Solution Review: Re-arrange Sorted Array in Max/Min Form

This review provides a detailed analysis to solve the Re-arrange Sorted Array in Max/Min Form challenge.

We'll cover the following

Solution #1: Creating a New Array

```java
    //Create a result array to hold re-arranged version of given arr
    int[] result = new int[arr.length];

    int pointerSmall = 0;     //PointerSmall => Start of arr
    int pointerLarge = arr.length - 1;   //PointerLarge => End of arr

    //Flag which will help in switching between two pointers
    boolean switchPointer = true;

    for (int i = 0; i < arr.length; i++) {
      if (switchPointer)
        result[i] = arr[pointerLarge--]; // copy large values
      else
        result[i] = arr[pointerSmall++]; // copy small values

      switchPointer = !switchPointer;   // switching between samll and large
    }

    for (int j = 0; j < arr.length; j++) {
      arr[j] = result[j];    // copying to original array
    }
  }

  public static void main(String args[]) {

    int[] arr = {1, 2, 3, 4, 5, 6};
    System.out.print("Array before min/max: ");
    for (int i = 0; i < arr.length; i++)
      System.out.print(arr[i] + " ");
    System.out.println();

    maxMin(arr);

    System.out.print("Array after min/max: ");
    for (int i = 0; i < arr.length; i++)
      System.out.print(arr[i] + " ");
    System.out.println();
  }
}
```

Explanation

In this solution, we first create an empty array whose size is equal to the size of the original array, then we start iterating over the array and store the maximum number in the start of the array and then the minimum number next to it and so on. In the end, store the *result* array to the original array.

Time Complexity

The time complexity of this problem is $O(n)$ as the array is iterated over once.

Solution #2: Using $O(1)$ Extra Space

```
class CheckMaxMin {

  public static void maxMin(int[] arr) {
    int maxIdx = arr.length - 1;
    int minIdx = 0;
    int maxElem = arr[maxIdx] + 1; // store any element that is greater than the maximum el
    for( int i = 0; i < arr.length; i++) {
      // at even indices we will store maximum elements
      if (i % 2 == 0){
        arr[i] += (arr[maxIdx] % maxElem ) * maxElem;
        maxIdx -= 1;
      }
      else { // at odd indices we will store minimum elements
        arr[i] += (arr[minIdx] % maxElem ) * maxElem;
        minIdx += 1;
      }
    }
    // dividing with maxElem to get original values.
    for( int i = 0; i < arr.length; i++) {
      arr[i] = arr[i] / maxElem;
    }
  }

  public static void main(String args[]) {

    int[] arr = {1,2,3,4,5,6,7,8,9};
    System.out.print("Array before min/max: ");
    for (int i = 0; i < arr.length; i++)
      System.out.print(arr[i] + " ");
    System.out.println();

    maxMin(arr);

    System.out.print("Array after min/max: ");
    for (int i = 0; i < arr.length; i++)
      System.out.print(arr[i] + " ");
    System.out.println();
  }
}
```

Explanation

This solution is very smart. We actually store two elements at one index mathematically. The original element is stored as the remainder, while the max/min element is stored as the multiplier. The following line achieves this;

```
arr[i] += (arr[maxIdx] % maxElem ) * maxElem;
```

Here, `arr[maxId]` is stored as the *multiplier*. Whereas, `arr[i]` is stored as the *remainder*. For example in the array, `[1, 2, 3, 4, 5, 6, 7, 8, 9]`, the `maxElem` which is any element greater than the maximum element in the array, in this case, is `10` and `91` is stored at index `0`. With `91`, we can get the original element, `1`, using the expression `91%10` as well as the new element, `9`, using the expression `91/10`.

This allows us to swap the numbers in place without losing any data or using any extra space. To get the final array, we simply divide each element by `maxElem` as done in the last `for` loop.

Time Complexity

Time Complexity

The time complexity of this solution is in $O(n)$. The space complexity is constant.

After these mind crunching exercises, hopefully, you have become familiar enough with arrays. Let's move towards the `array` interview questions for a sound drill.

← Back

Challenge 10: Rearrange Sorted Array...

Next →

Array Interview Questions

✓ Mark as Completed

Report an Issue

Ask a Question (https://discuss.educative.io/tag/solution-review-re-arrange-sorted-array-in-maxmin-form__arrays__data-structures-for-coding-interviews-in-java)