Summary Ranges

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

For example, given `[0,1,2,4,5,7]`, return `["0->2","4->5","7"].`

Credits:
Special thanks to @jianchao.li.fighter for adding this problem and creating all test cases.

Solution 1

```
List<String> list=new ArrayList();
    if(nums.length==1){
        list.add(nums[0]+"");
        return list;
    }
    for(int i=0;i<nums.length;i++){
        int a=nums[i];
        while(i+1<nums.length&&(nums[i+1]-nums[i])==1){
            i++;
        }
        if(a!=nums[i]){
            list.add(a+"->"+nums[i]);
        }else{
            list.add(a+"");
        }
    }
    return list;
```

written by tanchiyuxx original link here

Solution 2

```
   vector<string> summaryRanges(vector<int>& nums) {
    const int size_n = nums.size();
    vector<string> res;
    if ( 0 == size_n) return res;
    for (int i = 0; i < size_n;) {
        int start = i, end = i;
        while (end + 1 < size_n && nums[end+1] == nums[end] + 1) end++;
        if (end > start) res.push_back(to_string(nums[start]) + "->" + to_string(
nums[end]));
        else res.push_back(to_string(nums[start]));
        i = end+1;
    }
    return res;
}
```

written by lchen77 original link here

Solution 3

Three versions of the same algorithm, all take O(n) time.

Solution 1

Just collect the ranges, then format and return them.

```python
def summaryRanges(self, nums):
    ranges = []
    for n in nums:
        if not ranges or n > ranges[-1][-1] + 1:
            ranges += [],
        ranges[-1][1:] = n,
    return ['->'.join(map(str, r)) for r in ranges]
```

Solution 2

A variation of solution 1, holding the current range in an extra variable `r` to make things easier. Note that `r` contains at most two elements, so the `in`-check takes constant time.

```python
def summaryRanges(self, nums):
    ranges, r = [], []
    for n in nums:
        if n-1 not in r:
            r = []
            ranges += r,
        r[1:] = n,
    return ['->'.join(map(str, r)) for r in ranges]
```

Solution 3

A tricky short version.

```python
def summaryRanges(self, nums):
    ranges = r = []
    for n in nums:
        if `n-1` not in r:
            r = []
            ranges += r,
        r[1:] = `n`,
    return map('->'.join, ranges)
```

About the commas :-)

Three people asked about them in the comments, so I'll also explain it here as well. I have these two basic cases:

```
ranges += [],
r[1:] = n,
```

Why the trailing commas? Because it turns the right hand side into a tuple and I get the same effects as these more common alternatives:

```
ranges += [[]]
or
ranges.append([])

r[1:] = [n]
```

Without the comma, ...

- `ranges += []` wouldn't add `[]` itself but only its elements, i.e., nothing.
- `r[1:] = n` wouldn't work, because my `n` is not an iterable.

Why do it this way instead of the more common alternatives I showed above? Because it's shorter and faster (according to tests I did a while back).

written by StefanPochmann original link here