HW 4: Fun with Ladders

Jonathan Friesen ~ jtf698

Vincent Ip ~ vti57

Design Rationale

Creating various objects like a dictionary, a word ladder solver, or an exception serve purposes just like real life objects. To actually create a word ladder, someone would grab a dictionary, have a process of looking through the dictionary for a word ladder, and something to indicate whether there will be a ladder or not. We solved this problem using a depth-first search, but an alternate method would be to use a breadth-first-search. From a programming perspective a DFS is easier to implement because a recursive solution is directly applicable. From a User perspective it doesn't exactly matter either way, although a BFS could be faster in some cases. Another alternate design choice would be to use a DFS but use it iteratively. This saves immense amount of stack space from a programming perspective, but increases the complexity of the program. The program could be expanded using any dictionary, and add future words. Our design could also be expanded to use words of longer or shorter length. Our design has distinct data structures and interactions that increase the modularity of the program. All of the sub methods implemented in our word ladder solver class are private and hide non-important info from the user.