Home Work № 10

CSCI B505 Applied Algorithms 11/21/2022

Instructions

1. Under any circumstances the deadline will NOT be extended.
2. Code should be written in python language in a Jupyter notebook .
3. Unless mentioned in the questions, feel free to use built-in python functions
4. Upload Jupyter notebook and html file of the Jupyter notebook in canvas. No other forms submissions is considered as valid submission. Make sure to have the output of the required cells of the jupyter notebook and its html version before making submission.
5. Plagiarism is unacceptable and we have ways to find it. So do not do it.
6. There are four problems. Each problem is worth 25 points.
7. The code should readable with variables named meaningfully.
8. Write test cases wherever required so that they cover all scenarios.
9. students are expected to use the function signature given in the question or they'll get a penalty.

Problem 1

Alice is learning a new language Y(not programming language). She seeks help from her friend Bob who is an expert in that language. She started reading books which were written in language Y and when ever she encounters a word whose meaning she is not familiar with, she asks the synonym of the word from Bob and Bob responds with synonyms of the words. Alice notes down them in her notes. As this process goes , Alice noted down large list of synonyms in her notes. As she is going through the notes , she made a glaring observation that she can combine some of the synonyms. Let suppose if **word1 word2 word3** are synonyms and **word2 word5** are synonyms then she can combine them into a single list **word1 word2 word3 word5**

```
Example:

    syn_list = [['oranges','dogs', 'apples'],['peach', 'mango'],['dogs', 'cats']]
    output = [['oranges','dogs', 'apples','cats'],['peach','mango']]
    Note that ordering of the output is not important
```

Following the below given method signature

```

def get_combined_list(syn_list):
  ## your code goes here
  ## return combined list
  pass
```

Problem 2

Two pairs (a, b) and (c, d) are said to be symmetric if c is equal to b and a is equal to d. For example, (10, 20) and (20, 10) are symmetric. Given an array of pairs find all symmetric pairs in it.

Note: Solve using Hashing and return only one of the symmetric pair in output

```
Example 1:
    Input:
        Array_pairs = [[11, 20], [30, 40], [5, 10], [40, 30], [10, 5]]
    Output:
        [[30,40],[10,5]]

Example 2:
    Input:
        Array_pairs = [[11, 20], [40, 30], [10, 5]]
    Output:
        []
```

Use following function signature:

```

def SymmetricPairs(Array_Pairs:list[list]):
  ### Logic ###
  ## return list of symmetric pairs
  pass
```

Problem 3

There exists a custom dictionary that supports the following operations:

a) **Add(key_k,val_v):** If no key exists in the dictionary with value key_k, add the (key_k,value_v) pair to the dictionary. Else update the value corresponding to the key key_k to value_v

b) **Add_to_keys(diff):** Adds diff to every key in the dictionary at that point of time.

c) **Add_to_vals(diff):** Adds diff to every value in the dictionary at that point of time.

d) **Return(key_k):** Returns the value corresponding to the key key_k from the dictionary.

Now the custom dictionary is initially empty and you are given two lists queries and values of equal length n.
Every element in queries belongs to the set of strings {‘Add’,’Add_to_keys’,’Add_to_vals’,’Return’}.
For 0<=i<n, if queries[i]==‘Add’ values[i] is a list of two integers key_k as values[i][0] and val_v as values[i][1], else if queries[i]!=‘Add’ values[i] is a list of 1 integer.
You have to traverse the queries and values list from 0 to n-1 and address each index by performing the operation at queries[i] with parameters as values[i].
If there are m elements with value ‘Return’ in list queries, your task is to return a list ’result’ of length m where the first element is the returned value of the operation ’d’(given above) when it is called by list queries for the first time and the second element is the returned value of the operation ’d’ when it is called by list queries for the second time and so on.

Now each type of query is supposed to be addressed in O(1) time(not amortized) complexity. Given the task is very brute and requires very less logic if the time constraint is compromised, 40% of the score would be awarded for the problem if any of the functions do not meet the time complexity requirements.

```
Example:
queries=  ['Add','Add_to_vals','Return','Add','Add_to_keys','Add_to_vals','Return']

values= [ [1,2],[2],[1],[2,3],[1],[-1],[3] ]

Output:
result=[4,2]
```

```python
def custom_dict(queries,values):
  ### Logic ###
  ## return result list
  pass
```

Problem 4

Erin is a wanderlust, and she has a list of cities that she wants to visit. She would like to visit some cities before others. For example, she wants to visit London before Medellín, Medellín before São Paulo, Prague before Berlin, and so on. She has a huge list of such priorities. Help Erin form a travel plan, so she can visit the cities in order. A travel plan is a list of cities such that if Erin prioritizes city A over city B, city A has to occur earlier than city B in that list.

Write a function *get_travel_plan* that takes a list of cities and a list of priorities as input and returns another list which is a travel plan for Erin.

Note: priorities is a list of tuples (A, B) such that Erin prioritizes city A over city B.

```
## This is how the usage looks like

Example 1:

cities = ['London', 'Berlin', 'Medellín', 'São Paulo', 'Prague', 'Ladakh', 'Nice']

priorities = [('London', 'Medellín'), ('Medellín', 'São Paulo'), ('Prague', 'Berlin')]

when invoked get_travel_plan(cities, priorities) will give ouput:

['Nice', 'London', 'Medellín', 'Prague', 'São Paulo', 'Berlin', 'Ladakh']

Example 2:

cities = ['New York', 'Honolulu']

priorities = [('New York', 'Honolulu'), ('Honolulu', 'New York')]

when invoked get_travel_plan(cities, priorities) will give ouput:
[]

These priorities cannot be turned into a travel plan!
```

use the following function signature:

```python

def get_travel_plan(cities, priorities):
  pass
```