Graph $G = (V, E)$

Adjacency Lists:

1: 2, 4
2: 1, 2, 4, 5
3:
4: 1, 2, 5
5: 2, 4, 6
6: 5

Adjacency Matrix

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Directed Graph

Adjacency Lists:

1: 4, 5
2: 1, 4
3: 2, 3
4: 1, 5
5: 4, 3
6: 5, 3

Adjacency Matrix

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Breadth-first Search

r: s, v
s: r, w
t: u, w, x
u: t, y
v: r
w: s, t, x
x: t, w, y
y: u, x

Depth-first Search

1: 3, 5, 6, 9
2: 3, 4, 7
3: 1, 2, 4, 7, 9
4: 2, 3
5: 1, 7
6: 1
7: 2, 3, 5
8:
9: 1, 3

DFS of undirected graph

Every edge in G can be classified as one of the following types of edges in the dfs forest:

tree edges

back edges
(join ancestor - descendant)

NO CROSS EDGES

Depth-first Search of Directed Graph

r: s, u
s: v
t: s, w
u: s
v: u
w:

DFS of directed graph

Every edge in G can be classified as one of the following types of edges in the dfs forest:

tree edges

back edges
(from descendant to ancestor)

forward edges
(from ancestor to descendant)

cross edges
BUT ONLY FROM RIGHT TO LEFT
(later to earlier)

DFS:

```
for each vertex $u \in V[G]$ do
    color[$u$] $\leftarrow$ white
```

```
for each vertex $u \in V[G]$ do
    if color[$u$] $=$ white
        then dfs-visit($u$)
```

```
dfs-visit($u$)
color[$u$] $\leftarrow$ gray

for each vertex $v \in Adj[u]$ do
    if color[$v$] $=$ white

    then
        $p[v] \leftarrow u$
        dfs-visit($v$)

color[$u$] $\leftarrow$ black
```

Modify dfs to:

- label vertices in order first visited (gray)
- label vertices in order dfs finished (black)
- count # of components in undirected graph
- detect cycle in directed graph
- detect cycle in undirected graph
- do topological sort of a DAG
- find strongly connected components in DAG