cs3821 Assignment 3

Jennifer Xu (z5207930)

Question 3

3.1 Table

x_1	x_2	x_3
4	0	0
2	1	0
1	0	3
0	2	0

3.2 Brute Force Algorithm - Pseudo Code

```
nSolutions = 0
def formS(array, target):
    Global nSolutions
    # Base Case: Array is empty
    if (len(array) == 0):
        if (target != 0): return False
        nSolutions += 1
        return True

    for i in range(target + 1):
        # Recursively call formS
        # each time taking out the last item in the array
        formS(array[:-1], target - i * array[-1])
```

Total number of solutions is found with the variable `nSolutions`.

3.3 Time Complexity

M is the smallest value in C.

N is the length of C.

S is the target number.

Each recursive call has a single subproblem. Each subproblem has a max of S/M recursive calls. The recursion terminates after being called N times. Therefore it's S/M problems wide and N problems wide. Hence the upper asymptotic bound is N^(S/M). Note: Since our target sum is able to keep decreasing, the expected time complexity is much smaller.

Therefore, the big-O complexity is O(n^n).

3.4 Dynamic Programming Algorithm

Our subproblem, P(n), would be to find the number of solutions for our target sum n using our coefficients C.

Hence, the number of solutions P(n) would be found by adding the amount we already know of to the amount of to P(n - C[i]).

For example, given S = 4, C = [1, 2, 3],

P(3) = P(3 - C[0]) + P(3 - C[1]) + P(3 - C[0])
= P(2) + P(1) + P(0)

...And so on, repetitively using our subcases and building up the number of ways we can find P(n).

To do this, we utilise a table P, size S, to record the number of solutions for P(n). First we iterate through C, and then in a nested iteration, we iterate through P(c[i]) to P(S).

Therefore by the time we finish our nested while loop, P[S] will hold the number of solutions to the number of combinations of the linear equation represented by C to achieve the target S.