228. Summary Ranges

April 2, 2016 | 21K views

Given a sorted integer array without duplicates, return the summary of its ranges.

Example 1:

```
Input:  [0,1,2,4,5,7]
Output: ["0->2","4->5","7"]
Explanation: 0,1,2 form a continuous range; 4,5 form a continuous range.
```

Example 2:

```
Input:  [0,2,3,4,6,8,9]
Output: ["0","2->4","6","8->9"]
Explanation: 2,3,4 form a continuous range; 8,9 form a continuous range.
```

Solution

Intuition

A range covers consecutive elements. If two adjacent elements have difference larger than 1, the two elements does not belong to the same range.

Algorithm

To summarize the ranges, we need to know how to separate them. The array is sorted and without duplicates. In such array, two adjacent elements have difference either 1 or larger than 1. If the difference is 1, they should be put in the same range; otherwise, separate ranges.

We also need to know the start index of a range so that we can put it in the result list. Thus, we keep two indices, representing the two boundaries of current range. For each new element, we check if it extends the current range. If not, we put the current range into the list.

Don't forget to put the last range into the list. One can do this by either a special condition in the loop or putting the last range in to the list after the loop.

Java

```
public class Solution {
    public List<String> summaryRanges(int[] nums) {
        List<String> summary = new ArrayList<>();
        for (int i = 0, j = 0; j < nums.length; ++j) {
            // check if j + 1 extends the range [nums[i], nums[j]]
            if (j + 1 < nums.length && nums[j + 1] == nums[j] + 1)
                continue;
            // put the range [nums[i], nums[j]] into the list
            if (i == j)
                summary.add(nums[i] + "");
            else
                summary.add(nums[i] + "->" + nums[j]);
            i = j + 1;
        }
        return summary;
    }
}
```

Java (Alternative)

```
public class Solution {
    public List<String> summaryRanges(int[] nums) {
        List<String> summary = new ArrayList<>();
        for (int i, j = 0; j < nums.length; ++j){
            i = j;
            // try to extend the range [nums[i], nums[j]]
            while (j + 1 < nums.length && nums[j + 1] == nums[j] + 1)
                ++j;
            // put the range into the list
            if (i == j)
                summary.add(nums[i] + "");
            else
                summary.add(nums[i] + "->" + nums[j]);
        }
        return summary;
    }
}
```

Complexity Analysis

- Time complexity : $O(n)$. Each element is visited constant times: either in comparison with neighbor or put in the result list.
- Space complexity : $O(1)$. All the auxiliary space we need is the two indices, if we don't count the return list.

Rate this article:

Previous Next

Comments: 11 Sort By

Type comment here... (Markdown is supported)

Preview Post

arghya ★ 79 August 7, 2017 2:56 AM
I agree with @fishercoder. I believe this should be marked as "easy".
50 Share Reply
SHOW 2 REPLIES

fishercoder ★ 774 December 30, 2016 1:55 AM
Just thinking this problem might be labeled as EASY instead of MEDIUM. the idea and intuition are both pretty straightforward. It's a good one for beginners. Just my thoughts. Thanks. ^ ^
22 Share Reply

chrislzm ★ 1461 February 23, 2018 11:25 AM
There's a faster solution with best case O(1) time.

We take advantage of the fact that there are no duplicates in the array. If the difference in value between two array elements is equal to the difference in their indexes, then we immediately know that it's a range and there's no need to iterate through it.

Read More
9 Share Reply
SHOW 3 REPLIES

floribon ★ 50 March 28, 2017 3:23 AM
+1, should be easy :) Instead of a for loop one can use a while and move two pointers, the slow one and the fast one.
6 Share Reply

algopps ★ 95 August 4, 2019 7:02 AM
This problem should be easy.
0 Share Reply

deleted_user ★ 198 February 14, 2019 1:08 AM
A nice clean solution:

```
class Solution {
    public List<String> summaryRanges(int[] nums) {
```

Read More
0 Share Reply

LearningMind ★ 213 January 1, 2019 7:14 AM
Why are you not considering return list for space complexity?
0 Share Reply
SHOW 1 REPLY

sonal_savaliya ★ 3 May 14, 2020 8:25 AM
Someone, please explain to me, what ranges[-1][-1] does?
I mean ranges[-1] will return the last value of the list. So, what value ranges[-1][-1] will return?
0 Share Reply

jjchoi08 ★ 1 January 16, 2020 11:37 PM
What could be an acceptable & brut-force python solutions for this question?
0 Share Reply

ping_pong ★ 826 November 16, 2019 4:44 PM
More cleaner version.

```
public List<String> summaryRanges(int[] nums) {
    if (nums == null || nums.length == 0) return new ArrayList<>();
```

Read More
0 Share Reply