

10 - Searching & Sorting

Ex. No. : 10.1

Date: 29/05/2024

Register No.: 231401074

Name: P.C.Pavan

Merge Sort

Write a Python program to sort a list of elements using the merge sort algorithm.

For example:

Input	Result
5 6 5 4 3 8	3 4 5 6 8

Answer:

```
def mergeSort(arr):  
    if len(arr) > 1:  
        mid = len(arr) // 2  
        left_half = arr[:mid]  
        right_half = arr[mid:]  
  
        mergeSort(left_half)  
        mergeSort(right_half)  
  
        i = j = k = 0  
        while i < len(left_half) and j < len(right_half):  
            if left_half[i] < right_half[j]:  
                arr[k] = left_half[i]  
                i += 1  
            else:  
                arr[k] = right_half[j]
```

```

        j += 1

    k += 1

while i < len(left_half):
    arr[k] = left_half[i]
    i += 1
    k += 1

while j < len(right_half):
    arr[k] = right_half[j]
    j += 1
    k += 1

n = int(input().strip())
arr = list(map(int, input().strip().split()))

mergeSort(arr)
print(*arr)

```

Input	Expected	Got	
5 6 5 4 3 8	3 4 5 6 8	3 4 5 6 8	
9 14 46 43 27 57 41 45 21 70	14 21 27 41 43 45 46 57 70	14 21 27 41 43 45 46 57 70	
4 86 43 23 49	23 43 49 86	23 43 49 86	

Ex. No. : 10.2

Date: 29/05/2024

Register No.: 231401074

Name: P.C.Pavan

Bubble Sort

Given an list of integers, sort the array in ascending order using the *Bubble Sort* algorithm above. Once sorted, print the following three lines:

1. [List](#) is sorted in numSwaps swaps., where numSwaps is the number of swaps that took place.
2. First Element: firstElement, the *first* element in the sorted [list](#).
3. Last Element: lastElement, the *last* element in the sorted [list](#).

For example, given a worst-case but small array to sort: $a=[6,4,1]$. It took 3 swaps to sort the array. Output would be

Array is sorted in 3 swaps.

First Element: 1

Last Element: 6

Input Format

The first line contains an integer, n , the size of the [list](#) a .

The second line contains n , space-separated integers $a[i]$.

Constraints

- $2 \leq n \leq 600$
- $1 \leq a[i] \leq 2 \times 10^6$.

Output Format

You must print the following three lines of output:

1. [List](#) is sorted in numSwaps swaps., where numSwaps is the number of swaps that took place.
2. First Element: firstElement, the *first* element in the sorted [list](#).
3. Last Element: lastElement, the *last* element in the sorted [list](#).

Sample Input 0

3

1 2 3

Sample Output 0

[List](#) is sorted in 0 swaps.

First Element: 1

Last Element: 3

For example:

Input	Result
3 3 2 1	List is sorted in 3 swaps. First Element: 1 Last Element: 3
5 1 9 2 8 4	List is sorted in 4 swaps. First Element: 1 Last Element: 9

Answer:

```
def bubbleSort(arr):  
    n = len(arr)  
    numSwaps = 0  
  
    for i in range(n):  
        swapped = False  
  
        for j in range(0, n-i-1):  
            if arr[j] > arr[j+1]:  
                arr[j], arr[j+1] = arr[j+1], arr[j]  
                numSwaps += 1  
                swapped = True  
  
        if not swapped:  
            break  
  
    print("List is sorted in", numSwaps, "swaps.")  
    print("First Element:", arr[0])  
    print("Last Element:", arr[-1])
```

```

n = int(input().strip())
arr = list(map(int, input().strip().split()))
bubbleSort(arr)

```

Input	Expected	Got	
	3 3 2 1	List is sorted in 3 swaps. First Element: 1 Last Element: 3	List is sorted in 3 swaps. First Element: 1 Last Element: 3
	5 1 9 2 8 4	List is sorted in 4 swaps. First Element: 1 Last Element: 9	List is sorted in 4 swaps. First Element: 1 Last Element: 9

Ex. No. : 10.3

Date: 29/05/2024

Register No.: 231401074

Name: P. C. Pavan

Peak Element

Given an list, find peak element in it. A peak element is an element that is greater than its neighbors.

An element $a[i]$ is a peak element if

$A[i-1] \leq A[i] \geq A[i+1]$ for middle elements. $[0 < i < n-1]$

$A[i-1] \leq A[i]$ for last element $[i=n-1]$

$A[i] \geq A[i+1]$ for first element $[i=0]$

Input Format

The first line contains a single integer n , the length of A .

The second line contains n space-separated integers, $A[i]$.

Output Format

Print peak numbers separated by space.

Sample Input

5

8 9 10 2 6

Sample Output

10 6

For example:

Input	Result
4 12 3 6 8	12 8

Answer:

```
def find_peaks(nums):
    peaks = []
    for i in range(len(nums)):
        if i == 0:
            if nums[i] >= nums[i+1]:
                peaks.append(nums[i])
        elif i == len(nums) - 1:
            if nums[i] >= nums[i-1]:
                peaks.append(nums[i])
        else:
            if nums[i] >= nums[i-1] and nums[i] >= nums[i+1]:
                peaks.append(nums[i])
    return peaks
```

```
n = int(input())
nums = list(map(int, input().split()))
peaks = find_peaks(nums)
print(' '.join(map(str, peaks)))
```

Input	Expected	Got	
7 15 7 10 8 9 4 6	15 10 9 6	15 10 9 6	
4 12 3 6 8	12 8	12 8	

Ex. No. : 10.4

Date: 29/05/2024

Register No.: 231401074

Name: P.C.Pavan

Binary Search

Write a Python program for binary search.

For example:

Input	Result
1 2 3 5 8 6	False
3 5 9 45 42 42	True

Answer:

```
def binary_search(arr, target):
```

```
    left, right = 0, len(arr) - 1
```

```
    while left <= right:
```

```
        mid = (left + right) // 2
```

```
        if arr[mid] == target:
```

```
            return True
```

```
        elif arr[mid] < target:
```

```
            left = mid + 1
```

```
        else:
```

```
            right = mid - 1
```

```
    return False
```

```
arr_input = input()
```

```
target_input = input()
```

```
arr = list(map(int, arr_input.split(',')))
```

```
target = int(target_input)
```

```
arr.sort()
```

```
result = binary_search(arr, target)
```

```
print(result)
```

Input	Expected	Got	
1,2,3,5,8 6	False	False	
3,5,9,45,42 42	True	True	
52,45,89,43,11 11	True	True	

Ex. No. : 10.5

Date: 29/05/2024

Register No.: 231401074

Name: P.C.Pavan

Frequency of Elements

To find the frequency of numbers in a list and display in sorted order.

Constraints:

$1 \leq n$, $\text{arr}[i] \leq 100$

Input:

1 68 79 4 90 68 1 4 5

output:

1 2

4 2

5 1

68 2

79 1

90 1

For example:

Input	Result
4 3 5 3 4 5	3 2 4 2 5 2

Answer:

```
def frequencySorted(arr):  
    freq_dict = {}  
    for num in arr:  
        if num in freq_dict:  
            freq_dict[num] += 1  
        else:  
            freq_dict[num] = 1  
    sorted_freq = sorted(freq_dict.items())  
    for key, value in sorted_freq:  
        print(key, value)  
arr = list(map(int, input().strip().split()))  
frequencySorted(arr)
```

Input	Expected	Got
4 3 5 3 4 5	3 2 4 2 5 2	3 2 4 2 5 2
12 4 4 4 2 3 5	2 1 3 1 4 3 5 1 12 1	2 1 3 1 4 3 5 1 12 1
5 4 5 4 6 5 7 3	3 1 4 2 5 3 6 1 7 1	3 1 4 2 5 3 6 1 7 1