Day 2 Codelity

**Lesson 1**

https://app.codility.com/demo/results/trainingXJBWC7-T4Y/

**Lesson 2**

https://app.codility.com/demo/results/training6VUBVJ-ZVH/

https://app.codility.com/demo/results/training4FMC79-9XX/

**Lesson 3**

https://app.codility.com/demo/results/trainingSWQ7FK-CAF/

https://app.codility.com/demo/results/trainingMUPAKA-2HV/

https://app.codility.com/demo/results/training5ZEK4E-C9X/

**BINARY GAP**

­Code

using System;

// you can also use other imports, for example:

// using System.Collections.Generic;

// you can write to stdout for debugging purposes, e.g.

// Console.WriteLine("this is a debug message");

class Solution {

public int solution(int N) {

int n = N;

string binary="";

while (n > 0)

{

binary = (n % 2) + binary;

n /= 2;

}

int max = 0;

int currMax = 0;

bool gateOpen = false;

foreach (char a in binary)

{

if (gateOpen && a == '0') {

currMax++;

}

if (!gateOpen && a == '1')

{

gateOpen = true;

}

else if (gateOpen && a == '1') {

max = Math.Max(max, currMax);

currMax = 0;

}

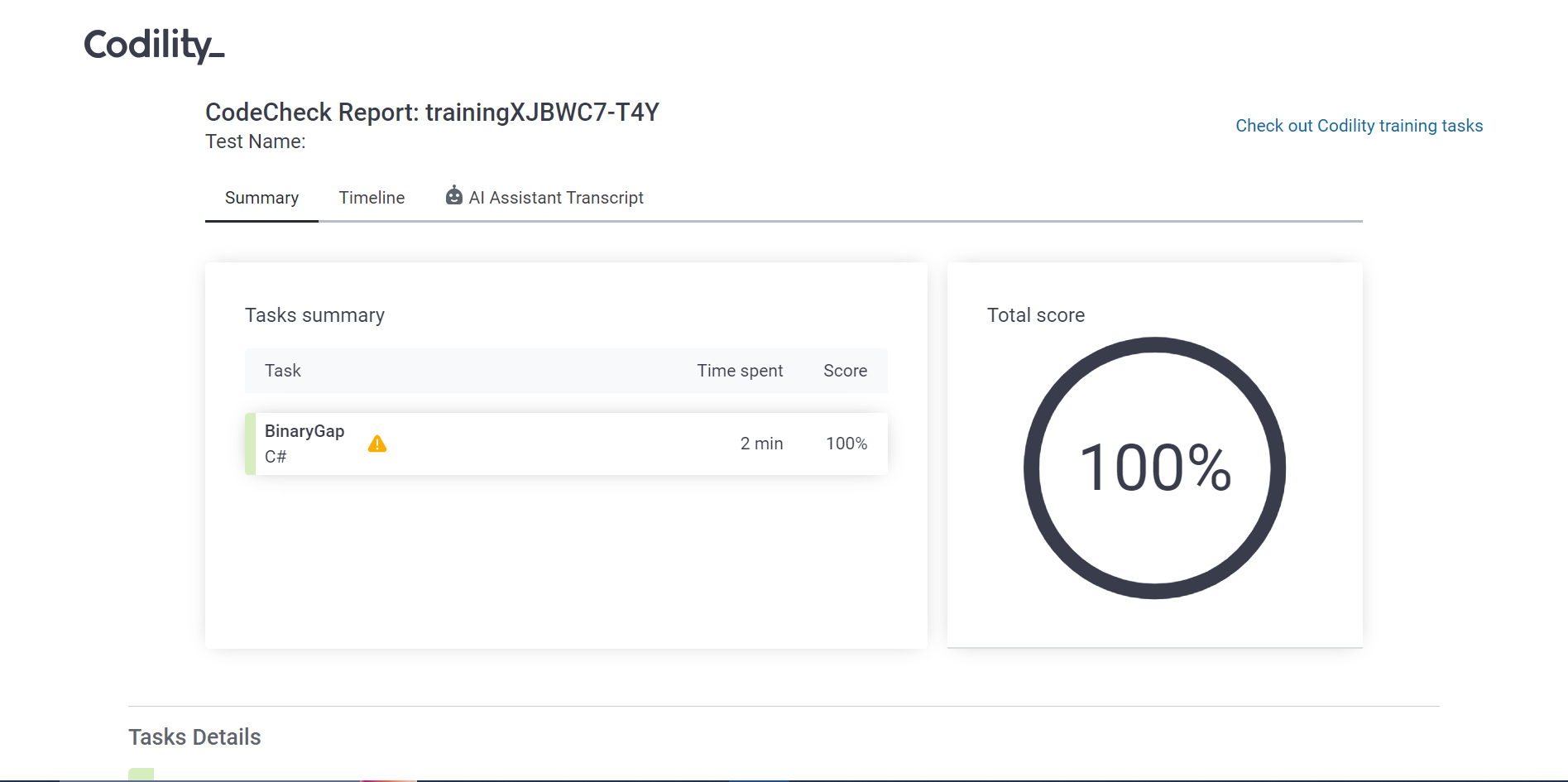
}

return max;

}

}

Output



**CYCLIC ROTATION**

­Code

class Solution {

public static void reverse(int nums[], int start, int end){

while(start < end){

int temp = nums[start];

nums[start] = nums[end];

nums[end] = temp;

start++;

end--;

}

}

public int[] solution(int[] A, int K) {

if(A.length == 0) return A;

K %= A.length;

reverse(A, 0, A.length - 1);

reverse(A, 0, K - 1);

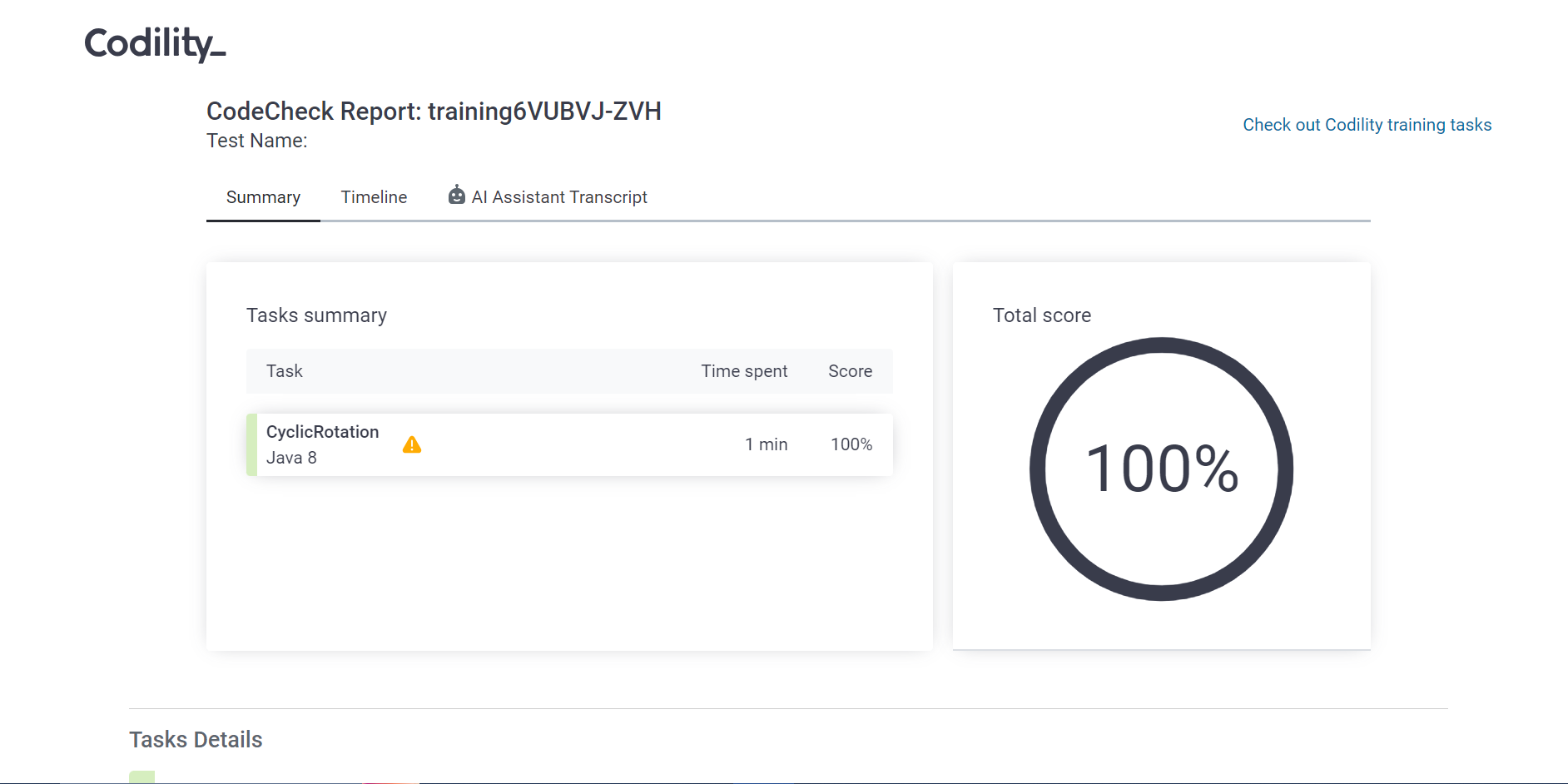
reverse(A, K, A.length - 1);

return A;

}

}

Output



**OddOccurrencesInArray**

­Code  
using System;

using System.Collections.Generic;

class Solution {

public int solution(int[] A) {

Dictionary<int, int> dict = new Dictionary<int, int>();

for (int i = 0; i < A.Length; i++) {

if(dict.ContainsKey(A[i])) {

dict[A[i]] = dict[A[i]] + 1;

}

else {

dict.Add(A[i], 1);

}

}

int ans = 0;

foreach(KeyValuePair<int, int> entry in dict) {

if(entry.Value == 1) {

ans = entry.Key;

}

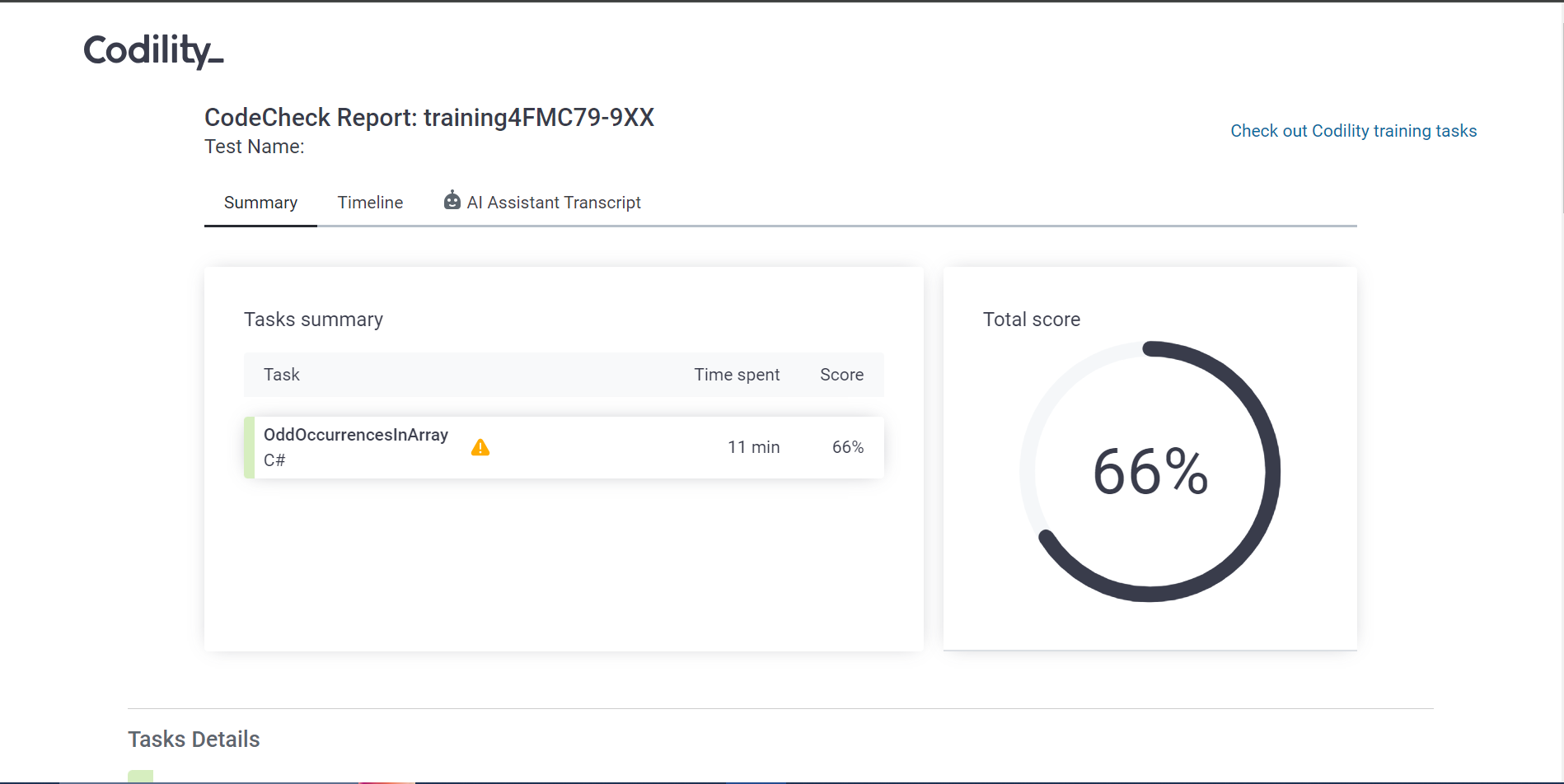
}

return ans;

}

}

Output



**TapeEquilibrium**

­Code  
def solution(A):

n = len(A)

prefix\_sums = [0] \* (n + 1)

for i in range(n):

prefix\_sums[i + 1] = prefix\_sums[i] + A[i]

min\_diff = float('inf')

for i in range(n):

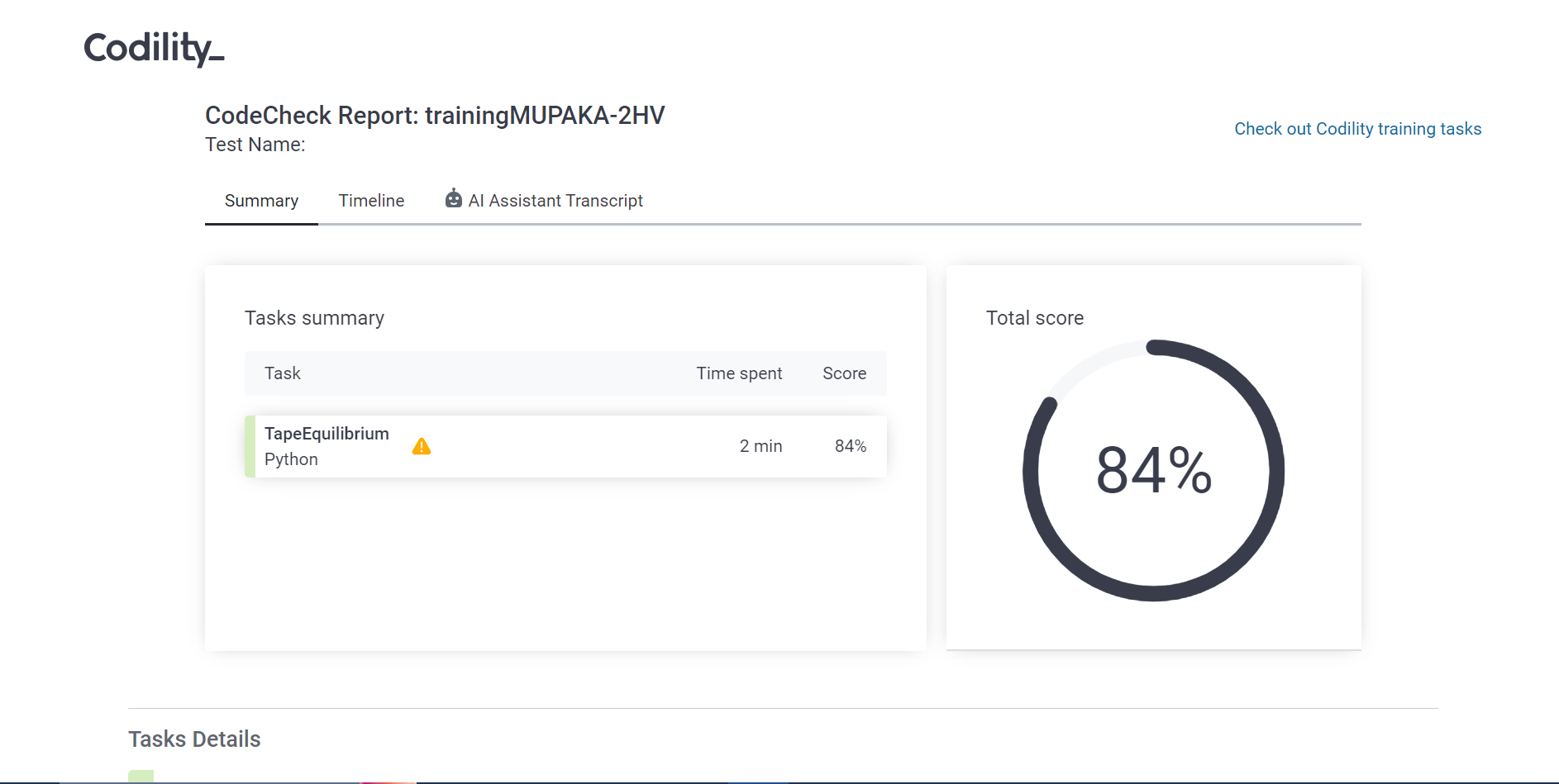
left\_sum = prefix\_sums[i]

right\_sum = prefix\_sums[n] - prefix\_sums[i]

min\_diff = min(min\_diff, abs(left\_sum - right\_sum))

return min\_diff

Output



**PermMissingElem**

­Code  
def solution(A):

N = len(A)

expected\_sum = (N + 1) \* (N + 2) // 2

actual\_sum = sum(A)

return expected\_sum - actual\_sum

Output

