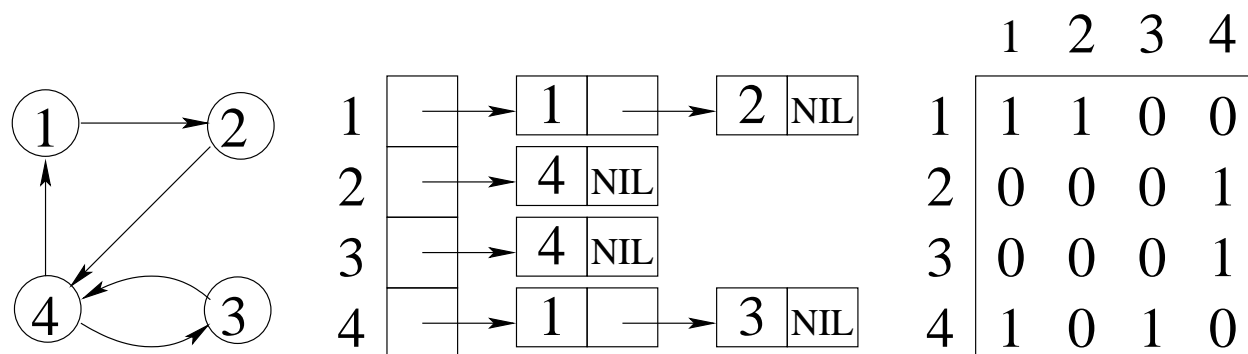
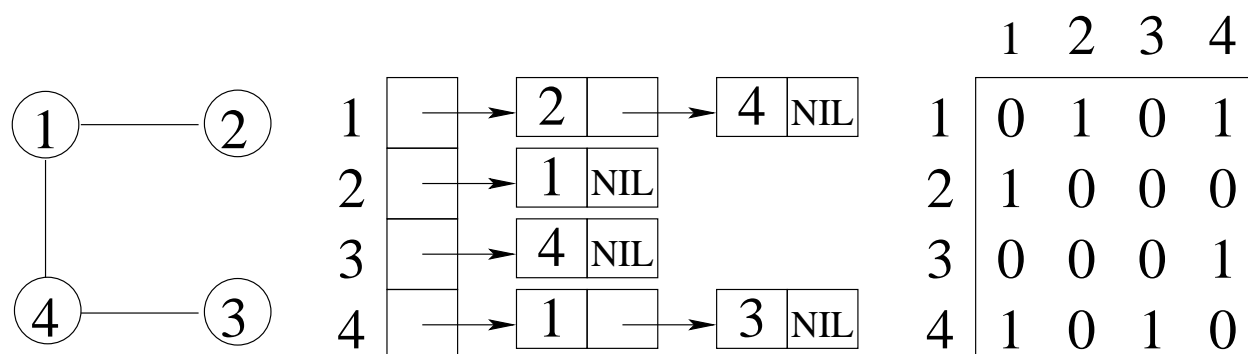


Graph Representations

A graph can be represented as an adjacency list or adjacency matrix.



How efficient are these representations?

Space?

Determining if (u, v) is an edge in the graph?

Visiting each edge once?

Breadth-First Search

BFS(G, s)

for each vertex v in graph G
 do $status[v] \leftarrow$ UNDISCOVERED
 $Q \leftarrow$ NEW-QUEUE(s)
 $status[s] \leftarrow$ DISCOVERED
 $d[s] \leftarrow 0$
 $p[s] \leftarrow$ NIL

while NOT-EMPTY(Q)
 do $u \leftarrow$ DEQUEUE(Q)
 for each vertex v adjacent from u
 do if $status[v] =$ UNDISCOVERED
 then ENQUEUE(Q, v)
 $status[v] \leftarrow$ DISCOVERED
 $d[v] \leftarrow d[u] + 1$
 $p[v] \leftarrow u$
 $status[u] \leftarrow$ FINISHED

BFS is $O(V + E)$.

BFS(G, s) finds shortest paths from s .

Depth-First Search

DFS(G)

```

for each vertex  $v$  in graph  $G$ 
  do  $status[v] \leftarrow$  UNDISCOVERED
   $time \leftarrow 0$ 
  for each vertex  $v$  in graph  $G$ 
    do if  $status[v] =$  UNDISCOVERED
      then  $p[v] \leftarrow$  NIL
       $time \leftarrow$  DFS-VISIT( $v, time + 1$ )

```

DFS-VISIT($u, time$)

```

 $status[u] \leftarrow$  DISCOVERED
 $d[u] \leftarrow time$ 
for each vertex  $v$  adjacent from  $u$ 
  do if  $status[v] =$  UNDISCOVERED
    then  $p[v] \leftarrow u$ 
     $time \leftarrow$  DFS-VISIT( $v, time + 1$ )
 $status[u] \leftarrow$  FINISHED
return  $f[u] \leftarrow time + 1$ 

```

DFS is $\Theta(V + E)$.

$d[v]$ and $f[v]$ form a parenthesis structure.

Topological Sort

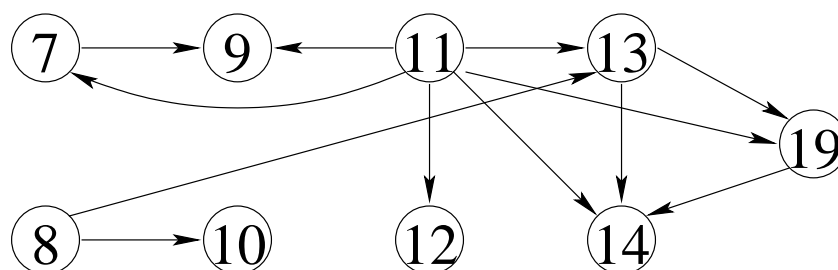
A topological sort of a directed acyclic graph orders the vertices. $u \prec v$ if (u, v) is an edge.

TOPOLOGICAL-SORT(G)

Call **DFS(G)**.

As each vertex is finished,

insert the vertex onto the front a linked list.



TOPOLOGICAL-SORT is $\Theta(V + E)$.

If a path from u to v , then either

Case 1: **DFS-VISIT**(u) before **DFS-VISIT**(v).

$d[u] < d[v] < f[v] < f[u]$ because

DFS-VISIT(u) discovers v .

Case 2: **DFS-VISIT**(v) before **DFS-VISIT**(u).

$d[v] < f[v] < d[u] < f[u]$ because

DFS-VISIT(v) doesn't discover u .

Strongly Connected Components

A strongly connected component of a directed graph is a maximal set of vertices where every vertex can reach the others.

STRONGLY-CONNECTED-COMPONENTS(G)

Call DFS(G) saving finishing times $f(u)$.

Compute G^T , i.e., reverse all the edges.

Call DFS(G^T), loop order is decreasing $f(u)$.

Each DFS(G^T) tree is a SCC.

Let C be a SCC in a graph G . Let u be the first vertex in C discovered by DFS(G).

$v \in C$ implies $f[v] \leq f[u]$.

v can reach u and $v \notin C$ implies $f[u] < f[v]$

implies v can't reach u in G^T

implies DFS(G^T) discovers u in main loop.

u can reach v and $v \notin C$ implies $f[v] < f[u]$

implies u can't reach v in G^T

implies v won't be in u 's tree.