**Precept Topics**

- Graphs and Digraphs
- Graph search: DFS and BFS

**Relevant Material**

- Book chapters: 4.1 and 4.2.

## A. RECAP: Graphs, Digraphs and Graph Search

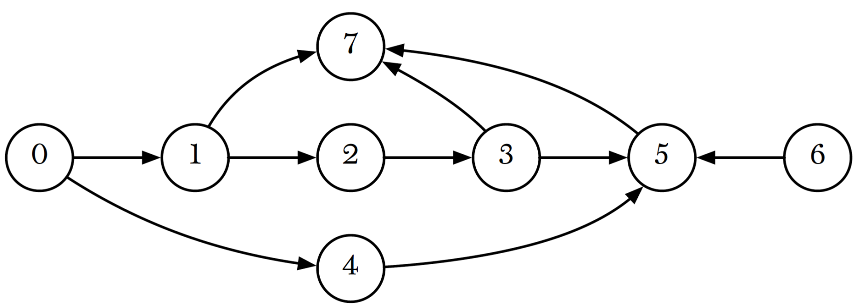
Your preceptor will review the material covered in this week's lectures.

## B. EXERCISE: Shortest Common Ancestor (~40 minutes)

In a directed graph, a vertex $x$ is an ancestor of $v$ if there exists a (directed) path from $v$ to $x$. Given two vertices $v$ and $w$ in a rooted directed acyclic graph (DAG), a shortest common ancestor `sca(v, w)` is a vertex $x$ which:

- is an ancestor to both $v$ and $w$;
- minimizes the **sum of the distances** from $v$ to $x$ and $w$ to $x$ (this path, which goes from $v$ to $x$ and $x$ to $w$ – in reverse direction – is the shortest ancestral path between $v$ and $w$).

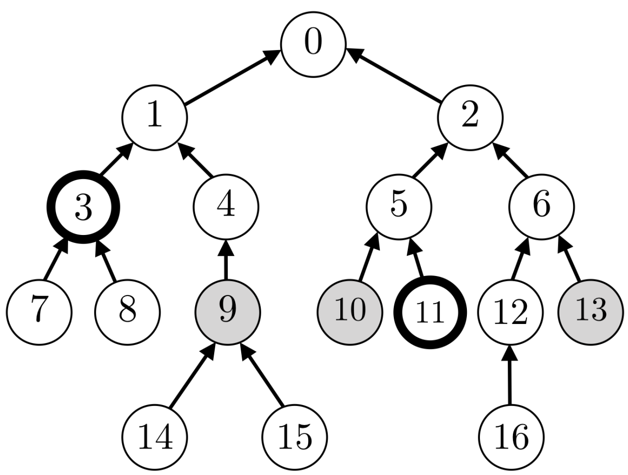
**a)** In the following digraph, compute the (smallest) sum of the path lengths from vertices $1$ and $4$ to all common ancestors. Find the shortest common ancestor and the shortest ancestral path.



**b)** Describe an algorithm for calculating the shortest common ancestor of two vertices $v$ and $w$. Your algorithm should run in linear time (i.e., proportional to $V + E$).

**c)** How would your algorithm differ if we are interested in the shortest ancestral path between two sets of vertices $A$ and $B$, rather than two vertices? (The shortest path between two sets is the shortest among any vertex $v$ in $A$ and any vertex $w$ in $B$.

In the following example, $A = \{3, 11\}$ and $B = \{9, 10, 13\}$. The shortest common ancestor is $5$ (between $10$ and $11$).



**C. EXERCISE: Rooted DAGs** (~15 minutes)

In order to compute shortest common ancestors, we assumed the graph has a particular structure: it is directed and acyclic (i.e., is a DAG) and is also rooted (i.e., there is a unique common ancestor to all nodes). We will see now how to check for both of these things. Once again, our goal is to achieve a $\Theta(V+E)$-time algorithm.

**a)** Describe an algorithm that uses graph search to detect if a directed graph has a cycle.

**Optional:** Code this algorithm in the Ed Lesson.

**b)** Describe an algorithm that detects if a DAG is rooted.

**D. CHALLENGE PROBLEM (optional)**

An Eulerian cycle in a digraph is a cycle that uses each edge exactly once (vertices may repeat). First, design an algorithm that detects if a digraph has an Eulerian cycle; then extend this algorithm to return such a cycle if one exists.