



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.

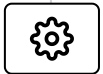
We'll cover the following

- Solution: A Hash Table to Deduce The Starting Point
- Time Complexity

Solution: A Hash Table to Deduce The Starting Point

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

```
19         from_loc = key
20         break
21         # Trace complete path
22         to = my_dict.get(from_loc)
23         while to is not None:
24             result.append([from_loc, to])
25             from_loc = to
26             to = my_dict.get(to)
27         return result
28
```



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.

Interviewing soon? We've partnered with Hired so that companies apply to you instead of you applying to them. [See how](#) ⓘ





Challenge 4: Trace the Complete Path ...



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



Mark as Completed



Report an Issue

