Solution Review: Find Two Numbers that Add up to "k"

This review provides a detailed analysis of the different ways to solve the find two numbers that add up to k.

We'll cover the following

Solution #1: Brute Force

```
def find_sum(lst, k):
    # iterate lst with i
    for i in range(len(lst)):
        # iterate lst with j
        for j in range(len(lst)):
            # if sum of two iterators is k
            # and i is not equal to j
            # then we have our answer
            if(lst[i]+lst[j] is k and i is not j):
                return [lst[i], lst[j]]


print(find_sum([1, 2, 3, 4], 5))
```

This is the most time intensive but intuitive solution. Traverse the whole list of size, say **s**, for each element in the list and check if any of the two elements add up to the given number `k`. So, using two nested for-loops each iterating over the entire list will serve the purpose.

Time Complexity

Since we iterate over the entire list of k elements, n times in the worst case, therefore, the time complexity is $O(n^2)$.

Solution #2: Sorting the List

```
def binary_search(a, item):
    first = 0
    last = len(a) - 1
    found = False
    index = -1
    while first <= last and not found:
        mid = (first + last) // 2
```

```
        if a[mid] == item:
            index = mid
            found = True
        else:
            if item < a[mid]:
                last = mid - 1
            else:
                first = mid + 1
    if found:
        return index
    else:
        return -1


def find_sum(lst, k):
    lst.sort()
    for j in range(len(lst)):
        # find the difference in list through binary search
        # return the only if we find an index
        index = binary_search(lst, k -lst[j])
        if index is not -1 and index is not j:
            return [lst[j], k -lst[j]]



print(find_sum([1, 5, 3], 2))
print(find_sum([1, 2, 3, 4], 5))
```

While solution #1 is very intuitive, it is not very time efficient. A better way to solve this challenge is by first sorting the list. Then for each element in the list, use a binary search to look for the difference between that element and the intended sum. In other words, if the intended sum is `k` and the first element of the sorted list is $\mathbf{a_0}$, then we will do a binary search for $\mathbf{k\text{-}a_0}$. The search is repeated for every $\mathbf{a_i}$ up to $\mathbf{a_n}$ until one is found." You can implement the `binary_search()` function however you like, recursively or iteratively.

Time Complexity

Since most optimal comparison-based sorting functions take $O(nlogn)$, let's assume that the Python `.sort()` function takes the same. Moreover, since binary search takes $O(logn)$ time for a finding a single element, therefore a binary search for all n elements will take $O(nlogn)$ time."

Solution #3: Moving indices

```
def find_sum(lst, k):
    # sort the list
    lst.sort()
    index1 = 0
    index2 = len(lst) - 1
    result = []
```

```
    sum = 0
    # iterate from front and back
    # move accordingly to reach the sum to be equal to k
    # returns false when the two indices meet
    while (index1 != index2):
        sum = lst[index1] + lst[index2]
        if sum < k:
            index1 += 1
        elif sum > k:
            index2 -= 1
        else:
            result.append(lst[index1])
            result.append(lst[index2])
            return result
    return False


print(find_sum([1, 2, 3, 4], 5))
print(find_sum([1, 2, 3, 4], 2))
```

Time Complexity

The linear scan takes $O(n)$ and sort takes $O(nlogn)$. The time complexity becomes $O(nlogn) + O(n)$ because the sort and the linear scan are done one after the other. The overall would be $O(nlogn)$ in the worst case.

> **Note:** The solution provided above is not the optimal solution for this problem. We can write a more efficient solution using hashing. We will cover that approach in Hashing Chapter: Challenge 8 (https://www.educative.io/courses/data-structures-in-python-an-interview-refresher/B13DmMq4YRJ)

Back

Challenge 3: Find Two Numbers that ...

Next

Challenge 4: List of Products of all Ele...

Completed

Report an Issue

Ask a Question (https://discuss.educative.io/tag/solution-review-find-two-numbers-that-add-up-to-k__introduction-to-lists__data-structures-for-coding-interviews-in-python)