起始当作A的话，A会被拆成左右两部分（左树、右树）；然后再接着拆左树，变成左左树和左右树，当然，只有左左树符合了terminate的条件，才会进行左右树，若左右树也已经符合了terminate条件，才会进入到merge方法，将左左树和右右树合并，这时A的左树已经有序了；当merge方法执行完之后，去拆A的右树，一样的道理，假设此右树只能再拆一层，拆出了右左树之后（代码18行），符合了terminate条件（代码15行），那就去执行下一步代码拆右左树（代码19行），也符合了terminate条件之后（代码15行），执行merge方法，A的右树也合并成功了；最后执行merge方法将A的左右子树合并。

文本

描述已自动生成

*mergeSort*(array, left, mid);

*mergeSort*(array, left, mid);

*mergeSort*(array, left, mid);

*mergeSort*(array, mid + 1, right);

*merge*(array, left, mid, right);

*mergeSort*(array, mid + 1, right);

*mergeSort*(array, left, mid);

*mergeSort*(array, mid + 1, right);

*merge*(array, left, mid, right);

*merge*(array, left, mid, right);

*mergeSort*(array, mid+1, right);

*mergeSort*(array, left, mid);

*mergeSort*(array, left, mid);

*mergeSort*(array, mid + 1, right);

*merge*(array, left, mid, right);

*mergeSort*(array, mid+1, right);

*mergeSort*(array, left, mid);

*mergeSort*(array, mid + 1, right);

*merge*(array, left, mid, right);

*merge*(array, left, mid, right);

*merge*(array, left, mid, right);