Programming and Data Structures in Python, 2023

Graded Assignment 2, 1 Oct 2023, due 9 Oct 2023

Write three Python functions as specified below. Combine the text for all three functions together into a single file. Your function will be called automatically with various inputs and should return values as specified. Do not write commands to read any input or print any output.

- You may define additional auxiliary functions as needed.
- In all cases you may assume that the value passed to the function is of the expected type, so your function does not have to check for malformed inputs.

Note

- Test on Swayam portal
- Official submissions on Moodle

---

1. Write a Python function `histogram(l)` that takes as input a list of integers with repetitions and returns a list of pairs as follows:.
    - for each number `n` that appears in `l`, there should be exactly one pair `(n,r)` in the list returned by the function, where `r` is the number of repetitions of `n` in `l`.
    - the final list should be sorted in ascending order by `r`, the number of repetitions. For numbers that occur with the same number of repetitions, arrange the pairs in ascending order of the value of the number.

    For instance:

    ```
    >>> histogram([13,12,11,13,14,13,7,7,13,14,12])
    [(11, 1), (7, 2), (12, 2), (14, 2), (13, 4)]

    >>> histogram([7,12,11,13,7,11,13,14,12])
    [(14, 1), (7, 2), (11, 2), (12, 2), (13, 2)]

    >>> histogram([13,7,12,7,11,13,14,13,7,11,13,14,12,14,14,7])
    [(11, 2), (12, 2), (7, 4), (13, 4), (14, 4)]
    ```

2. An airline has assigned each city that it serves a unique numeric code. It has collected information about all the direct flights it operates, represented as a list of pairs of the form `(i,j)`, where `i` is the code of the starting city and `j` is the code of the destination.

    It now wants to compute all pairs of cities connected by one intermediate hope — city `i` is connected to city `j` by one intermediate hop if there are direct flights of the form `(i,k)` and `(k,j)` for some other city `k`. The airline is only interested in one hop flights between different cities — pairs of the form `(i,i)` are not useful.

    Write a Python function `onehop(l)` that takes as input a list of pairs representing direct flights, as described above, and returns a list of all pairs `(i,j)`, where `i != j`, such that `i` and `j` are connected by one hop. Note that it may already be the case that there is a direct flight from `i` to `j`. So long as there is an intermediate `k` with a flight from `i` to `k` and from `k` to `j`, the list returned by the function should include `(i,j)`. The input list may be in any order. The pairs in the output list should be in lexicographic (dictionary) order. Each pair should be listed exactly once.

    Here are some examples of how your function should work.

    ```
    >>> onehop([(2,3),(1,2)])
    [(1, 3)]

    >>> onehop([(2,3),(1,2),(3,1),(1,3),(3,2),(2,4),(4,1)])
    [(1, 2), (1, 3), (1, 4), (2, 1), (3, 2), (3, 4), (4, 2), (4, 3)]

    >>> onehop([(1,2),(3,4),(5,6)])
    []
    ```

3. Let us consider polynomials in a single variable x with integer coefficients. For instance:

   $3x^4 - 17x^2 - 3x + 5$

   Each term of the polynomial can be represented as a pair of integers (coefficient,exponent). The polynomial itself is then a list of such pairs.

   We have the following constraints to guarantee that each polynomial has a unique representation:

   - Terms are sorted in descending order of exponent
   - No term has a zero cofficient
   - No two terms have the same exponent
   - Exponents are always nonnegative

   For example, the polynomial introduced earlier is represented as:

   ```
   [(3,4),(-17,2),(-3,1),(5,0)]
   ```

   The zero polynomial, 0, is represented as the empty list `[]`, since it has no terms with nonzero coefficients.

   Write Python functions for the following operations:

   ```
   addpoly(p1,p2)
   multpoly(p1,p2)
   ```

   that add and multiply two polynomials, respectively.

   You may assume that the inputs to these functions follow the representation given above. Correspondingly, the outputs from these functions should also obey the same constraints.

   You can write auxiliary functions to "clean up" polynomials – e.g., remove zero coefficient terms, combine like terms, sort by exponent etc. Build a library of functions that can be combined to achieve the desired format.

   You may also want to convert the list representation to a dictionary representation and manipulate the dictionary representation, and then convert back.

   Some examples:

   ```
   >>> addpoly([(4,3),(3,0)],[(-4,3),(2,1)])
   [(2, 1),(3, 0)]

   Explanation: (4x^3 + 3) + (-4x^3 + 2x) = 2x + 3

   >>> addpoly([(2,1)],[(-2,1)])
   []

   Explanation: 2x + (-2x) = 0

   >>> multpoly([(1,1),(-1,0)],[(1,2),(1,1),(1,0)])
   [(1, 3),(-1, 0)]

   Explanation: (x - 1) * (x^2 + x + 1) = x^3 - 1
   ```
