Algorithmic Thinking in Problem Solving
Exam 2

Instructions

Solve **5** of the following problems using dynamic programming (you **must** use dynamic programming to earn points. Other solutions will not be accepted).

For each problem, you must do the following:

1. Define the problem/solution recursively (this is the most important step – if you don't do this correctly, you will automatically **get a 0** for the problem regardless of the code you write).
2. **Briefly** talk about how you plan to store solutions to sub-problems and combining them to solve the global problem (talk about the data structure/variables you'll use to solve the problem).
3. Talk about how you used **IDEAL** and **Duke 7** to tackle the problem
4. Code your solution.

Upload everything to **GitHub**. Use readme files (one per problem) to do steps 1 and 2.

Each problem will be graded as follows:

- Step 1: 70%
- Step 2: 20%
- Step 3: 12%

Notice that:

- There are 2 extra credit points per problem
- You can get a 90 on the final exam without having to code anything.

---

1 - Minimum Falling Path Sum

Given a **square** array of integers `A`, we want the **minimum** sum of a *falling path* through `A`.

A falling path starts at any element in the first row, and chooses one element from each row. The next row's choice must be in a column that is different from the previous row's column by at most one.

Example 1:

Input: [[1,2,3],
[4,5,6],
[7,8,9]]
Output: 12
Explanation:
The possible falling paths are:

- [1,4,7], [1,4,8], [1,5,7], [1,5,8], [1,5,9]
- [2,4,7], [2,4,8], [2,5,7], [2,5,8], [2,5,9], [2,6,8], [2,6,9]
- [3,5,7], [3,5,8], [3,5,9], [3,6,8], [3,6,9]

The falling path with the smallest sum is [1,4,7], so the answer is 12.

Note:

1. 1 <= A.length == A[0].length <= 100
2. -100 <= A[i][j] <= 100

---

2 - Palindromic Substrings

Given a string, your task is to count how many palindromic substrings in this string.

The substrings with different start indexes or end indexes are counted as different substrings even they consist of same characters.

Example 1:

Input: "abc"
Output: 3
Explanation: Three palindromic strings: "a", "b", "c".

Example 2:

Input: "aaa"
Output: 6
Explanation: Six palindromic strings: "a", "a", "a", "aa", "aa", "aaa".

Note:

1. The input string length won't exceed 1000.

---

3 - Arithmetic Slices

A sequence of number is called arithmetic if it consists of at least three elements and if the difference between any two consecutive elements is the same.

For example, these are arithmetic sequence:

1, 3, 5, 7, 9
7, 7, 7, 7
3, -1, -5, -9

The following sequence is not arithmetic.

1, 1, 2, 5, 7

A zero-indexed array A consisting of N numbers is given. A slice of that array is any pair of integers (P, Q) such that 0 <= P < Q < N.

A slice (P, Q) of array A is called arithmetic if the sequence:
A[P], A[p + 1], ..., A[Q - 1], A[Q] is arithmetic. In particular, this means that P + 1 < Q.

The function should return the number of arithmetic slices in the array A.

Example:

A = [1, 2, 3, 4]

return: 3, for 3 arithmetic slices in A: [1, 2, 3], [2, 3, 4] and [1, 2, 3, 4] itself.

---

4 - Minimum ASCII Delete Sum for Two Strings

Given two strings s1, s2, find the lowest ASCII sum of deleted characters to make two strings equal.

Example 1:

Input: s1 = "sea", s2 = "eat"
Output: 231
Explanation: Deleting "s" from "sea" adds the ASCII value of "s" (115) to the sum.
Deleting "t" from "eat" adds 116 to the sum.
At the end, both strings are equal, and 115 + 116 = 231 is the minimum sum possible to achieve this.

Example 2:

Input: s1 = "delete", s2 = "leet"
Output: 403

Explanation: Deleting "dee" from "delete" to turn the string into "let",
adds 100[d]+101[e]+101[e] to the sum. Deleting "e" from "leet" adds 101[e] to the sum.
At the end, both strings are equal to "let", and the answer is 100+101+101+101 = 403.
If instead we turned both strings into "lee" or "eet", we would get answers of 433 or 417, which are higher.

Note:

- 0 < s1.length, s2.length <= 1000.
- All elements of each string will have an ASCII value in [97, 122].

---

5 - Maximum Length of Pair Chain

You are given n pairs of numbers. In every pair, the first number is always smaller than the second number.

Now, we define a pair (c, d) can follow another pair (a, b) if and only if b < c. Chain of pairs can be formed in this fashion.

Given a set of pairs, find the length longest chain which can be formed. You needn't use up all the given pairs. You can select pairs in any order.

Example 1:

Input: [[1,2], [2,3], [3,4]]
Output: 2
Explanation: The longest chain is [1,2] -> [3,4]

Note:

1. The number of given pairs will be in the range [1, 1000].

---

6 - Integer Break

Given a positive integer *n*, break it into the sum of **at least** two positive integers and maximize the product of those integers. Return the maximum product you can get.

Example 1:

Input: 2
Output: 1

Explanation: 2 = 1 + 1, 1 × 1 = 1.

Example 2:

Input: 10
Output: 36
Explanation: 10 = 3 + 3 + 4, 3 × 3 × 4 = 36.

Note: You may assume that *n* is not less than 2 and not larger than 58.

---

7 - Partition to K Equal Sum Subsets

Given an array of integers nums and a positive integer k, find whether it's possible to divide this array into k non-empty subsets whose sums are all equal.

Example 1:

Input: nums = [4, 3, 2, 3, 5, 2, 1], k = 4
Output: True
Explanation: It's possible to divide it into 4 subsets (5), (1, 4), (2,3), (2,3) with equal sums.

Note:

- 1 <= k <= len(nums) <= 16.
- 0 < nums[i] < 10000.

---

8 - Perfect Squares

Given a positive integer *n*, find the least number of perfect square numbers (for example, 1, 4, 9, 16, ...) which sum to *n*.

Example 1:

Input: *n* = 12
Output: 3
Explanation: 12 = 4 + 4 + 4.

Example 2:

Input: $n = 13$
Output: 2
Explanation: $13 = 4 + 9$.