**Word Ladder**

Given two words (start and end), and a dictionary, find the length of shortest transformation sequence from start to end, such that only one letter can be changed at a time and each intermediate word must exist in the dictionary. For example, given:

start = "hit"

end = "cog"

dict = ["hot","dot","dog","lot","log"]

**from** queue **import** Queue  
**from** string **import** ascii\_lowercase  
  
  
**class** WordNode:  
  
 **def** \_\_init\_\_(self, word, numsteps):  
 self.word = word  
 self.numsteps = numsteps  
  
  
**def** ladderlength(beginword, endword, dictionary):  
 qu = Queue()  
 qu.put(WordNode(beginword, 1))  
  
 **while**(**not** qu.empty()):  
 top = qu.get()  
 word = top.word  
  
 **if** word == endword:  
 **return** top.numsteps  
  
 i = 0  
 **while** i < len(word):  
 **for** j **in** ascii\_lowercase:  
 **if** word[i] != j:  
 newword = word[:i] + j + word[i+1:]  
  
 **if** newword **in** dictionary:  
 qu.put(WordNode(newword, top.numsteps+1))  
 *# to eliminate cycles, remove the new word* dictionary.remove(newword)  
 i = i + 1  
 **return** 0  
  
  
start = **"hit"**end = **"cog"**dictionary = {**"hot"**, **"dot"**, **"dog"**, **"lot"**, **"log"**, **"cog"**}  
  
print(**"Word Ladder Length: "**, ladderlength(start, end, dictionary))