[CS 11 25.1] Lab 7h – Substrings and Superstrings

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

Sub [CS 11 25.1] Lab Exercise 7

My submissions

Points: 140 (partial)
Time limit: 4.0s
Memory limit: 1G

Problem type

Allowed languages
py3

Problem Statement

Given strings s and t, we say s is a **substring** of t, equivalently t is a **superstring** of s, if s appears somewhere in the string t consecutively somewhere. For example, `ring` is a substring of `rings`, `string` and `strings`, but not of `rinig`. Also, `rings`, `string` and `strings` are superstrings of `ring`.

You are given a sequence of strings. For each string, please give the *set* of all of its superstrings and substrings in the sequence.

Task Details

In this problem, you will be provided a module named `oj_strs` whose contents are exactly as follows:

```
from dataclasses import dataclass

@dataclass
class StrData:
    superstrings: set[str]
    substrings: set[str]
```

Import it using

```
from oj_strs import StrData
```

Your task is to implement a function called `get_string_data`. It takes a single argument, a `tuple` or `list` of n `str`s representing the strings.

It must return a `dict[str, StrData]`, that is, a `dict` whose keys are `str`s and values are of type `StrData`. The keys must be all the distinct strings among those in the input, and for each key s, the corresponding value should contain two `set[str]`s, one containing the superstrings of s in the input, the other containing the substrings of s in the input.

Restrictions

Note that some names are banned. Here are a few of them: `input`, `type`. This is not an exhaustive list. (If you accidentally use a variable name that turns out to be banned, please rename it.)

The following names are allowed: `map`, `filter`.

The following imports are allowed:

- `count`, `islice`, `chain`, `takewhile`, `starmap` and `zip_longest` from `itertools`.
- `cache`, `lru_cache`, `total_ordering`, `partial`, `reduce` and `wraps` from `functools`.
- `randint`, `randrange` and `choice` from `random`.
- `Fraction` from `fractions`.
- `dataclass` from `dataclasses`.
- `contextmanager` from `contextlib`.
- `Enum`, `auto` from `enum`.

(Read the docs to learn what they do!)

Anonymous functions are allowed.

Inner functions are allowed.

Classes, dataclasses and enums are allowed.

For this problem in particular:

- The following imports are allowed: `StrData` from `oj_strs`.
- The source code limit is 3000.

Example Calls

Example 1 Function Call

```
get_string_data(('a', 'banana', 'anna', 'ann', 'an', 'hannah
'an'))
```

Example 1 Return Value

```
{
    'a': StrData(
        superstrings={'a', 'banana', 'anna', 'ann', 'an',
'hannah'},
        substrings={'a'},
    ),
    'an': StrData(
        superstrings={'banana', 'anna', 'ann', 'an', 'hannah'},
        substrings={'a', 'an'},
    ),
    'hannah': StrData(
        superstrings={'hannah'},
        substrings={'a', 'anna', 'ann', 'an', 'hannah'},
    ),
    'anna': StrData(
        superstrings={'anna', 'hannah'},
        substrings={'anna', 'ann', 'an', 'a'},
    ),
    'banana': StrData(
        superstrings={'banana'},
        substrings={'an', 'a', 'banana'},
    ),
    'ann': StrData(
        superstrings={'anna', 'hannah', 'ann'},
        substrings={'an', 'a', 'ann'},
    ),
}
```

Testing

To test your program locally, you should create a file called `oj_strs.py` and save the code above to it. Note that this `oj_strs.py` is **not** to be submitted! The judge has its own version of `oj_strs.py`. The `oj_strs.py` you create is only for your own testing.

Constraints

- The function `get_string_data` will be called at most 20 times.
- $0 \le n \le 50$
- Each string has length between 0 and 50, inclusive and consists of lowercase English letters.

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 120 ❤️ points if you solve all test cases where:
  - each string is nonempty.
- You get 20 🔴 points if you solve all test cases.

Clarifications

Report an issue

No clarifications have been made at this time.