TEST CASE 1 (on 87 line critical code): simply place System.out.println("Junction X) uniquely identifying each junction in the code

I reached 14 junctions in total and was content this was done accurately

TEST CASE 2 (on 87 line critical code): run through all 4 scenarios and record the junctions it traversed...

//int[] nums = new int[]{0,2,3,4,6,8,9}; //Test case 1 PASS Junction 2,6,10,11,12,3,5,14,13,4

//int[] nums = new int[]{-1}; //Test case 2 PASS Junction 1

//int[] nums = new int[]{0}; //Test case 3 PASS Junction 1

//int[] nums = new int[]{0,1,2,4,5,7}; //Test case 4 PASS Junction 2,3,5,6,10,14,9,7

However it can be seen that it has not passed through Junction 8.

```
68          }
69          else    ////not consecutive match
70          {
71              System.out.println("JUNCTION 6");
72              if (k==nums.length-2)
73              {
74                  System.out.println("JUNCTION 7");
75                  System.out.println("counter:" + counter + "------------------------------LAST TO LAST ITEM :" + nums[k]);
76
77                  if (counter==0)
78                  {
79                      System.out.println("JUNCTION 8");
80                      end = String.valueOf(nums[k]);
81                      System.out.println("HERE");
82                      sm.add(start+"->"+end);
83                      start=String.valueOf(nums[k+1]);
84                      sm.add(start);
85                  }
86                  else
```

I found it quite strange since I was patching my code inline with the outcome.. Perhaps as the complexity increased, it faded out of scope?

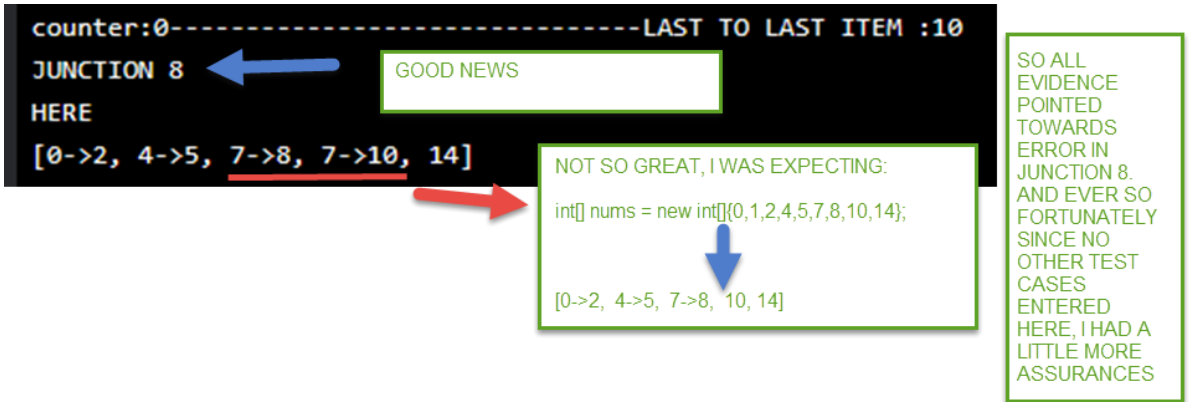
TEST CASE 3 (on 87 line critical code): So the most logical action entailed devising a scenario of numbers which will invoke this segment.

The only outstanding scenario is to explore having more standalone numbers at end of an existing scenario provided in the challenge...

```
//For some reason, no test cases above went through junction 8. All others are valid.
//So I devised a new test case below and it passes through it...
//new test case:
int[] nums = new int[]{0,1,2,4,5,7,8,10,14};
```

I have increased the new test case as follows. It can be seen that 4 - >5, **7, 8, 10, 14** are now all standalone....

This was the original challenge by Programiz. It can be seen that only 7 is detached from 4 -> 5



TEST CASE 4 (on 87 line critical code): Trying to understand the logistics in this area of code

TEST CASE 5 (on 87 line critical code): Running the code again with new scenario:

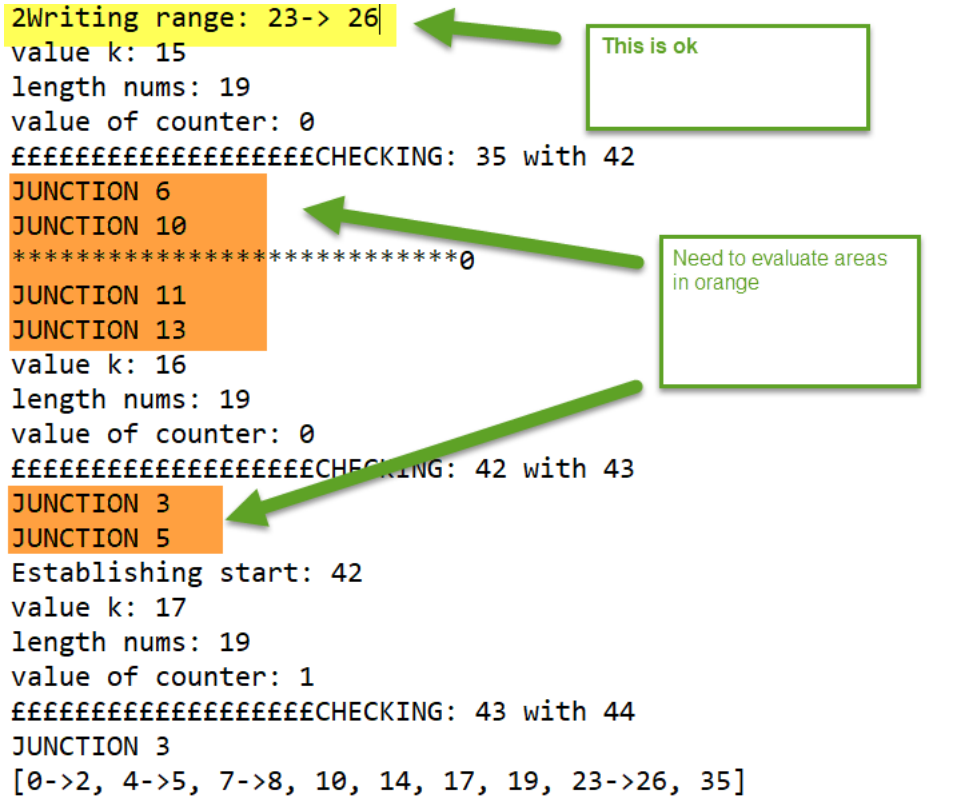
```
int[] nums = new int[]{0,1,2,4,5,7,8,10,14,17,19};
```

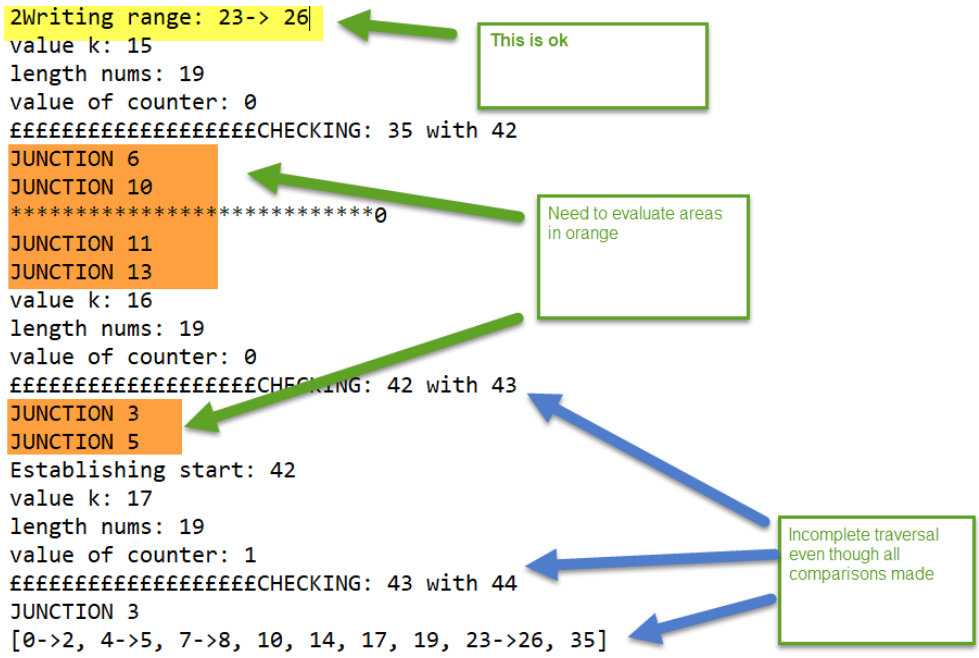
```
[0->2, 4->5, 7->8, 10, 14, 17, 19]
```

It has passed perfectly… The only extension would be to try summary ranges after 19 and then back to standalone range…

TEST CASE 6 (on 87 line critical code): extending beyond above range

```
//new test case:
int[] nums = new int[]{0,1,2,4,5,7,8,10,14,17,19,23,24,25,26,35,42,43,44};
```


2Writing range: 23-> 26
value k: 15
length nums: 19
value of counter: 0
ffffffffffffffffffCHECKING: 35 with 42
JUNCTION 6
JUNCTION 10
****************************0
JUNCTION 11
JUNCTION 13
value k: 16
length nums: 19
value of counter: 0
ffffffffffffffffffCHECKING: 42 with 43
JUNCTION 3
JUNCTION 5
Establishing start: 42
value k: 17
length nums: 19
value of counter: 1
ffffffffffffffffffCHECKING: 43 with 44
JUNCTION 3
[0->2, 4->5, 7->8, 10, 14, 17, 19, 23->26, 35]
This is ok
Need to evaluate areas in orange


2Writing range: 23-> 26
value k: 15
length nums: 19
value of counter: 0
ffffffffffffffffffCHECKING: 35 with 42
JUNCTION 6
JUNCTION 10
****************************0
JUNCTION 11
JUNCTION 13
value k: 16
length nums: 19
value of counter: 0
ffffffffffffffffffCHECKING: 42 with 43
JUNCTION 3
JUNCTION 5
Establishing start: 42
value k: 17
length nums: 19
value of counter: 1
ffffffffffffffffffCHECKING: 43 with 44
JUNCTION 3
[0->2, 4->5, 7->8, 10, 14, 17, 19, 23->26, 35]
This is ok
Need to evaluate areas in orange
Incomplete traversal even though all comparisons made

I introduced this area of code and with better debugging:

```
if ((nums[k]+1)==(nums[k+1]))
{
    System.out.println("JUNCTION 3");
    System.out.println("VAL COUNTER: " + counter);
    if (counter==0 && (k==nums.length-2))
    {
        System.out.println("JUNCTION 4");
        start=String.valueOf(nums[k]);
        end = String.valueOf(nums[k+1]);
        sm.add(start+"->"+end);
        System.out.println("5Writing range: " + start + "-> " + end);
        break;
    }

    if (counter!=0 && (k==nums.length-2))
    {
        end=String.valueOf(nums[k+1]);
        sm.add(start +"->"+end);
        System.out.println("6Writing range: " + start + "-> " + end);
        System.out.println("Terminating summary range join");
    }
```

Introduced this area of code to accommodate for termination of a summary range (not

```
value of counter: 0
ffffffffffffffffffCHECKING: 35 with 42
JUNCTION 6
JUNCTION 10
**************************0
JUNCTION 11
JUNCTION 13
6Writing Standalone: 35
value k: 16
length nums: 19
value of counter: 0
ffffffffffffffffffCHECKING: 42 with 43
JUNCTION 3
VAL COUNTER: 0
JUNCTION 5
Establishing start: 42
value k: 17
length nums: 19
value of counter: 1
ffffffffffffffffffCHECKING: 43 with 44
JUNCTION 3
VAL COUNTER: 1
Terminating summary range join
[0->2, 4->5, 7->8, 10, 14, 17, 19, 23->26, 35, 42->44]
```

So I am content now, I will just run one more scenario.

TEST CASE (on 87 line critical code):

```
int[] nums = new int[]{0,1,2,4,5,7,8,10,14,17,19,23,24,25,26,35,42,47};
```

```
[0->2, 4->5, 7->8, 10, 14, 17, 19, 23->26, 35, 42, 47]
```

TEST CASE (on 58 line critical code): All original challenges as per Programiz and my extension scenarios:
PASS

I am now going over my original code and will run the newly devised test cases there. But I will build up slightly slowly

TEST CASE (on 58 line critical code):

```
int[] nums = new int[]{0,1,2,4,5,7,8,10,14,17,19};
```

```
Value of counter: 0
Standalone: 17
[0->2, 4->5, 7->8, 10, 14, 17]
```

I added following code to accommodate for missed out 19

```
if (counter==0)
{
    sm.add(String.valueOf(nums[k]));
    System.out.println("Standalone: " + nums[k]);

    //it now declares the start to be the next number in the array
    start = String.valueOf(nums[k+1]);


    //NEW CODE
    if (k==nums.length-2)
    {
        sm.add(String.valueOf(start));
    }
```

TEST CASE (on 58 line critical code): Retesting the above = PASS

```
int[] nums = new int[]{0,1,2,4,5,7,8,10,14,17,19};
```

```
[0->2, 4->5, 7->8, 10, 14, 17, 19]
```

TEST CASE (on 58 line critical code): I added a few more standalone ranges

```
int[] nums = new int[]{0,1,2,4,5,7,8,10,14,17,19,25,29,30};
```

```
[0->2, 4->5, 7->8, 10, 14, 17, 19, 25, 29->30]
```

I will now try the same additional official test cases from my other code:

TEST CASE (on 58 line critical code): ALL PASSED

```
//INITIAL CHALLENGE
//int[] nums = new int[]{0,2,3,4,6,8,9};   //Test case 1
//int[] nums = new int[]{-1};   //Test case 2
//int[] nums = new int[]{0};   //Test case 3
//int[] nums = new int[]{0,1,2,4,5,7};   //Test case 4

//new test cases:
//int[] nums = new int[]{0,1,2,4,5,7,8,10,14,17,19};
//int[] nums = new int[]{0,1,2,4,5,7,8,10,14,17,19,25,29,30};
//int[] nums = new int[]{0,1,2,4,5,7,8,10,14};   //PASS
//int[] nums = new int[]{0,1,2,4,5,7,8,10,14,17,19,23,24,25,26,35,42,43,44};
//int[] nums = new int[]{0,1,2,4,5,7,8,10,14,17,19,23,24,25,26,35,42,47};
```