第一章：基础

1.1 基础编程模型

JAVA语言简介：略

1.2 数据抽象

1.2.1 使用抽象数据类型

1.2.1.1 抽象数据（ADT）类型的API

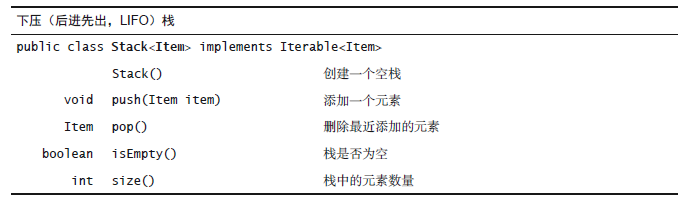
我们使用应用程序编程接口（API）来说明抽象数据类型的行为。他讲列出所有构造函数和实例方法并简要描述他们的功用。

JAVA语言简介：略

1.3背包、队列和栈

1.3.1 API





1.3.1.1 泛型

集合类的抽象数据类型的一个关键特性是我们可以用他们存储任意类型的数据。一种特别的Java机制能够做到这一点，他被成为泛型，也叫作参数化类型。

1.3.1.4 背包

背包是一种不支持从中删除元素的集合数据类型——他的目的就是帮助用户收集元素并迭代遍历所有收集到的元素（用例也可以检查背包是否为空或者获取背包中元素的数量）。

1.3.1.5 先进先出队列

先进先出队列是功能给予FIFO策略的集合类型。

1.3.1.6 下压栈

下压栈是一种给予后进先出LIFO策略的集合类型。

1.3.4 综述

Java内置了数组，链表也很容易实现。而这常常被称为顺序存储和链式存储。

在研究新的应用领域问题的时候，我们将会按照一下步骤识别目标并且使用数据抽象解决问题：

1. 定义API
2. 根据特定的引用场景开发用例代码
3. 描述一种数据结构，并在API所对应的抽象数据类型的实现中根据他定义类的实例变量
4. 分析算法的性能特点

1.4 算法分析

JAVA中有一个表示计时器的抽象数据类型Stopwatch（）

用幂次法则估计的时间模型

1.4.1.6 总结

对于大多数程序，得到其运行时间的数学模型所需的步骤如下：

* 确定输入模型，定义问题的规模；
* 识别内循环；
* 根据内循环中的操作确定成本模型；
* 对于给定的输入，判断这些操作的执行频率。这可能需要进行数学分析。

1.4.5.1 热身运动2-sum

1.5 案例研究：union-find算法

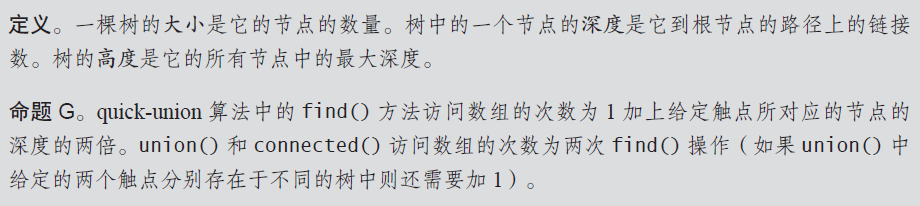
第一种算法：

|  |
| --- |
|  |

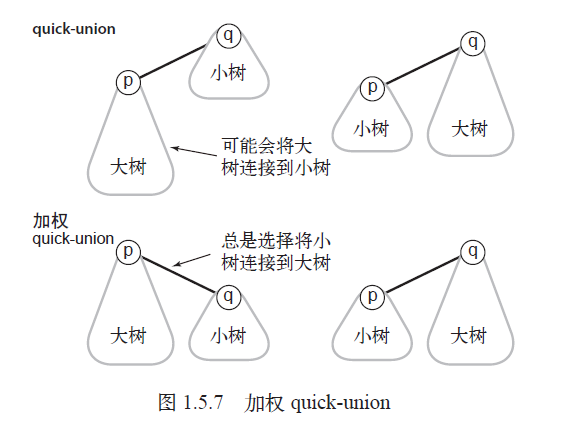
第二种算法：

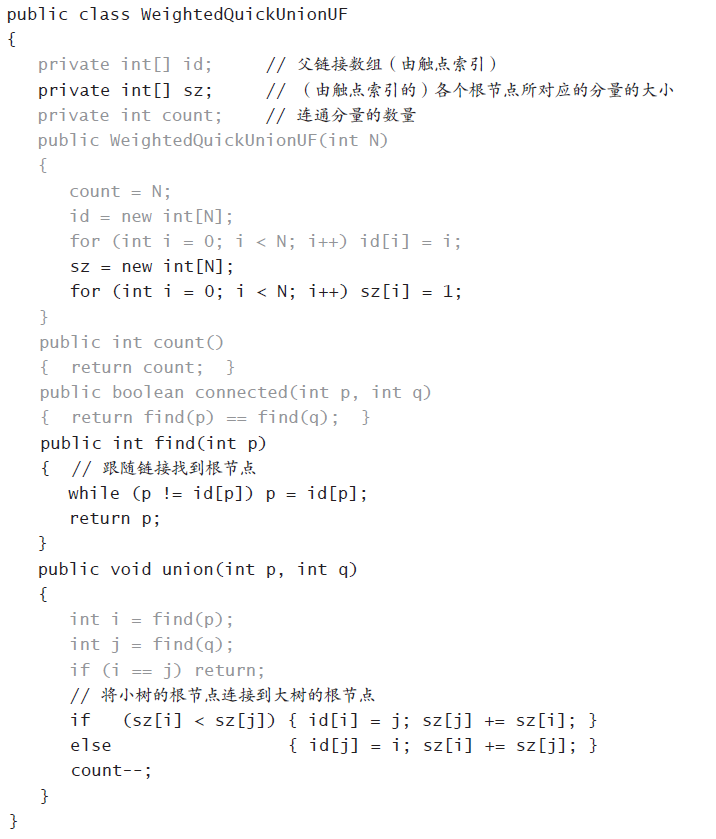
|  |
| --- |
| 其可视化图像为： |

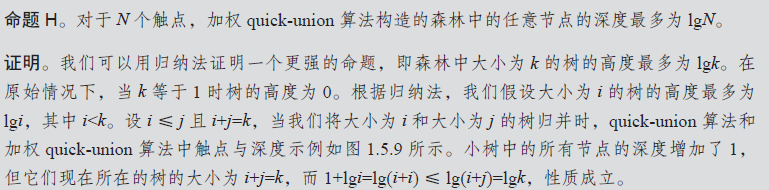
在这里引出了树的概念，

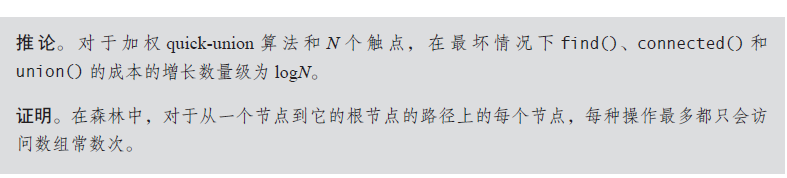


再次改进算法，提出了加权quick-union算法









研究问题时的基本步骤

* 完整而详细的定义问题
* 简介的实现一种初级算法，给出一个精心组织的开发用力并使用实际数据作为输入
* 当实现所能解决的问题的最大规模达不到期望时，决定改进还是放弃
* 逐步改进实现，通过经验性分析或数学分析验证改进后的效率
* 用更高层次的抽象表示数据结构或算法来设计更高级的改进版本
* 如果可能尽量为最坏情况下的性能提供保证，但在处理普通数据时也要有良好的性能
* 在适当的时候讲更细致的深入研究留给有经验的研究者并继续解决下一个问题