DA 203: Introduction to computing for artificial intelligence and machine learning 2:1

[1]

Instructions

1. `Published Date: 08-Feb-2022`
2. `Submission Date: 25-Feb-2022`
3. Codes should be original and should not be copied.
4. Plagiarised reports/codes results in zero mark.

1 Problem 1 - Count in Array [20 Marks]

Mark Split up

- Codes - 10 marks
- PDF - 10 marks

Primary Objective :

You are given an integer list with n numbers. You are supposed to return a list of integers which occur more than or equal to [10 Marks]x times in the given array from the function `counter(n,x)`.

For eg :

```
>>> a = [1,2,2,3,3,3,4,4,4,4,5,5,5,5,5,5]
>>> k = counter(a,3);
>>> print(k)
 [3,4,5]

>>> a = [1,2,3,5,4,4,5,5,4,5,2,3,3,5,5]
>>> k = counter(a,2);
>>> print(k)
 [2,3,4,5]
```

Experiemental parameters :

n = 1 to 1000000 x = 1 to 1000

Room for Improvement

The general order of complexity(time) of this algorithm is in the order of O(N2). Now we need to optimise this algorithm such that it runs in O(N) complexity (time).

Note: this algorithm can be improved little bit to O(NlogN) complexity by sorting the array first and identifying the duplicates. however that is not the scope of this problem for this problem set.

[1] Problem set 2

Deliverables in Moodle :

- Complete the `counter()` function which performs the above operation in O(N2)
- Complete the `counter_optimised()` function which performs the above operation in O(N) time.

Deliverables in PDF :

Write a code which performs the experiments to compute the Big'O complexity of both the experiments experimentally

1. Write a short note on how you were able to achieve O(N) complexity for the given problem.
2. Plot the size of array (n) vs timeTaken(t) for the `counter()` function and plot an asymptotical line which denotes the theoritical limit for O(N2).
3. Identify the C value that was used to construct the asymptotical line in the previous graph.
4. Plot the log-log plot of size vs time (log(size) and log(time)) plot and identify the slope of the line.plot also the asymptodical limit in the same plot in log scale. what is the value of slope of the line ?
5. Plot the size of array (n) vs timeTaken(t) for the `counter_optimised()` function and plot an asymptotical line which denotes the theoritical limit for O(N2).
6. Identify the C value that was used to construct the asymptotical line in the previous graph.
7. Plot the log-log plot of size vs time (log(size) and log(time)) plot and identify the slope of the line. plot also the asymptodical limit in the same plot in log scale. what is the value of slope of the line ?

Note : The Array can be sorted first O(NlogN) and then the duplicates can be identified in O(N), which makes that overall algorithm to have O(NlogN) complexity. However that intermediate improvement need not be considered for this assignment.

Experimental Parameters : For computing the order of complexity of the algorithms theoritically follow the below conventions.

- For each size , perform 100 operations and identify the time taken for that size to be the average of those 100 iterations
- The 'x' value in should be differernt for each iteration.
- The array of size n should be kept constant across all iterations.

Submission Instructions

- The codes for the functions have to be submitted in the moodle. (for identifying the correctness of these functions)
- The reports and images (for the 7 questions) have to be submitted as a PDF.

Hint for O(N): { } in python.

2 Problem-2 - Budget App [20 marks]

Mark Split up

- Codes - 20 marks

Complete the `Category` class in `budget.py`. It should be able to instantiate objects based on different budget categories like *food*, *clothing*, and *entertainment*. When objects are created, they are passed in the name of the category. The class should have an instance variable called `ledger` that is a list. The class should also contain the following methods:

- A `deposit` method that accepts an amount and description. If no description is given, it should default to an empty string. The method should append an object to the ledger list in the form of {`"amount": amount, "description": description`}.
- A `withdraw` method that is similar to the `deposit` method, but the amount passed in should be stored in the ledger as a negative number. If there are not enough funds, nothing should be added to the ledger. This method should return `True` if the withdrawal took place, and `False` otherwise.
- A `get_balance` method that returns the current balance of the budget category based on the deposits and withdrawals that have occurred.
- A `transfer` method that accepts an amount and another budget category as arguments. The method should add a withdrawal with the amount and the description "Transfer to [Destination Budget Category]". The method should then add a deposit to the other budget category with the amount and the description "Transfer from [Source Budget Category]". If there are not enough funds, nothing should be added to either ledgers. This method should return `True` if the transfer took place, and `False` otherwise.
- A `check_funds` method that accepts an amount as an argument. It returns `False` if the amount is greater than the balance of the budget category and returns `True` otherwise. This method should be used by both the `withdraw` method and `transfer` method.

When the budget object is printed it should display: * A title line of 30 characters where the name of the category is centered in a line of `*` characters. * A list of the items in the ledger. Each line should show the description and amount. The first 23 characters of the description should be displayed, then the amount. The amount should be right aligned, contain two decimal places, and display a maximum of 7 characters. * A line displaying the category total.

Here is an example of the output:

```
*************Food*************
initial deposit        1000.00
groceries               -10.15
restaurant and more foo -15.89
Transfer to Clothing    -50.00
Total: 923.96
```

Besides the `Category` class, create a function (outside of the class) called `create_spend_chart` that takes a list of categories as an argument. It should return a string that is a bar chart.

The chart should show the percentage spent in each category passed in to the function. The percentage spent should be calculated only with withdrawals and not with deposits. Down the left side of the chart should be labels 0 - 100. The "bars" in the bar chart should be made out of the "o" character. The height of each bar should be rounded down to the nearest 10. The horizontal line below the bars should go two spaces past the final bar. Each category name should be written vertically below the bar. There should be a title at the top that says "Percentage spent by category".

This function will be tested with up to four categories.

Look at the example output below very closely and make sure the spacing of the output matches the example exactly.

```
Percentage spent by category
100|
 90|
 80|
 70|
 60| o
 50| o
 40| o
 30| o
 20| o  o
 10| o  o  o
  0| o  o  o
    ----------
     F  C  A
     o  l  u
     o  o  t
     d  t  o
        h
        i
        n
        g
```

For eg:

The following code code should provide the output as shown below

```
class Category:
    ##TO DO

def create_spend_chart(categories):
    ##TO DO

food = Category("Food")
food.deposit(1000, "initial deposit")
food.withdraw(10.15, "groceries")
food.withdraw(15.89, "restaurant and more food for dessert")
print(food.get_balance())
clothing = Category("Clothing")
food.transfer(50, clothing)
clothing.withdraw(25.55)
clothing.withdraw(100)
```

```
auto = Category("Auto")
auto.deposit(1000, "initial deposit")
auto.withdraw(15)

print(food)
print(clothing)

print(create_spend_chart([food, clothing, auto]))
```

output

```
973.96
*************Food*************
initial deposit        1000.00
groceries               -10.15
restaurant and more foo -15.89
Transfer to Clothing    -50.00
Total: 923.96
***********Clothing***********
Transfer from Food       50.00
                        -25.55
Total: 24.45
Percentage spent by category
100|
 90|
 80|
 70|
 60| o
 50| o
 40| o
 30| o
 20| o  o
 10| o  o  o
  0| o  o  o
    ----------
     F  C  A
     o  l  u
     o  o  t
     d  t  o
        h
        i
        n
        g
```

Note : The output should be concatenated (along with newline char) and returned as a single string value to be printed. Do not terminate the last line of output string with a newline charecter. Here, The new line at end of every output is due to the newline added due to the `print()` function.

Submission Instructions

- The codes for the functions have to be submitted in the moodle. All test cases will be evaluated on moodle.

- No need for PDF report for this problem.

3 Problem 3 - Data Visualisation [10 Marks]

You have been provided with dataset of imports and exports made by India from 2010 to 2018.

Data set Description

The dataset has following columns

- HS Code - Code used by Exports for identifying a type of product
- Commodity - Description of the commodity being exported
- Value - Value of exported commodity in millions
- Country - Country to which the export/import happened
- Year

use pandas for reading and storing data.

Visualisation

Use the above dataset to visualise the following scenarios using matplotlib/seaborn in python.

1. A multiple line chart with the total export/import value across years from 2010 to 2018. (x - years, y- total value of export and import)
2. A multiple barchart (stacked next to each other) with the top three export category which was exported highest in a given year. [x - years , y - 3 bars with total export value of those categories]
3. A multiple barchart (stacked next to each other) with the top three import countries which we imported highest value from in a given year. [x - years , y - 3 bars with total import value of those categories].
4. A pie chart of average export value across all years per each category (only top 6 , include all the remaining in "others" category)
5. A simple forecast of the total export and import value during the year 2019 and 2020