Joseph Martella
Professor Neville
CS 251 – Project 5 Report
April 13, 2011

- I chose to store the information in sysnsets.txt in a Hashtable. I thought this was the best data structure for multiple reasons. First, I only needed to store 2 parts of the data, the ID number and each noun. I chose a Hashtable with a <String, String> set up so I could store multiple ID numbers for each noun if they appeared in more than one synset. Storing the ID number as a String rather than an Integer allowed me to easily concatenate new ID numbers if necessary and split them up later. Using a Hashtable for this information also allowed me to create a simple isNoun() method that just needs to a quick lookup (O(1) because of how I set it up) in the Hashtable to determine whether or not the noun we are searching for is in the synset file.
- I didn't need to store the information in hypernyms.txt for long-term use. I read the data in line by line, split it up, and simply constructed the appropriate Digraph by adding edges. So, as far as data structures go, I stored the information in a Digraph that I ultimately used to construct the SAP.
- My algorithm to compute the SAP starts with running a BFS, using the class provided by the book, on the two points we are trying to find the SAP between. Then I iterate over each vertex in the graph and determine the total length of the path between the two vertices by adding the lengths of their individual paths to the vertex. If this length is the smallest one I found so far, I set it as the minimum. The worst case running time of this algorithm is approximately O(2(E+V) + V). This is because worst case for BFS is E+V because if the whole graph is connected, every edge and vertex will be touched, and in my algorithm, BFS is run twice, so you get 2(E+V). Then, I iterate over all the vertices to find the shortest length, so that's where the +V comes from. The best case running time is O(V). If the graph is not connected and the two vertices we are searching for the SAP for is the same vertex, the BFS on each will be O(1), but the algorithm would still iterate over each vertex in the graph, V.