

Candidate Report: trainingMQYA6B-987

[Check out Codility training tasks](#)

Test Name:

Summary

Timeline

Tasks summary

Task	Time spent	Score
ReplacingBooks Go	1 min	53%

Total score


53%

Tasks Details

1. ReplacingBooks

Given a list of integers, return the maximum number of consecutive integers equal to each other after replacing at most K of them.

Medium

Task Score

53%

Correctness

57%

Performance

50%

Task description

There are N obligatory books in a school program syllabus. The program also defines the order in which books should be read. Each book comes from a specific age, such as the enlightenment or the baroque period. The more books in a row the students read from any given age, the more they learn about it. Moreover, if they read a book from a different age, they will get distracted.

Teachers are allowed to replace K books from the program with alternatives. They want students to learn as much as possible from a single age (although they have not picked a particular specific age). The amount learned can be measured by the number of consecutive books from the same age read by the students. What is the maximum number of consecutive books from the same age after replacing at most K of them?

Note that the new books (after replacement) can be any books from the chosen age. They do not need to be listed in the syllabus,

Solution

Programming language used: Go

Total time used: 1 minutes ?

Effective time used: 1 minutes ?

Notes: not defined yet

Task timeline



so the teacher can always find K books from the same age.

20:36:45

20:37:24

Write a function:

```
func Solution(A []int, K int) int
```

that, given an array of integers A of length N, representing the ages of consecutive books from the school program syllabus, and an integer K, returns the maximum number of consecutive books from the same age after replacing at most K of them.

Examples:

1. Given A = [1, 1, 3, 4, 3, 3, 4] and K = 2, the function should return 5. Teachers can replace books from age 4 with books from age 3.

2. Given A = [4, 5, 5, 4, 2, 2, 4] and K = 0, the function should return 2. Teachers are not allowed to replace any books.

3. Given A = [1, 3, 3, 2] and K = 2, the function should return 4. Teachers can replace all the books from other ages with books from age 3.

Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [1..100,000];
- K is an integer within the range [0..N];
- each element of array A is an integer within the range [1..100,000].

Copyright 2009–2021 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Code: 20:37:24 UTC, go,
final, score: 53

[show code in pop-up](#)

```
1 package solution
2
3 func max(x, y int) int {
4     if x > y {
5         return x
6     }
7     return y
8 }
9
10 func Solution(arr []int, k int) int {
11     n := len(arr)
12
13     numIndices := make(map[int][]int)
14     for i, num := range arr { // O(N)
15         numIndices[num] = append(numIndices[num], i)
16     }
17
18     // Now find the best solution for each unique number
19     // i.e. for each number, what is the max. consecutive sequence
20     // we can get with at most k replacements. We will iterate
21     // over every possible starting gap sequence.
22     ans := 1
23     for num := range numIndices { // O(M)
24         indices := numIndices[num]
25         l := len(indices)
26
27         var currentGap int
28         var currentGapDelta int
29         for begin := range indices { // O(N)
30             numUsed := 0
31             total := 0
32             terminated := false
33             for i := begin; !terminated && i < l; i++ {
34                 // Consider the most of the current gap sequence
35                 if i == 0 {
36                     currentGap = indices[i] - begin
37                     total++ // For the one occurrence
38                 } else {
39                     if currentGap <= k - numUsed {
40                         currentGapDelta = currentGap - numUsed
41                     } else {
42                         currentGapDelta = k - numUsed
43                         terminated = true
44                     }
45                 } else {
46                     currentGap = indices[i] - indices[i-1]
47                 }
48                 if currentGap <= k - numUsed {
49                     if total == 0 {
50                         total += 2 // Since we have two occurrences
51                     } else {
52                         total++ // Only the new one
53                     }
54                     currentGapDelta = currentGap - numUsed
55                 } else {
56                     if total == 0 {
57                         total++
58                     }
59                     // The entire gap cannot be replaced
60                     // max possible as we have k replacements
61                     currentGapDelta = k - numUsed
62                     terminated = true
63                 }
64             }
65             total += currentGapDelta
66             numUsed += currentGapDelta
67         } // for i
```

```
68
69         // Consider the trailing gap too.
70         if !terminated {
71             currentGap = n - 1 - indices[l-
72             if currentGap <= k-numUsed {
73                 currentGapDelta = currentG
74             } else {
75                 currentGapDelta = k - numUs
76                 terminated = true
77             }
78             total += currentGapDelta
79             numUsed += currentGapDelta
80         }
81
82         ans = max(ans, total)
83     } // for begin
84 } // for num - total O(M * N * N)
85
86 return ans
87 }
```

Analysis summary

The following issues have been detected: wrong answers, timeout errors.

Analysis

Detected time complexity: **$O(N \cdot \log(N)^2)$**
or **$O(N^2)$**

expand all	Example tests	
▶	example1 First example test.	✓ OK
▶	example2 Second example test.	✓ OK
▶	example3 Third example test.	✓ OK
expand all	Correctness tests	
▶	small_random Small random tests.	✓ OK
▶	distinct All values are distinct.	✓ OK
▶	k_zero Tests where K = 0.	✓ OK
▶	answer_all Tests where the best interval is equal to the entire input.	✓ OK
▶	start_with_wrong Tests where best interval does not start or does not end with book which should be chosen for optimal result.	✗ WRONG ANSWER got 16 expected 18
expand all	Correctness/performance tests	
▶	least_common_age_dominates Tests where the overall least common	✗ WRONG ANSWER got 4 expected 6

age dominates the best interval.		
▶	large_big_answer Tests with big answers.	✗ TIMEOUT ERROR Killed. Hard limit reached: 6.000 sec.
▶	max_n Tests with maximum array length.	✗ TIMEOUT ERROR running time: 5.512 sec., time limit: 0.100 sec.
▶	max_distinct_or_all_same Maximum tests with all values either distinct or identical.	✗ TIMEOUT ERROR Killed. Hard limit reached: 6.000 sec.
expand all Performance tests		
▶	medium_random Medium random tests.	✓ OK
▶	semilarge_random Semi-large random tests to distinguish between $O(n^2)$ and $O(n \cdot \log(n)^2)$ solutions.	✓ OK
▶	semilarge_max_ans Semi-large tests with maximum answer.	✓ OK
▶	large_random Large random tests.	✗ TIMEOUT ERROR running time: 0.388 sec., time limit: 0.100 sec.

The PDF version of this report that may be downloaded on top of this site may contain sensitive data including personal information. For security purposes, we recommend you remove it from your system once reviewed.