WordNet Noun Semantics

[Sample Input/Output 2](#_Toc467211995)

[Problem Definition 2](#_Toc467211996)

[What’s WordNet? 2](#_Toc467211997)

[Required Semantics 5](#_Toc467211998)

[Descriptions and Suggestions 5](#_Toc467211999)

[Shortest Common Ancestor 5](#_Toc467212000)

[Semantic Relatedness (for Question1) 6](#_Toc467212001)

[Outcast Detection (for Question2) 7](#_Toc467212002)

[Project Requirements 8](#_Toc467212003)

[Required Implementation 8](#_Toc467212004)

[Input 8](#_Toc467212005)

[Output 8](#_Toc467212006)

[Test Cases 8](#_Toc467212007)

[Deliverables 9](#_Toc467212008)

[Implementation (60%) 9](#_Toc467212009)

[Document (40%) 9](#_Toc467212010)

[Milestones 9](#_Toc467212011)

[BONUSES 10](#_Toc467212012)

[FIRST: More Efficient Calculation of SCA between Two Synsets IDs 10](#_Toc467212013)

[Level 2: Efficient Calculation of Distance and SCA between Two Nouns 10](#_Toc467212014)

# Sample Input/Output

1. What’s the **semantic relation** between the following pair of words?
2. white\_marlin **vs.** mileage 🡺 23 (too far apart)
3. jump **vs.** run 🡺 4 (too close)
4. What’s the **odd word** in the following list?
5. horse zebra cat bear table 🡺 table
6. water soda bed orange\_juice milk apple\_juice tea coffee 🡺 bed

# Problem Definition

## What’s WordNet?

### Terminologies & Usage

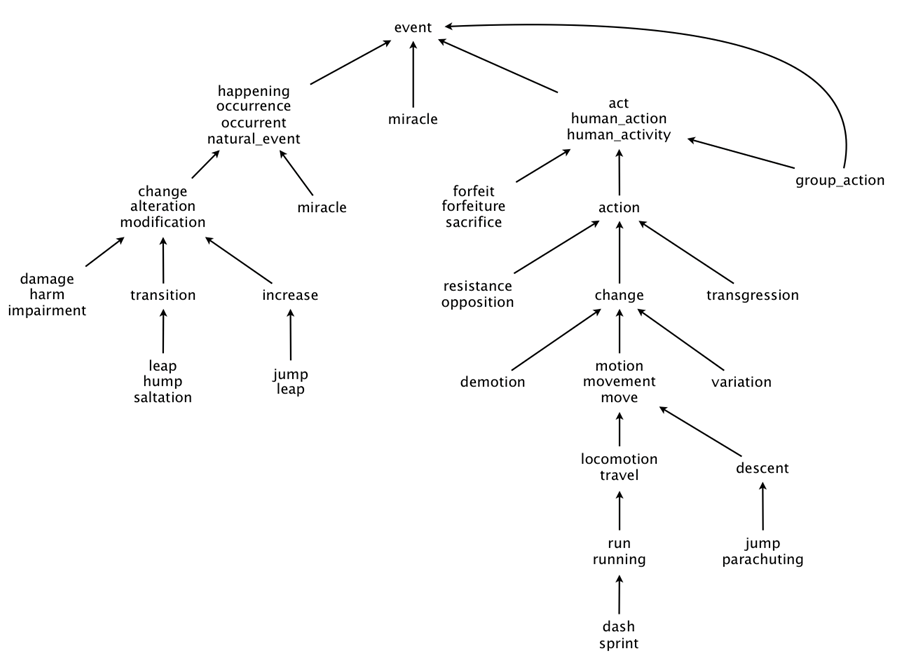
The following table explains three main terminologies that will be used throughout the document

|  |  |
| --- | --- |
| [WordNet](http://wordnet.princeton.edu/) | [WordNet](http://wordnet.princeton.edu/) 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 [Watson](http://en.wikipedia.org/wiki/Watson_(computer)) question-answering computer system. |
| **Synsets** | *Group of words that have same meaning*  WordNet groups words into sets of synonyms (words that have same meaning) 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. |
| **is-a relationship** | WordNet describes semantic relationships between synsets. One such relationship is the **is-a relationship**, which connects a hyponym (more specific synset, dealt as a child) to a hypernym (more general synset as a parent). For example, the synset { gate, logic gate } is considered as a parent of { AND circuit, AND gate } because an AND gate **is-a** kind of logic gate |

### WordNet is a Graph!

**As shown in Figure 1: Nouns** in theWordNet can be represented as a **directed graph**. Each vertex v is an integer that represents a synset, and each directed edge v→w represents the **is-a relationship** (v **is-a** w). A small subgraph of the WordNet digraph appears below.

Figure 1 WordNet Graph



### Some Properties

1. ***Rooted* Directed Acyclic Graph:** it is acyclic and has one vertex—the root—that is an ancestor of every other vertex. The root synset of the entire graph is “**entity**”.
2. ***Not necessarily a tree*:** a synset can have more than one parent (e.g. the synset “group\_action” in the above graph has TWO parents).
3. ***A noun can appear in more than one synset*:** it will appear once for each meaning that the noun has. For example, “jump” appears in TWO sunsets in the above graph.
4. ***Vertex considered an ancestor of itself***.

### Input Files

WordNet comes in two data files. The files are in comma-separated values (CSV) format: each line contains a sequence of fields, separated by commas.

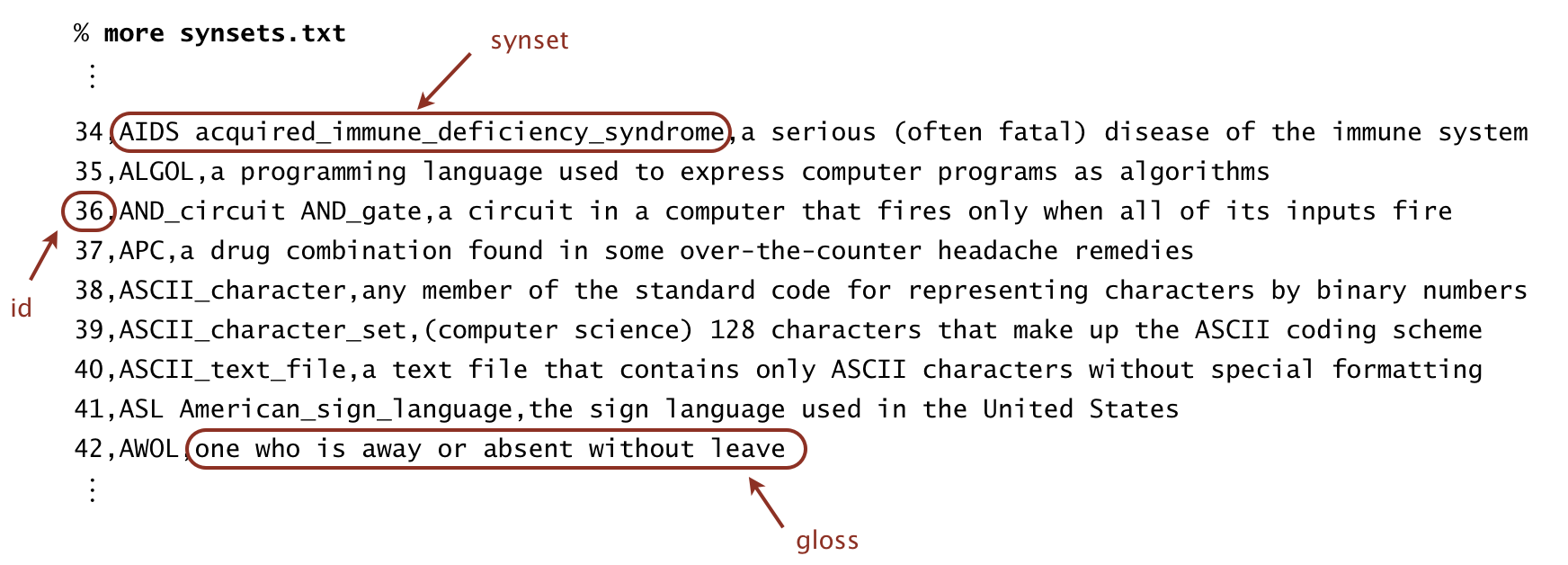
1. **List of synsets:**

The first file 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) (words are separated by spaces); and the third field is its dictionary definition (or gloss), which is not relevant to the project.

Simple example:

A line of the file will look like the following: it represent a synset with ***n*** words. Only highlighted part is the one we are interested in in the current project

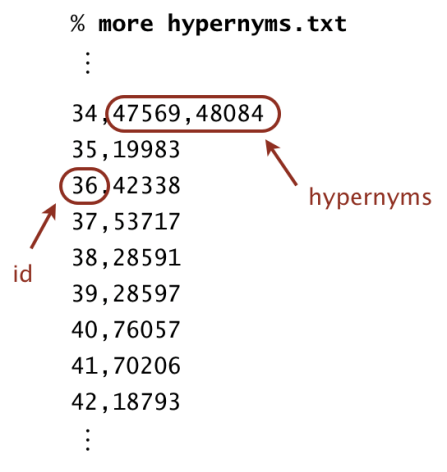
{ID},{word1} {word2} … {wordn},{gloss}

In more details the file will look like:

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).

1. **List of *hypernyms*:**

The second file 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.



For example, line 36 means that synset 36 (AND\_circuit AND\_Gate) has 42338 (gate logic\_gate) as its only hypernym (parent). Line 34 means that synset 34 (AIDS acquired\_immune\_deficiency\_syndrome) has two hypernyms (parents): 47569 (immunodeficiency) and 56099 (infectious\_disease).

## Required Semantics

Given the WordNet of nouns, it’s required to answer the following questions:

1. What’s the **semantic relation** between two nouns?
2. What’s the **odd word** in a given set of nouns?

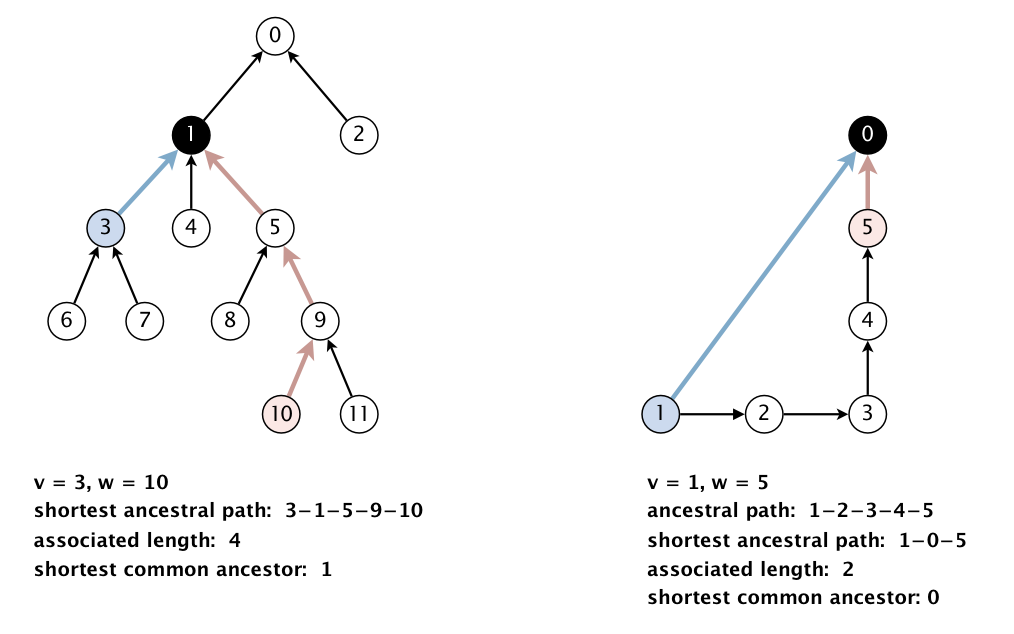
# Descriptions and Suggestions

The answers of both questions depend on finding “**Shortest common ancestor”** between nouns. Following are detailed description about this term and how to use it for answering the two questions.

## 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 (x is the nearest parent to both v and w)*. 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.

The following two examples find the shortest common ancestor of nodes with IDs 3 and 10, nodes 1 and 5. What is the nearest parent to both?

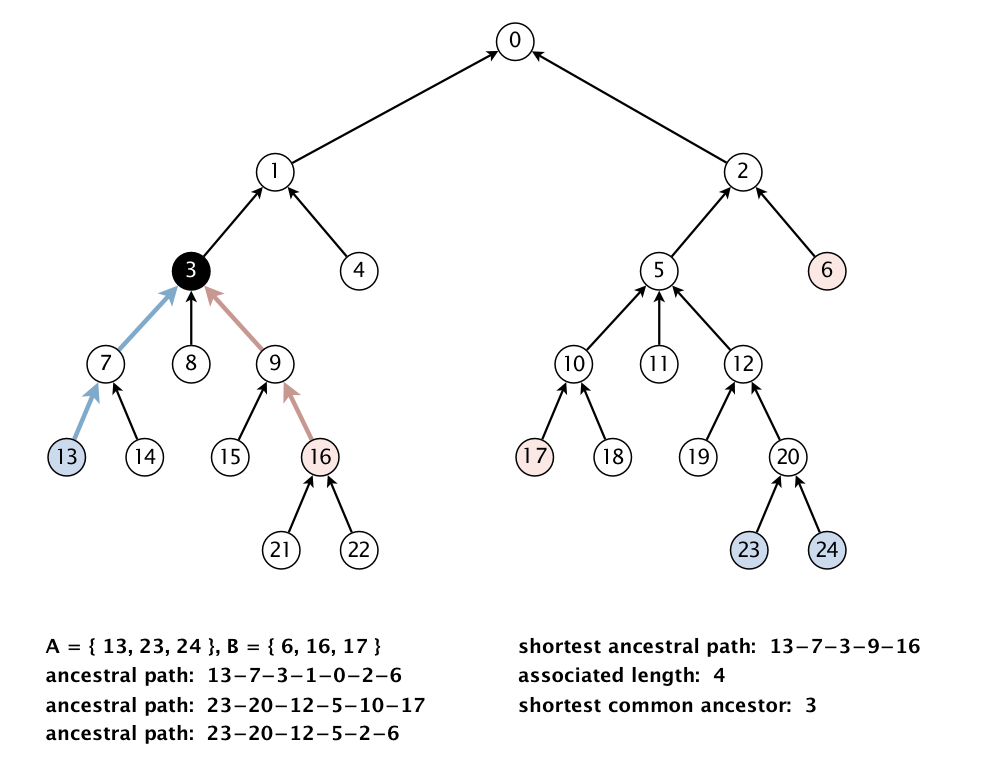


Thus, the ***shortest common ancestor*** indicates how two “nouns” are **semantically related** to each other. However, since a noun can appear in more than one synset, we need to:

1. Find all the synsets that each noun belongs to.

For example, as occur for “jump” noun in Figure 1. Say if the 2 synsets’ IDs are 9 and 24 then both of them shall be returned as jump belongs to both.

1. Generalize the 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* as shown in the figure below.



How will we use this to answer the questions?

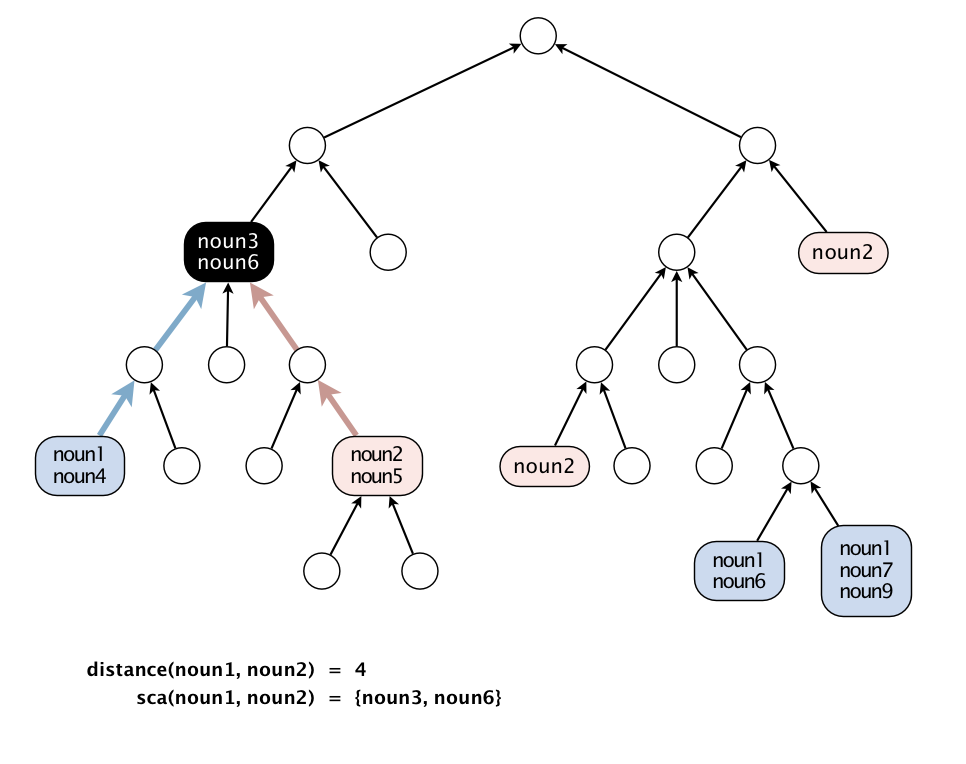
## Semantic Relatedness (for Question1)

Semantic relatedness refers to the degree to which two concepts are related. Measuring semantic relatedness is a challenging problem. For example, 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.

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*

*Refer to the figure below for a detailed example.*



## Outcast Detection (for Question2)

Given a list of WordNet nouns *x*1, *x*2, ..., *xn*, which noun is the least related to the others?

To identify *an outcast*,

* For each noun:
  + compute the sum of the distances between the current and every other one:

*di*   =   *distance*(*xi*, *x*1)   +   *distance*(*xi*, *x*2)   +   ...   +   *distance*(*xi*, *xn*)

and return a noun *xt* for which *dt* is the maximum. Note that because *distance*(*xi*, *xi*) = 0, it will not contribute to the sum.

# Project Requirements

## Required Implementation

|  |  |
| --- | --- |
| **Requirement** | **Performance** |
| 1. Construct the WordNet directed graph from the given input files. | **Space & Time:** **linear** in the input size (size of synsets & hypernyms files) |
| 1. EFFICIENTLY implement the following two mapping functions: | |
| * 1. Noun to SynsetsIDs: find the IDs of all synsets that the given noun belongs to. | **Time:** should be **less than** **O(N)**, N is number of synsets |
| * 1. SynsetsID to Nouns: find the nouns (synonyms) that belong to the given synset ID. | **Time:** should be **less than** **O(N)**, N is number of synsets |
| 1. EFFICIENTLY find the distance and the shortest common ancestor between: | |
| * 1. Two synsets IDs | **Space & Time:** **proportional to** **O(N + M)**, N is number of “***synsets***”, M is number of “***is a***-relations” |
| * 1. Two nouns |  |
| 1. Answer the two questions by finding:    1. Semantic relatedness between two nouns (distance and shortest common ancestor)    2. Outcast noun in a given list of nouns | |

## Input

1. **WordNet File1:** List of synsets
2. **WordNet File2:** List of hypernyms
3. **Semantic Relations File:** contains a list of noun pairs. First line contains the number of pairs (N). Next N lines contain the noun pairs, one pair per line. Each line contains two nouns that are separated by comma.
4. **Outcast Detection File:** contains set of cases, each case has a list of words. First line contains the number of cases (N). Next N lines contain the cases, one case per line. Each case contains a list of nouns that are separated by comma.

## Output

1. Semantic relatedness (distance and shortest common ancestor) of each pair of nouns in the “**Semantic Relations File**”.
2. Detect the outcast (odd) noun from each case of nouns in the “**Outcast Detection File**”.

## Test Cases

* Sample Case:
* Few test cases with small values that can be traced.
* Complete Test:
* Large cases for massive testing.

# Deliverables

## Implementation (60%)

1. Graph construction
2. Two mapping functions:
   1. Noun to SynsetsIDs
   2. SynsetsID to Nouns
3. Distance and the shortest common ancestor between:
   1. Two synsets IDs
   2. Two words
4. Answer the two questions by finding:
   1. Semantic relatedness
   2. Outcast noun

## Document (40%)

1. Describe the data structure(s) you used to store the information of “synsets” file. Why did you make this choice?
2. Describe the data structure(s) you used to store the information of “hypernyms” file. Why did you make this choice?
3. Describe your algorithm to compute the shortest common ancestor. Show the detailed analysis of the corresponding code?

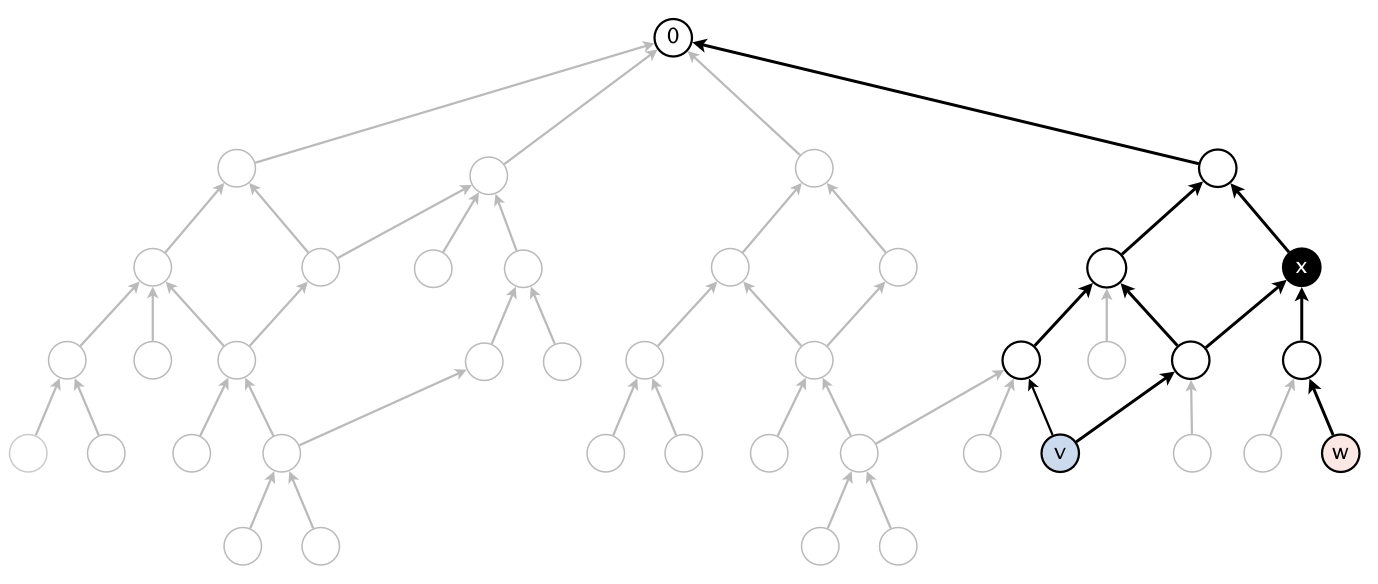
# Milestones

|  |  |  |
| --- | --- | --- |
|  | **Deliverables** | **Due to** |
| **Milestone1** | 1. Graph construction 2. Two mapping functions 3. Shortest common ancestor between two synsets IDs 4. Documentation | **(START of Week Before Practical Exam)** |
| **For Milestone1:**   * + **MUST** deliver the required tasks and **ENSURE** it’s worked correctly   + **MUST** deliver in your scheduled time (TO BE ANNOUNCED) | | |

# BONUSES

## FIRST: More Efficient Solution of SCA between Two Synsets IDs

Calculating the distance and the shortest common 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 can examine only the highlighted vertices and edges and it cannot initialize any vertex-indexed arrays.



## SECOND: Efficient Calculation of Distance and SCA between Two Nouns

Should be **proportional** to **O(N + M)**, when N is number of “synsets” and M is number of “is a-relations”