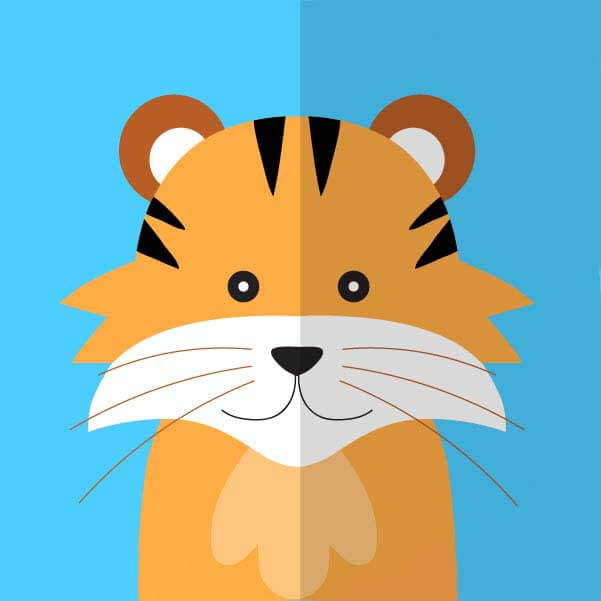
**Mapreduce的经典排序（快排 & 归并）**

[](https://www.jianshu.com/u/50a35b488a7e)

[super\_wing](https://www.jianshu.com/u/50a35b488a7e) 关注

2017.12.12 13:44\* 字数 301 阅读 406评论 0喜欢 1

**1，快速排序**

基本思想：选择一个基准元素,通常选择第一个元素或者最后一个元素,通过一趟扫描，将待排序列分成两部分,一部分比基准元素小,一部分大于等于基准元素,此时基准元素在其排好序后的正确位置,然后再用同样的方法递归地排序划分的两部分。

image.png

public class QuickSort {

public static void main(String[] args) {

int[] a={49,38,65,97,76,13,27,49,78,34,12,64,1,8};

System.out.println("排序之前：");

for (int i = 0; i < a.length; i++) {

System.out.print(a[i]+" ");

}

//快速排序

quick(a);

System.out.println();

System.out.println("排序之后：");

for (int i = 0; i < a.length; i++) {

System.out.print(a[i]+" ");

}

}

private static void quick(int[] a) {

if(a.length>0){

quickSort(a,0,a.length-1);

}

}

private static void quickSort(int[] a, int low, int high) {

if(low<high){ //如果不加这个判断递归会无法退出导致堆栈溢出异常

int middle = getMiddle(a,low,high);

quickSort(a, 0, middle-1);

quickSort(a, middle+1, high);

}

}

private static int getMiddle(int[] a, int low, int high) {

int temp = a[low];//基准元素

while(low<high){

//找到比基准元素小的元素位置

while(low<high && a[high]>=temp){

high--;

}

a[low] = a[high];

while(low<high && a[low]<=temp){

low++;

}

a[high] = a[low];

}

a[low] = temp;

return low;

}

}

**2，归并排序**

基本思想：对于给定的一组集合，利用递归与分治技术将数据序列划分成为越来越小的子集合，再对子集合排序，最后再用递归方法将排好序的子集合合并成为越来越大的有序序列。  
经过第一轮比较后得到最小的记录，然后将该记录的位置与第一个记录的位置交换；接着对不包括第一个记录以外的其他记录进行第二次比较，得到最小记录并与第二个位置记录交换；重复该过程，知道进行比较的记录只剩下一个为止。

20160427172905073.png

public class MergeSort {

public static void main(String[] args) {

int[] data = {50,10,90,30,70,40,80,60,20};

print(data,true);

mergeSort(data, 0, data.length -1 );

print(data,false);

}

private static void mergeSort(int[] data, int left, int right) {

if (left < right) {

int middle = (left + right) / 2;

mergeSort(data, left, middle);

mergeSort(data, middle + 1, right);

merge(data, left, middle, right);

}

}

private static void merge(int[] data, int left, int middle, int right) {

int[] tmpArray = new int[data.length];

int mid = middle + 1;

int third = left;

int tmp = left;

while(left <= middle && mid <= right) {

if(data[left] <= data[mid]) {

tmpArray[third++] = data[left++];

}else {

tmpArray[third++] = data[mid++];

}

}

// 剩余部分依次放入临时数组（实际上两个while只会执行其中一个）

while (mid <= right) {

tmpArray[third++] = data[mid++];

}

while (left <= middle) {

tmpArray[third++] = data[left++];

}

// 将临时数组中的内容拷贝回原数组中

// （原left-right范围的内容被复制回原数组）

while (tmp <= right) {

data[tmp] = tmpArray[tmp++];

}

}

private static void print(int[] data, boolean before) {

if(before) {

System.out.println("排序之前：");

}else {

System.out.println("排序之后：");

}

for (int i = 0; i < data.length; i++) {

System.out.print(data[i] + ", ");

}

System.out.println();

}