CMSC 142 Project 2nd semester 2023-2024

The Subset Sum Solver

There are many definitions for the Subset Sum problem. For this project, we'll use this:

> Given a set S of unique positive integers and a positive integer k, is there a subset of S that adds up to k?

Example:

```
S = {3, 21, 17, 14, 8, 1, 9, 34, 5, 11}
k = 22
Subsets (6):
{3, 14, 5}
{3, 8, 11}
{21, 1}
{17, 5}
{14, 8}
{8, 9, 5}
```

Requirement #1: **A solver program (C program)**
This solver program should:
(1) identify whether there are any valid solutions (subsets). If there are, your program should produce all valid subsets. This includes the correct number of valid subsets. If there aren't, print a message saying so.
(2) read inputs from a text file with the following format:

```
2                           //# of sets
22                          //target sum for the first set
3 21 17 14 8 1 9 34 5 11    //integers of the first set
70                          //target sum for the second set
67 4 33 19 12               //integers of the second set
```

You are **required to use the iterative backtracking algorithm** discussed in the laboratory in creating the Subset Sum Solver. You can modify the basic algorithm (e.g. you can add branch and bound). You must code in C.

Requirement #2: **Analysis / Essay (PDF)**
This should contain your thoughts and analysis on your solver and the Subset Sum Problem. Some questions/ideas that you can answer:
1) How would you describe the difference between the dynamic programming solution compared to your solver? If your solver only needs to produce one solution, how does it compare performance-wise?
2) Graph the real-time running time of your solver while N gets larger.
3) Besides the number of elements, does anything else affect the real-time running time? e.g. variation of inputs, size of the inputs, distance of the target, etc.