



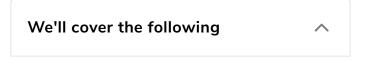






Solution Review: Word Formation Using a Hash Table

This review provides a detailed analysis of the solution to the Word Formation Using a Hash Table Challenge.



- Solution: Iterative Word Matching
 - Time Complexity

Solution: Iterative Word Matching

```
main.py
HashTable.py
HashEntry.py
     from HashTable import HashTable
  2
  3
     def is_formation_possible(lst, word):
  5
         if len(word) < 2 or len(lst) < 2:</pre>
              return False
  9
         hash table = HashTable()
         for elem in lst:
 10
 11
              hash_table.insert(elem, True)
 12
 13
          for i in range(1, len(word)):
```

```
8/30/22, 12:55 PM
                            Solution Review: Word Formation Using a Hash Table - Data Structures for Coding Interviews in Python
                        # Slice the word into two strings in each iteration
                        first = word[0:i]
          15
                        second = word[i:len(word)]
          16
                        check1 = False
          17
          18
                        check2 = False
          19
          20
                        if hash_table.search(first) is not None:
         21
                            check1 = True
                        if hash_table.search(second) is not None:
          22
         23
                            check2 = True
          24
                       # Return True If both substrings are present in the hash tabl
         25
                        if check1 and check2:
          26
                            return True
          27
          28
                                                                                             []
```

This is as efficient as the implementation as the trie implementation. We insert all the dictionary words into a hash table.

Just like before, a for loop begins and slices the word into two substrings in each iteration. Whenever both substrings are found in the hash table, the function returns True.

Note: The solution only works for two words and not more.

Time Complexity

We perform the insert operation \mathbf{m} times for a list of size \mathbf{m} . After that, we linearly traverse the word of size \mathbf{n} once. Furthermore, we slice strings of size \mathbf{n} in each iteration. Hence the total time complexity is $O(m+n^2)$.



companies apply to you instead of you applying to them. how ①









Next →

Challenge 7: Word Formation Using a ...

Challenge 8: Find Two Numbers that ...



Mark as Completed



Report an Issue

