439. 线段树的构造 II

线段树是一棵二叉树，他的每个节点包含了两个额外的属性`start`和`end`用于表示该节点所代表的区间。start和end都是整数，并按照如下的方式赋值：

- 根节点的 *start* 和 *end* 由 `build` 方法所给出。
- 对于节点 A 的左儿子，有 `start=A.left, end=(A.left + A.right) / 2`。
- 对于节点 A 的右儿子，有 `start=(A.left + A.right) / 2 + 1, end=A.right`。
- 如果 *start* 等于 *end*, 那么该节点是叶子节点，不再有左右儿子。

对于给定数组设计一个`build`方法，构造出线段树.样例:给出`[3,2,1,4]`，线段树将被这样构造

```
                [0,  3] (max = 4)
                 /           \
        [0,  1] (max = 3)     [2, 3]  (max = 4)
        /        \              /            \
[0, 0](max = 3)  [1, 1](max = 2)[2, 2](max = 1) [3, 3] (max = 4)
```

```
/* public class SegmentTreeNode {
 *     public int start, end, max;
 *     public SegmentTreeNode left, right;
 *     public SegmentTreeNode(int start, int end, int max) {
 *         this.start = start;
 *         this.end = end;
 *         this.max = max
 *         this.left = this.right = null;
 *     }
 * }*/
public class Solution {
    public SegmentTreeNode build(int[] A) {
        return buildHelper(0, A.length - 1 , A);
    }
    public SegmentTreeNode buildHelper( int start, int end, int[] A ){
        if( start > end )  return null;    //1. 终止条件1.1
        if( start == end )                 //1. 终止条件1.2
            return new SegmentTreeNode( start, end, A[start] );
        // 2.递归构造线段树
        int mid = ( start + end )/2;  //[ start, mid] [mid+1, end]
        SegmentTreeNode node = new SegmentTreeNode(start, end, A[start]);
        node.left = buildHelper( start, mid, A );
        node.right = buildHelper( mid + 1, end, A );
        // 3.从左右节点中选择最大的赋值
        if( null != node.left )
            node.max = Math.max( node.max, node.left.max );
        if( null != node.right )
            node.max = Math.max( node.max, node.right.max );
        return node;
    }
}
```