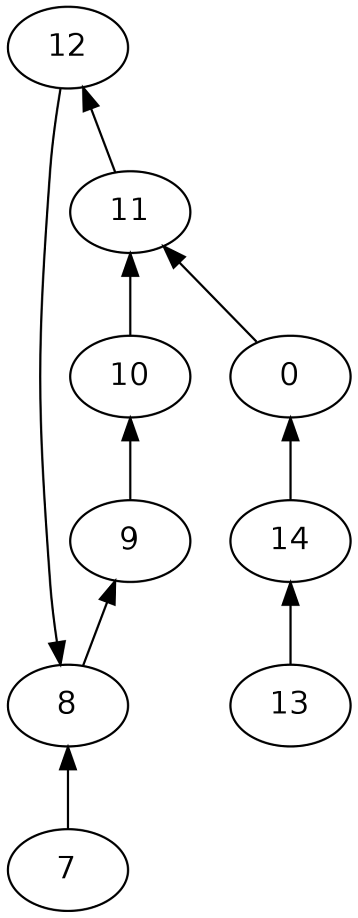
COS226 Week 7 Activity

1. Wordnet.

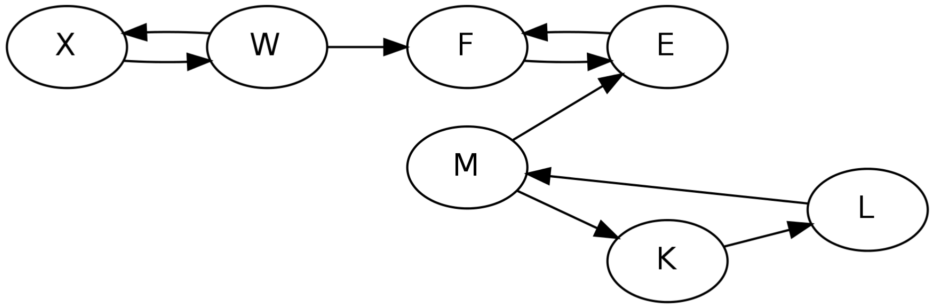
   (a) Given the digraph below, compute SAP.ancestor(0, 7) and SAP.length(0, 7).

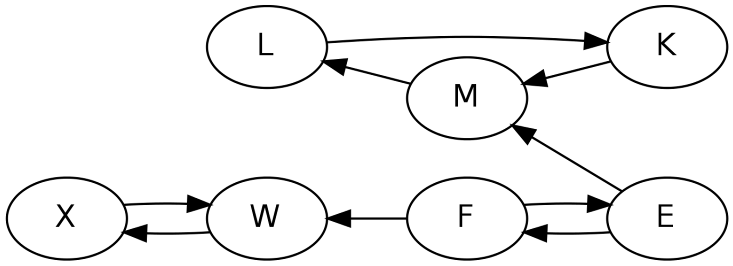


   (b) Describe an algorithm for calculating SAP.ancestor(int v, int w). Your algorithm should work even if the graph contains cycles.

   (c) How would your algorithm differ if you had Iterables for v and w?

2. For this problem, use the letter-based graphs below. Assume that adjacency lists appear in alphabetical order. Assume that vertices are processed in alphabetical order. The top graph is G, and the bottom graph is G^R.





(a) Find the reverse postorder of G^R.

(b) Using DFS, G, and your answer to (a), find the strongly connected connected components of G. Do this by writing the id[] of each vertex next to the vertex. Assume the first SCC is given id 0, the second SCC is given id 1, etc.

(c) For (b), if we used the postorder of G instead of the reverse postorder of G^R, would we get the correct SCCs? To save time, have a free hint: The postorder of G is FEMLKWX.

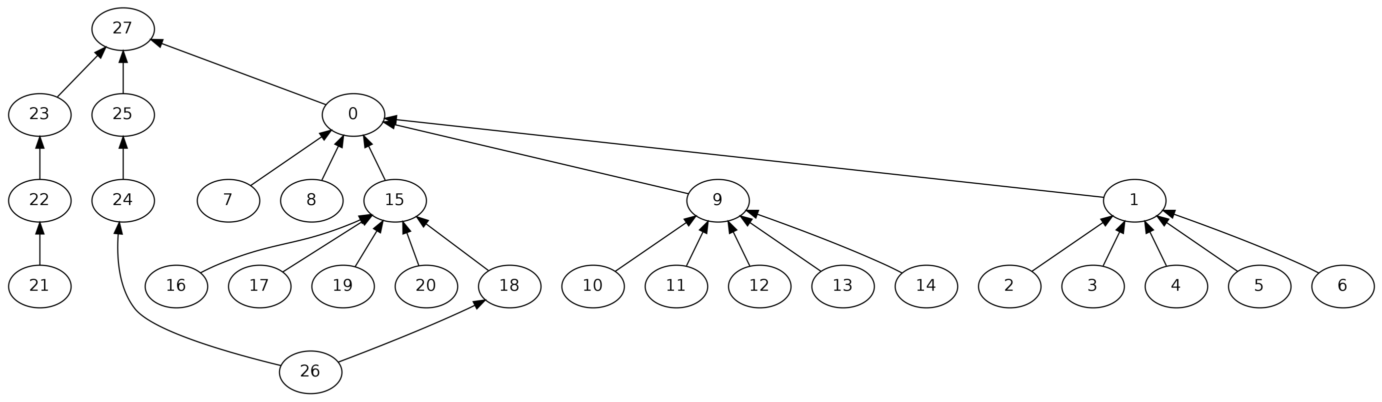
(d) Extra challenging: Add two nodes and three edges to the graph such that the postorder of G does not provide a valid set of SCCs. Hint: We need a new SCC that points into an existing SCC, but which comes first in the postorder of G.

```
0,juvenile juvenile_person
1,young_person youth younker
  spring_chicken
2,slip
3,schoolchild school-age_child pupil
4,puppy pup
5,hobbledehoy
6,blade
7,preteen preteenager
8,ingenue
9,child kid youngster minor shaver
  nipper small_fry tiddler tike tyke
  fry nestling
10,waif street_child
11,urchin
12,toddler yearling tot bambino
13,sprog
14,silly
15,adolescent stripling teenager teen
16,young_buck young_man
17,rocker
18,punk_rocker punk
19,pachuco
20,mod
21,punk_rock punk
22, rock_'n'_roll rock'n'roll
   rock-and-roll rock_and_roll rock
   rock_music
23,popular_music popular_music_genre
24,urchin
25,echinoderm
26,Frankie_the_urchin
27,entity
```

Synsets.txt

```
1,0
2,1
3,1
4,1
5,1
6,1
7,0
8,0
9,0
10,9
11,9
12,9
13,9
14,9
15,0
16,15
17,15
18,15
19,15
20,15
21,22
22,23
24,25
26,24,18
0,27
23,27
25,27
```

Hypernyms.txt



```
Wordnet w.sap("punk", "waif")
  returns "juvenile juvenile_person"

SAP s.ancestor({18, 21}, {10})
  returns 0
```

SAP Class API:

```
// constructor takes a digraph (not necessarily a DAG)
public SAP(Digraph G)

// length of shortest ancestral path between v and w; -1 if no such path
public int length(int v, int w)

// a common ancestor of v and w that participates in a shortest ancestral path; -1 if no such path
public int ancestor(int v, int w)

// length of shortest ancestral path between any vertex in v and any vertex in w; -1 if no such path
public int length(Iterable<Integer> v, Iterable<Integer> w)

// a common ancestor that participates in shortest ancestral path; -1 if no such path
public int ancestor(Iterable<Integer> v, Iterable<Integer> w)

// do unit testing of this class
public static void main(String[] args)
```

```
// inputs are file names: reads files, builds graph and other data structures, tests graph is a rooted DAG
public WordNet(String synsets, String hypernyms)

// returns all WordNet nouns
public Iterable<String> nouns()

// is the word a WordNet noun?
public boolean isNoun(String word)

// distance between nounA and nounB (defined below)
public int distance(String nounA, String nounB)

// a synset (second field of synsets.txt) that is the common ancestor of nounA and nounB
// in a shortest ancestral path (defined below)
public String sap(String nounA, String nounB)

// do unit testing of this class
public static void main(String[] args)
```