Solution 1:

Array 写法

Public class MergeSort {

public int[] mergeSort(int[] array) {

if(array == null)

return array;

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

return array;

}

public int[] mergeSort(int[] array, int left, int right) {

if (left >= right) {

return;

}

int Mid = left + (right - left)/2;

mergeSort(array, left, Mid);

mergeSort(array, Mid + 1; right);

merge(array, left, Mid, right);

}

private void merge(int[] array, int left, int mid, int right) {

int[] helper = new int[array.length];

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

helper[i] = array[i]

}

int leftIndex = left;

int rightIndex = mid + 1;

while( leftIndex <= mid && rightIndex <= right) {

if (helper[leftIndex] <= helper[rightIndex]) {

array[left++] = helper[leftIndex++];

} else {

array[left++] = helper[rightIndex++];

}

}

}

while (leftIndex <= mid) {

array[left++] = helper[leftIndex++];

}

}

}

Solution 2:

ArrayList 写法

Public class MergeSort {

public ArrayList<Integer> mergeSort(ArrayList<Integer> array) {

if( array == null) {

return array;

}

return mergeSort(array, 0, array.size() - 1);

}

public ArrayList<Integer> mergeSort(ArrayList<Integer> array, int left, int right) {

ArrayList<Integer> result = new ArrayList<>();

if (left > right) {

return result;

}

if (left == right) {

result.add(array.get(left));

return result;

}

int mid = left + (left + right) / 2;

List<Integer> reLeft = mergeSort(array, left, mid);

List<Integer> reRight = mergeSort(array, mid + 1, right);

merge(reLeft, reRight, result);

return result;

}

private void merge(ArrayList<Integer>left, ArrayList<Integer> right, ArrayList<Integer> result) {

int l = 0;

int r = 0;

while(l < left.size() && r < right.size(){

result.add(left.get(i++);

} else {

result.add(right.get(r++);

}

}

while (l < left.size()){

result.add(left.get(i++));

}

while (r < right.size()) {

result.add(right.get(r++));

}

}

Time Complexity: O(nlogn)

Space Complexity: O(n)

}

Compare to sample solution, we're using the same array to bound memory consumption:

不用考虑gc，只有O(n)的空间复杂的，只是增加了一次helper array

Public class MergeSort {

public int[] mergeSort(int[] array) {

if(array == null)

return array;

int[] helper = new int[array.length];

mergeSort(array, helper,0, array.length - 1);

return array;

}

public int[] mergeSort(int[] array, int[] helper, int left, int right) {

if (left >= right) {

return;

}

int Mid = left + (right - left)/2;

mergeSort(array, helper, left, Mid);

mergeSort(array, helper, Mid + 1; right);

merge(array, helper, left, Mid, right);

}

private void merge(int[] array, int[] helper, int left, int mid, int right) {

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

helper[i] = array[i]

}

int leftIndex = left;

int rightIndex = mid + 1;

while( leftIndex <= mid && rightIndex <= right) {

if (helper[leftIndex] <= helper[rightIndex]) {

array[left++] = helper[leftIndex++];

} else {

array[left++] = helper[rightIndex++];

}

}

}

while (leftIndex <= mid) {

array[left++] = helper[leftIndex++];

}

}

}