PRINCETON UNIVERSITY

COS 226–Algorithms and Data Structures

Week 7: *Directed Graph and MST Algorithms (Algs. §4.2 and §4.3)*

Version: April 4, 2018

Exercise 1 – Directed Graphs

```
1  private void traversal(Digraph G, int s) {
2      Queue<Integer> q = new Queue<Integer>();
3      for (int v = 0; v < G.V(); v++)
4        distTo[v] = INFINITY;
5      distTo[s] = 0;
6      marked[s] = true;
7      q.enqueue(s);
8      while (!q.isEmpty()) {
9        int v = q.dequeue();
10       for (int w : G.adj(v)) {
11         if (!marked[w]) {
12           edgeTo[w] = v;
13           distTo[w] = distTo[v] + 1; /* line A */
14           marked[w] = true;          /* line B */
15           q.enqueue(w);
16         }
17       }
18     }
19 }
```

A. Briefly describe what the traversal method does and how? Include the purpose of line A.

B. What if the marked array is not updated (i.e line B removed)?

C. What is the order of growth of the method in terms of V and E?

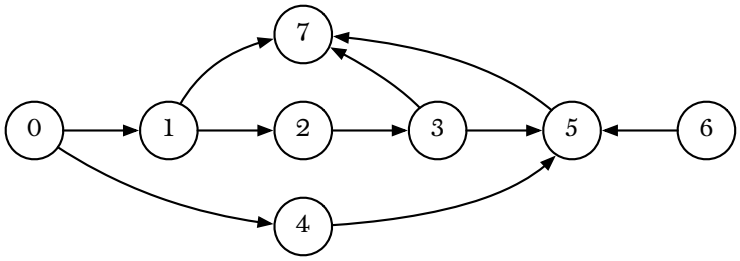
Exercise 2 – Shortest Common Ancestor

The following exercise looks at the algorithm required to compute the shortest common ancestor in a directed acyclic graph, which is an essential component of the WordNet assignment's `ShortestCommonAncestor` class.

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* $\mathrm{sca}(v, w)$ is a vertex x which:

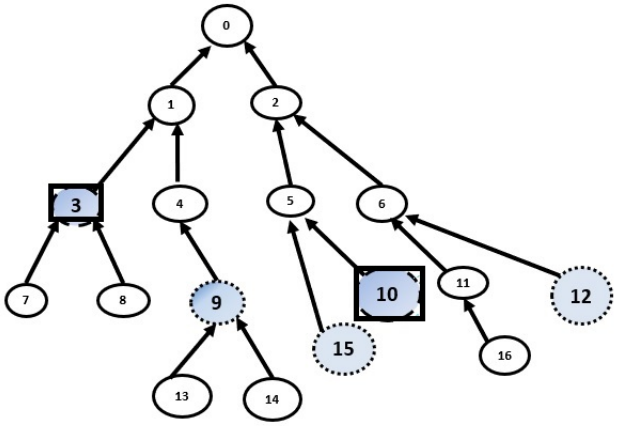
- is an ancestor to both v and w;
- minimizes the sum of the distances from v to x and w to x (this path, which goes from v to x to w, is the *shortest ancestral path* between v and w).

A. In the following digraph, find the shortest common ancestor of vertices 1 and 4, and give the sum of the path lengths from these vertices to all common ancestors, and then circle the shortest.



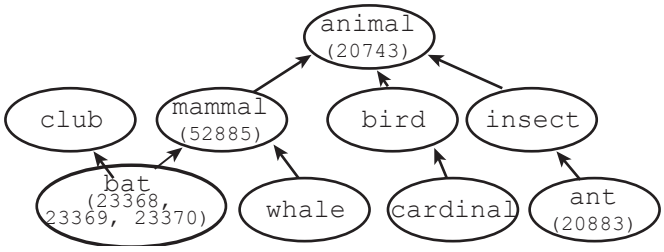
B. Describe an algorithm for calculating the shortest common ancestor of two vertices v and w. Your algorithm should run in linear time (proportional to $V + E$).

C. A shortest ancestral path of two subsets of vertices A and B over all pairs of vertices v and w, with v in A and w in B. How would your algorithm differ if A and B were two sets of vertices rather than single vertices? In the example below $A = 3, 10$ and $B = 9, 12, 15$. Recall that the shortest path from a set of vertices A to another vertex w (not in A) is the minimum of the shortest paths from any of the vertices $v \in A$ to w.



Exercise 3 – Design of WordNet (discussion)

Here is a composite representation of a very small portion of the synsets and hypernyms files:



Some example lines from the synset file:

20743,animal animate_being beast brute creature fauna,a living organism characterized by voluntary movement

23368,bat,a club used for hitting a ball in various games

23369,bat at-bat,(baseball) a turn trying to get a hit; “he was at bat when it happened”; “he got four hits in four at-bats”

23370,bat chiropteran,nocturnal mouselike mammal with forelimbs modified to form membranous wings and anatomical adaptations for echolocation by which they navigate