HW 5: Backtracking + Greedy Algorithm

Question 1:

a. describe backtracking problem

```
function amount_helper(A, S, start, array):
    if (A == 0):        // base case
        return []

    for i in range(length of A):
        append.A[i]          // Add to list
        A[i] - A[add] > target    // if list > than target
        POP. A[i]  // remove that item AKA backtracking
    return result
function Amount(A, S):
    amount_helper(A, S, 0, [])
    return final_result
```

* my way of solving requires a helper. It will go through the list, append items unless goes over target sum. if it equals 0 then it will return that list.

b. in gradescope

c. the time complexity is $O(2^n)$ = exponential. Backtracking means going through the whole list + returning all results that equal target or what you're solving. For the example above, it would have to check the list w/ all of the ways it could satisfy. So this is a, can be, slow as it can exponentially grow.

Question 2:

a.

```
def feedDog(hunger_level, biscuit_size)
    for i in range(hunger_level):
        for j in range(biscuit_size):
            if(biscuit ≥ hunger)
                count += 1
                biscuit -= hunger[i]
return count
```

in this, I would iterate through both list then if there are more biscuits than hunger, add to count and remove that biscuit and continue

For the greedy aspect, I can sort in desc order to feed the most

HW 5:

Question 2

C. time complexity

The time complexity of this problem is $O(n^2)$. this is due to the fact that I will have two loops iterating, but it's just the length of both lists, so its $\Theta(n^2)$