CS 101 - PW14 - Fall 23 > aisha.batool@sse.habib.edu.pk

You can view this report online at : https://www.hackerrank.com/x/tests/1752284/candidates/58098272/report

Full Name:	Instructor
Email:	aisha.batool@sse.habib.edu.pk
Test Name:	**CS 101 - PW14 - Fall 23**
Taken On:	18 Nov 2023 01:49:00 PKT
Time Taken:	10 min 56 sec/ 10080 min
Work Experience:	> 5 years
Invited by:	Aisha
Skills Score:	Problem Solving (Basic) 10/10 Problem Solving (Intermediate) 10/10
Tags Score:	CS101 20/20 Dictionaries 10/10 Easy 20/20 Hash Map 20/20 Interviewer Guidelines 20/20 Problem Solving 20/20 Python 10/10 Python 3 10/10 Sorting 10/10 Theme: E-commerce 10/10

100%

60/60

scored in **CS 101 - PW14 - Fall 23** in 10 min 56 sec on 18 Nov 2023 01:49:00 PKT

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	**Say my date, say my date** > **Coding**	51 sec	10/ 10	✓
Q2	**Price Check HU** > **Coding**	1 min 2 sec	10/ 10	✓
Q3	**Merge by Value** > **Coding**	5 min 46 sec	10/ 10	✓
Q4	**Words' biggest bucket** > **Coding**	18 sec	10/ 10	✓
Q5	**Word morphisms** > **Coding**	45 sec	10/ 10	✓
Q6	**Grouping Transactions by Items' Names HU** > **Coding**	54 sec	10/ 10	✓

QUESTION 1

Say my date, say my date > Coding

Correct Answer

Score 10

QUESTION DESCRIPTION

Background

In Python, a dictionary can be used to avoid a sequence of if-elif-else statements.

Challenge

Write a function called `print_dates_in_long_form` that accepts a list of date dictionaries `t` as a parameter, and prints dates in "*month dd*, *yyyy*" format, each date on a separate line. A single date dictionary object contains the keys `'year'`, `'month'`, and `'day'`, with associated numeric values.

Note

Dates should not be checked for validity. Dates should be printed in the same order as in the list. Your code must use the provided dictionary called `month_names` that contains a translation from the month number to the month name.

Sample

```
>>> print_dates_in_long_form([{'day': 12, 'month': 12, 'year': 1996}, {'day':
8, 'month': 12, 'year': 1995}, {'day': 30, 'month': 4, 'year': 1999}, {'day':
30, 'month': 7, 'year': 1998}])
December 12, 1996
December 8, 1995
April 30, 1999
July 30, 1998
```

Input/Output

Input consists of a list literal that HackerRank will read in as `t` and pass to your function.

Constraints

- `t` will contain at least one date.

INTERVIEWER GUIDELINES

```
def date_to_long_form(date):
    return month_names[date['month']] + ' ' + str(date['day']) + ', ' +
str(date['year'])

def print_dates_in_long_form(dates):
    for date in dates:
        print(date_to_long_form(date))
```

CANDIDATE ANSWER

Language used: **Python 3**

```

def date_to_long_form(date):
    return month_names[date['month']] + ' ' + str(date['day']) + ', ' +
str(date['year'])

def print_dates_in_long_form(dates):
    for date in dates:
        print(date_to_long_form(date))
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	1	0.0145 sec	9.39 KB
Testcase 1	Easy	Sample case	✓ Success	1	0.0153 sec	9.33 KB

Testcase 2	Easy	Sample case	✓ Success	1	0.0144 sec	9.55 KB
Testcase 3	Easy	Sample case	✓ Success	1	0.0144 sec	9.41 KB
Testcase 4	Easy	Hidden case	✓ Success	1.5	0.0159 sec	9.4 KB
Testcase 5	Easy	Hidden case	✓ Success	1.5	0.0147 sec	9.38 KB
Testcase 6	Easy	Hidden case	✓ Success	1.5	0.0184 sec	9.64 KB
Testcase 7	Easy	Hidden case	✓ Success	1.5	0.0155 sec	9.6 KB

No Comments

QUESTION 2

✓ Correct Answer

Score 10

Price Check HU › Coding

Easy | Problem Solving | Interviewer Guidelines | Hash Map

QUESTION DESCRIPTION

There is a shop with old-style cash registers. Rather than scanning items and pulling the price from a database, the price of each item is typed in manually. This method sometimes leads to errors. Given a list of items and their correct prices, compare the prices to those entered when each item was sold. Determine the number of errors in selling prices.

Example

products = ['eggs', 'milk', 'cheese']

productPrices = [2.89, 3.29, 5.79]

productSold = ['eggs', 'eggs', 'cheese', 'milk']

soldPrice = [2.89, 2.99, 5.97, 3.29].

```
           Price
Product    Actual   Expected   Error
eggs       2.89     2.89
eggs       2.99     2.89       1
cheese     5.97     5.79       1
milk       3.29     3.29
```

The second sale of eggs has a wrong price, as does the sale of cheese. There are *2* errors in pricing.

Function Description

Complete the function *priceCheck* in the editor below.

priceCheck has the following parameter(s):

string *products[n]:* each *products[i]* is the name of an item for sale

string *productPrices[n]:* each *productPrices[i]* is the price of *products[i]*

string *productSold[m]:* each *productSold[j]* is the name of a product sold

float *soldPrice[m]:* each *soldPrice[j]* contains the sale price recorded for *productSold[j]*.

Returns:

int: the number of sale prices that were entered incorrectly

Constraints

- $1 \le n \le 10^5$
- $1 \le m \le n$
- $1.00 \le productPrices[i], soldPrice[j] \le 100000.00$, where $0 \le i < n$, and $0 \le j < m$

▼ **Input Format for Custom Testing**

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer *n* the size of the *products* array.
The next *n* lines each contain an element *products[i]*.
The next line contains an integer *n,* the size of the *productPrices* array.
The next *n* lines each contain an element *productPrices[i]*.
The next line contains an integer *m*, the size of the *productSold* array.
The next *m* lines each contain an element, *productSold[j]*.
The next line contains an integer, *m*, the size of the *soldPrice* array.
The next *m* lines each contain an element *soldPrice[j]*.

▼ Sample Case 0

Sample Input 0

```
STDIN        Function
-----        --------
4        →   products[] size n = 4
rice     →   products=['rice', 'sugar', 'wheat', 'cheese']
sugar
wheat
cheese
4        →  productPrices[] size n = 4
16.89    →  productPrices=[16.89, 56.92, 20.89, 345.99]
56.92
20.89
345.99
2        →  productSold[] size m = 2
rice     →  productSold =['rice', 'cheese']
cheese
2        →  soldPrice[] size m = 2
18.99    →  soldPrice =[18.99, 400.89]
400.89
```

Sample Output 0

```
2
```

Explanation 0

```
          Price
Product   Actual  Expected  Error
rice       18.99   16.89     1
cheese    400.89  345.99     1
```

The sales of *rice* and *cheese* were at the wrong prices. So, the number of sale prices that were entered incorrectly is 2.

▼ Sample Case 1

Sample Input 1

```
STDIN         Function
-----         --------
3           →  n = 3 .The size of the products array
chocolate   →  products=[chocolate, cheese, tomato]
cheese
tomato
3           →  n = 3 .The size of the productPrices array
15.00       →  productPrices=[15.00, 300.90, 23.44]
300.90
23.44
```

```
3          →  m = 3 .The size of the productSold array
chocolate  →  productSold=[chocolate, cheese, tomato]
cheese
tomato
3          →  m = 3 .The size of the soldPrice array
15.00      →  soldPrice =[15, 300.90,10.00]
300.90
10.00
```

Sample Output 1

```
1
```

Explanation 1

```
            Price
Product     Actual   Expected  Error
chocolate    15.00    15.00
cheese      300.90   300.90
tomato       10.00    23.44    1
```

Only the *tomato* sale does not match the price list. So, the number of sale prices that were entered incorrectly is 1.

INTERVIEWER GUIDELINES

▼ Hint 1

Think of a data structure which can help you efficiently store the prices for each of the product.

Ans - Hash Table

▼ Hint 2

If we store the prices of each of the products in the hash table, think about how we can calculate the number of products sold at a wrong selling price.

▼ Solution

Concepts covered: Basic Programming, Data Structures, Hashing, Loops

The problem tests the candidate's ability to efficiently use the knowledge of the data structures which can store a string, value pair. The candidate is required to come up with a solution to efficiently find the value corresponding to a given string and use conditional operators to check for equality of the value.

Optimal Solution:

Instead of iterating over the products for each of the items sold, we can perform some preprocessing to store the prices of each of the products efficiently. A hash table (C++ map or a Python dictionary) can be used here. After preprocessing, we can retrieve the prices of each of the items in O(1).

Time Complexity - O(N + M), here we have assumed that the string size for the name of the products is small.

```
def priceCheck(products, productPrices, productSold, soldPrice):
    # create a hash map of tuples, key = product, value = correct price
    prices = dict((prod, prod_price) for prod, prod_price in
zip(products, productPrices))

    ans = 0
    # iterate the array of sales comparing prices with hash map
    for prod, sell_price in zip(productSold, soldPrice):
        if prices[prod] != sell_price:
            ans += 1
    return ans
```

Brute Force Approach: We can iterate over the sold products and for each of the products, we iterate over the list of products to find its actual selling price. If the selling price differs from the sold price, we increment the answer.

```
def priceCheck(products, productPrices, productSold, soldPrice):

    ans = 0
    for prod, pr in zip(productSold, soldPrice):
        idx = products.index(prod)
        if pr != productPrices[idx]:
            ans += 1
    return ans
```

Time Complexity - O(N x M), where N is the number of products and M is the number of items sold.

Error Handling:

1. If the implemented hash table in the respective languages is not used, an efficient hashing function should be employed. A Rabin Karp hash is usually successful in such cases.
2. Some languages such as C++ and python, sometimes give precision errors while comparing floating-point values. In such cases, it is advisable to store them as a string and compare the strings. Although the test set contains precision only up to 2 decimal places, hence the precision errors are less prone to occur.

▼ Complexity Analysis

Time Complexity - O(N + M), We iterate over the products to efficiently store the prices in a hash table in O(N) time. We then iterate over the products and retrieve the actual selling price. Assuming the retrieval is O(1) for the hash table, we perform O(M) more operations. Hence, the total time is O(N + M).

Space Complexity - O(N)
The total space taken by the hash table for storing the products is of the order of the number of products i.e. O(N).

CANDIDATE ANSWER

Language used: **Python 3**

```

def priceCheck(products, productPrices, productSold, soldPrice):

    ans = 0
    for prod, pr in zip(productSold, soldPrice):
        idx = products.index(prod)
        if pr != productPrices[idx]:
            ans += 1
    return ans

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Medium	Sample case	✓ Success	1	0.0205 sec	10.8 KB
TestCase 1	Easy	Sample case	✓ Success	1	0.0259 sec	10.6 KB
TestCase 2	Easy	Sample case	✓ Success	1	0.0228 sec	10.7 KB
TestCase 3	Easy	Sample case	✓ Success	1	0.0263 sec	10.7 KB
TestCase 4	Medium	Sample case	✓ Success	1	0.021 sec	10.5 KB
TestCase 5	Medium	Hidden case	✓ Success	1	0.023 sec	10.9 KB

TestCase 6	Medium	Hidden case	✓ Success	1	0.0298 sec	11.2 KB
TestCase 7	Medium	Hidden case	✓ Success	1	0.0534 sec	11.9 KB
TestCase 8	Hard	Hidden case	✓ Success	1	0.9263 sec	24.9 KB
TestCase 9	Hard	Hidden case	✓ Success	1	1.6562 sec	32.6 KB

No Comments

QUESTION 3

✓

Correct Answer

Score 10

Merge by Value › Coding

QUESTION DESCRIPTION

Problem

Write a function named `merge_value` that takes two dictionaries `d1` and `d2` as parameters and builds a dictionary that contains every value from `d1` and `d2` as key. The corresponding key in `d1` and `d2` becomes the value in the merged dictionary. For multiple values in the merged dictionary for the same key, the values are put in a list. The function *returns* a sorted list of the (key, value) pairs in the merged dictionary.

Sample

```
>>> d2 = {i:chr(64+i) for i in range(1,11)}
>>> d2
{1: 'A', 2: 'B', 3: 'C', 4: 'D', 5: 'E', 6: 'F', 7: 'G', 8: 'H', 9: 'I',
10: 'J'}
>>> d3 = {i-1:chr(64+i) for i in range(1,11)}
>>> d3
{0: 'A', 1: 'B', 2: 'C', 3: 'D', 4: 'E', 5: 'F', 6: 'G', 7: 'H', 8: 'I',
9: 'J'}
>>> merge_val(d2,d3)
{'B': [2, 1], 'C': [3, 2], 'D': [4, 3], 'I': [9, 8], 'A': [1, 0], 'G': [7,
6], 'E': [5, 4], 'F': [6, 5], 'J': [10, 9], 'H': [8, 7]}
```

Input Format

The input contains `d1` and `d2` on separate lines.

Output Format

The output should be a sorted list of the (key, value) pairs in the merged dictionary.

INTERVIEWER GUIDELINES

Solution

```
def merge_value(d1, d2):
    d = {}
    # Iterate over the items (k, v) of d1 and d2. Insert every newly
encountered
    # v into a new dictionary as a key with [k] as the value. If v is
    # encountered again, store the corresponing key in the new dictionary
in the
    # previously created list.
    for k,v in list(d1.items()) + list(d2.items()):
        # dict.get() eliminates the need for if-else
        d[v] = d.get(v,[]) + [k]
    return d
```

CANDIDATE ANSWER

Language used: **Python 3**

```
def merge_value(d1, d2):
    d = {}
    # Iterate over the items (k, v) of d1 and d2. Insert every newly
encountered
    # v into a new dictionary as a key with [k] as the value. If v is
    # encountered again, store the corresponing key in the new dictionary in
the
    # previously created list.
    for k,v in list(d1.items()) + list(d2.items()):
        # dict.get() eliminates the need for if-else
        d[v] = d.get(v,[]) + [k]
    return d
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	2	0.0233 sec	9.45 KB
Testcase 1	Easy	Hidden case	✓ Success	2	0.0148 sec	9.25 KB
Testcase 2	Easy	Hidden case	✓ Success	2	0.0153 sec	9.3 KB
Testcase 3	Easy	Hidden case	✓ Success	2	0.0179 sec	9.48 KB
Testcase 4	Easy	Hidden case	✓ Success	2	0.0155 sec	9.42 KB

No Comments

QUESTION 4

✓

Correct Answer

Score 10

Words' biggest bucket › Coding

CS101 Python Dictionaries

QUESTION DESCRIPTION

As a budding cryptanalyst (someone who creates and cracks security codes), Carol is well aware of the fact that the most frequently occurring letters in the English language corpus (body of text) are "e," "t," and "a," in that order. A colleague of hers, Bob, has wagered that the most frequently occurring letters that begin a word must also follow the same trend. Carol, of course, would like nothing better than to prove him wrong.

Carol needs your help. To research this claim, she would like you to write a program for her that takes a corpus as input, and outputs the most frequently occurring letter that begins a word (mode), along with the number of occurrences (frequency). If there is a tie between two or more letters (more than one mode), the program should output all the letters (sorted in alphabetical order) and their frequencies as a list of nested lists.

For example, if the input text is, "She sells 'seashells' by the seashore," then the output is a singleton list [['s', 4]], since the letter S begins four words in the sentence. If the input text is, "A noisy noise annoys only a **nosy** oyster," then the output is [['a', 3], ['n', 3]], since both the letters A and N start a word in the sentence three times each. Note that words inside the sentence may begin with punctuation marks. Words will be separated by at least one space character.

Function Description

Complete the function *biggest_bucket*. The function must return a list of nested lists, where each nested list contains a single-character string and an integer.

biggest_bucket has the following parameter:

corpus: a string

Note

- Input and output will be handled by HackerRank--do not take values as input or print values yourself!

Constraints

- String *corpus* is non-empty.
- Words in *corpus* will be separated by at least one space character in between them.

▼ Input Format For Custom Testing

The first (and only) line contains a string denoting the corpus.

▼ Sample Case 0

Sample Input For Custom Testing

```
"Well," said Pooh, "what I like best," and then he had to stop and think.
Because although Eating Honey was a very good thing to do, there was a
moment just before you began to eat it which was better than when you
were, but he didn't know what it was called."
```

Sample Output

```
[['w', 10]]
```

Explanation

Ten words begin with the letter W in the given text, which is the most frequent of all the letters in the alphabet. Note that the first word "Well" begins with a quotation mark, and is counted as a word that begins with the letter W.

▼ Sample Case 1

Sample Input For Custom Testing

```
How much Hungarian wood could a hungry woodchuck chuck if a woodchuck
could chuck Hungarian wood?
```

Sample Output

```
[['c', 4], ['h', 4], ['w', 4]]
```

Explanation

There is a three-way tie between the letters C, H, and W--each letters begins four words in the given sentence. The letters are sorted in alphabetical order when returning the list of nested lists.

INTERVIEWER GUIDELINES

```
def biggest_bucket(corpus):
  '''Return a list of most frequently occurring letters that begin a
word
  (mode) and their number of occurrences (frequencies) as nested
lists.'''
  buckets, mode, mode_freq = {}, [], 0
  for word in corpus.lower().split():
    for letter in word:
      if letter.isalpha():
        buckets[letter] = buckets.get(letter, 0) + 1
        if buckets[letter] > mode_freq:
          mode_freq = buckets[letter]
          mode = [[letter, buckets[letter]]]
        elif buckets[letter] == mode_freq:
          mode.append([letter, buckets[letter]])
```

```
                break
        return sorted(mode)
```

CANDIDATE ANSWER

Language used: **Python 3**

```
def biggest_bucket(corpus):
    '''Return a list of most frequently occurring letters that begin a word
    (mode) and their number of occurrences (frequencies) as nested lists.'''
    buckets, mode, mode_freq = {}, [], 0
    for word in corpus.lower().split():
        for letter in word:
            if letter.isalpha():
                buckets[letter] = buckets.get(letter, 0) + 1
                if buckets[letter] > mode_freq:
                    mode_freq = buckets[letter]
                    mode = [[letter, buckets[letter]]]
                elif buckets[letter] == mode_freq:
                    mode.append([letter, buckets[letter]])
                break
    return sorted(mode)

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	1	0.0164 sec	9.65 KB
Testcase 1	Easy	Sample case	✓ Success	1	0.0161 sec	9.39 KB
Testcase 2	Easy	Sample case	✓ Success	1	0.0142 sec	9.61 KB
Testcase 3	Easy	Sample case	✓ Success	1	0.0162 sec	9.45 KB
Testcase 4	Easy	Sample case	✓ Success	1	0.0862 sec	9.64 KB
Testcase 5	Easy	Hidden case	✓ Success	1	0.0145 sec	9.35 KB
Testcase 6	Easy	Hidden case	✓ Success	1	0.0159 sec	9.44 KB
Testcase 7	Easy	Hidden case	✓ Success	1	0.0158 sec	9.35 KB
Testcase 8	Easy	Hidden case	✓ Success	1	0.0144 sec	9.3 KB
Testcase 9	Easy	Hidden case	✓ Success	1	0.0151 sec	9.52 KB

No Comments

QUESTION 5

✓

Correct Answer

Score 10

Word morphisms › Coding CS101 Python 3

QUESTION DESCRIPTION

Background

Many words share a common pattern of letter arrangement. For example, both the words APPLE and GOOEY have letters in the pattern 1-2-2-3-4. Carefully mapping each letter in one word (APPLE) to a suitable letter (A -> G, P -> O, L -> E, E -> Y) will give us the second word (GOOEY). See more examples below.

Task

Given a list of mappings, translate a given word. Each mapping is a nested list containing a pair of numbers. Each number is an ordinal for a letter. The first number is the ordinal of the letter to translate from. The second number is the ordinal of the letter to translate to.

Function Description

To implement the given task, write the following function:

1. ***substitute*** that takes two arguments: ***subs***, ***word***.The function should **return** a string that contains the translated ***word***, a string. ***subs*** is a list of nested lists, each of which contains a pair of integers.

Constraints

- Input and output will be handled by HackerRank--you should not read input or display values yourself.
- Each integer in the mapping (nested list) is a valid ordinal.
- Mapping for all letters will be provided.

▼ Input Format For Custom Testing

The first line contains *subs*, a list of nested lists, each containing a pair of integers.

The second line contains *word*, a string.

▼ Sample Case 0

Sample Input For Custom Testing

```
[[97, 103], [112, 111], [108, 101], [101, 121]]
apple
```

Sample Output

```
gooey
```

Explanation

Letter 'a' corresponds to ordinal 97, which is translated to 'g' with an ordinal of 103.

Letter 'p' corresponds to ordinal 112, which is translated to 'o' with an ordinal of 111.

Letter 'p' is translated to 'o' as above.

Letter 'l' corresponds to ordinal 108, which is translated to 'e' with an ordinal of 101.

Letter 'e' is translated to 'y' with an ordinal of 121.

Word 'gooey' is returned.

▼ Sample Case 1

Sample Input For Custom Testing

```
[[102, 112], [97, 108], [99, 97], [101, 121], [98, 114], [111, 111], [107,
109]]
facebook
```

Sample Output

```
playroom
```

Explanation

Letter 'f' corresponds to ordinal 102, which is translated to 'p' with an ordinal of 112.

Letter 'a' corresponds to ordinal 97, which is translated to 'l' with an ordinal of 108.

Letter 'c' corresponds to ordinal 99, which is translated to 'a' with an ordinal of 97.

Letter 'e' corresponds to ordinal 101, which is translated to 'y' with an ordinal of 121.

Letter 'b' corresponds to ordinal 98, which is translated to 'r' with an ordinal of 114.

Letter 'o' corresponds to ordinal 111, which is translated to 'o' with the same ordinal.

Letter 'o' is translated to 'o' as above.

Letter 'k' corresponds to ordinal 107, which is translated to 'm' with an ordinal of 109.

Word 'playroom' is returned.

INTERVIEWER GUIDELINES

```
def substitute(subs, word):
    d = {}
    for sub in subs:
        key = sub[0]
        value = chr(sub[1])
        d[key] = value
    result = ''
    for letter in word:
        ordinal = ord(letter)
        result += d[ordinal]
    return result
```

CANDIDATE ANSWER

Language used: **Python 3**

```
def substitute(subs, word):
    d = {}
    for sub in subs:
        key = sub[0]
        value = chr(sub[1])
        d[key] = value
    result = ''
    for letter in word:
        ordinal = ord(letter)
        result += d[ordinal]
    return result

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	✓ Success	1	0.0328 sec	10.1 KB
TestCase 1	Easy	Sample case	✓ Success	1	0.0194 sec	10.3 KB
TestCase 2	Easy	Sample case	✓ Success	1	0.0226 sec	10.3 KB
TestCase 3	Easy	Sample case	✓ Success	1	0.0291 sec	10 KB
TestCase 4	Easy	Sample case	✓ Success	1	0.0219 sec	10.1 KB
TestCase 5	Easy	Hidden case	✓ Success	1	0.022 sec	10 KB
TestCase 6	Easy	Hidden case	✓ Success	1	0.0223 sec	10.1 KB
TestCase 7	Easy	Hidden case	✓ Success	1	0.021 sec	9.86 KB
TestCase 8	Easy	Hidden case	✓ Success	1	0.0223 sec	10.3 KB
TestCase 9	Easy	Hidden case	✓ Success	1	0.0238 sec	10.3 KB

No Comments

QUESTION 6

✓

Correct Answer

Grouping Transactions by Items' Names HU › Coding

Easy | Problem Solving

Theme: E-commerce | Interviewer Guidelines | Sorting | Hash Map

Score 10

QUESTION DESCRIPTION

For a given array of transactions, group all of the transactions by item name. Return an array of strings where each string contains the item name followed by a space and the number of associated transactions.

Note: Sort the array descending by transaction count, then ascending alphabetically by item name for items with matching transaction counts.

Example

transactions = ['notebook', 'notebook', 'mouse', 'keyboard', 'mouse']

There are two items with *2* transactions each: *'notebook'* and *'mouse'.* In alphabetical order, they are *'mouse', 'notebook'.*

There is one item with *1* transaction: '*keyboard'.*

The return array, sorted as required, is *['mouse 2', 'notebook 2', 'keyboard 1'].*

Function Description

Complete the function *groupTransactions* in the editor below.

groupTransactions has the following parameter(s):

string transactions[n]: each *transactions[i]* denotes the item name in the i^{th} transaction

Returns:

string[]: an array of strings of "item name[space]transaction count" sorted as described

Constraints

- $1 \le n \le 10^5$
- 1 ≤ length of *transactions[i]* ≤ 10
- *transactions[i]* contains only lowercase English letters, ascii[a-z]

▼ Input Format Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains a single integer, *n*, the size of *transactions*.

Each of the next *n* lines contains a string, the item name for *transactions[i]*.

▼ Sample Case 0

Sample Input

```
STDIN     Function
-----     -----
4      →  transactions[] size n = 4
bin    →  transactions = ['bin', 'can', 'bin', 'bin']
can
bin
bin
```

Sample Output

```
bin 3
can 1
```

Explanation

- There is one item '*bin*' with *3* transactions.
- There is one item '*can*' with *1* transaction.
- The return array sorted descending by transaction count, then ascending by name is *['bin 3', 'can 1'].*

▼ Sample Case 1Sample Input

Sample Input

```
STDIN     Function
-----     -----
3      →  transactions[] size n = 3
banana →  transactions = ['banana', 'pear', 'apple']
pear
apple
```

Sample Output

```
apple 1
banana 1
pear 1
```

Explanation

- There is one item '*apple*' with *1* transaction.
- There is one item *'banana'* with *1* transaction.
- There is one item '*pear*' with *1* transaction.
- The return array sorted descending by transaction count, then ascending by name is *['apple 1', 'banana 1', 'pear 1']*.

INTERVIEWER GUIDELINES

▼ Hint 1

Think of a data structure which can help you efficiently store and retrieve the prices for each of the products.

Ans - hash table

▼ Hint 2

If we store the prices of each of the products in a hash table, think about how we can calculate the number of products.

▼ Solution

Concepts covered: Basic Programming, Data Structures, Hashing, Sorting, Loops

The problem tests the candidate's ability to efficiently use the knowledge of the data structures which can store a string, value pair. The candidate is required to come up with a solution to efficiently find the value corresponding to a given string. The candidate also needs to store the count corresponding to each of the products efficiently as a tuple, then sort them as asked described.

Optimal Solution:

Instead of iterating over the products for each of the sold items, we can use a data structure to store the frequency of each of the products efficiently. A hash table (C++ map or a Python dictionary) can be used here. We can iterate over the array and increment the count of each of the products while iterating. We can use an inbuilt sort function provided in most of the languages or alternatively implement an efficient sorting algorithm such as merge sort or quicksort.

Time Complexity - O(N LogN), here we have assumed that the string size for the name of the products is small.

```
def groupTransactions(transactions):
    # Write your code here
    count = dict()
    # create a hash map, key = product, value = count
    for prod in transactions:
        if prod not in count:
            count[prod] = 0
            count[prod] += 1
    # perform a 2 way sort, descending by count, increasing by product
name
    ans = sorted(count.items(), key=lambda x: (-x[1],x[0]))
    ans = [str(i) + " " + str(j) for i, j in ans]
    return ans
```

Brute Force Approach: We create a set of products sold. For each of the products, we iterate over the array to count its frequency. We store this in the array and can use Bubble sort to sort the array. Time Complexity - O(N^2), where N is the number of products sold.

```
def groupTransactions(transactions):
    # Write your code here
    products = set(transactions)

    ans = []
    for prod in products:
        ans.append((prod, transactions.count(prod)))

    ans.sort(key=lambda x:(-x[1],x[0]))
    return [str(i) + " " + str(j) for i, j in ans]
```

Error Handling:

1. If the implemented hash table in the respective languages is not used, an efficient hashing function should be employed. A Rabin Karp hash is usually successful in such cases.

2. While sorting, it is very important to notice that the count is sorted in decreasing order while the names are sorted in increasing order. One efficient way to handle it is to sort the count considering the negatives of the count as the key.

▼ Complexity Analysis

Time Complexity - O(N logN), We iterate over the products to efficiently store the count in a hash table which takes O(N) time. We then sort the array of tuples which stores the pair of products and count. In the worst-case scenario, the sorting functions usually take O(N logN) time. Here, we have assumed the lengths of strings are small.

Space Complexity - O(N)

The total space taken by the hash table for storing the products is of the order of the number of products i.e. O(N).

CANDIDATE ANSWER

Language used: **Python 3**

```

def groupTransactions(transactions):
    # Write your code here
    count = dict()
    # create a hash map, key = product, value = count
    for prod in transactions:
        if prod not in count:
            count[prod] = 0
        count[prod] += 1
    # perform a 2 way sort, descending by count, increasing by product name
    ans = sorted(count.items(), key=lambda x: (-x[1],x[0]))
    ans = [str(i) + " " + str(j) for i, j in ans]
    return ans

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	✓ Success	0.5	0.0192 sec	10.7 KB
TestCase 1	Easy	Sample case	✓ Success	0.5	0.0846 sec	10.9 KB
TestCase 2	Easy	Sample case	✓ Success	0.5	0.0265 sec	10.7 KB
TestCase 3	Easy	Sample case	✓ Success	0.5	0.1463 sec	10.7 KB
TestCase 4	Easy	Sample case	✓ Success	0.5	0.0836 sec	10.9 KB
TestCase 5	Easy	Hidden case	✓ Success	0.5	0.0227 sec	10.9 KB
TestCase 6	Easy	Hidden case	✓ Success	1	0.019 sec	10.8 KB
TestCase 7	Easy	Hidden case	✓ Success	1	0.4709 sec	36.3 KB
TestCase 8	Easy	Hidden case	✓ Success	1	0.2571 sec	36.2 KB
TestCase 9	Easy	Hidden case	✓ Success	1	0.2388 sec	36.4 KB
TestCase 10	Easy	Hidden case	✓ Success	1	0.2455 sec	36.2 KB
TestCase 11	Easy	Hidden case	✓ Success	1	0.292 sec	36.2 KB
TestCase 12	Easy	Hidden case	✓ Success	1	0.3251 sec	36.1 KB

No Comments

PDF generated at: 17 Nov 2023 21:01:56 UTC