JAVA基础应用:如何实现希尔排序算法 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/182/2021_2022_JAVA_E5_9F_BA_E7_A1_80_c104_182502.htm package Utils.Sort. /** *希尔排序，要求待排序的数组必须实现Comparable接口 */ public class ShellSort implements SortStrategy { private int[] increment. /** *利用希尔排序算法对数组obj进行排序 */ public void sort(Comparable[] obj) { if (obj == null) { throw new NullPointerException("The argument can not be null!"). } //初始化步长 initGap(obj). //步长依次变化（递减） for (int i = increment.length - 1 .i >= 0 .i--) { int step = increment[i]. //由步长位置开始 for (int j = step .j { Comparable tmp. //如果后面的小于前面的（相隔step），则与前面的交换 for (int m = j .m >= step .m = m - step) { if (obj[m].compareTo(obj[m - step]) { tmp = obj[m - step]. obj[m - step] = obj[m]. obj[m] = tmp. } //因为之前的位置必定已经比较过，所以这里直接退出循环 else { break. } } } } } /** *根据数组的长度确定求增量的公式的最大指数,公式为pow(4, i) - 3 * pow(2, i) 1和9 * pow(4, i) - 9 * pow 2, i) 1 *@return int[] 两个公式的最大指数 *@param length 数组的长度 */ private int[] initExponent(int length) { int[] exp = new int[2]. exp[0] = 1. exp[1] = -1. int[] gap = new int[2]. gap[0] = gap[1] = 0. //确定两个公式的最大指数 while (gap[0] { exp[0] . gap[0] = (int)(Math.pow(4, exp[0]) - 3 * Math.pow(2, exp[0]) 1). } exp[0]--. while (gap[1] { exp[1] . gap[1] = (int)(9 * Math.pow(4, exp[1]) - 9 * Math.pow(2, exp[1]) 1). } exp[1]--. return exp. } private void initGap(Comparable[] obj) { //利用公式初始化增量序列 int

exp[] = initExponent(obj.length). int[] gap = new int[2]. increment = new int[exp[0] exp[1]]. //将增量数组由大到小赋值 for (int i = exp[0] exp[1] - 1 .i >= 0 .i--) { gap[0] = (int)(Math.pow(4, exp[0]) - 3 * Math.pow(2, exp[0]) 1). gap[1] = (int)(9 * Math.pow(4, exp[1]) - 9 * Math.pow(2, exp[1]) 1). //将大的增量先放入增量数组，这里实际上是一个归并排序 //不需要考虑gap[0] == gap[1]的情况，因为不可能出现相等。 if (gap[0] > gap[1]) { increment[i] = gap[0]. exp[0]--. } else { increment[i] = gap[1]. exp[1]--. } } } } 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com