Home Work № 3

CSCI B505 Applied Algorithms 09/16/2022

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. For Problems 2, 5 and 6 , use pen and paper and upload the same as pdf.
7. There are six problems. Problem 3 - 25 points, rest of the problems 15 points each.
8. The code should readable with variables named meaningfully
9. Write test cases wherever required so that they cover all scenarios.

Problem 1

In Unix, you can execute a command in two ways:

- Entering its name, e.g., “cp” or “ls”.
- Entering “! <index>”. This notation is used to repeat the indexth (1-based) command since the start of the session. For example, suppose that the user has entered the following commands:

ls, cp, mv, mv, mv, !1, !3, !6.

“!1” would trigger the execution of “ls”, “!3” would repeat “mv” and “!6” would execute “!1” which in turn would trigger the execution of “ls”.

You are provided with a sequence of commands that the user has entered in the terminal since the start of the session. Each command can be one of the following: “cp”, “ls”, “mv” or “! <index>”. Calculate the number of times each of “cp”, “ls” and “mv” commands are executed

and return the array of three integers in the following form: [# of times for “cp”, # of times for “ls”, # of times for “mv”].

Example:

commands = [“ls”, “cp”, “mv”, “mv”, “mv”, “!1”, “!3”, “!6”], the output should be [1, 3, 4].

- First “ls” was executed once.
- Then “cp” was executed once.
- After that “mv” was executed three times.
- Then “!1” was executed, triggering the execution of commands [0] = “ls”.
- Then “!3” was executed, triggering the execution of commands [2] = “mv”.
- Finally, “!6” was executed, triggering commands [5] = “!1”, which in turn would trigger commands [0] = “ls”

Constraints:

1 <= commands. length <= 500

```
def commands_count (List[str]):
    ###############
         logic
    ###############
    return list[integers]
```

Problem 2

Find the Time Complexities of the following functions:

```
a. def find_complexity(n):
        I=1,s=1
        while s<=n:
            i+=1
            s+=I
        return
```

```
b. def find_complexity2(n):
        for i in range(n//2,n+1):
            for j in range(1,n-(n//2)+1):
                k=1
                while k<=n:
```

```
6                       k*=2
7           return
```

```
c. def find_complexity3(n):
        if(n<=2):
            return 1
        else:
            return find_complexity3(math.floor(math.sqrt(n))+1)
```

```
d. def find_complexity4(n, epsilon):
        s, e = 1, n
        m = (s+e)/2

        while abs(n - m*m) > epsilon:
            if m*m > n:
                e = m
            else:
                s = m
            m = (s+e)/2
        return m
```

Problem 3

Yagna and Alex are discussing about the implementation of a lookup dictionary of words with O(logn) complexity. Yagna came up with an idea of usage of skip lists. Lets help Yagna to implement a lookup dictionary using skip list.
Following are the instructions for SkipList:

- Create a class with LookUpSkipList. It should have a constructor which takes the list of words and probability “P” with which the element is chosen to the next level.
- Class should have the methods lookup_search, delete, insert, print.
- Insert method inserts the word into skiplists.
- Delete method deletes the word from skiplists.
- Lookup_search method checks the presence of the word and if word presents returns the True else insert the word to the skiplist.
- Print method prints the words from bottom level to top level. Bottom level contains all the words.

- All the methods except print should be implemented in O(logn) time complexity.

Note: The sorting of words can be like lexicographical order / dictionary order. All the words are in Lower case. For comparing the strings you can use inbuilt python functions.

Example:

lexicographical order : aa->aaa->ab->az->cd->df-> ja-> jb-> ... -> z->za..

```
Evaluation Usage will be like this:

lu_sk_lists = LookUpSkipList([‘‘iub’’,’’usa’’,’’there’’,’’sort’’,’’god’’],0.6)
lu_sk_lists.print()
lu_sk_lists.insert(‘‘word’’)
lu_sk_lists.delete(‘‘there’’)
lu_sk_lists.lookup_search(‘‘iub’’)
lu_sk_lists.lookup_search(‘‘present’’)
```

Problem 4

A basket has ‘n’ fruits of two types - apples and oranges. They decide to play a game. The game is played in multiple rounds until one of the fruit types wins (either apple or orange). In each round, every ‘eligible’ fruit gets to perform one of the two actions.

Action1:

Disqualify one member of another fruit type from the game, so that they cannot perform any action in the current round or any subsequent rounds. (Disqualified fruit becomes ‘ineligible’ and cannot perform any actions)

Action2:

If all members of another fruit type are disqualified(ineligible), declare his fruit type as the winner of the game.

Every round begins with fruits performing actions in the given order, and every fruit chooses the action best for his fruit type, predict which fruit type will be the winner!. Time complexity $O(n)$ and space complexity $O(n)$

Input: A string s of length ‘n’ representing order in which ‘n’ fruits can perform an action in every round. (‘A’: apple, ‘O’: orange) **Output:** Winner type: string (either “apple” or “orange”)

```
Example 1:

Input: s= ‘‘AO’’ (here n=2)
Output: ‘‘apple’’

Explanation:

In round1, fruits start performing actions in the given order (left to right).
```

First, ‘A’ will choose Action1 and disqualify ‘O’ from the game.
Now second ‘O’ is permanently disqualified
and cannot perform any actions. Round 1 ends.

In round 2: ‘A’ will choose Action2 as he’s the only ‘eligible’ member
in the basket as ‘O’ was disqualified
in the previous round. Hence the winner fruit type is ‘‘apple’’.

Example 2:

Input: s= ‘‘AOO’’ (here n=3)
Output: ‘‘orange’’

First fruit, ‘A’ will choose Action1 and disqualify second ‘O’
from the game. Now second ‘O’ is permanently disqualified
and cannot perform any actions, so it is skipped.
Next, 3rd fruit ‘O’ will choose Action1 and
disqualify first fruit ‘A’. Round 1 ends.

In round 2: Now both first ‘A’ and second ‘O’ are disqualified
after round1, the only eligible fruit left is third ‘O’,
so he will choose Action2 and declare fruit type ‘‘orange’’ as the winner.

```
def predict_winner(string s):
    ##########
       logic here
    ##########
    # ans: a string, either "apple" or "orange"
    return ans
```

Problem 5

Using the definitions of asymptotic notations.

- prove that $f(n) = 5 * n^3 + 2 * n^2 + 7 * n + 1$ is $O(n^3), \theta(n^3)$.

- prove that $f(n) = 5 * n^3 + 2 * n^2 + 7 * n + 1$ is $\omega(n^2), \Omega(n^2)$

Hint: You need to find $c0$ and $n0$

Problem 6

Erlich is standing at a point on a road, which is assumed to be infinitely extending right and left. On this road there is a buried treasure chest, and he can only find it when his steps passes over it. Erlich wants to find the treasure with least walking effort as he is lazy, and takes the help of Richard who comes up with a scanning strategy. At each iteration, Erlich needs to walk to the right (or left) by some number of steps, returns back to original position and then moves to the other direction with some number of steps and returns back again to the starting position. If the treasure is touched, Erlich immediately stops since the goal is reached.
Assuming the treasure is buried n steps right or left of the initial position, compute the number of steps Erlich will spent as a function of n.Analyze your function with the asymptotic notation to provide an upper bound.
Hint: You can assume that he increases his steps exponentially farther from the initial steps.You can provide your own strategy as well