软件测试

罗梵峰

2021/5/16

目录

- 信息收集答疑、招新题re答疑
- 软件测试，（想不出来案例分析）
- 后续汇编语言王爽、CSAPP

你看完了我发的PPT吗

为什么是我来分享软件测试……

- 课程《软件工程》，软件质量保证PPT有2个
- 课程《软件测试与质量保证》，PPT有11个

学习软件测试有用吗？

- 无用：
- 测试工程师在鄙视链的位置比较低，不需要计算机专业
- 开发转测试简单，测试转开发难

- 有用：
- 写漂亮的综合设计文档、毕业设计文档
  - 当然，要完整的测试，可能不能只看那个黑盒白盒的PPT
  - 集成测试，功能测试，性能测试……
- 有利于编程时自测。面试，写代码之前编测试数据是加分行为
  - TDD，Test-driven development，测试驱动开发

一名测试工程师走进酒吧……

- 测试是无法覆盖所有的情况的
- 鲁棒性、健壮性：不要随便崩溃

以下是 Oracle 数据库开发人员的日常：

以下是 Oracle 数据库开发人员的日常：

- 开始处理一个新的 bug 。
- 花两周的时间试图理解 20 个不同的 flag ， 这些 flag 以神秘的方式相互交互，导致这个困境。
- 再添加一个 flag 来处理新的特殊场景。添加几行代码来检查此 flag ， 并解决有问题的情况，规避该 bug 。
- 将更改提交到包含大约100-200台服务器的测试服务器集群，这些服务器将编译代码，构建新的 Oracle 数据库，并以分布式方式运行数百万个测试。
- 回家。第二天来上班，继续处理别的 bug 。测试可能需要20-30个小时才能完成。
- 再回家。再来上班，检查你的集群测试结果。顺利的话，会有大约100个失败的测试。倒霉的话，将有大约1000个失败的测试。随机选择一些测试并试图搞清楚你的假设出了什么问题。或许还需要考虑10多个 flag 才能真正理解 bug 的本质。
- 再添加一些 flag 以尝试解决问题。再次提交更改以进行测试。再等20-30个小时。
- 来来回回重复两周，直到你得到了将这些 flag 组合起来的“神秘咒语”。
- 终有一天，你会成功，不再出现测试失败。
- 为你的新更改添加100多个测试，以确保下一个不幸接触这段新代码的开发人员永远不会破坏你的修复。
- 提交最后一轮测试的成果。然后提交以供审核。审查本身可能还需要2周到2个月。所以接下来继续去处理下一个 bug 。
- 在2周到2个月之后，一切已就绪，代码将最终合并到主分支中。

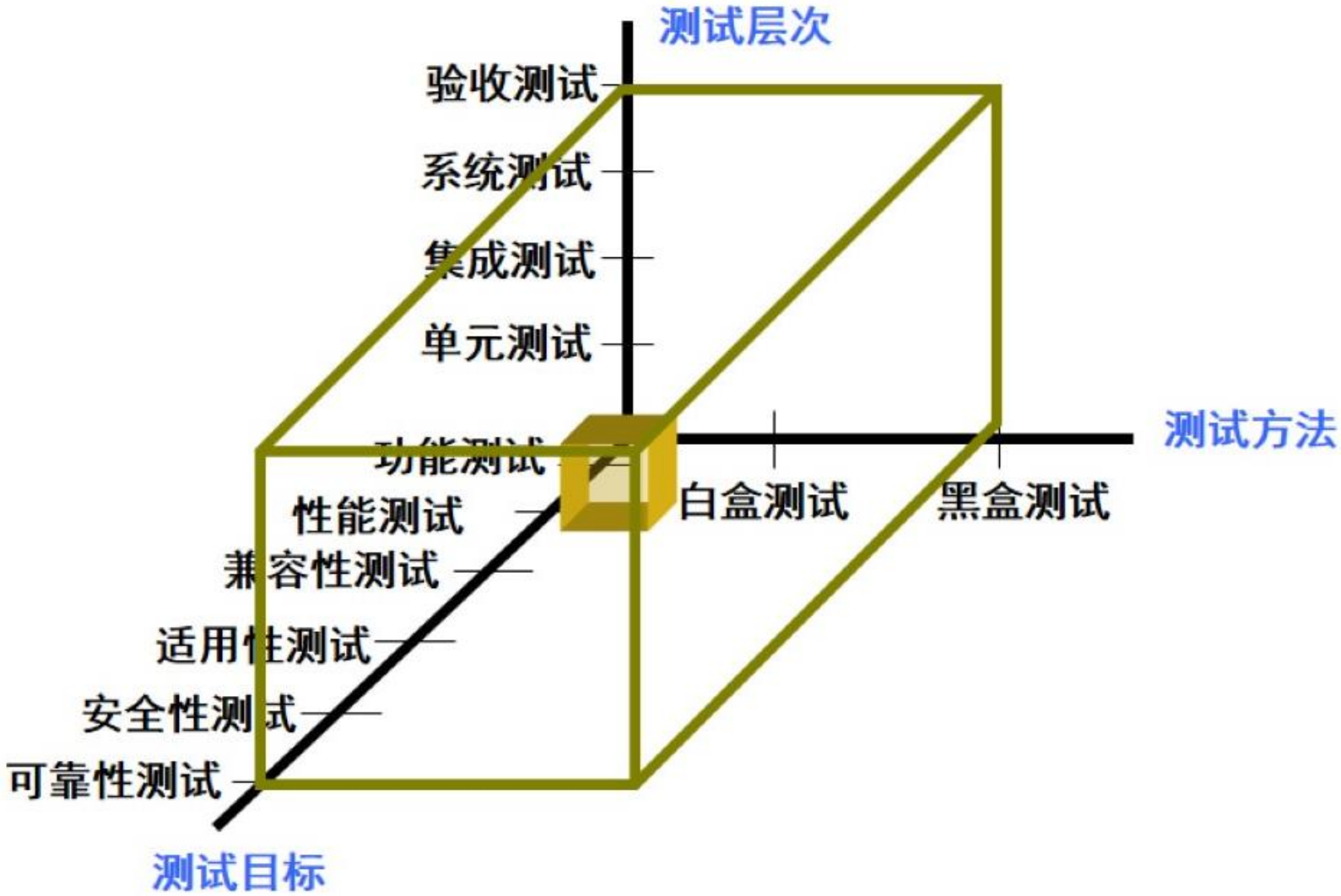
中文：https://www.oschina.net/news/101928/about-oracle-database-code

英文：https://news.ycombinator.com/item?id=18442637

- 概念：回归测试

- 为什么要产生软件工程、为什么要软件测试、为什么会有软件安全缺陷
  - 因为系统是复杂的，需求是复杂的

2.2 软件测试分类



典型软件测试的三维空间

软件测试分类	软件测试方法
黑盒测试	基于直觉与经验法
	等价类划分法
	边界值分析法
	判定表法
	因果图法
	正交实验法
	功能图法
白盒测试	语句覆盖法
	判定覆盖法
	条件覆盖法
	判定条件法
	条件组合覆盖法
	基本路径覆盖法

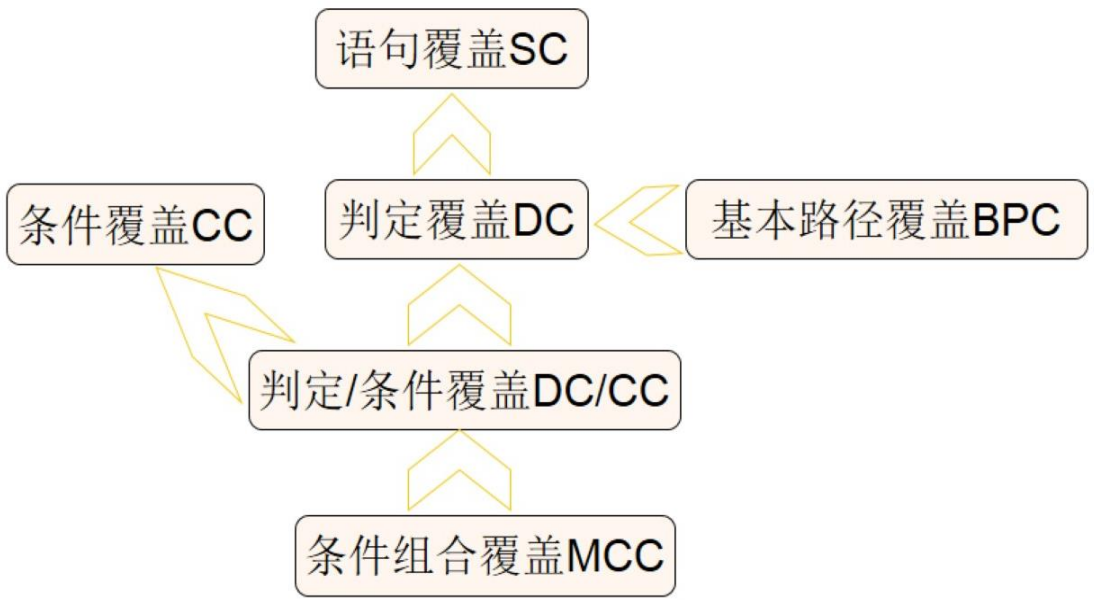
吴晓华:

六种覆盖标准的对比

发现错误能力 弱→强		
	语句覆盖	每条语句至少执行一次
	判定覆盖	每个判定的每个分支至少执行一次
	条件覆盖	每各判定的每个条件应取到各种可能的值
	判定/条件覆盖	同时满足判定覆盖和条件覆盖
	条件组合覆盖	每个判定中各条件的每一种组合至少出现一次
	路径覆盖	使程序中每一条可能的路径至少执行一次

陆鑫:

各个逻辑覆盖方法之间关系



单元测试，桩函数stub

- https://blog.csdn.net/chengyq116/article/details/104681324

课程实验

- Junit 4，单元测试
- QTP，功能测试
- Load Runner，性能测试

测试用例编写，案例分析

- 99+9999=9:98

测试用例编写，案例分析

- 7-23 还原二叉树
  - https://pintia.cn/problem-sets/15/problems/838
  - 一个满分答案
  - https://blog.csdn.net/qq_42623428/article/details/83626437
- 测试用例设计
  - sample
  - 完全右斜
  - 完全左斜
  - 最小N
  - 最大N随机

逆向工程与软件测试的关系

- 没啥特别大的关系，就是猜

模糊测试方法就是通过一个模糊器来构造或产生大量的、随机的数据作为系统输入，从而检验系统在各种数据情况下是否会出现问题。

模糊测试方法一般是通过测试工具来自动执行，其步骤如下：

① 测试工具通过随机或半随机方式产生大量数据

② 测试工具将生成的数据发送给被测试的系统

③ 测试工具检测被测试系统的状态，如是否响应、响应是否正确

④ 根据被测试系统的状态判断是否存在潜在的安全漏洞。

例 windows NT在键盘或鼠标大量随机输入的情况下，有20%的程序会崩溃，还有24%的程序会挂起。

例 模糊测试方法可模拟黑客对系统发动攻击测试，完成安全性测试，并能应用于服务器的容错性测试。

- https://mp.weixin.qq.com/s/y1WiiO67E13iSu3erl7xhg
- “熊孩子”乱敲键盘就攻破了Linux桌面，大神：17年前我就警告过你们
- 导致这个bug的具体行为是：长按“e”键，并在虚拟键盘上选中“ē”。
- 现在，Linux Mint已经为这个漏洞推出了一个新补丁，不过需要自己手动安装。

对排，OI：blind fuzzing

- 一种黑盒fuzzing
- 完全随机vs有侧重的随机（我也不懂，不要问我）
- 要满足题目约束的合法数据

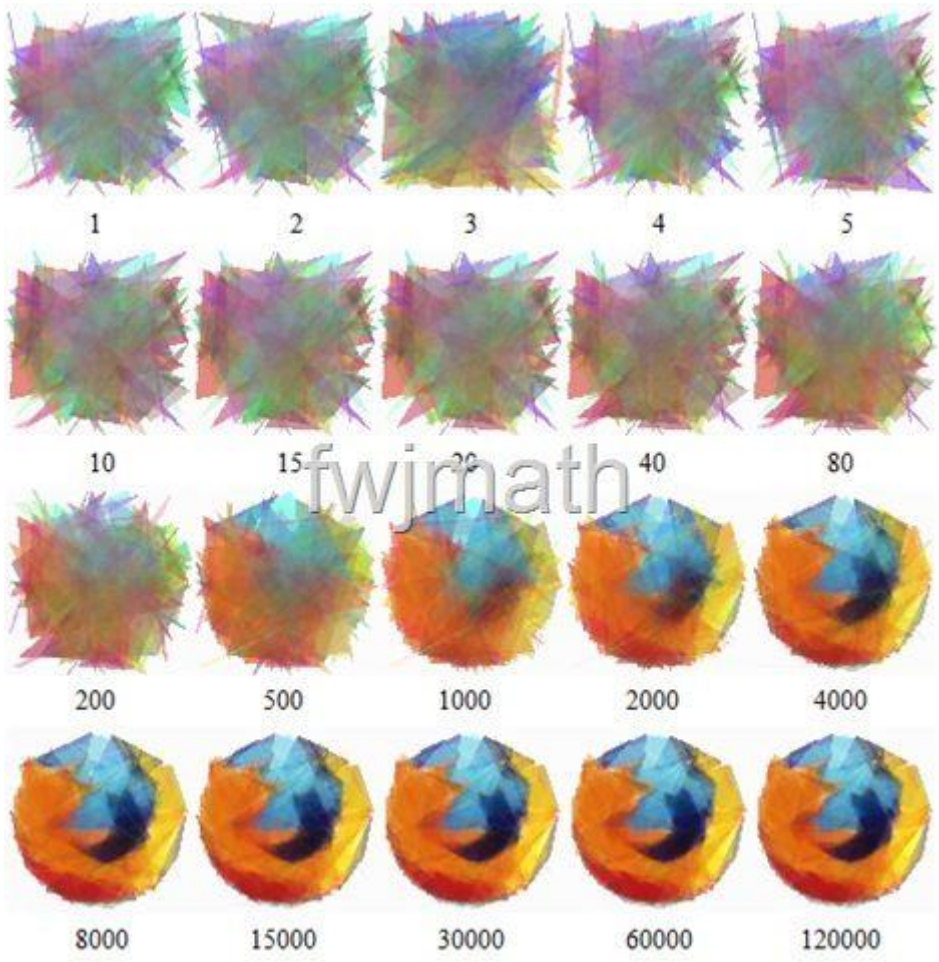
- https://github.com/luogu-dev/cyaron

Fuzzing技术与漏洞挖掘（Web、Pwn）

- 在我的理解里面，Web的Fuzzing就是用电脑跑字典来猜WAF规则，避免用手动猜

- Pwn的Fuzzing是用啥？AFL
  - AFL 这个革命性的二进制漏洞 Fuzz 工具的作者，也是《The Tangled Web: A Guide to Securing Modern Web Applications》这本 Web 安全圣经的作者。
  - https://www.zhihu.com/question/63422112/answer/226070886
- AFL以提高分支覆盖为目标，使用类似遗传算法对输入数据做变异

遗传算法



遗传算法，用100个半透明三角形把Firefox的图标尽可能像地画出来
https://www.zhihu.com/question/23293449/answer/120185075

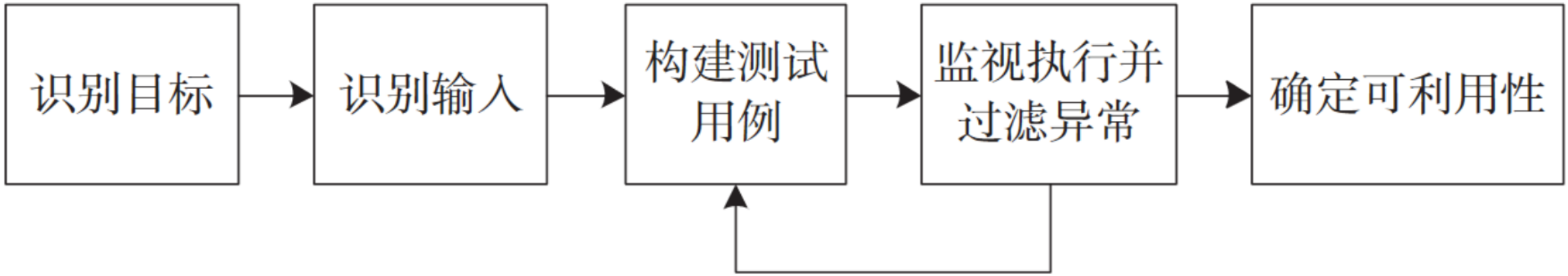


图 6 模糊测试的基本过程

二进制程序漏洞挖掘关键技术研究综述_王夏菁_2017

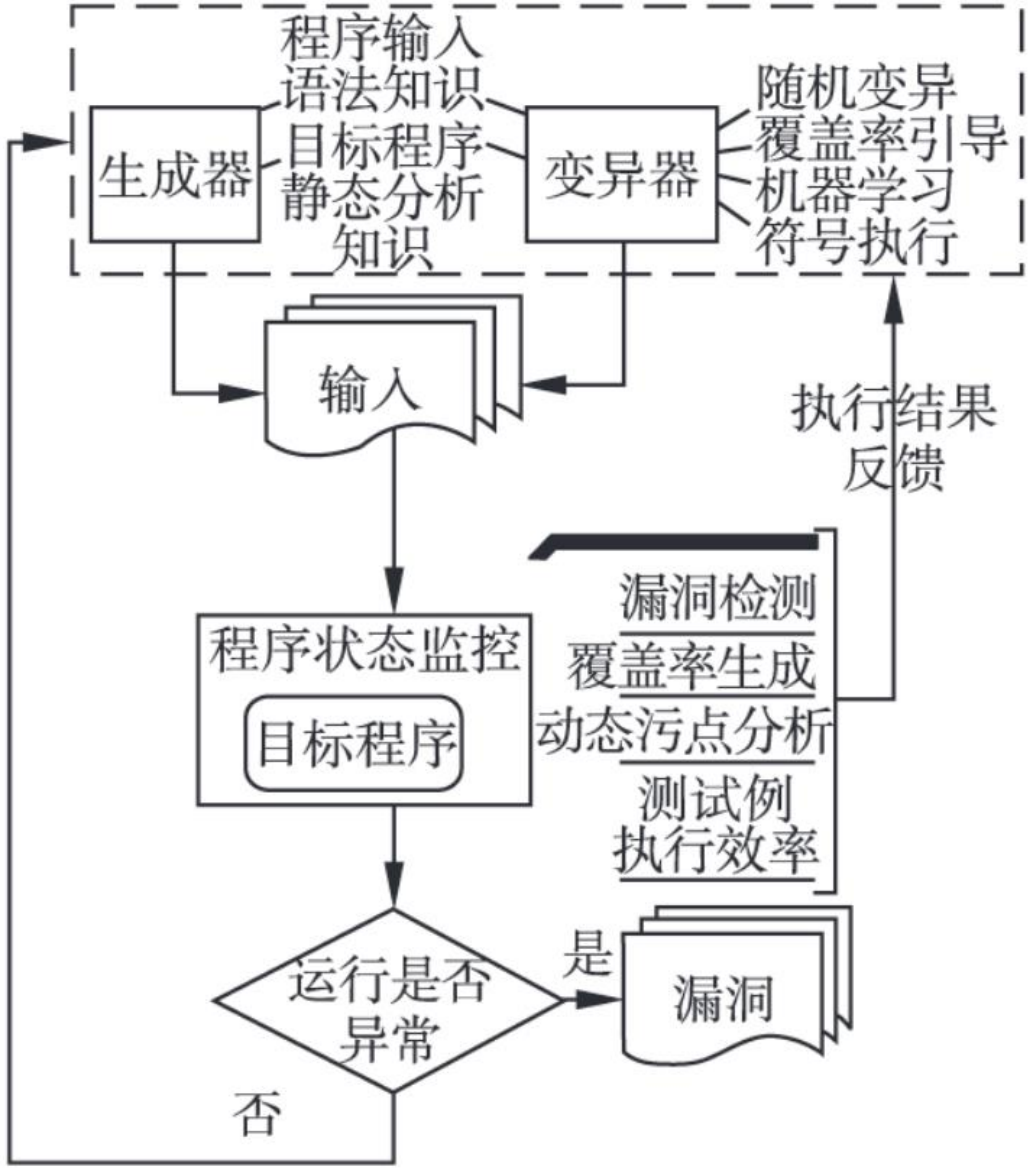


图 1　模糊测试

Fig. 1　Fuzzing test

操作系统内核模糊测试技术综述_李贺_2019

Pwn、逆向工程自动化

后续规划

- 书：汇编语言王爽、CSAPP
- 汇编语言可以不会写，自己整理GDB命令，或者按照别人的整理操作一遍
- CSAPP Lab要写

- 【精校中英字幕】2015 CMU 15-213 CSAPP 深入理解计算机系统课程视频
- https://www.bilibili.com/video/BV1iW411d7hd

CSAPP酷炫例子

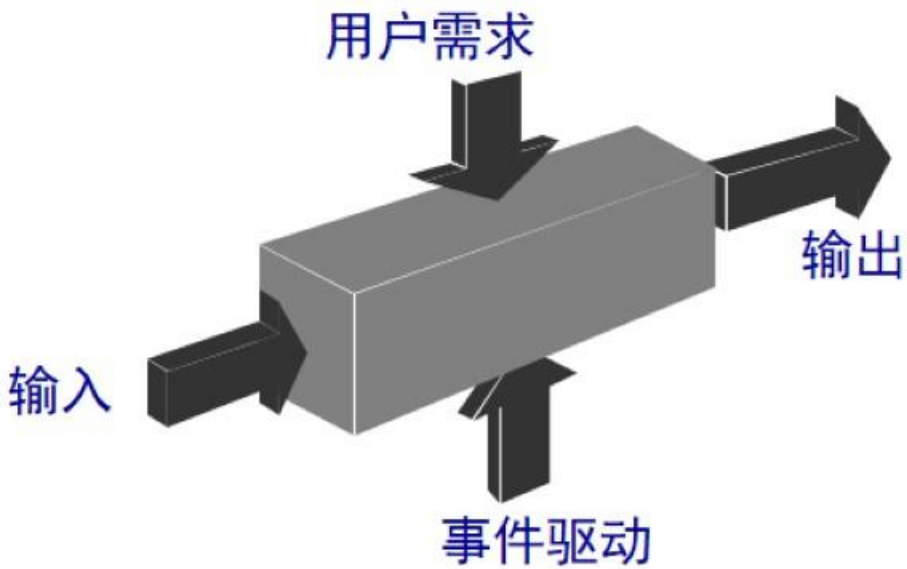
- 整数满足结合律、交换律
- 整数的“非预期”行为
  - 200*300*400*500=-884901888
- 浮点数不满足结合律
  - (3.14+1e20)-1e20=0.0
  - 3.14+(1e20-1e20)=3.14

Q&A

2.5 黑盒测试与白盒测试

一、黑盒测试

黑盒测试是指在测试中，把程序看作一个不能打开的黑盒子。在完全不考虑程序内部结构和内部特性的情况下，对程序功能进行测试，检查程序功能是否按照需求规格说明书的规定正常使用、是否能适当地接收输入数据而产生正确的输出信息。



黑盒测试是从用户观点出发开展的测试，其目的是尽可能发现软件的外部行为错误。黑盒测试常用于发现以下缺陷：

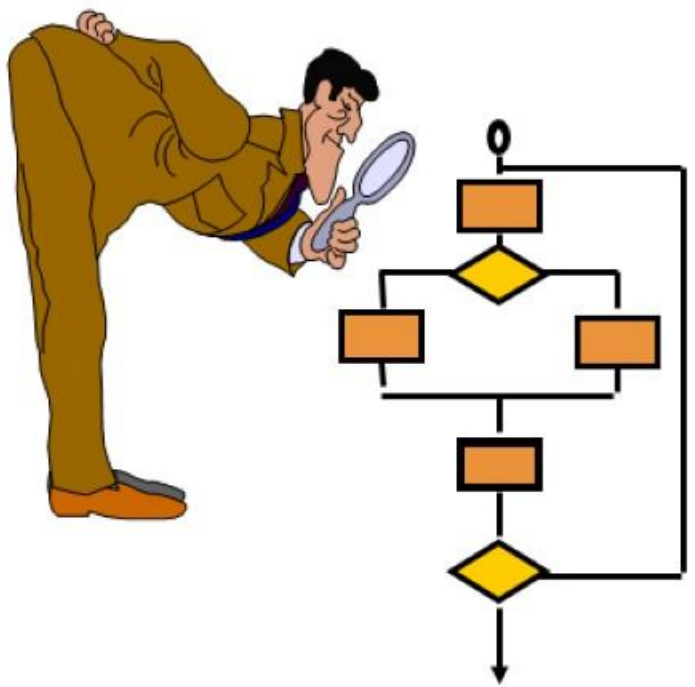
- 检测软件是否有错误的功能或有功能遗漏
- 不能正确地接收输入数据、输出错误的结果
- 功能操作逻辑不合理、不够方便
- 界面出错、扭曲或不美观
- 安装过程中出现问题，安装步骤不清晰、不灵活
- 系统初始化存在问题

黑盒测试特点：

- 黑盒测试与软件具体实现无关，所以如果软件实现发生了变化，测试用例仍然可以使用；
- 黑盒测试用例设计可以和软件实现同时进行，因此可以压缩项目总的开发时间。

二、白盒测试

白盒测试是指在了解被测程序内部逻辑结构情况下，对该程序的内部变量、逻辑结构、运行路径进行测试，检验被测程序的内部动作或运行功能是否符合设计规格要求。



白盒测试常用于发现以下缺陷：

- 程序逻辑错误
- 程序状态异常
- 程序路径无法跳转
- 变量遗漏初始化
- 。。。

白盒测试原则：

- 在执行测试时，先考虑各个分支被覆盖
- 再考虑完成所有逻辑条件分别为“真值”和“假值”的测试
- 若有更高要求，测试出程序流程图中所有独立路径

白盒测试的局限：

- 在一定规模的分支程序中，穷举路径测试不太可能
- 穷举路径测试不能检测出程序违反了设计规范
- 穷举路径测试不能检测出编程遗漏路径
- 可能发现不了一些与数据相关的异常错误

三、测试方法组合

	白盒测试方法	黑盒测试方法
静态测试方法	静态-白盒测试方法 (对源程序代码的语法检查、扫描、评审等)	静态-黑盒测试方法 (对需求文档、需求规格说明书的审查活动，一些非技术性文档测试等)
动态测试方法	动态-白盒测试方法 (在单元测试中，一边运行代码，一边对结果进行检查、验证和调试等)	动态-黑盒测试方法 (在运行程序时，通过数据驱动对软件进行功能测试，从用户角度验证软件的各项功能)

问题：举例说明，什么情况下采用黑盒测试方法？什么情况下使用白盒测试方法？

测试用例编写，案例分析

- 7-43 字符串关键字的散列映射 (25 分)
  - https://pintia.cn/problem-sets/15/problems/890
  - 一个满分答案
  - https://blog.csdn.net/ysm1575491969/article/details/114336422
- 测试用例设计
  - 无冲突
  - 有冲突
  - 有重复关键字
  - 最大和最小字符串，以及不足3位的字符串
  - 最大N，随机
- 作为数据结构习题，考冲突再散列；哈希表大小为素数

- shellcode常用字符串hash
  - https://www.fireeye.com/blog/threat-research/2012/11/precalculated-string-hashes-reverse-engineering-shellcode.html

```
acc := 0;
for c in input_string do
   acc := ROR(acc, 13);
   acc := acc + c;
end
```

- ROR是循环右移指令，把目的操作数整体右移由源操作数指定的位数，被移出的位依次回填到左边空出的位，同时移进标志位。

什么是密码学哈希函数

- 为了保证哈希函数在密码学上的安全性，必须满足以下3个条件
- 抗冲突（collision-resistance）
  - 简单来说，哈希函数抗冲突指的是不同的输入不能产生相同的输出。抗冲突并不是说不会有冲突，只不过找到有冲突的两个输入的代价很大，不可承受。这就好像暴力破解一个有效期为20年的密码，整个破解过程长达30年，虽然最后密码被破解了，但是由于密码有效期过了，所以也就失去了意义。
- 信息隐藏（information hiding）
  - 这个特性是指如果知道了哈希函数的输出，不可能逆向推导出输入。
- 可隐匿性（puzzle friendly）
  - 如果有人希望哈希函数的输出是一个特定的值（意味着有人事先知道了哈希函数的输出结果），只要输入的部分足够随机，在足够合理的时间内都将不可能破解。这个特性主要是为了对付伪造和仿制。
- https://blog.csdn.net/yangxingpa/article/details/82632808