**Java排序算法(三)--归并排序（MergeSort）递归与非递归的实现**

原创 2016年03月21日 09:39:22

* 标签：
* [java](http://so.csdn.net/so/search/s.do?q=java&t=blog) /
* [归并排序](http://so.csdn.net/so/search/s.do?q=%E5%BD%92%E5%B9%B6%E6%8E%92%E5%BA%8F&t=blog) /
* [排序算法](http://so.csdn.net/so/search/s.do?q=%E6%8E%92%E5%BA%8F%E7%AE%97%E6%B3%95&t=blog)
* 3096

归并有递归和非递归两种。

归并的思想是：   
1.将原数组首先进行两个元素为一组的排序，然后合并为四个一组，八个一组，直至合并整个数组；   
2.合并两个子数组的时候，需要借助一个临时数组，用来存放当前的归并后的两个数组；   
3.将临时数组复制回原数组对应的位置。

非递归的代码如下：

package mergesort;

import java.util.Arrays;

import java.util.Random;

import java.util.Scanner;

//归并排序的非递归算法

public class MergeSort{

public static void main(String args[]){

MergeSort mer = new MergeSort();

int[] array = mer.getArray();

System.out.println("OriginalArray:" + Arrays.toString(array));

mer.mergeSort(array);

System.out.println("SortedArray:" + Arrays.toString(array));

}

public int[] getArray(){

Scanner cin = new Scanner(System.in);

System.out.print("Input the length of Array:");

int length = cin.nextInt();

int[] arr = new int[length];

Random r = new Random();

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

arr[i] = r.nextInt(100);

}

cin.close();

return arr;

}

public void mergeSort(int[] a){

int len = 1;

while(len < a.length){

for(int i = 0; i < a.length; i += 2\*len){

merge(a, i, len);

}

len \*= 2;

}

}

public void merge(int[] a, int i, int len){

int start = i;

int len\_i = i + len;//归并的前半部分数组

int j = i + len;

int len\_j = j +len;//归并的后半部分数组

int[] temp = new int[2\*len];

int count = 0;

while(i < len\_i && j < len\_j && j < a.length){

if(a[i] <= a[j]){

temp[count++] = a[i++];

}

else{

temp[count++] = a[j++];

}

}

while(i < len\_i && i < a.length){//注意：这里i也有可能超过数组长度

temp[count++] = a[i++];

}

while(j < len\_j && j < a.length){

temp[count++] = a[j++];

}

count = 0;

while(start < j && start < a.length){

a[start++] = temp[count++];

}

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47
* 48
* 49
* 50
* 51
* 52
* 53
* 54
* 55
* 56
* 57
* 58
* 59
* 60
* 61
* 62
* 63
* 64

递归算法的实现代码如下：

package mergesort;

public class MergeSort {

public static void mergeSort(int[] data,int left,int right){ //left,right均为数字元素下标

if(left<right){

int half=(left+right)/2;

mergeSort(data,left,half);

mergeSort(data,half+1,right);

merge(data,left,right);

}

}

public static void merge(int []a,int l,int h){

int mid=(l+h)/2;

int i=l;

int j=mid+1;

int count=0;

int temp[]=new int[h-l+1];

while(i<=mid&&j<=h){

if(a[i]<a[j]){

temp[count++]=a[i++];

}else{

temp[count++]=a[j++];

}

}

while(i<=mid){

temp[count++]=a[i++];

}

while(j<=h){

temp[count++]=a[j++];

}

count=0;

while(l<=h){

a[l++]=temp[count++];

}

}

public static void printArray(int arr[]){

for(int k=0;k<arr.length;k++){

System.out.print(arr[k]+"\t");

}

}

public static int[] getArray(){

// int[] data={4,2,3,1};

int[] data={543,23,45,65,76,1,456,7,77,88,3,9};

return data;

}

public static void main(String args[]){

int[]a=getArray();

System.out.print("数组排序前：");

printArray(a);

System.out.print("\n");

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

System.out.print("归并排序后：");

printArray(a);

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47
* 48
* 49
* 50
* 51
* 52
* 53
* 54
* 55
* 56

归并排序的时间复杂度为O(n\*log2n),空间复杂度为O(n)

归并排序是一种稳定的排序方法。