Sub Sort

January 25, 2018

1 Question

Given an array of integers, write a method to find indices m and n such that if you sorted elements m through n, the entire array would be sorted. Minimize n - m (that is find the smallest such sequence).

2 Explanation and Algorithm

The question is asking to find the smallest part in an array we should sort in order for the entire array to be sorted. By knowing this we know that the array is divided into 3 different parts (or subpart). A right subpart (the right part of the array that is order, a middle part (the part that is unsorted), and the left subpart (the left part of the array that is sorted). These subpart are easy to generate. We start from the left and right sides, then work our way inward. When an element is out of order, then we have found the end of our increasing or decreasing subsequence. We know that the array should look like this: left subpart $<$ middle subpart $<$ right subpart. In other words, the minimum element of the middle subpart should be greater than every element in the left subpart AND the max of the middle subpart should be lesser than every element in the right subpart. What we do is we find the minimum element of the middle and right subparts; we also find the maximum element of the middle and left subparts. With these two elements, we find at which index they should be placed. Wherever the minimum element of the middle and right subparts should go is your m index, and wherever the maximum element of the middle and left subparts are is your n index.

3 Hints

1. Would it help to know the longest sorted sequence at the beginning and end?
2. We can think about the array as divided into three subarrays: LEFT, MIDDLE, RIGHT. LEFT and RIGHT are both sorted. The MIDDLE elements are in an arbitrary order. We need to expand MIDDLE until we could sort those elements and then have the entire array sorted.
3. Consider the three subarrays: LEFT, MIDDLE, RIGHT. Focus on just this question: Can you sort middle such that the entire array becomes sorted:? How would you check this?
4. In order to be able to sort MIDDLE and have the whole array become sorted, you need MAX(LEFT) $<=$ MIN(MIDDLE and RIGHT) and MAX(LEFT and MIDDLE) $<=$ MIN(RIGHT).
5. Can you expand the middle until the earlier condition is met?

4 Code

```
void findUnsortedSequence(int [] array){
   //find left subpart
    int end_left = findEndOfLeftSubsequence(array);
    if( end_left >= array.length -1) return; // Array is already sorted
    //find right subpart
    int start_right = findStartOfRightSubsequence(array);

    //get min and max
    int max_index = end_left;
    int min_index = start_right;

    for(int i=end_left+!; i<start_right; i++){
      if(array[i] < array[min_index]) min_index = i;
        if(array[i] < array[max_index]) max_index = i;
    }

    //slide left until less than array[min_index]
    int left_index = shrinkLeft(array, min_index, end_left);
    //slide right until greater than array[max_index]
    int right_index = shrinkRight(array, max_index, start_right);

    System.out.println(left_index + " " + right_index);
}

int findEndOfLeftSubsequence(int [] array){
      for(int i=1; i<array.length; i++){
      if(array[i]<array[i-1])
         return i-1;
    }
    return array.length -1;
}
int findStartOfRightSubsequence(int [] array){
      for(int i=array.length-2; i>=0; i--){
      if(array[i]>array[i+1])
         return i+1;
    }
    return 0;
}
int shrinkLeft(int [] array, int min_index, int start){
   int comp = array[min_index];
    for (int i=start-1; i>=0; i--) {
      if(array[i] <= comp) return i+1;
    }
    return 0;
}
int shrinkRight(int [] array, int min_index, int start){
   int comp = array[max_index];
    for (int i=start; i<array.length; i++) {
      if(array[i] >= comp) return i-1;
    }
    return array.length-1;
}
```

5 Big-O Analysis

You are worst case running through this O(N) times.

6 Source

Taken from Cracking the Coding Interview by Gayle Laakmann McDowell.