# 时间复杂度和空间复杂度



稳定的排序：

直接插入，冒泡排序，归并排序，基数排序

# 冒泡排序稍微优化

|  |
| --- |
| public static void bubbleSort1(int[] a, int n) {  int i,j;  for (i=n-1; i>0; i--) {  // 将a[0...i]中最大的数据放在末尾  for (j=0; j<i; j++) {  if (a[j] > a[j+1]) {  // 交换a[j]和a[j+1]  int tmp = a[j];  a[j] = a[j+1];  a[j+1] = tmp;  }  }  }  } |

这种方式，如果第一趟就已经排好序了，后面的那么多遍历比较都是没有任何意义的。

所以优化一下，定义一个flag，如果发生了交换，就继续遍历再跑一趟，如果没有交换，说明已经有序，则退出外层for循环。

|  |
| --- |
| public static void bubbleSort2(int[] a, int n) {  int i,j;  int flag; // 标记  for (i=n-1; i>0; i--) {  flag = 0; // 初始化标记为0  // 将a[0...i]中最大的数据放在末尾  for (j=0; j<i; j++) {  if (a[j] > a[j+1]) {  // 交换a[j]和a[j+1]  int tmp = a[j];  a[j] = a[j+1];  a[j+1] = tmp;  flag = 1; // 若发生交换，则设标记为1  }  }  if (flag==0)  break; // 若没发生交换，则说明数列已有序。  }  } |

# 快速排序

|  |
| --- |
| public class QuickSort {  public static void main(String[] args) {  int[] src = {7,9,5,2,1,12,53,45,86};  Qsort(src,0,src.length-1);  for (int i : src) {  System.out.println(i);  }  }  static void Qsort(int src[], int low, int high){  if(low>=high){  return;  }  int temp = src[low];  while (low<high) {  while (low<high&&src[high]>=temp) {  high--;  }  src[low] = src[high];  while (low<high&&src[low]<=temp) {  low++;  }  src[high] = src[low];    }  src[low] = temp;  Qsort(src,0,low-1);  Qsort(src,low+1,high);  }  } |

快速排序在序列乱的情况下，效率较高，如果数据有序时，会退化成冒泡排序。