TEST CASE:

```
0.1f,0.2f,0.1f
```

```
CHECKING: 0.1 with 0.2
START: 0.1


0.1
0.2
0.0
TRACK1
K!=nums.length-2
********WRITTEN END----------------------------------------------------------------------: 0.1
This is counter at the moment:  0
It is not possible to trigger hasTransition if counter is 0 since can not see transition in opposite direction
CURRENT START: 0.1
-------------------------------------Stored start -> end: 0.1->0.2
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$CURRENT list: []
CURRENT START: 0.1
WHEN--------------------------------------------
CHECKING: 0.2 with 0.1
START: 0.1
0.1
0.2
0.1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
0.1
0.1
0.3
0.1
HEREEEEE
COUNTER IS 0------------
5Writing range: 0.2-> 0.1
[0.2->0.1]
```

We can see it has stored the value

I can see that in my code, I had this already planned as an issue, but for some reason including my documentation, I thought logic was too casual..
We know if k==nums.length-2 (this will be absolute last index for k)...
So need to evaluate in which circumstances it would be detrimental to write the store...

```
//in descending sequence
if (Math.abs(nums[k] - (nums[k+1] + difference)) <epsilon)
{
```

We know we are currently in descending sequence......
We know there is an ascending chain before
And if this is last execution, there can not be any circumstances in which it would be detrimental to write the store in my opinion....
It does not relate list being empty or not!

```
if (counter==0 && (k==nums.length-2))
{
    if(!(potentialfurtherAscendingBeyondThisStart=="") && !(potentialfurtherAscendingBeyondThisEnd=="") && isFirstOccurenceAscendingChain)
    {
        sm.add(potentialfurtherAscendingBeyondThisStart+"->"+nums[k]);
        System.out.println("-------------------2322USING STORED TO WRITE RANGE");
        System.out.println("9705Writing range: " + potentialfurtherAscendingBeyondThisStart + "-> " + nums[k]);
        potentialfurtherAscendingBeyondThisStart="";
        potentialfurtherAscendingBeyondThisEnd="";
    }

    //I feel we also need to trigger using stored value if   3.6, (3.5), 3.6 if the list is empty
    //this is a bit speculative but it seems quite logical.. otherwise we will run too far ahead without writin anything...

    //if (sm.isEmpty())
    //{
     //  S
    //    sm.add(potentialfurtherAscendingBeyondThisStart+"->"+potentialfurtherAscendingBeyondThisEnd);
     //   System.out.println("-------------------1980USING STORED TO WRITE RANGE");
      //  System.out.println("9705Writing range: " + potentialfurtherAscendingBeyondThisStart + "-> " + potentialfurtherAsce
    //    potentialfurtherAscendingBeyondThisStart="";
     //   potentialfurtherAscendingBeyondThisEnd="";
//    }
```

I can see this line of code appears when counter!=0

This is clearly not the case when it has reached nums.length==2 otherwise it would have written the store

```
                    }
    if ((Math.abs(nums[k] - (nums[k-1] + difference)) <epsilon) &&   (Math.abs(nums[k] - (nums[k+1] - difference))<epsilon))
                    {  System.out.println("Previous number less, next number greater");
                        if (!isFirstOccurenceAscendingChain)
                        {
                            start=String.valueOf(nums[k-1]);

                            isFirstOccurenceAscendingChain=true;
```

So to be safe, I need to include logic on the code below to perform action when isFirstOccurenceAscendingChain!=0 and perhaps also when sm.isEmpty()

```
if (sm.isEmpty() && !isFirstOccurenceAscendingChain
&& !(potentialfurtherAscendingBeyondThisStart=="")
&& !(potentialfurtherAscendingBeyondThisEnd==""))
{
    sm.add(potentialfurtherAscendingBeyondThisStart+"->"+nums[k]);
    System.out.println("-------------------1980USING STORED TO WRITE RANGE");
    System.out.println("43434Writing range: " + potentialfurtherAscendingBeyondThisStart + "-> " + potentialfurtherAscendingBeyondThisE
    potentialfurtherAscendingBeyondThisStart="";
    potentialfurtherAscendingBeyondThisEnd="";
}
```

```
[0.1->0.2, 0.2->0.1]
```

I will need to run through several test cases for peace of mind.
On paper, it just does not seem it will interfere...

```
336
337             start=String.valueOf(nums[k]);
338             end = String.valueOf(nums[k+1]);
339             sm.add(start+"->"+end);
340             System.out.println("5Writing range: " + start + "-> " + end);
341             isFirstOccurenceAscendingChain=false;
342             isFirstOccurenceAscendingChainNoTransition =false;
343             break;
```