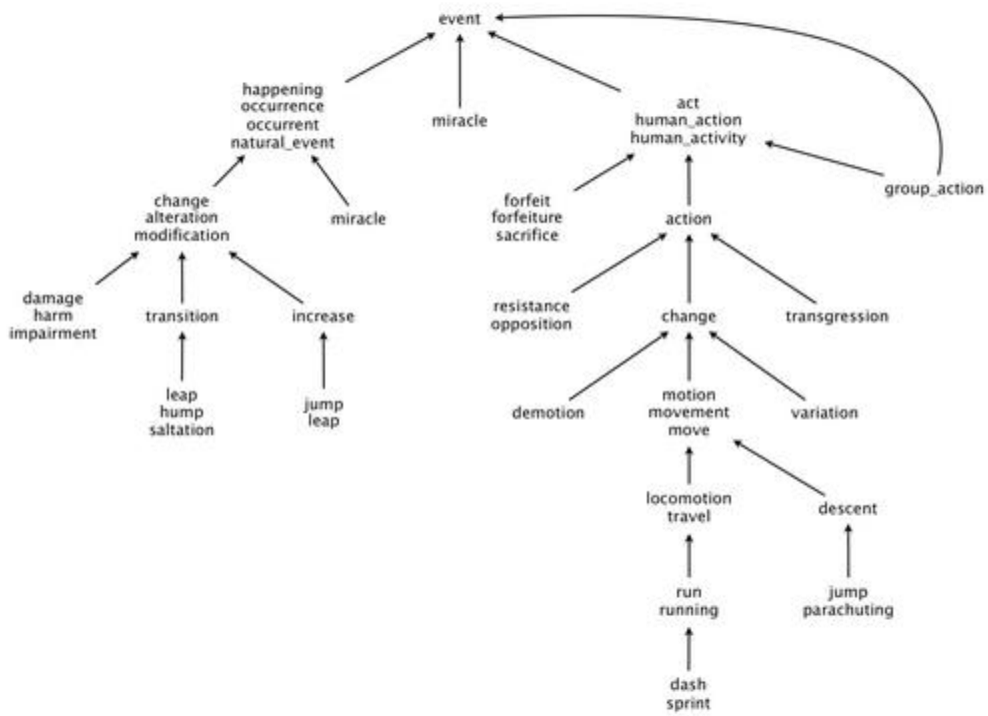
Program 7: CS 2420

Fall 2015 Graphs - Wordnet

History: WordNet

Suppose you had a graph between sets of nouns in which an arc A -> B indicates that the set of nouns A is a more specific case of the set of nouns B. Such a graph would allow us to reason about similarities and differences.   The graph below is an example of what this might look like.



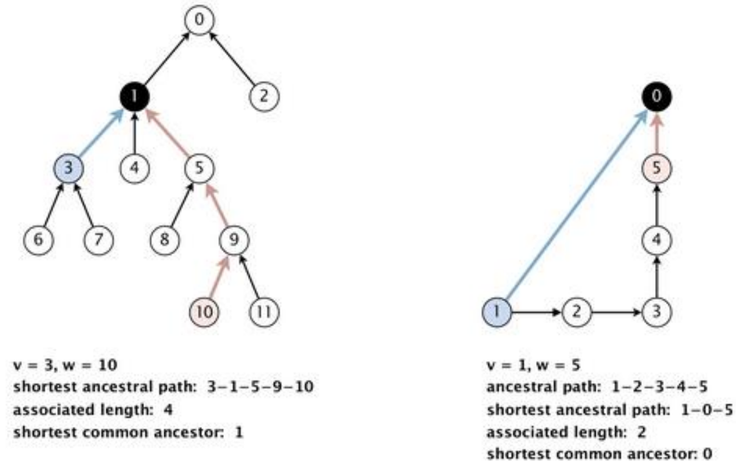
Measuring the semantic relatedness of two nouns. Semantic relatedness refers to the degree to which two concepts are related. Measuring semantic relatedness is a challenging problem. For example, you consider *George W. Bush* and *John F. Kennedy* (two U.S. presidents) to be more closely related than *George W. Bush* and *chimpanzee* (two primates). It might not be clear whether *George W. Bush* and *Eric Arthur Blair* are more related than two arbitrary people. However, both *George W. Bush* and *Eric Arthur Blair* (aka George Orwell) are famous communicators and, therefore, closely related.

WordNet is a semantic lexicon for the English language that computational linguists and cognitive scientists use extensively. For example, WordNet was a key component in IBM's Jeopardy-playing Watson computer system.

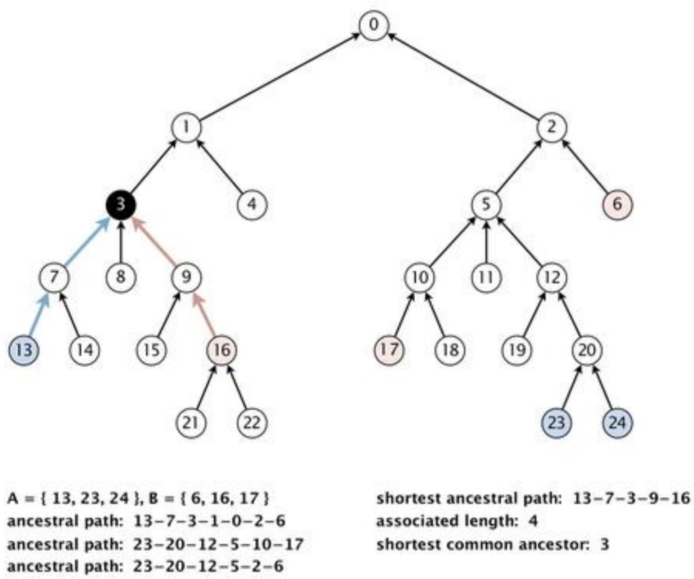
Your Assignment:

Given a directed acyclic graph, find the shortest path between any two nodes, the length of the path, and the shortest common ancestor.

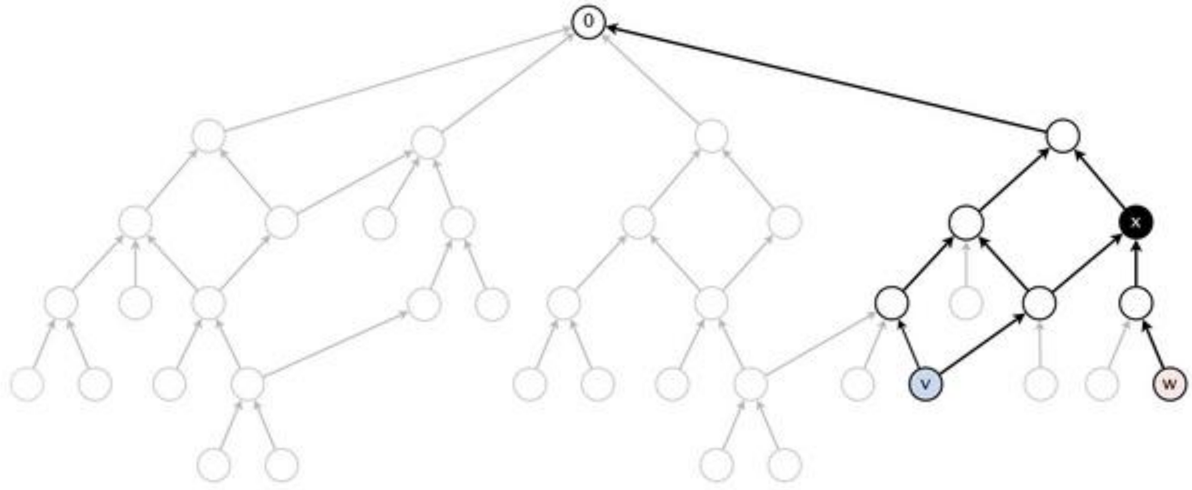
Shortest common ancestor. An *ancestral path* between two vertices (*v* and *w)* in a rooted DAG is a directed path from *v* to a common ancestor *x*, together with a directed path from *w* to the same ancestor *x*. A *shortest ancestral path* is an ancestral path of minimum total length. We refer to the common ancestor in a shortest ancestral path as a *shortest* *common ancestor*. Note that a shortest common ancestor always exists because the root is an ancestor of every vertex. Note also that an ancestral path is a path, but not a directed path.



We generalize the notion of shortest common ancestor to *subsets* of vertices. A shortest ancestral path of two subsets of vertices *A* and *B* is a shortest ancestral path over all pairs of vertices *v* and *w*, with *v* in *A* and *w* in *B*.



Additional performance requirements. For full credit, the methods length() and ancestor() should take time proportional to the number of vertices and edges reachable from the argument vertices (or better). For example, to compute the shortest common ancestor of v and w in the digraph below, your algorithm should examine only the highlighted vertices and edges.



Hint:

Every ancestor of a node v (in the above example) needs to know (1) the length of the shortest path to it from v (2) the predecessor responsible for that shortest path. If you do a breadth first traversal from v and recorded the best length, you wouldn't have to keep changing the length of the best path. Storing the predecessor help you it recreating the best path.

BONUS feature: Outcast detection. Given a list of nouns $x_1, x_2, ..., x_n$, which noun is the least related to the others? To identify *an outcast*, compute the sum of the lengths between each noun and every other one:

$d_i = length(x_i, x_1) + length(x_i, x_2) + ... + length\ x_i, x_n)$

and return a noun x_t for which d_t is maximum. Note that because *length* $(x_i, x_i) = 0$, it will not contribute to the sum.

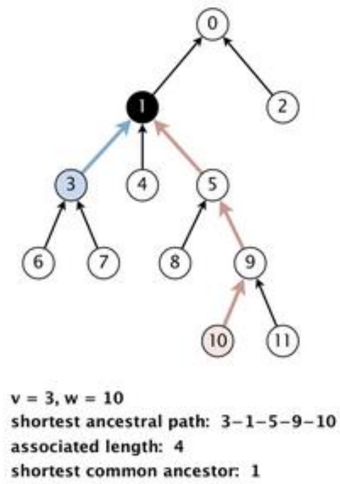
Input:
The input will be the number of nodes, number of edges, and each edge (A-> B is represented by A B)

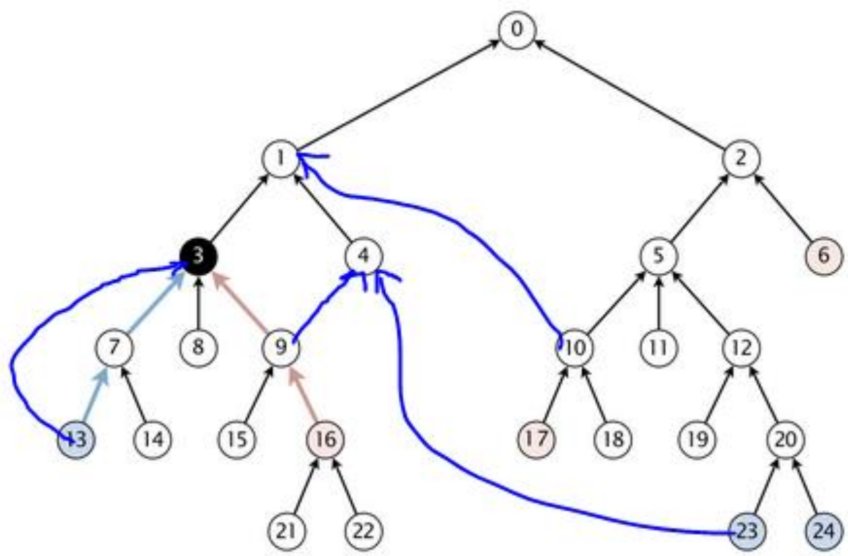
Testing:
a. Allow the user to select which file he/she will use for input:digraph1.txt, digraph2,txt, or digraph3.txt. The first two files are provided. digraph3.txt is one you make up.
b. All sets of nodes will be entered by a list of node numbers followed by a negative number.
c. Allow the user to select from the following commands:

i. (10 points) Given two nodes in the graph, find the shortest common ancestor, shortest ancestral path, and associated length
ii. (10 points) Given two subsets of nodes, find the shortest common ancestor, shortest ancestral path, and associated length
iii. (5 points) BONUS: Given a set of nodes find the outcast

Here are the two provided input files and their associated graphs.

```
12
11
 6  3
 7  3
 3  1
 4  1
 5  1
 8  5
 9  5
10  9
11  9
 1  0
 2  0
```



<table>
<tr>
<td>25
28
1 0
2 0
3 1
4 1
5 2
6 2
7 3
8 3
9 3
10 6
11 5
12 6
13 7
14 7
15 9
16 9
17 10
18 10
19 12
20 12
21 16
22 16
23 20
24 20
13 3
9 4
10 1
23 4</td>
<td></td>
</tr>
</table>

ADDITIONAL BONUS (10 points) If you have time, try doing this for actual noun sets. See the instructions on the assignment page.