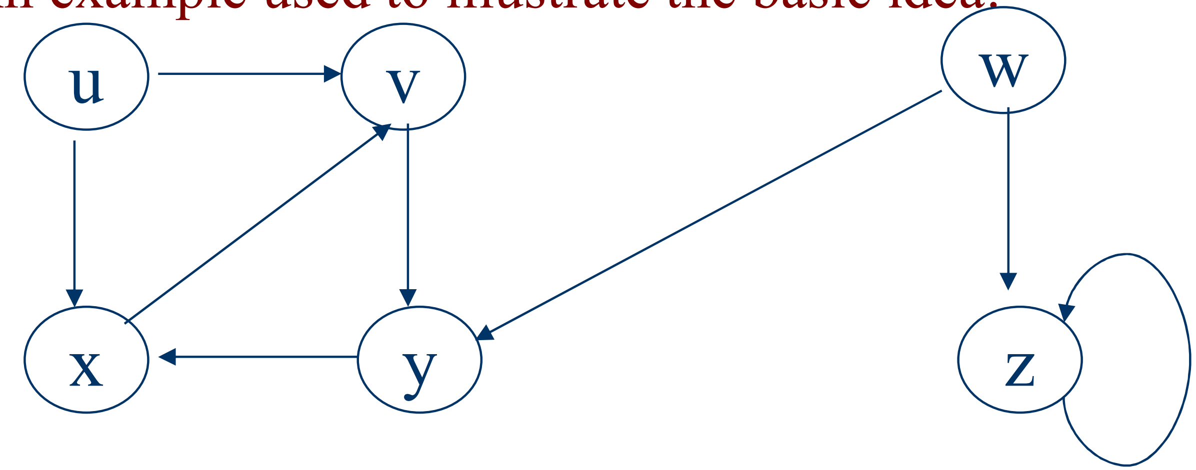
Lecture 3
Depth First Search

- The DFS algorithm
- The time complexity of DFS algorithm
- Properties of the DFS

DFS Algorithm

- The basic idea of Depth First Search algorithm
  - Deeper: 检测最新扫描的顶点的出边，选择一条。
  - Backtrack:当前顶点的所有出边都检测完,
    则从哪来回哪去。

An example used to illustrate the basic idea:



Four Arrays for DFS Algorithm

- $u.\,color$: the color of each vertex u
  - WHITE means undiscovered
  - GRAY means discovered but not finished processing
  - BLACK means finished processing.
- $u.\pi$:the predecessor of u, indicating the vertex from which u is discovered.
- $u.d$: discovery time, a counter indicating when vertex u is discovered.
- $u.f$: finishing time, a counter indicating when the processing of vertex u (and the processing of all its descendants) is finished.

DFS Algorithm

DFS(G)

for each vertex $u \in V[G]$
    do $u.\,color \leftarrow$ WHITE
        $u.\pi \leftarrow$ NIL
$\boxed{time \leftarrow 0}$
for each vertex $u \in V[G]$
    do if $u.\,color =$ WHITE
        then DFS-VISIT(u)

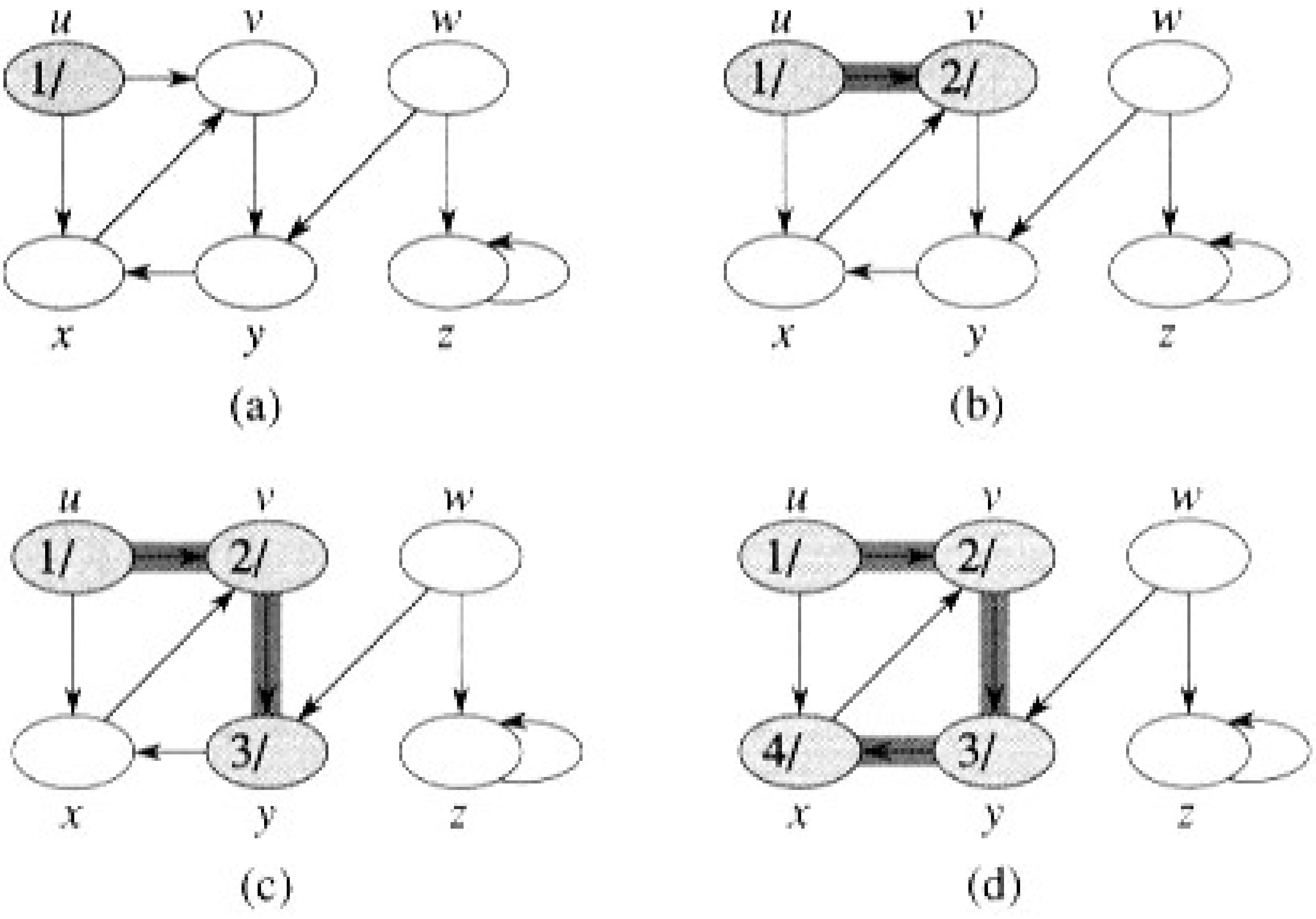
a global variable

DFS-VISIT(u)

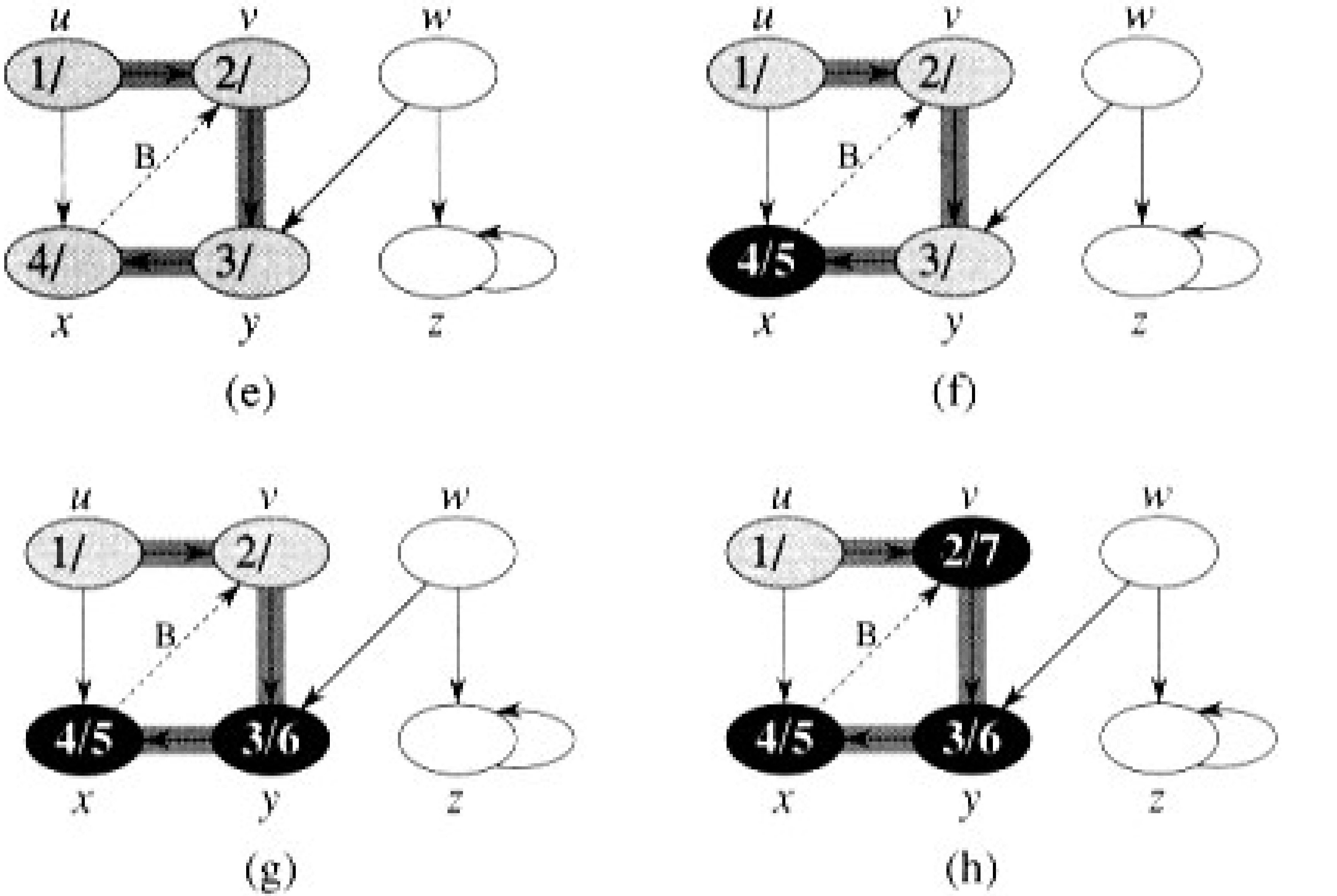
$u.\,color =$ GRAY
$u.d \leftarrow \boxed{time \leftarrow time + 1}$
for each $v \in Adj[u]$
    do if $v\,.\,color =$ WHITE
        then $v\,.\pi \leftarrow u$
            DFS-VISIT(v)
$u.\,color \leftarrow$ BLACK
$u.f \leftarrow \boxed{time \leftarrow time + 1}$

$u.d$: discovery time
$u.f$: finishing time

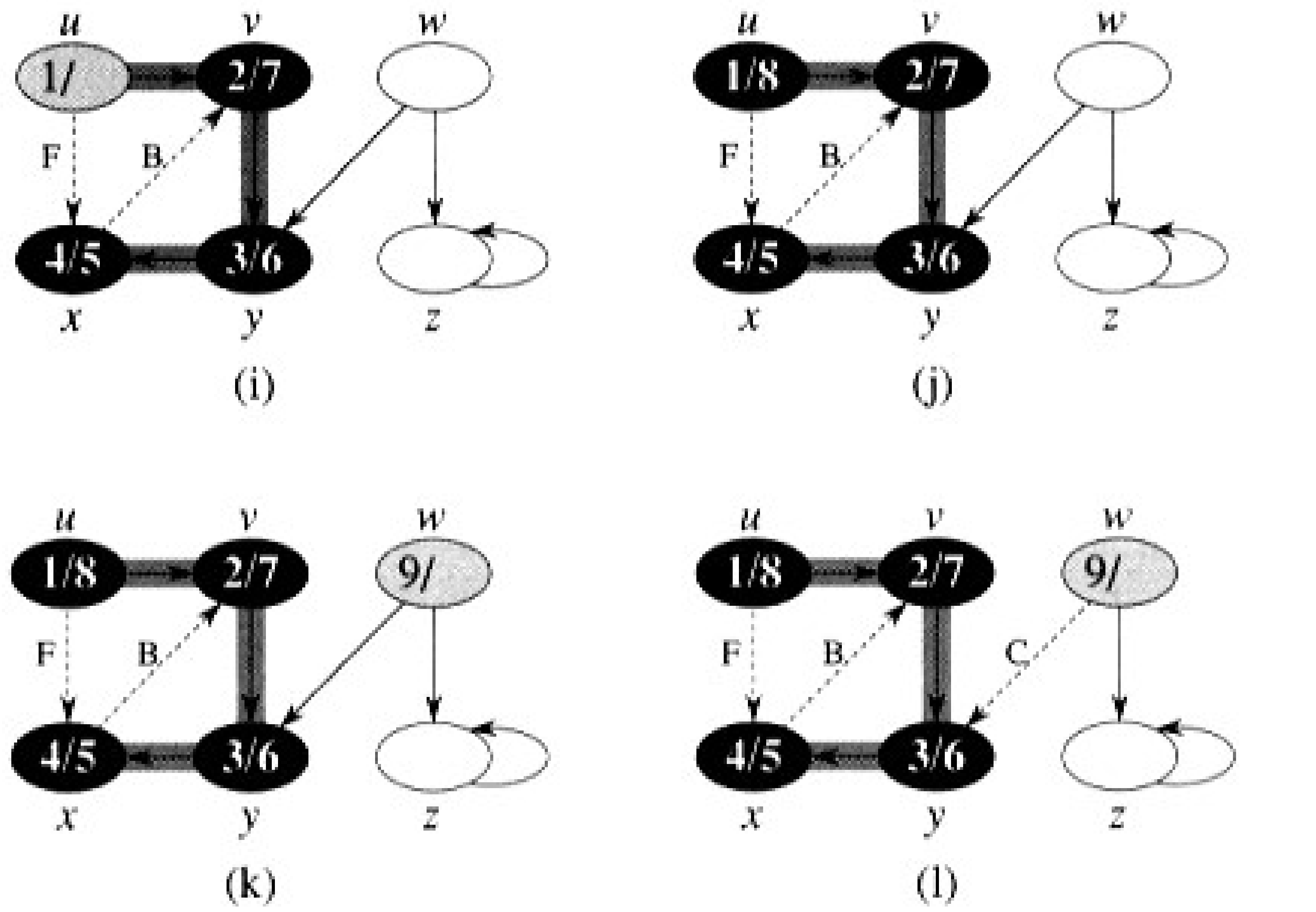
DFS Algorithm (Example –directed graph)



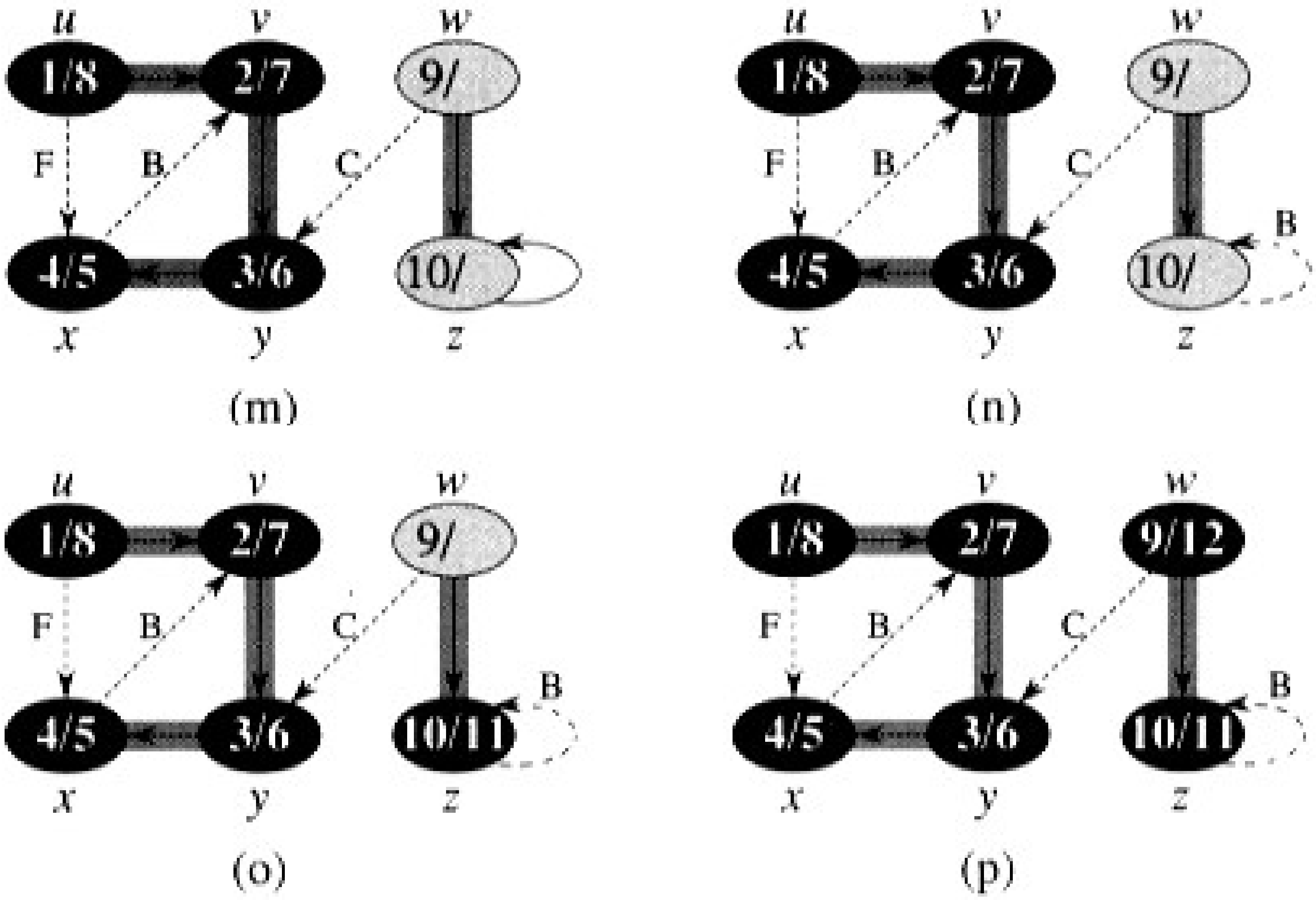
Example –directed graph (continued)



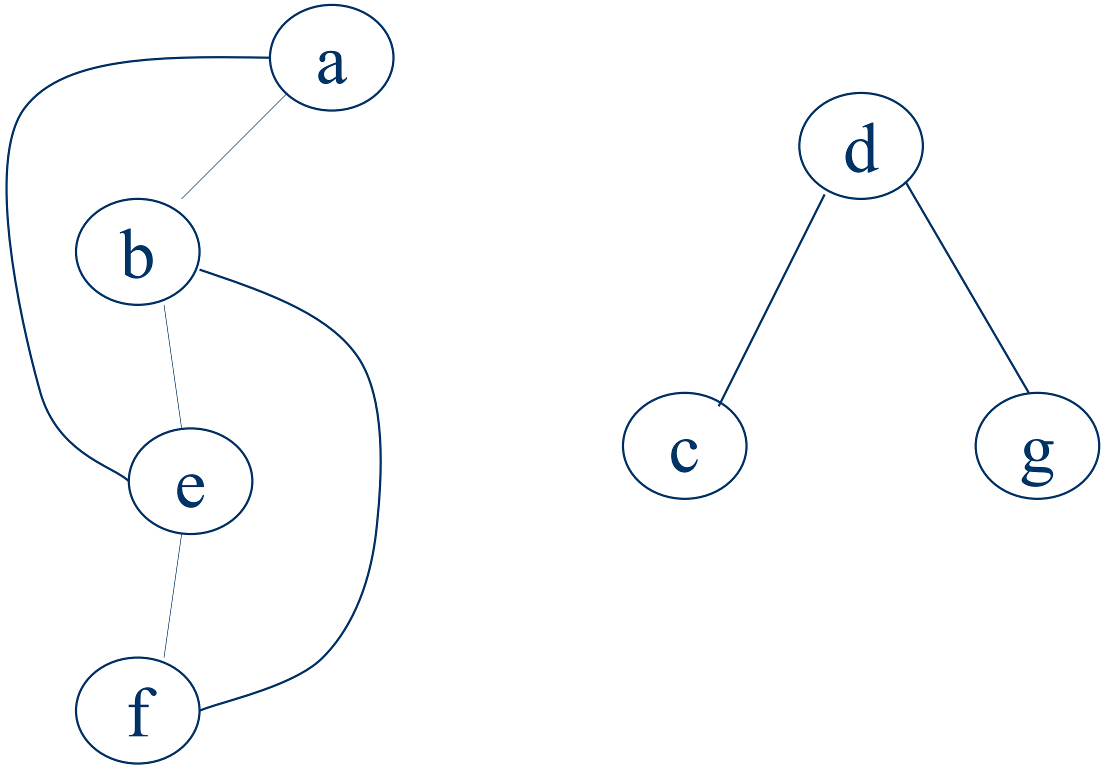
Example –directed graph (continued)



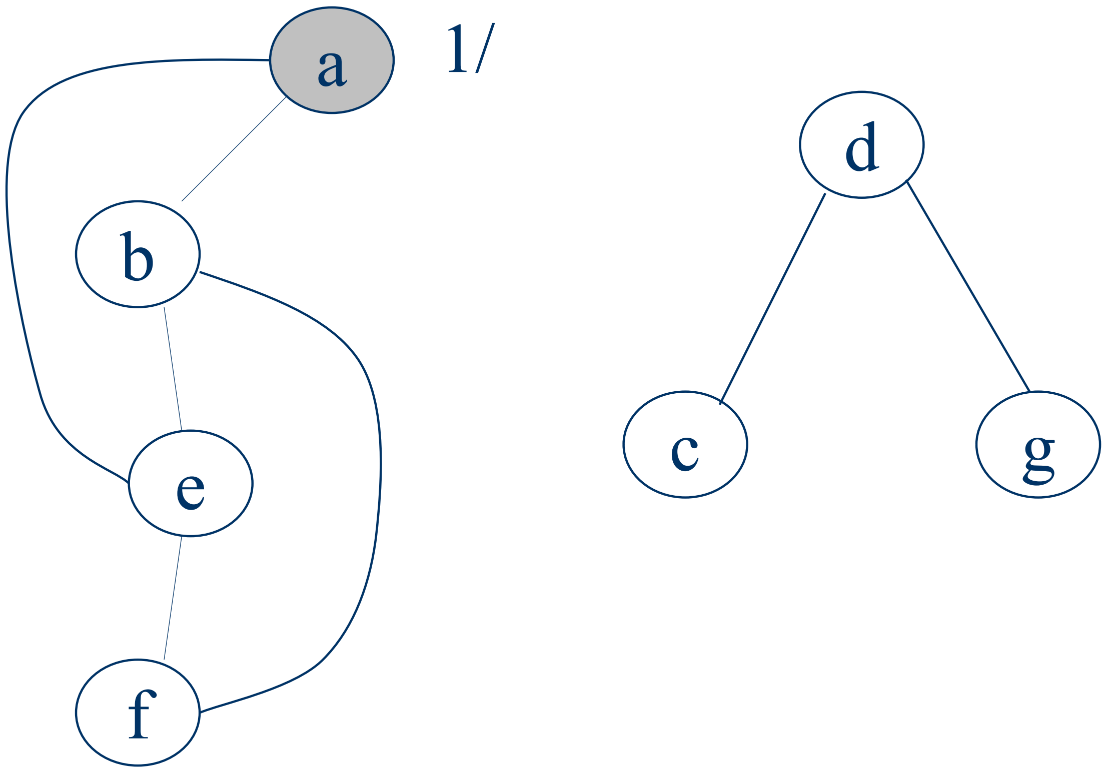
Example –directed graph (continued)



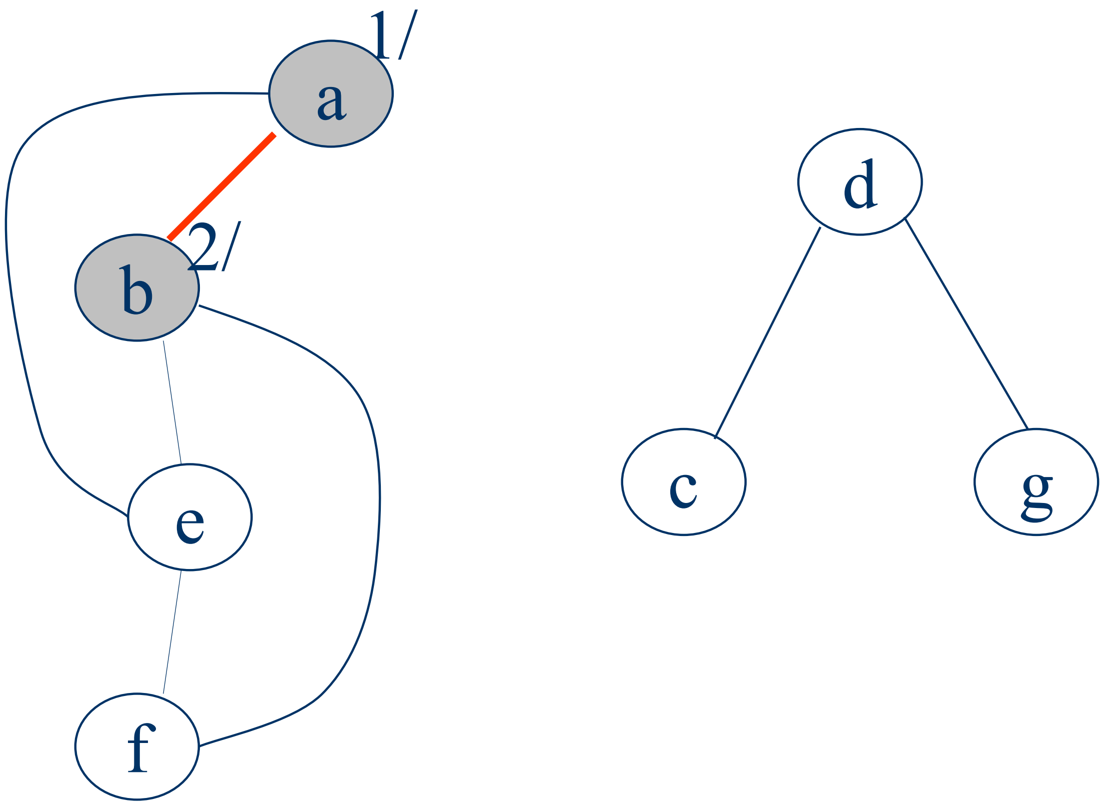
DFS Algorithm (Example – undirected graph)



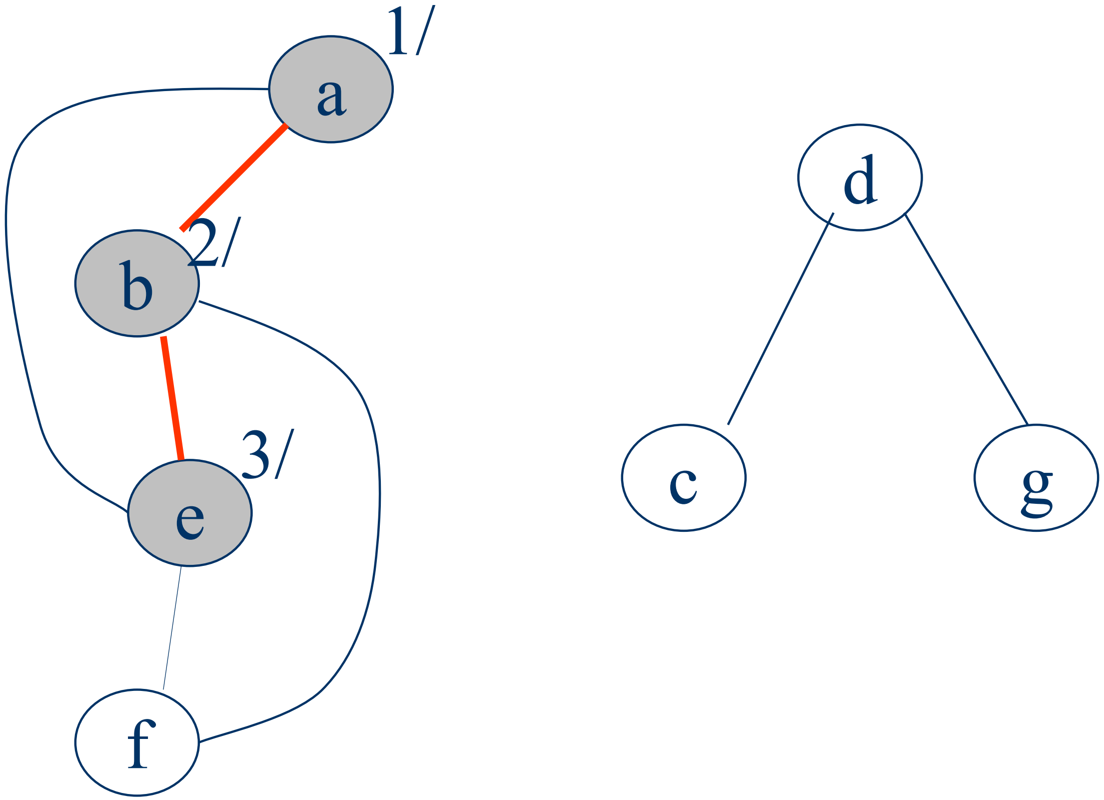
Example – undirected graph (continued)



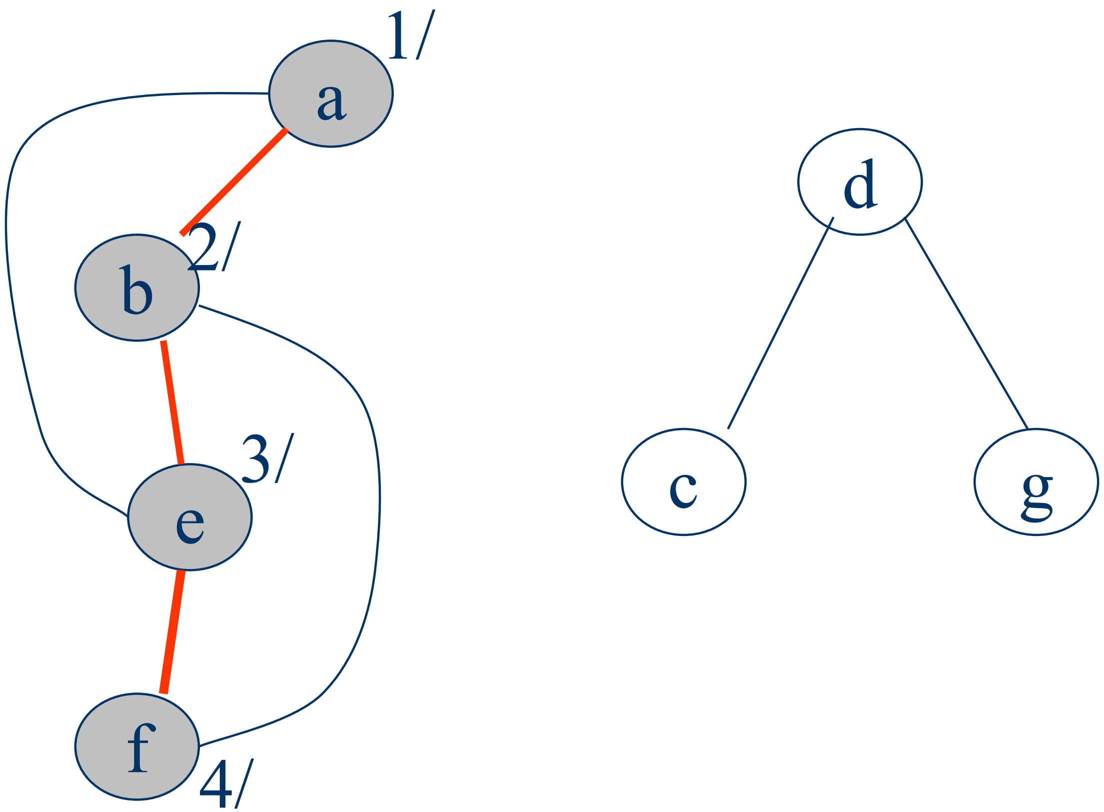
Example – undirected graph (continued)



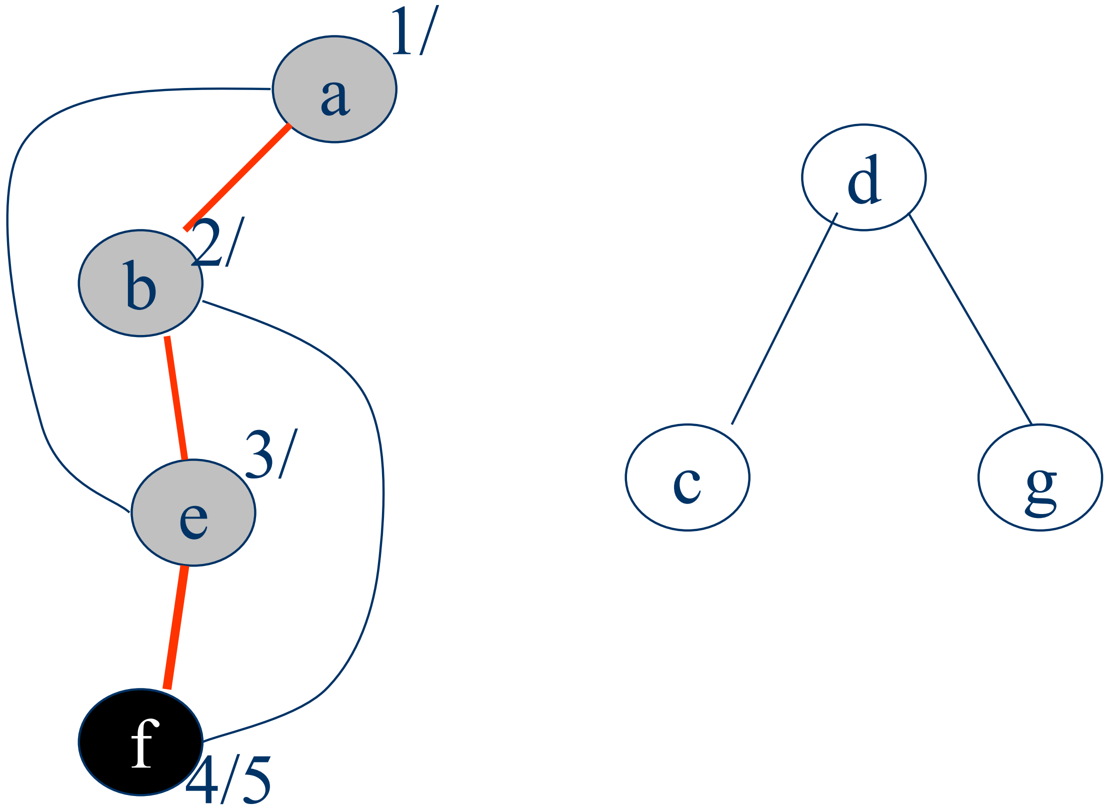
Example – undirected graph (continued)



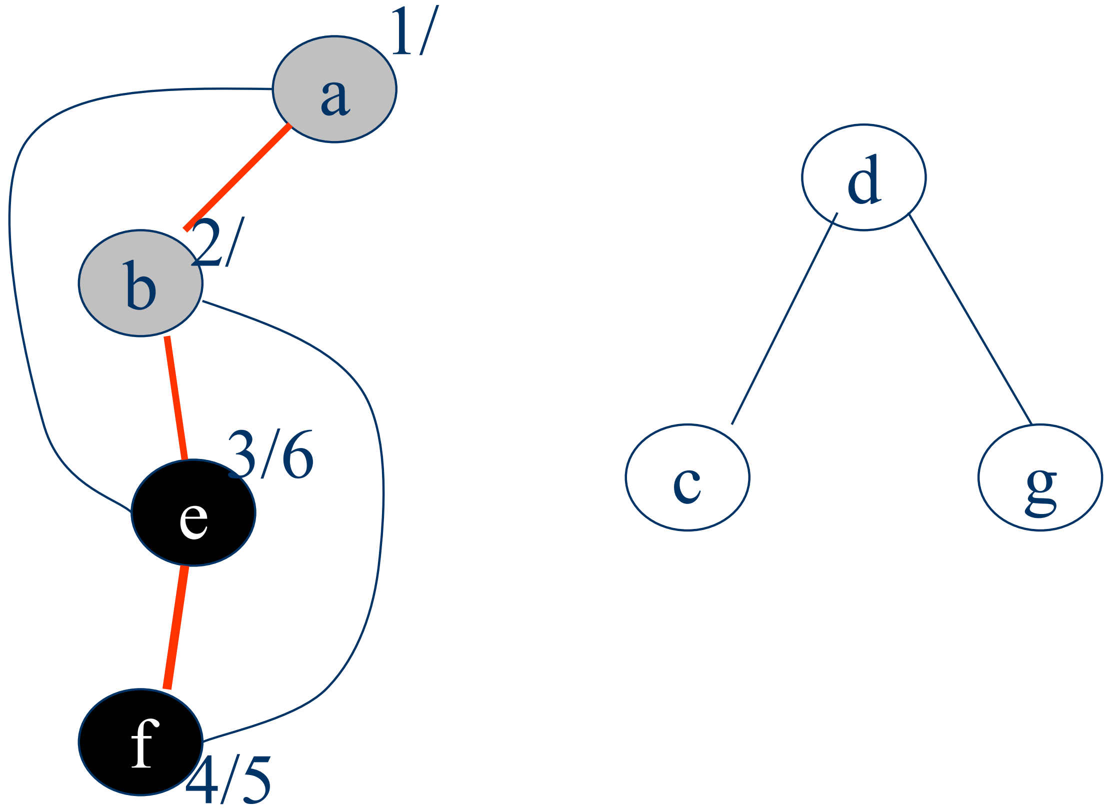
Example – undirected graph (continued)



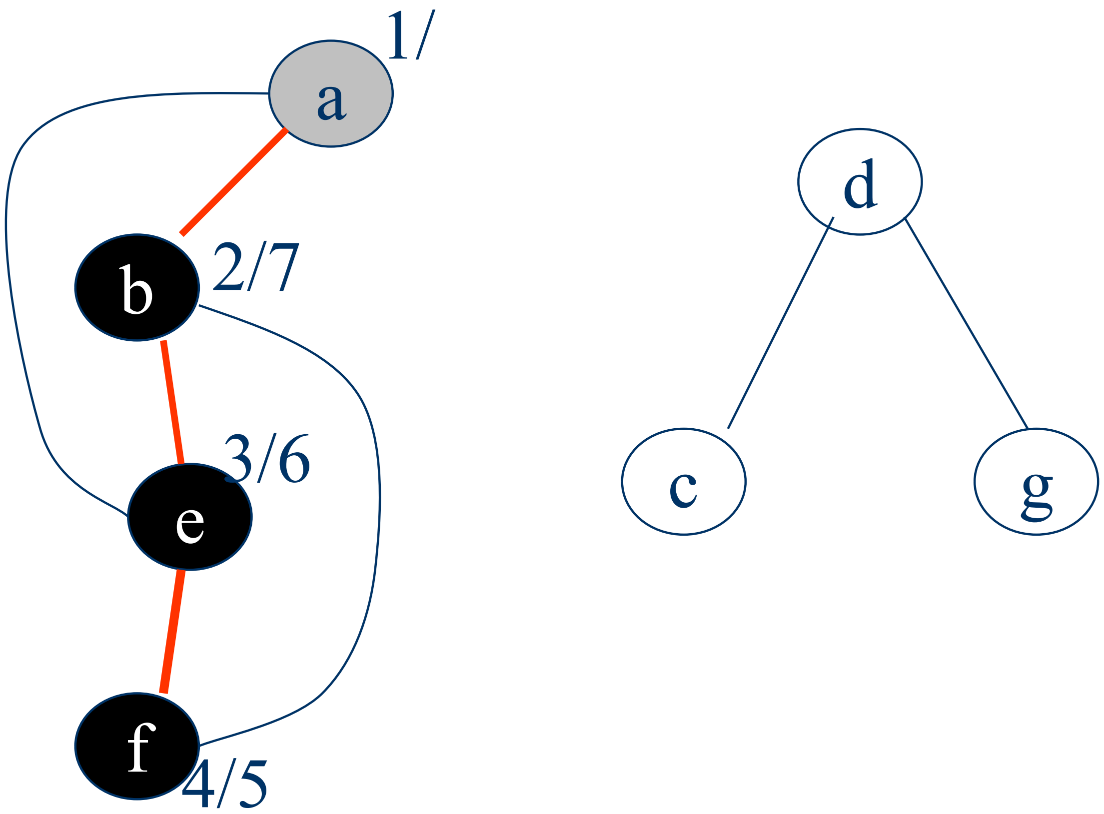
Example – undirected graph (continued)



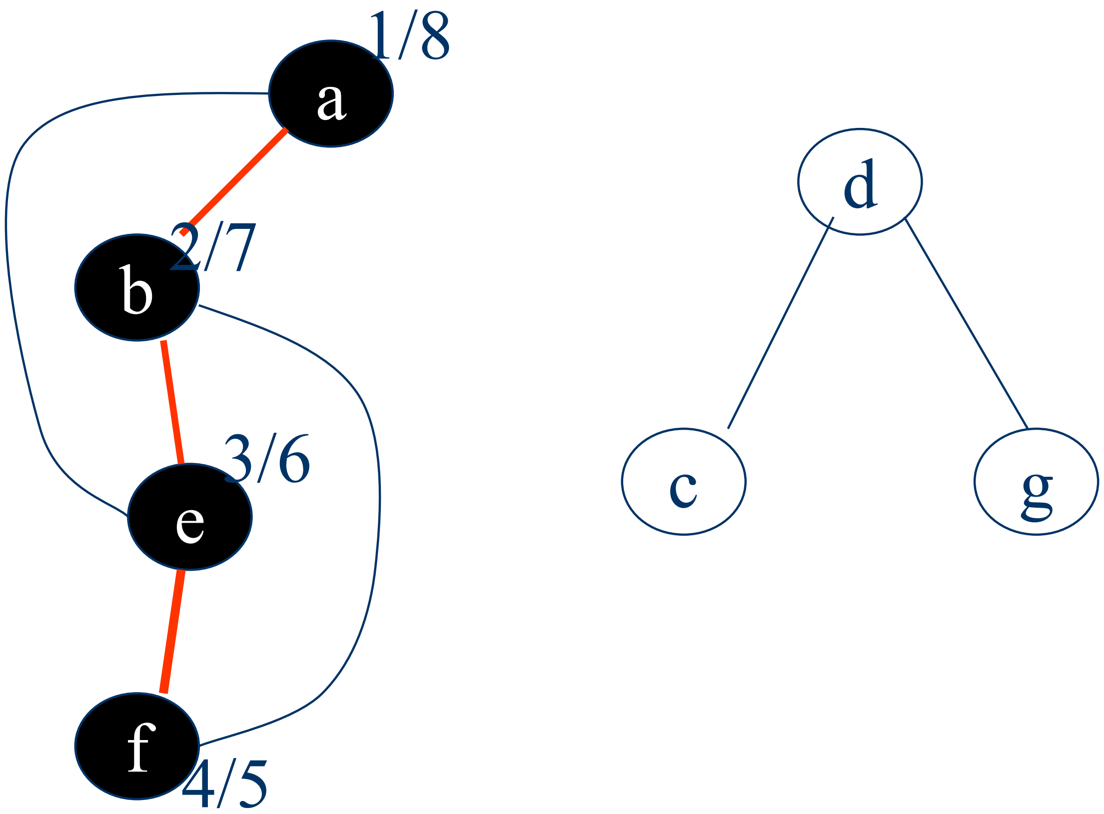
Example – undirected graph (continued)



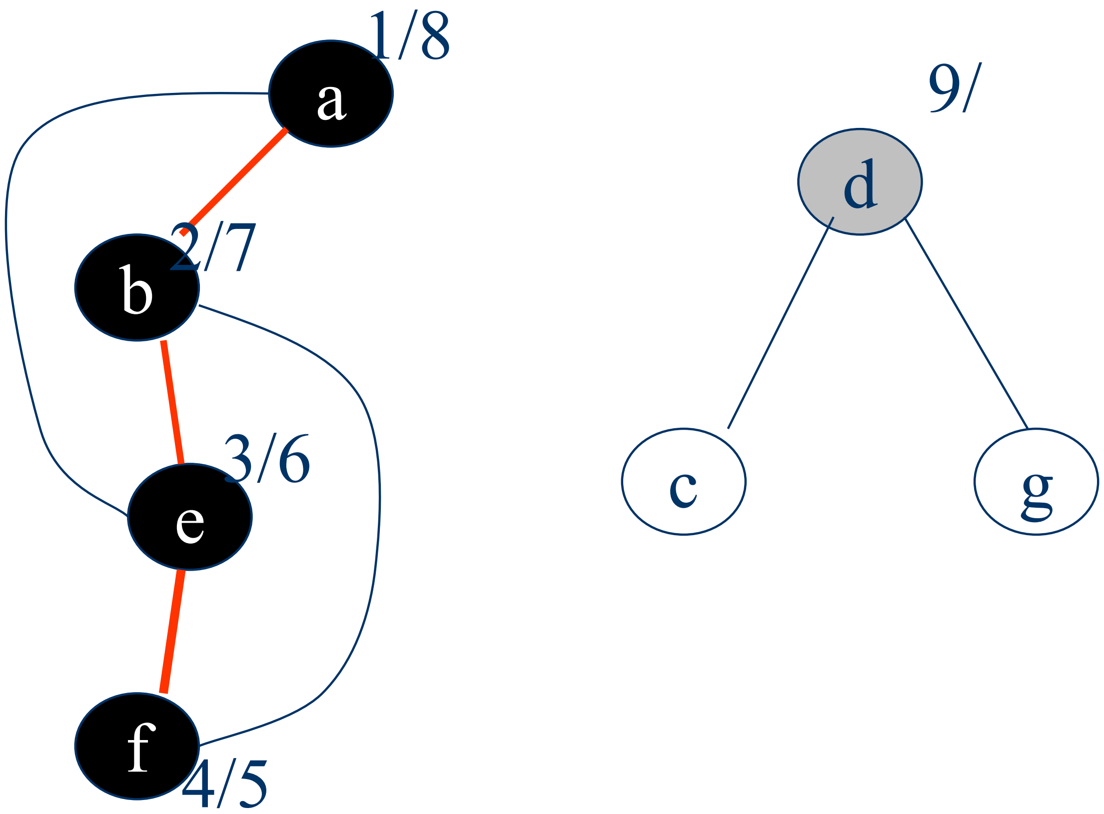
Example – undirected graph (continued)



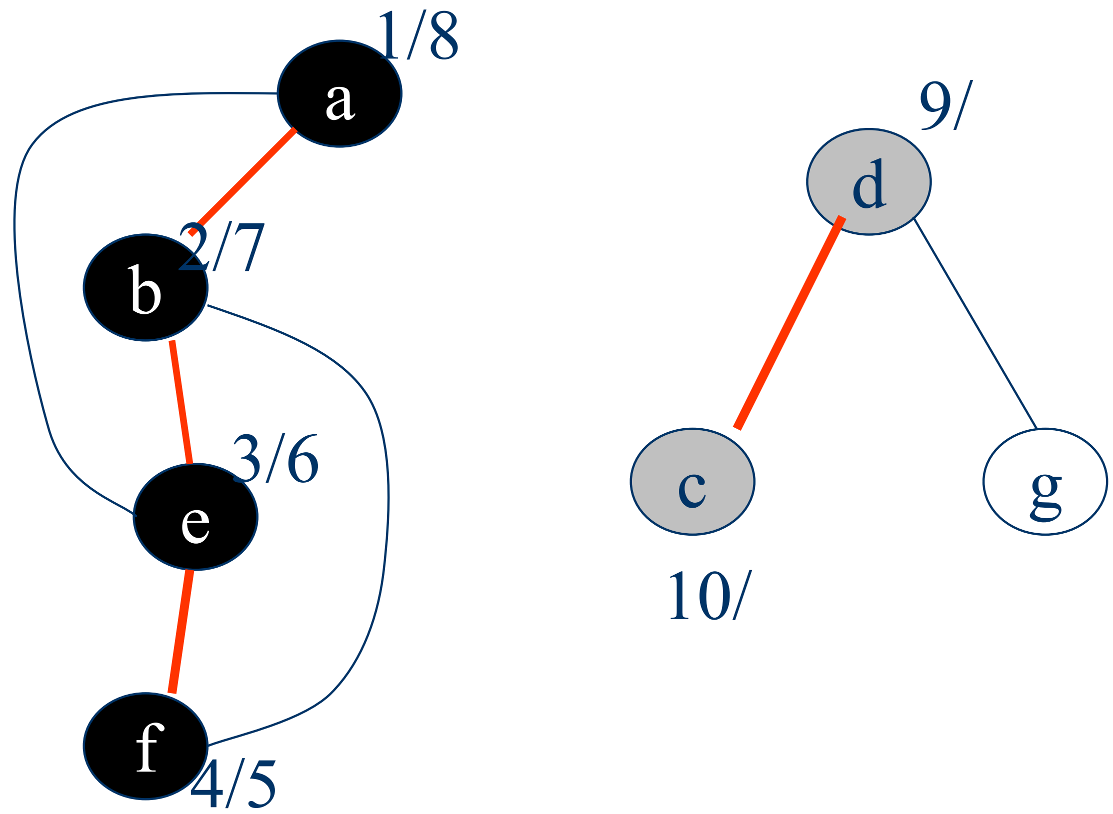
Example – undirected graph (continued)



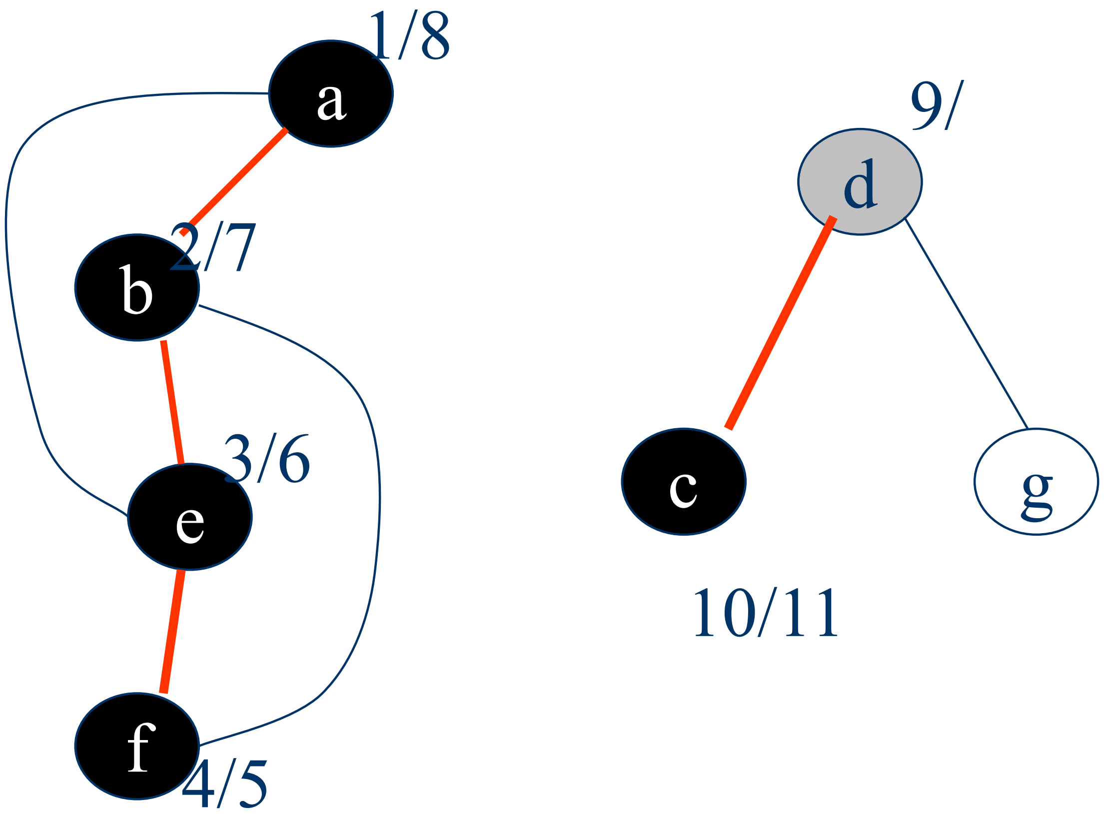
Example – undirected graph (continued)



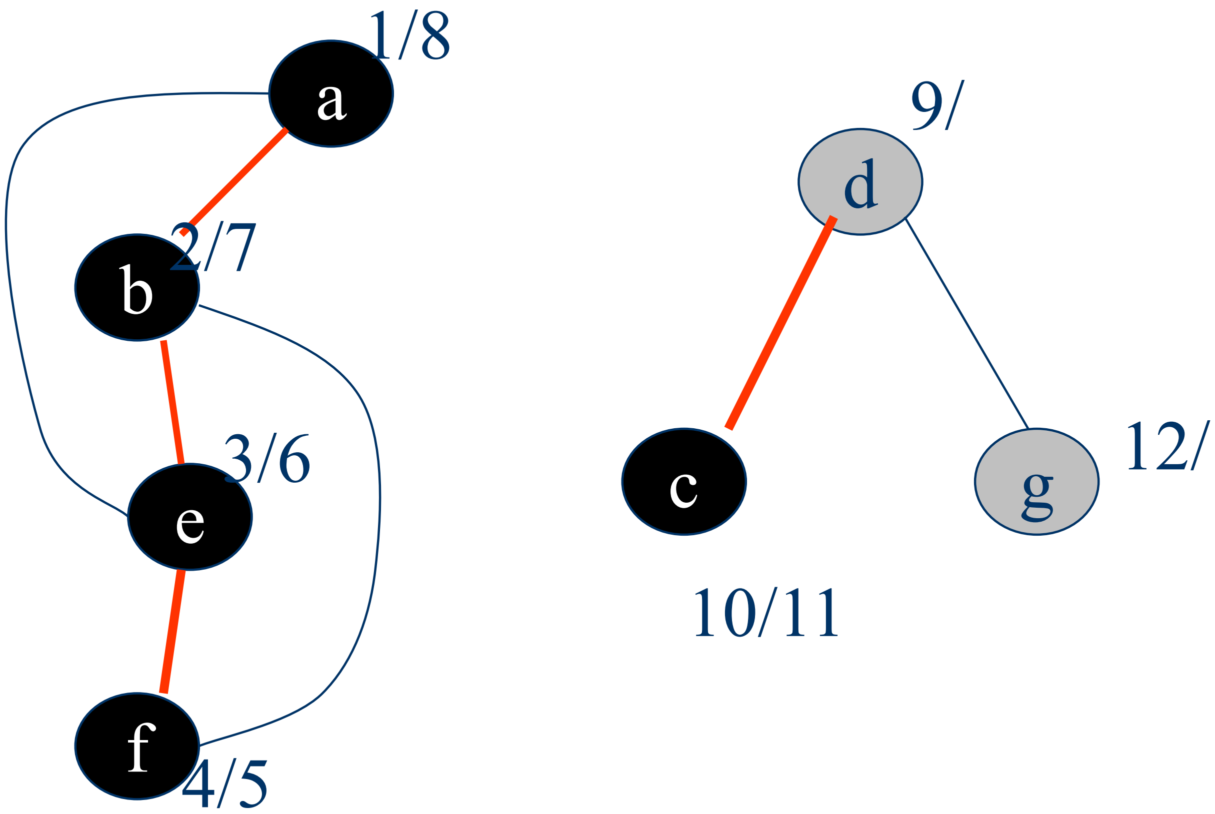
Example – undirected graph (continued)



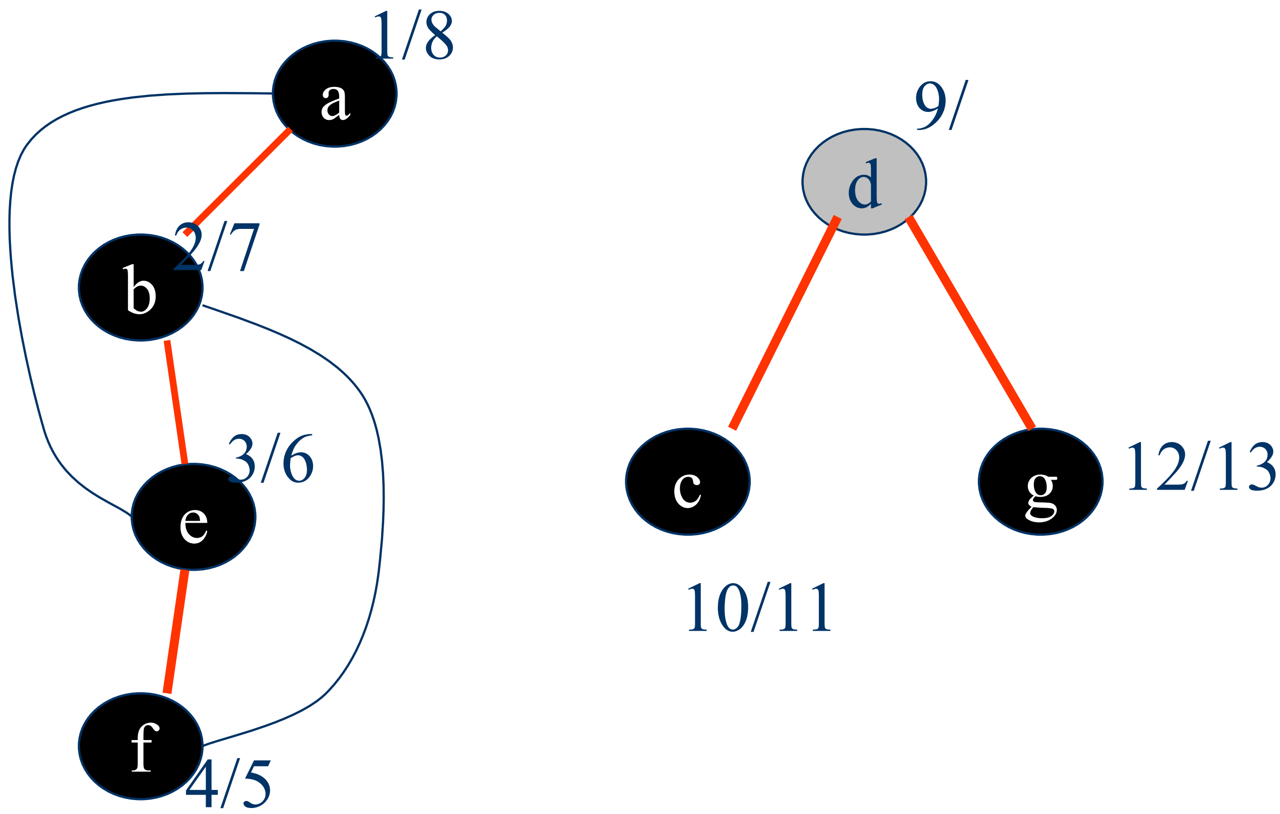
Example – undirected graph (continued)



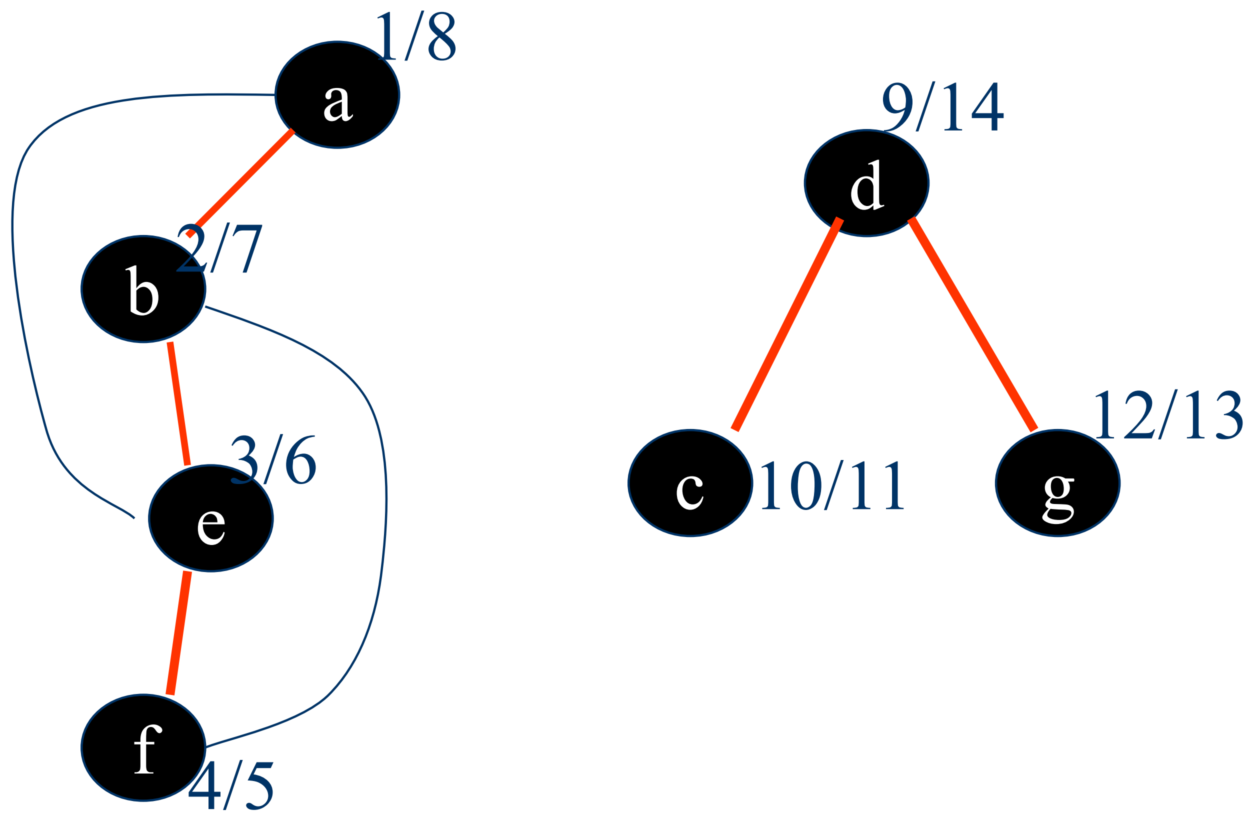
Example – undirected graph (continued)



Example – undirected graph (continued)



Example – undirected graph (continued)



What does DFS do?

Given a graph G, it traverses all vertices of G and

- Constructed a collection of rooted trees
- Outputs three arrays: d, f, π

DFS forest: DFS creates a **depth-first forest**

$F = (V, E_f)$,

where $E_f = \{(u.\pi, u) \mid u \in V, u.\pi \neq \text{Nil}\}$

π is computed in the DFS-VISIT calls

Running time analysis of DFS

DFS(G)

for each u in V

do u.color ← white;

u.π ← NIL;

time=0; $O(V)$

for each u in V

do if u.color white // $O(V)$

then DFS-VISIT(u); //T(u)

DFS-VISIT(u)

u.color =gray;

time ← time+1;

u.d ← time; // $O(1)$

for each v ∈ adj[u]

do if v.color =white // $O(|adj[u]|)$

then v.π ← u;

DFS-VISIT(v);

u.color ← black;

time ← time+1;

u.f ← time; // $O(1)$

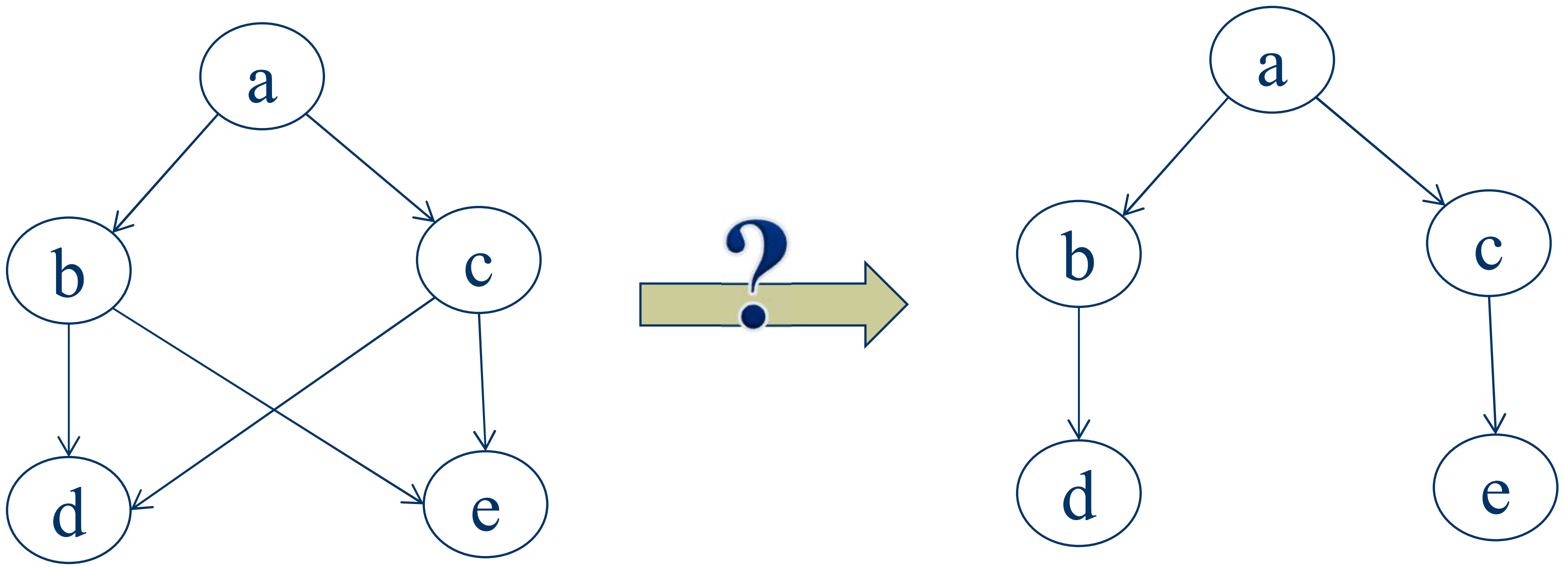
Running time analysis of DFS

- Since each vertex is grayed exactly once, for each u, DFS-VISIT(u) is called exactly once.
- During DFS-VISIT(u) , Adj[u] is scanned exactly once, so T(u)= O(1 + |Adj[u]|).
- So, $\sum$ T(u) = O(V+E)

Hence, total time $T(V, E) = O(V+E)$

Problem

- Consider: how does the order of vertices in adjacency list influence the result of DFS?



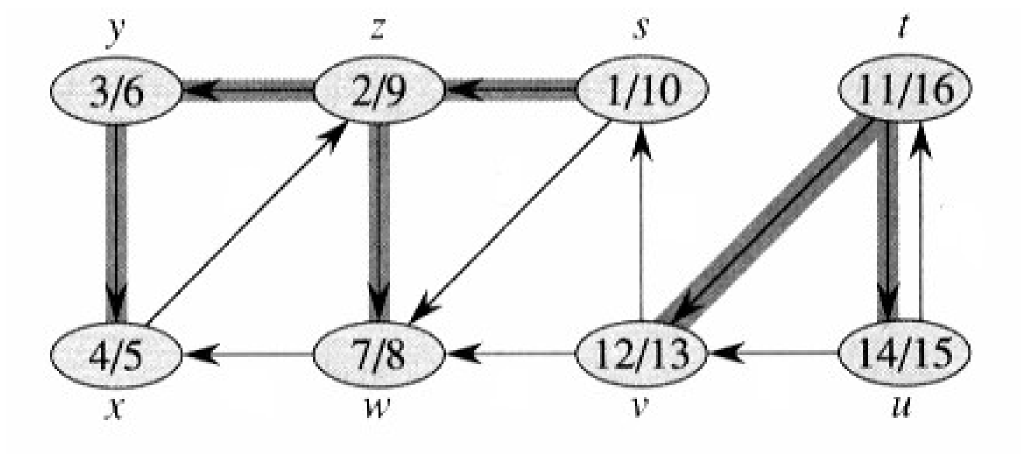
Properties of DFS (1)

- **Definition**: In a rooted forest(especially, DFS forest), any vertex ***u*** on the simple path from the root to ***v*** is called an ancestor of ***v***, and ***v*** is called a descendant of ***u***.
- $u = v.\pi$ if and only if DFS-VISIT(v) was called during a search of u's adjacency list. u is called the parent of v.
- Vertex v is a descendant of vertex u in the depth-first forest if and only if v is discovered during the time in which u is gray.

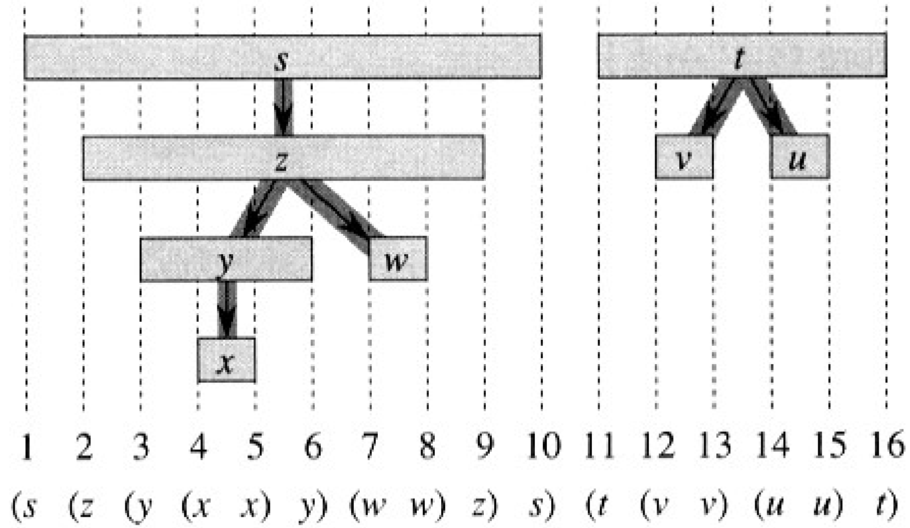
Properties of DFS (2)

- The discovery and finishing time have **parenthesis structure**.

In detail, if we use "(*u*" to represent the discovery of *u*, and "*u*)" the finishing of *u*, then the history of discoveries and finishings makes a well-formed expression in the sense that the parentheses are properly nested



(a)



(b)

Properties of DFS (2)

- Theorem 22.7 parenthesis theorem(括号定理): In any depth-first search of a (directed or undirected) G=(V, E), for any two vertices u and v, exactly one of the following three conditions holds:
  - the intervals $[v.d, v.f]$ and $[u.d, u.f]$ are entirely disjoint, and neither u nor v is a descendant of the other in the depth-first forest.
  - the interval $[u.d, u.f]$ is contained entirely within the interval $[v.d, v.f]$, and u is a descendant of v in a depth-first tree, or
  - the interval $[v.d, v.f]$ is contained entirely within the interval $[u.d, u.f]$, and v is a descendant of u in a depth-first tree.

Proof of *Parenthesis Theorem*

- We begin with the case in which $u.d < v.d$. We consider two subcases, according to whether $v.d < u.f$ or not.
  - $v.d < u.f$, so v was discovered while u was still gray, which implies that v is a descendant of u. Moreover, since v was discovered more recently than u, all of its outgoing edges are explored, and v is finished, before the search returns to and finishes u. In this case, therefore, the interval $[v.d, v.f]$ is contained entirely within the interval $[u.d, u.f]$.
  - In the other subcase, $u.f < v.d$, and by inequality (22.2), thus, $u.d < u.f < v.d < v.f$, that is, the intervals $[u.d, u.f]$ and $[v.d, v.f]$ are disjoint. Because the intervals are disjoint, neither vertex was discovered while the other was gray, and so neither vertex is a descendant of the other.
- The case in which $v.d < u.d$ is similar, with the roles of u and v reversed in the above argument

Nesting of descendants' intervals

- Corollary 22.8: Vertex v is a proper descendant of vertex u in the depth-first forest for a graph G if and only if $u.d < v.d < v.f < u.f$.

Properties of DFS (3)

- ***Theorem 22.9*** (White-Path Theorem)
  - In a depth-first forest of a (directed or undirected) graph $G = (V, E)$, vertex v is a descendant of vertex u if and only if at time $u.d$ that the search discovers u, vertex v can be reached from u along a path consisting entirely of white vertices.

Proof of *White-Path Theorem*

- =>Assume v is a descendant of u, there is a unique path from u to v in the DFS forest. Let w be an arbitrary vertex on the path from u to v in the depth-first forest, then w is a descendant of u. By the nesting of descendants' interval corollary, $u.d<w.d$, and so at time $u.d$, w is white.

Proof of *White-path theorem*

- <=(By contradiction) Suppose at time $u.d$, there is a path from u to v consisting only of white vertices, but v **does not** become a descendant of u in the depth-first tree. Without loss of generality, assume that every other vertex along the path becomes a descendant of u. **(otherwise, let v be the closest vertex to u along the path that does not become a descendant of u.)** Let w be the predecessor of v in the path. Hence, w is a descendant of u, and by Corollary *Nesting of descendants' intervals*, $w.f <= u.f$. Note that v must be discovered after u is discovered, but before w is finished. Therefore, $u.d<v.d< w.f <= u.f$. *Parenthesis Theorem* then implies that the interval [$v.d$, $v.f$] is a subinterval of [$u.d$,$u.f$]. By Corollary *Nesting of descendants' intervals*, v must be a descendent of u.

Classification of Edges (directed)

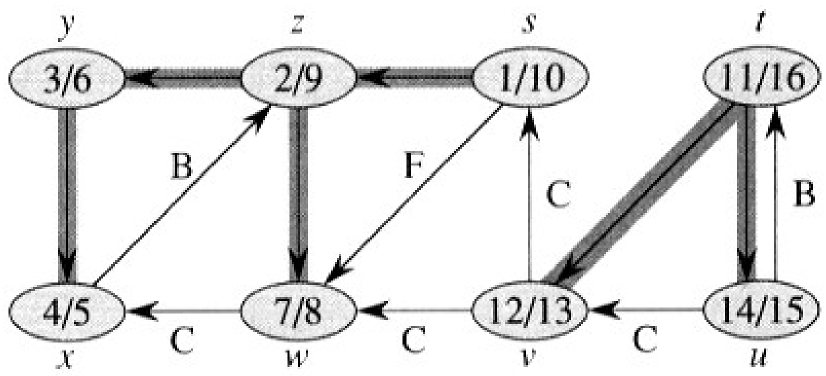
- Let $G_\pi = (V, E_\pi)$ be the depth-first forest produced by a depth first search on **directed** graph $G = (V, E)$, then the edges of G can be classified into four categories(**the non-tree edges are classified according to the relationship with the end points**):

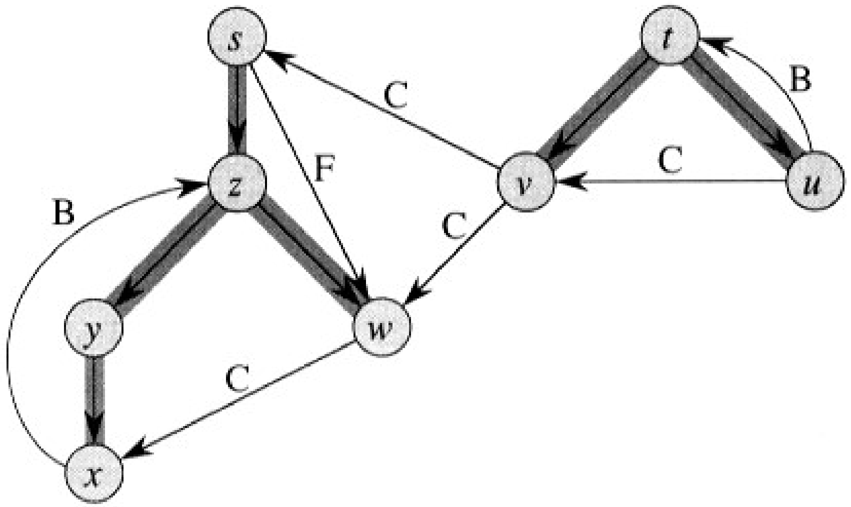
  1. **Tree edges:** edges in G_π。(树边)
  2. **Back edges:** edges (u, v) connecting a vertex u to an ancestor v. Self-loops in directed graphs are considered to be back edges.(返回边)
  3. **Forward edges:** those non-tree edges (u, v) connecting a vertex u to a descendant v.(前向边)
  4. **Cross edges:** all other edges. They can go between vertices in the same depth-first tree as long as neither vertex is ancestor of the other, or they go between vertices in different trees. (交叉边)

Recalling that for each pair of vertices u,v, there are only three probabilities between them:

u is a descendant of v, opposite, neither

Classification of Edges (Directed)





Observation (directed)

- In a depth-first search, an edge (u, v) can be classified according to the **color of v** when (u, v) is first explored:
  - WHITE indicates a tree edge
  - GRAY indicates a back edge
  - BLACK indicates a forward or cross edge
    - How to distinguish a forward edge and a cross edge?
      - $u.d < v.d$ means a forward edge
      - $u.d > v.d$ means a cross edge

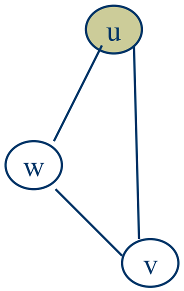
– **Accordingly, DFS algorithm can be modified to classify edges as it encounters them. How?**

Classification of Edges (Undirected)

- In an undirected graph, (u, v) and (v, u) are in fact the same edge. To avoid ambiguity, we classify the edge according to whichever of (u, v) or (v, u) is encountered first during the execution of the DFS algorithm.
- **Theorem** 22.10 In a depth-first search of an undirected graph G, every edge of G is either a tree edge or a back edge.
- 无向图中只有树边和返回边.

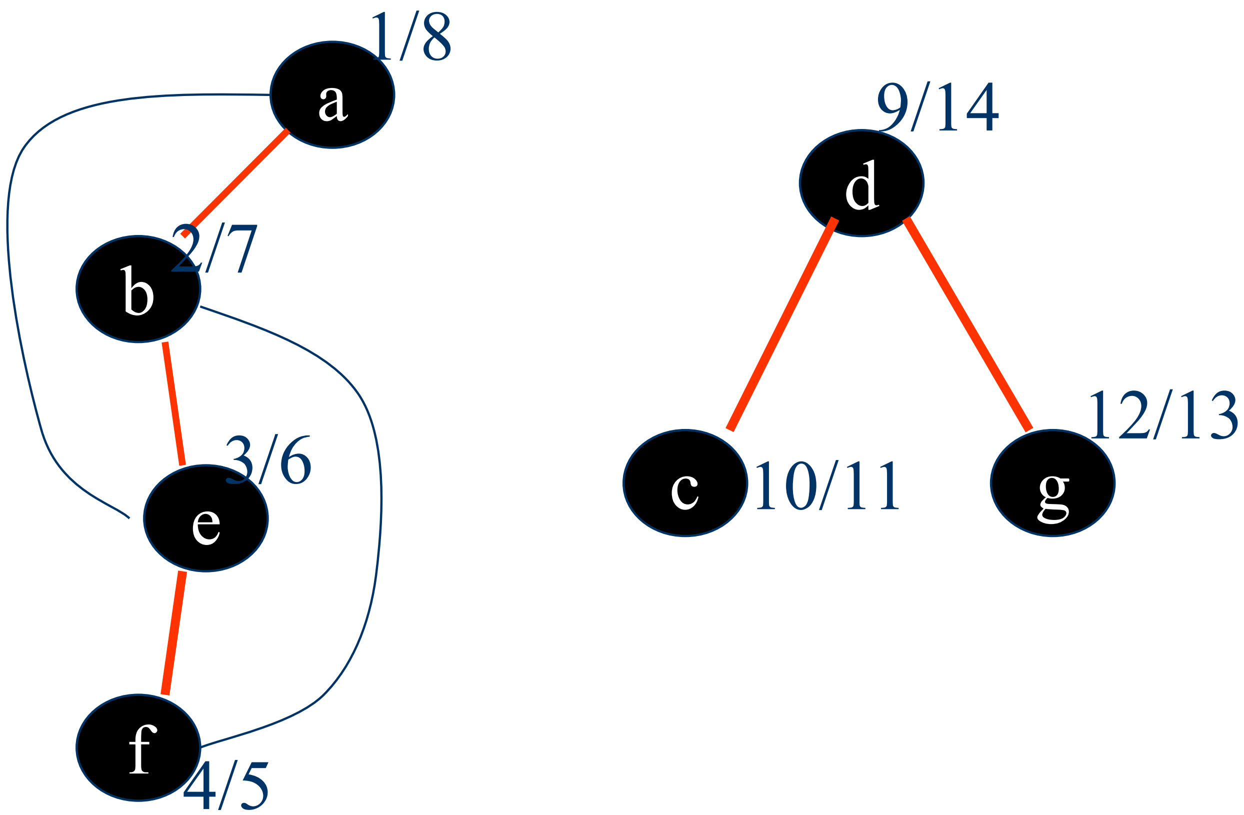
Proof of Theorem 22.10

- Let (u, v) be an arbitrary edge of G, and suppose without loss of generality that $u.d < v.d$. Then the search must discover and finish v before it finishes u (while u is gray), since v is on u's adjacency list.
  - If the first time that the search explores edge (u, v), it is in the direction from u to v, then v is undiscovered (white) until that time, for otherwise the search would have explored this edge already in the direction from v to u. Thus, (u, v) becomes a tree edge.



  - If the search explores (u, v) first in the direction from v to u, then (u, v) is a back edge, since u is still gray at the time the edge is first explored.

Classification of Edges (Undirected)



Conclusion

- The DFS algorithm
- The time complexity of DFS algorithm
- Properties of the DFS
  - v是u的后代当且仅当扫描v时,u是灰色的.
  - 括号定理
  - 后代的区间是祖先的子区间
  - 白色路径定理
- Classification of edges
  - 有向图4种
  - 无向图2种

Homework

- 22.3-2
- 22.3-6
- 22.3-12(a correctness proof is preferred)

Next Class

- Topological Sort
  -- (for directed acyclic graph)

- Strongly Connected Components Decomposing
  -- (for directed graph)