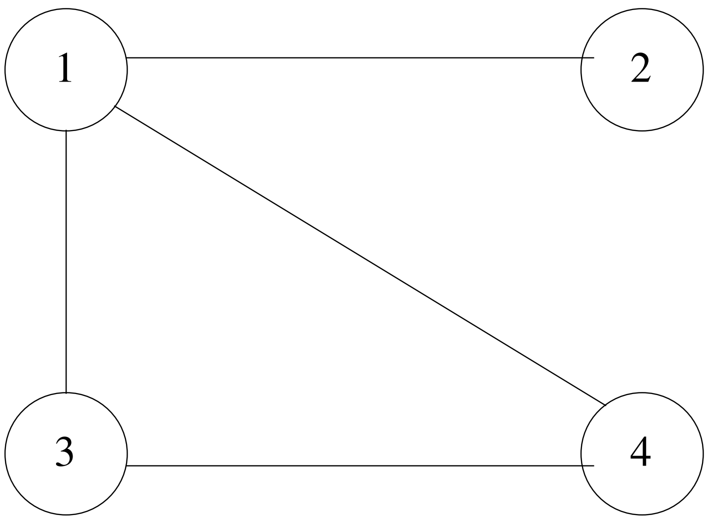
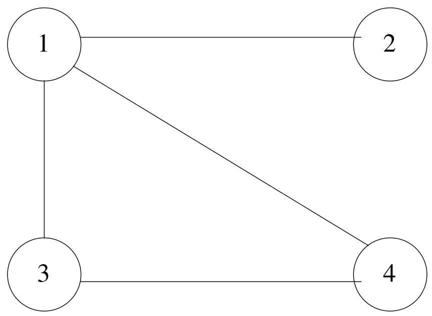
Graphs

- **Graph** $G = (V, E)$ **has vertices (nodes) V and edges (arcs) E.**
- **Graph can be directed or undirected**
- **Graph can represent any situation with objects and pairwise relationships.**



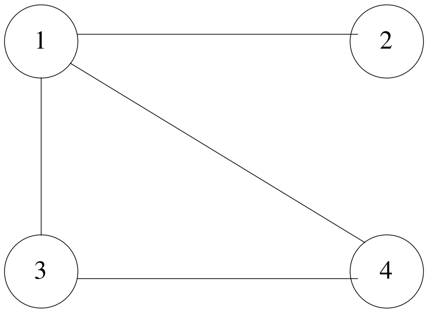
Representations



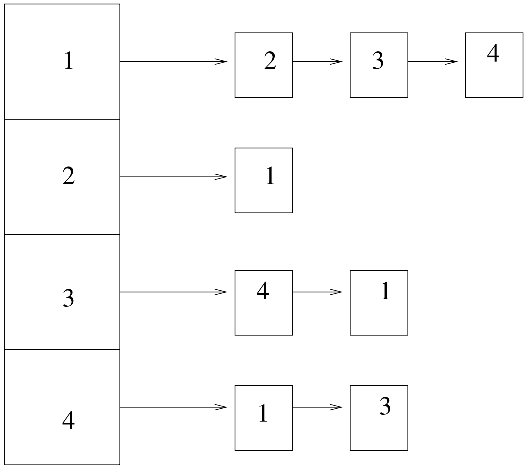
Adjacency Matrix

	1	2	3	4
1	0	1	1	1
2	1	0	0	0
3	1	0	0	1
4	1	0	1	0

Representations



Adjacency List



Comparison

	Space	**Query Time**	**All neighbors time**
Matrix	$O(V^2)$	$O(1)$	$O(V)$
List	$O(E)$	$O(\textbf{degree})$	$O(\textbf{degree})$

- **For a simple graph (no double edges)** $E \leq V^2 = O(V^2)$
- **For a connected graph** $E \geq V - 1$
- **For a tree** $E = V - 1$

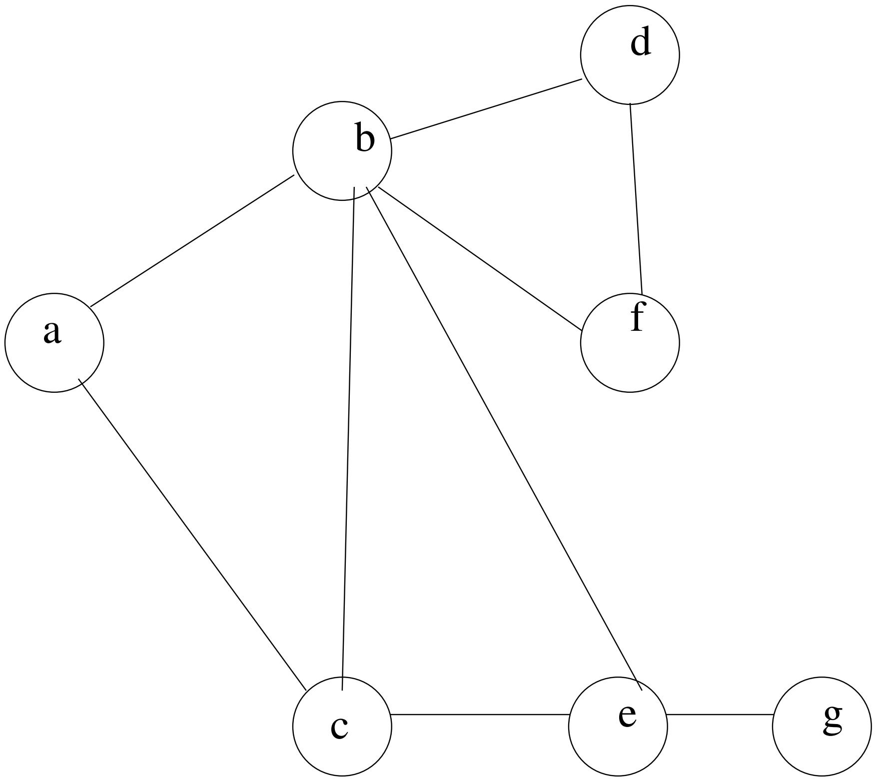
Breadth First Search

- Discover vertices in order of distance from the source.
- Works for undirected and directed graphs. Example is for undirected graphs.

Breadth First Search

```
BFS(G, s)
for each vertex u ∈ V[G] − {s}
        do color[u] ← WHITE
           d[u] ← ∞
           π[u] ← NIL
color[s] ← GRAY
d[s] ← 0
π[s] ← NIL
Q ← ∅
ENQUEUE(Q, s)
while Q ≠ ∅
        do u ← DEQUEUE(Q)
           for each v ∈ Adj[u]
                do if color[v] = WHITE
                      then color[v] ← GRAY
                           d[v] ← d[u] + 1
                           π[v] ← u
                           ENQUEUE(Q, v)
           color[u] ← BLACK
```

Example



Running Time:

for each $u \in V$
  do for each $v \in \mathbf{Adj}(v)$
    do Something $O(1)$ **time**

Each edge and vertex is processed once:

$$O(E + V) = O(E)$$

Depth First Search

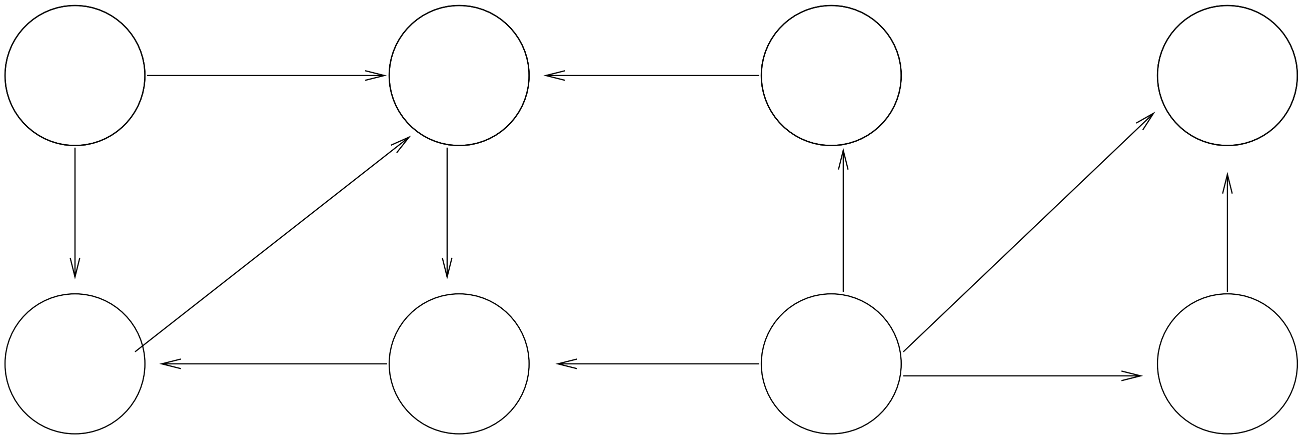
- **More interesting than BFS**
- **Works for directed and undirected graphs. Example is for directed graphs.**
- **Time stamp nodes with discovery and finishing times.**
- **Lifetime: white, $d(v)$, grey, $f(v)$, black**

Code

```
DFS(G)
for each vertex u ∈ V[G]
        do color[u] ← WHITE
           π[u] ← NIL
time ← 0
for each vertex u ∈ V[G]
        do if color[u] = WHITE
              then DFS-VISIT(u)
```

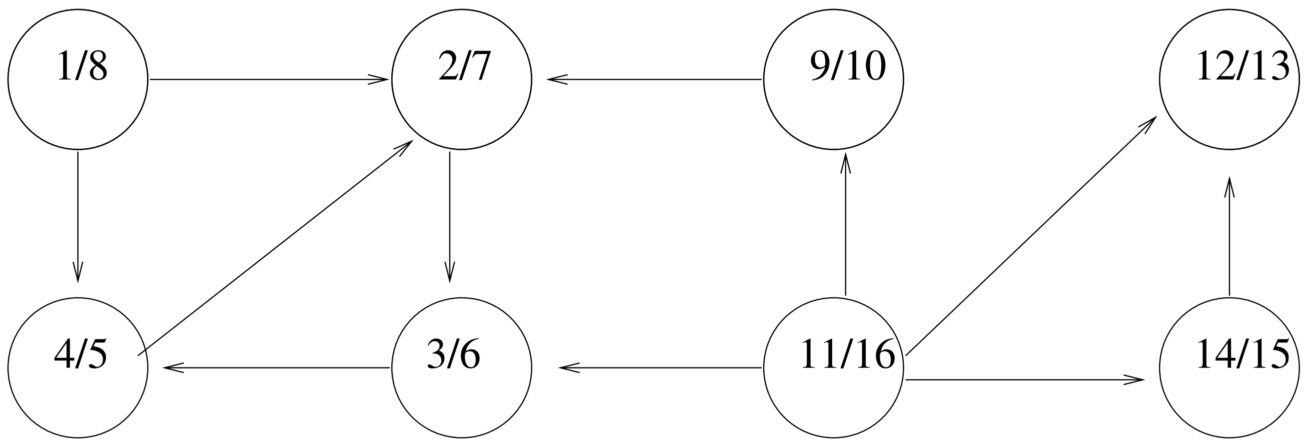
```
DFS-Visit(u)
color[u] ← GRAY                 ▷ White vertex u has just been discovered.
time ← time + 1
d[u] ← time
for each v ∈ Adj[u]             ▷ Explore edge (u, v).
        do if color[v] = WHITE
              then π[v] ← u
                   DFS-VISIT(v)
color[u] ← BLACK                ▷ Blacken u; it is finished.
f[u] ← time ← time + 1
```

Example



Labeled

$d(v)/f(v)$



Structure

Parenthesization

If we represent the discovery of vertex u with a left parenthesis "$(u$" and represent its finishing by a right parenthesis "$u)$", then the history of discoveries and finishings makes a well-formed expression in the sense that the parentheses are properly nested.

Parenthesis theorem In any depth-first search of a (directed or undirected) graph $G = (V, E)$, for any two vertices u and v, exactly one of the following three conditions holds:

- the intervals $[d[u], f[u]]$ and $[d[v], f[v]]$ are entirely disjoint, and neither u nor v is a descendant of the other in the depth-first forest,
- the interval $[d[u], f[u]]$ is contained entirely within the interval $[d[v], f[v]]$, and u is a descendant of v in a depth-first tree, or
- the interval $[d[v], f[v]]$ is contained entirely within the interval $[d[u], f[u]]$, and v is a descendant of u in a depth-first tree.

Nesting of descendants' intervals

Vertex v is a proper descendant of vertex u in the depth-first forest for a (directed or undirected) graph G if and only if $d[u] < d[v] < f[v] < f[u]$.

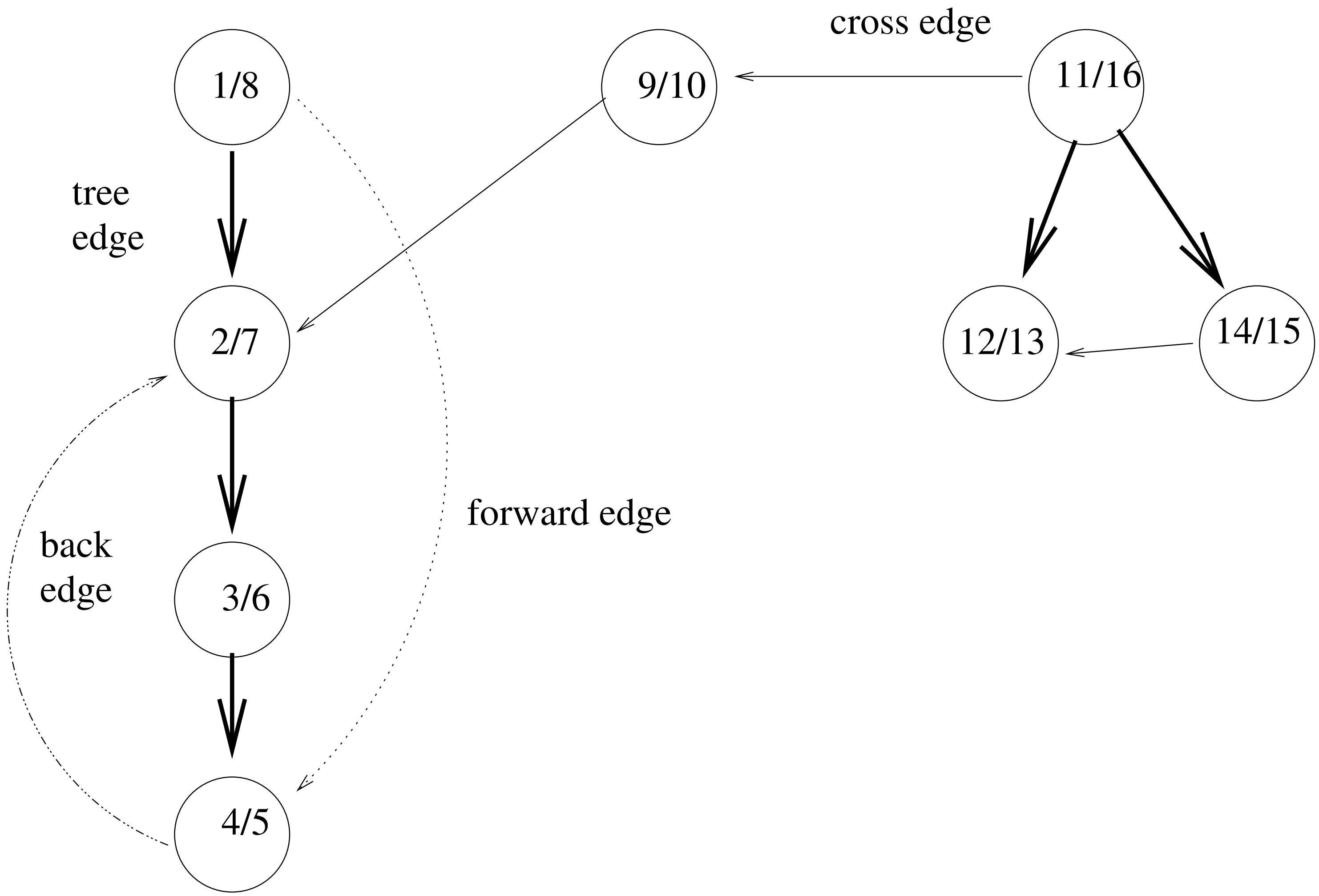
More Structure

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 the time $d[u]$ that the search discovers u, vertex v can be reached from u along a path consisting entirely of white vertices

Edge classification

1. **Tree edges** are edges in the depth-first forest G_π. Edge (u, v) is a tree edge if v was first discovered by exploring edge (u, v).
2. **Back edges** are those edges (u, v) connecting a vertex u to an ancestor v in a depth-first tree. Self-loops, which may occur in directed graphs, are considered to be back edges.
3. **Forward edges** are those nontree edges (u, v) connecting a vertex u to a descendant v in a depth-first tree.
4. **Cross edges** are all other edges. They can go between vertices in the same depth-first tree, as long as one vertex is not an ancestor of the other, or they can go between vertices in different depth-first trees.



cross edge
1/8
9/10
11/16
tree
edge
2/7
12/13
14/15
forward edge
back
edge
3/6
4/5