Week 2 Live Coding Solution

Live Coding Problem 1

Write a function `Findpeak(L)` that accepts a list L of `n` distinct elements and returns the peak element of the list. A list element is a peak if it is greater than its neighbors. For corner elements, we need to consider only one neighbor. Write a solution of O(logn) complexity. Consider that there is only one peak is in the list, L.

Sample Input 1

```
[5, 10, 20, 15]
```

Output

```
20
```

Sample Input 2

```
[1,2,3,4,5,6,7,8]
```

Output

```
8
```

Solution Code

Solution Code

```
def findPeakUtil(arr, low, high, n):
    mid = low + (high - low)/2
    mid = int(mid)

    if ((mid == 0 or arr[mid - 1] < arr[mid]) and (mid == n - 1 or arr[mid + 1] < arr[mid])):
        return arr[mid]

    elif (mid > 0 and arr[mid - 1] > arr[mid]):
        return findPeakUtil(arr, low, (mid - 1), n)
    else:
        return findPeakUtil(arr, (mid + 1), high, n)

def Findpeak(L):
    n = len(L)
    return findPeakUtil(L,0,n-1,n)
```

Suffix code(visible)

```
L = eval(input())
res = Findpeak(L)
print(res)
```

Public Test case

Input 1

```
[5, 10, 20, 15]
```

Output

```
20
```

Input 2

```
[1,2,3,4,5,6,7,8]
```

Output

```
8
```

Input 3

```
[1,2,3,4,5,6,7,8,7,6]
```

Output

```
8
```

Input 4

```
[7,6,5,4,3,2,1]
```

Output

```
7
```

Note:- Some test cases are very big to check the efficiency of the solution code for this problem because of that, these test cases are not included in the solution file.

Live Coding-Problem 2

Write a Python function `findCommonElements(L1, L2)` that accepts two integer lists `L1` and `L2` of same length `n` and return a list that contains elements that are common to both lists. Write a efficient solution that runs in `O(nlogn)` time.

- `L1` contains all distinct integers and `L2` contains all distinct integers, but there can be many elements common between `L1` and `L2`.
- Returned list contains all elements that are common to `L1` and `L2`. The elements in the returned list can be in any order.

For example.

if `L1` = [5, 8, 2] and `L2` = [6, 8, 1] then, `findCommonElements(L1, L2)` should return list [8].

if `L1` = [3, 7, 2, 9, 5] and `L2` = [6, 3, 7, 5, 4] then, `findCommonElements(L1, L2)` should return list [3, 7, 5].

Elements in returned list can be in any order.

Do not use Python set built-in function

Sample input

```
[23, 24, 18, 22, 20, 10, 17, 12, 16, 19, 21, 15, 14, 11, 13]
[23, 22, 33, 24, 31, 21, 20, 26, 30, 29, 25, 27, 28, 34, 32]
```

Output

```
[20, 21, 22, 23, 24]
```

Sample input 2

```
[3, 7, 2, 9, 5]
[6, 3, 7, 5, 4]
```

Output

```
[3, 5, 7]
```

Solution Code

Solution Code

```
def binarySearch(L, k):
  s = len(L)
  if(s < 1):
    return False
  left = 0
  right = s - 1
```

```
  while(left <= right):
    mid = (left + right)//2
    if (k == L[mid]):
      return True
    elif (k < L[mid]):
      right = mid - 1
    else:
      left = mid + 1
  return False
def findCommonElements(L1, L2):
  L1.sort()
  L3 = []
  for item in L2:
    if (binarySearch(L1, item) == True):
      L3.append(item)
  return L3

```

Suffix Code(Visible)

```
A = eval(input())
B = eval(input())
result = findCommonElements(A, B)
result.sort()
print(result)
```

Public Testcase

Input 1

```
[23, 24, 18, 22, 20, 10, 17, 12, 16, 19, 21, 15, 14, 11, 13]
[23, 22, 33, 24, 31, 21, 20, 26, 30, 29, 25, 27, 28, 34, 32]
```

Output

```
[20, 21, 22, 23, 24]
```

Input 2

```
[3, 7, 2, 9, 5]
[6, 3, 7, 5, 4]
```

Output

```
[3, 5, 7]
```

Note:- Some test cases are very big to check the efficiency of the solution code for this problem because of that, these test cases are not included in the solution file.

Live Coding-Problem 3

You have a deck of shuffled cards ranging from 0 to 100,000,000. There are 2 sub-ordinate below you and two subordinates below them and it goes on.

- The job of the sub-ordinate is to split the deck of cards that they received and give it to two sub-ordinate of them. If they receive a deck of cards from their subordinates, they merge it in an ascending order and give it their higher level.
- If a subordinate received only two card, then he/she himself/herself arrange in ascending order give it back that to the superior.
- If a subordinate received only one card, then he/she will give back that to the superior.

```
Terminology:

(67) subordinate number 67

[1, 3, 5, 2] -> [1, 2, 3, 5] deck of cards got -> deck of cards returned

--------------------------------

(0) [3, 1, 2, 0, 5] -> [0, 1, 2, 3, 5]
|
|--(1) [3, 1] -> [1, 3]
|
|--(2) [2, 0, 5] -> [0, 2, 5]
    |
    |--(3) [2] -> [2]
    |
    |--(4) [0, 5] -> [0, 5]
```

Your task is to find how many people (including you) are required to sort the cards and print the sorted deck of cards and number of people required as a tuple.

Write the function **def subordinates(L):**

```
def subordinates(L):
```

Sample input 1

```
[194, 69, 103, 150, 151, 44, 103, 98]
```

Output

```
([44, 69, 98, 103, 103, 150, 151, 194], 7)
```

Sample input 2

```
[10, 33, 45, 67, 92, 100, 5]
```

Output

```
([5, 10, 33, 45, 67, 92, 100], 7)
```

Solution Code

Solution Code

```
def merge(A, B):
  m, n = len(A), len(B)
  C, i, j, k = [], 0, 0, 0
  while k < m + n:
    if i == m:
      C.extend(B[j:])
      k = k + n - j
    elif j == n:
      C.extend(A[i:])
      k = k + m - i
    elif A[i] < B[j]:
      C.append(A[i])
      i, k = i + 1, k + 1
    else:
      C.append(B[j])
      j, k = j + 1, k + 1
  return C

def mergesort(L):
  global c
  c += 1
  n = len(L)
  if n == 2:
    return L if L[0] < L[1] else L[::-1]
  if n <= 1:
    return L
  m = n//2
  l = mergesort(L[:m])
  r = mergesort(L[m:])

  L_ = merge(l, r)

  return L_

def subordinates(L):
  return mergesort(L), c
c=0
```

Suffix code(Visible)

```
print(subordinates(eval(input())))
```

Public Testcase

Input 1

```
[10, 33, 45, 67, 92, 100, 5, 99, 105]
```

Output

```
([5, 10, 33, 45, 67, 92, 99, 100, 105], 9)
```

Input 2

```
[194, 69, 103, 150, 151, 44, 103, 98]
```

Output

```
([44, 69, 98, 103, 103, 150, 151, 194], 7)
```

Input 3

```
[10, 33, 45, 67, 92, 100, 5]
```

Output

```
([5, 10, 33, 45, 67, 92, 100], 7)
```

Private Testcase

Input 1

```
[10, 33, 45, 67, 92, 100, 5, 99, 105, 26, 201]
```

Output

```
([5, 10, 26, 33, 45, 67, 92, 99, 100, 105, 201], 13)
```

Input 2

```
[10, 33, 45, 67, 92, 100, 5, 99, 105, 26]
```

Output

```
([5, 10, 26, 33, 45, 67, 92, 99, 100, 105], 11)
```

Input 3

```
[136, 152, 147, 3, 14, 43, 162, 169, 5, 10, 108, 182, 98, 71, 25, 122, 86,
82, 25, 176, 53, 14, 53, 85, 25, 110, 41, 5, 46, 168, 192, 140, 82, 109, 37,
165, 41, 146, 48, 123, 166, 17, 85, 87, 118, 133, 108, 170, 119, 198, 53,
154, 55, 193, 129, 189, 77, 69, 87, 80, 23, 58, 48, 79, 12, 4, 66, 30, 182,
118, 89, 140, 89, 15, 169, 101, 172, 0, 53, 24, 149, 168, 106, 41, 49, 148,
151, 88, 95, 192, 7, 101, 50, 40, 196, 72]
```

Output

```
([0, 3, 4, 5, 5, 7, 10, 12, 14, 14, 15, 17, 23, 24, 25, 25, 25, 30, 37, 40,
41, 41, 41, 43, 46, 48, 48, 49, 50, 53, 53, 53, 53, 55, 58, 66, 69, 71, 72,
77, 79, 80, 82, 82, 85, 85, 86, 87, 87, 88, 89, 89, 95, 98, 101, 101, 106,
108, 108, 109, 110, 118, 118, 119, 122, 123, 129, 133, 136, 140, 140, 146,
147, 148, 149, 151, 152, 154, 162, 165, 166, 168, 168, 169, 169, 170, 172,
176, 182, 182, 189, 192, 192, 193, 196, 198], 127)
```

Input 4

```
[147, 3, 14, 43, 162, 169, 5, 10, 108, 182, 98, 71, 25, 122, 86, 82, 25, 176,
53, 14, 53, 85, 25, 110, 41, 5, 46, 168, 192, 140, 82, 109, 37, 165, 41, 146,
48, 123, 166, 17, 85, 87, 118, 133, 108, 170, 119, 198, 53, 154, 55, 193,
129, 189, 77, 69, 87, 80, 23, 58, 48, 79, 12, 4, 66, 30, 182, 118, 89, 140,
89, 15, 169, 101, 172, 0, 53, 24, 149, 168, 106, 41, 49, 148, 151, 88, 95,
192, 7, 101, 50, 40]
```

Output

```
([0, 3, 4, 5, 5, 7, 10, 12, 14, 14, 15, 17, 23, 24, 25, 25, 25, 30, 37, 40,
41, 41, 41, 43, 46, 48, 48, 49, 50, 53, 53, 53, 53, 55, 58, 66, 69, 71, 77,
79, 80, 82, 82, 85, 85, 86, 87, 87, 88, 89, 89, 95, 98, 101, 101, 106, 108,
108, 109, 110, 118, 118, 119, 122, 123, 129, 133, 140, 140, 146, 147, 148,
149, 151, 154, 162, 165, 166, 168, 168, 169, 169, 170, 172, 176, 182, 182,
189, 192, 192, 193, 198], 119)
```

Live Coding-Problem 4

Given a list `L` of random numbers and another number `pairSum`, find whether there exists two numbers in the list such that their sum is equal to `pairSum`.

Write a Python function ***findPair(L, pairSum)*** that return `True` if there exist a pair of integers in `L` whose sum is equal to `x`, `False` otherwise.. Try to write a solution which is O(nlogn) or better.

Hint: Try to sort the list first.

For example consider the below list. We need to find if there exists any pair whose sum is equal to 21. 11+10 = 21. So the function should return True.

For the same list if we want to find if there exist any pair whose sum is equal to 2. Clearly there is no such pair so the function should return False.

Sample Input 1

```
10 4 11 5 1 8 7
21
```

Sample Output 1

```
True
```

Sample Input 2

```
10 4 11 5 1 8 7
2
```

Sample Output 2

```
False
```

Solution Code

Solution Code

```python
def findPair(L, x):
  L.sort()
  left = 0
  right = len(L) - 1
  while(left < right):
    sum = L[left] + L[right]
    if (sum == x):
      return True
    elif (sum > x):
      right -= 1
    else:
      left += 1
  return False
```

Suffix Code(Visible)

```
L = [int(item) for item in input().split()]
pairsum = int(input())
print(findPair(L, pairsum))
```

Public Testcase

Input 1

```
3825 29286 12635 73450 69276 1052 67615 27144 32660
3825
```

Output

```
False
```

Input 2

```
3825 29286 12635 73450 69276 1052 67615 27144 32660
33111
```

Output

```
True
```

Input 3

```
3825 29286 12635 73450 69276 1052 67615 27144 32660
102736
```

Output 3

```
True
```

Input 4

```
7625 33422 45912 24600 38030 64596 35281 49247 808 45679 38388 37078 7997
43351 81537 87537 17318 9801
41047
```

Output

```
True
```

Private Testcase

Input 1

```
7625 33422 45912 24600 38030 64596 35281 49247 808 45679 38388 37078 7997
43351 81537 87537 17318 9801
79334
```

Output

```
True
```

Input 2

```
7625 33422 45912 24600 38030 64596 35281 49247 808 45679 38388 37078 7997
43351 81537 87537 17318 9801
71452
```

Output

```
True
```

Input 3

```
7625 33422 45912 24600 38030 64596 35281 49247 808 45679 38388 37078 7997
43351 81537 87537 17318 9801 57304 41499 75008 80402 15792 8367 28861 10894
26350 27407 99522 30410 89571 41049 97045 83297 26106 70244 48446 61643 77322
62453 70568 77003 84457 52794 6887 21907 28722 38845 2772 20660 54014 77490
85218
42168
```

Output

```
True
```

Input 4

```
7625 33422 45912 24600 38030 64596 35281 49247 808 45679 38388 37078 7997
43351 81537 87537 17318 9801 57304 41499 75008 80402 15792 8367 28861 10894
26350 27407 99522 30410 89571 41049 97045 83297 26106 70244 48446 61643 77322
62453 70568 77003 84457 52794 6887 21907 28722 38845 2772 20660 54014 77490
85218
42000
```

Output

```
False
```