[CS 11 25.1] Lab 3h – 寿司

Cheatsheet is available here: https://oj.dcs.upd.edu.ph/cs11cheatsheet/

Submit solution [CS 11 25.1] Lab Exercise 3

My submissions

Points: 190 (partial)
Time limit: 4.0s
Memory limit: 2G

Problem type

Allowed languages
py3

Problem Statement

You are at a sushi restaurant that uses a conveyor belt, and there are n plates of sushi (numbered 1 to n) that are going to pass by your table. Each plate of sushi has a *spiciness* s.

To make sure you enjoy your meal, you are going to abide by two main rules:

- You will get the first plate of sushi that passes by your table.
- Let s be the spiciness of the last plate of sushi you got, and let t be the spiciness of a plate that passes by your table. You will get this plate if and only if s and t differ by no more than k.

Which plates of sushi are you going to get?

Task Details

Your task is to implement a function named `get_plates`, which should have the following *signature*:

```
def get_plates(conveyor_belt, k):
```

Copy

The above says that it has two arguments `conveyor_belt` and k.

- `conveyor_belt` is a `tuple` of integers (`int`s) denoting the spiciness of the plates of sushi in the conveyor belt. Note that the first element of `conveyor_belt` corresponds to the first plate of sushi that will pass by your table.
- k is an integer.

The function must return a list of integers denoting the spiciness of the plates of sushi that you will get.

Restrictions

- The following symbols can now be used: `list`, `set`, `dict`, `enumerate`, `append`, `pop`, `extend`, `remove`, `sort`, `sorted`, `insert`, `clear`, `reverse`, `reversed`.
- Loops are allowed.
- Recursion is *disallowed*.
- Comprehensions are *disallowed*.
- Your source code must have at most 400 bytes.

Examples

Example 1 Function Call

```
get_plates((1, 10, 3, 2, 7, 5, 8), 3)
```

Copy

Example 1 Return Value

```
[1, 3, 2, 5, 8]
```

Copy

Constraints

- The function `get_plates` will be called at most $70{,}000$ times.
- $0 \le n \le 350{,}000$
- $0 \le s, k \le 10^{20}$
- The sum of n across all calls to `get_plates` will be $\le 350{,}000$.

Scoring

Note: New tests may be added and all submissions may be rejudged at a later time. (All future tests will satisfy the constraints.)

- You get 90 ❤️ points if you solve all test cases where:
  - $n \le 50$
  - The sum of n across all calls to `get_plates` will be ≤ 600.
- You get 50 🔴 points if you solve all test cases where:
  - $n \le 5{,}000$
  - The sum of n across all calls to `get_plates` will be $\le 10{,}000$.
- You get 50 🔴 points if you solve all test cases.

Clarifications

Report an issue

No clarifications have been made at this time.