Home Work № 6

CSCI B505 Applied Algorithms 10/17/2022

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 six problems. Problems 1 and 6- 20 points, problems 2, 3, 4 and 5 - 15 points each.
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

1.Given an integer $n(1 <= n <= 10^5)$, find the minimum number of 1's required to obtain n if you are allowed to hit the following three functions in any order any number of times . Desired time Complexity $(O(n^2))$, desired space complexity $(O(n))$.

```
i. def function1(a,b):
        return a+b

ii. def function2(a,b):
        return a*b

iii.def function3(a,b):
        return int(str(a)+str(b))
```

```
Example 1: Input:22
    Output:4
Explanation: On function3(1,1) we obtain 11(say a).
         On function1(1,1) we obtain 2(say b).
         Now we can hit function2(a,b) to obtain 22 which is the desired value.
         Here the number of 1's used are 4.

Example 2: Input: 9
       Output: 6
Explanation: function2(function1(function1(1,1),1),function1(1,function1(1,1)))

Example 3: Input: 1
       Output: 1
Explanation: Self explanatory
```

Problem 2

A robot can only walk in one of the four directions (left, right, top, bottom). The grid it is walking on is labeled with characters of the English alphabet. The programmers of the robot have now constrained its motion further. It can step from a grid cell A to grid cell B, if the character on cell A is less than the character on cell B.
The ordering on the characters is the lexicographic ordering.
Write a program that computes the maximum number of steps that the robot can take starting at any grid cell.

```
Ex -1:

    >>> max_steps([['d', 'b'], ['c', 'a']])
    2
    d b
    c a
    The robot can go from 'a' to 'b' and then to 'd'.

Ex -2:

    >>> max_steps([['t', 'o', 'y'], ['c', 'a', 't'], ['t', 'o' , 'p']])
    4
    t o y
    c a t
```

```
t o p
The robot can go from ‘a’ to ’o’ then to ’p’ then to ’t’ and finally to ’y’.
```

```
def max_steps(grid):
    ####logic###
    ##########
    return maximum_no_of_steps
```

Problem 3

Pushpa, a professional thief, wanted to steal coconuts from forest.In the forest the coconut trees are in a straight line. Each tree has a certain number of coconuts, the only constraint is that if you rob from the adjacent coconut trees, an alert will be sent to the forest officials who will arrest Pushpa. Write a program to find the maximum number of coconuts pushpa can steal without getting caught.

```
Example1:
list = [2,7,9,3,1]
output = 12
because he can rob from coconut tree 0, tree 2 tree 4 and
none of these trees are adjacent. So the output will be  2+ 9+ 1 = 12
```

```
def get_max_coconuts(nums):
    ## logic####
    ## #########
    return max_no_can_steal_without_getting_caught
```

Problem 4

Ujwala has given a list of non-negative integers and a target, the task is to find the sum of sub list numbers which are divisible by target and return those list of numbers. If sum is not found in list return empty list.

NOTE: Solve by using Dp, 1<=A<=100

```
Example 1:
    Input : arr[] = {3, 1, 7, 5};
        target = 6;
    Output : [1,5]

Example 2:
    Input : arr[ ] = {1, 6};
        target = 5;
    Output : [ ]
```

```
def subset_divisible(list A=[],target):
    ####logic###
    ##########
    return [list of divisible subset sum]
```

Problem 5

You are given a string ‘s’ and two characters c1 and c2. You can choose any one of the two characters and add it to any position in the string to create a new string (exactly once). For example, if s= ‘cdaabcacd’ and c1= ‘a’ and c2=’d’, you could create a new string ‘cdaab**d**cacd’ or ‘**a**cdaabcacd’ etc.(note that the character can be added to the beginning or the end of the string ‘s’). Your job is to find the **maximum** number of times string ‘c1c2’ can occur as a substring of the modified string using **Dynamic Programming**. The substring can be contiguous or non-contiguous (i.e for a string ‘abcdef’ ,‘ab’,’ce’, ‘ef’,’be’ all are valid substrings). Assume all characters and string s consists of lowercase english letters.

```
    Example:
    s = "bcedecd", c1 = ‘b’, c2= ‘d’
    output: 4
    Explanation:
    We can form  a modified string by adding c1 in s such that the
    modified string is ‘‘bcbedecd’’, the substring ‘c1c2’ (‘bd’) occurs 4 times
    in the modified string as: [‘‘bcbedecd’’,‘‘bcbedecd’’,‘‘bcbedecd’’,‘‘bcbedecd’’]
Any other modified string will have the occurrence of ‘c1c2’ 4 times or less.
Time complexity constraint: O(n) where n is the length of string s.Space complexity <= O(n)
```

```
def get_max_sbstr(s,c1,c2):
    ####logic####
```

```
3
4       ############
5       return ans
```

Problem 6

It is the year 11984 (yes, 11984), and the world is a dystopian society.
After decades of searching, the resistance has finally found Rick Sanchez's Portal Gun which lets them open portals to anywhere in the universe. This will be their key to winning the "war". Unfortunately, the Portal Gun is password protected, and they need your help to crack the password.
You know that the password is obtained by deleting some digits from the password file which you obtained from the Portal Gun's Persistent Random Access Memory (PRAM). The password file is just a list of decimal digits. Note that the password cannot be empty!
As a first step, you would like to find the total number of possible passwords. Write a program that computes the number of passwords.
Full credit only for the most efficient solution.

```
Examples:

 >>> num_passwords([1, 2, 1, 3])
     Output : 13

 The different possible passwords are {1, 2, 3, 12, 11, 13, 21,
 23, 121, 123, 113, 213, 1213}.

 >>> num_passwords([9, 9, 9, 9])
     Output : 4

 The different possible passwords are {9, 99, 999, 9999}.
```

```
1
2   def num_passwords(digit_list):
3         ####logic####
4
5       ############
6       return ans
```