Module02_Day07_Problem_Solving

December 16, 2022

Problem_Solving

[ ]:
```
def first_recurring_char(a):
    n=len(a)
    for i in range(n-1):
        count = 1
        for j in range(i+1,n):
            if a[i] == a[j]:
                count += 1
            if count > 1:
                return count,a[i]
    return -1
```

[ ]:
```
first_recurring_char("nterviewbit")
```

[ ]:
```
(2, 't')
```

[ ]:
```
def recurring_char(a):
    n=len(a)
    for i in range(n):
        if a[i] in a[:i]:
            return a[i]
    return -1
```

[ ]:
```
recurring_char("nterviewbit")
```

[ ]:
```
'e'
```

$ * $ Sets are always constant while searching # O(1)

[ ]:
```
def opt_recucring_char(a):
    n=len(a)
    recurring = set()
    for i in range(n):
        if a[i] not in recurring: # This is O(1) because sets search in␣
  ↪constant time, even dictionaries
            recurring.add(a[i])
        else:
            return a[i]
    return -1
```

[ ]:
```
opt_recucring_char("scaler"), opt_recucring_char("nterviewbit")
```

[ ]:
```
(-1, 'e')
```

[ ]:
```
def floor(arr,num):
    end = len(arr) -1
    start = 0
    while(start<=end):
        if num > arr[-1]: return arr[-1]
        if num < arr[0]: return "None"
        mid = (start+end)//2
        if arr[mid]< num < arr[mid+1] or num==arr[mid]:
            return arr[mid]
        elif arr[mid] < num:
            start = start + 1
        else:
            end = end-1
    return -1
```

[ ]:
```
floor([-5,2,3,6,9,10,11,14,18],-7)
```

[ ]:
```
'None'
```

[ ]:
```
def opt_floor(arr,num):
    end = len(arr) -1
    start = 0
    ans = None
    while(start<=end):
        mid = (start+end)//2
        if num==arr[mid]:
            return arr[mid]
        elif arr[mid] < num:
            ans = arr[mid]
            start = start + 1
        else:
            end = end-1
    return ans
```

[ ]:
```
opt_floor([-5,2,3,6,9,10,11,14,18],7)
```

[ ]:
```
6
```

[ ]:
```
def isPowerOf2(n):

    isTwoMultiple = 1
```

```python
    while n > 1:
        if n % 2 == 0:
            n = n//2
        else:
            return 0

    return isTwoMultiple
```

[ ]:

```python
import math

def super_opt_isPowerof2(n):
    ans = math.log2(n)
    if ans == int(ans) :
        return 1
    else:
        return 0
```

[ ]:

```python
def isPowerOf2_opt(n):
    return n & (n-1) == 0

isPowerOf2_opt(1)
```

[ ]:

```
True
```

[ ]:

```python
def euc_distance(A,B):
    x1 = A[0]
    x2 = B[0]
    y1 = A[1]
    y2 = B[1]

    distance =  ((x2-x1)**2 + (y2-y1)**2)**0.5
    return round(distance,2)

def nearestNeighbour(lst,loc):
    distance = list()
    for cord in lst:
        distance.append(euc_distance(loc,cord))
    return distance, min(distance)
```

[ ]:

```python
nearestNeighbour([(1,2),(3,6),(-1,5),(-1,-2),(-3,4),(2,2)], (2,3))
```

[ ]:

```
([1.41, 3.16, 3.61, 5.83, 5.1, 1.0], 1.0)
```

[ ]:

```python
# NLP:
# n-gram -> You are givrn with a paragraph

# "be the change you want to see in the world" , n = 2
# "(be,the),(the,change),(change,you),(you,want)........(the,world)"

def ngram(arr,step):
    arr = arr.split()
    n = len(arr)
    tokens = list()

    for i in range(n - step + 1):
        tokens.append(arr[i:i+step])
    return tokens
```

```
[ ]: ngram("be the change you want to see in the world",2)
```

```
[ ]: [['be', 'the'],
      ['the', 'change'],
      ['change', 'you'],
      ['you', 'want'],
      ['want', 'to'],
      ['to', 'see'],
      ['see', 'in'],
      ['in', 'the'],
      ['the', 'world']]
```

```
[ ]: def foo(n):
         return math.log2(n), 10 ,n**0.5 , 100/n, n
```

```
[ ]: foo(10000)
```

```
[ ]: (13.287712379549449, 10, 100.0, 0.01, 10000)
```

```
[ ]: s1 = "silent"
     s1 = set(s1)
     s2 = set("listen")
     s2 == s1
```

```
[ ]: True
```

```
[ ]: s = {"scaler": [5, 7, 5, 4, 5], "is": [6, 7, 4, 3, 3], "best": [9, 9, 6, 5, 5]}
     for key in s:
         print(s[key])
```

```
[5, 7, 5, 4, 5]
[6, 7, 4, 3, 3]
[9, 9, 6, 5, 5]
```

```
[ ]:
```