

**2025年春季学期  
计算学部《软件工程》课程**

**实验报告**

**Lab3 代码评审与单元测试**

|  |  |  |
| --- | --- | --- |
| **姓名** | **学号** | **联系方式** |
|  |  | Email/手机号码 |

**目 录**

[1 实验要求 1](#_Toc168176490)

[2 在IDE中配置代码审查与分析工具 1](#_Toc168176491)

[2.1 Checkstyle 1](#_Toc168176492)

[2.2 SpotBugs 1](#_Toc168176493)

[2.3 EclEmma 1](#_Toc168176494)

[2.4 Junit 1](#_Toc168176495)

[3 Checkstyle所发现的代码问题清单及原因分析 1](#_Toc168176496)

[4 SpotBugs所发现的代码问题清单及原因分析 1](#_Toc168176497)

[5 针对Lab1的黑盒测试 2](#_Toc168176498)

[5.1 所选的被测函数及其需求规约 2](#_Toc168176499)

[5.2 等价类划分结果 2](#_Toc168176500)

[5.3 测试用例设计 2](#_Toc168176501)

[5.4 JUnit测试代码 3](#_Toc168176502)

[5.5 JUnit单元测试结果 3](#_Toc168176503)

[5.6 未通过测试的原因分析及代码修改 4](#_Toc168176504)

[6 针对Lab1的白盒测试 5](#_Toc168176505)

[6.1 所选的被测函数 5](#_Toc168176506)

[6.2 程序流程图 6](#_Toc168176507)

[6.3 控制流图 6](#_Toc168176508)

[6.4 圈复杂度计算与基本路径识别 6](#_Toc168176509)

[6.5 测试用例设计 6](#_Toc168176510)

[6.6 JUnit测试代码 7](#_Toc168176511)

[6.7 JUnit单元测试结果 7](#_Toc168176512)

[6.8 代码覆盖度分析 8](#_Toc168176513)

[6.9 未通过测试的原因分析及代码修改 8](#_Toc168176514)

[7 计划与实际进度 9](#_Toc168176515)

[8 小结 9](#_Toc168176516)

[文档全部完成之后，请更新上述区域]

# 实验要求

简要复述实验手册中要求达到的实验目标与过程。

# 在IDE中配置代码审查与分析工具

简要描述在IDE中安装和配置Checkstyle、SpotBugs、EclEmma、JUnit等插件或IDE自带插件的过程。

## Checkstyle

可以利用插件中包含的基本配置（Sun 或 Google ），也可以自定义配置（可选）。

## SpotBugs

## EclEmma

## Junit

# Checkstyle所发现的代码问题清单及原因分析

针对同种类型的问题，只需要列出一个典型代表即可。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 问题描述 | 类型 | 所在代码行号 | 修改策略 |
|  |  |  |  |  |
|  |  |  |  |  |

# SpotBugs所发现的代码问题清单及原因分析

针对同种类型的问题，只需要列出一个典型代表即可。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 优先级 | 问题描述 | 违反的规则集 | 所在代码行号 | 修改策略 |
|  |  |  |  |  |
|  |  |  |  |  |

# 针对Lab1的黑盒测试

## 所选的被测函数及其需求规约

只需要给出名字即可，无需给出代码。

给出该函数所应满足的需求规约 (函数功能、输入描述、输出描述)。

## 等价类划分结果

请根据自己的情况扩展该表格，给各个等价类唯一的编号。

可根据需要，增加下表的行数。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 约束条件说明 | 有效等价类及其编号 | | 无效等价类及其编号 | |
|  |  | (1) |  | (4) |
|  |  | (2) |  | (5) |
|  |  | (3) |  | (6) |

## 测试用例设计

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例编号 | 输入 | 期望输出 | 所覆盖的等价类编号 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Junit单元测试代码

针对5.3中的每一个测试用例，把其测试代码粘贴如下，代码必须是完整的。

|  |  |
| --- | --- |
| 测试用例  编号 | JUnit测试代码 |
|  |  |
|  |  |
|  |  |
|  |  |

## JUnit单元测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例编号 | 期望输出 | 实际输出 | 是否通过测试，请给出屏幕截图 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## 未通过测试的原因分析及代码修改

请简要分析自己的Lab1代码为何未通过5.5节表格中某些测试用例的原因，并通过修改代码消除此类不符合需求的Bug，必要时给出修改后的代码。

若5.5节表格中没有未通过的测试用例，本节可空。

注意：虽然本部分为黑盒测试，但发现错误之后仍然需要对代码进行修改。此时，需要根据测试结果对代码加以修改，目的是“满足需求”。

此外，此部分是否发现Lab1的Bug，不影响Lab1实验结果的评判。

修改代码之后，请重新填写下表，尽可能保证所有测试用例都能通过测试。

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例编号 | 期望输出字符串 | 实际输出字符串 | 是否通过测试，请给出屏幕截图 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Git操作记录

给出本地创建Lab3b分支，以及推送到Github上操作命令的截图；

给出本地合并Lab3b分支到master分支，以及推送到Github上操作命令的截图。

# 针对Lab1的白盒测试

## 所选的被测函数

注意：不能与5.1节所选函数重复。

|  |  |  |  |
| --- | --- | --- | --- |
| 被测函数的名称 | shortest\_path(g: DirectedGraph, s: str, t: str) | | |
| 功能描述 | 采用单源 Dijkstra 算法，返回有向图 *g* 中从 *s* 到 *t* 的最短距离及路径 | | |
| 被测函数的代码  （以IDE环境下的截图方式给出，确保能看清楚，并保留IDE为每行代码分配的行号，后续各部分均以此行号为准。如果一屏截取不下，可以分多屏截取，均插入右侧格中） |  | | |
| 输入参数列表 | 参数名 | 含义 | 数据类型 |
|  | 当前多项式 | 字符串 |
|  | 当前指令 | 字符串 |
| 输出参数 | 含义 | | 数据类型 |
|  | | 字符串 |
| 代码总行数 |  | | |
| 包含的循环数 |  | | |
| 包含的判定数 |  | | |

## 程序流程图

## 控制流图

## 圈复杂度计算与基本路径识别

圈复杂度为：（请给出计算过程）

* **节点数 N = 12**
* **边数 E = 15**
* **出口数 P = 1**

V(G)=E−N+2P=15−12+2=5V(G)=E-N+2P = 15-12+2 = \boxed{5}V(G)=E−N+2P=15−12+2=5​

| **基本路径编号** | **关键行号序列** | **说明 / 覆盖的判定分支** |
| --- | --- | --- |
| **P1** | 11 ➜ 12 ➜ 13(T) ➜ 20(F) ➜ 21 ➜ 22(loop F) ➜ 23 | 起点弹出即 v==t，成功回溯路径并返回 |
| **P2** | 11 ➜ 12 ➜ 13(F) ➜ 14(T → continue) ↺ 11 … pq 为空 ➜ 20(T) (不可达返回) | 触发 d != dist[v] 分支，最终 unreachable |
| **P3** | 11 ➜ 12 ➜ 13(F) ➜ 14(F) ➜ 15–19 (一次松弛) ↺ 11 ➜ 12 ➜ 13(T) ➜ 20(F) ➜ 21–22 ➜ 23 | 正常松弛后再次出队即命中目标 |
| **P4** | 11 ➜ 12 ➜ 13(F) ➜ 14(F) ➜ 15–17(F) (松弛失败) ↺ 15(for 结束) ↺ 11 … pq 为空 ➜ 20(T) | for-loop 未更新任何边，导致不可达 |
| **P5** | 11 ➜ 12 ➜ 13(T) ➜ 20(F) ➜ 21 ➜ 23 | 特殊情形：s == t（while 22 不进入循环） |

注意：各基本路径要使用6.1节表格里给出的行号。

## 测试用例设计

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例编号 | 输入数据 | 期望的输出 | 所覆盖的基本路径编号 |
|  | A → A（起点与终点相同） | |  | | --- | |  |  |  | | --- | | dist = 0 path = ["A"] | | P1 |
|  | A → F（默认图） | |  | | --- | |  |  |  | | --- | | dist = 3 path = ["A","B","C","F"] | | P3 |
|  | |  | | --- | |  |  |  | | --- | | A → E（默认图） | | dist = 2 path = ["A","D","E"] 或 ["A","B","E"] | P3 |
|  | A → Y 并 向图添加孤立边 X → Y | dist =inf path = [] | P4 |
|  | A → F 并 添加劣路径 A → X → E | dist = 3 path = ["A","B","C","F"] | P2 |

## JUnit测试代码

针对6.5中的每一个用例，把其测试代码粘贴如下，代码必须是完整的。

|  |  |
| --- | --- |
| 测试用例编号 | jUnit测试代码 |
|  |  |
|  |  |
|  |  |
|  |  |

## JUnit单元测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例编号 | 期望输出 | 实际输出 | 是否通过测试，请给出屏幕截图 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## 代码覆盖度分析

给出EclEmma或IDE自带工具的代码覆盖度分析报告的截图。

## 未通过测试的原因分析及代码修改

分析自己的Lab1代码为何未通过6.7节表格中某些测试用例的原因，并通过修改代码消除此类BUG。必要时给出修改后的代码。

若6.7节表格中没有未通过的测试用例，本节可空。

修改代码之后，请重新填写下表，保证所有测试用例都能通过测试。

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例编号 | 期望输出 | 实际输出 | 是否通过测试，请给出屏幕截图 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Git操作记录

给出本地创建Lab3w分支，以及推送到Github上操作命令的截图；

给出本地合并Lab3w分支到master分支，以及推送到Github上操作命令的截图。

# 计划与实际进度

|  |  |  |  |
| --- | --- | --- | --- |
| 任务名称 | 计划时间长度（分钟） | 实际耗费时间（分钟） | 提前或延期的原因分析 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 小结