Algorithm Find the number of nodes in the shortest path to convert *begin_word* to *end_word* using only valid conversions

Require: The length of both the words should be same
Ensure: You can only change one character at a time

```
1:  function WORD_LADDER(begin_word, end_word, word_list)
    ▷ dictionary is a hashmap containing all the valid words in the dictionary
    ▷ visited is a hashmap containing all the strings which are already processed in the queue
2:      for each word in word_list do
3:          dictionary.insert(word)
4:      queue.push(begin_word, 1)
5:      visited.insert(begin_word)
6:      while queue is not empty do                                  ▷ Perform BFS
7:          (current_string, level) ← queue.front
8:          queue.pop
9:          if current_string == end_word then
10:             return level
11:         for Each element in current_string do
12:             for Each char in the alphabet do
13:                 backup ← element
14:                 Replace element by char                          ▷ Create the new string
15:                 if current_string is in dictionary then
16:                     if current_string is not in visited then
17:                         queue.push(current_string, level + 1)
18:                         visited.push(current_string)
19:                 Replace char by backup                           ▷ Get the original string back
20:     return Not Possible
```