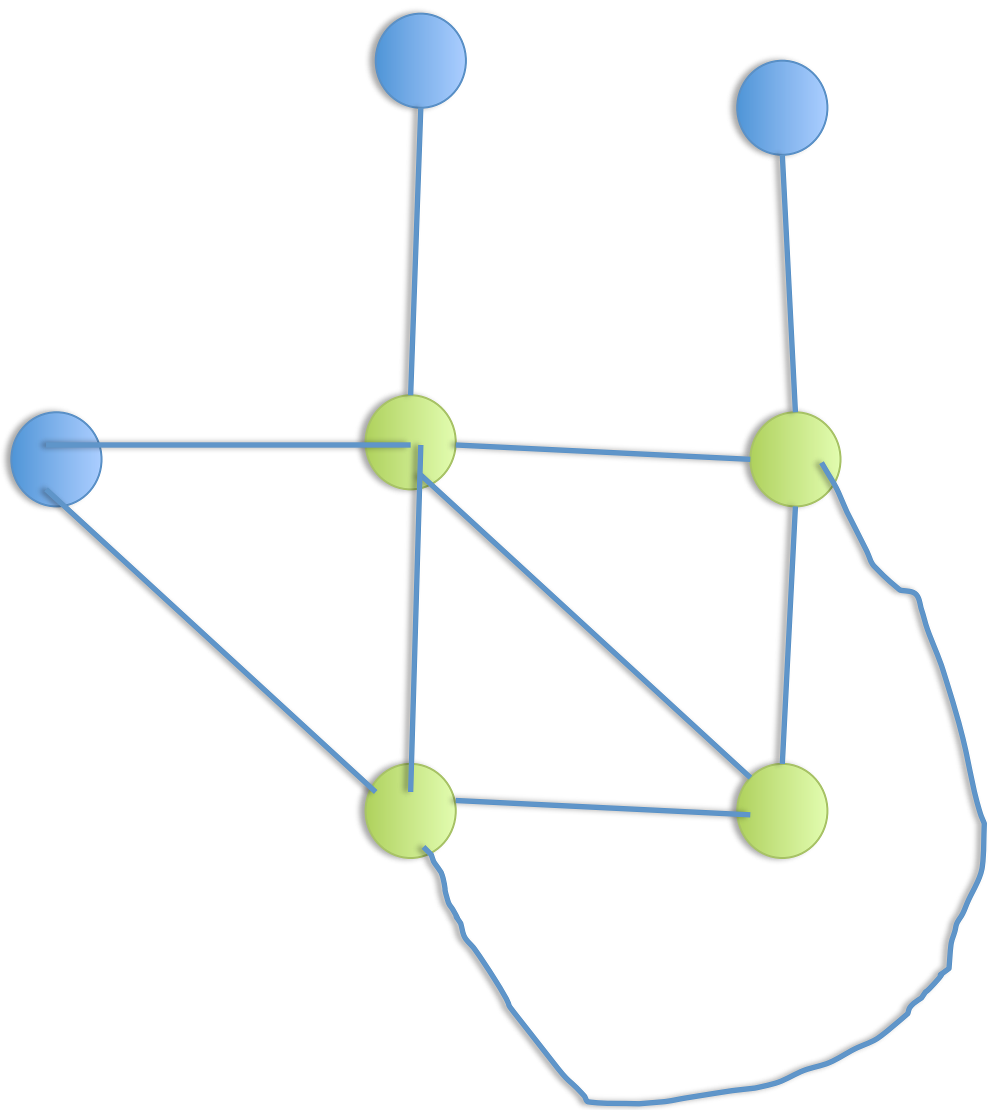
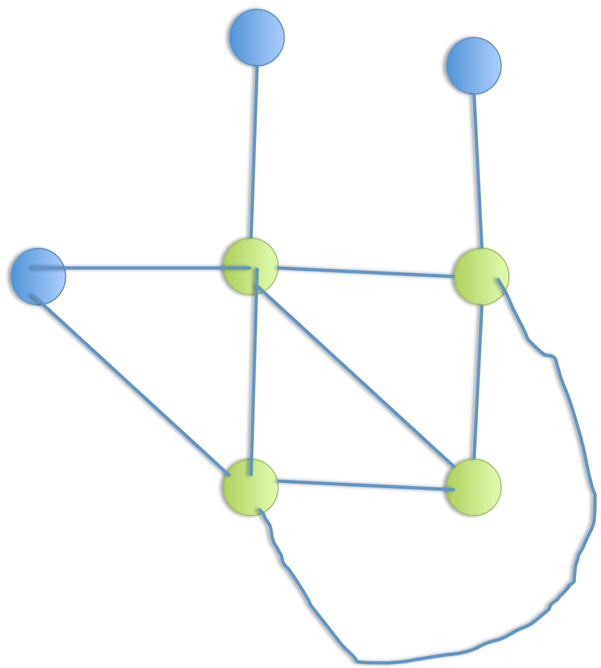
Graphs

- **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.**



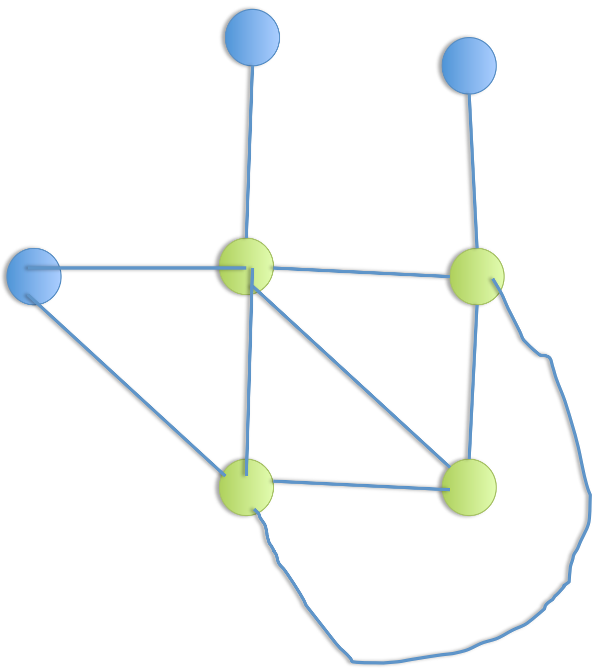
Representations



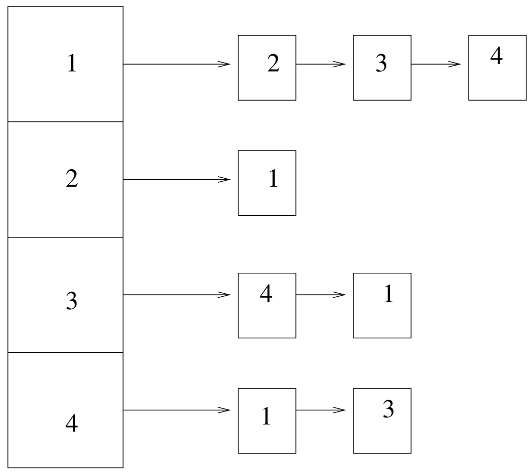
Adjacency Matrix

	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


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

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** $E = V - 1$

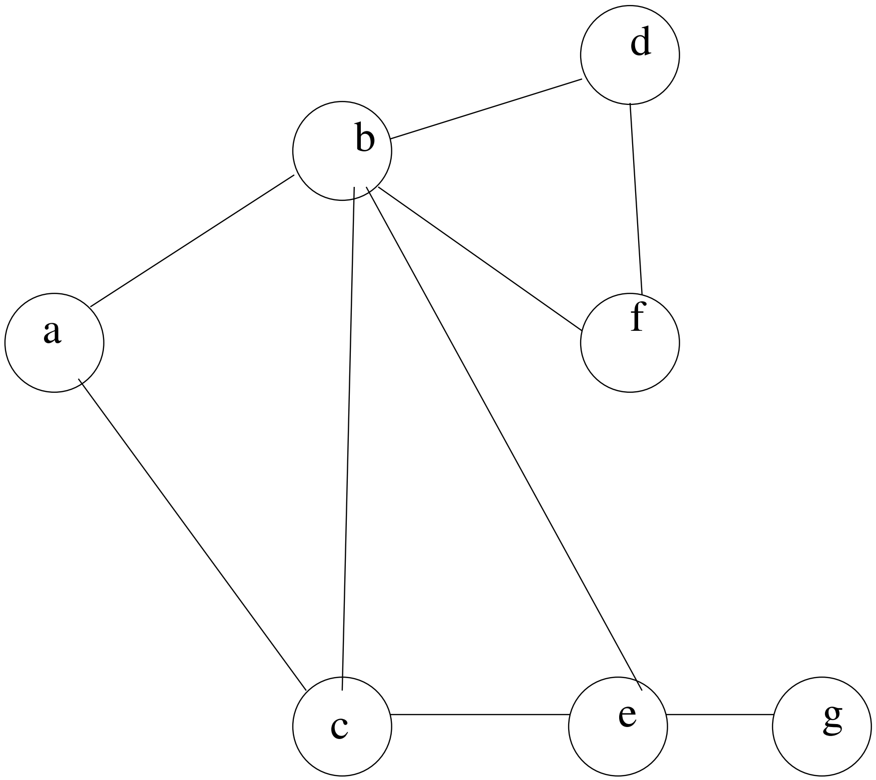
Breadth First Search

- **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 $v \in$ **Adj**(v)
    do Something $O(1)$ **time**

Each edge and vertex is processed once:

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

Depth First Search

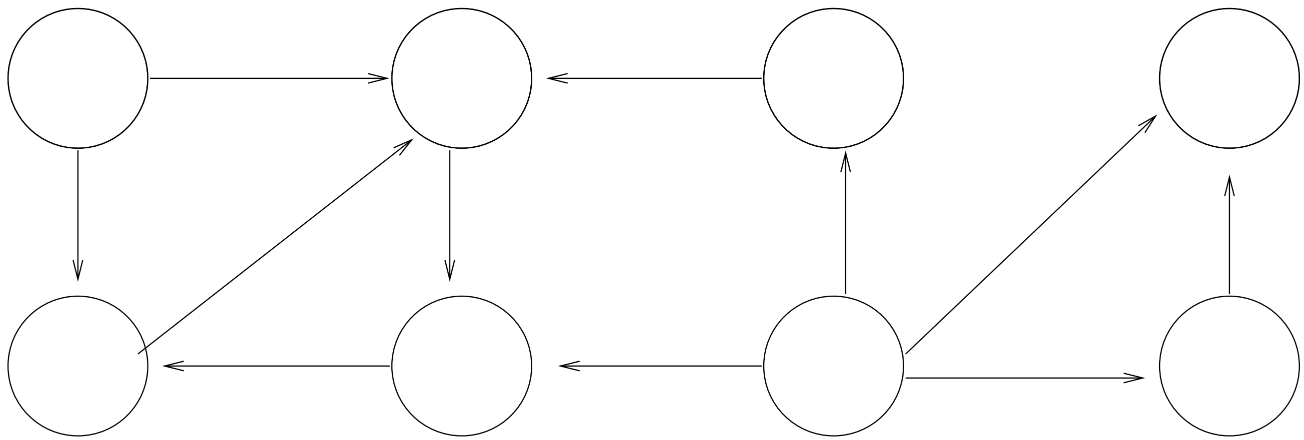
- **More interesting than BFS**
- **Works for directed and undirected graphs. Example is for directed graphs.**
- **Time stamp nodes with discovery and finishing times.**
- **Lifetime: white, $d(v)$, grey, $f(v)$, black**

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

$d(v)/f(v)$