mailed-assignment

November 25, 2023

1. REVERSE THE STRING Given an input strings, reverse the order of the words. A word is defined as a sequence of non-space characters. The words in s will be separated by at least one space. Return a string of the words in reverse order concatenated by a single space. Note that s may contain leading or trailing spaces or multiple spaces between two words. The returned string should only have a single space separating the words. Do not include any extra spaces

[ ]:
```
def reverse_string(n):
  n=n.split()
  reverse=n[::-1]
  reverse_join= " ".join(reverse)
  return (str(reverse_join))
```

[ ]:
```
s = "the sky is blue"
reverse_string(s)
```

[ ]:
```
'blue is sky the'
```

[ ]:
```
s = "hello world"
reverse_string(s)
```

[ ]:
```
'world hello'
```

[ ]:
```
s = "a      good example"
reverse_string(s)
```

[ ]:
```
'example good a'
```

2. STRING COMPRESSION Given an array of characters chars, compress it using the following algorithm: Begin with an empty string s. For each group of consecutive repeating characters in chars: If the group's length is 1, append the character to s. Otherwise, append the character followed by the group's length. The compressed string s should not be returned separately, but instead, be stored in the input character array chars. Note that group lengths that are 10 or longer will be split into multiple characters in chars. After you are done modifying the input array, return the new length of the array. You must write an algorithm that uses only constant extra space

[ ]:
```
def compress(n):
  dic=dict()
  new_list=[]
  final_list=[]
  for fea in n:
    dic.update({fea:n.count(fea)})
  for features in dic:
    if dic[features]>1:
      new_list.append(features+str(dic[features]))
    elif dic[features]==1:
      new_list.append(features)
  string="".join(new_list)
  for num in string:
    final_list.append(str(num))
  print(final_list)
```

```
[ ]: chars = ["a", "a", "b", "b", "c", "c", "c"]
      compress(chars)
```

```
['a', '2', 'b', '2', 'c', '3']
```

```
[ ]: chars = ["a"]
     compress(chars)
```

```
['a']
```

```
[ ]: chars = ["a", "b", "b", "b", "b", "b", "b", "b", "b", "b", "b", "b", "b"]
     compress(chars)
```

```
['a', 'b', '1', '2']
```

3. Given Temperatures Given an array of integer's temperatures represents the daily temperatures, return an array answer such that answer[i] is the number of days you have to wait after the ith day to get a warmer temperature. If there is no future day for which this is possible, keep answer[i] == 0 instead.

```
[ ]: def temp_days(n):
       Days=[]
       if n[0] == n[(len(n)-1)]:
         for fea in range(0,(len(n)-2)):
           if n[fea]<n[fea+1]:
             Days.append(1)
           elif n[fea]>n[fea+1]:
             Days.append(n[fea]-n[fea+1])
         Days.append(0)
         Days.append(0)
       else:
         for fea in range(0,(len(n)-1)):
           if n[fea]<n[fea+1]:
             Days.append(1)
```

```
      elif n[fea]>n[fea+1]:
        Days.append(n[fea]-n[fea+1])
    Days.append(0)







  print(Days)
```

[ ]:
```
temperatures = [73, 74, 75, 71, 69, 72, 76, 73]
temp_days(temperatures)
```

```
[1, 1, 4, 2, 1, 1, 0, 0]
```

[ ]:
```
temperatures = [30, 40, 50, 60]
 temp_days(temperatures)
```

```
[1, 1, 1, 0]
```

[ ]:
```
temperatures = [30, 60, 90]
temp_days(temperatures)
```

```
[1, 1, 0]
```

4. Find the closest pair from two sorted arrays Given two sorted arrays and a number x, find the pair whose sum is closest to x and the pair has an element from each array. We are given two arrays ar1 [0…m-1] and ar2 [0..n-1] and a number x, we need to find the pair ar1 [i]

- ar2 [j] such that absolute value of (ar1 [i] + ar2[j] - x) is minimum.

[ ]:
```
def sorted_array(n1,n2,x):
  d=dict()
  dic=dict()
  add=[]
  subtract=[]
  keep=[]
  for fea in n1:
    for num in n2:
      d.update({fea+num:[fea,num]})
      add.append(fea+num)
  for features in add:
    if features < x:
      k=x-features
      dic.update({k:features})
  express=dic[min(dic)]
  for alph in d:
    if alph == express:
      value=d[alph]
  print("Output is",value[0] ,"and",value[1])
```

[ ]:
```
ar1 = (1, 4, 5, 7)
ar2= (10, 20, 30, 40)
x = 32
```

[ ]:
```
sorted_array(ar1,ar2,x)
```

```
Output is 1 and 30
```

[ ]:
```
ar1=1, 4, 5, 7
ar2=10, 20, 30, 40
x = 58
sorted_array(ar1,ar2,x)
```

```
Output is 7 and 40
```

5. Longest palindromic string Given a string s, return the longest palindromic substring in s. Example 1: Input: s = “babad” Output: “bab” Explanation: “aba” is also a valid answer. Example 2: Input: s = “cbbd” Output: “bb”

[ ]:
```
def palindro(n):
  new_alph=n[::-1]
  if n==new_alph:
    result=new_alph
    print(result)
    Break


  else:
    box=[]
    for fea in range(len(n)):
      n=n[fea:(len(n)-1+fea)]
      box.append(n)
    for features in box:
      if len(features)==1:
        Break
      elif len(features)==0:
        Break
      else:
        palindro(features)
```

[ ]:
```
s="cbbd"
palindro(s)
```

```
bb
```

```
[ ]: s="babad"
     palindro(s)
```

```
bab
aba
```

6. Longest substring without repeating character Given a string s, find the length of the longest substring without repeating characters.

```
[ ]: def substring(n):
       box=[]
       for fea in n:
         box.append(fea)
       box.remove(box[0])
       new_str="".join(box)
       print(len(set(new_str)))
```

```
[ ]: s = "abcabcbb"
      substring(s)
```

```
3
```

```
[ ]: s = "bbbbb"
     substring(s)
```

```
1
```

```
[ ]: s = "pwwkew"
     substring(s)
```

```
3
```

7. SORT colors Given an array nums with n objects colored red, white, or blue, sort them in-place so that objects of the same color are adjacent, with the colors in the order red, white, and blue. We will use the integers 0, 1, and 2 to represent the color red, white, and blue, respectively. You must solve this problem without using the library's sort function.

```
[ ]: def Sort(n):
       for fea in range(len(n)):
         for num in range(fea+1,len(n)):
           if n[fea]>=n[num]:
             n[fea],n[num]=n[num],n[fea]
       print(n)
```

```
[ ]: nums = [2, 0, 2, 1, 1, 0]
     Sort(nums)
```

```
[0, 0, 1, 1, 2, 2]
```

[ ]:
```
nums = [2, 0, 1]
 Sort(nums)
```

```
[0, 1, 2]
```

8. Removing stars from string You are given a string s, which contains stars *. In one operation, you can: Choose a star in s. Remove the closest non-star character to its left, as well as remove the star itself. Return the string after all stars have been removed. Note: The input will be generated such that the operation is always possible. It can be shown that the resulting string will always be unique.

[24]:
```
def remove_star(n):
  box=[]
  for fea in range(len(n)):
    if n[fea]=="*":
      box.pop()
    else:
      box.append(n[fea])
  new_str="".join(box)
  print(new_str)
```

[25]:
```
s = "leet**cod*e"
 remove_star(s)
```

```
lecoe
```

[26]:
```
s = "erase*****"
 remove_star(s)
```

[ ]:
```
```