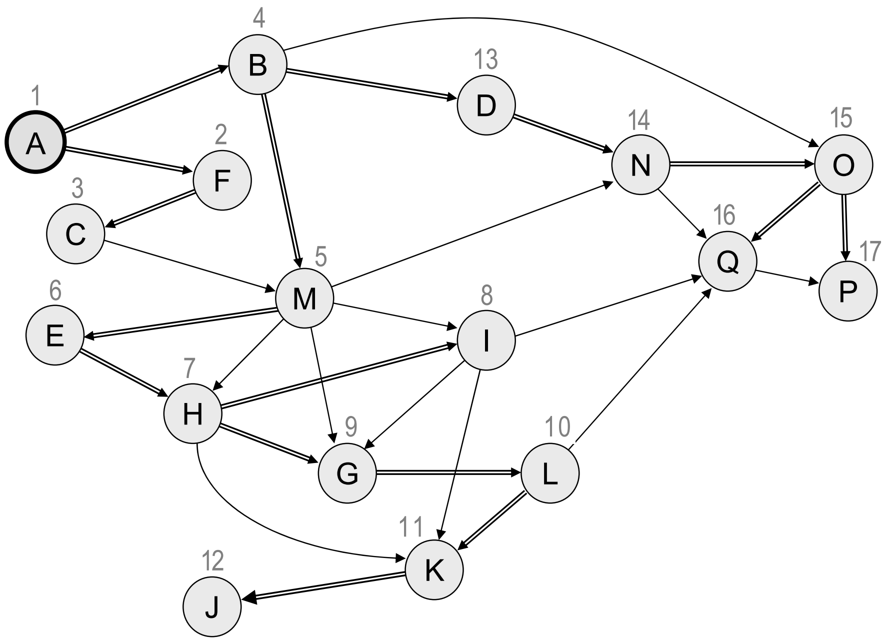
Applications of Depth-First Search: Topological Sort

Label the vertices of an acyclic digraph G by 1, 2, ..., n, so that

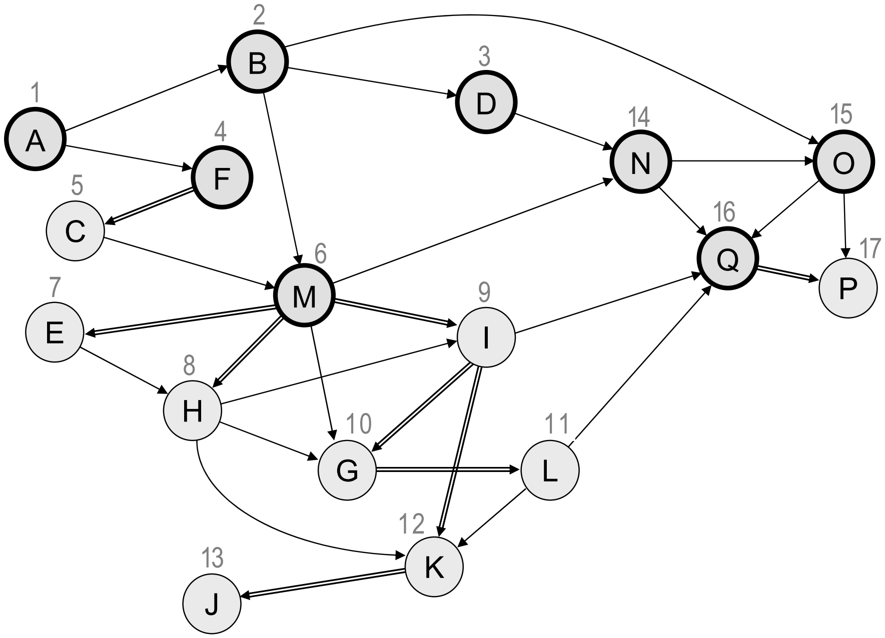
$$vw \text{ is an edge of } G \Rightarrow label[v] < label[w] .$$



Perform a depth-first search of the digraph, with these additions:

Initialization:	$k = n$;
Postorder processing of vertex v:	$label[v] = k$; $--k$;
Back edge processing of edge vw:	detect error (graph is not acyclic);

Another example of Topological Sort
(same digraph, different order to choosing verticies)



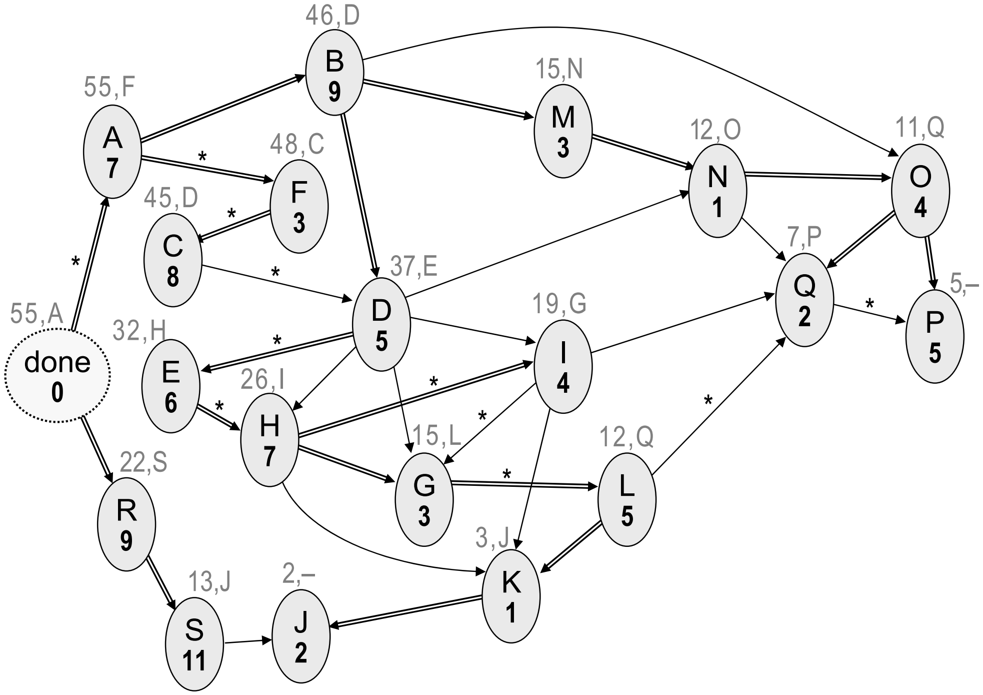
Vertices selected in reverse alphabetical order, when an arbitrary choice must be made. Thick border indicates a starting vertex in depth-first search.

Applications of Depth-First Search: Critical Path

We have a directed acyclic graph, in which each vertex *v* represents a task taking a known amount of time (*duration*[*v*]). An edge from *v* to *w* indicates that task *v* depends on task *w*; that is, *v* cannot start until *w* has finished. (Otherwise, tasks may be performed in parallel.)

Find the earliest possible finish time.

Find a critical path (a sequence of tasks, each dependent on the next, that prevents an earlier finish).



Adjoin a vertex labeled "*done*", with duration 0, and an edge from done to each source in the graph. Then perform a depth-first search, with these additions:

Initialization:	**for** (each vertex *v* of graph *)* *eft*[*v*] = *0*; *critDep*[*v*] = *null*;

Tree edge postorder processing and cross edge processing of edge vw:[1]	**if** (*eft*[*w*] > *eft*[*v*])
	eft[*v*] = *eft*[*w*];
	critDep[*v*] = *w*;

Postorder processing of vertex v:	*eft*[*v*] = *eft*[*v*] + *duration*[*v*];
Back edge processing edge vw:	detect error (graph is not acyclic);

Upon termination, *eft*[*v*] is the earliest finish time of task *v*. In particular, *eft*[*done*] is the earliest finish time for the entire set of tasks.

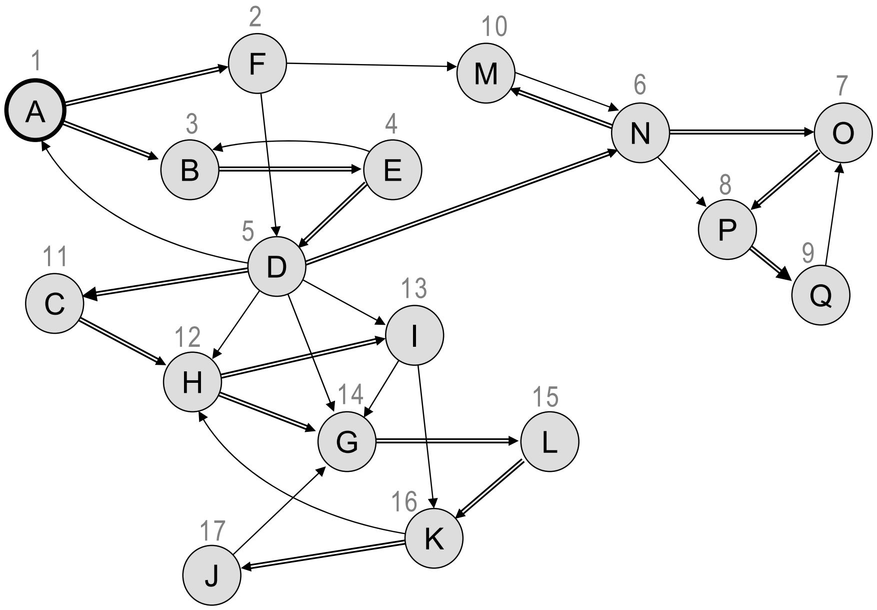
The critical path is $c_0, c_1, \ldots, c_k$, *where*

c_0 = *done,*
c_i = *critDep*[c_{i-1}],
critDep[c_k] = *null.*

[1] No harm in performing these operations for descendent edges as well.

Applications of Depth-First Search: Strongly Connected Components — Phase 1

Perform a depth-first search of the digraph, numbering the vertices as in topological ordering, except that back edges are not treated as an error.

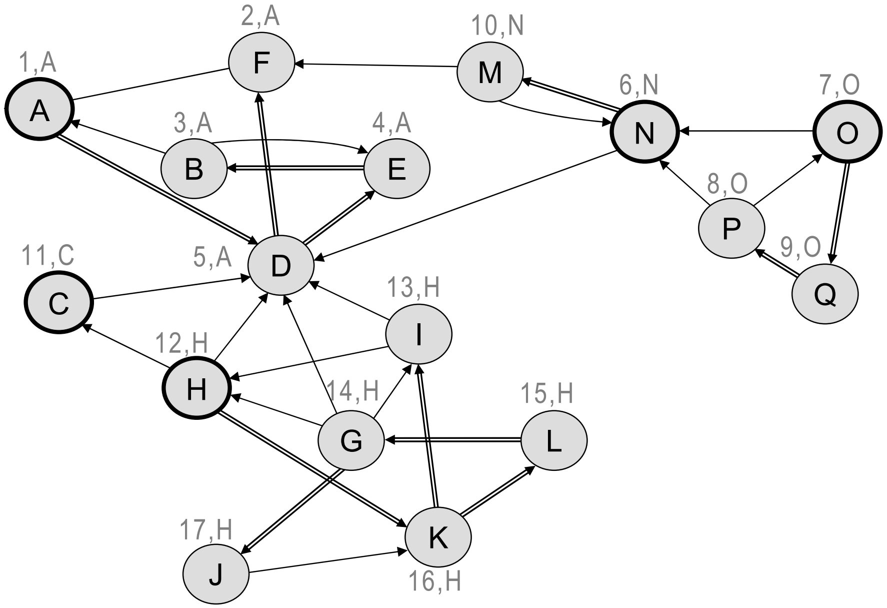


Strongly Connected Components — Phase 2

Form the transpose graph (edge directions reversed), retaining the vertex numbering of phase 1.

Perform a depth-first search of the transpose graph. In the outer loop (for loop), process the vertices in the order of their numbers (assigned in phase 1).

Each white vertex chosen in the outer loop is the leader of a strongly connected component, and the depth-first search from that leader processes the vertices of the component.



2,A
F
1,A
A
10,N
M
6,N
N
7,O
O
3,A
B
4,A
E
8,O
P
9,O
Q
11,C
C
5,A
D
13,H
I
12,H
H
14,H
G
15,H
L
17,H
J
K
16,H