

COS 226 Programming Assignment Checklist: WordNet

Frequently Asked Questions

How do I read input directly from a file, without redirecting standard input? Use the [In data type](#). (See pp. 82-83 of the textbook for more details.)

Can I read the synset or hypernym file twice? No, file I/O is very expensive so please read each file only once and store it in an appropriate data structure.

Any advice on how to read in and parse the synset and hypernym data files? Use the `readLine()` method in our `In` library to read in the data one line at a time. Use the `split()` method in Java's `String` library to divide a line into fields. You can find an example using `split()` in [Domain.java](#). Use `Integer.parseInt()` to convert string id numbers into integers.

What assumption can I make about the digraph `G` passed to the `SAP` constructor? It can be any digraph, not necessarily a DAG.

What data structure(s) should I use to store the synsets, synset ids, and hypernyms? This part of the assignment is up to you. You must carefully select data structures to achieve the specified performance requirements.

Do I need to store the glosses? No, you won't use them on this assignment.

Can I use my own Digraph class? No, it must have the same API as our [Digraph.java](#) class; otherwise, you are changing the API to the `SAP` constructor (which takes a `Digraph` argument). Do not submit `Digraph.java`.

Should I re-implement breadth-first search in my `SAP` class? No, you should call the relevant method(s) in [BreadthFirstDirectedPaths.java](#). You may modify [BreadthFirstDirectedPaths.java](#) to optimize your code, but if you do so, rename it, say to `DeluxeBFS.java`, and submit it.

I understand how to compute the `length(int v, int w)` method in time proportional to $E + V$ but my `length(Iterable<Integer> v, Iterable<Integer> w)` method takes time proportional to $a \times b \times (E + V)$, where a and b are the sizes of the two iterables. How can I improve it to be proportional to $E + V$? The key is using the constructor in `BreadthFirstDirectedPaths` that takes an iterable of sources instead of using a single source.

Is a vertex considered an ancestor of itself? Yes.

What is the root synset for the WordNet DAG?

```
40019,entity,that which is perceived or known or inferred to have its own distinct existence (living or nonliving)
```

Can a noun appear in more than one synset? Absolutely. It will appear once for each meaning that the noun has. For example, here are all of the entries in `synsets.txt` that include the noun `word`.

```
37559,discussion give-and-take word,an exchange of views on some topic; "we had a good discussion"; "we had a word or two about it"
50266,news intelligence tidings word,new information about specific and timely events; "they awaited news of the outcome"
60429,parole word word_of_honor,a promise; "he gave his word"
60430,password watchword word parole countersign,a secret word or phrase known only to a restricted group; "he forgot the password"
80883,word,a unit of language that native speakers can identify; "words are the blocks from which sentences are made"; "he hardly said"
80884,word,a brief statement; "he didn't say a word about it"
80885,word,a verbal command for action; "when I give the word charge!"
80886,word,a word is a string of bits stored in computer memory; "large computers use words up to 64 bits long"
```

Can a synset consist of exactly one noun? Yes. Moreover, there can be several different synsets that consist of the same noun.

```
71,Aberdeen,a town in western Washington
72,Aberdeen,a town in northeastern South Dakota
73,Aberdeen,a town in northeastern Maryland
74,Aberdeen,a city in northeastern Scotland on the North Sea
```

I'm an ontologist and I noticed that your `hypernyms.txt` file contains both *is-a* and *is-instance-of* relationships. Yes, you caught us. This ensures that every noun (except `entity`) has a hypernym. Here is an article on the [subtle distinction](#).

How can I make `SAP` immutable? You can (and should) save the associated digraph in an instance variable. However, because our `Digraph` data type is mutable, you must first make a defensive copy by calling the copy constructor.

Input, Output, and Testing

Some examples. Here are some interesting examples that you can use in your code.

- The synset `municipality` has two paths to `region`.

```
municipality -> administrative_district -> district -> region
municipality -> populated_area -> geographic_area -> region
```

- The synsets `individual` and `edible_fruit` have several different paths to their common ancestor `physical_entity`.

```
individual -> organism -> living_thing -> object
individual -> causal_agency -> physical_entity
edible_fruit -> garden_truck -> food -> solid -> physical_entity
edible_fruit -> fruit -> reproductive_structure -> plant_organ -> plant_part ->
    natural_object -> unit -> object -> physical_entity
```

- The following pairs of nouns are very far apart:

```
23 white_marlin, mileage
32 Black_Plague, black_marlin
32 American_water_spaniel, histology
32 Brown_Swiss, barrel_roll
```

- The following synset has many ancestors and paths to `entity`.

Ambrose Saint_Ambrose St._Ambrose

Possible progress steps

- The directory [wordnet](#) contains some sample input files. For convenience, the zip file [wordnet.zip](#) contains all of these files bundled together.
- Create the data type `SAP`. First, think carefully about designing a correct and efficient algorithm for computing the shortest ancestral path. Ask in the Discussion Forums if you're unsure. In addition to the `digraph*.txt` files, design small DAGs to test and debug your code.
- Read in and parse the files described in the assignment, `synsets.txt` and `hypernyms.txt`. Don't worry about storing the data in any data structures yet. Test that you are parsing the input correctly before proceeding.
- Create a data type `wordNet`. Divide the constructor into two subtasks.
 - Read in the `synsets.txt` file and build appropriate data structures.
 - Read in the `hypernyms.txt` file and build a `Digraph`.

If you read in `synsets.txt` first, you can identify the largest id before constructing the `Digraph`. Check that it is 82,191 but do not hardwire this number into your program because your program must work with any valid input file.

Optional Optimizations

There are a few things you can do to *significantly* speed up a sequence of SAP computations on the same digraph.

- The bottleneck operation is re-initializing arrays of length V to perform the BFS computations. This must be done once for the first BFS computation, but if you keep track of which array entries change, you can reuse the same array from computation to computation (re-initializing only those entries that changed in the previous computation). This leads to a dramatic savings when only a small number of entries change (which is the typical case for the wordnet digraph). Note that if you have any other loops that iterates through all of the vertices, then you must eliminate those loops too in order to achieve a sublinear running time.
- If you run the breadth-first searches from v and w simultaneously, then you can terminate the BFS from v (or w) as soon as the distance exceeds the length of the best ancestral path found so far.
- Implement a software cache of recently computed `length()` and `ancestor()` queries. Often, a client calls `ancestor()` immediately after calling `length()` or vice versa.

Enrichment

- This [applet](#) connects words by a chain of WordNet synonyms.
- This [paper](#) measures the semantic orientation of WordNet adjectives by computing their relative distance to "good" and "bad."