For each question, don't just write pseudocode/code with no explanation. You must summarize in English prose the intuition of your algorithm, and also explain why it works correctly.

I encourage you to try solving these completely on your own, just using the textbook and notes from class, but you can use the Internet to research algorithms if you want. In any case, your write up and explanations must be entirely your own words and thoughts. (I highly caution against using ChatGPT and friends because they have no reasoning/logic power behind the text that its algorithm generates.)

1 Equal Subset Sum

Let $x_1, x_2, \ldots, x_n$ be a set of integers, and let $S = \sum_{i=1}^{n} x_i$. Design an algorithm to partition the set into two subsets of equal sum, or determine that it is impossible to do so. The algorithm should run in time $O(nS)$. (Hint: Don't over think this; build on what we did in class.)

2 Maximum square submatrix

Given an $m \times n$ boolean matrix B, find its largest square submatrix whose elements are all zeroes. Design a dynamic programming algorithm and indicate its time efficiency. (The algorithm may be useful for, say, finding the largest free square area on a computer screen; or for selecting a construction site.)

3 Stacking Boxes

We have some boxes numbered 1 to N. The dimensions of all boxes are identical but their weights differ. Each box has a weight W_i, and a maximum load L_i. (You can think of this as having two arrays, `W` and `L`, where `W[i]`/`L[i]` is the weight/max load of box `i`).

You need to stack up a subset of the boxes (each box put directly on top of exactly one other box), subject to the following constraints:

- A lower numbered box cannot be put on one with a higher number;
- The total weight of all boxes upon a box should not exceed its maximum load.

Describe an efficient algorithm that finds the maximum number of boxes that can be stacked up according to the above constraints.

For example, suppose

$$W = [19, 7, 5, 6, 1]$$
$$L = [15, 13, 7, 8, 2]$$

Then your algorithm should produce 4.

(Hint: Maybe research the *0-1 Knapsack problem* and dynamic programming algorithm.)