缺陷修复技术

熊英飞
北京大学
2017

报告人介绍 – 熊英飞

- 2000~2004，电子科技大学本科
- 2004~2006，北京大学研究生
  - 导师：梅宏、杨芙清
- 2006~2009，日本东京大学博士
  - 导师：胡振江、武市正人
- 2009~2011，加拿大滑铁卢大学博士后
  - 导师：Krzysztof Czarnecki
- 2012~，北京大学“百人计划”研究员（Tenure-Track）
- 研究方向：软件分析、编程语言设计
- 主页：http://sei.pku.edu.cn/~xiongyf04/

北京大学软件工程研究所

- 国内最早开展软件工程研究、规模最大、最有影响力的软件工程研究团队
- 院士三名（含双聘一名），IEEE Fellow 2名，千人计划1名，博士生导师14名，硕士生导师13名
- 在软件工程顶级会议发表论文数大陆第一
- 获得ACM SIGSOFT杰出论文奖三次，占大陆获奖数一半
- 多名电子科技大学的优秀同学就读/毕业于软件工程研究所
  - 熊英飞（电子科大00级，北大04博）
  - 古亮（电子科大01级，北大05博）
  - 闫宁（电子科大02级，北大06博）
  - 陈俊宇（电子科大13级，北大17博）
  - 严润发（电子科大13级，16年北大实习）

编程语言与开发环境小组

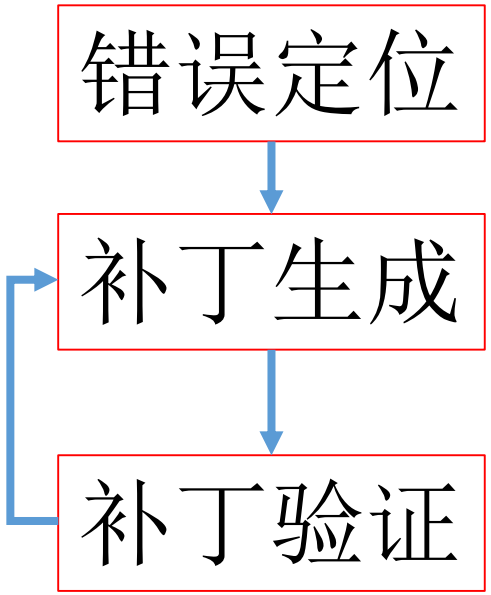
- 指导教师：熊英飞
- 紧密合作：
  - 张路教授（长江学者、杰青）
  - 郝丹副教授（青年长江学者、优青）
- 让计算机学会写程序，将程序员从繁重的体力劳动中解放出来
  - =让程序员下岗？
- 研究路线：从修复缺陷开始，逐步教会计算机写越来越复杂的程序
  - Issue=Bug Report+Feature Request

缘起

- 人和Bug的斗争从来没有停止过
- 缺陷检测：到底有没有Bug
  - 从上世纪60年代开始
  - 代表技术：软件测试、软件验证
- 缺陷定位：Bug在哪里
  - 从上世纪90年代开始
  - 代表技术：统计性调试
- 缺陷修复：自动消除Bug
  - 约从2000年之后开始
  - 代表技术：生成-验证缺陷修复技术

“生成-验证”缺陷修复

输入：一个程序和一组测试，至少有一个测试没有通过
输出：一个补丁，可以使程序通过所有测试



如何定位错误的位置？

基于频谱的错误定位

如何错误定位？

- 程序频谱(Program Spectrum)
  - 最早由威斯康星大学Tom Reps于1997年在处理千年虫问题时发明
  - 指程序执行过程中的统计量
- 基于频谱的错误定位
  - 佐治亚理工James Jone, Mary Jean Harrold等人2002把Tom Reps的方法通用化成通用调试方法
  - 主要用到的频谱信息为测试覆盖信息
  - 形式简单，效果较好
  - 使用最广泛的自动化错误定位方法

基于频谱的错误定位

- 基本思想
  - 被失败的测试用例执行的程序元素，更有可能有错误
  - 被成功的测试用例执行的程序元素，更有可能没有错误
- 程序元素可以定义在不同的粒度上
  - 基本块
  - 方法
  - 类
  - 文件

例子

		T15	T16	T17	T18
1	int count; int n; Ele *proc; List *src_queue, *dest_queue; if (prio >= MAXPRIO) /*maxprio=3*/	●	●	●	●
2	{return;}	●			

3	src_queue = prio_queue[prio];
	dest_queue = prio_queue[prio+1];
	count = src_queue->mem_count;
	if (count > 1) /* Bug*//* supposed : count>0*/ {
			●	●	●

4	n = (int) (count*ratio + 1);
	proc = find_nth(src_queue, n);
	if (proc) {
			●	●	

5	src_queue = del_ele(src_queue, proc);
	proc->priority = prio;
	dest_queue = append_ele(dest_queue, proc); }
			●	●	

	Pass/Fail of Test Case Execution :	**Pass**	**Pass**	**Pass**	**Fail**

计算程序元素的怀疑度

- a_{ef}：执行语句a的失败测试的数量，a_{nf}：未执行语句a的失败测试的数量
- a_{ep}：执行语句a的通过测试的数量，a_{np}：未执行语句a的通过测试的数量
- Tarantula: $\frac{a_{ef}}{a_{ef}+a_{nf}} / \left(\frac{a_{ef}}{a_{ef}+a_{nf}} + \frac{a_{ep}}{a_{ep}+a_{np}} \right)$
- Jaccard: $\frac{a_{ef}}{a_{ef}+a_{nf}+a_{ep}}$
- Ochiai: $\frac{a_{ef}}{\sqrt{(a_{ef}+a_{nf})(a_{ef}+a_{ep})}}$
- D*: $\frac{a_{ef}^{*}}{a_{nf}+a_{ep}}$，*通常设置为2或者3
- Naish1: $\begin{cases} -1 & a_{nf} > 0 \\ a_{np} & a_{nf} = 0 \end{cases}$

如何生成补丁？

GenProg

GenProg

- 2009年由弗吉尼亚大学Westley Weimers和Claire Le Goues提出
- 标志缺陷修复技术兴起的代表性工作
- 全自动修复程序中的缺陷，通过所有测试
- 遵循“生成——验证”过程

GenProg生成补丁

- 基本思路：天下程序一大抄
- 变异操作
  - 复制别的语句替换当前语句
  - 在当前语句之后插入别的语句
  - 删除当前位置语句
- 遗传操作
  - 选择两个适应度高的程序
  - 交换其中的变异操作
- 适应度计算
  - 通过测试越多，程序适应度越高

实验设置

- 2012年GenProg大型实验
- 实验对象：105个真实大型程序中的缺陷
  - 选择行数>50000行，测试>10个，修改历史>300的程序
  - 用最新版本的测试用例检查缺陷，如果旧版本不能通过新版本的某个测试用例，则最新的一个不能通过的旧版本作为有缺陷的程序
- 该测试集日后发展为ManyBugs标准测试集

实验效果

- 实验结论
  - 105个缺陷修复了55个
  - 总共花费403美元，平均7.32美元一个，每个耗时半天左右
  - 作者手动检查了2个缺陷，发现GenProg的修复都和人工修复是等价的

GenProg的改进-AE

- 2013年由Westley Weimer等人提出
- 在修改的时候避免产生等价变换
- 设置了一系列规则快速检查特定类型的等价变换
- 在GenProg的测试集上做了验证，开销约为原来的三分之一

GenProg工作影响

- 第一篇ICSE09论文被评为Distinguished Paper
- 7年总引用过1000次
- 论文主要博士生被CMU聘为Assistant Professor

程序员真的快要下岗了吗？

GenProg质疑

2011年Lionel Briand教学论文

- Andrea Arcuri, Lionel C. Briand: A practical guide for using statistical tests to assess randomized algorithms in software engineering. ICSE 2011: 1-10
- 指出现有很多论文统计方法应用不当
- 特别是很多方法连随机方法都不对比
  - 特别点名GenProg论文

RSRepair

- 起源于国防科技大学毛晓光老师团队的系列研究
- ICSM 2012: 将程序分块编译好，这样之后只需要重新编译变化的部分，加快编译速度
- ICSM 2013: 将测试排序技术和修复验证相结合，以期望更快的发现修复不成功
  - 需要放弃遗传算法，因为无法计算适应度
- ICSE 2014: GenProg中的遗传算法不如随机搜索
  - 主要原因：计算适应度函数代价太大
  - RSRepair：将GenProg中的遗传算法换成随机搜索，同时对测试排序，发现效果显著优于GenProg

PAR

- 香港科技大学Sung Kim等人提出
- 替换GenProg中的变异模板为人工模板，如：
  - 在问题语句前面插入null检查
  - 更改方法调用中的参数变量
  - 重新调用一个签名相同但名字不同的方法
- 在119个Java缺陷上做了验证
  - Par修复了27个，GenProg修复了16个
- 嗯？好像16/119和55/105差得有点多？
  - 莫非Java程序抄不出来？

Kali

- 源于2015年麻省理工大学Martin Rinard的论文
- 验证了GenProg、AE、RSRepair
- 以GenProg为例说明结果
  - 414个补丁中只有110个通过测试，修复18个缺陷，而不是55个（总共105个）
  - 110个通过测试的补丁中经过人工检验只有5个正确，该5个补丁修复了2个缺陷
  - 大多数补丁都是简单的删除出错的功能
- 专门设计了只删除功能的Kali，发现效果和GenProg相当

后GenProg时代工作

后GenProg时代

- Martin Rinard的论文暴露出现有修复技术的主要问题是不能以通过测试为目标
- 修复技术的目标调整为生成和原程序员补丁相同的补丁
- 基本手段：对补丁进行排序，优先验证能通过测试的补丁

DirectFix和Angelix

- 新加坡国立大学Ahibk Roychoudhury团队的工作
- 生成语法上差别最小的修复
  - 用语法树上被改动的元素个数定义差别
  - i < 1 ➔ i <= 1　　较好修复
  - i < 1 ➔ isZero(i) <= a*b+c+data.size()　　较差修复
- ManyBugs数据集上的缺陷修复数量
  - 105个缺陷，通过测试28个，正确10个
  - 正确率：35.7%
  - 召回率：9.5%

Qlose

- 微软Rishabh Singh等人的工作
- 把语法上的差别最小改成了语义差别最小
- 语义差别最小定义为
  - 运行是变量的取值差别最小
  - 执行的控制流差别最小
- 但只在作业程序上做了验证，没有在大型程序上验证

Prophet

- Martin Rinard团队龙凡的工作
- 用机器学习方法对可能生成的补丁进行排序，按正确的可能性从大到小排列
- ManyBugs数据集上的缺陷修复数量
  - 105个缺陷，通过测试42个，正确15个
  - 正确率：35.7%
  - 召回率：17.1%
- 目前C语言上最好的修复工具

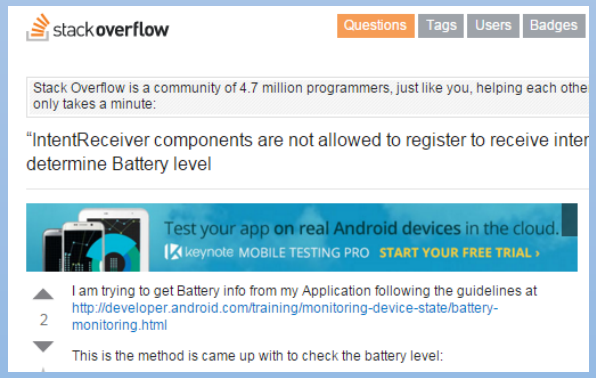
缺陷修复技术还有前途么？

正确率不到40%的技术在实践中基本无法使用

我们的工作

高正确率的缺陷修复（正确率80%左右）

从QA网站学习 [ASE15]



精准条件修复 [ICSE17]

[ASE15] Qing Gao, Hansheng Zhang, Jie Wang, Yingfei Xiong, Lu Zhang, Hong Mei. Fixing Recurring Crash Bugs via Analyzing Q&A Sites. ASE'15

[ICSE17] Yingfei Xiong, Jie Wang, Runfa Yan, Jiachen Zhang, Shi Han, Gang Huang, Lu Zhang. Precise Condition Synthesis for Program Repair. ICSE'17

电子科技大学13级严润发同学

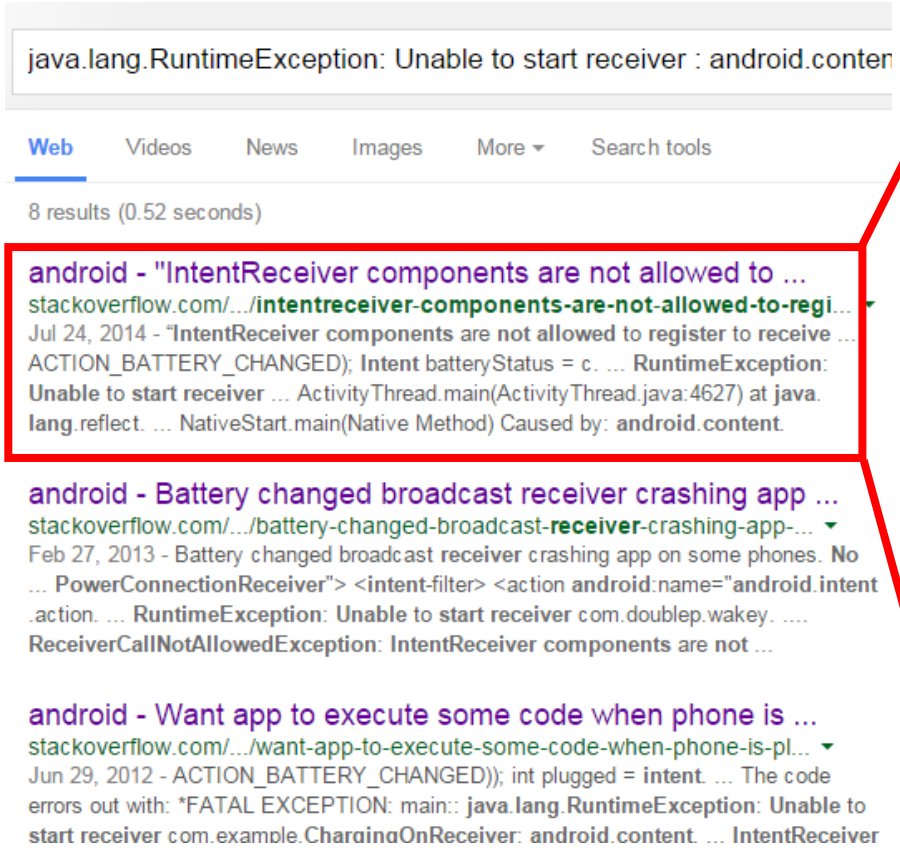
从QA网站学习

- 开发人员遇到未知错误的时候会怎么办？

```
29    public void onReceive (final Context context, final Intent intent) {
30       final int action = intent.getExtras().getInt(KEY_ACTION, -1);
31       final float bl = BatteryHelper.level(context);
32       LOG.i("AlarmReceiver invoked: action=%s bl=%s.", action, bl);
33       switch (action) {
...         ...
51       }
52    }
```

java.lang.RuntimeException: Unable to start receiver com.vaguehope.onosendai.update.AlarmReceiver:

从QA网站学习


java.lang.RuntimeException: Unable to start receiver : android.conten
Web
Videos
News
Images
More
Search tools
8 results (0.52 seconds)
android - "IntentReceiver components are not allowed to ...
stackoverflow.com/.../intentreceiver-components-are-not-allowed-to-regi...
Jul 24, 2014 - "IntentReceiver components are not allowed to register to receive ... ACTION_BATTERY_CHANGED); Intent batteryStatus = c. ... RuntimeException: Unable to start receiver ... ActivityThread.main(ActivityThread.java:4627) at java.lang.reflect. ... NativeStart.main(Native Method) Caused by: android.content.
android - Battery changed broadcast receiver crashing app ...
stackoverflow.com/.../battery-changed-broadcast-receiver-crashing-app-...
Feb 27, 2013 - Battery changed broadcast receiver crashing app on some phones. No ... PowerConnectionReceiver"> <intent-filter> <action android:name="android.intent.action. ... RuntimeException: Unable to start receiver com.doublep.wakey. .... ReceiverCallNotAllowedException: IntentReceiver components are not ...
android - Want app to execute some code when phone is ...
stackoverflow.com/.../want-app-to-execute-some-code-when-phone-is-pl...
Jun 29, 2012 - ACTION_BATTERY_CHANGED)); int plugged = intent. ... The code errors out with: *FATAL EXCEPTION: main:: java.lang.RuntimeException: Unable to start receiver com.example.ChargingOnReceiver: android.content. ... IntentReceiver


stackoverflow
Questions
Tags
Users
Badges
Stack Overflow is a community of 4.7 million programmers, just like you, helping each othe
only takes a minute:
"IntentReceiver components are not allowed to register to receive inter
determine Battery level
Test your app on real Android devices in the cloud.
keynote MOBILE TESTING PRO START YOUR FREE TRIAL ›
2
2
I am trying to get Battery info from my Application following the guidelines at http://developer.android.com/training/monitoring-device-state/battery-monitoring.html
This is the method is came up with to check the battery level:
public void sendBatteryInfoMessage(){
IntentFilter iFilter = new IntentFilter(Intent.ACTION_BATTERY_
Intent batteryStatus = c.registerReceiver(null, iFilter);

从QA网站学习的困难

- 自然语言理解是很困难的

Instead of:

4

```
context.registerReceiver(null, new IntentFilter(Intent.ACTION_BATTERY_CHANGED));
```

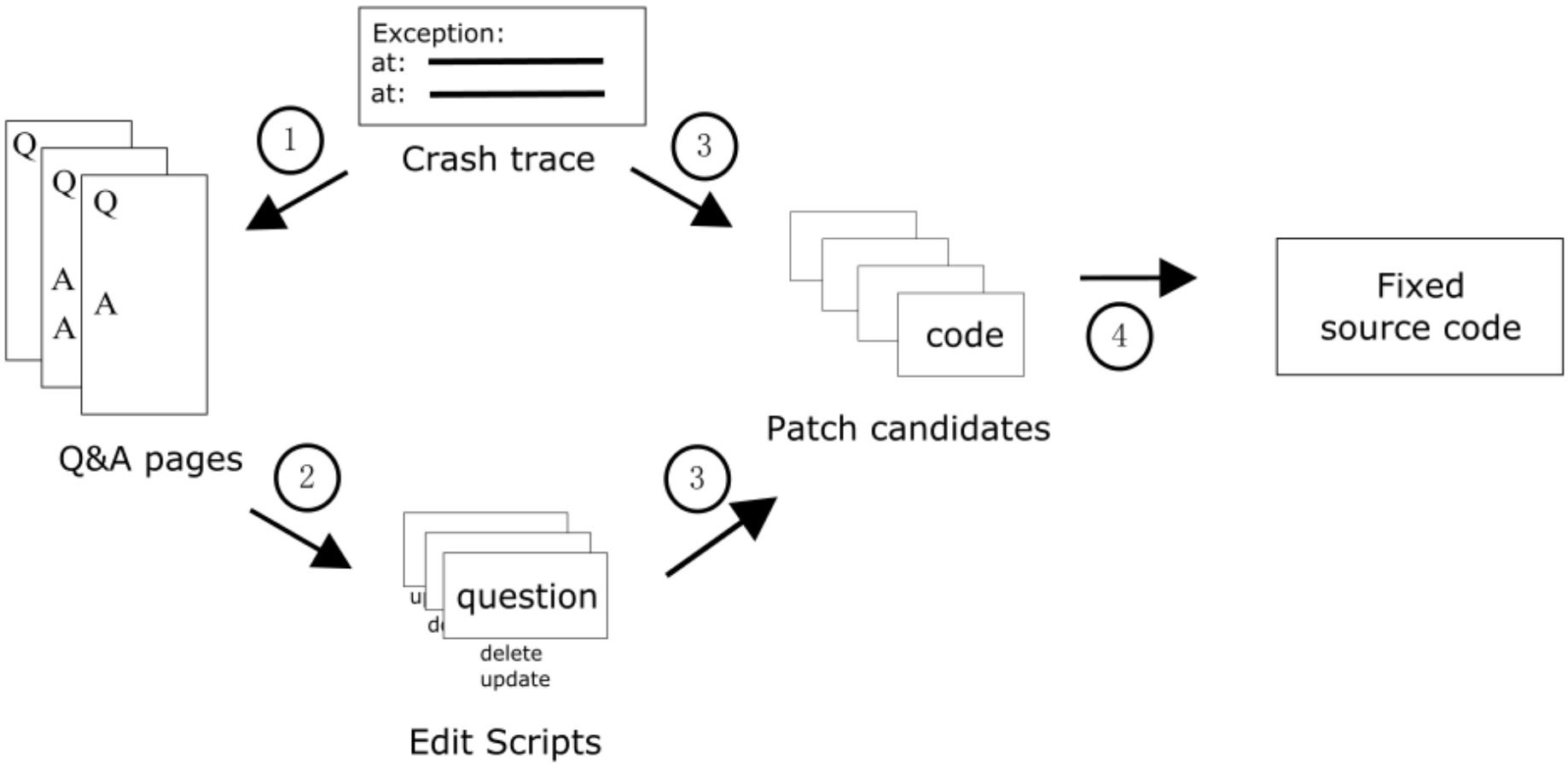
use:

```
context.getApplicationContext().registerReceiver(null, new IntentFilter(Intent.ACTION_BATTER
```

This is annoying -- `registerReceiver()` should be smarter than this -- but it's the workaround for this particular case.

- 观察：程序员常常只用编程语言语言交流的
- 解决方案：直接比较代码片段

方法概览


Exception:
at:
at:
Crash trace
1
3
Q
Q
Q
A
A
A
Q&A pages
code
Patch candidates
4
Fixed
source code
2
3
question
delete
update
Edit Scripts

实验效果

- 24个Android崩溃缺陷
  - 预先人工验证过在StackOverflow上能找到答案

- 正确修复：8
- 错误修复：2
- 正确率：80%
- 召回率：33%

精确条件修复

条件错误是很常见的

```
   lcm = Math.abs(a+b);
+ if (lcm == Integer.MIN_Value)
+   throw new ArithmeticException();
```

缺少边界检查

```
-  if (hours <= 24)
+ if (hours < 24)
      withinOneDay=true;
```

条件过强

```
-  if (a > 0)
+ if (a >= 0)
      nat++;
```

条件过弱

ACS修复系统

- ACS = Accurate Condition Synthesis
- 两组修复模板

条件修改

- 首先定位到有问题的条件，然后试图修改条件
  - 扩展：if ($D) => if ($D || $C)
  - 收缩：if ($D) => if ($D && $C)

返回预期值

- 在出错语句前插入如下语句
  - if ($C) throw $E;
  - if ($C) return $O;

挑战和解决方案

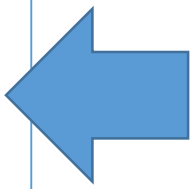
```
 int lcm=Math.abs(
    mulAndCheck(a/gdc(a,b),b));
+if (lcm == Integer.MIN_VALUE) {
+   throw new ArithmeticException();
+}
 return lcm;
```

测试 1:
Input: a = 1, b = 50
Oracle: lcm = 50

测试 2:
Input: a = Integer.MIN_VALUE, b = 1
Oracle: Expected(ArithmeticException)

正确条件:
lcm == Integer.MIN_VALUE

可以通过测试的条件:
- a > 1
- b == 1
- lcm != 50
- …



排序

排序方法1:
按数据依赖对变量排序

- 变量使用局部性：最近被赋值的变量更有可能被使用。

- 根据数据依赖对变量排序
  - lcm = Math.abs(mulAndCheck(a/gdc(a, b), b))
  - lcm > a, lcm > b

排序方法2:
根据Java文档过滤变量


```
/** ...
 * @throws IllegalArgumentException if initial is not between
 * min and max (even if it <em>is</em> a root)
**/
```

抛出IllegalArgumentException时，只考虑将“initial”变量用在条件里

排序方法3:
根据现有代码对操作排序

- 在变量上使用的操作跟该条件的上下文紧密相关

变量类型

```
Vector v = …;
if (v == null) return 0;
```

变量名字

```
int hours = …;
if (hours < 24)
      withinOneDay=true;
```

方法名字

```
int factorial() {
  …
  if (n < 21) {
    …
```

- 根据已有的代码库统计条件概率

Defects4J上的验证

Approach	Correct	Incorrect	Precision	Recall
ACS	18	5	78.3%	8.0%
jGenProg	5	22	18.5%	2.2%
Nopol	5	30	14.3%	2.2%
xPAR	3	–[4]	–[4]	1.3%[2]
HistoricalFix[1]	10(16)[3]	–[4]	–[4]	4.5%(7.1%)[2,3]

缺陷修复展望

- 虽然困难，但仍然充满希望的新领域
- 学术界最活跃的研究领域之一
  - 2013年、2016年均有Dagstuhl召开
- 工业界大量关注和投入
  - 谷歌、华为、360、富士通
- 最终通往自动编程的可行途径

- 欢迎同学们加入我们！