1. **归并排序**
2. **快速排序**
3. **冒泡排序**
4. **排序时间运算psvm**
5. 归并排序

import java.util.Arrays;  
  
public class MergeSortDemo *{* public static void main*(*String*[]* args*) {* int arr[] = { 8, 4, 5, 7, 1, 3, 6, 2 };  
 int temp[] = new int[arr.length]; *// 归并排序需要一个额外空间* mergeSort(arr, 0, arr.length - 1, temp);  
 System.out.println("归并排序后=" + Arrays.toString(arr));  
  
 }  
  
 *// 分+合方法* public static void mergeSort(int[] arr, int left, int right, int[] temp) {  
 if (left < right) {  
 int mid = (left + right) / 2; *// 中间索引  
 // 向左递归进行分解* mergeSort(arr, left, mid, temp);  
 *// 向右递归进行分解* mergeSort(arr, mid + 1, right, temp);  
 *// 合并* merge(arr, left, mid, right, temp);  
 }  
 }  
  
 *// 合并的方法  
 /\*\*  
 \*  
 \* @param arr 排序的原始数组  
 \* @param left 左边有序序列的初始索引  
 \* @param mid 中间索引  
 \* @param right 右边索引  
 \* @param temp 做中转的数组  
 \*/* public static void merge(int[] arr, int left, int mid, int right, int[] temp) {  
  
 int i = left; *// 初始化i, 左边有序序列的初始索引* int j = mid + 1; *// 初始化j, 右边有序序列的初始索引* int t = 0; *// 指向temp数组的当前索引  
  
 // (一)  
 // 先把左右两边(有序)的数据按照规则填充到temp数组  
 // 直到左右两边的有序序列，有一边处理完毕为止* while (i <= mid && j <= right) {*// 继续  
 // 如果左边的有序序列的当前元素，小于等于右边有序序列的当前元素  
 // 即将左边的当前元素，填充到 temp数组  
 // 然后 t++, i++* if (arr[i] <= arr[j]) {  
 temp[t] = arr[i];  
 t += 1;  
 i += 1;  
 } else { *// 反之,将右边有序序列的当前元素，填充到temp数组* temp[t] = arr[j];  
 t += 1;  
 j += 1;  
 }  
 }  
  
 *// (二)  
 // 把有剩余数据的一边的数据依次全部填充到temp* while (i <= mid) { *// 左边的有序序列还有剩余的元素，就全部填充到temp* temp[t] = arr[i];  
 t += 1;  
 i += 1;  
 }  
  
 while (j <= right) { *// 右边的有序序列还有剩余的元素，就全部填充到temp* temp[t] = arr[j];  
 t += 1;  
 j += 1;  
 }  
  
 *// (三)  
 // 将temp数组的元素拷贝到arr  
 // 注意，并不是每次都拷贝所有* t = 0;  
 int tempLeft = left; *//  
 // 第一次合并 tempLeft = 0 , right = 1 //第二次： tempLeft = 2 right = 3 //第三次： tL=0 ri=3  
 // 最后一次 tempLeft = 0 right = 7* while (tempLeft <= right) {  
 arr[tempLeft] = temp[t];  
 t += 1;  
 tempLeft += 1;  
 }  
  
 }  
  
}

2.快速排序

import java.util.Arrays;  
  
public class QuickSortDemo *{* public static void main*(*String*[]* args*) {* int*[]* arr = *{*25, 84, 21, 47, 15, 27, 68, 35, 20*}*;  
   
 *quickSort(*arr, 0, arr.length - 1*)*;  
  
 System.*out*.println*(*Arrays.*toString(*arr*))*;  
 *}* private static void quickSort*(*int*[]* arr, int left, int right*) {* if *(*left < right*) {* int partitionIndex = *partition(*arr, left, right*)*;  
 *quickSort(*arr, left, partitionIndex - 1*)*;  
 *quickSort(*arr, partitionIndex + 1, right*)*;  
 *}  
 }* private static int partition*(*int*[]* arr, int left, int right*) {* int pivot = arr*[*left*]*;  
 *//终止while循环以后left和right一定相等的* while *(*left < right*) {* while *(*left < right && arr*[*right*]* >= pivot*) {* --right;  
 *}* arr*[*left*]* = arr*[*right*]*;  
 while *(*left < right && arr*[*left*]* <= pivot*) {* ++left;  
 *}* arr*[*right*]* = arr*[*left*]*;  
 *}* arr*[*left*]* = pivot;  
 *//right可以改为left* return left;  
 *}  
}*

3.冒泡排序  
import java.util.Arrays;  
  
public class BubbleSort *{* public static void main*(*String*[]* args*) {* int arr*[]* = *{*3, 9, -1, 10, -2*}*;  
  
 int temp = 0; *// 临时变量* for *(*int i = 0; i < arr.length -1; i++*){* for *(*int j = 0; j< arr.length -1 -i;j++*){* if*(*arr*[*j*]* > arr*[*j + 1*]){* temp = arr*[*j*]*;  
 arr*[*j*]* = arr*[*j+1*]*;  
 arr*[*j + 1*]* = temp;  
 *}  
 }  
 }* System.*out*.println*(*"冒泡排序后的数组"*)*;  
 System.*out*.println*(*Arrays.*toString(*arr*))*;  
  
 *}  
}*

4.排序时间运算

public static void main*(*String*[]* args*) {  
  
 // 创建要给80000个的随机的数组* int*[]* arr = new int*[*80000*]*;  
 for *(*int i = 0; i < 80000; i++*) {* arr*[*i*]* = *(*int*) (*Math.*random()* \* 8000000*)*; *// 生成一个[0, 8000000) 数  
 }* Date date1 = new Date*()*;  
 SimpleDateFormat simpleDateFormat = new SimpleDateFormat*(*"yyyy-MM-dd HH:mm:ss"*)*;  
 String date1Str = simpleDateFormat.format*(*date1*)*;  
 System.*out*.println*(*"排序前的时间是=" + date1Str*)*;  
  
 *quickSort(*arr, 0, arr.length - 1*)*; *// 交换式* Date data2 = new Date*()*;  
 String date2Str = simpleDateFormat.format*(*data2*)*;  
 System.*out*.println*(*"排序后的时间是=" + date2Str*)*;  
  
*}*