# CSCI 3333 Project Three

# Finding Shortest Word Ladders

## Basic Part (100 points)

## The Word Ladder Problem:

In this homework assignment, I would like you to use some combination of graph, hash, list, stack, and queue data structures to solve the word ladder problem.

Given a start word and an end word, find a shortest sequence of words (*shortest word ladder*) from the start to the end so that

* Each word is a legal word from a given dictionary
* The next word is obtained from the previous word by substituting exactly one letter.
* If there is no such sequence, then say so.

**Note:** Assume all words are in lowercase. If they are not in lower case, then convert to lower for processing.

### Efficiency requirements:

* The dictionary will be provided later for this problem. This dictionary has over a quarter of million words. Given this large size, the design and implementation of your program must consider the run time efficiency, that is, the practical running time complexity of your program.
* Under the usual computing environment, e.g., a typical laptop or a desktop, your program shall be able to complete the work in no more than 1 minute of running time.
* One additional consideration is the space complexity. If you are not careful in your design of algorithms and data structures, you might try to use an adjacency matrix to represent the “word transition” graph, which might be too big to fit into your laptop or desktop.
* **Hints:**
  + You may consider designing a list of hash maps, one for each set of words with the same length.
  + You may consider which of the graph algorithms is best suited for your needs
    - Shortest path finding, such as Dijkstra’s algorithm
    - Breadth-first traversal
    - Depth-first traversal
  + You may also need to design data structures to record the paths for finding a word ladder. The question is:
    - Once a ladder is found, can you print it out easily?

### **Example:**

Given a start word ***zero*** and an end word ***five***, here is the ladder:

***zero => hero => here => hire => fire => five***

We assume that all the words are legal.

### **The Dictionary:**

The dictionary is given here: [The Dictionary for Word Ladders](https://faculty.utrgv.edu/zhixiang.chen/cs3333/3333/dic/wordLadder_dictionary.txt)

(<https://faculty.utrgv.edu/zhixiang.chen/cs3333/3333/dic/wordLadder_dictionary.txt>)

### **The Shortest-Path Problem:**

Consider each word in the dictionary as a node. If one word is obtained from another word by substituting exactly one letter, then there is an edge connecting these two words. In this setting, the word ladder problem is equivalent to find a shortest path from one word (node) to another word (node). If there is no such a path, then say so.

If you can build a graph as mentioned above for the words in the dictionary, then you can directly apply a shortest-path finding algorithm to solve the word ladder problem.

### The Challenge:

When the dictionary is large, like the one you are given, it is not easy to build a graph. It may take too long or too much space so that it becomes unrealistic to build a graph.

### **How to Overcome the Challenge for unweighted letters:**

Use the graph at conceptually level. During the solution process, there is no need to build the complete graph. Use some sort of breadth-first or depth-first search strategy to find the word ladder. Note: For any node representing a word, without accessing to a fully-built graph for the dictionary, you can find its adjacent nodes (or words you can obtained from it through substituting exactly one letter). Think about how to do so, how to do so as fast as you can.

**Cautions:**

* **The above suggestion works only for unweighted words.**
* **Caution: DO NOT PRINT OUT THE DATA FILES!**

### Specific Execution Time Requirement:

1. The loading time of your program shall not be more than three minutes and the execution time for each pair of two words shall not be more than two minutes (consider a typical laptop or a desktop). Otherwise, no credit will be given to your program.
2. Your program shall repeatedly ask the user to enter two words and then find the ladder for these two words or tell that no ladder exists. Each program will be tested by a fixed set of word pairs composed of
3. *fishery ---- compute (length = 23)*
4. *scroll ---- cloudy (length = 12)*
5. *scroll ---- sundae (no ladder)*
6. *computers ---- beautiful (no ladder)*
7. *gimlets ---- theeing (length = 21)*

## Optional Part for possible extra credits: The Weighted Word Ladder Problem

Now we consider a weighted version of the Word Ladder Problem. A weight table is given below:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **a** | **b** | **c** | **d** | **e** | **f** | **g** | **h** | **i** | **j** | **k** | **l** | **m** | **n** | **o** | **p** | **q** | **r** | **s** | **t** | **u** | **v** | **w** | **x** | **y** | **z** |
| **a** | 0 | 2 | 1 | 2 | 4 | 2 | 4 | 9 | 1 | 6 | 2 | 1 | 4 | 9 | 2 | 5 | 1 | 2 | 3 | 9 | 1 | 1 | 1 | 5 | 9 | 3 |
| **b** | 2 | 0 | 6 | 3 | 9 | 4 | 7 | 1 | 1 | 8 | 9 | 7 | 5 | 9 | 5 | 6 | 3 | 2 | 3 | 5 | 7 | 7 | 8 | 6 | 8 | 9 |
| **c** | 1 | 6 | 0 | 7 | 2 | 7 | 7 | 2 | 9 | 3 | 3 | 1 | 7 | 1 | 8 | 7 | 5 | 6 | 8 | 5 | 2 | 8 | 8 | 7 | 7 | 5 |
| **d** | 2 | 3 | 7 | 0 | 5 | 7 | 8 | 1 | 5 | 4 | 7 | 6 | 1 | 9 | 1 | 1 | 1 | 4 | 9 | 5 | 5 | 5 | 7 | 5 | 6 | 2 |
| **e** | 4 | 9 | 2 | 5 | 0 | 6 | 2 | 1 | 1 | 7 | 4 | 4 | 1 | 3 | 8 | 5 | 9 | 4 | 9 | 7 | 4 | 4 | 3 | 8 | 1 | 5 |
| **f** | 2 | 4 | 7 | 7 | 6 | 0 | 8 | 1 | 8 | 3 | 8 | 2 | 3 | 9 | 6 | 2 | 9 | 3 | 9 | 6 | 2 | 4 | 6 | 3 | 8 | 8 |
| **g** | 4 | 7 | 7 | 8 | 2 | 8 | 0 | 2 | 7 | 1 | 3 | 2 | 9 | 7 | 3 | 4 | 2 | 7 | 9 | 5 | 3 | 2 | 4 | 1 | 1 | 1 |
| **h** | 9 | 1 | 2 | 1 | 1 | 1 | 2 | 0 | 1 | 2 | 3 | 1 | 5 | 7 | 1 | 4 | 4 | 7 | 3 | 6 | 4 | 7 | 1 | 1 | 1 | 2 |
| **i** | 1 | 1 | 9 | 5 | 1 | 8 | 7 | 1 | 0 | 6 | 1 | 3 | 5 | 1 | 2 | 1 | 5 | 6 | 5 | 2 | 8 | 2 | 9 | 5 | 7 | 6 |
| **j** | 6 | 8 | 3 | 4 | 7 | 3 | 1 | 2 | 6 | 0 | 4 | 1 | 6 | 6 | 7 | 1 | 5 | 8 | 7 | 7 | 5 | 7 | 2 | 7 | 4 | 6 |
| **k** | 2 | 9 | 3 | 7 | 4 | 8 | 3 | 3 | 1 | 4 | 0 | 2 | 2 | 7 | 7 | 2 | 5 | 2 | 1 | 5 | 9 | 4 | 1 | 9 | 3 | 5 |
| **l** | 1 | 7 | 1 | 6 | 4 | 2 | 2 | 1 | 3 | 1 | 2 | 0 | 9 | 1 | 9 | 4 | 1 | 1 | 9 | 1 | 1 | 6 | 9 | 3 | 9 | 4 |
| **m** | 4 | 5 | 7 | 1 | 1 | 3 | 9 | 5 | 5 | 6 | 2 | 9 | 0 | 1 | 2 | 9 | 9 | 5 | 2 | 2 | 1 | 3 | 6 | 9 | 7 | 5 |
| **n** | 9 | 9 | 1 | 9 | 3 | 9 | 7 | 7 | 1 | 6 | 7 | 1 | 1 | 0 | 1 | 2 | 3 | 5 | 7 | 9 | 1 | 5 | 3 | 7 | 3 | 6 |
| **o** | 2 | 5 | 8 | 1 | 8 | 6 | 3 | 1 | 2 | 7 | 7 | 9 | 2 | 1 | 0 | 2 | 7 | 9 | 2 | 4 | 6 | 1 | 3 | 5 | 5 | 5 |
| **p** | 5 | 6 | 7 | 1 | 5 | 2 | 4 | 4 | 1 | 1 | 2 | 4 | 9 | 2 | 2 | 0 | 1 | 1 | 5 | 1 | 6 | 2 | 2 | 4 | 1 | 1 |
| **q** | 1 | 3 | 5 | 1 | 9 | 9 | 2 | 4 | 5 | 5 | 5 | 1 | 9 | 3 | 7 | 1 | 0 | 8 | 2 | 6 | 1 | 1 | 7 | 8 | 9 | 9 |
| **r** | 2 | 2 | 6 | 4 | 4 | 3 | 7 | 7 | 6 | 8 | 2 | 1 | 5 | 5 | 9 | 1 | 8 | 0 | 7 | 4 | 9 | 3 | 2 | 7 | 1 | 6 |
| **s** | 3 | 3 | 8 | 9 | 9 | 9 | 9 | 3 | 5 | 7 | 1 | 9 | 2 | 7 | 2 | 5 | 2 | 7 | 0 | 2 | 4 | 4 | 2 | 2 | 5 | 4 |
| **t** | 9 | 5 | 5 | 5 | 7 | 6 | 5 | 6 | 2 | 7 | 5 | 1 | 2 | 9 | 4 | 1 | 6 | 4 | 2 | 0 | 5 | 1 | 6 | 1 | 5 | 6 |
| **u** | 1 | 7 | 2 | 5 | 4 | 2 | 3 | 4 | 8 | 5 | 9 | 1 | 1 | 1 | 6 | 6 | 1 | 9 | 4 | 5 | 0 | 1 | 6 | 2 | 2 | 1 |
| **v** | 1 | 7 | 8 | 5 | 4 | 4 | 2 | 7 | 2 | 7 | 4 | 6 | 3 | 5 | 1 | 2 | 1 | 3 | 4 | 1 | 1 | 0 | 4 | 7 | 5 | 7 |
| **w** | 1 | 8 | 8 | 7 | 3 | 6 | 4 | 1 | 9 | 2 | 1 | 9 | 6 | 3 | 3 | 2 | 7 | 2 | 2 | 6 | 6 | 4 | 0 | 4 | 2 | 8 |
| **x** | 5 | 6 | 7 | 5 | 8 | 3 | 1 | 1 | 5 | 7 | 9 | 3 | 9 | 7 | 5 | 4 | 8 | 7 | 2 | 1 | 2 | 7 | 4 | 0 | 4 | 6 |
| **y** | 9 | 8 | 7 | 6 | 1 | 8 | 1 | 1 | 7 | 4 | 3 | 9 | 7 | 3 | 5 | 1 | 9 | 1 | 5 | 5 | 2 | 5 | 2 | 4 | 0 | 2 |
| **z** | 3 | 9 | 5 | 2 | 5 | 8 | 1 | 2 | 6 | 6 | 5 | 4 | 5 | 6 | 5 | 1 | 9 | 6 | 4 | 6 | 1 | 7 | 8 | 6 | 2 | 0 |

Note that this table is symmetric with respect to diagonal line. The plain text file of the above cost matrix is available at the Blackboard.

For example, the graph below shows edge weights as determined by the table

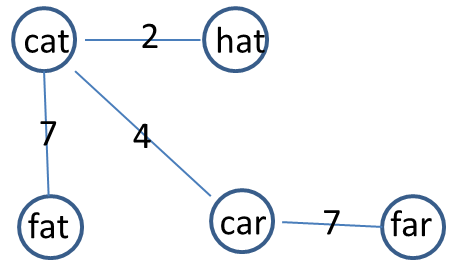
****

Figure 1. A weighted graph of letters

### 

Your need to re-do Basic Part with the weight word graphs as determined by the weight table. For a pair of the start word and the end word, this time you have to find the shorted weighted ladder from the start to the end. Specific execution time requirements from the unweighted word ladder problem apply to the weighted word ladder problem, and the same test word pairs will be used.

### Efficiency requirements:

* Under the usual computing environment, e.g., a typical laptop or a desktop, your program shall be able to complete the work in no more than 1 minute of running time.