Kevin Sekuj
2/08/2022
CS325: Analysis of Algorithms
Homework 5

1.
    a. We solve this problem with a backtracking algorithm. First, we can sort our input to avoid appending duplicate subarrays to our return array. Next, we initialize our base cases in our backtracking algorithm. We require two – one for a case where the path we're on cannot lead to an answer, or when the path does lead to an answer – depending on whether we've reached our target sum or gone over it.

       If our current path hasn't reached an answer yet, then we continue our backtracking by incrementing our index variable which represents the position in our current subarray.
    b. Amount.py
    c. Time complexity: **Exponential**
2.
    a. We can solve this problem with a greedy algorithm. Since our goal is to maximize the amount of satisfied dogs, and we're given two input arrays, we can begin our algorithm by sorting both arrays. This is useful because it ensures we can assign dog food to dogs in increasing order – so the dogs with the smallest appetite will receive the appropriate biscuit size first, and we can increment our satisfied dogs counter. At that point, we simply advance both pointers and continue searching. So, we greedily give biscuits to dogs, because we're making the optimal decision at each iteration of the algorithm.
    b. FeedDog.py
    c. Time complexity: O(NLogN) + O(MLogM) where N and M are input arrays