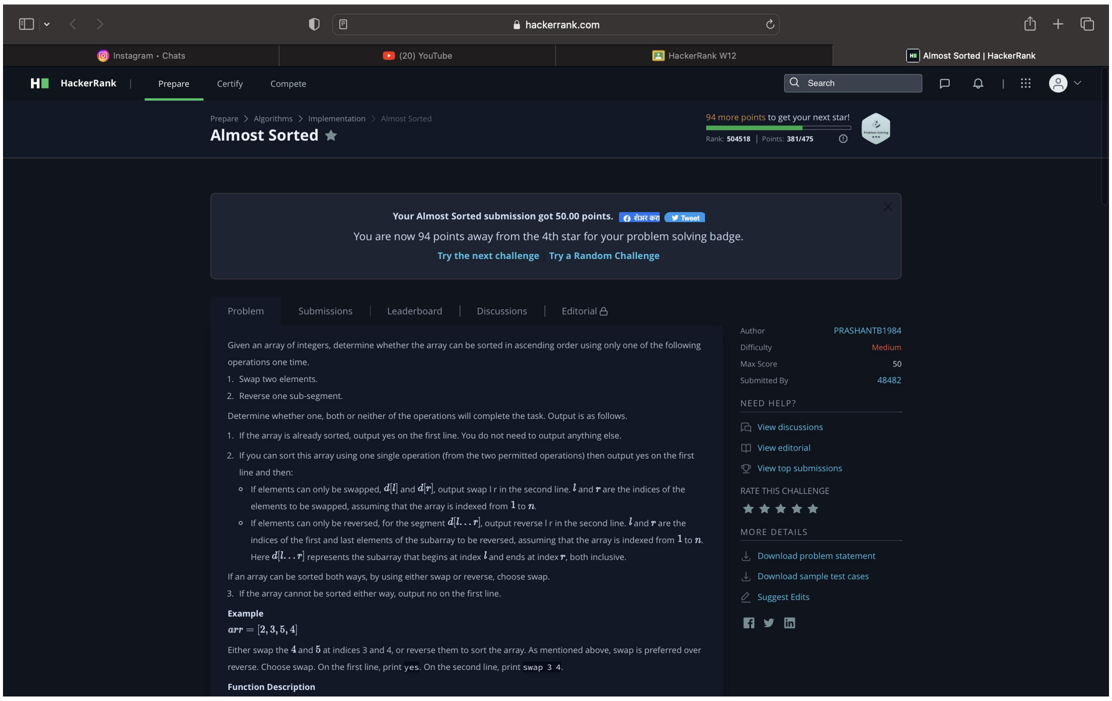
Advanced Programming Practices
Hackerrank Week-12

Name: Ronit Kumar Reg: RA2111032010009 Section: T2 Branch:CSE w/s OT

Ques 1: Almost Sorted

Problem:



Almost Sorted ★

Your Almost Sorted submission got 50.00 points.

You are now 94 points away from the 4th star for your problem solving badge.

Try the next challenge | Try a Random Challenge

Given an array of integers, determine whether the array can be sorted in ascending order using only one of the following operations one time.

1. Swap two elements.
2. Reverse one sub-segment.

Determine whether one, both or neither of the operations will complete the task. Output is as follows.

1. If the array is already sorted, output yes on the first line. You do not need to output anything else.
2. If you can sort this array using one single operation (from the two permitted operations) then output yes on the first line and then:
   - If elements can only be swapped, $d[l]$ and $d[r]$, output swap l r in the second line. l and r are the indices of the elements to be swapped, assuming that the array is indexed from 1 to n.
   - If elements can only be reversed, for the segment $d[l...r]$, output reverse l r in the second line. l and r are the indices of the first and last elements of the subarray to be reversed, assuming that the array is indexed from 1 to n. Here $d[l...r]$ represents the subarray that begins at index l and ends at index r, both inclusive.

If an array can be sorted both ways, by using either swap or reverse, choose swap.

3. If the array cannot be sorted either way, output no on the first line.

Example

$arr = [2, 3, 5, 4]$

Either swap the 4 and 5 at indices 3 and 4, or reverse them to sort the array. As mentioned above, swap is preferred over reverse. Choose swap. On the first line, print `yes`. On the second line, print `swap 3 4`.

Function Description

Author PRASHANTB1984
Difficulty Medium
Max Score 50
Submitted By 48482

NEED HELP?

View discussions
View editorial
View top submissions

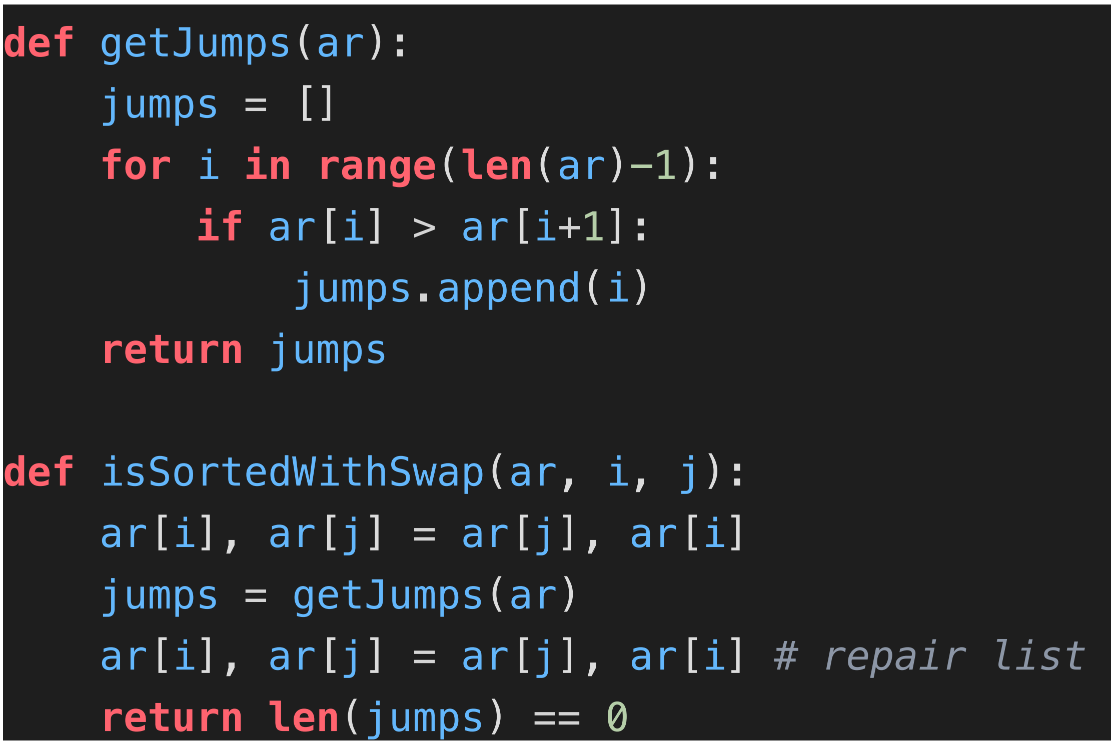
RATE THIS CHALLENGE

★ ★ ★ ★ ★

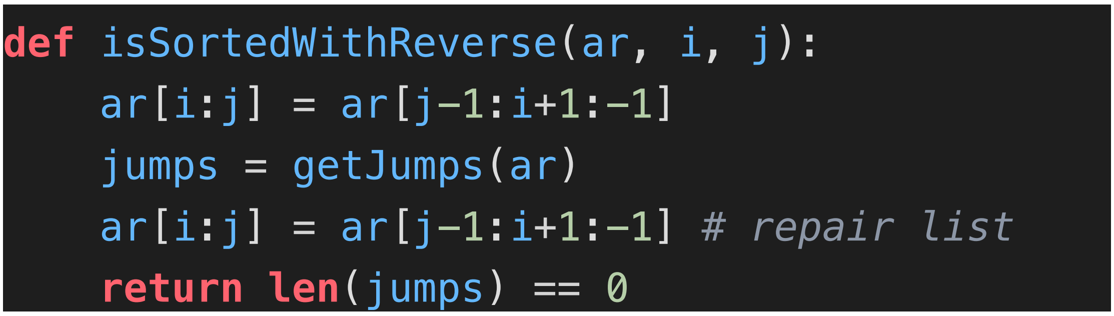
MORE DETAILS

Download problem statement
Download sample test cases
Suggest Edits

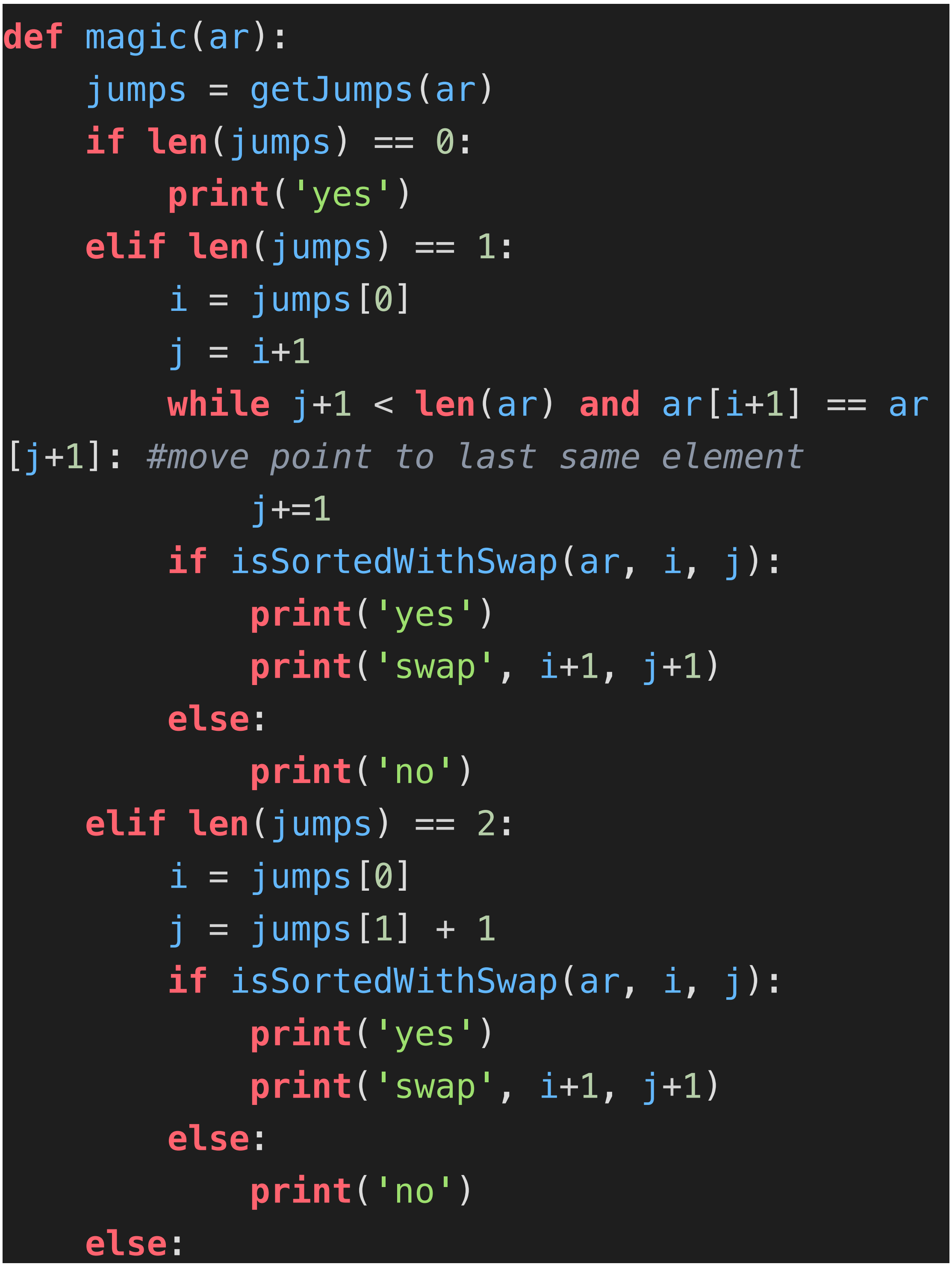
Code:



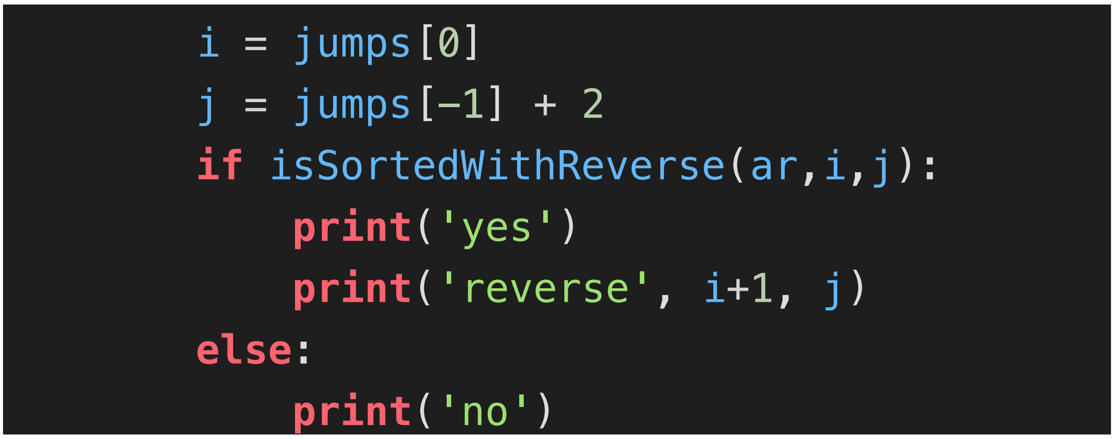
```python
def getJumps(ar):
    jumps = []
    for i in range(len(ar)-1):
        if ar[i] > ar[i+1]:
            jumps.append(i)
    return jumps

def isSortedWithSwap(ar, i, j):
    ar[i], ar[j] = ar[j], ar[i]
    jumps = getJumps(ar)
    ar[i], ar[j] = ar[j], ar[i] # repair list
    return len(jumps) == 0
```



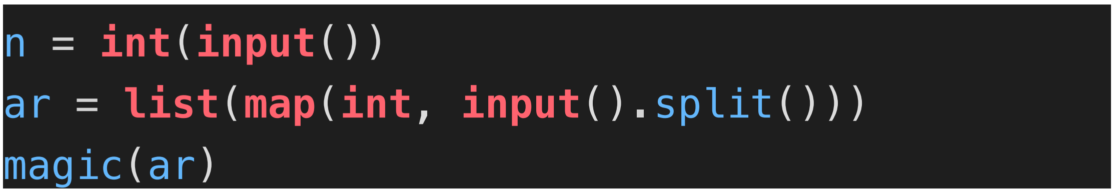
```
def isSortedWithReverse(ar, i, j):
    ar[i:j] = ar[j-1:i+1:-1]
    jumps = getJumps(ar)
    ar[i:j] = ar[j-1:i+1:-1] # repair list
    return len(jumps) == 0
```



```
def magic(ar):
    jumps = getJumps(ar)
    if len(jumps) == 0:
        print('yes')
    elif len(jumps) == 1:
        i = jumps[0]
        j = i+1
        while j+1 < len(ar) and ar[i+1] == ar[j+1]: #move point to last same element
            j+=1
        if isSortedWithSwap(ar, i, j):
            print('yes')
            print('swap', i+1, j+1)
        else:
            print('no')
    elif len(jumps) == 2:
        i = jumps[0]
        j = jumps[1] + 1
        if isSortedWithSwap(ar, i, j):
            print('yes')
            print('swap', i+1, j+1)
        else:
            print('no')
    else:
```

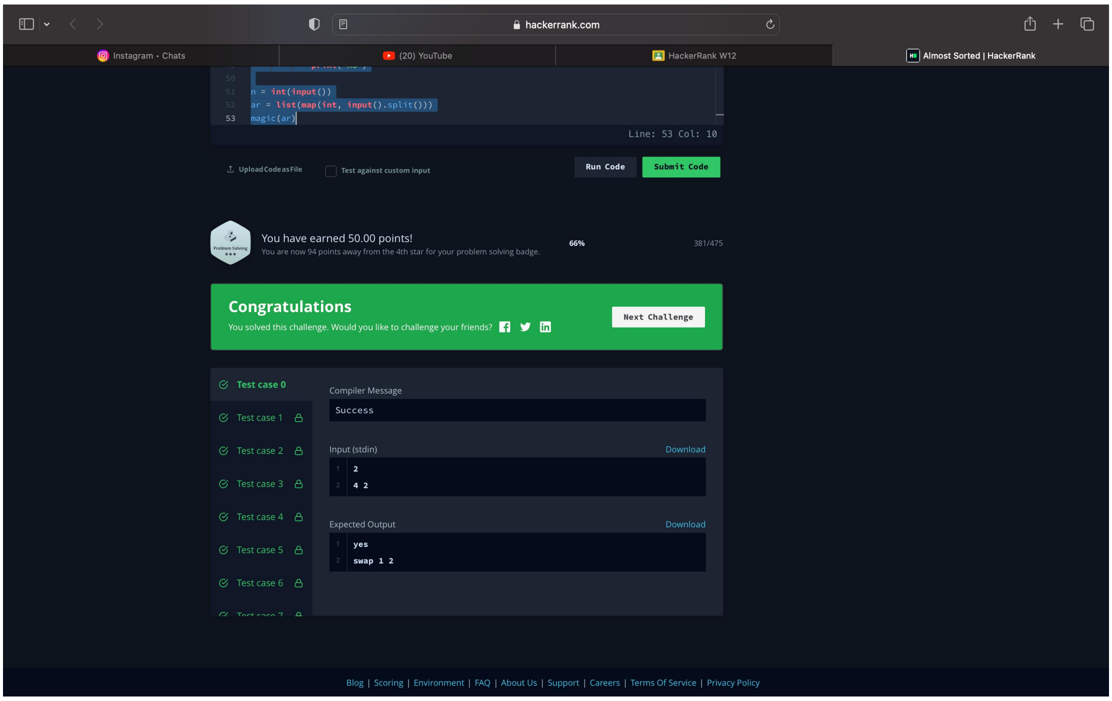


```
        i = jumps[0]
        j = jumps[-1] + 2
        if isSortedWithReverse(ar,i,j):
            print('yes')
            print('reverse', i+1, j)
        else:
            print('no')
```



```
n = int(input())
ar = list(map(int, input().split()))
magic(ar)
```

Output:



```
n = int(input())
ar = list(map(int, input().split()))
magic(ar)
```

You have earned 50.00 points!

You are now 94 points away from the 4th star for your problem solving badge.

Congratulations

You solved this challenge. Would you like to challenge your friends?

Compiler Message

```
Success
```

Input (stdin)

```
2
4 2
```

Expected Output

```
yes
swap 1 2
```

Download
Download
Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy