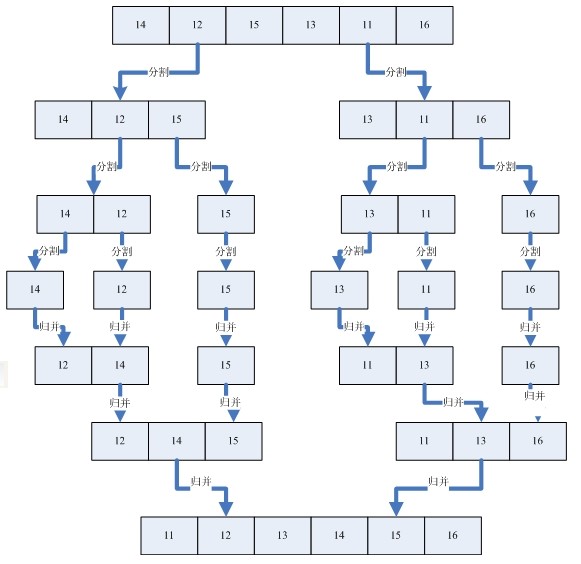
归并排序:

想：**归并（Merge）排序法**是将两个（或两个以上）有序表合并成一个新的有序表，即把待排序序列分为若干个子序列，每个子序列是有序的。然后再把有序子序列合并为整体有序序列

**1.两路归并排序算法思路**

①把 n 个记录看成 n 个长度为1的有序子表

②进行两两归并使记录关键字有序，得到 n/2 个长度为 2 的有序子表



③重复第②步直到所有记录归并成一个长度为 n 的有序表为止

直白的所就是分为两个步骤

现将数组中的元素拆分成一个一个的子集,使用合并将子集变成有序

拆分是很简单的操作,每次都按照原有的数组元素进行一半拆分 开始元素的位置+最后一个元素的位置/2

如何将将二个有序数列合并。只要从比较二个数列的第一个数，谁小就先取谁，取了后就在对应数列中删除这个数。然后再进行比较，如果有数列为空，那直接将另一个数列的数据依次取出即可。

|  |
| --- |
| import java.util.Arrays;  public class MergeSort {  public static void main(String[] args) {  int[] array = new int[10];  for(int i = 0;i<array.length;i++){  array[i] = (int)(Math.random()\*100);  }  System.out.println(Arrays.toString(array));  mergeSort(array, 0, array.length - 1);  System.out.println(Arrays.toString(array));  }    public static void mergeSort(int[] array, int left, int right) {  if (left < right) {  int center = (left + right) / 2;  // 将数组拆分为两份，并递归拆分子数组，直到数组中只有一个元素  mergeSort(array, left, center);  mergeSort(array, center + 1, right);  // 合并相邻数组  merge(array, left, center, right);    }  }    // 合并子数组的函数  public static void merge(int[] array, int left, int center, int right) {  // 临时数组，用于排序  int[] tempArray = new int[array.length];  // 用于将排好序的临时数组复制回原数组  int mark = left;  // 第二个数组的左端  int mid = center + 1;  // 用于临时数组的下标  int tempLeft = left;  while (left <= center && mid <= right) {  // 从两个子数组中取出最小的放入临时数组，即按从小到大的顺序重新排布  if (array[left] <= array[mid]) {  tempArray[tempLeft++] = array[left++];  } else {  tempArray[tempLeft++] = array[mid++];  }  }  // 剩余部分依次放入临时数组  while (left <= center) {  tempArray[tempLeft++] = array[left++];  }  while (mid <= right) {  tempArray[tempLeft++] = array[mid++];  }  // 将中间数组中的内容复制回原数组  while (mark <= right) {  array[mark] = tempArray[mark++];  }  }  } |