First of all I read through the description of a problem a couple of times until I “internalized” and understood it. Then I thought of the possible solutions and since the problem doesn’t seem trivial I started to search for helpful resources online.

Initially I found a term “Levenshtein distance” which I heard of earlier in context of predictive text search (like autocomplete/suggestions that can sometimes be seen while searching). However, after reading about it it didn’t fit the problem very well, as it was too broad, it included insertions and deletions, and the problem was only about substitutions for the words of equal length.

After further reading I found about “Hamming distance” which basically fit the problem quite well, as following (citing from Wikipedia):

*“In*[*information theory*](https://en.wikipedia.org/wiki/Information_theory)*, the****Hamming distance****between two*[*strings*](https://en.wikipedia.org/wiki/String_(computer_science))*of equal length is the number of positions at which the corresponding*[*symbols*](https://en.wikipedia.org/wiki/Symbol)*are different. In other words, it measures the minimum number of substitutions required to change one string into the other, or the minimum number of errors that could have transformed one string into the other. In a more general context, the Hamming distance is one of several*[*string metrics*](https://en.wikipedia.org/wiki/String_metric)*for measuring the*[*edit distance*](https://en.wikipedia.org/wiki/Edit_distance)*between two sequences.”*

Next I implemented a function using TDD approach which counts Hamming distance between 2 words. The implementation is rather naïve, but it’s simple and I don’t think it would cause performance problems within the defined constraints.

After that I still thought about the final solution, as the implemented class didn’t solve the original problem as it only measure the distance between 2 words, however it doesn’t return the words between the “start” and “end” word.

In order to do this, after additional reading I decided to implement another class that would build a graph of word nodes connected with other word nodes by a smallest possible Hamming distance of 1.

That means word/node in graph would reference only the words that have 1 letter different than it.

A hashtable referencing the graphs by a word was created in order to quickly retrieve the graph for start and end word in order to use these for finding the shortest path and words in between.

The last part – that is finding the shortest path and displaying the words on this path was done by traversing the previously created graph using the BFS (Breadth First Search) algorithm.

If the path cannot be found as there are no words in dictionary that would allow to create the path program will throw exception.

Resources I researched (in random order):

<https://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Spring2006/assignments/editdistance/Levenshtein%20Distance.htm>

<https://stackoverflow.com/questions/41789767/finding-the-shortest-path-nodes-with-breadth-first-search/48260217#48260217>

<https://stackoverflow.com/questions/8379785/how-does-a-breadth-first-search-work-when-looking-for-shortest-path>

<https://stackoverflow.com/questions/47336884/how-to-find-shortest-path-when-given-a-list-generated-from-a-bfs>

<https://en.wikipedia.org/wiki/Graph_traversal>

<https://ru.wiktionary.org/w/index.php?title=%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:Search&ns6=1&search=%D0%BC%D0%B5%D1%82%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC&fulltext=%D0%A0%D0%B0%D1%81%D1%88%D0%B8%D1%80%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9+%D0%BF%D0%BE%D0%B8%D1%81%D0%BA&searchToken=1a6ly7c5vvdzczia36ireohym>

<https://stackoverflow.com/questions/11811918/how-to-compute-shortest-distance-between-two-words>

<https://en.wikipedia.org/wiki/Edit_distance>

<https://stackoverflow.com/questions/2205540/algorithm-to-transform-one-word-to-another-through-valid-words>

<https://softwareengineering.stackexchange.com/questions/288962/hamming-distance-using-a-dictionary>

<https://medium.com/basecs/breaking-down-breadth-first-search-cebe696709d9>

<https://en.wikipedia.org/wiki/Breadth-first_search>

<https://en.wikipedia.org/wiki/Hamming_distance>

<https://en.wikipedia.org/wiki/Levenshtein_distance>

<https://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Spring2006/assignments/editdistance/Levenshtein%20Distance.htm>

<https://en.wikibooks.org/wiki/Algorithm_Implementation/Strings/Levenshtein_distance>