**WORDNET**

**PROBLEM STATEMENT:**

Wordnet is a semantic lexicon for the English language that computational linguists and cognitive scientists use extensively. 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 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 is a kind of logic gate.

**RELATED CONCEPTS:**

* Digraph
* BreadthFirstDirectedPath

**API:**

**SAP :**

public class SAP{

*/\*\*.*

*\* SAP class constructor*

*\* @param d the diagraph to be passed.*

*\*/*

public SAP (Digraph d)

*/\*\**

*\* This method determines the length of the shortest ancestral path between v and w*

*\* @param V the source vertex V*

*\* @param W the target vertex W*

*\* @return returns the path and returns -1 if no such path*

*\*/*

public int length (int V , int W)

*/\*\*.*

*\* This method returns the ancestor pertaining to the shortest length*

*\* @param v the group of vertices*

*\* @param w the group of vertices*

*\* @return the shortest ancestor*

*\*/*

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

*/\*\*.*

*\* This method returns the ancestor pertaining to the shortest length*

*\* @param v a ver tex*

*\* @param w another vertex*

*\* @return the shortest ancestor*

*\*/*

public int ancestor (int V, int W)

*/\*\**

*\* This method determines the length of the shortest ancestral path between v and w*

*\* @param V the source vertex V*

*\* @param W the target vertex W*

*\* @return returns the path and returns -1 if no such path*

*\*/*

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

}

**WordNet:**

public class WordNet {

*/\*\*.*

*\* This constructor creates the data structure for synsets and hypernyms.*

*\* @param synsets the synonym set*

*\* @param hypernyms the hypernyms file*

*\*/*

public WordNet(String synsets , String hypernyms)

*/\*\**

*\* This method returns all the WordNet nouns*

*\* @return returns the wordnet nouns.*

*\*/*

public Iterable<String> nouns()

*/\*\*.*

*\* This method checks whether the given noun is a wordNet noun*

*\* @param word the noun to be checked*

*\* @return returns true if it is a WordNet noun and false otherwise*

*\*/*

public boolean isNoun (String word)

*/\*\*.*

*\* This method yields the distance between nounA and nounB.*

*\* @param nounA the source noun*

*\* @param nounB the target noun*

*\* @return returns the shortest distance.*

*\*/*

public int distance (String nounA, String nounB)

*/\*\*.*

*\* This method returns a synset that is the common ancestor of nounA*

*\* and nounB in a shortest ancestral path*

*\* @param nounA the source noun*

*\* @param nounB the target noun*

*\* @return the shortest ancestor.*

*\*/*

public String sap(String nounA, String nounB)

}

**Outcast:**

public class Outcast {

*/\*\**

*\* This constructor takes a WorNet object*

*\* @param wordnet the given object.*

*\*/*

public Outcast (WordNet wordnet)

*/\*\*.*

*\* This method returns an outcast based on the given array of WordNet nouns*

*\* @param nouns the array of WordNet nouns*

*\* @return returns the outcast*

*\*/*

public String outcast (String[] nouns**)**

**}**