1. Rocky Mountain High!

Suppose you are given an n x m matrix of positive integers denoting an elevation map of a park in the Rocky Mountains. Your goal is to design a path from the left side of the map (column 1) to the right side of the map (column m). Each step in the path must proceed by going either straight right to the next column, or diagonally up and right, or diagonally down and right (no going straight up or backwards). The cost of each step in the path is the absolute value of the difference in elevations between the two cells being traversed. The total cost of a path is the sum of each step in the path. The goal is to compute a minimum possible cost path (hikers want a path that is as level as possible) connecting the left side of the map to the right side. Design an algorithm to compute the cost of the shortest possible path in O(nm) time.

2. Zippers

Given three strings, you are to determine whether the third string can be formed by combining the characters in the first two strings. The first two strings can be mixed arbitrarily, but the characters from each must stay in their original order in the third string.

For example, consider forming "tcarete" from "cat" and "tree":
String A**: c a t**
String B: **t r e e**
String C: **t c a r e t e**

As you can see, we can form string C by selecting the first character of "tree", followed by the first 2 characters of "cat", followed by the second and third characters of "tree", followed by the last character of "cat" and "tree" respectively.

As a second example, consider forming "catrtee" from "cat" and "tree":
String A: **c a t**
String B: **t r e e**
String C: **c a t r t e e**

The answer for this input is also 'yes'

Finally, notice that it is impossible to form "cttaree" from "cat" and "tree", meaning the answer for this input would be 'no'.

Zipper Problem
Input
- String A = *a1, a2, a3, ... ,* a_n
- String B = *b1, b2, b3, ...,* b_m
- String C = *c1, c2, c3, ...,* c_{m+n}

Output: Output *yes* if A and B can be combined (zippered) into string C.
Output *no* if A and B cannot be combined to form C.

Design an O(nm) time algorithm to solve this problem.

3. **Skyrim Loot Problem**:

You are playing Skyrim, and your character can carry a total weight of W in their inventory, where W is some positive integer. You enter the bandits' lair, where you find n pieces of loot. The first piece of loot weighs w_1 pounds and has a value of v_1 gold pieces, where w_1 and v_1 are positive integers. The second item in the lair has weight w_2 and value v_2, etc. Your goal is to select a subset of the n items to take such that the total value is as large as possible, but the sum of the weights is less than or equal to your weight capacity W. In other words:

Input:

- a positive integer W
- n pairs of positive integers denoting the weight and value of each object: (w_1,v_1), $(w_2,v_2),\ldots,(w_n,v_n)$

Output: The maximum possible total value you can obtain by selecting a subset of the n items while keeping the total weight of the selected items to be at most W.

Design an algorithm to solve the Skyrim loot problem. Analyze the run time of your solution in terms of W and n.