实验十四

实验目的：

1. 了解软件测试
2. 深入理解白盒测试和黑盒测试
3. 了解符号测试
4. 了解差分测试

实验内容：

阅读下面软件测试相关资料（或查阅其它相关资料），了解软件测试的基本概念、主要技术、重要挑战等

Software Testing-A Research Travelogue (2000–2014).pdf

1、基本概念

（1）测试目的：确定软件是否满足用户需求和规格说明书中的规定。

（2）测试方法：黑盒测试、白盒测试、灰盒测试等。

（3）测试级别：单元测试、集成测试、系统测试、验收测试等。

（4）测试设计：测试用例设计、缺陷管理、测试报告撰写等。

2、主要技术

（1）自动化测试：对于重复性的测试工作，可以通过编写自动化脚本来自动执行测试。

（2）性能测试：通过模拟真实使用场景来测试软件性能，包括响应时间、负载测试等。

（3）安全测试：测试软件系统是否能够抵御潜在的攻击并保证数据的机密性、完整性和可用性。

（4）移动端测试：针对移动设备的特殊属性和操作系统进行测试。

3、重要挑战

（1）测试资源受限：软件测试需要投入大量人力、财力和时间，但同时也面临着各种资源短缺的问题。

（2）测试用例设计难度高：针对不同场景设计全面有效的测试用例是一项具有挑战性的任务。

（3）软件更新快速：软件迭代速度越来越快，测试过程需要跟随开发流程来及时检测新的问题。

（4）测试结果可靠性差：由于测试过程受到多个因素的影响，测试的结果可能存在误报或漏洞。

综上所述，软件测试在软件开发过程中扮演着至关重要的角色。通过掌握基本概念、主要技术和解决重要挑战，可以更好地提高软件质量和保证用户满意度。

4、其他方面

(1) 测试覆盖率

测试覆盖率是指对于软件规格说明书中给出的需求或者功能模块，测试用例执行的比例。测试覆盖率高，表示测试用例设计充分，可以有效地找到潜在问题，从而更好地保证软件质量。

(2)回归测试

回归测试是指在软件代码发生变动后对已有的测试用例再进行一次测试，以确保修改过的代码不会影响原先已经测试通过的功能模块。回归测试可以很好地帮助开发人员或测试人员确认新改动是否正确。

(3)探索性测试

探索性测试是一种不依赖任何测试计划和测试用例的测试方式，它是根据测试人员的经验和直觉进行测试，逐步挖掘出软件系统中存在的问题。探索性测试可以发现一些非常难以预测的问题，并在最短时间内找到尽可能多的缺陷。

(4) 竞品测试

竞品测试是指针对与本产品功能相似的其他竞争产品进行测试，比较差异之处并提出改进。通过竞品测试能够更好地了解市场状况，发现风险并提高产品的竞争力。

尽管软件测试存在着一些重要的挑战和问题，但是通过合理的测试流程、工具和经验，可以帮助我们有效地降低质量风险并提高软件可靠性。同时，在实施软件测试时，需要不断积累经验和学习最新技术，以适应快速变化的需求。

阅读下面白盒测试和黑盒测试相关资料（或查阅其它相关资料），深入理解白盒测试和黑盒测试，总结其特点 （保存到每个小组选定的协作开发平台上，以组为单位）

WhiteBox.pdf

BlackBox.pdf

在设计软件的过程中，对于软件的测试环节是必不可少的，没有测试的软件不能称得上完全。而对代码的测试又分为两类，分别是白盒测试和黑盒测试。两种测试方法侧重点不同，白盒测试侧重于

代码逻辑，是站在编程者角度对自己所写的代码是否正确的一种检测，而黑盒测试侧重于实现功能，更大程度是站在用户的角度进行的。黑盒测试提供输入与期望输出，重在判断代码能否按照期望的情况执行对应功能。

黑盒测试由于测试目的是站在用户角度看程序是否能正确执行实现所需要的功能，因此不考虑内部实现方法，而是从需求角度进行测试。黑盒测试可以测试代码功能完整性，正确性，是否能正确执行正确终止。黑盒测试想要测试完全，应当对输入划分等价类，在测试代码中包含所有可能的情况。同时选取数据选取在容易出错的边界可以增加测试的可靠性。主要包括如下方面。

1、正确性：计算结果，命名等方面。

2、可用性：是否可以满足软件的需求说明。

3、边界条件：输入部分的边界值，就是使用一般书中说的等价类划分，试试最大最小和非法数据等。

4、性能：正常使用的时间内系统完成一个任务需要的时间，多人同时使用的时候响应时间在可以接受范围内。 J如果在测试过程中发现性能问题，修复起来是非常艰难的，因为这常常意味着程序的算法不好，结构不好，或者设计有问题。因此在产品开发的开始阶段，就要考虑到软件的性能问题。

5、压力测试： 多用户情况可以考虑使用压力测试工具，建议将压力和性能测试结合起来进行。如果有负载平衡的话还要在服务器端打开监测工具，查看服务器 CPU使用率，内存占用情况，如果有必要可以模拟大量数据输入，对硬盘的影响等等信息。

6、错误恢复：错误处理，页面数据验证，包括突然间断电，输入脏数据等。

7、安全性测试：这个领域正在研究中，防火墙、补丁包、杀毒软件等的就不必说了，不过可以考虑。在外国有一种专门干这一行的人叫安全顾问，可以审核代码，提出安全建议，出现紧急事件时的处理办法等。

8、兼容性：不同浏览器，不同应用程序版本在实现功能时的表现不同的上网方式。

白盒测试主要面向代码逻辑，是在能看到全部代码的前提下进行的测试。白盒测试分为静态测试和动态测试，静态测试是看代码不运行，对代码本身进行分析，在基础上确定代码的正确性以及完整性。而动态测需要运行代码，这种测试除了结果外，另一个需要注意的就是覆盖情况，覆盖情况越好说明程序测试越完全。常见的覆盖情况如下。

1、语句覆盖：对于程序中的每个语句都进行了执行，没有没执行过的代码。

2、判定覆盖：对于每一个条件分支，都进行过哦执行。

3、条件覆盖：对于条件分支的每个不同条件都进行了覆盖。

4、判定/条件覆盖：同时满足上面两条。

5、条件组合覆盖：每一个分支的每个条件组合都进行过。

6、路径覆盖：对于程序多个条件分支的每一个路径都进行过覆盖。

以上的几种覆盖方式，越向下，覆盖越完全，同时所需要的成本就越高。白盒测试本身为了准确性就需要对路径的完全覆盖，因此白盒测试是高成本的测试方法。

白盒测试因为可以看到代码，因此和黑盒测试相比可以更好更完全的寻找到代码的错误和漏洞，但是由于白盒测试面向代码设计测试用例，如果代码不完全或者有细微的错误在特定数据，白盒测试还是不能发现问题。而黑盒测试由于侧重于功能，在面对功能不完全的问题上比白盒测试效果更好，而且白盒测试相对成本高，实际两种测试方法还是按照需要进行选择。

黑盒测试和白盒测试的区别：

1）定义：

白盒测试需要从代码句法发现内部代码在算法，溢出，路径，条件等等中的缺点或者错误，进而加以修正。而黑盒测试着重测试软件功能，它并不涉及程序的内部结构和内容特性。黑盒测试并不能取代白盒测试，它与白盒是互补的测试方法，它很可能发现白盒测试不易发现的其他类型错误。

2）测试目的：

黑盒测试的目的是检测是否有不正确或遗漏的功能；数据或者参数上，输入能否正确接收；是否有数据结构错误或外部信息访问错误；性能上是否能够满足要求；是否有初始化或终止性错误。而白盒测试的目的是通过在不同点检查程序的状态，确定实际的状态是否与预期的状态一致，而不顾它的功能。

3）检测方式：

白盒测试是穷举路径测试，黑盒测试是穷举输入测试，这两种方法是基于完全不同的观点，反应了事物的两个极端，它们各有侧重和优势，但不能彼此替代。在现代的测试理念中，这两种测试方法不是截然分开的，而是交叉使用。

阅读下面符号测试 （Symbolic Testing）相关资料（或查阅其它相关资料），了解符号测试的基本概念、主要技术、重要挑战等

A Survey of Symbolic Execution Techniques.pdf

Symbolic Execution and Program Testing.pdf

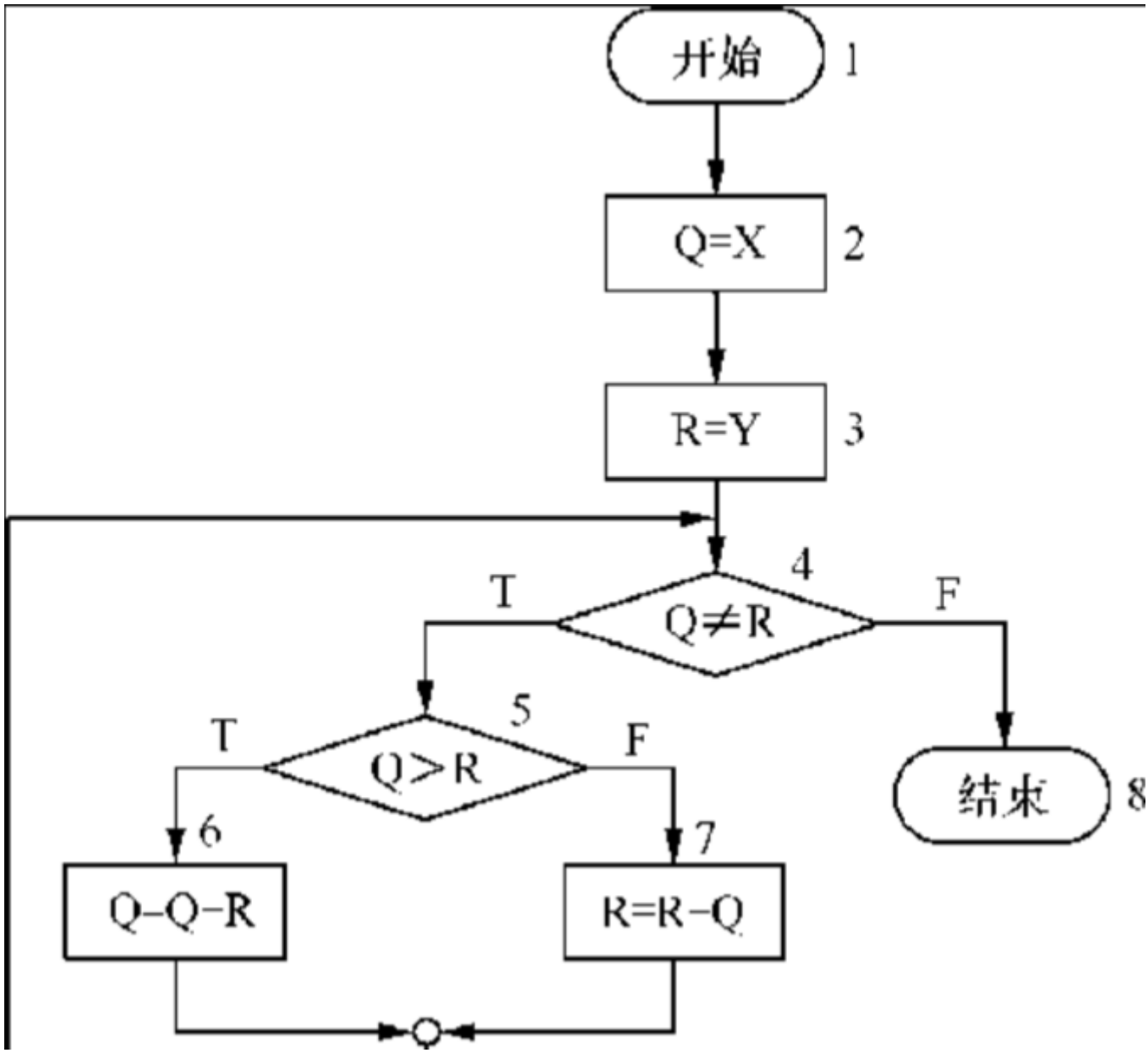
Symbolic Execution for Software Testing-Three Decades Later.pdf

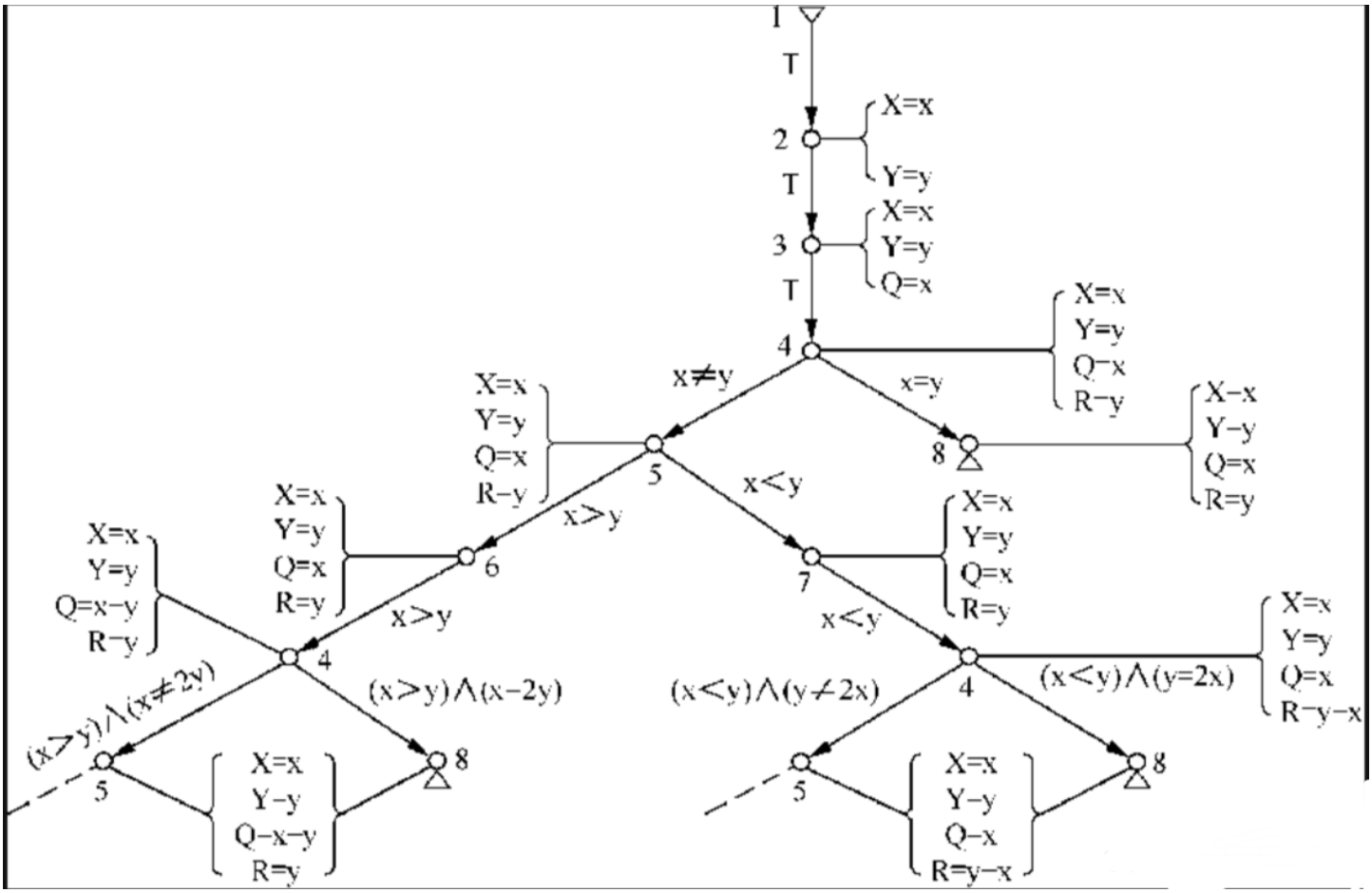
1、符号测试基本概念

符号测试是指对一种符号系统进行评估或分析的过程。这种符号系统可以是一种编程语言、通信协议、数学公式、音乐符号等等。符号测试旨在确定符号系统是否满足特定的标准、规范或预期的行为，并揭示其潜在的问题或缺陷。

普通测试执行的是算术运算，符号测试则是执行代数运算，可以代表一类的测试。其中符号值可以是初等符号值，也可以是表达式。初等符号是任何变量值的字符串，表达式则是数字、算术运算符和符号值的组合。

符号执行树：在条件语句中，判断条件就是谓词，可能是符号表达式，可取真假。不断构造下去，构成了一个二叉树，称为符号执行树。将各个分支点的谓词条件累积在一起，用逻辑乘符号联接在一起，得到的这个逻辑表达式称为路径条件。





需要注意的是，在符号执行树中，节点对应的符号值，对应的是流程图中该标号之前的值。举例来说：在符号执行树第6号节点处的符号值，是还没有执行Q=Q-R时，各符号的值。

2、符号测试主要技术

- 黑盒测试：黑盒测试关注符号系统的外部行为，而不考虑内部实现细节。测试人员根据符号系统的规范或要求设计测试用例，验证输入符号是否产生预期的输出符号。
- 白盒测试：白盒测试考虑符号系统的内部结构和实现细节。测试人员使用这些了解，设计测试用例以覆盖不同的路径、条件和代码段，以发现潜在的错误或缺陷。
- 静态测试：静态测试是在符号系统的开发和设计阶段进行的，而不需要实际执行代码。静态测试技术包括代码审查、静态分析和符号系统模型验证等，以发现设计问题和潜在的错误。
- 动态测试：动态测试是在运行时执行符号系统的测试用例，观察其行为和输出。动态测试技术包括功能测试、性能测试、安全测试和兼容性测试等。
- 单元测试：单元测试是对符号系统中最小功能单元（如函数、方法）进行测试的过程。它通常由开发人员执行，并使用测试框架和断言来验证代码的正确性。
- 集成测试：集成测试是测试符号系统中不同组件或模块之间的交互和整合。它确保各个组件在协调和协作时正常工作，并验证其接口的正确性。

3、符号测试重要挑战：

- 复杂性管理：符号系统可能非常复杂，包含大量的符号和规则。测试人员需要有效地管理复杂性，包括测试用例的设计、管理和执行，以确保测试的全面性和有效性。

- 覆盖率问题：确定如何设计足够的测试用例，以便覆盖符号系统的各个功能、路径和状态。测试人员需要使用测试设计技术，如等价类划分、边界值分析、决策表和状态转换图等，以确保高效的覆盖率。
- 难以预测的行为：某些符号系统可能在特定情况下表现出非确定性或不可预测的行为。这可能是由于复杂的逻辑、多线程环境或依赖于外部因素等原因。测试人员需要考虑边界条件、异常情况和不常见的使用方式，以发现潜在的问题。
- 测试环境准备：为了进行符号测试，可能需要设置适当的测试环境。这包括符号系统的配置、模拟的外部系统或网络连接等。测试人员需要确保测试环境的准备和管理，以便有效地执行测试并获得可重复的结果。
- 持续集成和自动化：对于大型符号系统，持续集成和自动化测试是重要的挑战。这包括建立持续集成管道、自动化测试脚本和工具，以便频繁地执行测试，并及时检测和解决问题。
- 缺陷管理：在符号测试过程中，发现的问题或缺陷需要进行有效的管理。这包括识别、记录、跟踪和报告缺陷，并与开发团队合作进行修复和验证。

阅读下面差分测试 （Differential Testing）相关资料（或查阅其它相关资料），了解差分测试的基本原理、主要应用等

Differential Testing for Software.pdf

Feedback-Directed Differential Testing of Interactive Debuggers.pdf

差分测试是一种软件测试技术，其基本思想是比较两个或多个软件版本的输出差异，从而发现潜在的错误和漏洞。

差分测试的基本流程如下：

1.确定基准版本：选择一个已经测试过并被认为是正确的软件版本作为基准版本，作为比较的标准。

2.生成变异版本：通过修改源代码、更改编译参数、插入错误或其他方式来产生一个或多个变异版本。变异版本应该与基准版本相似但有所不同。

3.构建测试用例：确定一组测试用例，这些测试用例应该能够覆盖软件的不同功能和使用场景。

4.运行测试用例：将基准版本和变异版本都输入相同的测试用例，并记录它们的输出结果。

5.比较输出结果：比较基准版本和变异版本的输出结果，如果有差异，则表明出现了错误或漏洞。

6.分析差异：对差异进行分析，确定差异的原因和影响范围，然后进行错误修复或漏洞修复。

差分测试被广泛应用于以下几个方面：

1.编译器开发：差分测试是编译器开发中的一种重要技术，可以用来测试编译器的正确性和性能。编译器的变异版本可以通过修改编译器源代码、更改编译器参数等方式来产生。

2.操作系统开发：差分测试在操作系统开发中也有广泛应用，可以用来测试操作系统的正确性、稳定性和安全性。操作系统的变异版本可以通过修改内核代码、更改系统配置等方式来产生。

3.网络协议开发：差分测试在网络协议开发中也很常见，可以用来测试协议的正确性、可靠性和安全性。网络协议的变异版本可以通过修改协议实现、更改协议参数等方式来产生。

4.加密算法开发：差分测试在加密算法开发中也有广泛应用，可以用来测试加密算法的安全性和可靠性。加密算法的变异版本可以通过修改加密算法实现、更改加密算法参数等方式来产生。

5.软件安全测试：差分测试也可以用来测试软件的安全性，比如检测软件是否容易受到缓冲区溢出、跨站脚本等攻击。在这种情况下，变异版本通常是通过插入恶意代码、更改输入数据等方式来产生。