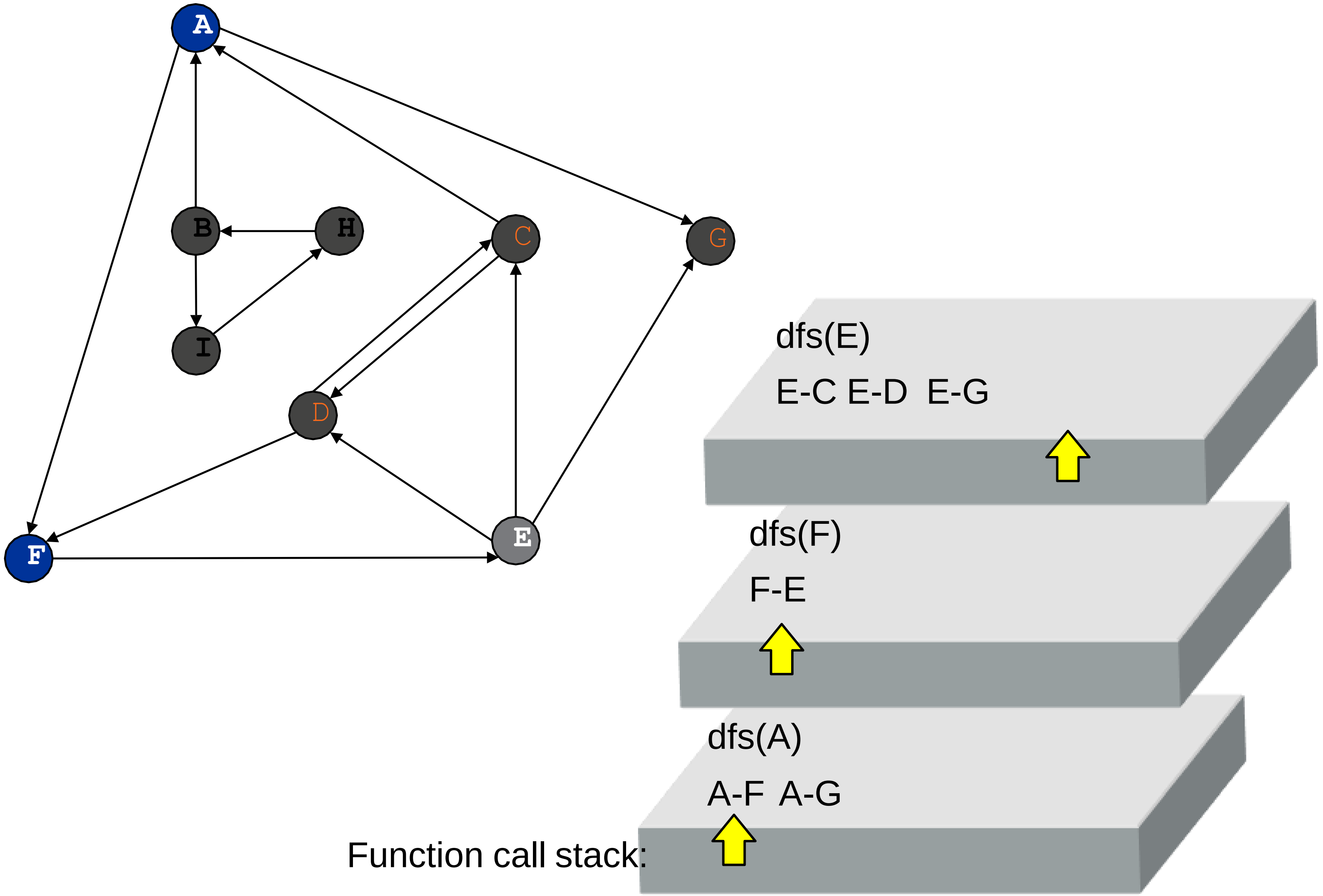


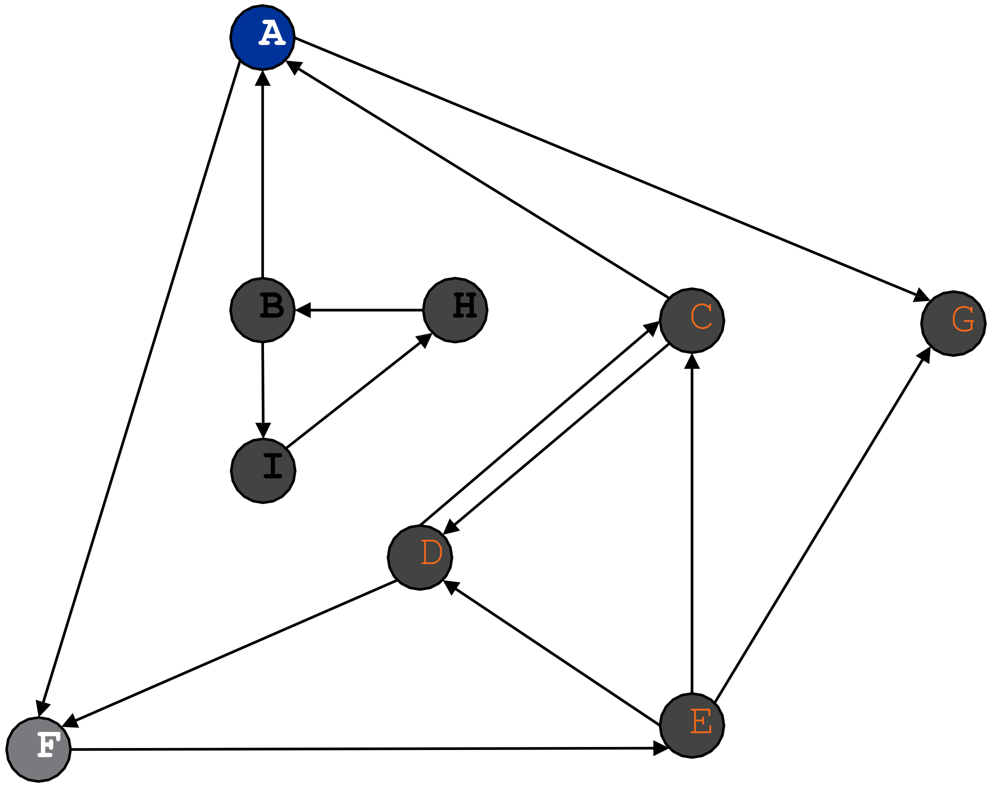
Function call stack:



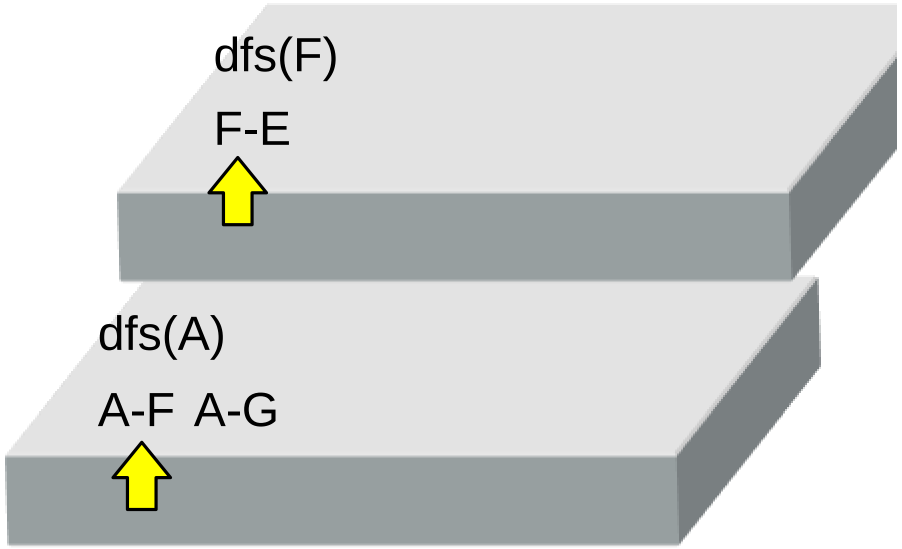

A
B
H
C
G
I
D
E
F
dfs(E)
E-C E-D E-G
dfs(F)
F-E
dfs(A)
A-F A-G
Function call stack:

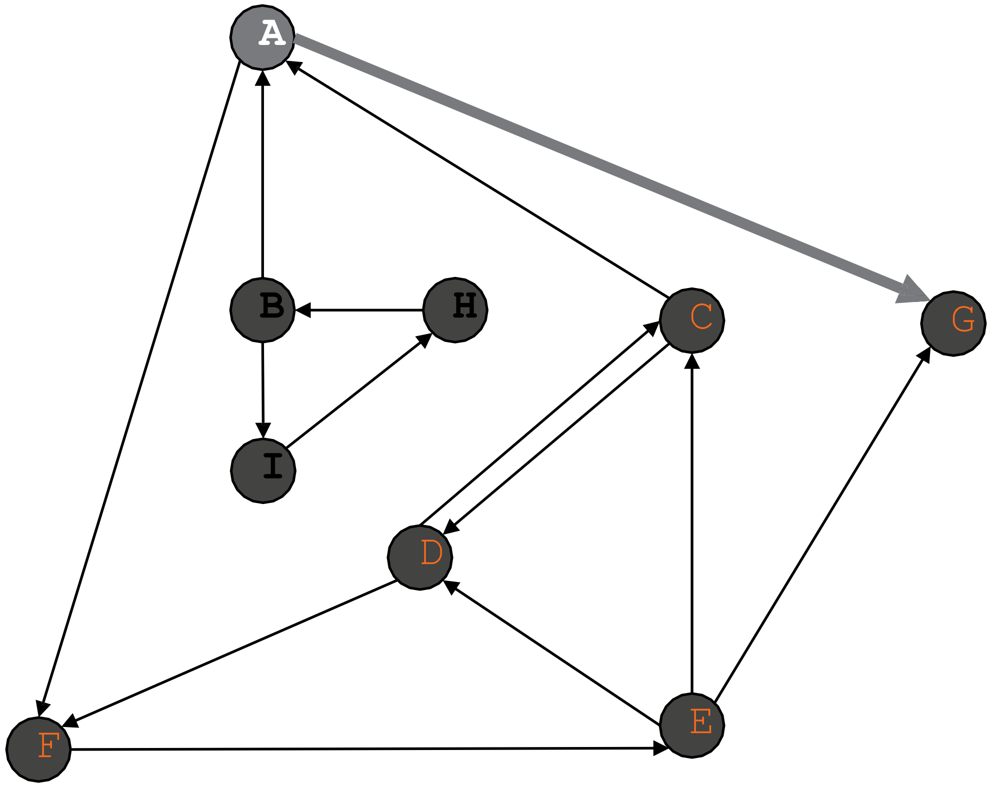

A
B
H
C
G
I
D
E
F
dfs(G)
dfs(E)
E-C E-D E-G
dfs(F)
F-E
dfs(A)
A-F A-G
Function call stack:


A
B
H
I
C
G
D
F
E
dfs(E)
E-C E-D E-G
dfs(F)
F-E
dfs(A)
A-F A-G
Function call stack:

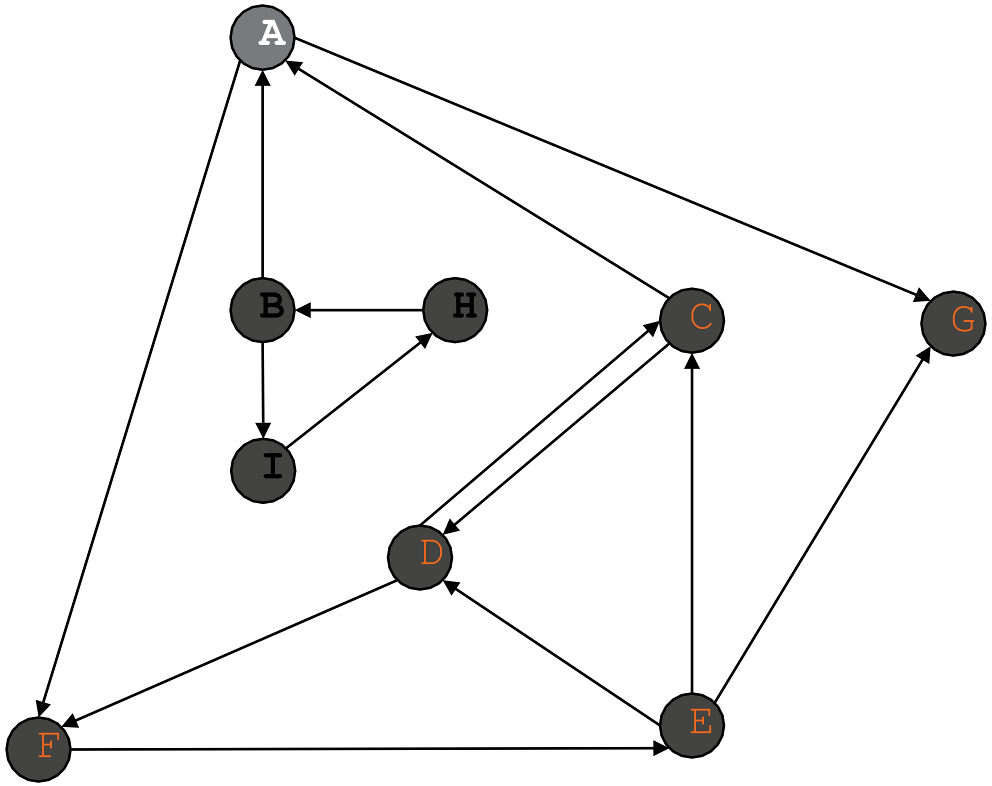


Function call stack:

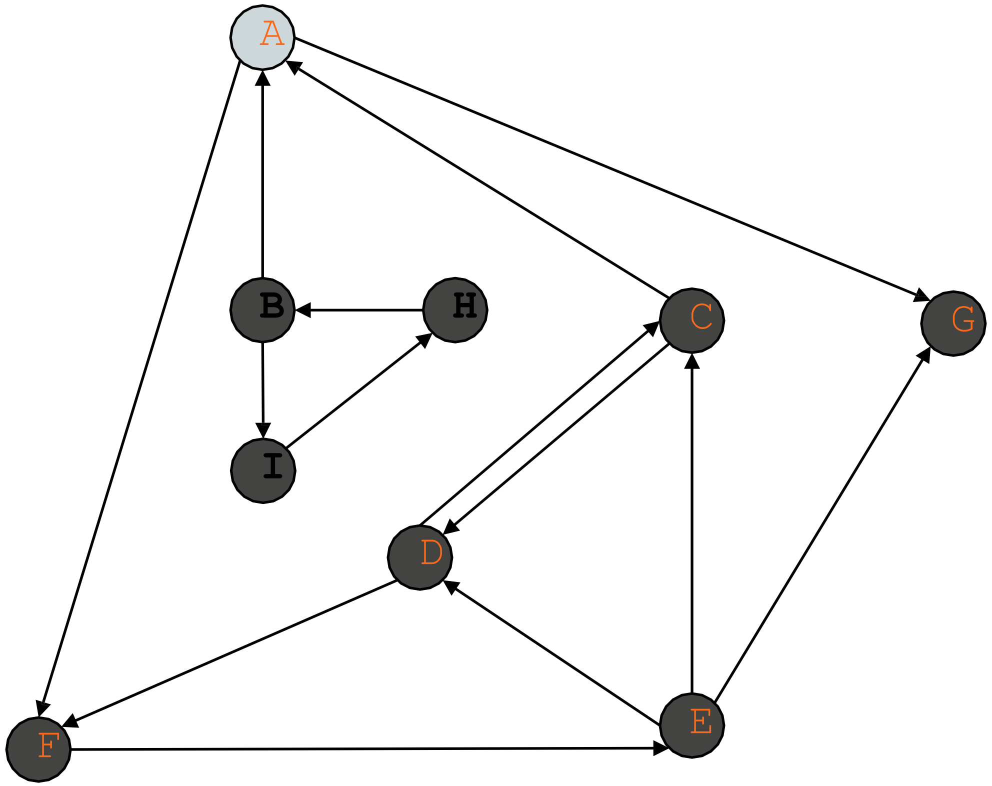




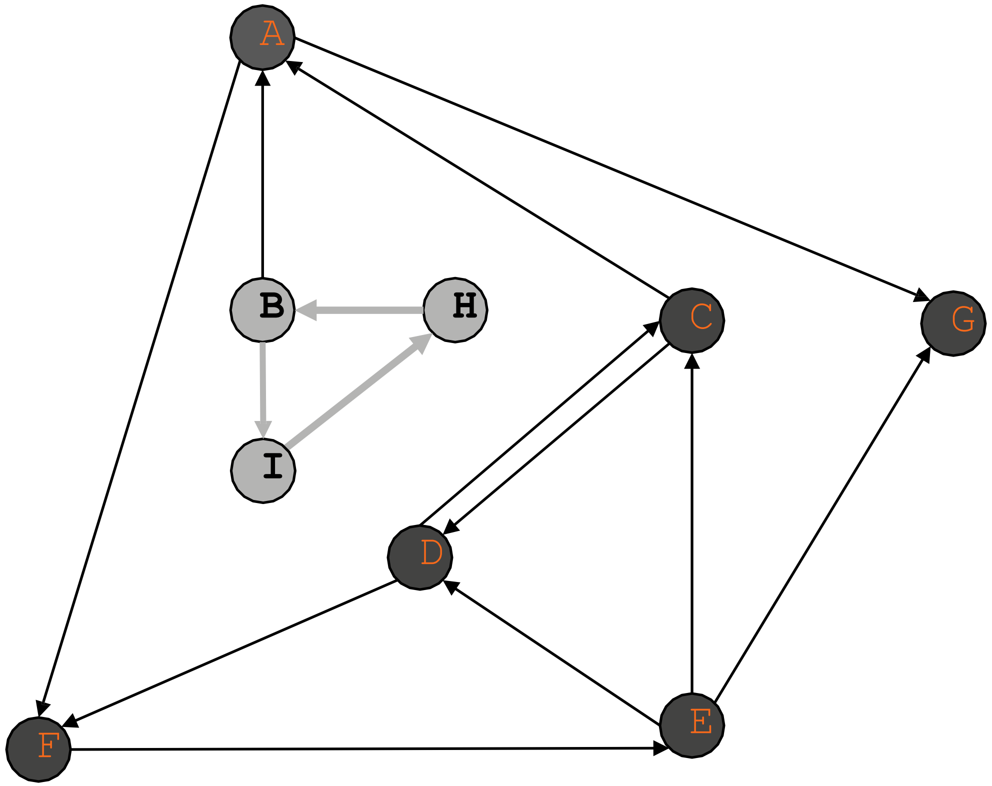
Function call stack:



Function call stack:



Nodes reachable from A: A, C, D, E, F, G



Nodes reachable from A: A, C, D, E, F, G

Running Time of DFS Algorithm

Lines 1-2 and lines 4-5 of DFS take time $\Theta(V)$, exclusive of the time to execute the calls to DFS-VISIT.

The procedure DFS_VISIT is called exactly once for each vertex $v \in V$. During an execution DFS_VISIT(v), the loop on lines 4-7 is executed $|Adj[v]|$ times. Since $\Sigma|Adj[v]| = \Theta(E)$, the total cost of executing lines 2-6 of DFS-VISIT is $\Theta(E)$.

So the total running time of DFS is $\Theta(V+E)$.

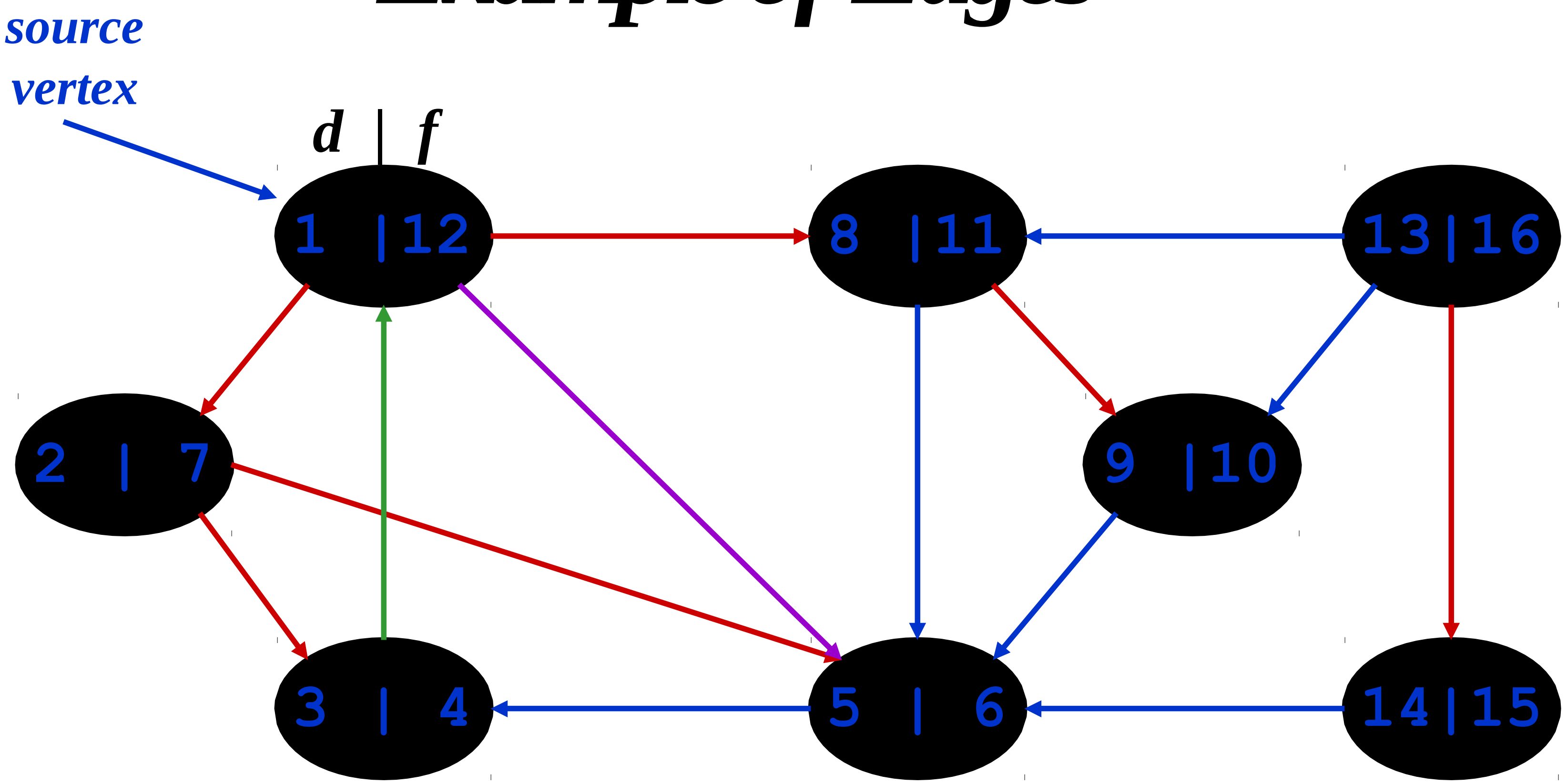
DFS: Application

- Topological Sort
- Strongly Connected Component

Classification of Edges

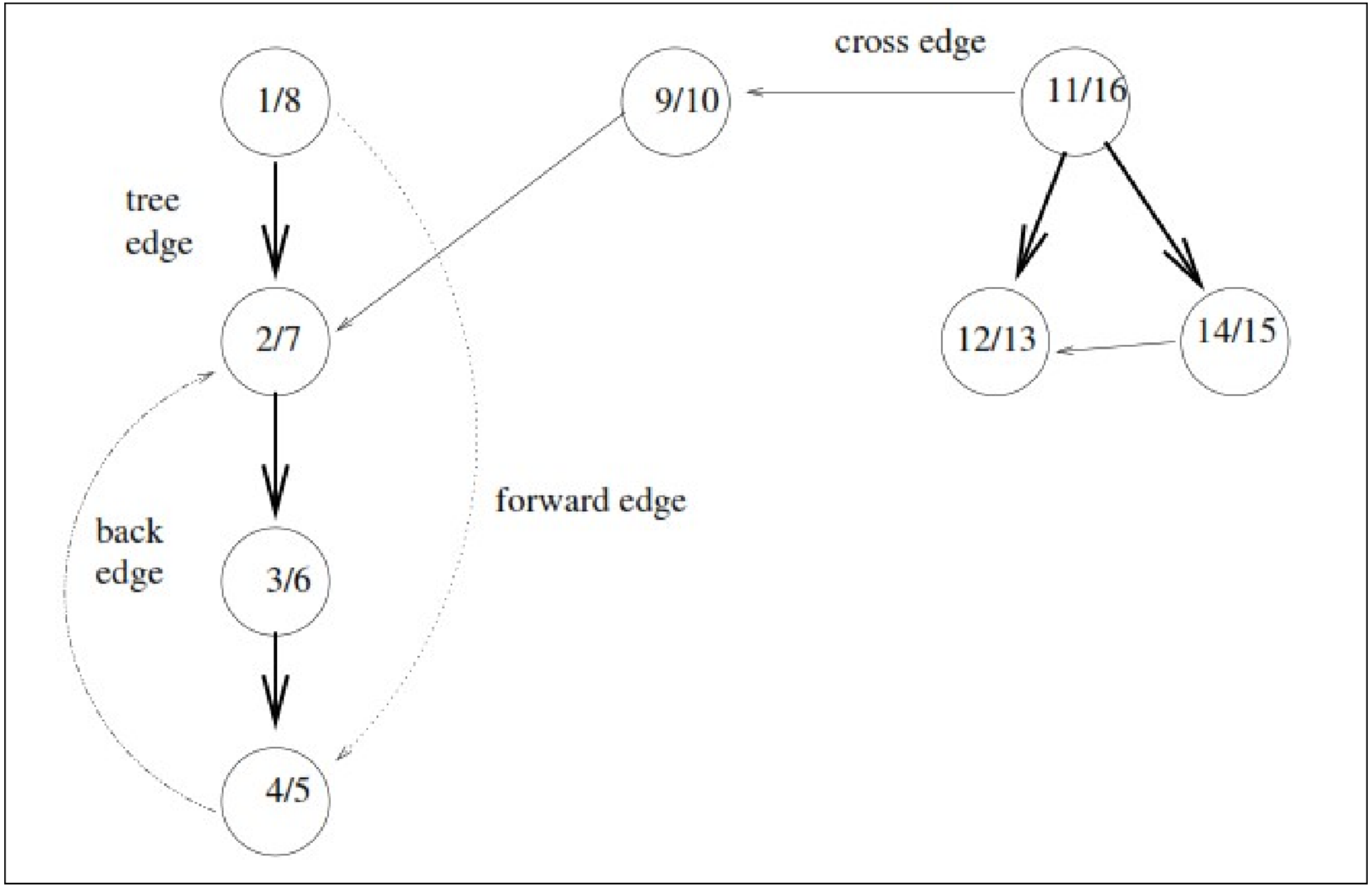
- Another interesting property of DFS is that the DFS can be used to classify the edges of the input graph G=(V,E). The DFS creates depth first forest which can have four types of edges:
- **Tree edge:** Edge (u,v) is a tree edge if **v** was first discovered by exploring edge (u,v). **White** color indicates tree edge.
- **Back edge:** Edge (u,v) is a back edge if it connects a vertex **u** to a ancestor **v** in a depth first tree. **Gray** color indicates back edge.
- **Forward edge:** Edge (u,v) is a forward edge if it is non-tree edge and it connects a vertex **u** to a descendant **v** in a depth first tree. **Black** color indicates forward edge.
- Cross edge: rest all other edges are called the cross edge. **Black** color indicates forward edge.

Example of Edges



Tree edges   Back edges   Forward edges   Cross edges

Example of Edges



Thank you