Gondola

Solutions

Overview

This is an easy problem. Even though it might seem that there are three separate tasks (as indicated by the grouping of subtasks), the task is

actually incremental. The three parts (check whether there is a solution – find one solution – count all solutions) are closely related.

Subtasks 1-3

In subtask 1, once we see the first gondola (i.e., `inputSeq[0]`), the rest is uniquely determined. We just need to iterate through the sequence and check whether everything matches.

- N = len(sequence)
- for n:=1..N: if sequence[n] % N != (sequence[0]+n) % N: return False
- return True

The same code will actually solve subtask 2.

The pseudocode for the general subtask 3 is:

- If there exist one of the original gondolas: check whether the other original gondolas are in the expected places, if not, return `false`.
- Return `true` if all the values are distinct, `false` otherwise.

Subtasks 4-6

A simple solution for subtask 4: if the largest number in the sequence is `n`, terminate, otherwise output the only missing number and terminate.

Solutions for subtasks 5 and 6 share the same idea, the difference is that subtask 4 allows its inefficient implementations. There are many possible solutions. Here is one of them.

- Collect all non-original gondolas. For each of them determine the original gondola it replaced.
- Sort these records according to the new gondola number.
- In sorted order, replace the original gondolas by new ones until the expected numbers are reached.

Note that the case where no original gondolas are present may require special attention – not just for the contestants, but also in the grader for these subtasks.

Subtasks 7-10

Counting the repair sequences directly is hard. One way of doing it is by asking the question: `How many repair sequences start with gondola x being replaced?` for each `x`. Each choice of `x` leads us to a new state with one fewer gondolas to replace.

Using this idea we can now solve subtask 8 by dynamic programming: for each admissible state of the lift we compute the number of ways in which it can be solved.

Subtasks 9 and 10 require one additional insight. (This is, probably, the only tricky part of this problem, and the insight needed is not too hard.)

Instead of looking at the old gondola that is being removed, we will simply look at the new gondola that is being added. How many different

options do we have for its place? If the new gondola is present in the final sequence, its place is uniquely determined. Otherwise, the number of places where

this new gondola can be added is simply the number of places that end up having a gondola with a larger number.

Additionally, we need to multiply the result by n if none of the original gondolas is present. (All n cyclic rotations of the original sequence are now possible,

and the repair sequences for different rotations are necessarily distinct.) We outline the algorithm as follows:

- Run the algorithm for subtasks 1-3 to check whether the input sequence is valid or not.
- If valid, for each replaced gondola we find the original gondola it ultimately replaced, and we sort these records according to the new gondola number.
- The total number of possibilities can now be computed by multiplying the number of possible locations for each gondola between $n+1$ and the largest replaced gondola number present. Note the multiplicative operation is modular here.
- Finally, multiply by n if all original gondolas are replaced.