



Solution Review: Check if Lists are Disjoint

This review provides a detailed analysis solution to the Check if Lists are Disjoint Challenge.

We'll cover the following



- Solution: Use a Set
- Time Complexity

Solution: Use a Set

```
1 def is_disjoint(list1, list2):
2     s = set(list1) # Create set of list1 elements
3     # iterate list 2
4     for elem in list2:
5         # if element in list1 then return False
6         if elem in s:
7             return False
8     # Return True if no common element
9     return True
10
11
12 list1 = [9, 4, 3, 1, -2, 6, 5]
13 list2 = [7, 10, 8]
14 list3 = [1, 12]
15 print(is_disjoint(list1, list2))
16 print(is_disjoint(list1, list3))
17
```





Nothing tricky going on here. The problem is very similar to the previous one. All we have to do is create a set for `list1` and as soon as we find value from `list2` or `list3` in the `set`, we can conclude that the two lists are not disjoint. Since the `set` uses a hash table under the hood, the complexity for checking an element to be in the `set` will be $O(1)$.

Time Complexity#

For a lookup list with m elements and a subset list with n elements, the time complexity is $O(m+n)$.

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

[← Back](#)[Next →](#)

Challenge 2: Check if Lists are Disjoint

Challenge 3: Find Symmetric Pairs in a...

☒ Mark as Completed



Report an Issue



