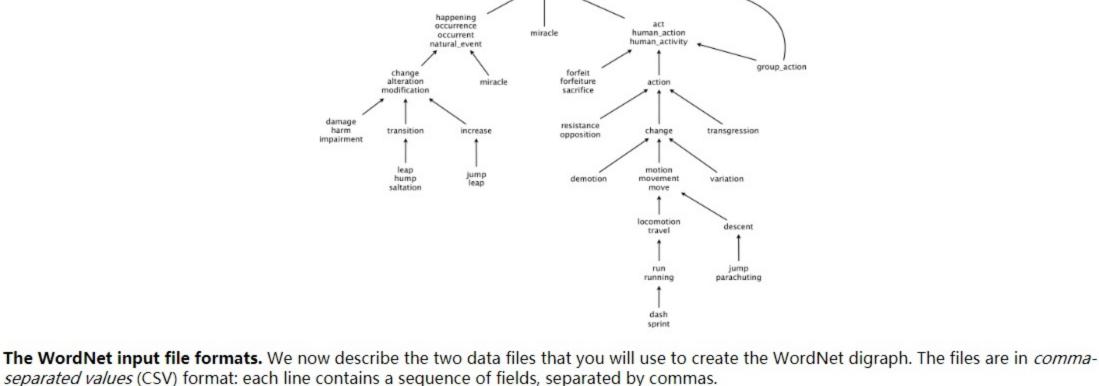
Submit

is a kind of logic gate. The WordNet digraph. Your first task is to build the WordNet digraph: each vertex  $\nu$  is an integer that represents a synset, and each directed edge  $v \rightarrow w$  represents that w is a hypernym of v. The WordNet digraph is a rooted DAG: it is acyclic and has one vertex—the root—that is an ancestor of every other vertex. However, it is not necessarily a tree because a synset can have more than one hypernym. A small subgraph of the WordNet digraph appears below.

describes semantic relationships between synsets. One such relationship is the is-a relationship, which connects a hyponym (more specific synset) to a hypernym (more general synset). For example, the synset { gate, logic gate } is a hypernym of { AND circuit, AND gate } because an AND gate

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. WordNet groups words into sets of synonyms called synsets. For example, { AND circuit, AND gate } is a synset that represent a logical gate that fires only when all of its inputs fire. WordNet also



 List of synsets. The file synsets.txt contains all noun synsets in WordNet, one per line. Line i of the file (counting from 0) contains the information for synset i. The first field is the synset id, which is always the integer i, the second field is the synonym set (or synset); and the

third field is its dictionary definition (or *gloss*), which is not relevant to this assignment. % more synsets.txt synset

35,ALGOL,a programming language used to express computer programs as algorithms

34 (AIDS acquired\_immune\_deficiency\_syndrome), a serious (often fatal) disease of the immune system

```
(36)AND_circuit AND_gate, a circuit in a computer that fires only when all of its inputs fire
                                            37,APC,a drug combination found in some over-the-counter headache remedies
                                           38,ASCII_character,any member of the standard code for representing characters by binary numbers
                                            39,ASCII_character_set,(computer science) 128 characters that make up the ASCII coding scheme
                                            40,ASCII_text_file,a text file that contains only ASCII characters without special formatting
                                            41,ASL American_sign_language,the sign language used in the United States
                                            42,AWOL one who is away or absent without leave
  For example, line 36 means that the synset {AND_circuit, AND_gate } has an id number of 36 and its gloss is a circuit in a computer that fires only
  when all of its inputs fire. The individual nouns that constitute a synset are separated by spaces. If a noun contains more than one word, the
  underscore character connects the words (and not the space character).
• List of hypernyms. The file hypernyms.txt contains the hypernym relationships. Line i of the file (counting from 0) contains the hypernyms of
```

synset i. The first field is the synset id, which is always the integer i, subsequent fields are the id numbers of the synset's hypernyms. % more hypernyms.txt

34 47569,48084 35,19983

hypernyms

(36)42338

```
37,53717
                                                                      38,28591
                                                                      39,28597
                                                                      40,76057
                                                                      41,70206
                                                                      42,18793
     For example, line 36 means that synset 36 (AND_circuit AND_Gate) has 42338 (gate logic_gate) as its only hypernym. Line 34 means that synset 34
     (AIDS acquired_immune_deficiency_syndrome) has two hypernyms: 47569 (immunodeficiency) and 48084 (infectious_disease).
WordNet data type. Implement an immutable data type WordNet with the following API:
```

public class WordNet {

// returns all WordNet nouns

// constructor takes the name of the two input files public WordNet(String synsets, String hypernyms)

```
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)
      }
Corner cases. Throw an IllegalArgumentException in the following situations:

    Any argument to the constructor or an instance method is null

    The input to the constructor does not correspond to a rooted DAG.

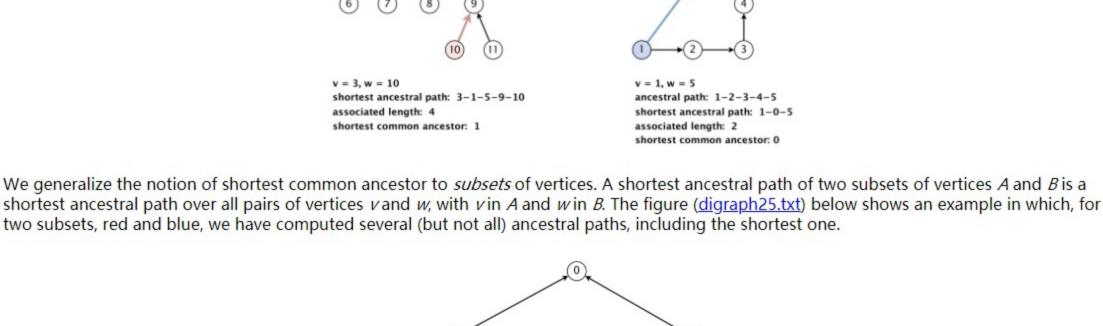
    Any of the noun arguments in distance() or sap() is not a WordNet noun.
```

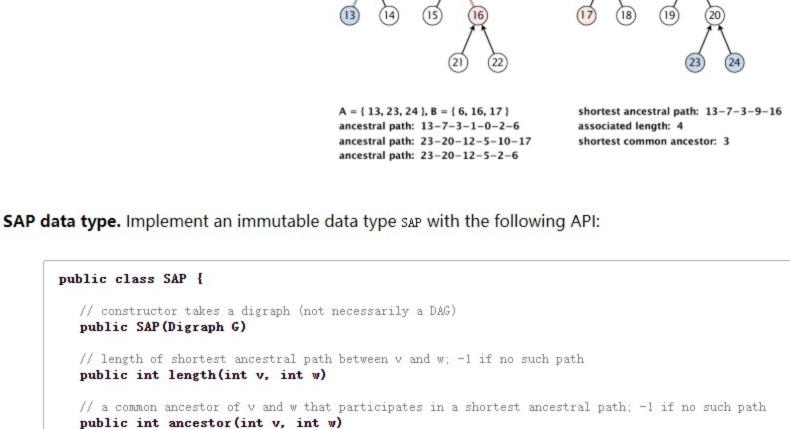
You may assume that the input files are in the specified format. Performance requirements. Your data type should use space linear in the input size (size of synsets and hypernyms files). The constructor should

- take time linearithmic (or better) in the input size. The method is Noun() should run in time logarithmic (or better) in the number of nouns. The
- methods distance() and sap() should run in time linear in the size of the WordNet digraph. For the analysis, assume that the number of nouns per synset is bounded by a constant.

**Shortest ancestral path.** An ancestral path between two vertices  $\nu$  and  $\omega$  in a digraph is a directed path from  $\nu$  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 also that an ancestral path is a path, but not a directed path.





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

public int length(Iterable(Integer) v, Iterable(Integer) w)

public int ancestor(Iterable(Integer) v, Iterable(Integer) w)

public class SAP {

public SAP (Digraph G)

SAP sap = new SAP(G); while (!StdIn.isEmpty()) {

% cat digraph1.txt

}

11

7 8

5 1 9 5

int v = StdIn.readInt(); int w = StdIn.readInt();

int length = sap.length(v, w); int ancestor = sap.ancestor(v, w);

StdOut.printf("length = %d, ancestor = %d\n", length, ancestor);

```
Corner cases. Throw an IllegalArgumentException in the following situations:

    Any argument is null

    Any vertex argument is outside its prescribed range

    Any iterable argument contains a null item

Performance requirements. All methods (and the constructor) should take time at most proportional to E + V in the worst case, where E and V are
the number of edges and vertices in the digraph, respectively. Your data type should use space proportional to E + V.
Test client. The following test client takes the name of a digraph input file as as a command-line argument, constructs the digraph, reads in vertex
pairs from standard input, and prints out the length of the shortest ancestral path between the two vertices and a common ancestor that
participates in that path:
       public static void main(String[] args) {
          In in = new In(args[0]);
           Digraph G = new Digraph(in);
```

% java-algs4 SAP digraph1.txt

This is the notion of distance that you will use to implement the distance() and sap() methods in the WordNet data type.

distance(noun1, noun2) = 4

sca(noun1, noun2) = {noun3, noun6}

length = 4, ancestor = 1

length = 3, ancestor = 5

length = 4, ancestor = 0

length = -1, ancestor = -1

// length of shortest ancestral path between any vertex in v and any vertex in w: -1 if no such path

// a common ancestor that participates in shortest ancestral path; -1 if no such path

Here is a sample execution:

```
1 0
       2 0
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 (a.k.a. George Orwell) are famous communicators and,
therefore, closely related.
We define the semantic relatedness of two WordNet nouns x and y as follows:

 A = set of synsets in which x appears

 B = set of synsets in which y appears

    distance(x, y) = length of shortest ancestral path of subsets A and B
```

sca(x, y) = a shortest common ancestor of subsets A and B

3 11

9 12

7 2

noun2 noun2 noun5 noun1

noun2

```
Outcast detection. Given a list of WordNet nouns x_1, x_2, ..., x_p, which noun is the least related to the others? To identify an outcast, compute the
sum of the distances between each noun and every other one:
     d_i = distance(x_i, x_1) + distance(x_i, x_2) + ... + distance(x_i, x_n)
and return a noun x_t for which d_t is maximum. Note that distance(x_i, x_i) = 0, so it will not contribute to the sum.
Implement an immutable data type Outcast with the following API:
          public Outcast (WordNet wordnet)
                                                   // constructor takes a WordNet object
          public String outcast (String[] nouns) // given an array of WordNet nouns, return an outcast
          public static void main(String[] args) // see test client below
Assume that argument to outcast() contains only valid wordnet nouns (and that it contains at least two such nouns).
```

The following test client takes from the command line the name of a synset file, the name of a hypernym file, followed by the names of outcast

WordNet wordnet = new WordNet(args[0], args[1]); Outcast outcast = new Outcast(wordnet); for (int t = 2; t < args.length; t++) { In in = new In(args[t]); String[] nouns = in.readAllStrings();

```
% cat outcast5.txt
horse zebra cat bear table
% cat outcast8.txt
water soda bed orange_juice milk apple_juice tea coffee
% cat outcast11.txt
apple pear peach banana lime lemon blueberry strawberry mango watermelon potato
% java-algs4 Outcast synsets.txt hypernyms.txt outcast5.txt outcast8.txt outcast11.txt
outcast5.txt: table
outcast8.txt: bed
outcast11.txt: potato
```

StdOut.println(args[t] + ": " + outcast.outcast(nouns));

Analysis of running time (optional). Analyze the effectiveness of your approach to this problem by giving estimates of its time requirements.

- and the number of edges *E* in the digraph. Give the order of growth of the best-case running time of the same methods.
- Web submission. Submit a .zip file containing WordNet. java, SAP. java, Outcast. java, any other supporting files (excluding algs4. jar). You may not call any library functions except those in java. lang, java. util, and algs4. jar.

public class Outcast {

files, and prints out an outcast in each file:

public static void main(String[] args) {

}

}

Copyright © 2006.

Here is a sample execution:

This assignment was developed by Alina Ene and Kevin Wayne.