Solution Review: Trace the Complete Path of a Journey

This review provides a detailed analysis of the solution to the Trace the Complete Path of a Journey Challenge.

We'll cover the following

Solution: A Hash Table to Deduce The Starting Point

```
def trace_path(my_dict):
    result = []
    # Create a reverse dict of the given dict i.e if the given dict has (N,C
    # then reverse dict will have (C,N) as key-value pair
    # Traverse original dict and see if it's key exists in reverse dict
    # If it doesn't exist then we found our starting point.
    # After the starting point is found, simply trace the complete path
    # from the original dict.
    reverse_dict = dict()
    # To fill reverse dict, iterate through the given dict
    keys = my_dict.keys()
    for key in keys:
        reverse_dict[my_dict.get(key)] = key
    # Find the starting point of itinerary
    from_loc = None
    keys_rev = reverse_dict.keys()
    for key in keys:
        if key not in reverse_dict:
```

```
            from_loc = key
            break
            # Trace complete path
    to = my_dict.get(from_loc)
    while to is not None:
        result.append([from_loc, to])
        from_loc = to
        to = my_dict.get(to)
    return result

```

The first thing we need to do is find the starting point of the journey. A `reverse_dict` is created to switch the sources and destinations in the original `map`.

The key which does not appear in `reverse_dict` has never been a destination in `map`. Hence, it is the starting city.

From here, we simply traverse from city to city based on the previous destination.

Time Complexity

Although a hash table is created and traversed, both take the same amount of time. The complexity for this algorithm is *O(n)* where **n** is the number of source-destination pairs.

Challenge 4: Trace the Complete Path ...

Challenge 5: Find Two Pairs in List suc...

Mark as Completed

Report an Issue