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
马嘉良	2023120259	13596489104

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：

1. github 上 fork 题目仓库到个人账号中
2. Clone github 上 fork 后的仓库到本地
3. 创建 fix 分支，在 fix 分支上修改所分配题目代码中的所有错误，编写测试类，测试通过无误后，提交到本地仓库并推送到 github 上
4. 在 github 上向编程题目所在仓库提交 pull request

掌握基于 github 的软件项目协作开发命令和方法：

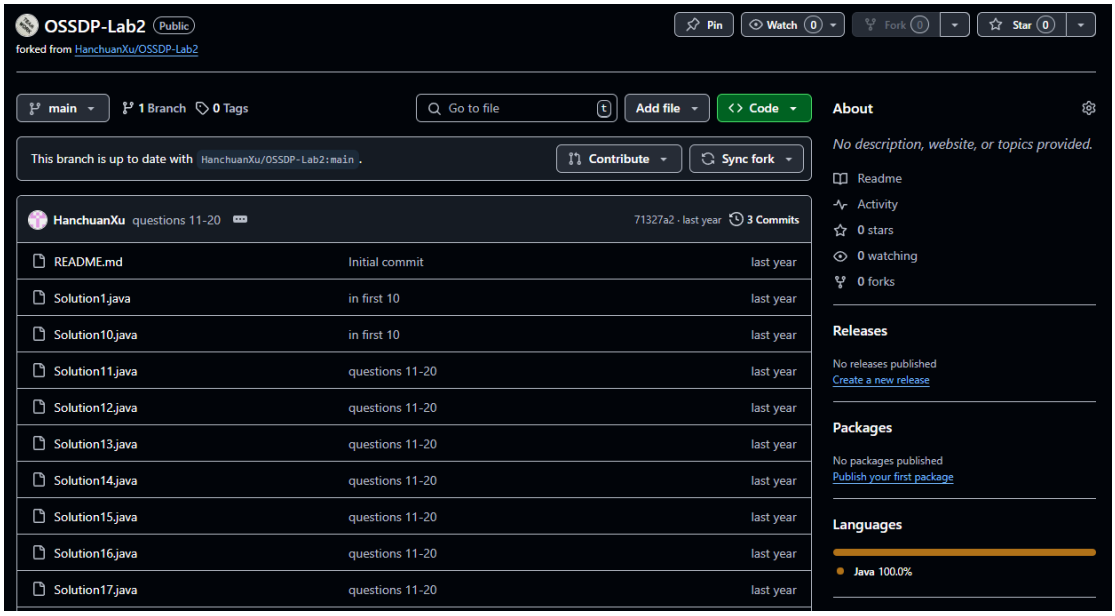
1. 创建一个新的仓库，将实验内容 1 中建立的代码仓库中所有内容（包含代码修改成功后的文件）复制到新仓库中，任选一位其他同学，相互 fork 对方在实验内容 2 中建立的新仓库。
2. 评审对方的程序代码和测试用例，对其中的文件进行修改（修改内容不强制要求：修改代码、添加注释、添加评论等均可），然后提交 PR
3. 通过 PR 进行交流
4. 检查无误后接受 PR

熟悉几个 github 中常用开源软件开发工具：

1. 熟悉 https://goodfirstissue.dev 网站的使用
2. 熟悉 https://open-leaderboard.x-lab.info
3. 安装并使用 https://github.com/hypertrons/hypertrons-crx

2 实验内容 1 发送 pull request

2.1 fork 项目



2.2 git 操作命令

git clone https://github.com/under-tree/OSSDP-Lab2.git
git checkout -b solution16-fix
git add Solution16.java L2023120259_16_Test.java
git commit -m "fix up and test"
git push -u origin solution16-fix

```
  [/e/x/courses/24F/C8/OSSDP-Lab2]
$ git status
On branch solution16-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)
        modified:   Solution16.java

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        L2023120259_16_Test.java

no changes added to commit (use "git add" and/or "git commit -a")
  [/e/x/courses/24F/C8/OSSDP-Lab2]
$ git add Solution16.java L2023120259_16_Test.java
  [/e/x/courses/24F/C8/OSSDP-Lab2]
$ git commit -m "fix up and test"
[solution16-fix 0403f8e] fix up and test
 2 files changed, 126 insertions(+), 13 deletions(-)
 create mode 100644 L2023120259_16_Test.java
```

```
  [/e/x/courses/24F/C8/OSSDP-Lab2]
$ git push -u origin solution16-fix
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 1.40 KiB | 1.40 MiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'solution16-fix' on GitHub by visiting:
remote:      https://github.com/under-tree/OSSDP-Lab2/pull/new/solution16-fix
remote:
To https://github.com/under-tree/OSSDP-Lab2.git
 * [new branch]      solution16-fix -> solution16-fix
branch 'solution16-fix' set up to track 'origin/solution16-fix'.
```

2.3 代码修改

```java
import java.util.*;

/*
 * @Description
 * 最大数
 * 给定一组非负整数 nums，重新排列每个数的顺序（每个数不可拆分）使之组成一个最大的整数。
 * 注意：输出结果可能非常大，所以你需要返回一个字符串而不是整数。
```

```
 *
 * 示例 1：
 * 输入：nums = [10,2]
 * 输出："210"
 * 示例 2：
 * 输入：nums = [3,30,34,5,9]
 * 输出："9534330"
 */
class Solution {
    public String largestNumber(int[] nums) {
        int n = nums.length;
        // 转换成包装类型，以便传入 Comparator 对象（此处为 lambda 表达式）
        Integer[] numsArr = new Integer[n];
        for (int i = 0; i < n; i++) {
            numsArr[i] = nums[i];
        }

        Arrays.sort(numsArr, (x, y) -> {
            String sx = x.toString() + y.toString();
            String sy = y.toString() + x.toString();
            return sy.compareTo(sx);
        });

        if (numsArr[0] == 0) {
            return "0";
        }
        StringBuilder ret = new StringBuilder();
        for (int num : numsArr) {
            ret.append(num);
        }
        return ret.toString();
    }
}
```

2.4 测试类代码

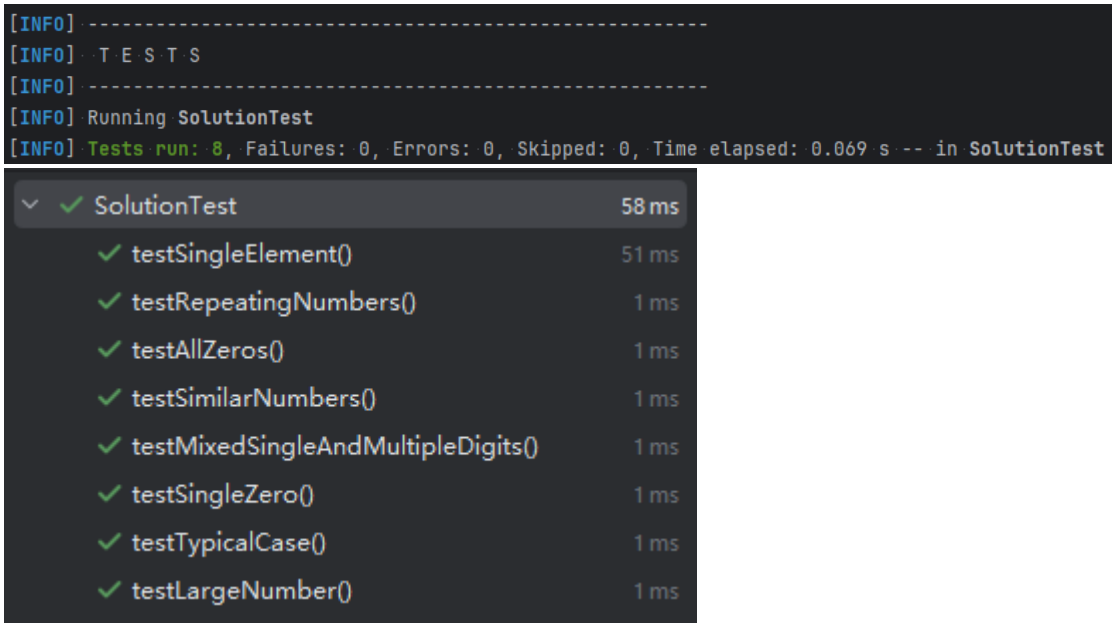
```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

/*
 * 测试用例设计总体原则
 * 1. 等价类划分：分为有效输入（包含正常数字组合）和无效输入（全为 0 的特殊情况）。
 * 2. 边界值分析：数组长度为 1、数组包含最大/最小值等边界情况。
```

```
 * 3. 特殊输入测试：重复数字、多个数字组成相同大小的结果（如 121 和 12）。
 * 4. 典型功能测试：验证普通情况下的正确输出。
 */

class SolutionTest {
    private final Solution solution = new Solution();

    /**
     * 测试目的：
     * 验证普通情况下（包含多位数和单个位数混合的数组）是否返回正确结果。
     * 测试用例：
     * nums = [3, 30, 34, 5, 9]
     */
    @Test
    void testTypicalCase() {
        int[] nums = {3, 30, 34, 5, 9};
        String result = solution.largestNumber(nums);
        assertEquals("9534330", result);
    }

    /**
     * 测试目的：
     * 验证数组中只有一个元素时，是否直接返回该元素的字符串形式。
     * 测试用例：
     * nums = [10]
     */
    @Test
    void testSingleElement() {
        int[] nums = {10};
        String result = solution.largestNumber(nums);
        assertEquals("10", result);
    }

    /**
     * 测试目的：
     * 验证数组中所有元素均为 0 时，是否返回 "0"。
     * 测试用例：
     * nums = [0, 0, 0]
     */
    @Test
    void testAllZeros() {
        int[] nums = {0, 0, 0};
        String result = solution.largestNumber(nums);
        assertEquals("0", result);
```

```
    }

    /**
     * 测试目的：
     * 验证数组长度为 1 且为 0 的特殊情况。
     * 测试用例：
     * nums = [0]
     */
    @Test
    void testSingleZero() {
        int[] nums = {0};
        String result = solution.largestNumber(nums);
        assertEquals("0", result);
    }

    /**
     * 测试目的：
     * 验证数组中包含重复数字时，是否能够正确排序。
     * 测试用例：
     * nums = [1, 11, 111]
     */
    @Test
    void testRepeatingNumbers() {
        int[] nums = {1, 11, 111};
        String result = solution.largestNumber(nums);
        assertEquals("111111", result);
    }

    /**
     * 测试目的：
     * 验证边界情况，数组包含较大的值时是否能够正常处理。
     * 测试用例：
     * nums = [0, 999999999]
     */
    @Test
    void testLargeNumber() {
        int[] nums = {0, 999999999};
        String result = solution.largestNumber(nums);
        assertEquals("9999999990", result);
    }

    /**
     * 测试目的：
     * 验证特殊情况下，数组中的数字组成相同大小的结果。
```

```
     * 测试用例：
     * nums = [12, 121]
     */
    @Test
    void testSimilarNumbers() {
        int[] nums = {12, 121};
        String result = solution.largestNumber(nums);
        assertEquals("12121", result);
    }

    /**
     * 测试目的：
     * 验证数字中包含多个不同的个位数和多位数混合情况。
     * 测试用例：
     * nums = [10, 2]
     */
    @Test
    void testMixedSingleAndMultipleDigits() {
        int[] nums = {10, 2};
        String result = solution.largestNumber(nums);
        assertEquals("210", result);
    }
}
```

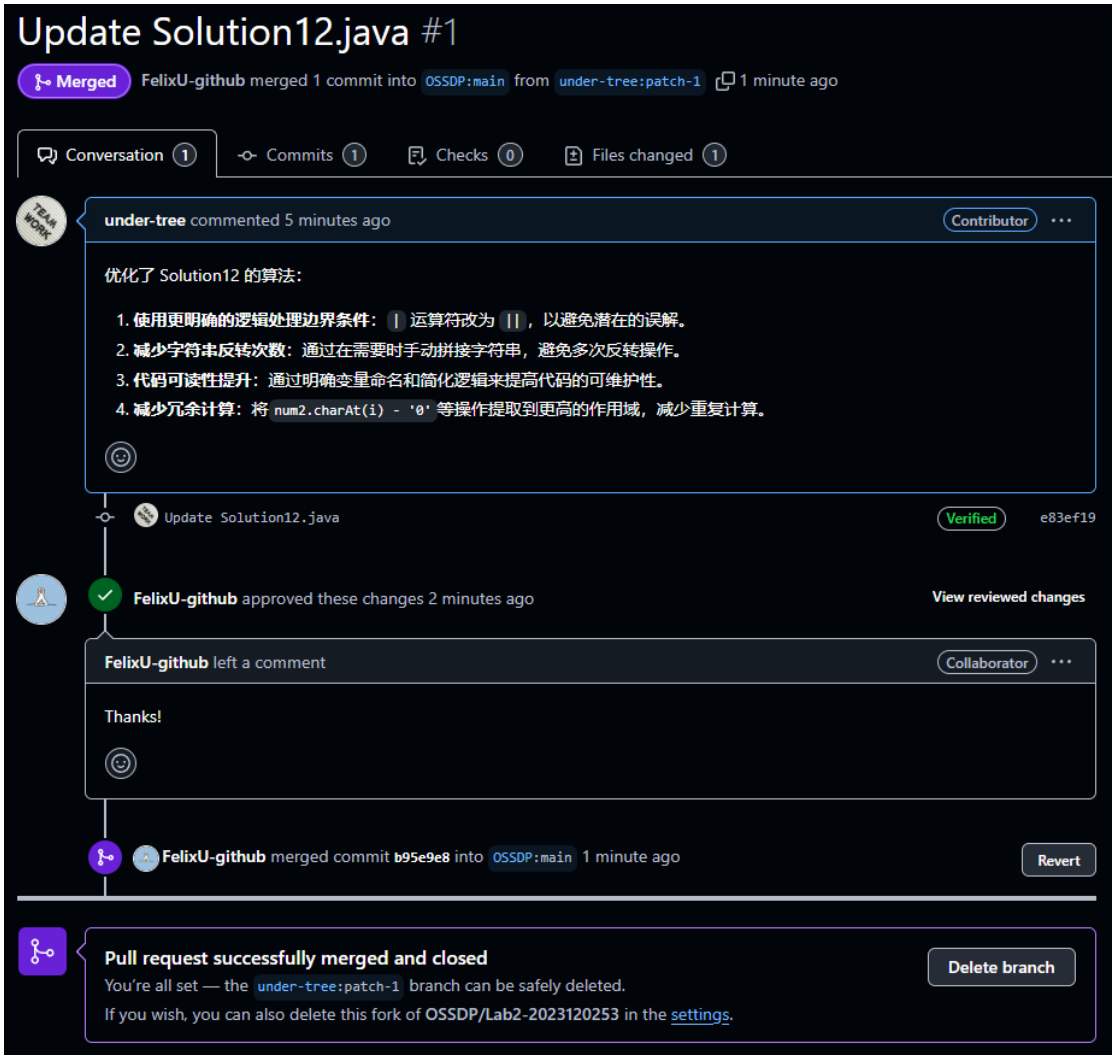
2.5 测试通过截图



3 实验内容 2 接受 pull request

接受张嘉夫同学的 PR：

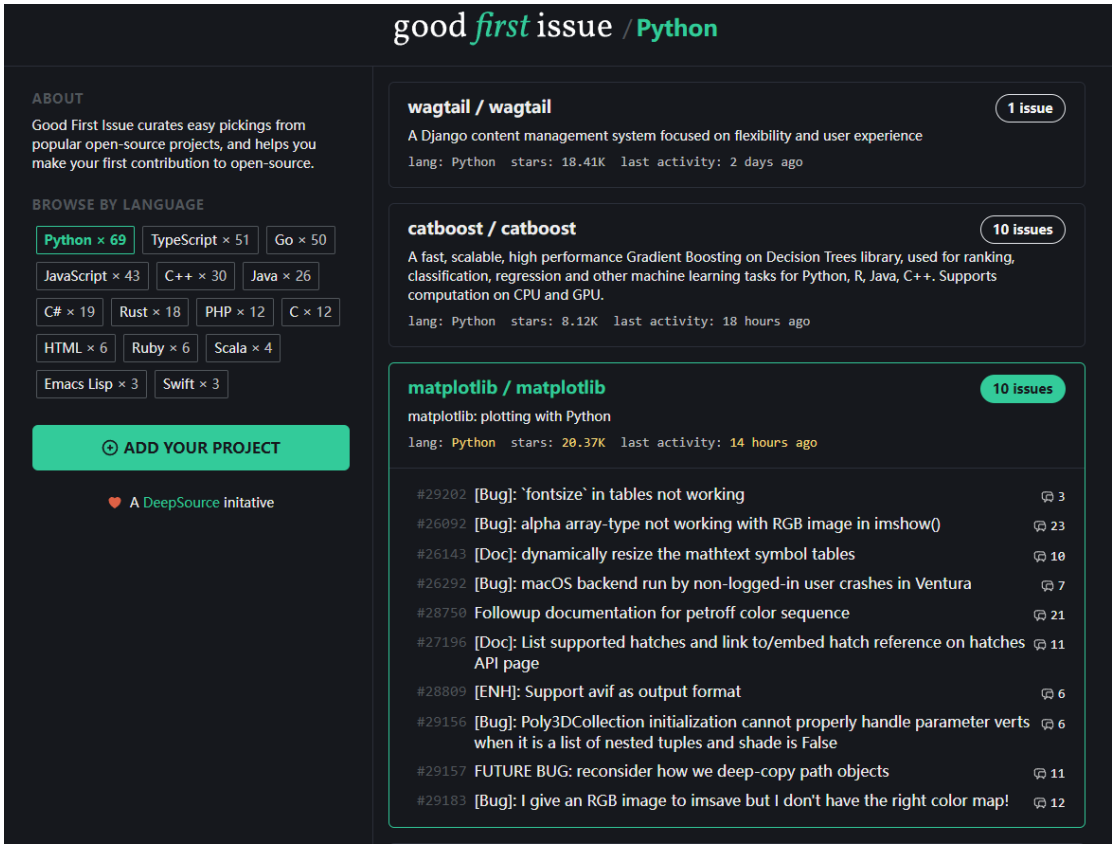
给余昊卿同学的项目提 PR：



4 实验内容 3 github 辅助工具

4.1 熟悉 GoodFirstIssue 工具

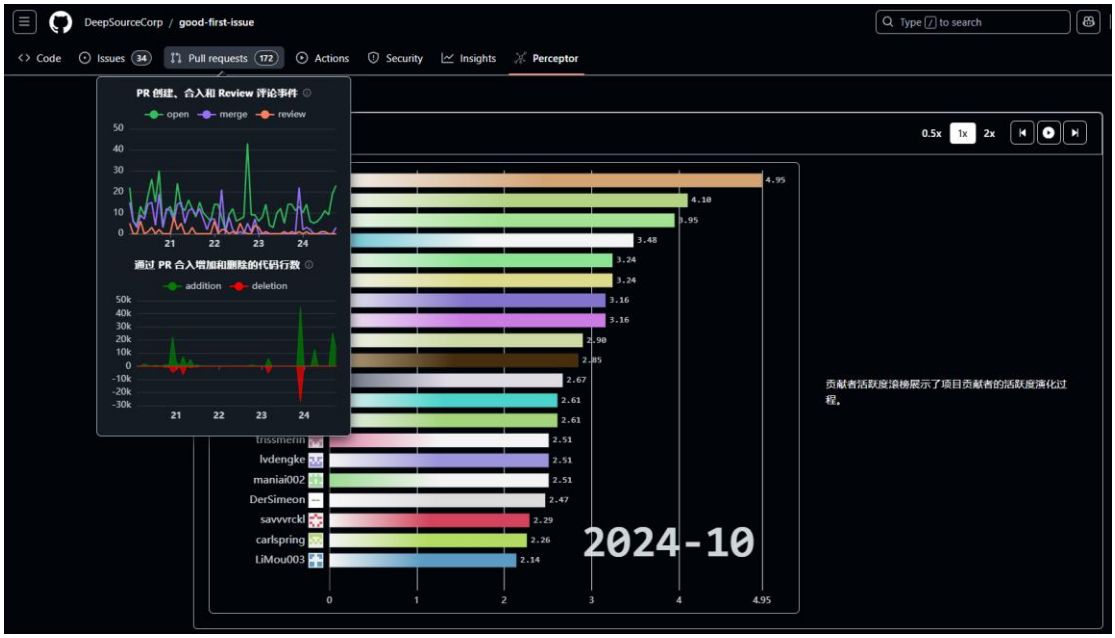
在 good first issue 网站可以通过标签筛选各语言的项目，并查看项目的信息，如下图所示：



为了使自己的项目被 good first issue 收录，可以向 good first issue 提交包含自己项目地址的 PR。项目要满足以下条件：创建 good first issue 标签，用来标记适合新手的简单问题，在项目的 issue 列表中，至少有三个 issue 使用了该标签；项目至少有十位贡献者；项目的 README.md 文件中有清晰的项目构建说明，项目中应该包含清晰的贡献指南（CONTRIBUTING.md）。

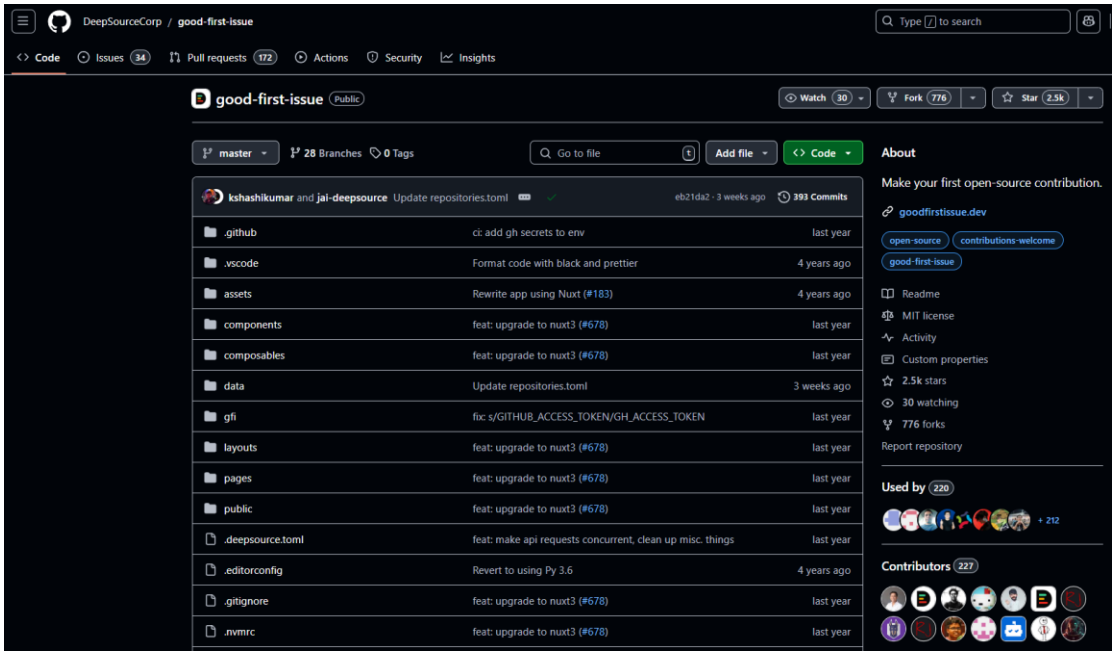
4.2 安装并使用 Hypercrx

使用插件后：



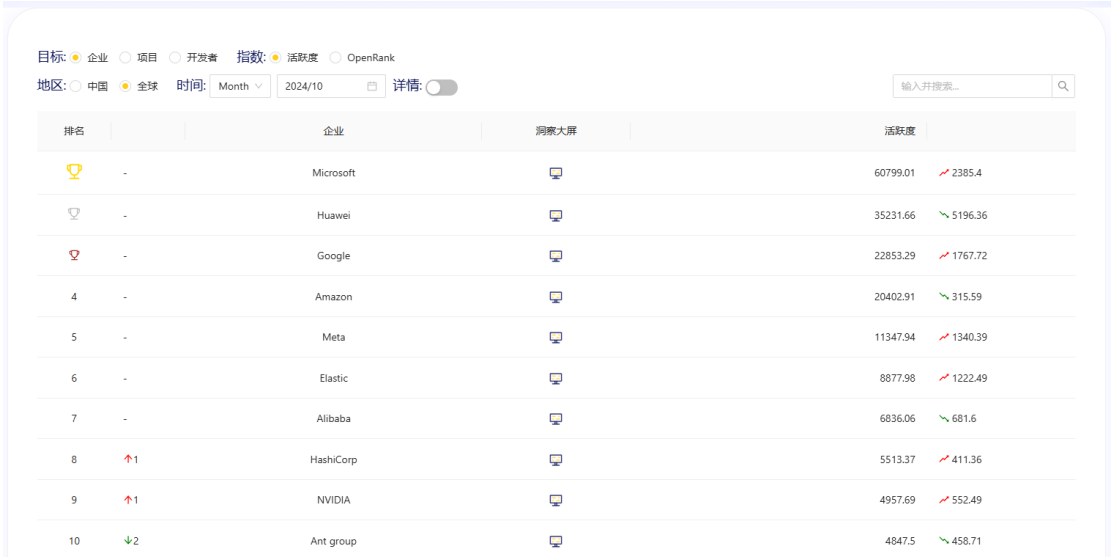
使用插件之前导航栏中没有 Perceptor 选项，鼠标悬浮在 Issues，Pull requests 等选项上时不会显示统计信息。

使用插件前：



4.3 利用 OpenLeaderboard 工具

在 OpenLeaderboard 上查看企业开源项目的活跃度：



目标: 企业 项目 开发者 指数: 活跃度 OpenRank
地区: 中国 全球 时间: Month 2024/10 详情:

排名		企业	洞察大屏	活跃度	
	-	Microsoft		60799.01	2385.4
	-	Huawei		35231.66	5196.36
	-	Google		22853.29	1767.72
4	-	Amazon		20402.91	315.59
5	-	Meta		11347.94	1340.39
6	-	Elastic		8877.98	1222.49
7	-	Alibaba		6836.06	681.6
8	↑1	HashiCorp		5513.37	411.36
9	↑1	NVIDIA		4957.69	552.49
10	↓2	Ant group		4847.5	458.71

开源项目活跃度用于衡量项目的动态性，基于开发者在特定时间内的贡献，包括代码提交、问题处理、代码审阅和讨论等活动。通过为不同活动分配权重，例如代码贡献较高、问题处理次之等，可以综合计算每个开发者的活跃度，然后汇总得出项目的整体活跃度。这种方法能过滤掉机械化的行为（如机器人提交）对活跃度的干扰。

开源项目的影响力基于开源协作网络计算，其核心是分析项目之间的协作关系。通过构建协作网络（例如开发者在多个项目中贡献导致的联系），应用类似 PageRank 的算法评估项目的影响力。高影响力的项目往往与更多项目有强关联，并影响这些项目的协作生态。例如，VSCode 和 Flutter 等项目在 2019 年具有显著的协作影响力。

价值流网络将开源生态中的实体（开发者、项目、公司等）纳入复杂的价值交换网络。通过价值流动模拟实体间的贡献和影响，比如开发者的贡献值会部分流向其参与的项目，而项目间的依赖关系也会传递价值。这个模型可以更全面地量化和评估开源生态中的价值分布，并与社会效用挂钩。

OpenRank 是一种泛化的影响力评价方法，结合活跃度、关注度和协作关系进行计算。类似 PageRank 算法，它通过分析项目间的依赖关系和开发者的参与度，建立稳定的价值网络，最终为每个项目分配一个代表其综合影响力的分值。此方法能够消除刷活跃度等非真实行为的影响，并展现真实的生态协作与依赖情况

5 小结

通过本次实验，我深入了解了基于 GitHub 的开源协作开发流程，对以下内容有了更全面的掌握：

代码托管与版本控制：

1. 学会了在 GitHub 平台上创建、管理、Fork 仓库，以及使用分支进行独立开发。
2. 理解了 Pull Request 的意义及其在协作开发中的作用。

协作开发技巧：

1. 熟悉了如何通过 Fork、Clone 和 Pull Request 等功能与他人协作开发项目。
2. 通过代码评审与 PR 评论，学习了如何与团队成员有效沟通，优化代码质量。

开源工具的使用：

1. 掌握了 Good First Issue 等工具的使用方法，为未来参与开源项目奠定了基础。
2. 了解了 Hypertrons 的功能，增强了对开源项目协作过程的理解。