目录

[冒泡排序 1](#_Toc501067929)

[选择排序 1](#_Toc501067930)

[二分法查找 2](#_Toc501067931)

[计算输入的整数在用16进制表示时含有多少个1 2](#_Toc501067932)

[集合合并 3](#_Toc501067933)

[遍历文件夹下所有"."开头的文件 3](#_Toc501067934)

[使用&&跳出多重嵌套循环 4](#_Toc501067935)

|  |  |
| --- | --- |
|  |  |

# 冒泡排序

|  |
| --- |
| public static void bubbleSort(int[] arr) {  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]) {  int temp = arr[j];  arr[j] = arr[j + 1];  arr[j + 1] = temp;  }}}  System.out.println(Arrays.toString(arr));} |

# 选择排序

|  |
| --- |
| 1. 从待排序序列中，找到关键字最小的元素； 2. 如果最小元素不是待排序序列的第一个元素，将其和第一个元素互换； 3. 从余下的 N - 1 个元素中，找出关键字最小的元素，重复1)、2)步，直到排序结束。 |
| public static void selectionSort(int[] arr) {  for (int i = 0; i < arr.length - 1; i++) {  int k = i;  for (int j = i + 1; j < arr.length; j++) {  if (arr[k] > arr[j]) {  k = j;  }  if (i != k) {  int temp = arr[i];  arr[i] = arr[j];  arr[j] = temp;  }}}  System.out.println(Arrays.toString(arr));} |

# 二分法查找

|  |
| --- |
| 当数据量很大适宜采用该方法。采用二分法查找时，数据需是有序不重复的。 |
| static int idx = 1;  public static void find(int leftIndex, int rightIndex, int val, int[] arr) {  int midIndex = (rightIndex + leftIndex) / 2;  int midVal = arr[midIndex]; // 找到中间的数  if (rightIndex >= leftIndex) {  if (midVal > val) {  idx++;  find(leftIndex, midIndex - 1, val, arr); // 在arr左边数中找  } else if (midVal < val) {  idx++;  find(midIndex + 1, rightIndex, val, arr); // 在arr的右边去查找  } else if (midVal == val) {  System.out.println("共查找了" + idx + "次" + "，在下标" + midIndex + "找到");  }  } else {  System.out.println("没有该数!");  } |

# 计算输入的整数在用16进制表示时含有多少个1

|  |
| --- |
| public static void main(String[] args) {  int count = 0;  String num = "17";  boolean isInt = isInteger(num);  if (isInt) {  int number = Integer.parseInt(num);  String hexValue = dec2Hex(number);  for (int i = 0; i < hexValue.length(); i++) {  String value = hexValue.charAt(i) + "";  if ("1".equals(value)) {  ++count;  }}  System.out.println(count);  }}  public static boolean isInteger(String input) { // 判断是不是整数  if (input != null && !"".equals(input.trim())) {  return input.toString().matches("-?[\\d]+");  } else {  return false;  }}  public static String dec2Hex(int dec) { // 可以使用Integer.toHexString()直接转  StringBuffer sb = new StringBuffer();  sb.append("Ox");  for (int i = 0; i < 8; i++) {  int tmp = (dec >> (7 - i % 8) \* 4) & 0x0f; ???  if (tmp < 10) {  sb.append(tmp);  } else {  sb.append((char) ('A' + (tmp - 10)));  }  }  return sb.toString();} |

# 集合合并

|  |
| --- |
| 给定一些字符集合，形式如：{a b c}, {b d}, {e f}, {g}, {g h}  要求将其中交集不为空的集合合并，合并完成后的所有集合之间无交集，例如上例应输出：{a b c d h}, {e f}, {g} |
| HashSet<String> hs1 = new HashSet<>();  hs1.add("a"); hs1.add("b"); hs1.add("c");  HashSet<String> hs2 = new HashSet<>();  hs2.add("b"); hs2.add("d");  HashSet<String> hs3 = new HashSet<>();  hs3.add("e"); hs3.add("f");  HashSet<String> hs4 = new HashSet<>();  hs4.add("g");  HashSet<String> hs5 = new HashSet<>();  hs5.add("d"); hs5.add("h");  List<HashSet<String>> list = new ArrayList<>();  list.add(hs1); list.add(hs2); list.add(hs3); list.add(hs4); list.add(hs5);  for (int i = 0; i < list.size() - 1; i++) {  for (int j = i + 1; j < list.size(); j++) {  if (list.get(i) != null) {  int index = -1;  Iterator<String> ite = list.get(j).iterator();  while (ite.hasNext()) {  if (list.get(i).contains(ite.next())) {  list.get(i).addAll(list.get(j)); // 把"后集合"合并到"前集合"  index = j;  }  }  if (index != -1) {  list.get(j).clear(); // 清空后"后集合"  }}}}  for (int i = 0; i < list.size(); i++) {  if (list.get(i).size() != 0) {  System.out.println(list.get(i));  }} |

# 遍历文件夹下所有"."开头的文件

|  |
| --- |
| File dir = new File("."); // 当前目录  File[] files = dir.listFiles(new FileFilter() {  public boolean accept(File pathname) {  return pathname.getName().startsWith(".");  }  });  for (File file : files) {  System.out.println(file.getName());  } |

# 使用&&跳出多重嵌套循环

|  |
| --- |
| int arr[][] = { { 1, 2, 3 }, { 4, 5, 6, 7 }, { 9 } };  boolean found = false;  for (int i = 0; i < arr.length && !found; i++) {  for (int j = 0; j < arr[i].length; j++) {  if (arr[i][j] == 5) {  found = true;  System.out.println("5的位置是：i=" + i + ",j=" + j);  break;  }}} |