Daily Coding Problem Blog

Daily Coding Problem #1

Problem

This problem was recently asked by Google.

Given a list of numbers and a number k, return whether any two numbers from the list add up to k.

For example, given `[10, 15, 3, 7]` and k of `17`, return true since `10 + 7` is `17`.

Bonus: Can you do this in one pass?

Solution

This problem can be solved in several different ways.

Brute force way would involve a nested iteration to check for every pair of numbers:

```
def two_sum(lst, k):
    for i in range(len(lst)):
        for j in range(len(lst)):
            if i != j and lst[i] + lst[j] == k:
                return True
  return False
```

This would take $O(N^2)$. Another way is to use a set to remember the numbers we've seen so far. Then for a given number, we can check if there is another number that, if added, would sum to k. This would be O(N) since lookups of sets are O(1) each.

```
def two_sum(lst, k):
    seen = set()
    for num in lst:
        if k - num in seen:
            return True
        seen.add(num)
    return False
```

Yet another solution involves sorting the list. We can then iterate through the list and run a binary search on `K - lst[i]`. Since we run binary search on N elements, this would take O(N log N) with O(1) space.

```
from bisect import bisect_left



def two_sum(lst, K):
    lst.sort()

    for i in range(len(lst)):
        target = K - lst[i]
        j = binary_search(lst, target)
        # Check that binary search found the target and that it's not in the same index
        # as i. If it is in the same index, we can check lst[i + 1] and lst[i - 1] to see
        #  if there's another number that's the same value as lst[i].
        if j == -1:
            continue
        elif j != i:
            return True
        elif j + 1 < len(lst) and lst[j + 1] == target:
            return True
        elif j - 1 >= 0 and lst[j - 1] == target:
            return True
    return False

def binary_search(lst, target):
    lo = 0
    hi = len(lst)
    ind = bisect_left(lst, target, lo, hi)

    if 0 <= ind < hi and lst[ind] == target:
        return ind
    return -1
```