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.

We'll cover the following

Solution: Using a Dictionary/Set

```
def find_symmetric(my_list):
    # Create an empty set
    pair_set = set()
    result = []
    # Traverse through the given list
    for pair in my_list:
        # Make a tuple and a reverse tuple out of the pair
        pair_tup = tuple(pair)
        pair.reverse()
        reverse_tup = tuple(pair)
        # Check if the reverse tuple exists in the set
        if(reverse_tup in pair_set):
            # Symmetric pair found
            result.append(list(pair_tup))
            result.append(list(reverse_tup))
        else:
            # Insert the current tuple into the set
            pair_set.add(pair_tup)
    return result

```

```
21
22  arr = [[1, 2], [4, 6], [4, 3], [6, 4], [5, 9], [3, 4],  ]]
23  symmetric = find_symmetric(arr)
24  print(symmetric)
25
```

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 **n** is the list size.

Report an Issue