









Solution Review: Find Symmetric Pairs in a List

This review provides a detailed analysis of the solution to the Find Symmetric Pairs in a List Challenge.



- Solution: Using a Dictionary/Set
 - Time Complexity

Solution: Using a Dictionary/Set

```
def find_symmetric(my_list):
2
        # Create an empty set
        pair_set = set()
3
        result = []
4
5
        # Traverse through the given list
        for pair in my list:
6
            # Make a tuple and a reverse tuple out of the pair
7
            pair_tup = tuple(pair)
8
9
            pair.reverse()
10
            reverse_tup = tuple(pair)
            # Check if the reverse tuple exists in the set
11
12
            if(reverse_tup in pair_set):
13
                # Symmetric pair found
14
                result.append(list(pair tup))
                result.append(list(reverse_tup))
15
16
            else:
                # Insert the current tuple into the set
17
                pair_set.add(pair_tup)
18
19
        return result
20
```

The solution above uses a Python set . However, a dictionary can be used as well. For each pair in the list, we create a **tuple** and a **reverse tuple**.

Note: Tuples are an immutable sequence of elements in python.

If the reverse tuple already exists in the set, we have found symmetric pairs.

If not, we can add the tuple to the list. If a symmetric pair exists, it will be able to find this pair later in the loop.

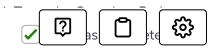
Time Complexity

The hash table lookups work in constant time. Hence, our traversal of the input list makes the algorithm run in O(n) where $\bf n$ is the list size.

Interviewing soon? We've partnered with Hired so that companies apply to you instead of you applying to them. See how ①



Next - C



! Report an Issue

