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
徐耀	2022211830	2752109281@qq.com/15318051170

目 录

1 实验要求

1.了解和掌握 GitHub 平台上的开源软件协作开发流程；
2.掌握 GitHub 上的项目协作开发操作，如 fork、clone、创建分支、提交 PR 等；
3.熟悉 GitHub 常用的开源开发工具，如 goodfirstissue.dev、open-leaderboard 等

2 实验内容 1 发送 pull request

2.1 fork 项目

首先，打开实验所在仓库，点击 fork 将实验仓库 main 分支 fork 到个人账号中，如图所示。

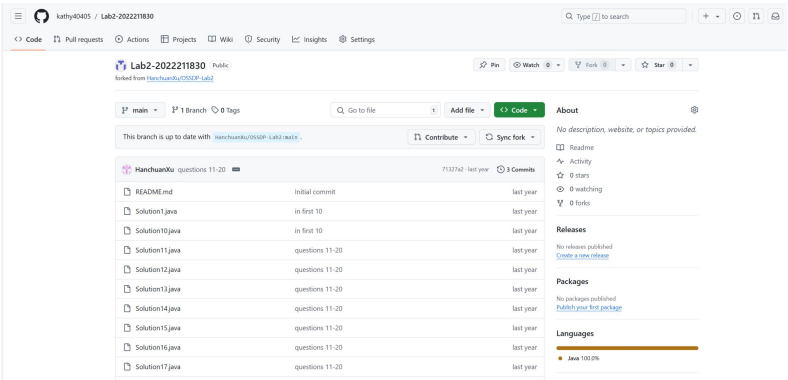

图 2-1 fork 后个人仓库截图

2.2 git 操作命令

打开 git bash，使用“git clone 仓库地址”的命令将题目仓库克隆到本地，并创建 fix 分支。

图 2-2 克隆到本地并创建 fix 分支

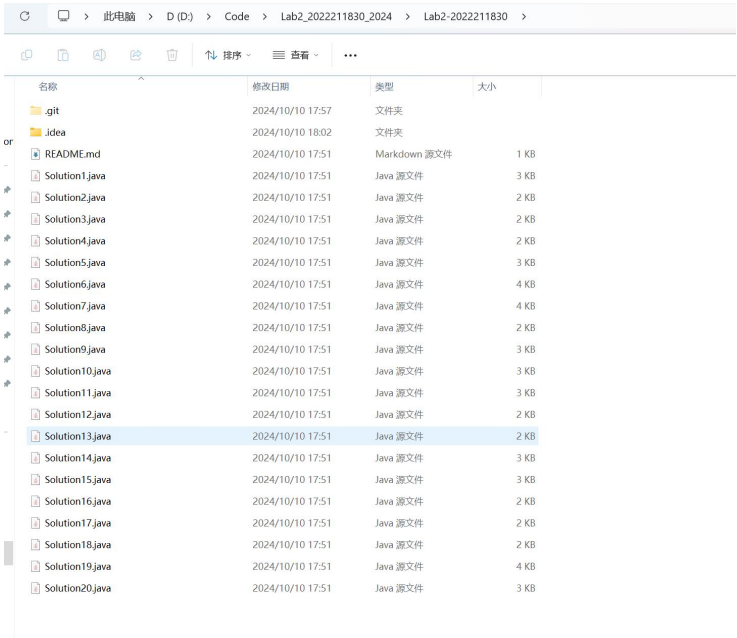

图 2-3 克隆后本地文件情况

随后在 IDEA 中打开项目文件，本人所分配到的题目为 Solution4。在 IDEA 中可以看到右下角出现 fix 分支标识，说明该文件在本地仓库的 fix 分支上。

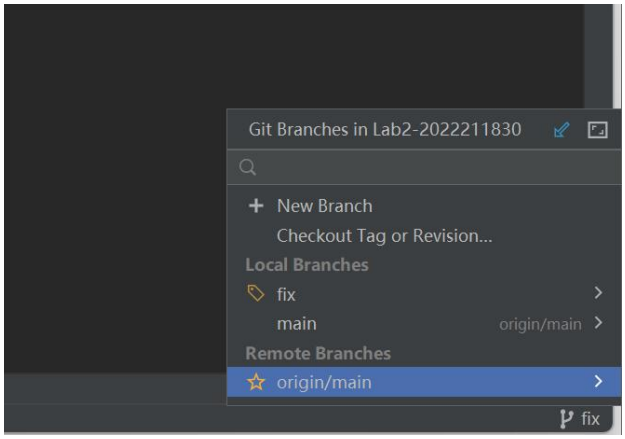

图 2-4 IDEA 右下角标识截图

修改代码并编写测试类后，保存文件至本地仓库 fix 分支上，并使用指令“git push origin fix”将分支推送到 GitHub 上（在此步骤之前已使用 ssh 链接本地仓库与远程仓库）。

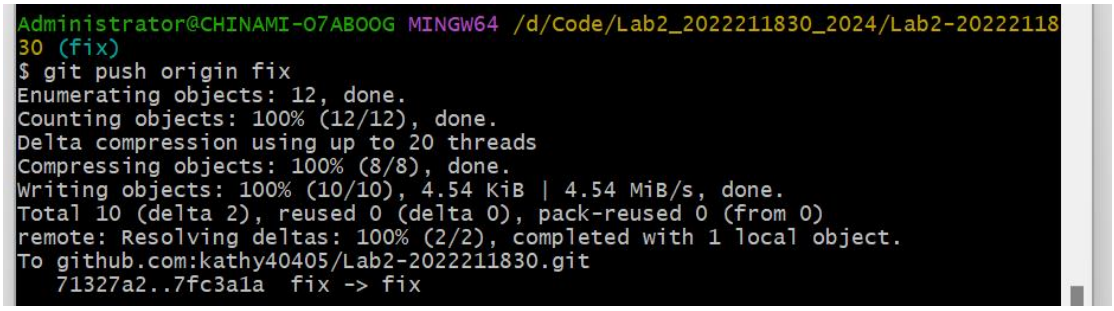

图 2-5 推送成功截图

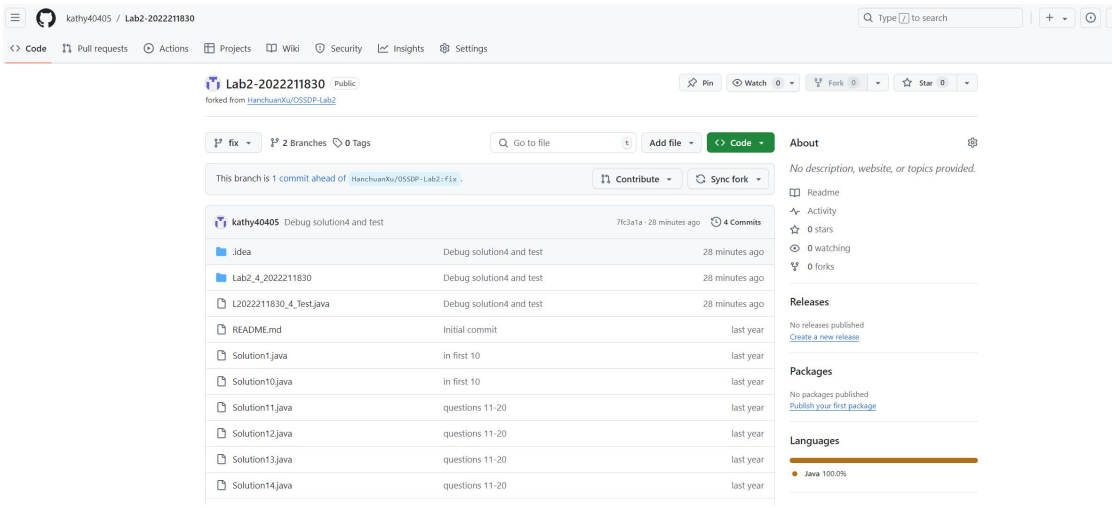

图 2-6 成功推送后 GitHub 上远程仓库截图

在 GitHub 上将 Lab2-2022211830 仓库中 fix 分支提交到实验仓库 fix 分支中，并写好 pull request。

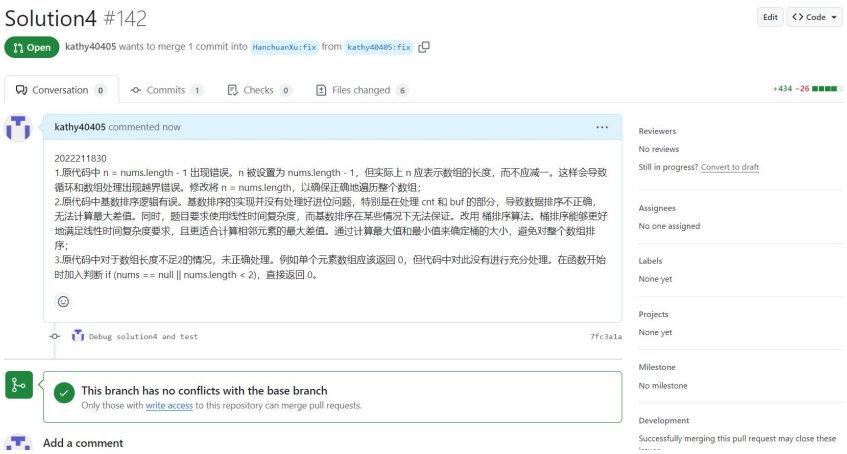

图 2-7 成功提交 PR 后截图

2.3 代码修改

修改思路：

1.原代码中 n = nums.length - 1 出现错误。n 被设置为 nums.length - 1，但实际上 n 应表示数组的长度，而不应减一。这样会导致循环和数组处理出现越界错误。修改将 n = nums.length，以确保正确地遍历整个数组；

2.原代码中基数排序逻辑有误。基数排序的实现并没有处理好进位问题，特别是在处理 cnt 和 buf 的部分，导致数据排序不正确，无法计算最大差值。同时，题目要求使用线性时间复杂度，而基数排序在某些情况下无法保证。改用 桶排序算法。桶排序能够更好地满足线性时间复杂度要求，且更适合计算相邻元素的最大差值。通过计算最大值和最小值来确定桶的大小，避免对整个数组排序；

3.原代码中对于数组长度不足 2 的情况，未正确处理。例如单个元素数组应该返回 0，但代码中对此没有进行充分处理。在函数开始时加入判断 if (nums == null || nums.length < 2)，直接返回 0。

修改后代码文件：

Solution4.java

2.4 测试通过截图

采用 IDEA 中 maven 工程中 junit 包中的测试工具，工程目录结构如下

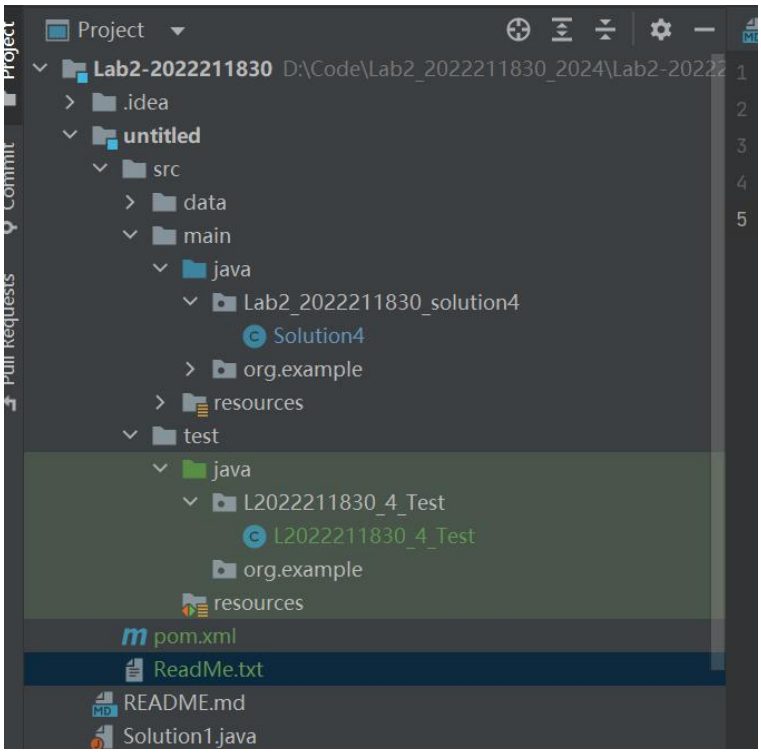

图 2-8 项目工程目录

测试用例设计原则：

1. 等价类划分原则：将输入划分为有效和无效类，确保每个类都有代表性的测试用例。

2. 边界值分析：在边界附近选择测试用例，确保极端情况得到正确处理。

3. 特殊情况：处理特殊输入，例如空数组和数组长度为 1 的情况。

测试类

1. testNormalCase：
   测试目的：测试正常情况下的最大差值
   用例：输入数组 [3, 6, 9, 1]
   期望输出：返回 3
2. testSingleElement：
   测试目的：测试空数组的情况
   用例：输入数组 []
   期望输出：返回 0
3. testEmptyArray：
   测试目的：测试空数组的情况
   用例：输入数组 []
   期望输出：返回 0
4. testNegativeNumbers：
   测试目的：测试包含负数的数组
   用例：输入数组 {-10, -5, -1, 0, 5}
   期望输出：返回 5
5. testRepeatedElements：

测试目的：测试包含重复元素的情况
用例：输入数组 [1, 1, 1, 1]
期望输出：返回 0

6. testLargeRange：
   测试目的：测试大范围数值的情况
   用例：输入数组 [1, 1000000000]
   期望输出：返回 999999999
7. testSortedArray：
   测试目的：测试有序数组的情况
   用例：输入数组 [1, 2, 3, 4, 5]
   期望输出：返回 1

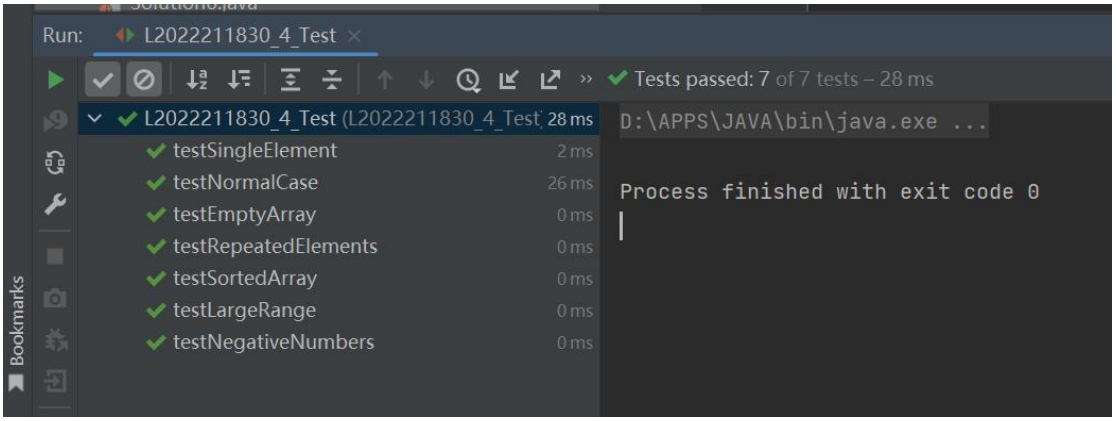

图 2-9 测试通过截图

3 实验内容 2 接受 pull request

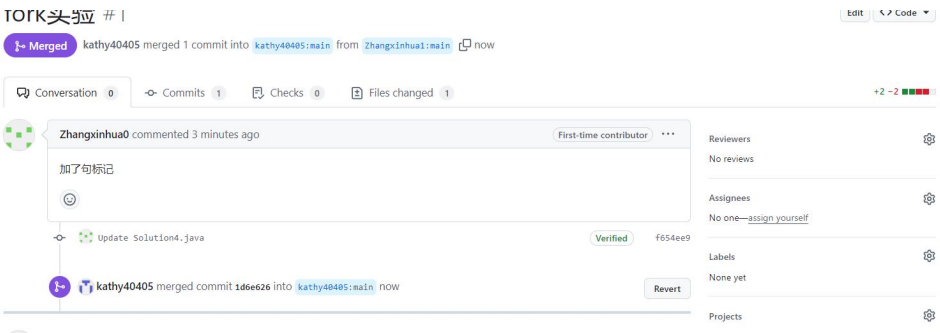

图 2-10 学号 2022211833 同学向我提交对代码文件的 PR

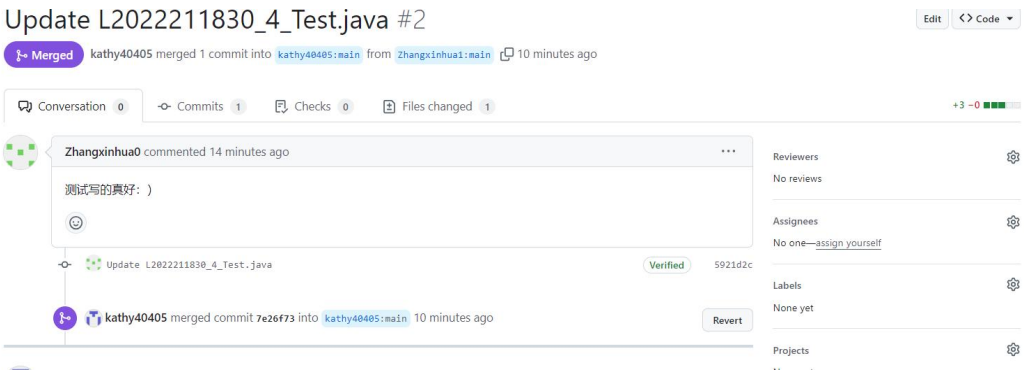

图 2-11 学号 2022211833 同学向我提交对测试用例的 PR

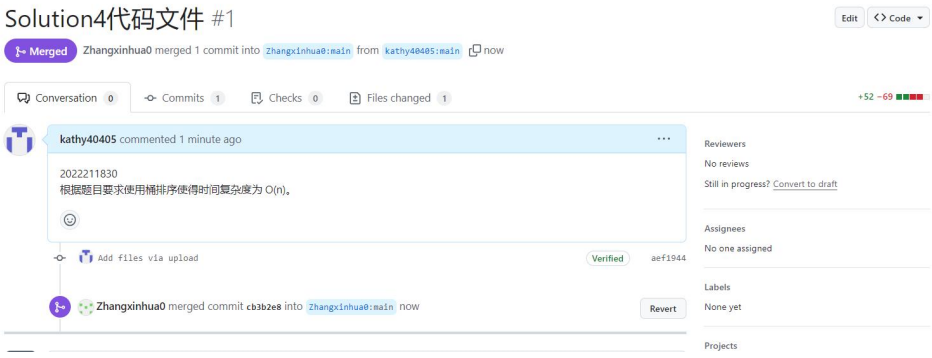

图 2-12 我向学号 2022211833 同学提交我对她代码文件的 PR

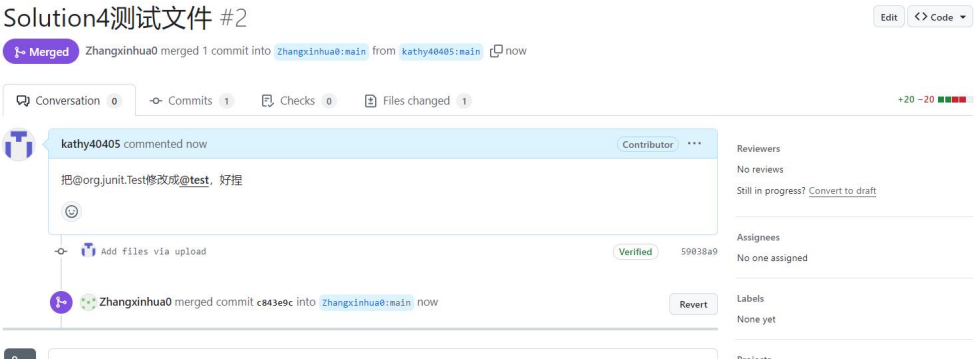

图 2-13 我向学号 2022211833 同学提交我对她测试用例的 PR

4 实验内容 3 github 辅助工具

4.1 熟悉 GoodFirstIssue 工具

（1）学习活跃度指标、影响力指标、价值流网络。通过阅读这三篇文章，我了解到衡量开源项目不仅仅是看代码的质量和活跃度，还需要从更宏

观的角度去评估其在整个开源生态系统中的影响力和价值流。活跃度指标为我们提供了一个基础的量化方法，但它过于简化且容易受到操纵。协作影响力指标则通过考虑项目间的网络关系，给出了一个更全面的视角，减少了单一指标的局限性。而价值流网络则进一步扩展了这一概念，试图将更多的实体和关系纳入考量，以更准确地反映开源项目的社会价值。

我认识到，任何单一的指标都难以全面反映一个项目的全貌。开源项目的成功不仅取决于代码的质量和社区的活跃度，还取决于其在更广泛生态系统中的作用和影响力。价值流网络模型提供了一种思考开源项目价值的新方式，它强调了项目的社会效用和经济价值，这对于理解开源项目在现实世界中的作用至关重要。

此外，这些文章也让我意识到，在设计衡量指标时，需要考虑到指标的可操作性、可扩展性和抗操纵性。一个好的指标体系应该能够引导社区向正面发展，同时能够适应不断变化的技术和社区环境。开源项目的成功不仅在于技术层面，还在于其能够创造的社会价值和经济影响。

（2）Good First Issue 是一个旨在帮助开发者找到适合初次贡献的开源项目的平台。它汇集了来自流行开源项目的简单问题，以便新手开发者可以轻松地开始他们对开源社区的贡献。以下让自己的开源项目被网站收录的步骤：

①项目要求：

首先要确保项目是开源的，并且托管在 GitHub 上。其次项目应该有一个活跃的社区和清晰的贡献指南。项目应该包含标记为“good first issue”的问题，该标签至少存在三个问题。默认情况下，此标签已存在于所有存储库中。项目至少有 10 名贡献者，并且包含一个带有项目详细设置说明的 README.md，以及一个带有新贡献者指南的 CONTRIBUTING.md。

②添加项目到 Good First Issue：

访问 https://goodfirstissue.dev 网站。点击“Add your project”按钮， 下滑找到 data/repositories.toml，在其中中添加存储库的路径（按字典顺序）。创建一个新的 pull-request。请在 PR 描述中添加指向存储库问题页面的链接。合并 pull request 后，更改将在 goodfirstissue.dev 上生效。

4.2 安装并使用 Hypercrx

访问并安装了 chrome 插件 Hypercrx，为它配置 GitHub 令牌，完成 Hypercrx 的安装配置。

图 4-1 配置 GitHub 令牌

在 GitHub 上搜索 star 在 100+的项目，随机选取一个，如下图所示。

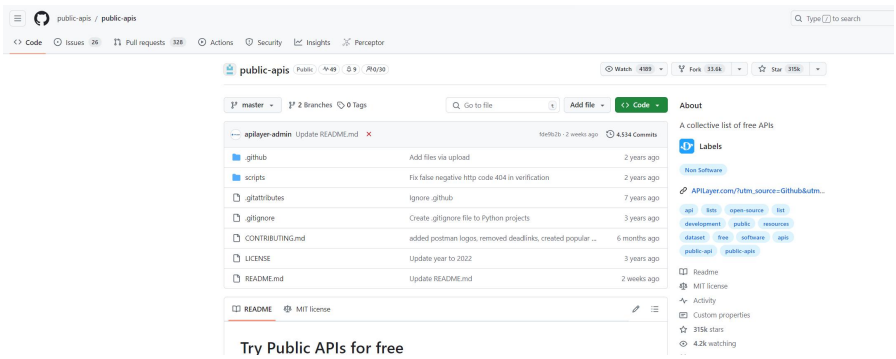

图 4-2 选取的 public-apis 项目情况

使用 Hypercrx 查看项目信息，如下图所示。

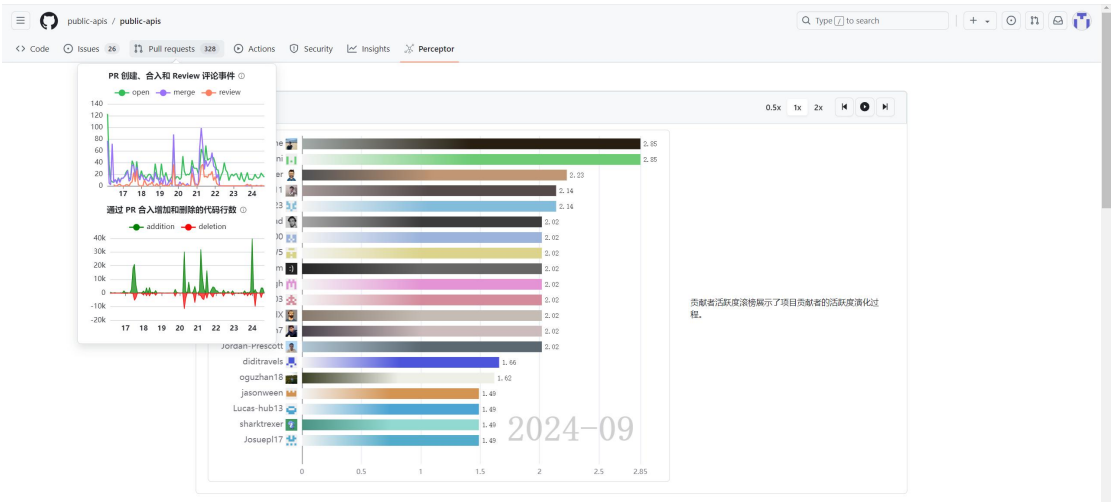

图 4-3 Hypercrx 查看项目的 PR 情况

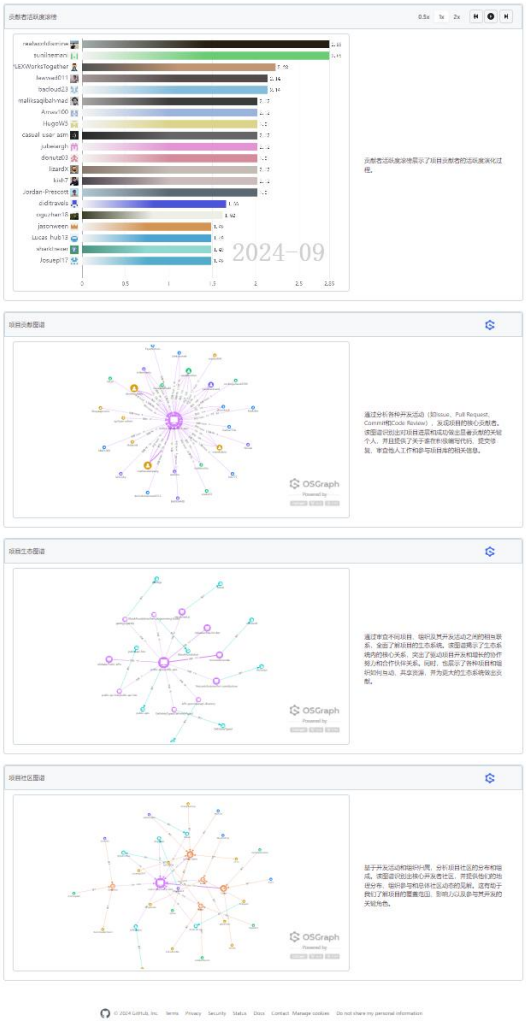

图 4-4 项目情况全览

在没有安装 Hypertrons CRX 插件之前，GitHub 项目页面只会展示项目的基本信息，如项目描述、代码、README 文件、issues 和 pull requests。此时，开发者需要手动查看项目的提交历史、issue 和 pull request 活跃度，以了解项目的健康状况和活跃度。这种方式显得较为繁琐且主观，缺乏自动化的分析工具和可视化数据。

而在安装并启用 Hypertrons CRX 插件后，GitHub 项目页面将会增加一个 "Analyze" 按钮。点击该按钮后，插件会自动分析项目的健康状况，并提供可视化的数据，如项目的活跃度、贡献者数量、提交频率、PR 和 Issue 活动等。插件不仅展示了项目的贡献者活动，还能评估项目的 影响力 和 健康度，帮助开发者更直观地了解项目的状态和长期稳定性。此外，插件为开发者提供了更多的自动化功能，极大地提高了评估开源项目的效率和准确性。

4.3 利用 OpenLeaderboard 工具

利用 OpenLeaderboard 工具了解 github 上开源项目的统计情况。

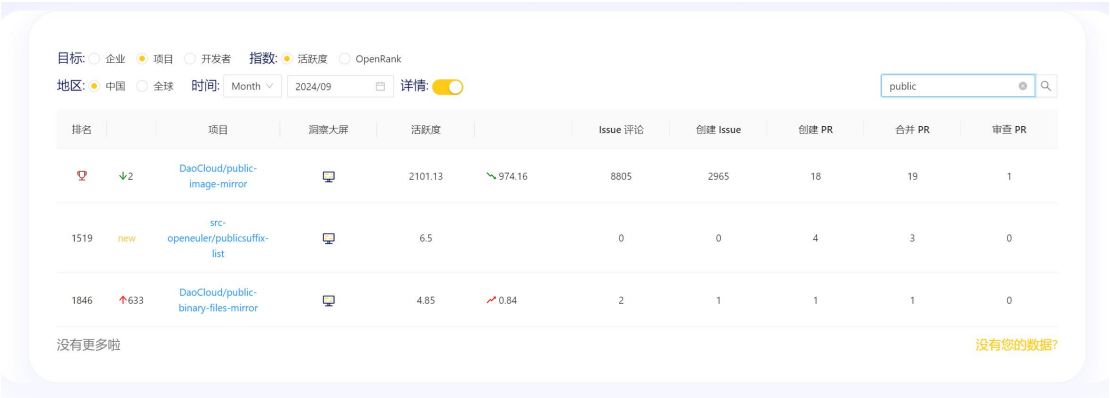

图 4-5 利用 OpenLeaderboard 工具了解 github 上开源项目

阅读所给文献后，有以下了解：

1. **活跃度指标**

   活跃度衡量的是一个开源项目的开发者参与程度。主要指标包括代码提交、合并请求、问题讨论等。更高的活跃度表明该项目有更多的开发者参与和维护，这通常意味着项目健康度较高。常用的衡量方法是对这些活动进行量化和加权，活动频繁且多样化的项目通常被认为更为成功 ；

2. **影响力指标**

   影响力指标则侧重于衡量项目在开源生态中的作用和传播程度。高影响力的项目不仅拥有大量的贡献者，还可能被其他项目依赖或与多个项目建立合作关系。例如，VSCode 和 Flutter 项目在开源生态中都表现出较高的协作影响力。这种影响力通常通过网络算法如 PageRank 来计算，通过项目之间的关联性和依赖性来评估其相对重要性 ；

3. **价值流网络**

   价值流网络模型描述了项目和开发者之间的价值传递和流动。在这个模型中，项目不仅通过开发者的贡献来增强自己的价值，还通过项目间的依赖关系和互动来获得更多的价值。开发者的活跃度、项目的依赖关系及关注度等都会影响这一价值流的形成。最终，项目和开发者在网络中的价值被不断传递和稳定，直到达成一个平衡状态 ；

4. **OpenRank 的计算原理**

   OpenRank 结合了活跃度、影响力和价值流网络的各项指标来计算项目的综合排名。它不仅考虑了项目的直接贡献度（通过活跃度指标），还分析了项目在生态中的影响力和价值流动情况。通过综合这三者，OpenRank 能够对开源项目进行全面的评价和排序，帮助用户了解项目在开源社区中的地位 。

5 小结

在本次实验中，我通过发送和接受 Pull Request (PR)的实践，体验了开源开发中的核心协作流程。首先，我在自己修改代码后，提交了 PR 进行代码审查，这一过程不仅涉及对代码进行修改，还要求编写单元测试，确保修

改后的代码满足功能需求。测试类的命名和编写符合规定的命名规则，并结合设计原则（如等价类划分），使测试更有针对性、覆盖面更广，确保代码的稳定性和准确性。

其次，我参与了审查其他同学提交的代码，并对其进行修改。这一过程帮助我提高了代码评审能力，特别是在发现潜在问题、提出修改建议、增强代码可读性等方面。通过这种互动，我能够更加深刻地理解其他同学的思路，并学习如何在团队协作中提供建设性反馈。总的来说，本次实验使我更加熟悉了 PR 的工作流，提升了编写高质量代码和进行有效团队合作的能力。

通过本次实验，我也意识到了测试的重要性以及代码质量审查中的细节问题，同时在代码审查中学会了如何提出合理的修改意见。在接受 PR 的过程中，我不仅提高了自己的编程能力，还学会了如何在团队中进行有效的沟通与协作。