Algorithms

DP Tabulation Homework 2

Mostafa S. Ibrahim
Teaching, Training and Coaching for more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

Problem #1: LeetCode 801 - Minimum Swaps To Make Sequences Increasing

You are given two integer arrays of the same length `nums1` and `nums2`. In one operation, you are allowed to swap `nums1[i]` with `nums2[i]`.

- For example, if `nums1 = [1,2,3,8]`, and `nums2 = [5,6,7,4]`, you can swap the element at `i = 3` to obtain `nums1 = [1,2,3,4]` and `nums2 = [5,6,7,8]`.

Return *the minimum number of needed operations to make* `nums1` *and* `nums2` ***strictly increasing***. The test cases are generated so that the given input always makes it possible.

An array `arr` is **strictly increasing** if and only if `arr[0] < arr[1] < arr[2] < ... < arr[arr.length - 1]`.

- `2 <= nums1.length <= 10^5`
- `nums2.length == nums1.length`
- `0 <= nums1[i], nums2[i] <= 2 * 10^5`

Example 1:

```
Input: nums1 = [1,3,5,4], nums2 = [1,2,3,7]
Output: 1
Explanation:
Swap nums1[3] and nums2[3]. Then the sequences are:
nums1 = [1, 3, 5, 7] and nums2 = [1, 2, 3, 4]
which are both strictly increasing.
```

Example 2:

```
Input: nums1 = [0,3,5,8,9], nums2 = [2,1,4,6,9]
Output: 1
```

- Note: Writing the memoization version is easy/medium
- But converting it to tabulation version is medium/hard
- **Optional**: write O(1) memory code

Problem #2: LeetCode 264 - Ugly Number II

An **ugly number** is a positive integer whose prime factors are limited to `2`, `3`, and `5`.

Given an integer `n`, return *the* `n^th` ***ugly number***.

- 1 <= n <= 1690
- Tip: Although the last number fits in 32 bit, some of your intermediate answers won't fit. Use 64 bit for languages like C++
  - This is a tricky overflow case :)
- Tip: don't try to write memoization code. Think DP-like tabulation approach
  - Optional: Can you do more analysis to find O(N) approach?

Example 1:

```
Input: n = 10
Output: 12
Explanation: [1, 2, 3, 4, 5, 6, 8, 9, 10, 12] is the sequence of the first
10 ugly numbers.
```

Example 2:

```
Input: n = 1
Output: 1
Explanation: 1 has no prime factors, therefore all of its prime factors
are limited to 2, 3, and 5.
```

Problem #3: LeetCode 376 - Wiggle Subsequence

A **wiggle sequence** is a sequence where the differences between successive numbers strictly alternate between positive and negative. The first difference (if one exists) may be either positive or negative. A sequence with one element and a sequence with two non-equal elements are trivially wiggle sequences.

- For example, `[1, 7, 4, 9, 2, 5]` is a **wiggle sequence** because the differences `(6, -3, 5, -7, 3)` alternate between positive and negative.
- In contrast, `[1, 4, 7, 2, 5]` and `[1, 7, 4, 5, 5]` are not wiggle sequences. The first is not because its first two differences are positive, and the second is not because its last difference is zero.

A **subsequence** is obtained by deleting some elements (possibly zero) from the original sequence, leaving the remaining elements in their original order.

Given an integer array `nums`, return *the length of the longest **wiggle subsequence** of* `nums`.

Example 1:

```
Input: nums = [1,7,4,9,2,5]
Output: 6
Explanation: The entire sequence is a wiggle sequence with differences (6,
-3, 5, -7, 3).
```

Example 2:

```
Input: nums = [1,17,5,10,13,15,10,5,16,8]
Output: 7
Explanation: There are several subsequences that achieve this length.
One is [1, 17, 10, 13, 10, 16, 8] with differences (16, -7, 3, -3, 6, -8).
```

Example 3:

```
Input: nums = [1,2,3,4,5,6,7,8,9]
Output: 2
```

Your task

- Develop an $O(n^2)$ time memoization algorithm similar to LIS
- Convert it to tabulation
- Find a smart observation to reduce the tabulation to O(n) time and memory
- Finally, switch it to O(n) time and O(1) memory

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."