哈尔滨工业大学 计算学部

2024 年秋季学期《开源软件开发实践》

Lab 2：开源软件开发协作流程

姓名	学号	联系方式
李宸	202211654	lichen2022@stu.hit.edu.cn

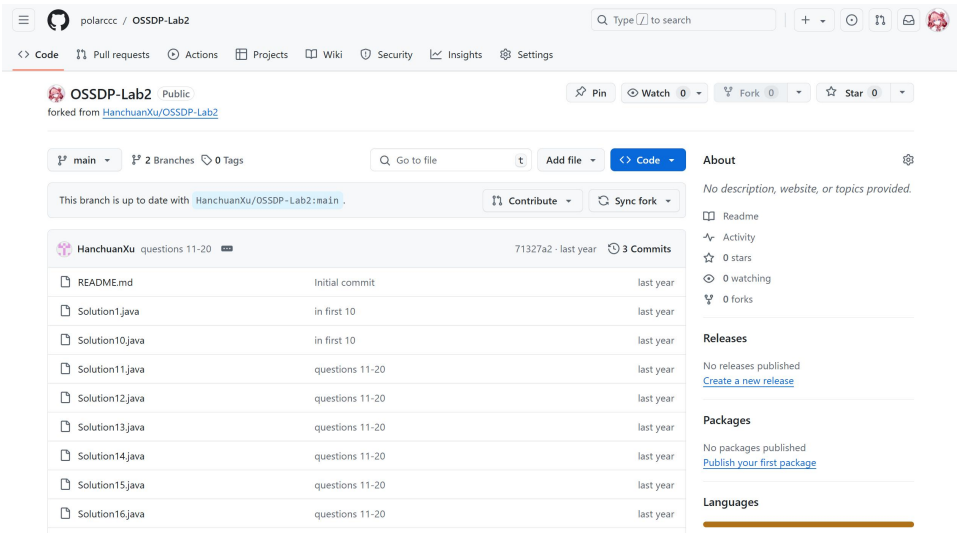
目 录

1 实验要求

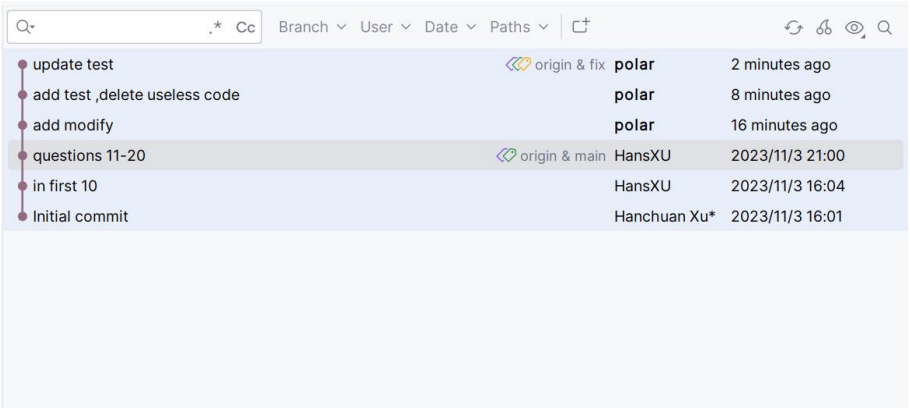
- 了解和掌握基于代码托管平台的开源软件协作开发过程
- 掌握基于 github 的软件项目协作开发命令和方法
- 熟悉几个 github 中常用开源软件开发工具。

2 实验内容 1 发送 pull request

2.1 fork 项目



2.2 git 操作命令



常见的 git 命令已经很熟悉了，这里用 GUI 界面完成的，更方便一些

2.3 代码修改

```
import java.util.*;

/*
 * @Description
 *
 * 比较版本号
 * 给你两个版本号 version1 和 version2 ，请你比较它们。
 * 版本号由一个或多个修订号组成，各修订号由一个 '.' 连接。每个修订号由 多位数字 组成，可能包含 前导零 。每个版本号至少包含一个字符。修订号从左到右编号，下标从 0 开始，最左边的修订号下标为 0 ，下一个修订号下标为 1 ，以此类推。例如，2.5.33 和 0.1 都是有效的版本号。
 * 比较版本号时，请按从左到右的顺序依次比较它们的修订号。比较修订号时，只需比较 忽略任何前导零后的整数值 。也就是说，修订号 1 和修订号 001 相等 。如果版本号没有指定某个下标处的修订号，则该修订号视为 0 。例如，版本 1.0 小于版本 1.1 ，因为它们下标为 0 的修订号相同，而下标为 1 的修订号分别为 0 和 1 ，0 < 1 。
 * 返回规则如下：
 * 如果 version1 > version2 返回 1，
 *  如果 version1 < version2 返回 -1，
 * 除此之外返回 0。
 *
 * 示例 1：
 * 输入：version1 = "1.01", version2 = "1.001"
 * 输出：0
 * 解释：忽略前导零，"01" 和 "001" 都表示相同的整数 "1"
 * 示例 2：
 * 输入：version1 = "1.0", version2 = "1.0.0"
 * 输出：0
 * 解释：version1 没有指定下标为 2 的修订号，即视为 "0"
 * 示例 3：
 * 输入：version1 = "0.1", version2 = "1.1"
 * 输出：-1
 * 解释：version1 中下标为 0 的修订号是 "0"，version2 中下标为 0 的修订号是 "1" 。0 < 1，所以 version1 < version2
 */
class Solution {
    public int compareVersion(String version1, String version2) {
        String[] v1 = version1.split("\\.");
        String[] v2 = version2.split("\\.");
        for (int i = 0; i < v1.length || i < v2.length; ++i) {
            int x = 0, y = 0;
            if (i < v1.length) {
                x = Integer.parseInt(v1[i]);
```

```
                }
                if (i < v2.length) {
                    y = Integer.parseInt(v2[i]);
                }
                if (x > y) {
                    return 1;
                }
                if (x < y) {
                    return -1;
                }
            }
            return 0;
        }
}
```

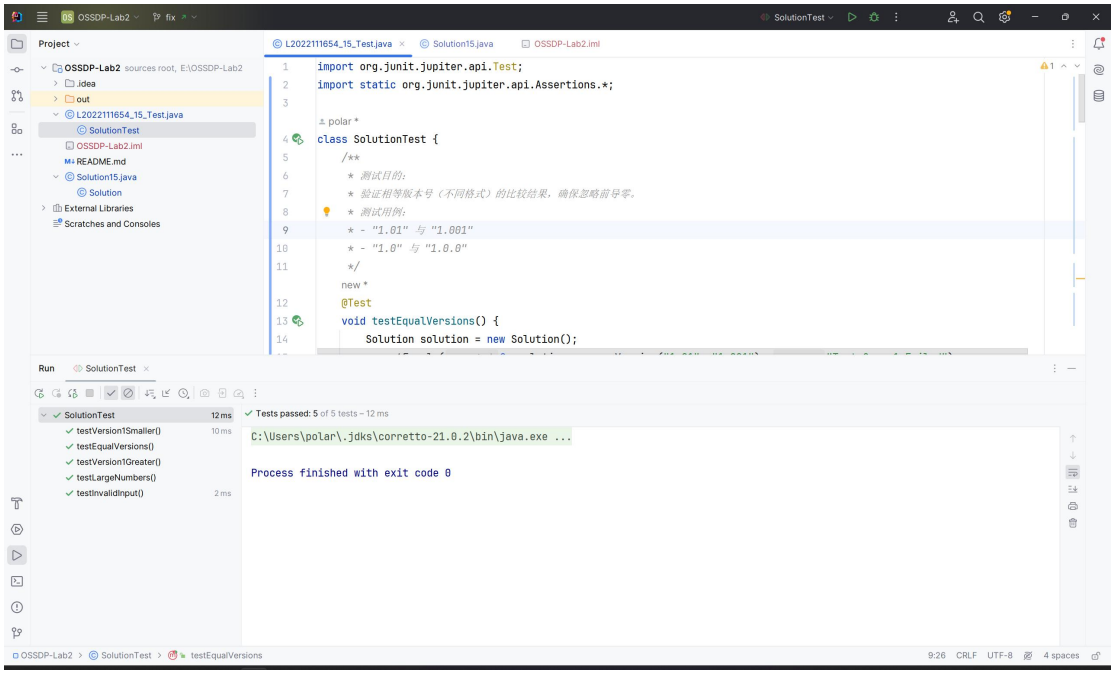
2.4 测试类代码

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class SolutionTest {
    /**
     * 测试目的：
     * 验证相等版本号（不同格式）的比较结果，确保忽略前导零。
     * 测试用例：
     * - "1.01" 与 "1.001"
     * - "1.0" 与 "1.0.0"
     */
    @Test
    void testEqualVersions() {
        Solution solution = new Solution();
        assertEquals(0, solution.compareVersion("1.01", "1.001"), "Test Case 1 Failed");
        assertEquals(0, solution.compareVersion("1.0", "1.0.0"), "Test Case 2 Failed");
        assertEquals(0, solution.compareVersion("0", "0.0.0"), "Test Case 3 Failed");
    }

    /**
     * 测试目的：
     * 验证当 version1 > version2 时的比较结果。
     * 测试用例：
     * - "1.2.3" 与 "1.2.2"
     * - "2.0" 与 "1.9.9"
     */
```

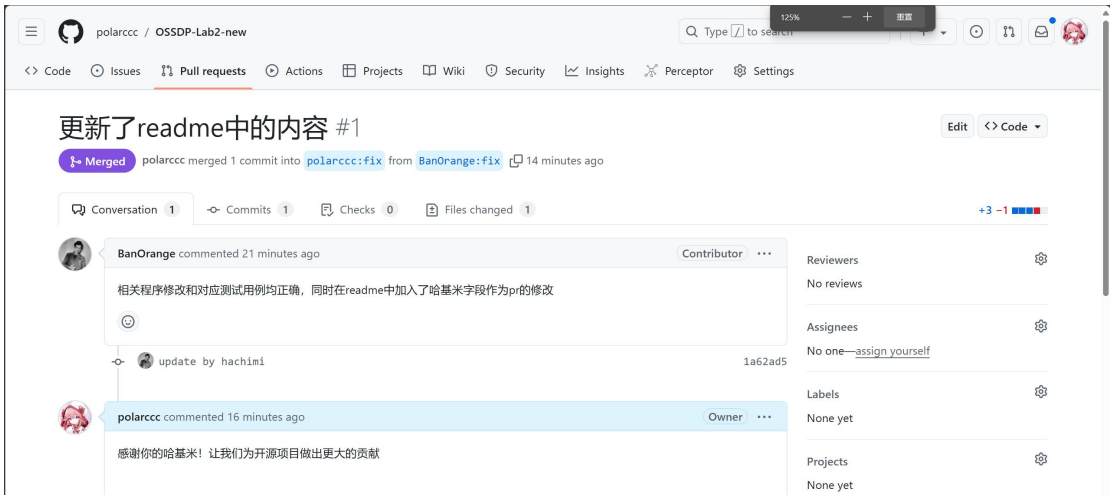
```
    @Test
    void testVersion1Greater() {
        Solution solution = new Solution();
        assertEquals(1, solution.compareVersion("1.2.3", "1.2.2"), "Test Case 4 Failed");
        assertEquals(1, solution.compareVersion("2.0", "1.9.9"), "Test Case 5 Failed");
        assertEquals(1, solution.compareVersion("1.0.1", "1"), "Test Case 6 Failed");
    }

    /**
     * 测试目的：
     * 验证当 version1 < version2 时的比较结果。
     * 测试用例：
     * - "0.1" 与 "1.1"
     * - "1.0.0" 与 "1.0.1"
     */
    @Test
    void testVersion1Smaller() {
        Solution solution = new Solution();
        assertEquals(-1, solution.compareVersion("0.1", "1.1"), "Test Case 7 Failed");
        assertEquals(-1, solution.compareVersion("1.0.0", "1.0.1"), "Test Case 8 Failed");
        assertEquals(-1, solution.compareVersion("1.0", "1.0.1.1"), "Test Case 9 Failed");
    }

    /**
     * 测试目的：
     * 验证边界情况，包括空字符串、null 和无效输入的处理。
     * 测试用例：
     * - "" 与 "1.0"
     * - null 与 "1.0"
     * - 非数字版本号 "a.b" 与 "1.0"
     */
    @Test
    void testInvalidInput() {
        Solution solution = new Solution();
        // 空字符串
        assertThrows(NumberFormatException.class, () -> solution.compareVersion("",
"1.0"), "Test Case 10 Failed");
        // null 值
        assertThrows(NullPointerException.class, () -> solution.compareVersion(null,
"1.0"), "Test Case 11 Failed");
        // 非数字字符
        assertThrows(NumberFormatException.class, () -> solution.compareVersion("a.b",
"1.0"), "Test Case 12 Failed");
    }
```

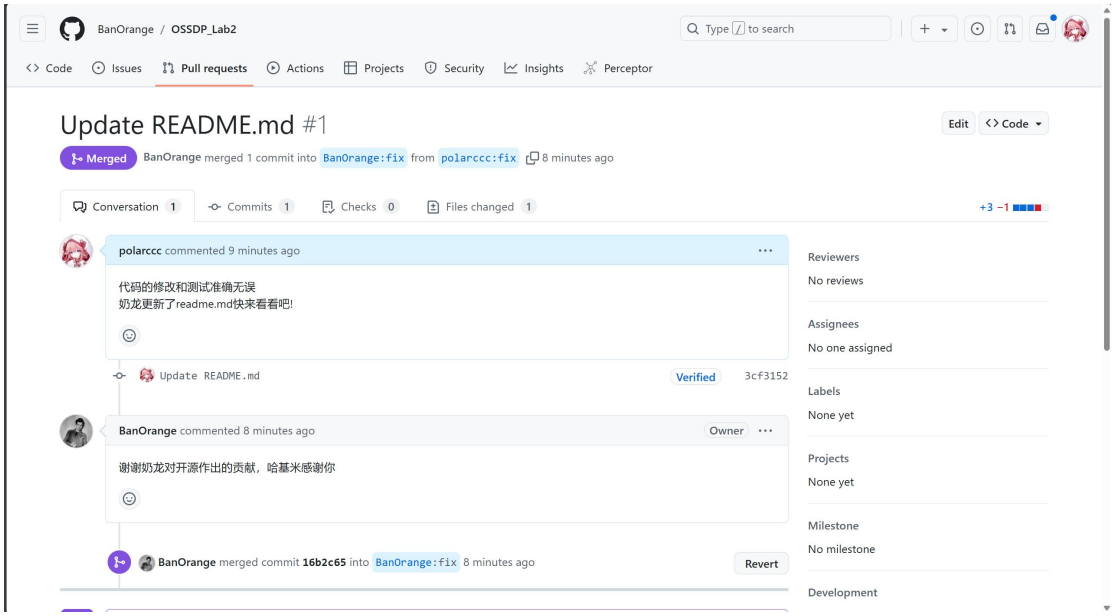
```
        /**
         * 测试目的：
         * 验证处理极大值版本号的能力。
         * 测试用例：
         * - "2147483647.0.0" 与 "2147483646.999.999"
         */
        @Test
        void testLargeNumbers() {
            Solution solution = new Solution();
            assertEquals(1, solution.compareVersion("2147483647.0.0", "2147483646.999.999"), "Test Case 13 Failed");
            assertEquals(-1, solution.compareVersion("1.0.0", "2147483647.0.0"), "Test Case 14 Failed");
        }
    }
```

2.5 测试通过截图



3 实验内容 2 接受 pull request





4 实验内容 3 github 辅助工具

4.1 熟悉 GoodFirstIssue 工具

如何让自己的开源项目被此网站收录：
1. 确保项目质量和透明度
开源项目需要有清晰的文档，例如：
README.md：描述项目的用途、安装方法和贡献指南。
CONTRIBUTING.md：详细说明如何贡献代码，包括代码风格、提交指南和测试方法。

2. 为新手标记适合的 Issue
在 GitHub 的 Issues 中创建任务，并添加 good first issue 标签。这些 Issue 应该：
是对项目重要的真实改进（非虚构或微不足道的任务）。
包含足够的上下文和详细说明。
附带指向相关文件或代码部分的链接，帮助新人快速上手。

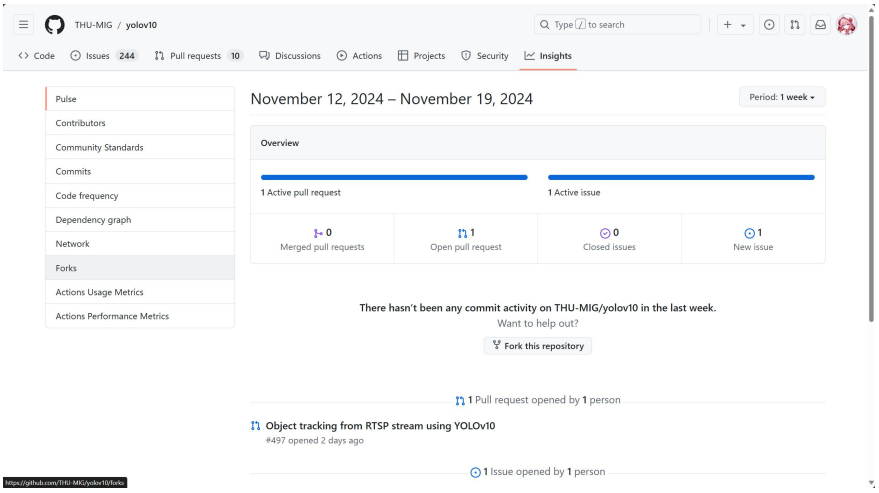
3. 使用描述性标签
除了 good first issue 标签，还可以添加：
documentation：如果 Issue 涉及文档改进。
bug：如果是修复问题。
enhancement：如果是增加新功能。 这样可以更好地吸引目标贡献者。

4. 使项目内容被 goodfirstissue.dev 自动收录
该网站自动从 GitHub 收集具有 good first issue 标签的 Issue。确保项目托管在 GitHub 上，并正确使用该标签。
网站会定期抓取公开的 Issue，无需额外提交。

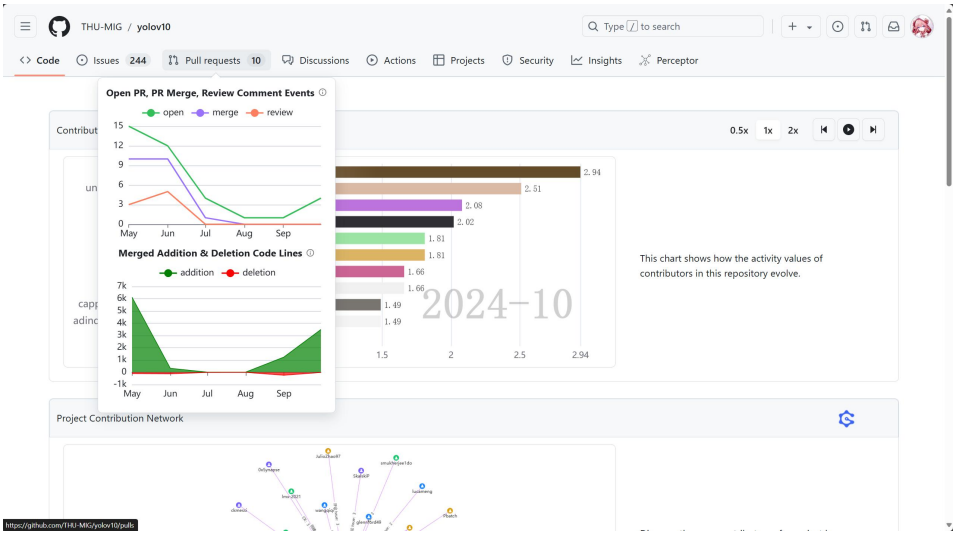
5. 增加社区参与度
在 Issue 中鼓励新手参与，例如提供指导性评论或参考链接。
对提出的问题进行快速回复，展示开放态度。

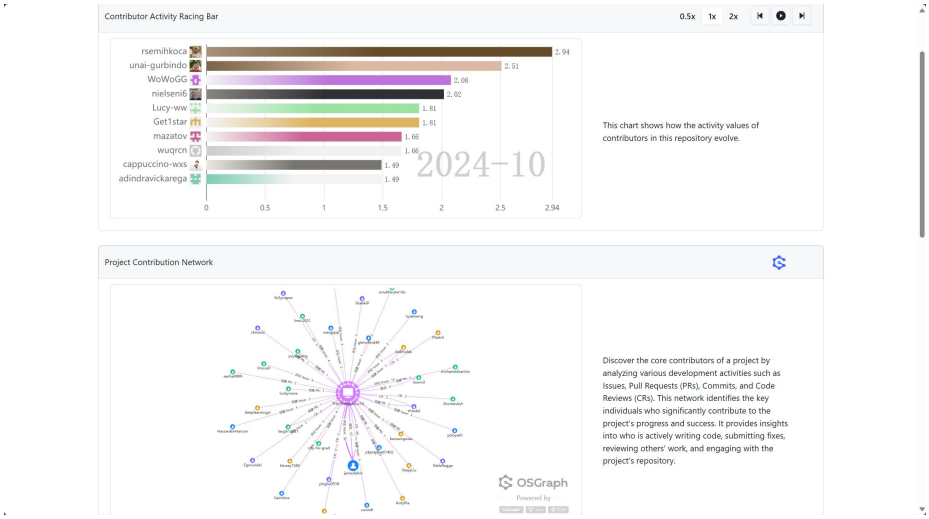
4.2 安装并使用 Hypercrx

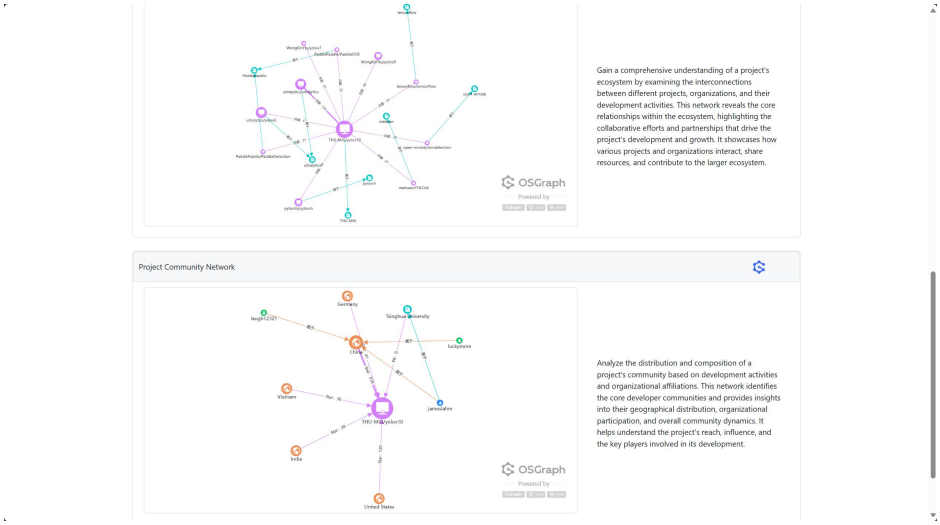
通过 https://github.com/hypertrons/hypertrons-crx 安装 Hypercrx 插件
使用插件前：


THU-MIG / yolov10
Code
Issues 244
Pull requests 10
Discussions
Actions
Projects
Security
Insights
Pulse
Contributors
Community Standards
Commits
Code frequency
Dependency graph
Network
Forks
Actions Usage Metrics
Actions Performance Metrics
November 12, 2024 – November 19, 2024
Period: 1 week
Overview
1 Active pull request
1 Active issue
0 Merged pull requests
1 Open pull request
0 Closed issues
1 New issue
There hasn't been any commit activity on THU-MIG/yolov10 in the last week.
Want to help out?
Fork this repository
1 Pull request opened by 1 person
Object tracking from RTSP stream using YOLOv10
#497 opened 2 days ago
1 Issue opened by 1 person

使用插件后：


THU-MIG / yolov10
Code
Issues 244
Pull requests 10
Discussions
Actions
Projects
Security
Insights
Perceptor
Open PR, PR Merge, Review Comment Events
open
merge
review
Merged Addition & Deletion Code Lines
addition
deletion
May
Jun
Jul
Aug
Sep
0.5x
1x
2x
2024-10
This chart shows how the activity values of contributors in this repository evolve.
Project Contribution Network


Contributor Activity Racing Bar
rsemihkoca
unai-gurbindo
WoWoGG
nielseni6
Lucy-ww
Get1star
mazatov
wuqrcn
cappuccino-wxs
adindravickarega
2024-10
This chart shows how the activity values of contributors in this repository evolve.
Project Contribution Network
OSGraph
Discover the core contributors of a project by analyzing various development activities such as Issues, Pull Requests (PRs), Commits, and Code Reviews (CRs). This network identifies the key individuals who significantly contribute to the project's progress and success. It provides insights into who is actively writing code, submitting fixes, reviewing others' work, and engaging with the project's repository.



4.3 利用 OpenLeaderboard 工具

1. 开源项目活跃度

开源项目活跃度用于衡量一个开源项目的生命力和社区参与程度，常见指标包括：

代码提交频率：项目在一定时间内的代码提交数量，反映开发者的活跃程度。

问题和讨论：GitHub Issues 和 Discussions 的数量及响应速度，反映社区参与度。

拉取请求（PR）数量及合并速度：衡量外部贡献者对项目的参与度和维护者的响应能力。

贡献者数量：项目中贡献代码、文档或其他资源的开发者总数。

发布版本更新频率：反映项目的持续开发和改进情况。

2. 影响力指标

影响力指标衡量项目在行业或社区中的受欢迎程度和影响力，主要包括：

Stars 数量：反映社区对项目的直接认可。

Forks 数量：代表开发者对项目的兴趣，表明项目可能被用作基础。

依赖性（Dependency）：其他项目或库依赖此项目的数量，越高越表明其影响力。

引用和传播：研究论文中引用项目的次数，或项目在博客、论坛、会议等中的传播情况。

3. 价值流网络

价值流网络（Value Stream Network）是一个图形化的模型，展示了开源项目在整个生态系统中的价值流动情况：

节点：代表开源项目、贡献者或用户。

边：表示贡献、依赖关系、反馈或协作。

意义：通过分析价值流网络，可以理解项目与生态系统中其他组件的关系，识别出关键项目或贡献者。

4. OpenRank 的计算原理

OpenRank 是 OpenLeaderboard 用于量化开源项目或个人影响力的评分机制，类似于 Google 的 PageRank 算法，核心在于网络分析。其计算原理包括：

网络构建
将开源社区视为一个图网络：

节点：代表项目、贡献者或组织。
边：表示项目之间的依赖、用户与项目的交互、贡献者与项目的关联等。
权重分配
根据节点之间的交互强度分配权重，例如：

高质量的依赖项目权重更高。
活跃的贡献者和维护者增加节点权重。
迭代计算
OpenRank 使用随机游走算法，在网络中迭代传播节点的权重：

节点的得分不仅取决于其直接连接的权重，还受其间接连接的影响。
每次迭代都会根据传递性调整节点得分，直到收敛。
综合调整
考虑外部因素，例如项目的影响力指标（Stars、Forks、引用等），对初始权重进行调整。

5 小结

本次实验，我先首先进行了题目代码的修改与测试，然后发送了 pull request。完成后与同学互相 fork 并发送 PR，充分交流了代码修改的意见，最后阅读并使用了实验资料提到的工具。

对于一些资料，https://goodfirstissue.dev 是一个帮助新人找到开源项目的优秀资源,它收集了标记为 good first issue 的 GitHub issue，适合新手参与。而 Hypercrx 插件提供了强大的可视化功能，能够直观地看出 GitHub 项目中 fork,pull request,merge 的柱状图，关系网等。