作业二 数据类型与算法介绍

1、什么是数据类型？什么是抽象数据类型？

数据类型是程序设计领域最重要的基本概念之一。在程序里描述的、通过计算机去处理的数据，通常都分属不同的类型，例如整数或浮点数等。每个类型包含一集合法的数据对象，并规定了这些对象的合法操作。

Java中的基本数据类型有8种：byte、short、int、long、float、double、char、boolean。除了基本类型（primitive type），剩下的都是引用类型（reference type）。基本数据类型是在内存中储存整数、小数或字符的，引用数据类型是在内存中存储对象的。

抽象数据类型（Abstract Data Type，ADT）是计算机科学中具有类似行为的特定类别的数据结构的数学模型；或者具有类似语义的一种或多种程序设计语言的数据类型。抽象数据类型的基本思想是抽象，或者说是数据抽象。抽象数据类型是间接定义的，通过其上的可执行的操作以及这些操作的效果的数学约束。

抽象数据类型的基本思想是把数据定义为抽象的对象集合，只为它们定义可用的合法操作，并不暴露其内部实现的具体细节，不论是数据的表示细节还是操作的实现细节。这样的类型称为不变数据类型，这种类型的对象成为不变对象。如果提供了第3类操作，对该类型的对象执行这种操作后，虽然对象依旧，但其内部状态已经改变。这样的数据类型称为可变数据类型，其对象称为可变对象。

2、抽象数据类型如何定义？

在Java编程语言中，可以通过创建抽象类或者接口来定义抽象数据类型（ADT）。以下是一个简单的例子：

```java

// 定义一个抽象数据类型，名为MyADT

public abstract class MyADT {

 // 抽象方法，没有具体的实现

 public abstract void method1();

 public abstract int method2();

}

// 具体的类，继承自MyADT，实现了抽象方法

public class ConcreteClass extends MyADT {

 public void method1() {

 // 具体的实现

 }

 public int method2() {

 // 具体的实现

 return 0;

 }

}

```

在这个例子中，`MyADT` 是一个抽象数据类型，它定义了一些抽象方法。`ConcreteClass` 是一个具体的类，它继承自 `MyADT` 并实现了所有的抽象方法。

抽象数据类型是一种理论模型，它定义了数据和操作数据的方法。在Java中，我们通常使用抽象类或接口来实现抽象数据类型。这些抽象方法在抽象类中没有具体的实现，具体的实现在继承抽象类或实现接口的类中完成。这样做的好处是，我们可以改变具体的实现，而不影响到使用这个数据类型的代码。这就是所谓的封装和数据抽象。

3、什么是算法？如何描述算法？

算法是一系列解决问题或完成特定任务的明确指令。它是计算机科学和编程的基础，因为它定义了计算机需要执行的步骤以达到预期的结果。

描述算法的方法有很多种，以下是一些常见的方法：

* 伪代码：这是一种非正式的语言，用来描述算法，它不需要遵循任何编程语言的语法，但其结构应该类似于一种编程语言。伪代码的目的是清晰、简洁地描述算法的主要思想。
* 流程图：这是一种图形化的表示方法，用于描述算法的步骤和控制流。流程图使用各种形状（如矩形、菱形和圆形）来表示不同类型的步骤，并使用箭头来表示控制流的方向。
* 文字描述：这是一种非正式的描述方法，用自然语言描述算法的步骤。这种方法通常用于解释算法的概念和主要思想。

无论使用哪种方法，描述算法时都应确保清晰、准确，并且容易理解。每个步骤都应该是明确的，没有歧义，以便其他人可以正确地实现和使用该算法。同时，算法应该具有确定性、可行性、有穷性、输入和输出。这五个特性是算法的基本特性。

4、算法具有哪些特性？

* 确定性：算法的每一步都具有明确的含义，不会有二义性，且在任何条件下，算法只有一条执行路径。
* 可行性：算法是能够被执行的，也就是说，算法中描述的操作都是可以通过已经实现的基本操作执行有限次来实现的。
* 输入：一个算法有零个或多个输入，以便能够处理不同的数据。
* 输出：一个算法应有一个或多个输出，这是算法计算的结果。
* 有穷性：算法必须总是在执行有限步之后结束，也就是说，算法不能无限制地运行下去。

5、如何评价算法？

* 正确性：算法应能正确解决问题，得出预期的结果。
* 可读性：算法应该清晰易懂，便于理解和实现。
* 健壮性：对于非法输入，算法应能做出适当的反应或处理，而不是产生错误或异常。
* 效率和存储量需求：评价算法的一个重要指标是它的运行时间和所需的存储空间。这通常通过算法的时间复杂度和空间复杂度来衡量。
* 时间复杂度：表示算法执行时间与问题规模之间的关系。常见的时间复杂度有：常数阶O(1)、对数阶O(logN)、线性阶O(n)、线性对数阶O(nlogN)、平方阶O(n²)、立方阶O(n³)、指数阶O(2^n)等。一般来说，时间复杂度越低，算法的效率越高。
* 空间复杂度：表示算法在运行过程中临时占用存储空间大小与问题规模之间的关系。空间复杂度过高的算法可能会因为占用过多的内存而导致运行效率降低。
* 优化性：算法是否具有可以进一步优化的空间或可能性。
* 复杂性：算法的复杂性是指算法的复杂度，包括时间复杂度和空间复杂度。复杂度越低，算法的效率越高。
* 稳定性：稳定的排序算法会保持相等元素的相对顺序。如果一个排序算法是稳定的，当有两个相等的元素，排序后这两个元素的相对位置不会改变。

6、理解并运行验证哥德巴赫猜想源代码。

代码及运行结果截图：

