Week 4 Programming Assignment

Due on 2019-08-29, 23:59 IST

Write two Python functions as specified below. Paste the text for both functions together into the submission window. 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.
- For each function, there are normally some public test cases and some (hidden) private test cases.
- "Compile and run" will evaluate your submission against the public test cases.
- "Submit" will evaluate your submission against the hidden private test cases. There are 10 private test cases, with equal weightage. You will get feedback about which private test cases pass or fail, though you cannot see the actual test cases.
- Ignore warnings about "Presentation errors".

---

1. Write a Python function frequency(l) that takes as input a list of integers and returns a pair of the form (minfreqlist,maxfreqlist) where
   - minfreqlist is a list of numbers with minimum frequency in l, sorted in ascending order

- maxfreqlist is a list of numbers with maximum frequency in l, sorted in ascending

For instance

```
>>> frequency([13,12,11,13,14,13,7,11,13,14,12])
([7], [13])

>>> frequency([13,12,11,13,14,13,7,11,13,14,12,14,14])
([7], [13, 14])

>>> frequency([13,12,11,13,14,13,7,11,13,14,12,14,14,7])
([7, 11, 12], [13, 14])
```

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 hop --- 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.

   For instance

```
>>> 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)])
[]
```

Private Test cases used for evaluation	Input	Expected Output	Actual Output	Status
Test Case 1	frequency([17322,271898,374,374,374,423432423,423432423,423432423,5325,5325,5325,5325,5325])	([17322, 271898], [5325])\n	([17322, 271898], [5325])\n	Passed
Test Case 2	frequency([17322,374,17322,374,17322,374])	([374, 17322], [374, 17322])\n	([374, 17322], [374, 17322])\n	Passed
Test Case 3	frequency([9842])	([9842], [9842])\n	([9842], [9842])\n	Passed
Test Case 4	frequency([-17322,-271898,-374,-374,-374,-423432423,-423432423,-423432423,-5325,-5325,-5325,-5325,-5325])	([-271898, -17322], [-5325])\n	([-271898, -17322], [-5325])\n	Passed
Test Case 5	frequency([-17322,-374,-17322,-374,-17322,-374])	([-17322, -374], [-17322, -374])\n	([-17322, -374], [-17322, -374])\n	Passed
Test Case 6	onehop([(1,3),(1,2),(2,3),(2,1),(3,2),(3,1)])	[(1, 2), (1, 3), (2, 1), (2, 3), (3, 1), (3, 2)]\n	[(1, 2), (1, 3), (2, 1), (2, 3), (3, 1), (3, 2)]\n	Passed
Test Case 7	onehop([(1, 2), (1, 3), (1, 4), (1, 5), (1, 6),	[(1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1,	[(1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1, 7),	Passed

```
(1, 7), (1, 8), (1, 9), (2, 1), (2, 3), (2, 4), (2, 5), (2, 6), (2, 7), (2, 8), (2, 9), (3, 1), (3, 2), (3, 4), (3, 5), (3, 6), (3, 7), (3, 8), (3, 9), (4, 1), (4, 2), (4, 3), (4, 5), (4, 6), (4, 7), (4, 8), (4, 9), (5, 1), (5, 2), (5, 3), (5, 4), (5, 6), (5, 7), (5, 8), (5, 9), (6, 1), (6, 2), (6, 3), (6, 4), (6, 5), (6, 7), (6, 8), (6, 9), (7, 1), (7, 2), (7, 3), (7, 4), (7, 5), (7, 6), (7, 8), (7, 9), (8, 1), (8, 2), (8, 3), (8, 4), (8, 5), (8, 6), (8, 7), (8, 9), (9, 1), (9, 2), (9, 3), (9, 4), (9, 5), (9, 6), (9, 7), (9, 8)])
```

```
7), (1, 8), (1, 9), (2, 1), (2, 3), (2, 4), (2, 5), (2, 6), (2, 7), (2, 8), (2, 9), (3, 1), (3, 2), (3, 4), (3, 5), (3, 6), (3, 7), (3, 8), (3, 9), (4, 1), (4, 2), (4, 3), (4, 5), (4, 6), (4, 7), (4, 8), (4, 9), (5, 1), (5, 2), (5, 3), (5, 4), (5, 6), (5, 7), (5, 8), (5, 9), (6, 1), (6, 2), (6, 3), (6, 4), (6, 5), (6, 7), (6, 8), (6, 9), (7, 1), (7, 2), (7, 3), (7, 4), (7, 5), (7, 6), (7, 8), (7, 9), (8, 1), (8, 2), (8, 3), (8, 4), (8, 5), (8, 6), (8, 7), (8, 9), (9, 1), (9, 2), (9, 3), (9, 4), (9, 5), (9, 6), (9, 7), (9, 8)]\n
```

```
(1, 8), (1, 9), (2, 1), (2, 3), (2, 4), (2, 5), (2, 6), (2, 7), (2, 8), (2, 9), (3, 1), (3, 2), (3, 4), (3, 5), (3, 6), (3, 7), (3, 8), (3, 9), (4, 1), (4, 2), (4, 3), (4, 5), (4, 6), (4, 7), (4, 8), (4, 9), (5, 1), (5, 2), (5, 3), (5, 4), (5, 6), (5, 7), (5, 8), (5, 9), (6, 1), (6, 2), (6, 3), (6, 4), (6, 5), (6, 7), (6, 8), (6, 9), (7, 1), (7, 2), (7, 3), (7, 4), (7, 5), (7, 6), (7, 8), (7, 9), (8, 1), (8, 2), (8, 3), (8, 4), (8, 5), (8, 6), (8, 7), (8, 9), (9, 1), (9, 2), (9, 3), (9, 4), (9, 5), (9, 6), (9, 7), (9, 8)]
\n
```

Test Case 8	onehop([(1, 2),(2,3),(3, 4),(4,5),(5, 1)])	[(1, 3), (2, 4), (3, 5), (4, 1), (5, 2)]\n	[(1, 3), (2, 4), (3, 5), (4, 1), (5, 2)]\n	Passed
Test Case 9	onehop([(1, 2),(2,3),(3, 4),(4,5),(5, 1),(5,6),(6, 7),(7,8),(8, 9),(9,5)])	[(1, 3), (2, 4), (3, 5), (4, 1), (4, 6), (5, 2), (5, 7), (6, 8), (7, 9), (8, 5), (9, 1), (9, 6)]\n	[(1, 3), (2, 4), (3, 5), (4, 1), (4, 6), (5, 2), (5, 7), (6, 8), (7, 9), (8, 5), (9, 1), (9, 6)]\n	Passed
Test Case 10	onehop([(1, 2),(2,1),(3, 4),(4,3)])	[]\n	[]\n	Passed

Due Date Exceeded.
10 out of 10 tests passed.
You scored 100.0/100.

Your last recorded submission was :

```
def frequency(l):
    new_l = list(set(l))
    freq_list = [l.count(x) for x in new_l]
    min_freq_list = [new_l[x] for x in range(len(freq_list)) if freq_list
    max_freq_list = [new_l[x] for x in range(len(freq_list)) if freq_list
    min_freq_list.sort()
    max_freq_list.sort()
    return (min_freq_list, max_freq_list)

def onehop(l):
    data = l
    data.sort(key=lambda tup: tup[0])
    all_pairs = []
    for e in l:
        x, y = e
        for e1 in l:
            if e != e1:
                xx, yy = e1
                if y == xx and x != yy and (x,yy) not in all_pairs:
                    all_pairs.append((x, yy))
    all_pairs = sorted(all_pairs, key=lambda tup: (tup[0], tup[1]))
    return all_pairs
import ast

def parse(inp):
  inp = ast.literal_eval(inp)
  return (inp)

fncall = input()
lparen = fncall.find("(")
rparen = fncall.rfind(")")
fname = fncall[:lparen]
farg = fncall[lparen+1:rparen]

if fname == "frequency":
  arg = parse(farg)
```

```
  print(frequency(arg))

if fname == "onehop":
  arg = parse(farg)
  print(onehop(arg))
```