1. [10 marks] Maximum omit-one subrange sum. We consider a variation on the maximum subrange sum problem studied in class. Let $A[1..n]$ be an integer array, and $A[i..j]$ be a non-empty subrange. The omit-one sum is the sum of all but the least element in $A[i..j]$, i.e. $S(i, j) = (\sum_{k=i}^{j} A[k]) - \min_{k=i}^{j} A[k]$. The problem asks for the maximum value of the omit-one sum, i.e. $\max_{1 \leq i \leq j \leq n} S(i, j)$. Design an efficient divide-and-conquer algorithm to solve this problem. You only need to compute the maximum value, not the subrange it comes from.

2. [10 marks] Weighted Longest Common Subsequence. Let $S = s_1s_2 \cdots s_m$ and $T = t_1t_2 \cdots t_n$ be two strings. Additionally, assume each letter $a$ in the alphabet $\Sigma$ is associated with a weight $w(a)$ that is given to you as part of the input. The weight of a string is the sum of the weights of all its letters. So if vowels are weighted 2 and consonants 5, the weight of `example` is 26.

Give an efficient algorithm to compute a common subsequence of the two input strings having the largest weight. If there are multiple subsequences with the same weight, you only need to return one.

3. [10 marks] Word Puzzle. Consider an $n \times n$ matrix $M$ of characters. The value of $M[i, j]$ is the character in the $i$th row and $j$th column of $M$. You are also given a string $s$ and asked to play the following game. You perform a walk in the matrix starting from the upper-left corner. At each step you are allowed to move either to
the right or downward by one step. Can you find a sequence of moves such that the sequentially-visited characters form the given string \( s \)?

For example: If \( M \) is given below and \( s = \text{CS341} \), then a solution is the walk \((1, 1) \rightarrow (1, 2) \rightarrow (2, 2) \rightarrow (3, 2) \rightarrow (3, 3)\).

Design an \( O(n^2) \) dynamic programming algorithm to answer this problem. Your algorithm needs to return \text{Fail} \) if there is no such path, and return the path as a list of coordinates if such a path exists. If multiple paths exist, you only need to return one.

4. [bonus; 10 extra credit marks only] Redo question 3, but now where you can start the walk at any position of the matrix. Your algorithm needs to run in \( O(n^3) \) time.