

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
余昊卿	2023120253	solightlyu@foxmail.com

## 目 录

1 实验要求 .....	1
2 实验内容 1 发送 pull request .....	1
2.1 fork 项目 .....	1
2.2 git 操作命令 .....	1
2.3 代码修改 .....	3
2.4 测试类代码 .....	4
2.5 测试通过截图 .....	6
3 实验内容 2 接受 pull request .....	7
4 实验内容 3 github 辅助工具 .....	8
4.1 熟悉 GoodFirstIssue 工具 .....	8
4.2 安装并使用 Hypercrx .....	8
4.3 利用 OpenLeaderboard 工具 .....	10
5 小结 .....	12

[文档全部完成之后, 请更新上述区域]

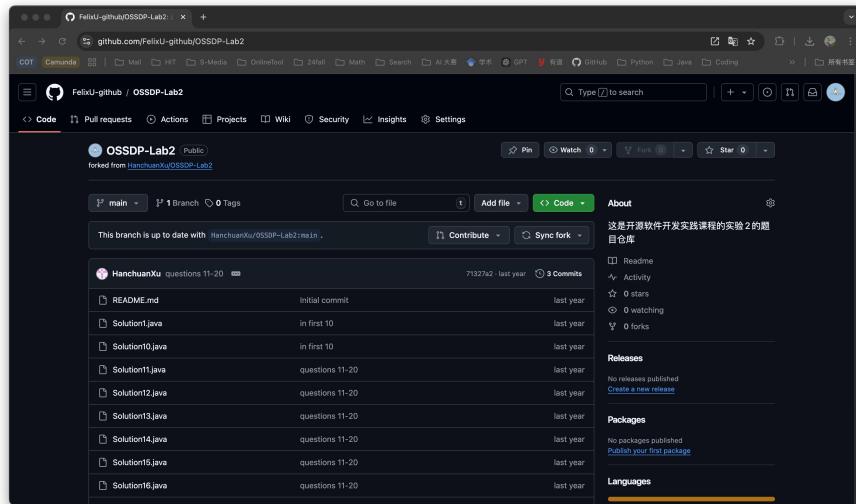
# 1 实验要求

- ① 了解和掌握基于代码托管平台的开源软件协作开发过程。
- ② 掌握基于 github 的软件项目协作开发命令和方法。
- ③ 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

fork 后的个人仓库中项目的界面截图：



### 2.2 git 操作命令

克隆指令：git clone [git@github.com:FelixU-github/OSSDP-Lab2.git](https://github.com/FelixU-github/OSSDP-Lab2.git)

```
Felix
Last login: Sun Dec  1 00:41:16 on ttys000
solightlyu@Yus-Mac ~ % cd IdeaProjects
solightlyu@Yus-Mac IdeaProjects % ls
MQdemo
MessageQueue
solightlyu@Yus-Mac IdeaProjects % git clone git@github.com:FelixU-github/OSSDP-Lab2.git
正克隆到 'OSSDP-Lab2'...
remote: Enumerating objects: 27, done.
remote: Counting objects: 100% (22/22), done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 27 (delta 1), reused 1 (delta 1), pack-reused 5 (from 1)
接收对象中: 100% (27/27), 20.04 KiB | 150.00 KiB/s, 完成.
处理 delta 中: 100% (1/1), 完成.
solightlyu@Yus-Mac IdeaProjects %
```

提交指令: git commit -m ‘描述样例 XXXX’

```

Commit
Terminal Local + v
SolutionTest.java

提交为空，但是存在尚未跟踪的文件（使用 "git add" 建立跟踪）
solighty@Yus-Mac OSSDP-Lab2 % touch .gitignore
... solighty@Yus-Mac OSSDP-Lab2 % git status
位于分支 fix
未跟踪的文件：
  (使用 "git add <文件>..." 以包含要提交的内容)
    .gitignore
      SolutionTest.java

提交为空，但是存在尚未跟踪的文件（使用 "git add" 建立跟踪）
solighty@Yus-Mac OSSDP-Lab2 % git add SolutionTest.java
solighty@Yus-Mac OSSDP-Lab2 % git commit -m 'Solution12-Test'
[fix c6b344b] Solution12-Test
 1 file changed, 65 insertions(+)
 create mode 100644 SolutionTest.java
solighty@Yus-Mac OSSDP-Lab2 % git status
位于分支 fix
① 无文件要提交，干净的工作区
git
OSSDP-Lab2 > .gitignore

```

远程推送指令: git push origin fix

```

Project
Terminal Local + v
fix

位于分支 fix
无文件要提交，干净的工作区
solighty@Yus-Mac OSSDP-Lab2 % git status
... solighty@Yus-Mac OSSDP-Lab2 % git push
致命错误：当前分支 fix 没有对应的上游分支。
为推送当前分支并建立与远程上游的链接，使用

git push --set-upstream origin fix

为了让没有追踪上游的分支自动配置，参见 'git help config' 中的 push.autoSetupRemote。

solighty@Yus-Mac OSSDP-Lab2 % git push origin fix
枚举对象中: 8, 完成。
对象计数中: 100% (8/8), 完成。
使用 10 个线程进行压缩。
压缩对象中: 100% (6/6), 完成。
写入对象中: 100% (6/6), 1.35 KB | 1.35 MiB/s, 完成。
总共 6 (差异 3)，重写 0 (差异 0)，包复用 0 (来自 0 个包)。
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
remote:
remote: Create a pull request for 'fix' on GitHub by visiting:
remote:   https://github.com/FelixU-github/OSSDP-Lab2/pull/new/fix
remote:
To github.com:FelixU-github/OSSDP-Lab2.git
 * [new branch]      fix -> fix
git
solighty@Yus-Mac OSSDP-Lab2 %

```

## 2.3 代码修改

Solution12 :

```
1.class Solution {  
2.    public String multiply(String num1, String num2) {  
3.        if (num1.equals("0") || num2.equals("0")) {  
4.            return "0";  
5.        }  
6.  
7.        String ans = "0"; //修改: 缺少分号  
8.        int m = num1.length(), n = num2.length();  
9.  
10.       for (int i = n - 1; i >= 0; i--) {  
11.           StringBuilder curr = new StringBuilder();  
12.           int add = 0;  
13.           // 补充后缀零  
14.           for (int j = n - 1; j > i; j--) {  
15.               curr.append(0);  
16.           }  
17.           int y = num2.charAt(i) - '0';  
18.           for (int j = m - 1; j >= 0; j--) {  
19.               int x = num1.charAt(j) - '0';  
20.               int product = x * y + add;  
21.               curr.append(product % 10);  
22.               add = product / 10;  
23.           }  
24.  
25.           if (add != 0) {  
26.               curr.append(add);  
27.           }  
28.           ans = addStrings(ans, curr.reverse().toString()); // 修正: 改为赋值操作, 而不是比较操作  
29.       }  
30.       return ans;  
31.   }  
32.  
33.   public String addStrings(String num1, String num2) {
```

```
34.         int i = num1.length() - 1, j = num2.length() - 1, add = 0;
35.         StringBuilder ans = new StringBuilder();
36.         while (i >= 0 || j >= 0 || add != 0) {
37.             int x = i >= 0 ? num1.charAt(i) - '0' : 0;
38.             int y = j >= 0 ? num2.charAt(j) - '0' : 0;
39.             int result = x + y + add;
40.             ans.append(result % 10);
41.             add = result / 10;
42.             i--;
43.             j--;
44.         }
45.         return ans.reverse().toString();
46.     }
47. }
```

## 2.4 测试类代码

```
1. /**
2. * 测试类: L2023120253_12_Test
3. * 用于测试 Solution 类中 multiply 方法的功能和边界情况。
4. * 等价类划分为有效输入和无效输入，重点验证方法的正确性、鲁棒性和异常处理。
5. */
6.class L2023120253_12_Test {
7.
8. /**
9. * 测试普通情况的字符串相乘。
10. */
11. @Test
12. public void testMultiply_Normal() {
13.     Solution solution = new Solution();
14.
15.     // 测试示例 1: 2 * 3 = 6
16.     assertEquals("6", solution.multiply("2", "3"));
17.
18.     // 测试示例 2: 123 * 456 = 56088
19.     assertEquals("56088", solution.multiply("123", "456"));
```

```
20.    }
21.
22.    /**
23.     * 测试边界情况：有一个数字为 0。
24.     */
25.    @Test
26.    public void testMultiply_Zero() {
27.        Solution solution = new Solution();
28.
29.        // 测试 0 * 12345 = 0
30.        assertEquals("0", solution.multiply("0", "12345"));
31.
32.        // 测试 12345 * 0 = 0
33.        assertEquals("0", solution.multiply("12345", "0"));
34.    }
35.
36.    /**
37.     * 测试数字为 1 的情况。
38.     */
39.    @Test
40.    public void testMultiply_One() {
41.        Solution solution = new Solution();
42.
43.        // 测试 1 * 9999 = 9999
44.        assertEquals("9999", solution.multiply("1", "9999"));
45.
46.        // 测试 9999 * 1 = 9999
47.        assertEquals("9999", solution.multiply("9999", "1"));
48.    }
49.
50.    /**
51.     * 测试大数字相乘。
52.     */
53.    @Test
54.    public void testMultiply_LargeNumbers() {
55.        Solution solution = new Solution();
56.
57.        // 测试大数乘法: 123456789 * 987654321
```

```

58.         assertEquals("121932631112635269", solution.multiply("123456789", "98
7654321"));
59.     }
60.
61.     /**
62.      * 测试只有一个数字为一位数，另一个为多位数的情况。
63.     */
64.     @Test
65.     public void testMultiply_OneDigitAndMultipleDigits() {
66.         Solution solution = new Solution();
67.
68.         // 测试 5 * 12345 = 61725
69.         assertEquals("61725", solution.multiply("5", "12345"));
70.
71.         // 测试 9 * 98765 = 888885
72.         assertEquals("888885", solution.multiply("9", "98765"));
73.     }
74. }

```

## 2.5 测试通过截图

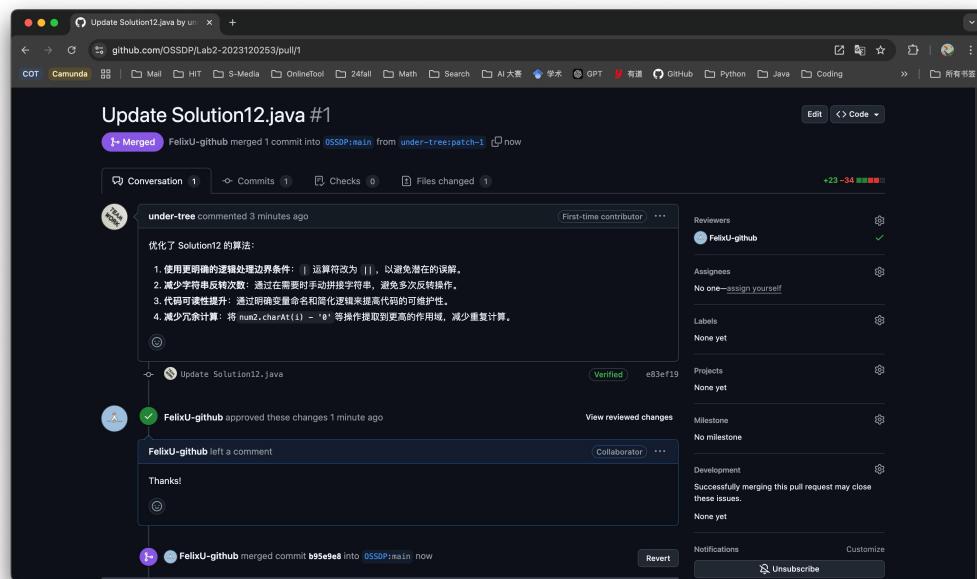
单元测试通过截图：

The screenshot shows an IDE interface with the following details:

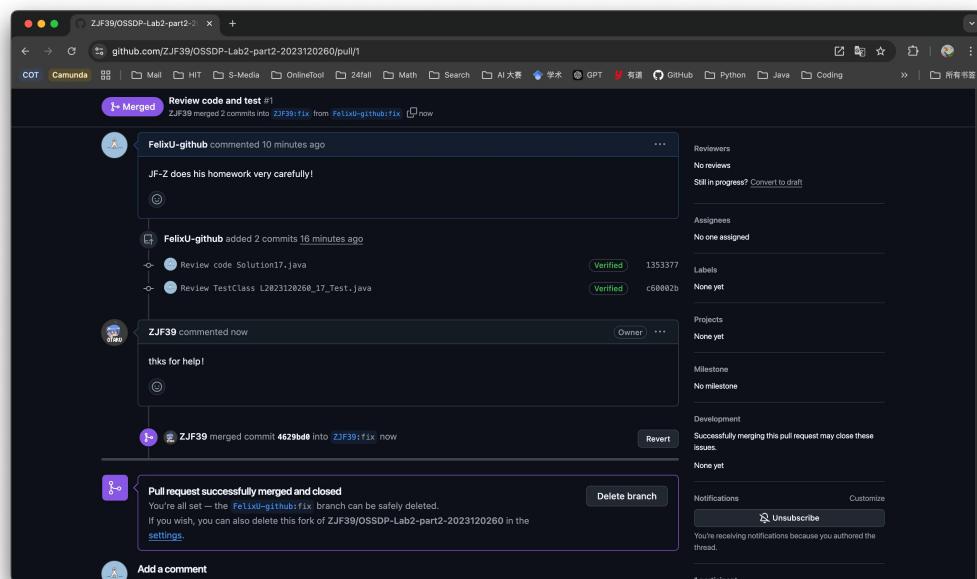
- Project View:** Shows the project structure for "Solution12". It includes a main package with "com.hit" and "Solution12" sub-packages, and a test package with "com.hit" and "L2023120253\_12\_Test" sub-packages.
- Code Editor:** The current file is "L2023120253\_12\_Test.java". The code defines a class "L2023120253\_12\_Test" with a single test method: "public void testMultiply\_Normal()".
- Run Tab:** The "Run" tab is selected, showing the output of the test run. It displays the results of five tests:
  - testMultiply\_LargeNumbers() - 11ms
  - testMultiply\_One() - 11ms
  - testMultiply\_Normal() - 11ms
  - testMultiply\_Zero() - 11ms
  - testMultiply\_OneDigitAndMultipleDigits() - 11ms
 The message "Tests passed: 5 of 5 tests – 11ms" is shown at the bottom of the run log.
- Status Bar:** At the bottom, it shows the file path "Solution12 > src > test > java > L2023120253\_12\_Test", the character count "5:1 (1736 chars, 74 line breaks)", and encoding "UTF-8".

### 3 实验内容 2 接受 pull request

接受 PR 截图：



请求 PR 截图：



## 4 实验内容 3 github 辅助工具

### 4.1 熟悉 GoodFirstIssue 工具

① 公开的仓库和社区：项目必须是公开的（Public），并且仓库的设置和问题的标签应该是清晰和一致的。项目应该有一个欢迎贡献者的社区氛围，可以在 README 或者 CONTRIBUTING 文件中说明如何参与贡献。

② 在项目的 GitHub 仓库中，至少有一个标记为 good first issue 的问题。确保问题描述清晰，提供足够的信息，以便初学者能够理解任务并进行贡献。

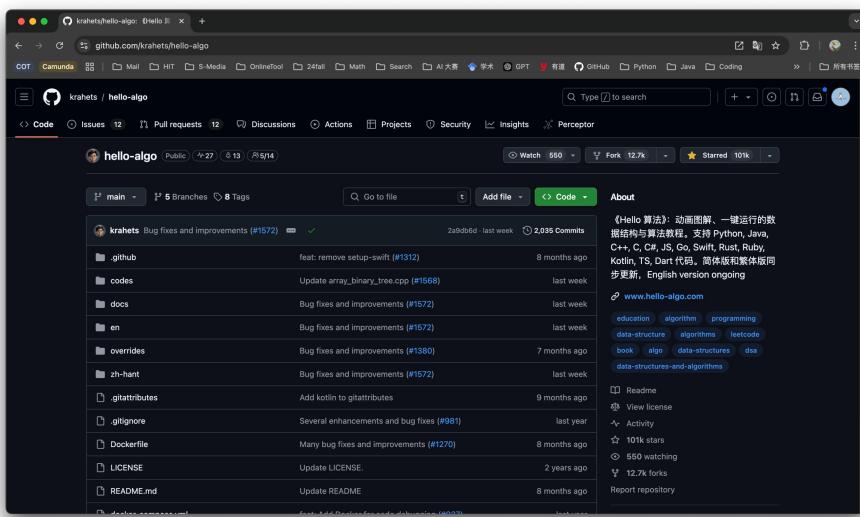
③ 项目需要具备良好的文档：在项目的仓库中应该有清晰的 README 文件，介绍项目的基本信息，包括如何开始使用、如何贡献等。如果有其他相关文档（如 CONTRIBUTING.md、CODE\_OF\_CONDUCT.md 等），也要提供，以帮助潜在的贡献者了解如何参与项目。

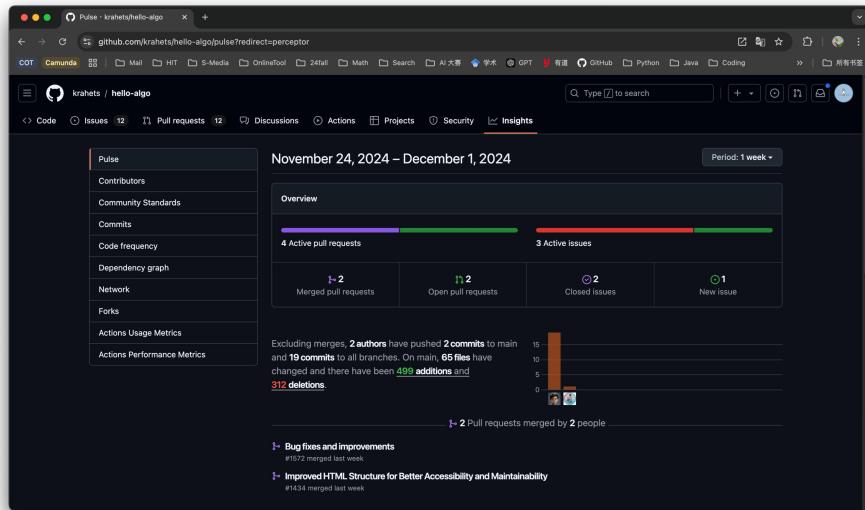
④ 积极维护活跃度：积极维护项目和问题，保持仓库的更新和活跃

⑤ 在 Good First Issue 网站提交申请。提交后，Good First Issue 网站的管理员会审核提交的项目。如果符合要求，提交的项目添加到平台上。

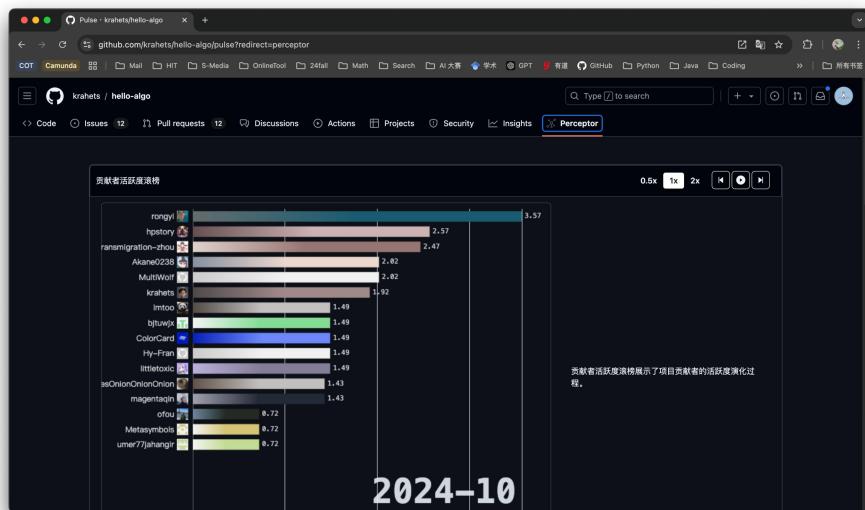
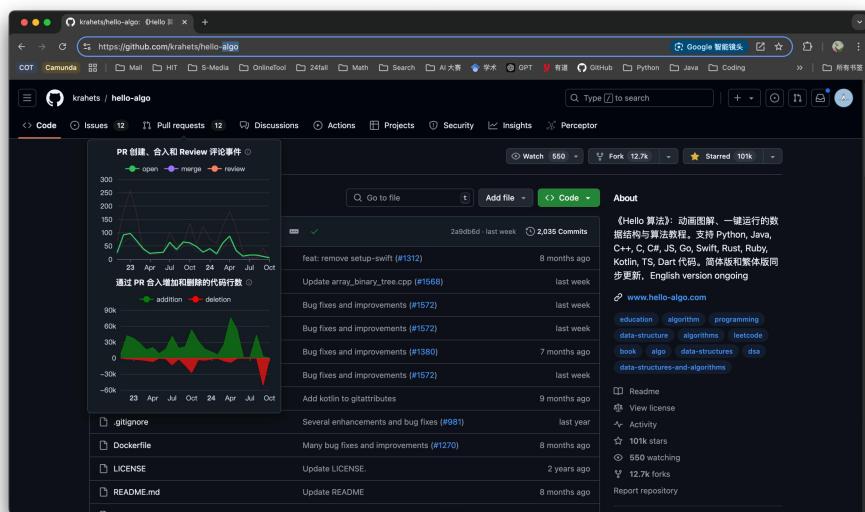
### 4.2 安装并使用 Hypercrx

安装 Hypercrx 插件前：





安装 Hypercrx 插件后：



## 4.3 利用 OpenLeaderboard 工具

利用 OpenLeaderboard 工具，可按照企业、项目、开发者，来分类查询排名。按照时间活跃度、openrank 指数来查询数据。

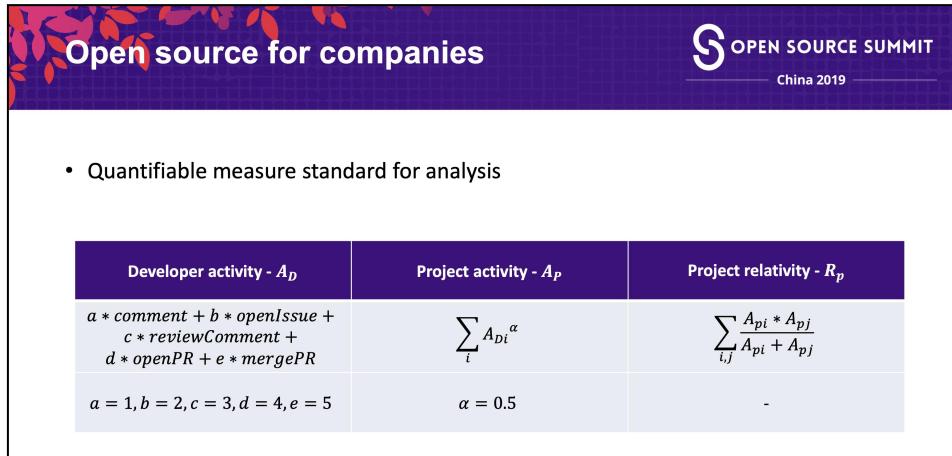
排名	企业	活跃度	Issue	评论	创建 Issue	创建 PR	合并 PR	审查 PR
1	Huawei	35231.66	5196.36	37408	16866	25812	18448	368
2	Alibaba	6836.06	681.6	6532	1271	4384	3748	2761
3	Ant group	4847.5	458.71	6379	1153	1861	1400	3572
4	Baidu	3862.95	184.78	12642	783	2405	1917	2517

还可观察详细数据可视化监控。



### 开源项目活跃度:

开源项目的活跃度指的是一个项目在开发、维护和社区参与方面的动态表现和持续投入。它反映了项目的生命力和社区互动程度。根据计算公式（图 1），按照每个 Issue 的评论数量、Issue 的数量、发起 PR 的数量、PR 的 review 评论数量和 PR 的 merge 数量来评判。此外也将 star 和 fork 数量作为一个关注指标，来辅助判断开源项目的活跃度。同时关注贡献者的数量及活跃程度，以及项目发布的更新节奏，都能进一步体现项目的持续发展能力和社区支持力度。



The screenshot shows a slide from the 'Open SOURCE SUMMIT China 2019' presentation. The title is 'Open source for companies'. The slide contains a bullet point: 'Quantifiable measure standard for analysis'. Below it is a table with three columns: 'Developer activity -  $A_D$ ', 'Project activity -  $A_P$ ', and 'Project relativity -  $R_p$ '. The first column contains the formula:  $a * comment + b * openIssue + c * reviewComment + d * openPR + e * mergePR$ . The second column contains the formula:  $\sum_i A_{Di}^\alpha$ . The third column contains the formula:  $\sum_{i,j} \frac{A_{pi} * A_{pj}}{A_{pi} + A_{pj}}$ . Below the table, there is a note:  $a = 1, b = 2, c = 3, d = 4, e = 5$  and  $\alpha = 0.5$ .

Developer activity - $A_D$	Project activity - $A_P$	Project relativity - $R_p$
$a * comment + b * openIssue + c * reviewComment + d * openPR + e * mergePR$	$\sum_i A_{Di}^\alpha$	$\sum_{i,j} \frac{A_{pi} * A_{pj}}{A_{pi} + A_{pj}}$
$a = 1, b = 2, c = 3, d = 4, e = 5$	$\alpha = 0.5$	-

(图 1)

### 影响力指标:

开源项目的影响力指标是对活跃度的一种补充，旨在评估项目在整个开源生态系统中的地位和关联性。通过构建开发者协作网络，分析开发者在不同项目中的活跃程度，确定项目之间的协作关联度。应用 PageRank 的算法，计算每个项目的协作影响力，反映其在开源社区中的重要性。可有效地解决单纯依赖活跃度统计可能导致的偏差，并且也能规避指标刷分带来的一些问题，提供了更全面的项目影响力评估。

### 价值网络流:

价值流网络是一种分析开源软件社会价值的模型，旨在从生产端到消费端全面衡量每个软件的社会价值，并反向推导出各开发者的贡献价值。通过构建包含开发者、项目、公司等多元节点的复杂网络，价值在这些节点间按照特定关系流动，最终在网络稳定时确定各节点的价值。这一模型为构建完整的开源经济生态体系奠定了基础。

### OpenRank:

OpenRank 是一种基于开发者行为数据的开源项目影响力评估方法，其核心原理是通过分析开发者在项目中的行为活跃度和跨项目的协作关系，构建项目间的协作网络，并利用加权 PageRank 算法计算每个项目的中心度（OpenRank 值）。该方法通过开发者行为映射出项目的活跃程度及其在协作网络中的关联性，无需深入分析代码内容即可评估项目的影响。

## 5 小结

本次实验实践了基于 GitHub 的开源软件协作开发流程，并通过具体的实验内容加深了我对相关工具和方法的理解。实验分为三个主要部分：发送 Pull Request、接受 Pull Request，以及熟悉 GitHub 常用辅助工具。

通过发送 Pull Request 的实验，我掌握了使用 Git 命令完成项目克隆、代码修改、提交和远程推送的完整流程，并成功进行了代码测试和问题修复。同时，通过 Pull Request，我体验了开源社区中常见的协作开发方式，感受到了代码质量和沟通的重要性。在 接受 Pull Request 的实验中，我们学习了如何评审代码更改并合并贡献者的请求，从而理解了代码审核和团队协作的重要性。

通过 GitHub 辅助工具的探索，我了解了 Good First Issue 平台、Hypercrx 插件和 OpenLeaderboard 平台产品的功能和应用场景。理解如何使用这些工具了解项目的活跃度，还学习了 OpenRank 和价值流网络分析开源项目影响力的方法。认识到，项目的影响力不仅仅依赖活跃度，还需要考虑其在开源生态系统中的地位和贡献。