归并排序（Merge Sort）

- 1945年由约翰·冯·诺伊曼（John von Neumann）首次提出

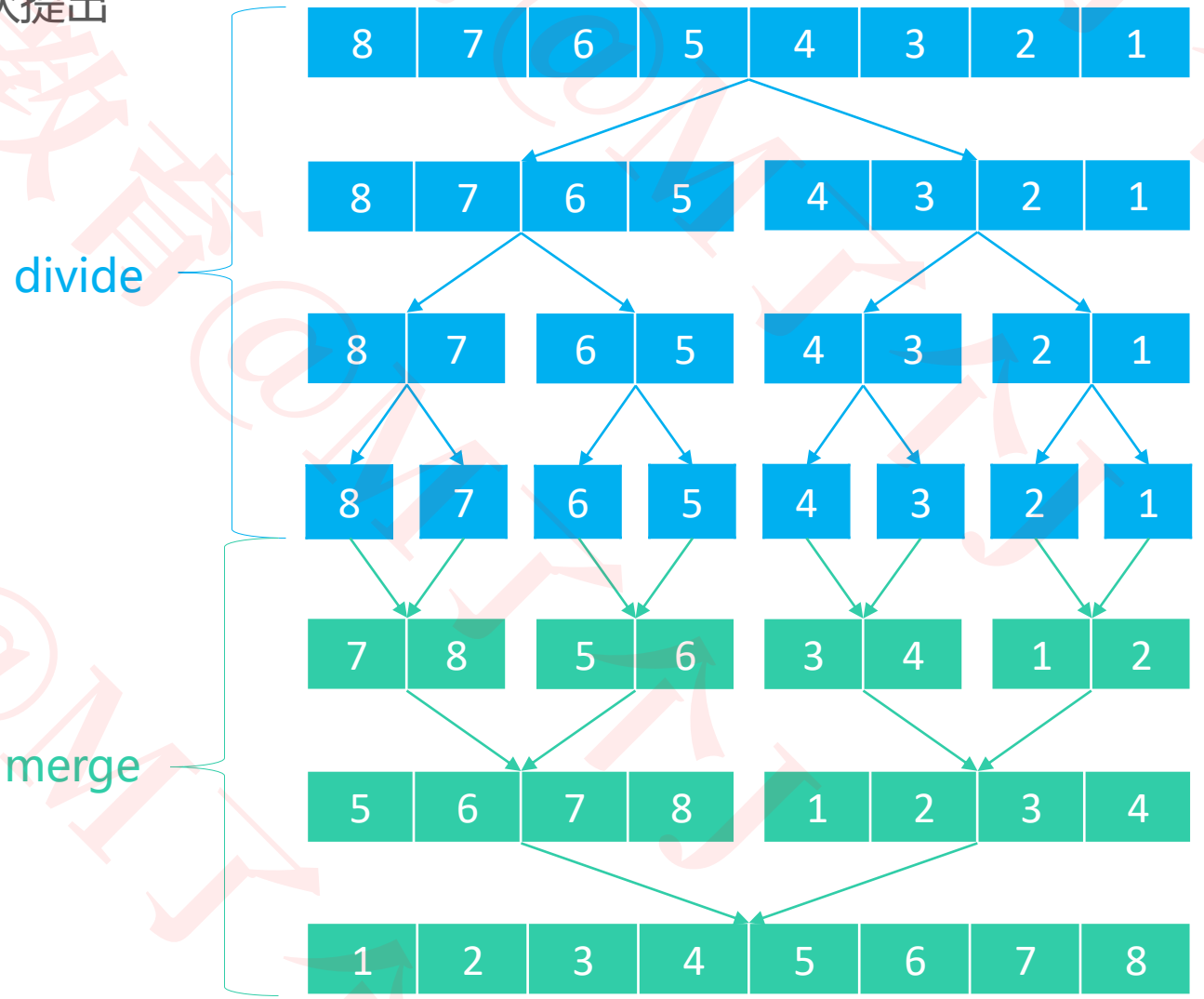
- 执行流程

① 不断地将当前序列平均分割成2个子序列

✓ 直到不能再分割（序列中只剩1个元素）

② 不断地将2个子序列合并成一个有序序列

✓ 直到最终只剩下1个有序序列

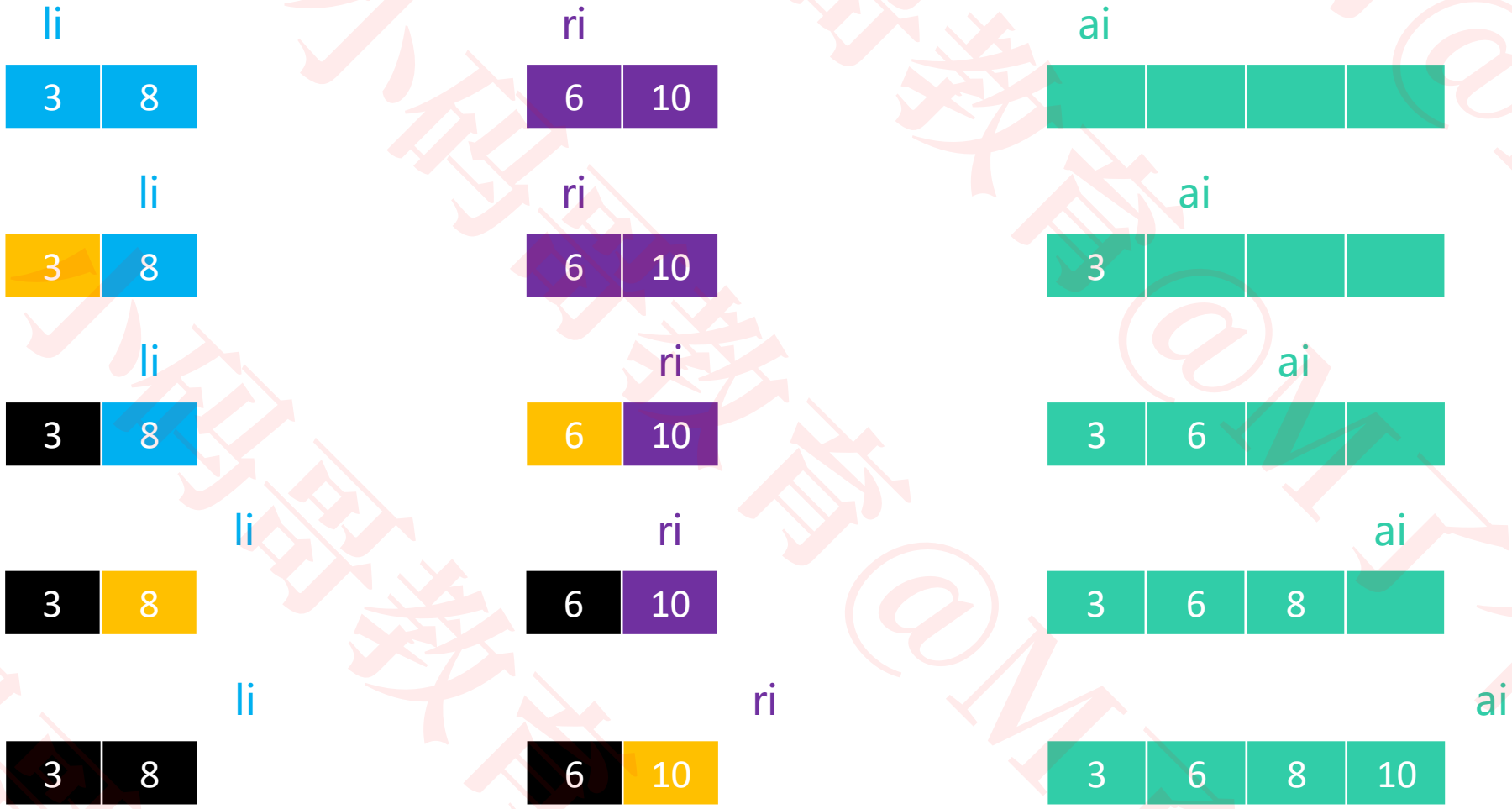


归并排序 – divide实现

```
// 准备一段临时的数组空间，在merge操作中使用
leftArray = (T[]) new Object[array.length >> 1];
sort(0, array.length);
```

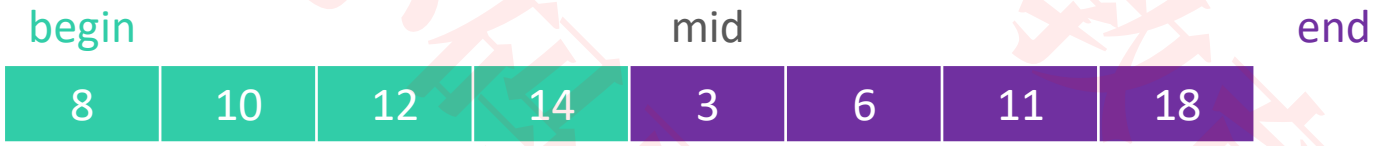
```
/**
 * [begin, end)
 */
private void sort(int begin, int end) {
    // 至少要有2个元素
    if (end - begin < 2) return;

    int mid = (begin + end) >> 1;
    sort(begin, mid);
    sort(mid, end);
    merge(begin, mid, end);
}
```

归并排序 – merge

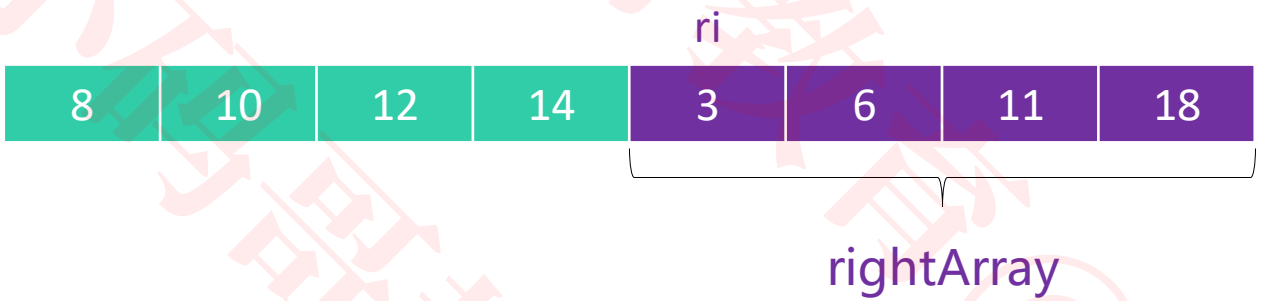

li
ri
ai
3
8
6
10
li
ri
ai
3
8
6
10
3
li
ri
ai
3
8
6
10
3
6
li
ri
ai
3
8
6
10
3
6
8
li
ri
ai
3
8
6
10
3
6
8
10

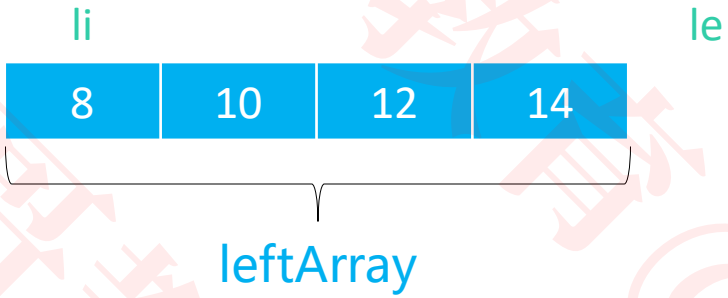
归并排序 – merge细节

- 需要 merge 的 2 组序列存在于同一个数组中，并且是挨在一起的



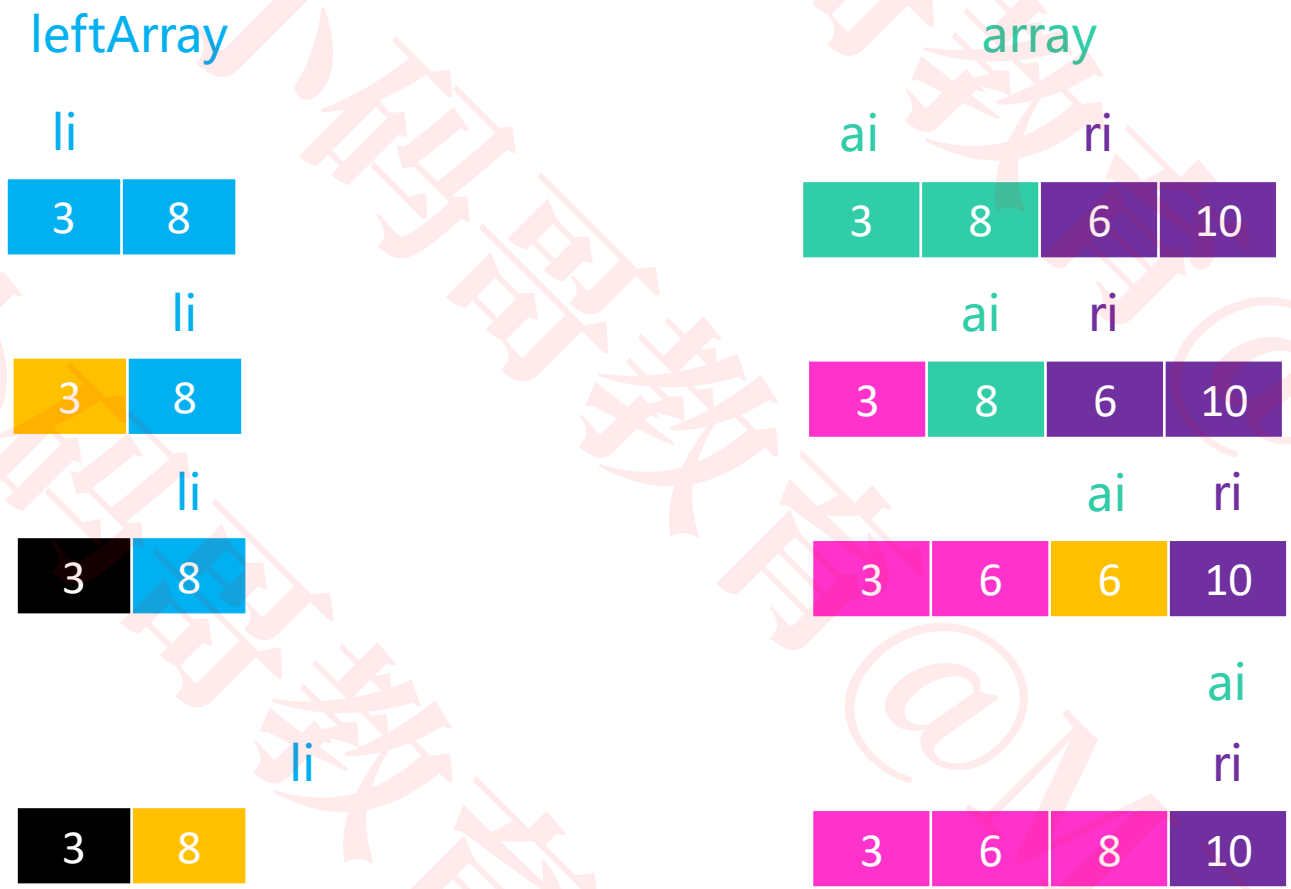
- 为了更好地完成 merge 操作，最好将其中 1 组序列备份出来，比如 [begin, mid)



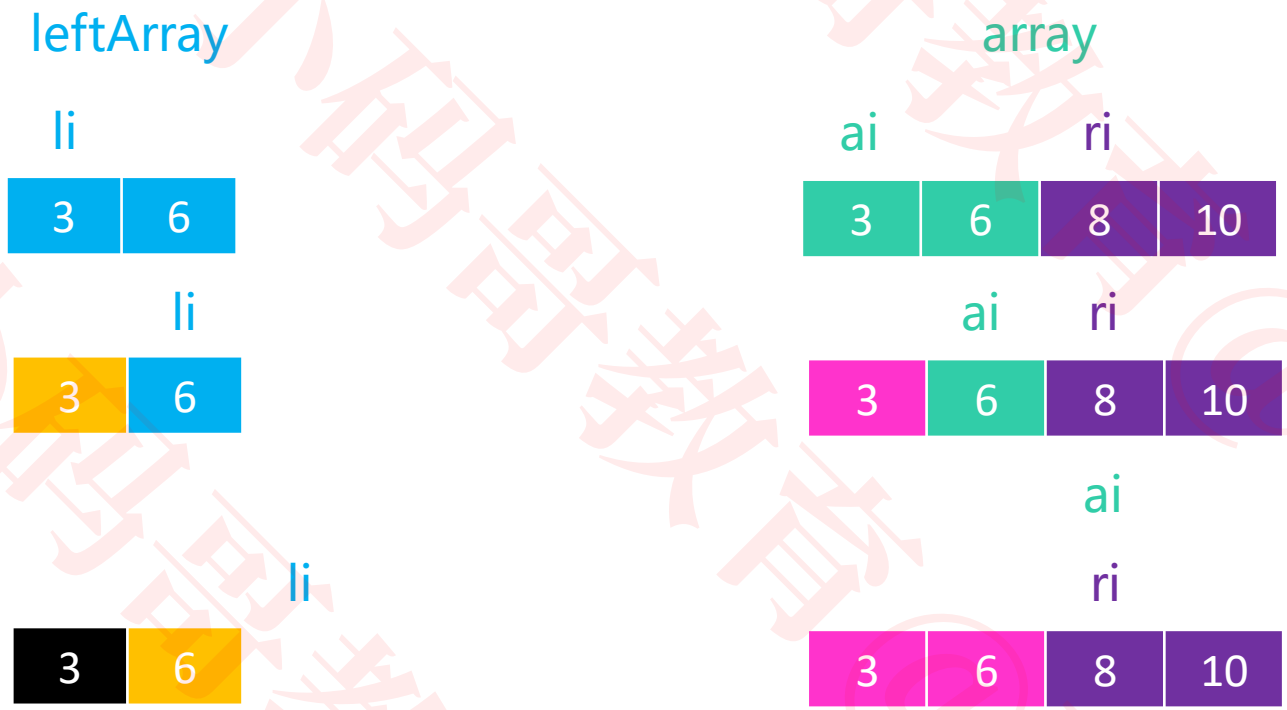


- li == 0，le == mid – begin
- ri == mid，re == end

归并排序 – merge


leftArray
li
3
8
li
3
8
li
3
8
li
3
8
array
ai
ri
3
8
6
10
ai
ri
3
8
6
10
ai
ri
3
6
6
10
ai
ri
3
6
8
10

归并排序 – merge – 左边先结束



归并排序 – merge – 右边先结束

小码哥教育 SEEMYGO

归并排序 – merge实现

```
/* [begin, mid), [mid, end) */
private void merge(int begin, int mid, int end) {
    int li = 0, le = mid - begin; // 左边数组(基于leftArray)
    int ri = mid, re = end; // 右边数组(基于array)
    int ai = begin; // array的索引
    for (int i = li; i < le; i++) { // 拷贝左边数组到leftArray
        leftArray[i] = array[begin + i];
    }
    while (li < le) {
        if (ri < re && cmp(array[ri], leftArray[li]) < 0) {
            array[ai++] = array[ri++]; // 拷贝右边数组到array
        } else {
            array[ai++] = leftArray[li++]; // 拷贝左边数组到array
        }
    } // cmp位置改为 <= 会失去稳定性
}
```

归并排序 – 复杂度分析

- 归并排序花费的时间
  - $T(n) = 2 * T(n/2) + O(n)$
  - $T(1) = O(1)$
  - $T(n)/n = T(n/2)/(n/2) + O(1)$

- 令 $S(n) = T(n)/n$
  - $S(1) = O(1)$
  - $S(n) = S(n/2) + O(1) = S(n/4) + O(2) = S(n/8) + O(3) = S(n/2^k) + O(k) = S(1) + O(\log n) = O(\log n)$
  - $T(n) = n * S(n) = O(n\log n)$

- 由于归并排序总是平均分割子序列，所以最好、最坏、平均时间复杂度都是 $O(n\log n)$，属于稳定排序

- 从代码中不难看出：归并排序的空间复杂度是 $O(n/2 + \log n) = O(n)$
  - $n/2$ 用于临时存放左侧数组，$\log n$ 是因为递归调用