100

Venkata Naga Sri Sai Pranavi Kolipaka

Other

PDF generated at: 27 Jun 2025 18:46:47 UTC

View this report on HackerRank

Score

100% • 80 / 80

scored in TIP102: Unit 4 Version A (Standard) - Summer 2025 in 65 min 30 sec on 27 Jun 2025 10:39:18 PDT

Candidate Information

Email	kolipakavnssaipranavi@gmail.com
Test	TIP102: Unit 4 Version A (Standard) - Summer 2025
Candidate Packet	View
Taken on	27 Jun 2025 10:39:18 PDT
Time taken	65 min 30 sec/ 90 min
Personal Member ID	129054
Email Address with CodePath	kolipakavnssaipranavi@gmail.com
Github username with CodePath	Pranavi2002
Invited by	CodePath

Suspicious Activity detected

Code similarity

Code similarity • 1 question

Skill Distribution

There is no associated skills data that can be shown for this assessment

Tags Distribution

There is no associated tags data that can be shown for this assessment

Questions

Coding Questions • 60 / 60

Status	No.	Question	Time Taken	Skill	Score	Code Quality
✓	1	Move Zeroes Coding	10 min 1 sec	-	20/20 ⚐	-

✓	2	Check if Array Is Sorted and Rotated Coding	19 min 23 sec	-	20/20	-
✓	3	Subarray Sum Equals K Coding	19 min 29 sec	-	20/20	-

Multiple Choice + Debugging • 20 / 20

Status	No.	Question	Time Taken	Skill	Score	Code Quality
✓	4	What is the time complexity of complex_function()? Multiple Choice	2 min 49 sec	-	5/5	-
✓	5	Which of the following data structures/algorithmic techniques would be most appropriate for implementing this function? Multiple Choice	4 min 7 sec	-	5/5	-
✓	6	What is the output of the following code? Multiple Choice	3 min 2 sec	-	5/5	-
✓	7	Fing the bug Coding	6 min 5 sec	-	5/5	-

1. Move Zeroes

Correct

Coding

Question description

Given a 0-indexed integer array `nums` of length `n` and an integer `target`, write a function `count_pairs()` that returns the number of pairs `(i, j)` where `0 <= i < j < n`, and `nums[i] + nums[j] < target`

Input: nums = [-1,1,2,3,1], target = 2
Output: 3
Explanation: There are 3 pairs of indices that satisfy the conditions in the statement:
- (0, 1) since 0 < 1 and nums[0] + nums[1] = 0 < target
- (0, 2) since 0 < 2 and nums[0] + nums[2] = 1 < target
- (0, 4) since 0 < 4 and nums[0] + nums[4] = 0 < target

Note that (0, 3) is not counted since nums[0] + nums[3] is not strictly less than the target.

Candidate's Solution

Language used: **Python 3**

```
#!/bin/python

import math
import os
import random
import re
import sys
import ast


#
# Complete the 'count_pairs' function below.
#
# The function is expected to return an INTEGER.
# The function accepts following parameters:
#  1. INTEGER_ARRAY nums
#  2. INTEGER target

# Psuedo-code:
#   take a variable count = 0
#   loop i starts from 0 until n - 1
#   loop j starts from i+1 until n
#   if nums[i] + nums[j] < target
#   count += 1
# return count

def count_pairs(nums, target):
    # Write your code here
    count = 0
    for i in range(0, len(nums) - 1):
        for j in range(i+1, len(nums)):
            if nums[i] + nums[j] < target:
                count += 1
    return count

if __name__ == '__main__':
    outfile = open(os.environ['OUTPUT_PATH'], 'w')
    input_data = sys.stdin.read().strip().splitlines()

    results = []

    for line in input_data:
        # Convert the line to list of lists
        nums, target = eval(line)
        result = count_pairs(nums, target)
        results.append(result)

    for res in results:
        outfile.write(str(res) + '\n')
    outfile.close()
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Hidden	Success	0	0.0279 sec	10.9 KB
Single Pair Less Than Target	Easy	Hidden	Success	0	0.0298 sec	10.9 KB

All Elements Form Valid Pairs	Easy	Hidden	Success	0	0.0308 sec	10.8 KB
No Valid Pairs	Easy	Hidden	Success	0	0.0284 sec	10.9 KB
Empty Array	Easy	Hidden	Success	0	0.0275 sec	10.9 KB
Single Element	Easy	Hidden	Success	0	0.0282 sec	10.9 KB
All Elements Greater Than or Equal to Target	Easy	Hidden	Success	0	0.0279 sec	10.9 KB
All Elements Less Than Target, Multiple Occurrences	Easy	Hidden	Success	0	0.0299 sec	10.6 KB
Duplicates with Same Result	Easy	Hidden	Success	0	0.0299 sec	10.9 KB
Pass/Fail Case	Easy	Hidden	Success	20	0.0352 sec	10.9 KB

No comments.

2. Check if Array Is Sorted and Rotated

✓ Correct

Coding

Question description

Given an array `nums`, return `True` if the array was originally sorted in non-decreasing order, then rotated **some** number of positions (including zero). Otherwise, return `False`.

There may be **duplicates** in the original array.

Note: Rotating an array by a certain number of positions means shifting each element that many positions to the right (or left). When shifted one position to the right, the last element in the array becomes the first element in the array. Formally, an array `A` rotated by `x` positions results in an array `B` of the same length such that `A[i] == B[(i+x) % A.length]`, where `%` is the modulo operation.

Example 1:
Input: nums = [3,4,5,1,2]
Output: True
Explanation: [1,2,3,4,5] is the original sorted array.
You can rotate the array by x = 3 positions to begin on the the element of value 3: [3,4,5,1,2].

Example 2:
Input: nums = [2,1,3,4]
Output: False
Explanation: There is no sorted array once rotated that can make nums.

Example 3:
Input: nums = [1,2,3]
Output: true
Explanation: [1,2,3] is the original sorted array.
You can rotate the array by x = 0 positions (i.e. no rotation) to make nums.

Candidate's Solution

Language used: **Python 3**

```python
#!/bin/python

import math
import os
import random
import re
import sys
import ast



#
# Complete the 'is_sorted_rotated' function below.
#
# The function is expected to return a BOOLEAN.
# The function accepts INTEGER_ARRAY nums as parameter.

#   Psuedo-code:
#   take length of nums array as n
#   initialize count to 0
#   loop i until n
#   if nums[i] > nums[(i+1) % n], means if the number is greater than its next number, check if the last num is greater than first num
#   increment count
#   count has to be less than or equal to 1 for the array to be sorted and rotated

def is_sorted_rotated(nums):
    # Write your code here
    n = len(nums)
    count = 0
    for i in range(n):
        if nums[i] > nums[(i+1) % n]:
            count += 1
            # if count > 1:
            #     return False
    # return True
    return count <= 1



if __name__ == '__main__':
    outfile = open(os.environ['OUTPUT_PATH'], 'w')
    input_data = sys.stdin.read().strip().split('\n')

    for line in input_data:
        nums = ast.literal_eval(line.strip())
        result = is_sorted_rotated(nums)
        outfile.write(str(result) + '\n')
    outfile.close()
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Basic Case	Easy	Hidden	Success	0	0.0286 sec	11 KB
Standard Case without Rotation	Easy	Hidden	Success	0	0.0271 sec	11 KB
No Rotation Needed	Easy	Hidden	Success	0	0.0286 sec	11 KB
Single Element	Easy	Hidden	Success	0	0.0283 sec	10.9 KB
Two Elements Sorted	Easy	Hidden	Success	0	0.0278 sec	11 KB
Two Elements Not Sorted	Easy	Hidden	Success	0	0.034 sec	10.9 KB
All Identical Elements	Easy	Hidden	Success	0	0.0287 sec	11 KB
Array Already Sorted with Rotation	Easy	Hidden	Success	0	0.0282 sec	11 KB
Empty Array	Easy	Hidden	Success	0	0.0276 sec	10.6 KB
Pass/Fail Case	Easy	Hidden	Success	20	0.0293 sec	10.9 KB

ⓘ No comments.

3. Subarray Sum Equals K

✓ Correct

Coding

Question description

Given an array of integers `nums` and an integer `k`, return the total number of continuous subarrays whose sum equals to `k`.

```
Example 1:
Input: [1,1,1], 2
Output: 2

Example 2:
Input: [1, 2, 3], 7
Output: 0
```

Candidate's Solution

Language used: Python 3

```python
#!/bin/python

import math
import os
import random
import re
import sys
import ast
import json


#
# Complete the 'subarray_sum' function below.
#
# The function is expected to return an INTEGER.
# The function accepts following parameters:
#  1. INTEGER_ARRAY nums
#  2. INTEGER k


def subarray_sum(nums, k):
    # Write your code here
    count = 0
    prefix_sum = 0
    prefix_map = {0: 1} # sum : frequency

    for num in nums:
        prefix_sum += num
        if prefix_sum - k in prefix_map:
            count += prefix_map[prefix_sum - k]
        prefix_map[prefix_sum] = prefix_map.get(prefix_sum, 0) + 1

    return count
if __name__ == '__main__':
    outfile = open(os.environ['OUTPUT_PATH'], 'w')
    input_data = sys.stdin.read().strip().splitlines()

    results = []

    for line in input_data:
        parts = json.loads(f"[{line}]")
        nums = parts[0]
        k = parts[1]
        result = subarray_sum(nums, k)
        results.append(result)

    for res in results:
        outfile.write(str(res) + '\n')
    outfile.close()
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Basic Case	Easy	Hidden	Success	0	0.031 sec	11 KB

Target Sum Found	Easy	Hidden	Success	0	0.0324 sec	11 KB
Multiple Subarrays for Target 0	Easy	Hidden	Success	0	0.0323 sec	10.9 KB
Single Element Equal to k	Easy	Hidden	Success	0	0.0306 sec	10.9 KB
No Valid Subarrays	Easy	Hidden	Success	0	0.0449 sec	10.8 KB
Negative Numbers	Easy	Hidden	Success	0	0.0307 sec	10.9 KB
All Zeros	Easy	Hidden	Success	0	0.0418 sec	11 KB
Target Sum Greater than Sum of All Elements	Easy	Hidden	Success	0	0.033 sec	10.9 KB
Large k Value	Easy	Hidden	Success	0	0.0335 sec	10.9 KB
Alternating Positive and Negative Values	Easy	Hidden	Success	0	0.0314 sec	11 KB
All Negative Elements, Target is Negative	Easy	Hidden	Success	0	0.0304 sec	10.9 KB

Pass/Fail Case	Easy	Hidden	Success	20	0.0292 sec	11 KB

ⓘ No comments.

4. What is the time complexity of complex_function()?

✓ Correct

Multiple Choice

Question description

What is the time complexity of `complex_function()`?

```
def complex_function(nums):
    total = 0
    for i in range(len(nums)):
        for j in range(i, len(nums)):
            if nums[i] < nums[j]:
                total += nums[j] - nums[i]
    return total
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

○ O(1)

○ O(log n)

○ O(n)

● O(n^2) ✓

ⓘ No comments.

5. Which of the following data structures/algorithmic techniques would be most appropriate for implementing this function?

✓ Correct

Multiple Choice

Question description

The function `find_mode()` takes in a list of elements `lst` and returns the mode of the the list. If there are multiple modes, it returns them all as a list. If there is a single mode it returns the mode as a single value. The mode is the element that occurs most often in the list.

Which of the following data structures/algorithmic techniques would be most appropriate for implementing this function?

```
# Example usage:
nums1 = [1, 2, 2, 3, 4]
print(find_mode(nums1))  # Output: 2

nums2 = [1, 2, 2, 3, 3, 4]
print(find_mode(nums2))  # Output: [2, 3]

nums3 = [1, 2, 3, 4, 5]
print(find_mode(nums3))  # Output: [1, 2, 3, 4, 5] (all values are unique)
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

○ stack

○ queue

○ two pointer

● frequency map ✓

ⓘ No comments.

6. What is the output of the following code?

✓ Correct

Multiple Choice

Question description

```
def process_numbers(nums, threshold):
    stack = []
    for num in nums:
        if num < threshold:
            stack.append(num)
        elif num <= 10 and stack:
            stack.pop()
    return stack
```

```
print(process_numbers([3, 5, 1, 9, 6, 15], 8))
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

- ○ [5, 1, 6]
- ● [3, 5, 6] ✓
- ○ [3, 5, 1, 6]
- ○ [3, 5, 1]

No comments.

7. Fing the bug

✓ Correct

Coding

Question description

The following code incorrectly implements `is_anagram()` . When implemented correctly, `is_anagram()` should accept two strings `s` and `t` and return `True` if `s` and `t` are anagrams of each other. Otherwise, it should return `False` .

An anagram is a word or phrase formed by rearranging the letters of a different word or phrase, using all the original letters exactly once.

Identify any bug(s) within the given implementation and correct the code so that it successfully passes the provided test cases.

Candidate's Solution

Language used: **Python 3**

```
#!/bin/python

import math
import os
import random
import re
import sys
import ast

from collections import defaultdict
def is_anagram(s, t):
    if len(s) != len(t):
        return False

    s_freq = defaultdict(int)
    t_freq = defaultdict(int)

    for char in s:
        s_freq[char] += 1

    for char in t:
        t_freq[char] += 1

    return s_freq == t_freq
if __name__ == '__main__':
    input_data = sys.stdin.read().strip()
    input_list = ast.literal_eval(input_data)

    s = input_list[0]
    t = input_list[1]

    result = is_anagram(s, t)
    print(result)
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Pass/Fail Case	Easy	Hidden	Success	5	0.0275 sec	10.9 KB

ⓘ No comments.