课程项目1

- 感谢唐浩、朱琪豪、肖元安、陈逸凡、叶振涛对开发包和评测平台的贡献
- 实现一个Java上的指针分析系统
- 提交一个自己编写的测试样例，包括代码和标准输出
- 测试用例评分规则：
  - 无法在测试程序上正常运行的不合格，0分
    - 如：超时（1分钟），崩溃
  - 在测试程序上能输出结果，但结果不健壮(unsound)，1分
  - 结果健壮，根据精度分数在1-2之间
- 总分：
  - **基础得分**：所有测试用例得分之和，满分100分
    - 公开的6个测试用例为54分，剩余测试用例为46分。
  - **Bonus分数**：学生提交测试用例的质量，满分10分
    - 除提交者外存在一组能完美通过可获得6分
    - 不是所有组都完美通过，则再奖励（未通过组数的比例*4分）
  - **总分**=min(100, 基础得分+Bonus分数)
  - 源代码和报告作为评分参考
- 组队完成：
  - 3名同学一个小组
  - 组内贡献不均等的，请在提交的时候说明

程序样例

输入程序：

```
  public static void main(String[] args) {
    BenchmarkN.alloc(1); //标记分配点，没有标记的默认编号为0
    A a = new A();
    BenchmarkN.alloc(2);
    A b = new A();
    BenchmarkN.alloc(3);
    A c = new A();
    if (args.length>1) a=b;
    BenchmarkN.test(1, a); //标记测试点编号和被测变量
    BenchmarkN.test(2, c);
  }
```

输出：

```
1: 1 2
2: 3
```

每行一个测试点，以测试点编号开头。
冒号后面是可能的分配点，多个分配点以空格分割

开发平台

- Java上常见静态分析平台（自学）：
  - 太阿
  - SOOT/Sootup
  - WALA
  - Chord
- 部分平台已经自带指针分析，要求
  - 不能直接调用平台的指针分析模块或依赖于指针分析的功能（比如控制流分析/调用图构建）
  - 可以使用平台提供的其他支撑，比如数据流分析框架，过程内控制流图构建，Java语言化简等

测试样例Java语言限制

- 不允许Java 1.4以后的语言特性
  - 包括泛型、枚举、lambda表达式等
- 不允许
  - 额外的`import`语句（即JDK只能使用java.lang下面内容），除了必要的`import benchmark.internal.Benchmark`和平凡的`import benchmark.objects.*`
  - 反射
  - annotation
- 允许
  - 数组和多维数组
  - 继承、重写
  - 接口
  - 强制类型转换
  - 异常

时间节点和提交内容

- 组队报给助教（ddl: 11月6日）
- 根据同学完成情况公开部分样例性质
- 开放排行榜（11月23日）
- 代码提交（11月25日）
  - Readme.pdf：A4两页以内，描述算法的主要设计思想，小组成员姓名、学号和分工，发邮件给助教
  - Code目录：项目源代码，发邮件给助教
  - 编译好的分析工具和测试样例：根据网站要求提交
- 现场报告（11月27日）
  - 排行榜前10组交流所采用的算法，每组报告8分钟，讨论2分钟

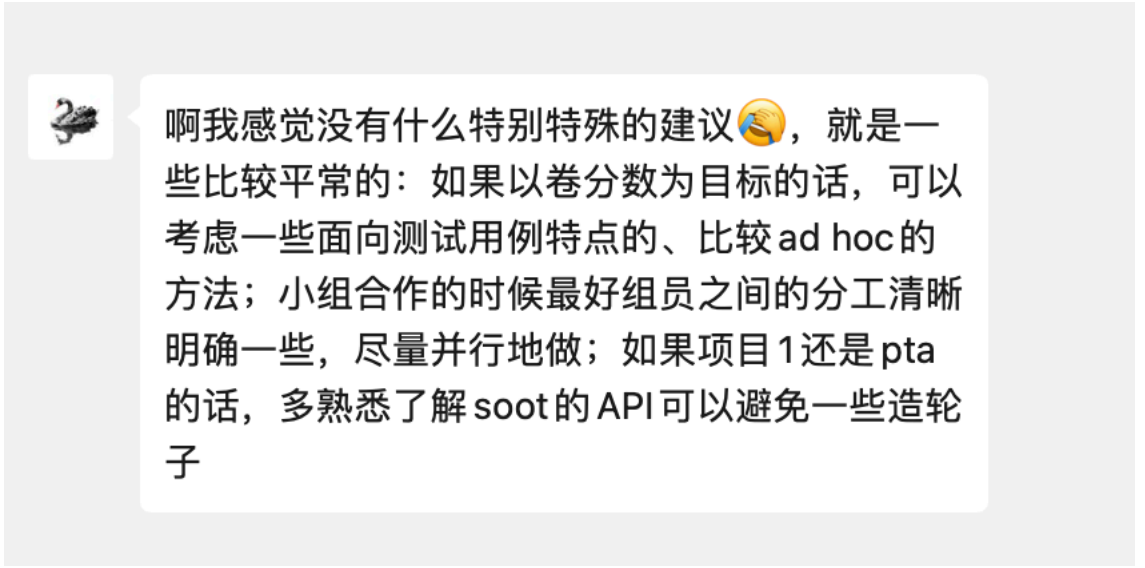
测试程序

- Linux JDK17 实时给出程序运行结果
- 请使用队长的学号提交
- 提交网站：162.105.88.145:11451
- 组队完成后，队长发邮件助教获得账号密码
  - 助教邮箱地址：kkogoro@stu.pku.edu.cn
  - 邮件标题：[Lab1组队]队长学号
  - 邮件正文：所有队员的学号和姓名

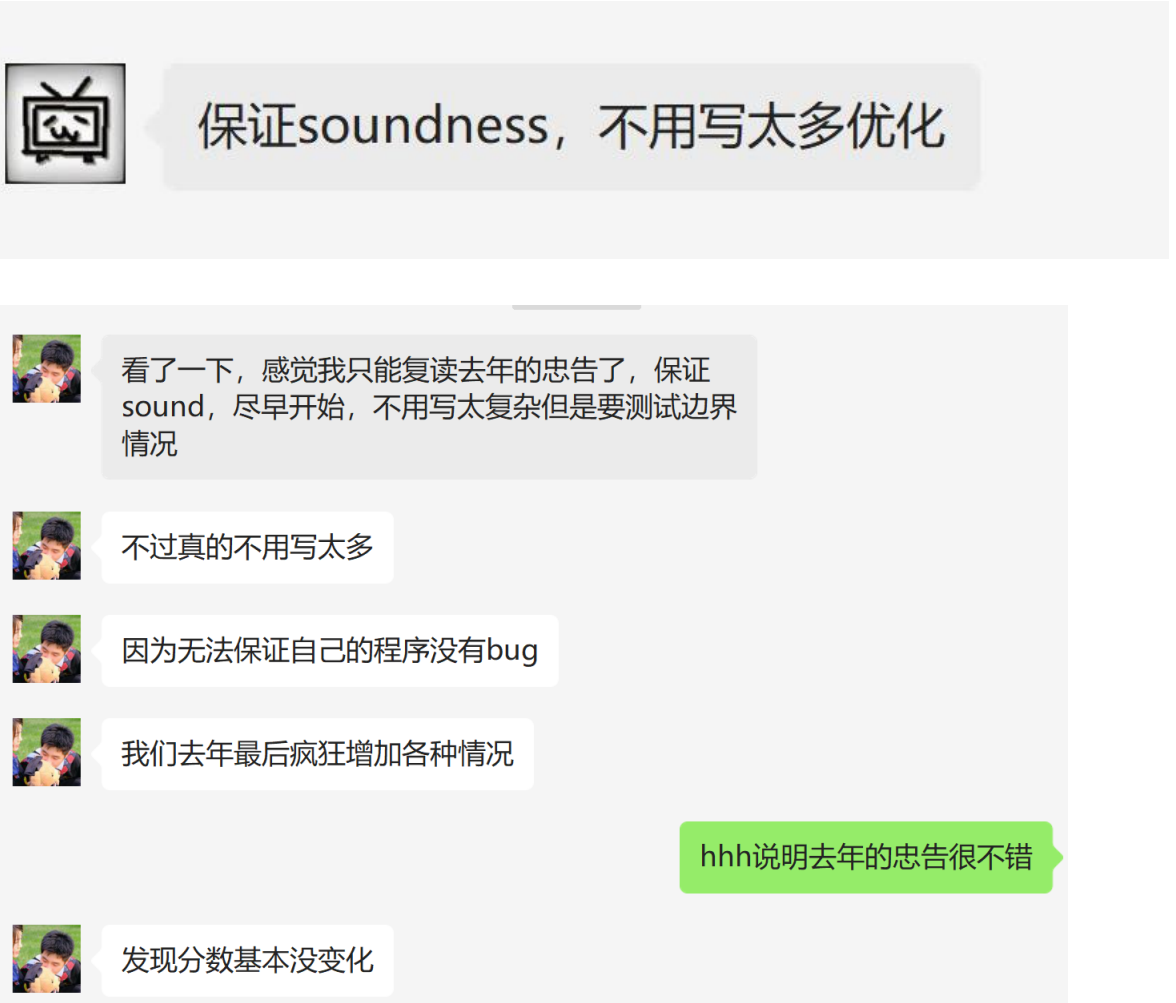
助教的忠告

- 太阿文档较少，可以早点熟悉一下API。
- 保证sound、不崩溃以及不超时是得分高的关键
- Corner Case是更进一步的关键
- 限制使用的计算资源，并行不是可行的优化策略
- 今年“公开测试用例”难度较往年有所增加，建议适当提前开始

一些透露了头像的前辈的忠告



一些透露了头像的前辈的忠告



一些透露了头像的前辈的忠告


我就记得流敏感不是那么重要，域敏感比较重要


保证分析结果sound比啥都重要！


还有吐槽一点..soot的javadoc约等于没有，需要靠函数名猜功能（x


建议先把算法要实现的规则写出来再写代码


不然debug会哭

一些不透露头像的前辈的忠告 (from 23 fall)

lab1的话我印象里就是规则的广度大于规则的“深度”，不一定要在某个上面做的多深入多精但一定要每个都去做一点，这样得分的效率会比较高；另外就是多人分工合作的话一定要用好git，及时commit和merge🤧不然真的会死掉

摆烂人的忠告是要抱大腿😽

第一个lab的tai-e框架文档不是很齐全，需要看代码里的注释来熟悉接口

第一个lab需要比较熟悉Java的语法，应该有部分测试点会涉及Java的一些不太常用的语法

助教注：今年ban掉了Java的一些不常见语法，比如static initializer block

一些不透露头像的前辈的忠告 (from 23 fall)