

哈尔滨工业大学 计算学部

2023 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
雷文俊	2023140022	514530706@qq.com/13394013296

目 录

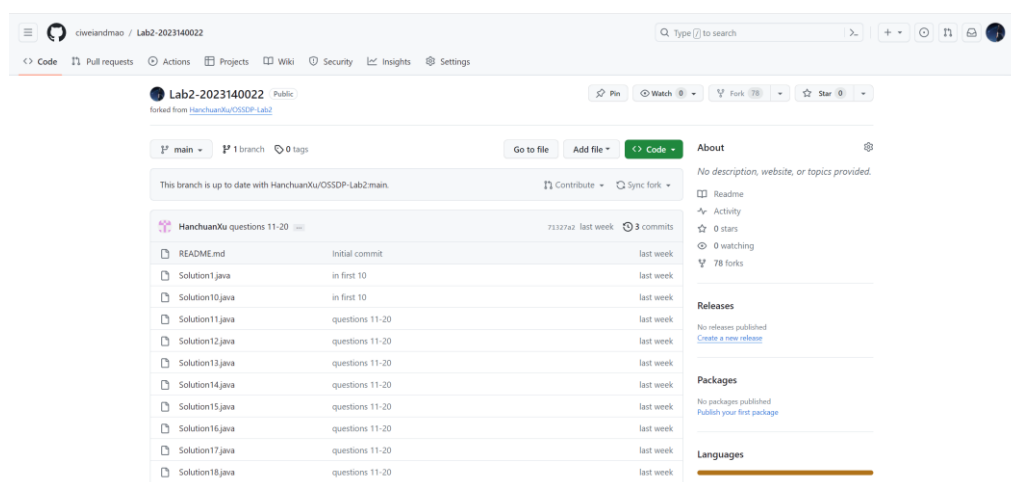
1 实验要求.....	1
2 实验内容 1 发送 pull request.....	1
2.1 fork 项目	1
2.2 git 操作命令.....	1
2.3 代码修改.....	3
2.4 测试通过截图.....	4
3 实验内容 2 接受 pull request.....	4
4 实验内容 3 github 辅助工具.....	6
4.1 熟悉 GoodFirstIssue 工具	6
4.2 安装并使用 Hypercrx.....	7
4.3 利用 OpenLeaderboard 工具	7
5 小结	9

1 实验要求

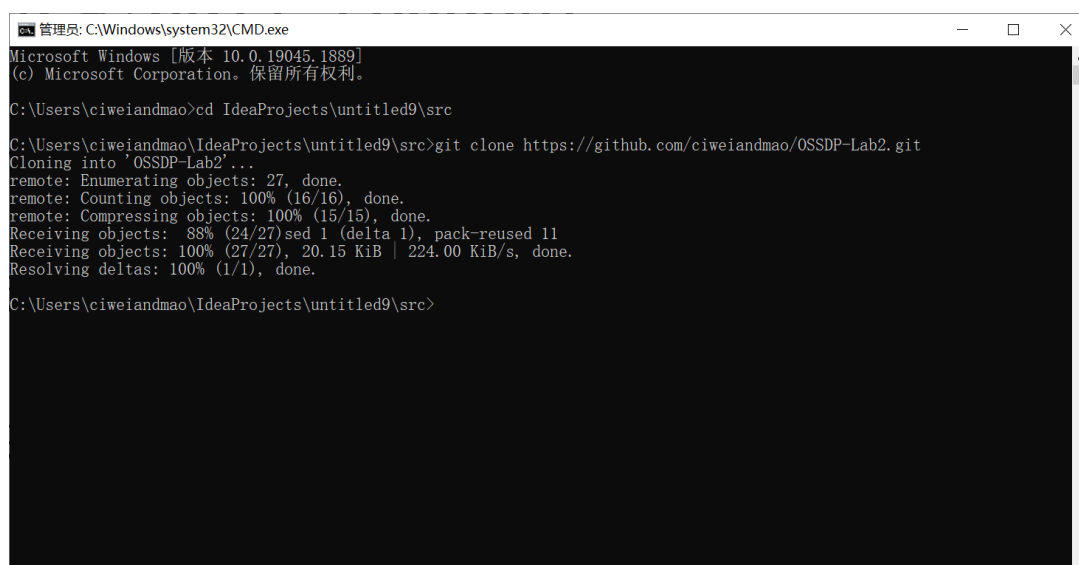
了解和掌握基于代码托管平台的开源软件协作开发过程
掌握基于 github 的软件项目协作开发命令和方法
熟悉几个 github 中常用开源软件开发工具

2 实验内容 1 发送 pull request

2.1 fork 项目



2.2 git 操作命令



```
管理员: C:\Windows\system32\CMD.exe

C:\Users\ciweindmao\IdeaProjects\untitled9>git commit
On branch fix
Your branch is up to date with 'origin/fix'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
  (commit or discard the untracked or modified content in submodules)
        modified:   src/OSSDP-Lab2 (modified content)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .idea/
        out/
        untitled9.iml

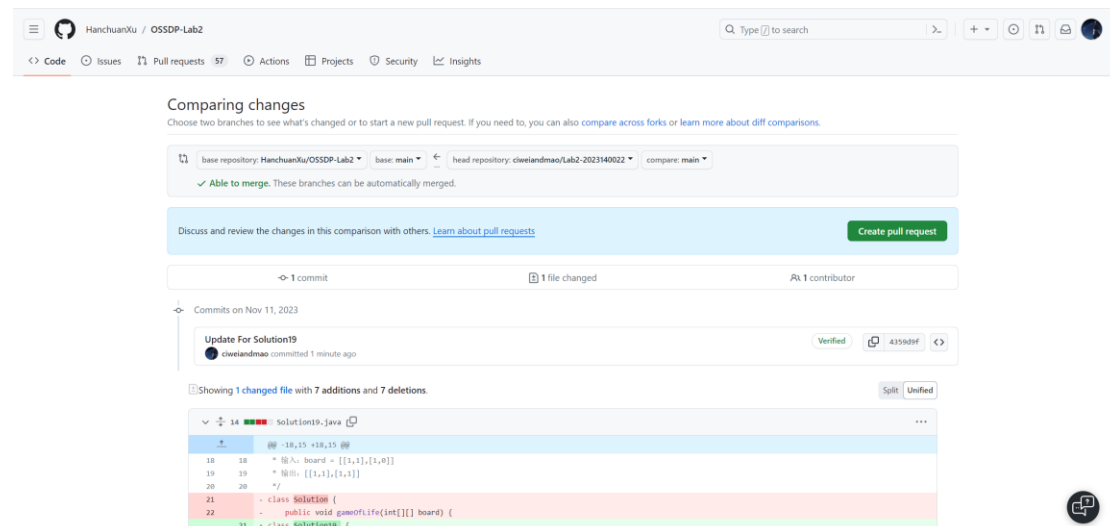
no changes added to commit (use "git add" and/or "git commit -a")
C:\Users\ciweindmao\IdeaProjects\untitled9>

管理员: C:\Windows\system32\CMD.exe

C:\Users\ciweindmao\IdeaProjects\untitled9>git push -u origin fix
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'fix' on GitHub by visiting:
remote:   https://github.com/OSSDP/Lab2-2023140022/pull/new/fix
remote:
To https://github.com/OSSDP/Lab2-2023140022.git
 * [new branch]      fix -> fix
branch 'fix' set up to track 'origin/fix'.

C:\Users\ciweindmao\IdeaProjects\untitled9>
```

发送 pull request:



Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks or learn more about diff comparisons.

base repository: HanchuanXu/OSSDP-Lab2 base: main head repository: ciweindmao/Lab2-2023140022 compare: main

✓ Able to merge. These branches can be automatically merged.

Discuss and review the changes in this comparison with others. [Learn about pull requests](#) [Create pull request](#)

1 commit 1 file changed 1 contributor

Commits on Nov 11, 2023

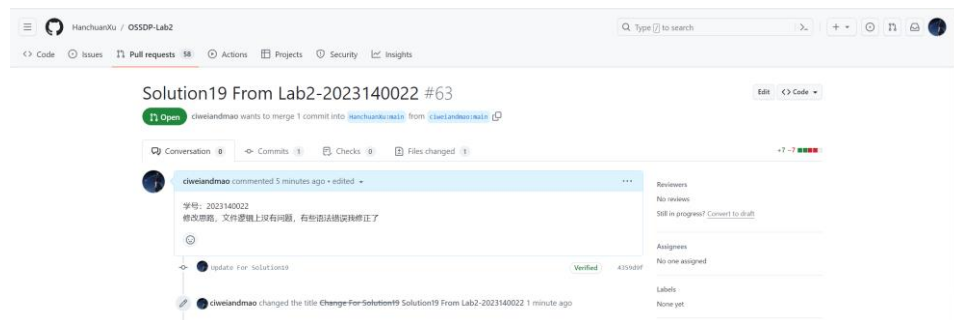
Update For Solution19

ciweindmao committed 1 minute ago

Showing 1 changed file with 7 additions and 7 deletions.

Split Unified

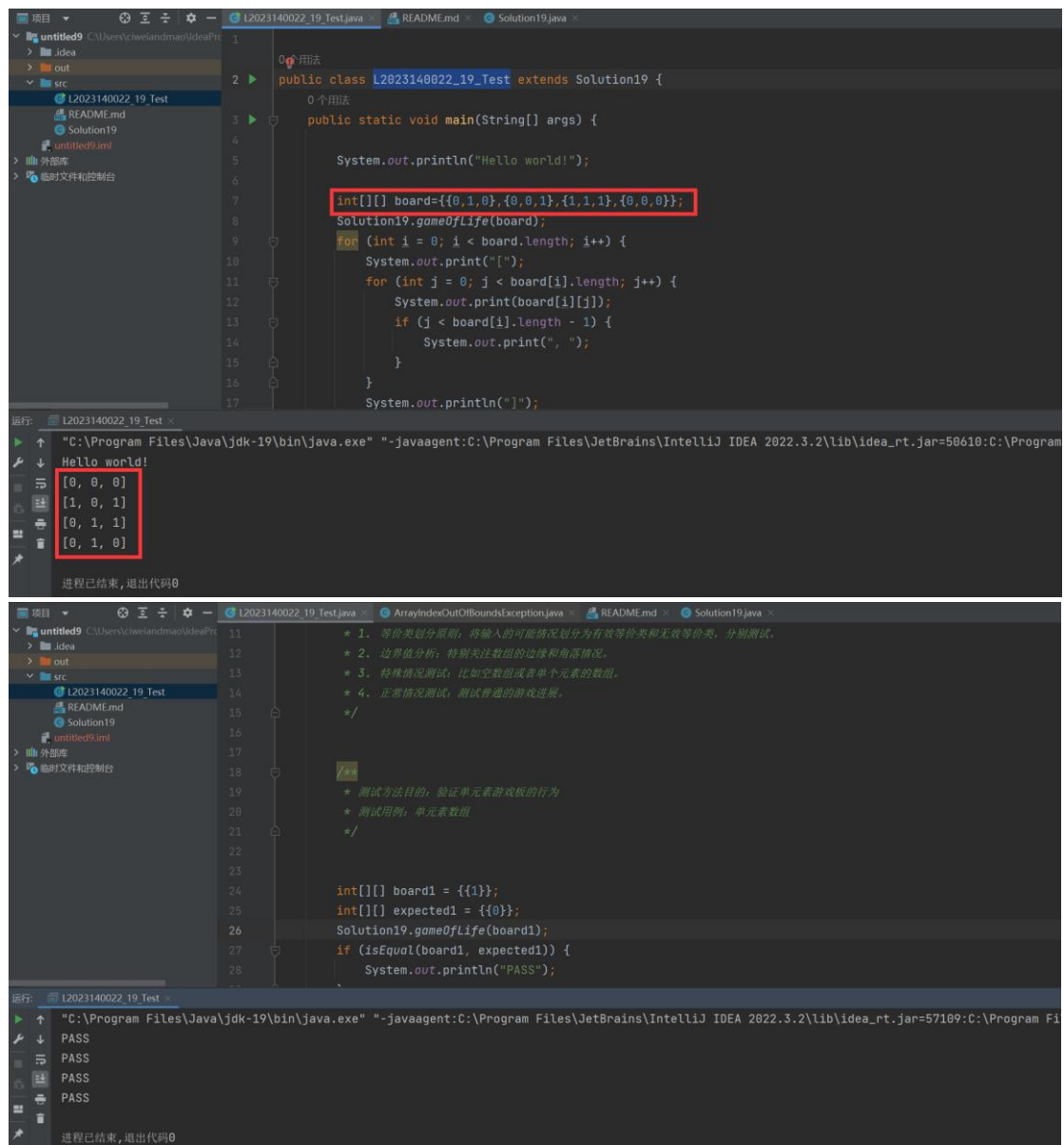
```
18 18 * 输入: board = [[1,1],[1,0]]
19 19 * 输出: [[1,1],[1,1]]
20 20 */
21 - class Solution {
22 -     public void gamofLife(int[][] board) {
21 + class Solution19 {
```



2.3 代码修改

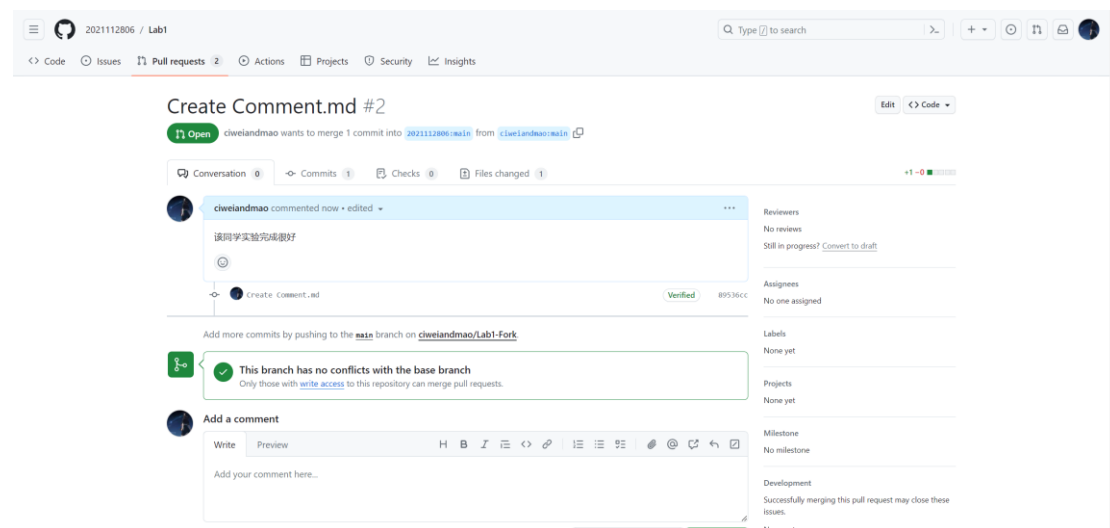
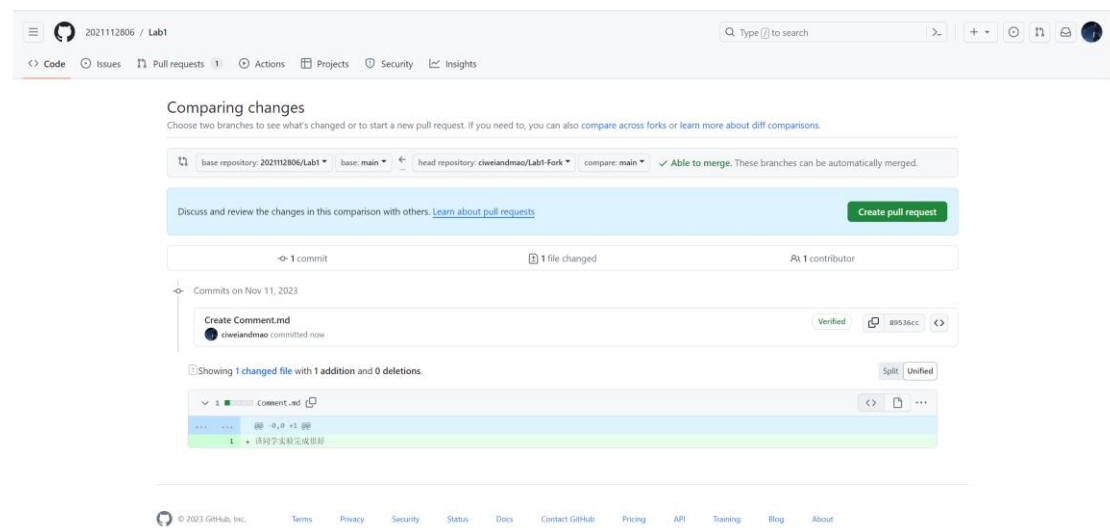
```
1 import java.util.*;
2
3 /*
4  * @Description:
5  * 生命游戏
6  * 根据百度百科, 生命游戏, 简称为 生命, 是英国数学家约翰·康威在 1970 年发明的细胞自动机。
7  * 给定一个包含 m × n 个格子的面板, 每一个格子都可以看成一个细胞。每个细胞都具有一个初始状态: 1 即为 活细胞 (Live), 或 0 即为 死细胞 (dead)。每个细胞与其八个相邻位置 (水平、垂直、对
8  * 角) 的活细胞 (Live) 的数目少于 2 个, 则该位置活细胞死亡;
9  * 如果活细胞周围八个位置有两个或三个活细胞, 则该位置活细胞仍然存活;
10  * 如果活细胞周围八个位置有超过三个活细胞, 则该位置活细胞死亡;
11  * 如果死细胞周围正好有三个活细胞, 则该位置死细胞复活;
12  * 下一个状态是通过将上述规则同时应用于当前状态下的每个细胞所形成的, 其中细胞的出生和死亡是同时发生的。给你 m × n 网格面板 board 的当前状态, 返回下一个状态。
13  *
14  * 示例 1:
15  * 输入: board = [[0,1,0],[0,0,1],[1,1,1],[0,0,0]]
16  * 输出: [[0,0,0],[1,0,1],[0,1,1],[0,1,0]]
17  * 示例 2:
18  * 输入: board = [[1,1],[1,0]]
19  * 输出: [[1,1],[1,1]]
20  */
21
22 2 个用法, 1 个测试用例
23 class Solution19 {
24     1 个用法
25     public static void gameOfLife(int[][] board) {
26         int[] neighbors = {0, 1, -1};
27
28         int rows = board.length;
29         int cols = board[0].length;
30
31         // 创建复制数组 copyBoard
32         int[][] copyBoard = new int[rows][cols];
33
34         // 从原数组复制一份到 copyBoard 中
35         for (int row = 0; row < rows; row++) {
36             for (int col = 0; col < cols; col++) {
37                 copyBoard[row][col] = board[row][col];
38             }
39         }
40
41         // 遍历面板每一个格子里的细胞
42         for (int row = 0; row < rows; row++) {
43             for (int col = 0; col < cols; col++) {
44
45                 // 对于每一个细胞统计其八个相邻位置里的活细胞数量
46                 int liveNeighbors = 0;
47
48                 for (int i = 0; i < 3; i++) {
49                     for (int j = 0; j < 3; j++) {
50
51                         if (!(neighbors[i] == 0 && neighbors[j] == 0)) {
52                             int r = (row + neighbors[i]);
53                             int c = (col + neighbors[j]);
54
55                             // 查看相邻的细胞是否是活细胞
56                             if ((r < rows && r >= 0) && (c < cols && c >= 0) && (copyBoard[r][c] == 1)) {
57                                 liveNeighbors += 1;
58                             }
59                         }
60                     }
61                 }
62
63                 // 规则 1 或规则 3
64                 if ((copyBoard[row][col] == 1) && (liveNeighbors < 2 || liveNeighbors > 3)) {
65                     board[row][col] = 0;
66                 }
67                 // 规则 4
68                 if (copyBoard[row][col] == 0 && liveNeighbors == 3) {
69                     board[row][col] = 1;
70                 }
71             }
72         }
73     }
74 }
```

2.4 测试通过截图

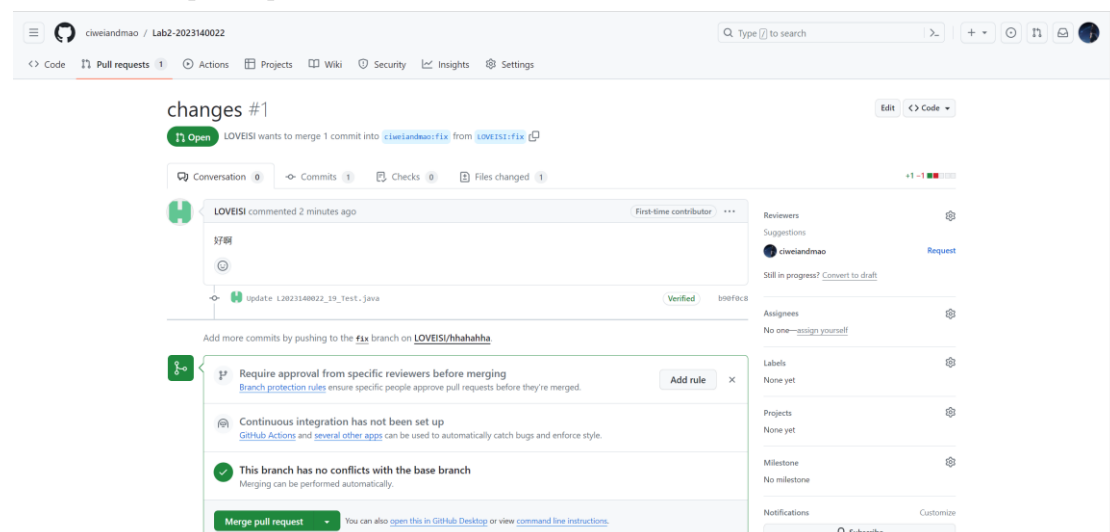


3 实验内容 2 接受 pull request

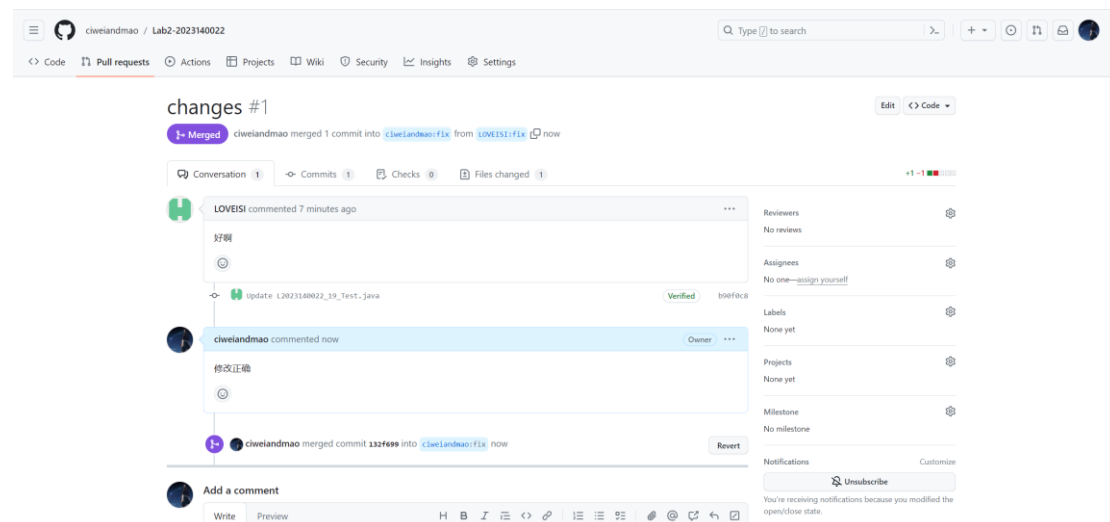
我向别人发起 pull request:



别人向我发起 pull request 交流:



我确认别人的分支并合并



4 实验内容 3 github 辅助工具

4.1 熟悉 GoodFirstIssue 工具

使自己项目被收入的方法如下：

添加新项目

欢迎您在 Good First Issue 中添加新项目，我们鼓励所有项目——新旧、大小。

请按照以下简单步骤操作：

- 我们的目标是缩小新开源贡献者的项目范围。为了保持 Good First Issue 中的项目质量，请确保您的 GitHub 存储库满足以下条件：
 - 该标签至少存在三个问题 `good first issue`。默认情况下，该标签已存在于所有存储库中。[如果没有，您可以按照此处的步骤操作。](#)
 - 它至少有 10 名贡献者。
 - 它包含包含项目详细设置说明的 README.md，以及包含新贡献者指南的 CONTRIBUTING.md。
 - 它得到积极维护。
- [在 data/repositories.toml](#) 中添加存储库的路径（按字典顺序）。
- 创建一个新的拉取请求。请在 PR 描述中添加指向存储库问题页面的链接。合并拉取请求后，更改将在 [goodfirstissue.dev](#) 上生效。

在本地设置项目

Good First Issue 有两个组件 - 使用 Nuxt.js 构建的前端应用程序和用 Python 编写的数据填充脚本。

要向网站提供新功能和更改，您需要在本地运行该应用程序。请按照以下步骤操作：

- 将项目克隆到本地。确保您的计算机上安装了 Python 3 和最新版本的 Node.js。
- 复制示例数据文件以供本地应用程序使用，并将其重命名为应用程序所需的文件名。**此步骤很重要，因为如果没有这些数据文件，前端应用程序将无法运行。**

```
$ cp data/generated.sample.json data/generated.json
$ cp data/tags.sample.json data/tags.json
```

- 构建前端应用程序并启动开发服务器。

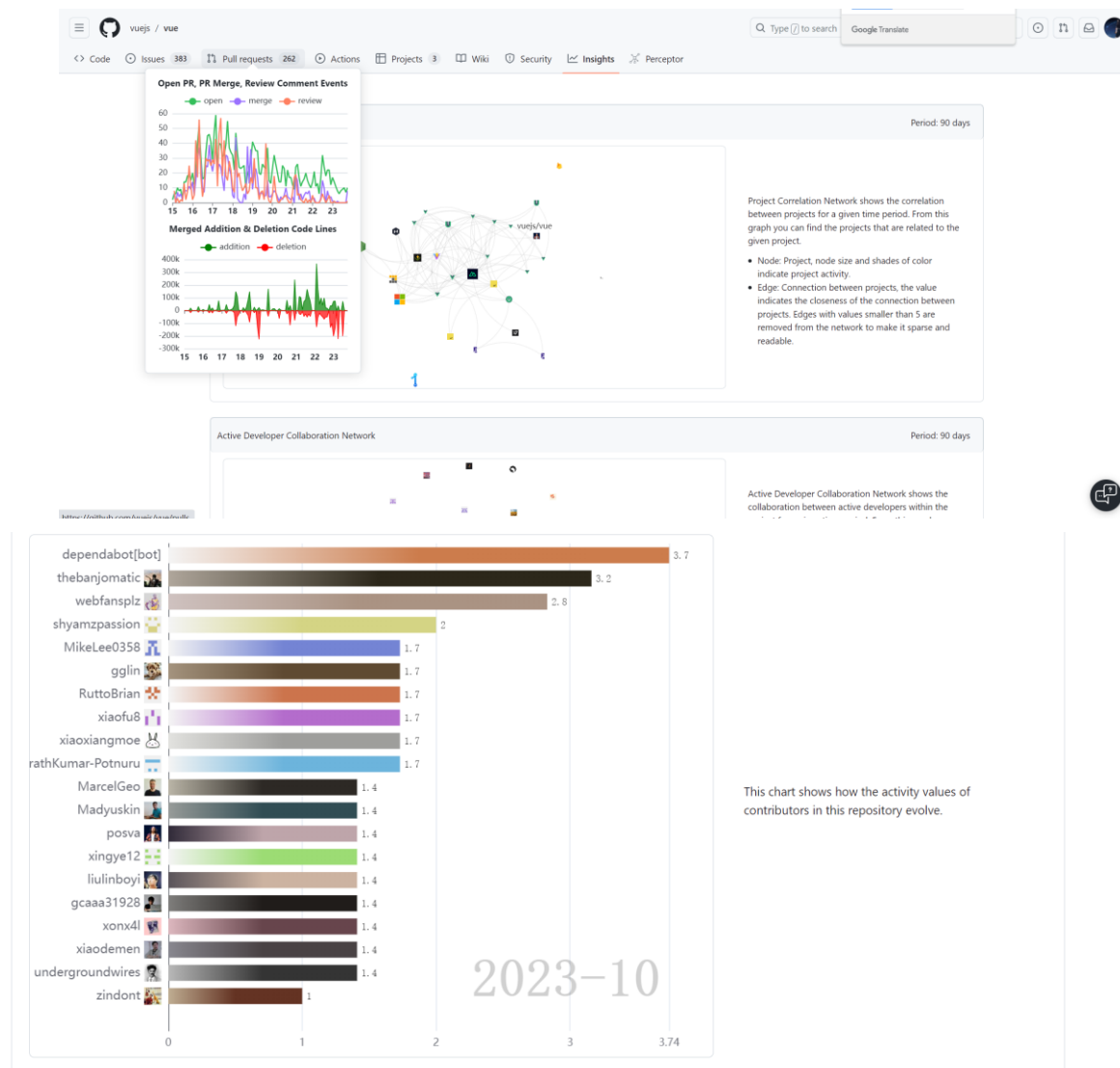
```
$ yarn # install the dependencies
$ yarn dev -o # start the development server
```

该应用程序应在您的浏览器中打开。

4.2 安装并使用 Hypercrx

选择 github 项目: <https://github.com/vuejs/vue>

选择 Perceptor 条目如下



4.3 利用 OpenLeaderboard 工具

github 上开源项目统计如下:



活跃度是一个用于衡量开源项目活跃程度和健康状况的指标。它帮助评估项目是否得到持续的维护和发展，以及社区是否活跃。一种基于 GitHub 行为数据的加权活跃度算法被提出，其计算方式是将特定行为（如 Issue 评论、发起 Issue、发起 PR、PR 上的代码 review 评论、PR 合入）的发生次数乘以一个权重值。每个开发者的活跃度通过开方转换为项目的活跃度，以降低核心开发者高活跃度对整体项目评估的影响。这种方法试图达到活跃度更加均衡分布的效果。

协作影响力是一种用于评价开源项目在整个开源生态中的重要性和影响力的指标。开源协作网络基于开发者在不同项目间的协作行为构建。如果一个开发者在两个项目上都非常活跃，这两个项目就被认为具有较高的协作关联度。这种关联通常反映了项目之间的依赖或合作关系，如项目间的上下游关系。协作关联度的计算使用调和平均的方法。仅当开发者在两个项目上都非常活跃时，才对这两个项目的关联度产生较大影响。利用 PageRank 算法，一种图分析算法，来计算每个项目的协作影响力。这个算法原先用于 Google 搜索引擎的网页排名，依据的是页面之间的引用关系。在开源协作网络中，协作影响力的思路是：具有较大影响力的项目会与更多其他项目有协作关系；与影响力较大的项目协作关联度高的项目也具有较大的影响力。

价值流网络是一个用于衡量和分析开源软件及其生态的复杂数学模型。它主要关注从生产端到消费端的整个流程，以更全面地衡量开源生态的价值。价值流网络考虑到了开源软件的生产侧和消费侧，强调不仅仅是谁在使用开源代码，还包括谁开发了这些代码。这种视角认为，即使两个开发者活跃度相同，优秀的开发者产生的价值与初学者是不同的。这个模型旨在构建从生产端到消费端的完整模型，直接衡量每个软件的社会价值，并反向推演每个开发者的价值。它考虑了开发者与项目之间的活跃度、项目之间的依赖关系、以及开发者之间的关注关系等多个方面。价值流网络部分基于协作影响力模型，考虑项目间的依赖和使用关系，以及开发者对项目的活跃度和关注度。这个模型可以更全面地反映开源项目的真实社会价值。价值流网络模型设计灵活，可以容纳更多元的开源数据，而不需要对模型进行大幅度的修改。业务模型的设计不受底层数学模型的限制，使得上层的开源生态描述可以在不关心底层数学模型的情况下进行。

OpenRank 是一种用于评估开源项目重要性的算法，受到 Google 的 PageRank 算法的启发。OpenRank 主要基于项目间的关系来进行评估。这些关系可能包括项目之间的依赖关系、开发者的贡献模式、以及项目间的合作关系等。OpenRank 借鉴了 PageRank 的核心思想，即一个项目的重要性不仅由自身的活跃度决定，还受到与其相关联的其他项目的影响。在 PageRank 中，一个网页的重要性由链接到它的其他网页的数量和质量共同决定。类似地，

在 OpenRank 中，一个开源项目的重要性由参与和引用它的其他项目共同影响。OpenRank 中可能会考虑不同类型的关系或活动的权重。例如，直接的代码贡献可能比简单的项目引用更重要。OpenRank 考虑了开源项目之间的网络效应。一个项目如果被许多其他重要的项目依赖或引用，那么该项目的重要性也会相应提高。OpenRank 的计算通常是迭代进行的，直至达到一个稳定的重要性分布。在每一轮迭代中，项目的重要性得分会根据与其相关的其他项目的当前重要性进行更新。虽然主要基于项目间的关系，OpenRank 还可能综合考虑其他因素，如项目活跃度、社区大小和参与度等，来更全面地评估项目的重要性。

5 小结

本次实验我更加熟悉了 git 的命令，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，熟悉了 github 中常用开源软件开发工具，特别是 GoodFirstIssue, Hypercrx, OpenLeaderboard，了解了活跃度，价值网络，协作影响力，OpenRank 等业内术语，学会了发送与接受 pull request，在本次实验中我受益匪浅。