

C#刷遍Leetcode面试题系列连载（1） - 入门与工具简介

DotNetCore学习站 1周前

以下文章来源于dotNET匠人，作者Bravo Yeung



dotNET匠人

分享.NET前沿技术、干货知识，Bravo Yeung与你同行💖



点击蓝字“dotNET匠人”关注我哟
加个“星标★”，每日 7:15，好文必达！



什么要刷LeetCode

大家都知道，很多对算法要求高一点的软件公司，比如美国的FLAGM (Facebook、LinkedIn、Amazon/Apple、Google、Microsoft)，或国内大厂BAT、TMD、华为，以及国内新兴的 AI 公司等，都对算法水平有所要求。据悉知名游戏公司的算法岗收入很高，相应的对算法要求也比较高。而这些公司，大多数会以 LeetCode 中的题目或基于 LeetCode 改进后的自家算法题库来考察候选人。

于是想进入上述大厂，定期做 LeetCode 题目很有必要。即使没打算进这些大厂，坚持做LeetCode，个人的算法水平、编程能力也会有较大提升。本文主要介绍 .NET 开发者如何入手刷 LeetCode 面试题。



刷LeetCode有哪些好处？

- 提升抽象思维水平

计算机中有很多抽象的数据结构，比如：List、Stack(栈)、Linked List(链表)、Hash Table(哈希表)、Heap(堆)、Tree等等，而LeetCode 上的大量高质量算法题基本上涵盖了所有这些数据结构的应用。怎么将这些题抽象成数学模型，转化为具体数据结构的应用，则是我们需要提升的地方，而这恰恰帮我们极大地提升了自己的抽象思维水平。

■ 时刻对时间、空间复杂度有所掌控

每一个算法实现都有其相应的时间复杂度和空间复杂度，而 LeetCode中的一些题对时间复杂度有明确要求，有的要求 $O(n)$ 即可，有的则要求 $O(1)$ 才行，否则代码无法 AC(Accepted)。

■ 对各个语言有更深入的理解

比如你上学时用 C++ 刷一遍，工作后再用 C# 刷一遍，最后再用 Python 刷一遍，那么你对各个语言的代码量和性能都会有更深地体会，也会知道用哪一种语言具体该怎么写出更简短、更优雅的代码。

LeetCode vs 传统的 OJ

中学有NOI信息竞赛，大学有ACM算法竞赛，按理说应该都有各自的 OJ，这两类 OJ 不是业内公司的真实面试题。NOI了解的不是特别多，据说以前用 Pascal 语言解题。相应地，我知道有个中学生比较多的 OJ 叫洛谷(<https://www.luogu.org/>)。而ACM有不少免费 OJ，国内就有一些ACM OJ，比如POJ(北大的)、HDU OJ(杭电的)、ZOJ(浙大的)、HUST OJ(华科的)。另外，还有些职场用的OJ，比如牛客网OJ、浙江大学计算机程序设计能力考试 PTA(拼题A，原全称为Programming Ability Test，简称PAT) 等等。

而国外有 UVAoj <http://uva.onlinejudge.org/>,

TopCoder (<http://www.topcoder.com/tc>)等等，另外还记得微软的校招笔试用的 OJ 是 hihoCoder <http://hihocoder.com/>。

而 LeetCode 较上述 OJ 而言有如下优势：

- 题基本上都来自于业内大公司的真实面试题
- 题目不是竞赛性质，难度也适中
- 支持多种主流编程语言C++/C/C#/Python/Java/js/Ruby/PHP/Kotlin等
- 不用处理输入输出问题，可以集中精力解决具体问题
- 提供Discuss环境，可参考他人代码
- 提供 执行用时分布图表，可看到自己提交的代码在所有已AC代码中的运行效率排名

传统的 OJ 对用户代码的判定状态有如下几种：

- 1、Accepted. ——通过! (AC)
- 2、Wrong Answer.——答案错。(WA)
- 3、RunTime Error.——程序运行出错，意外终止等。(RTE)
- 4、Time Limit Exceeded. ——超时。程序没在规定时间内出答案。(TLE)
- 5、Presentation Error. ——格式错。程序没按规定的格式输出答案。(PE)
- 6、Memory Limit Exceeded. ——超内存。程序没在规定空间内出答案。(MLE)
- 7、Compile Error. ——编译错。程序编译不过。(CE)

而在 LeetCode 中，应该是没有第5种状态的。

刷 OJ 时，大家还会常用两个词：

- **AK(ALL KILL)**，把比赛中所有题都做出来了，出题方需要考虑 **防AK** 策略。
- **1Y**: 第一次提交就正确了，也就是大家常说的“一遍过”

刷题时大家的一致感觉是：AC一时爽，一直AC一直爽！

LeetCode刷题时的心态建设

在本文开头，我们提到了很多行内名厂需要考察与 LeetCode 难度相当的算法题。但需要注意的是，面试时很可能面试官会对题目的解题要求进行另外的调整，比如要求更低的时间复杂度、更低的空间复杂度之类的，所以呢，面试算法题很重要的是要和面试官保持即时的沟通，而不是一上来就埋头写代码。

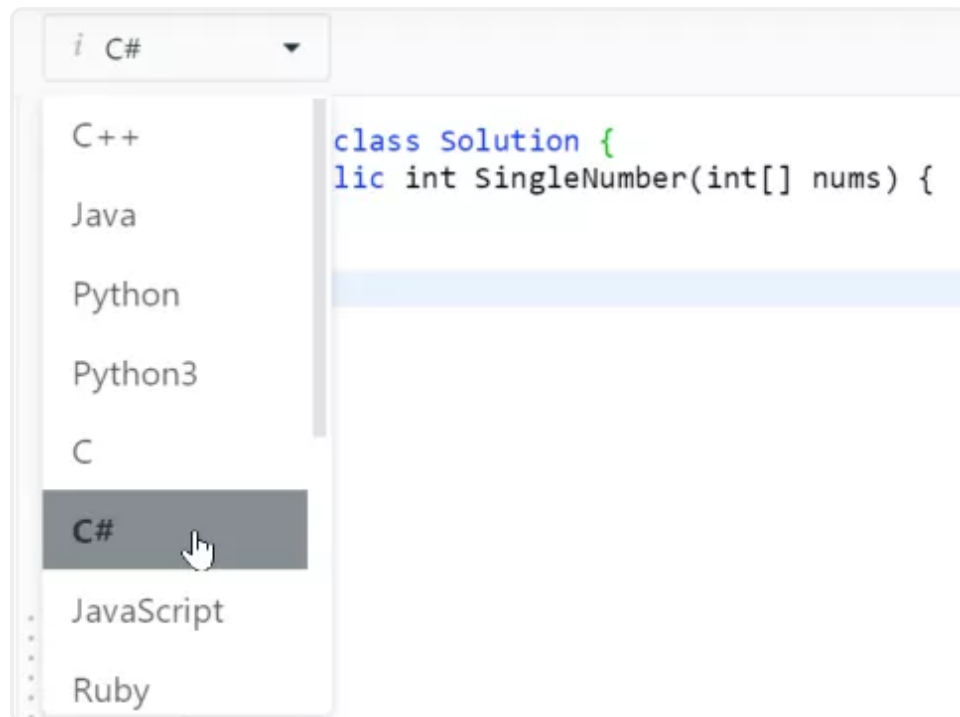
有的人刷 LeetCode 中 难度为 Easy 和 Medium 的题3遍左右后成功拿到微软Offer，还是Special Offer！也有人刷完LeetCode很多遍，也没能拿到大厂Offer呢。

更多算法题面试要点请参看**一亩三分地**论坛站长Warald 的文章 [Leetcode刷题五遍还没offer！举例分析为什么找工作光刷题不够](#)。

C#如何刷遍LeetCode

在 LeetCode 中提交 C# 代码有两种主流方式，下面以 LeetCode 中的136号题为例来说明。

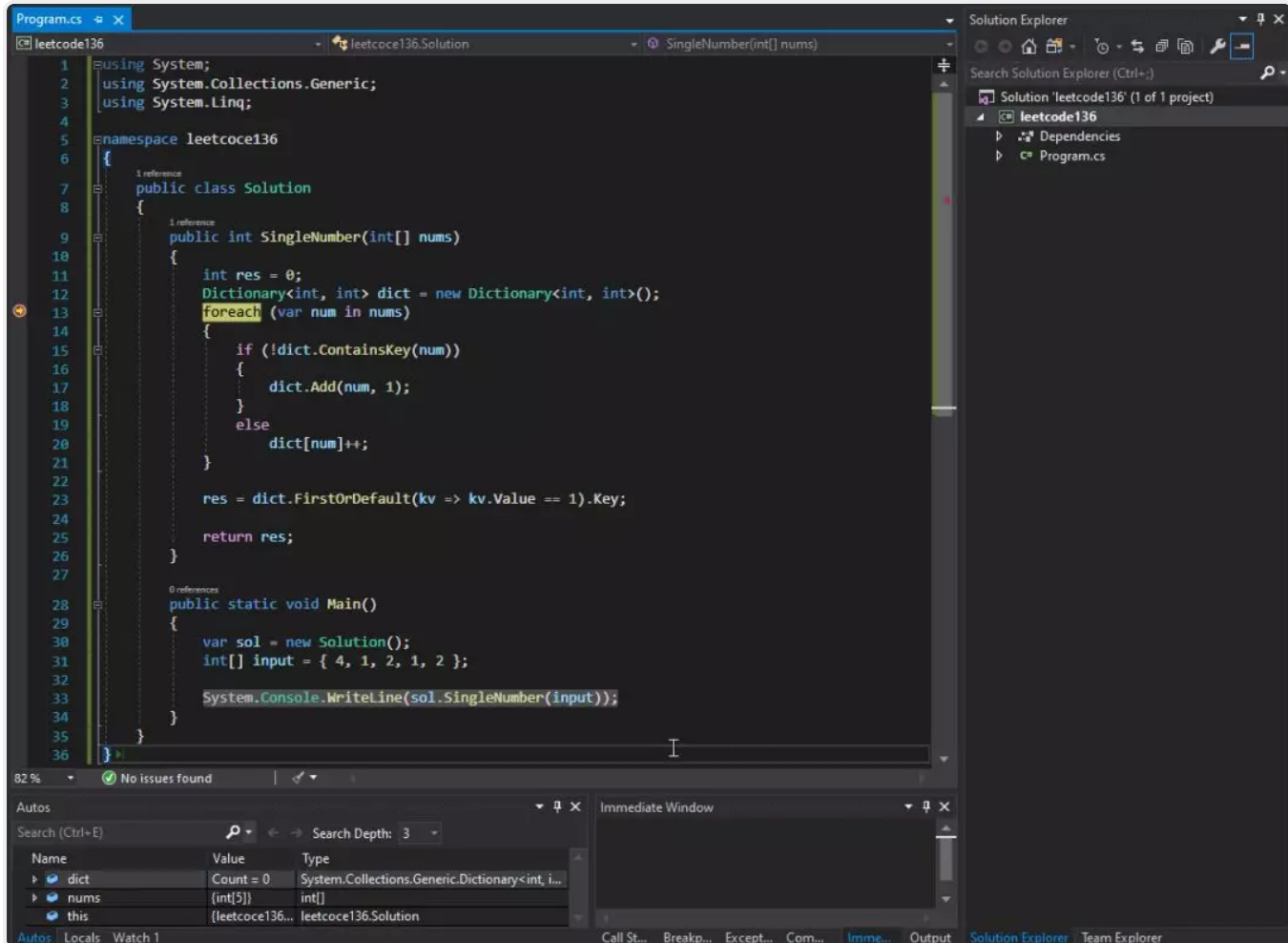
该题的中文版网址为: <https://leetcode-cn.com/problems/single-number/>，将代码语言选为C#，则默认的接口代码如下：



```
public class Solution {  
    public int SingleNumber(int[] nums) {  
  
    }  
}
```

选项1：VS本地Debug + 在线验证后提交

- 在本地Visual Studio中创建 .NET Core/Framework 项目
- 将所生成项目中的 Program.cs 中 `class Program`改为 `public class Solution`
- 接下来把相应的代码放在类 Solution 里面



如果需要本地测试，只需在该类里面加入 `主函数` 即可，然后在 `主函数` 中调用相应的函数，debug，观察调用时的各项值。

使用LeetCode的"执行代码"案例可以测试当前的测试用例，而界面上的"测试用例"可以自行修改。

```
1 public class Solution
2 {
3     public int SingleNumber(int[] nums)
4     {
5         int res = 0;
6         Dictionary<int, int> dict = new Dictionary<int, int>();
7         foreach (var num in nums)
8         {
9             if (!dict.ContainsKey(num))
10            {
11                dict.Add(num, 1);
12            }
13            else
14                dict[num]++;
15        }
16
17        res = dict.FirstOrDefault(kv => kv.Value == 1).Key;
18
19        return res;
20    }
21 }
```

测试用例 代码执行结果

已完成 执行用时: 100 ms

输入 [2,2,1]

输出 1 差别

预期结果 1

控制台 如何创建一个测试用例?

执行代码 提交

如果此时对代码比较有信心，可以直接点"Test"按钮左侧的"Submit"按钮提交代码了。

如果不幸，部分Test case无法通过，则可进行本地debug，下面是我给出的样例代码：

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace leetcoce136
{
    public class Solution
    {
        public int SingleNumber(int[] nums)
        {
            int res = 0;
            Dictionary<int, int> dict = new Dictionary<int, int>();
            foreach (var num in nums)
            {
                if (!dict.ContainsKey(num))
                {
                    dict.Add(num, 1);
                }
                else
                    dict[num]++;
            }

            res = dict.FirstOrDefault(kv => kv.Value == 1).Key;

            return res;
        }
    }
}
```

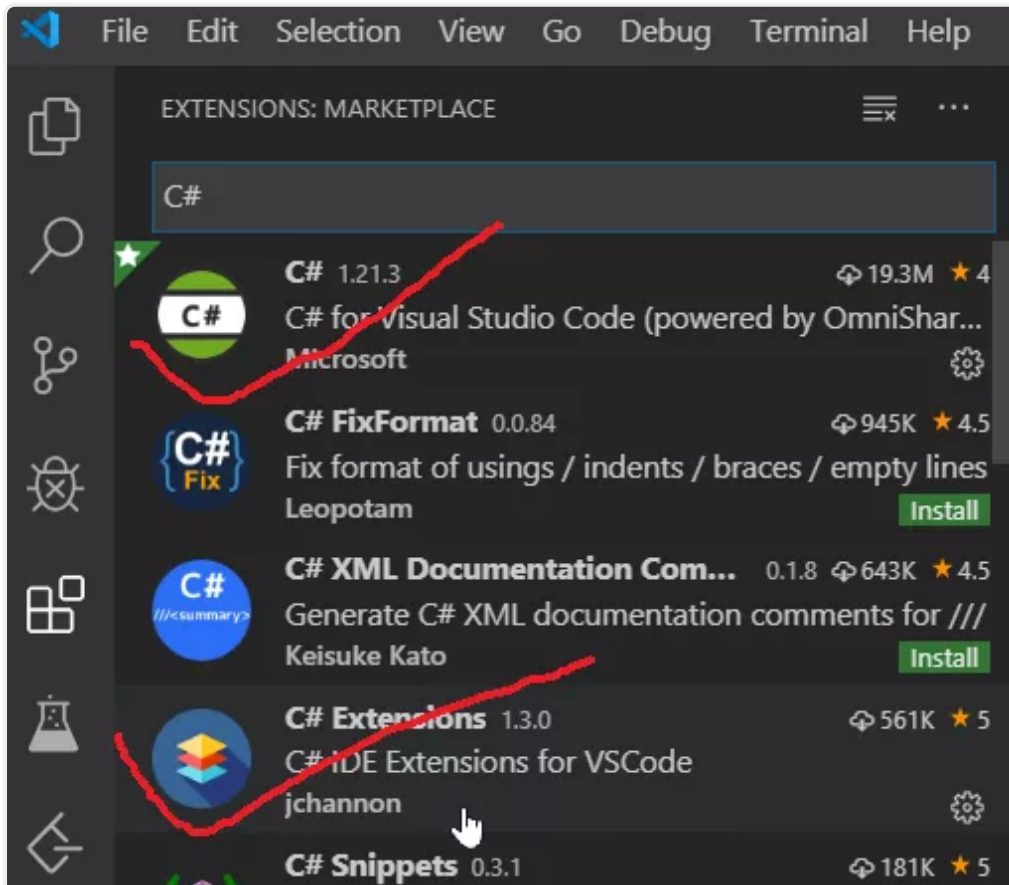
```
public static void Main()
{
    var sol = new Solution();
    int[] input = { 4, 1, 2, 1, 2 };

    System.Console.WriteLine(sol.SingleNumber(input));
}
}
```

选项2: VS Code本地Debug + 在LeetCode插件中验证和提交

安装C#相关插件

首先微软官方 OmniSharp 团队开发的 C# 插件是必须安装的，



另外再安装一下第3个插件，则对C#代码基本的智能感知，关键字高亮等功能就可以顺利使用了。

配置 .NET Core运行环境

先安装 .NET Core SDK 3.0，到官网 <https://dotnet.microsoft.com/download> 下载安装即可。

首先在 VS Code下方的Terminal窗口中依次输入如下命令：

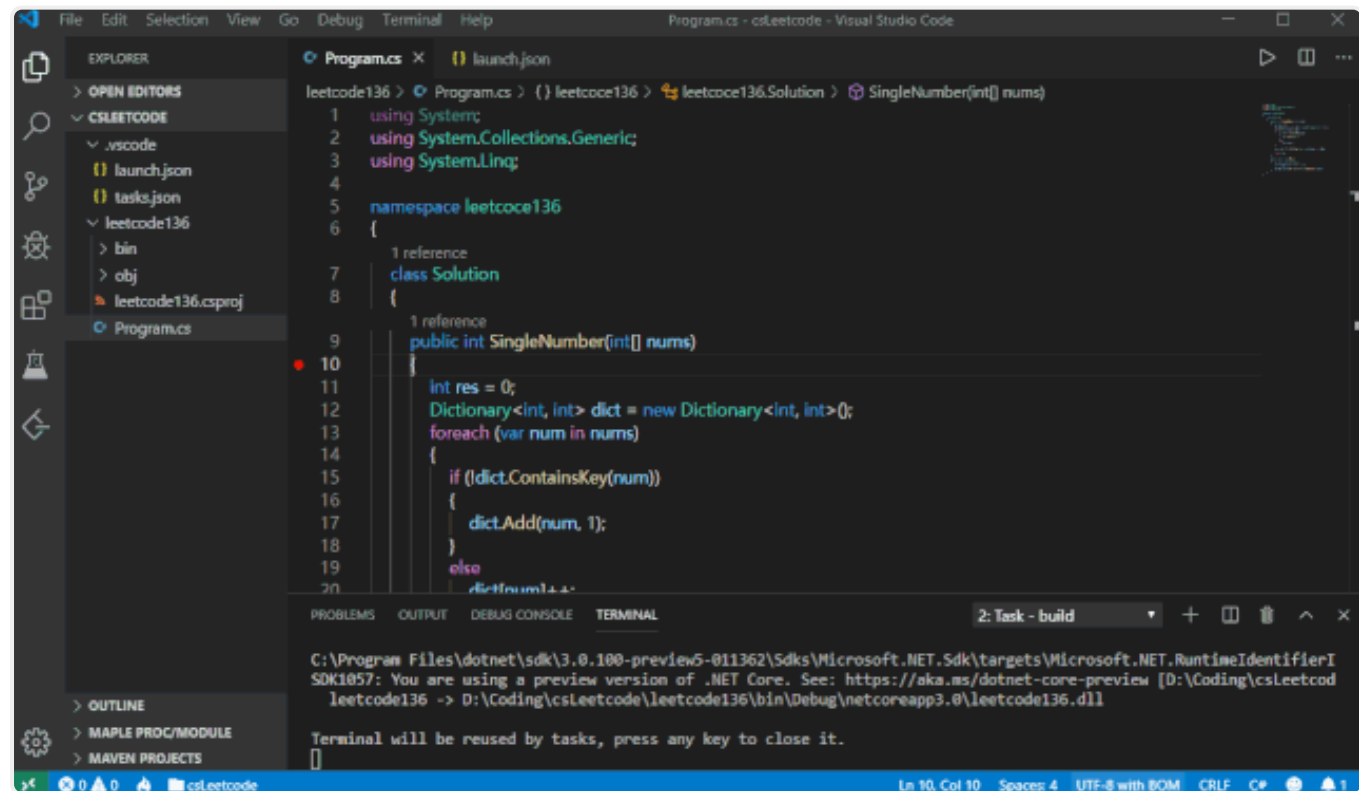
```
$ dotnet --version
3.0.100-preview5-011362

$ cd d:/Coding/csLeetcode
$ dotnet new console -o "leetcoce136"
```

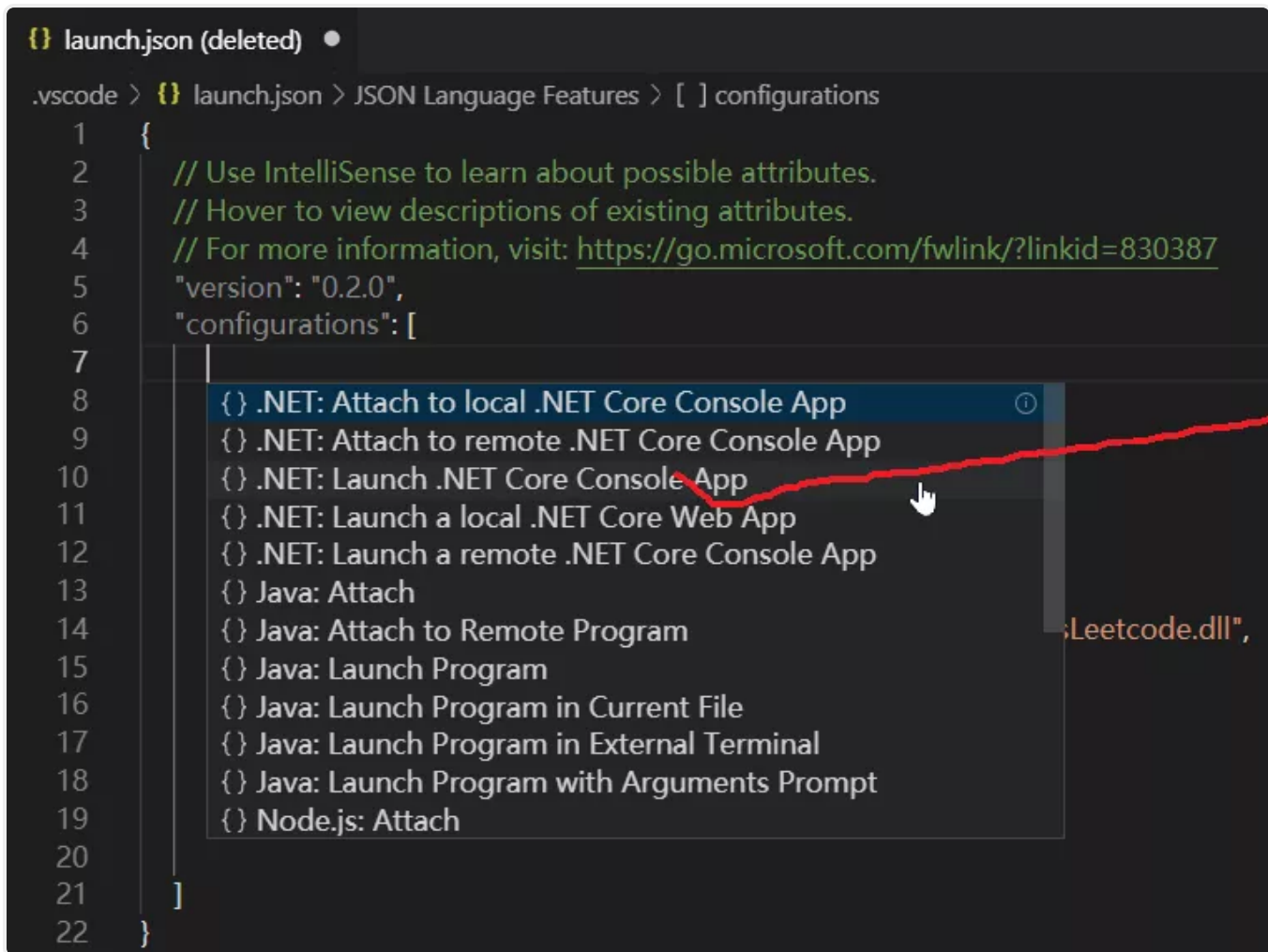
这里假设你以及有文件夹 `d:/Coding/csLeetcode`。

在VS Code中Debug C#

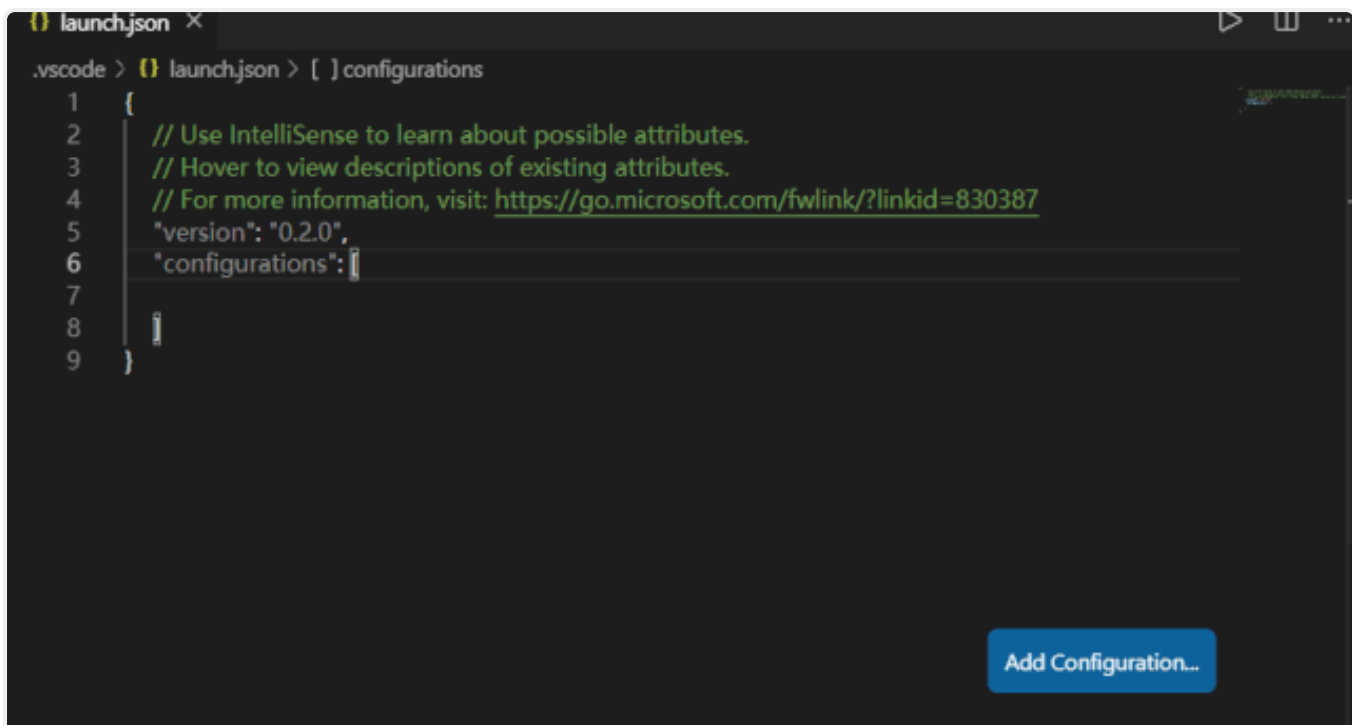
这里我们直接使用刚才带主函数的本地测试代码吧，将代码先拿过来，然后按 `F5`，选择 `.NETCore`，具体操作见下图：



在配置文件 `launch.json` 中，我们需要选择类型的是 `.NET:Launch.NETCoreConsoleApp`.



`launch.json` 的完整配置过程如下:

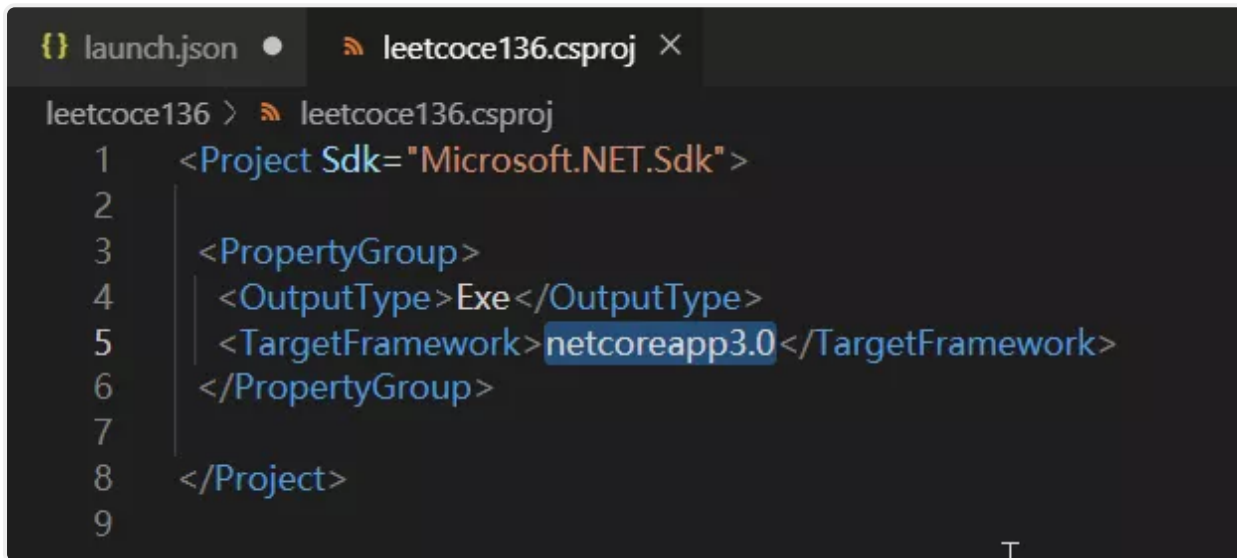


接下来我们需要修改其中的属性值 `program`:

默认的值为:

```
"program":"${workspaceFolder}/bin/Debug/<target-framework>/<project-name.dll>"
```

其中的 `<target-framework>` 是指目标运行环境，其具体版本可以在上面还原的项目文件 `leetcode136.csproj` 中看到：



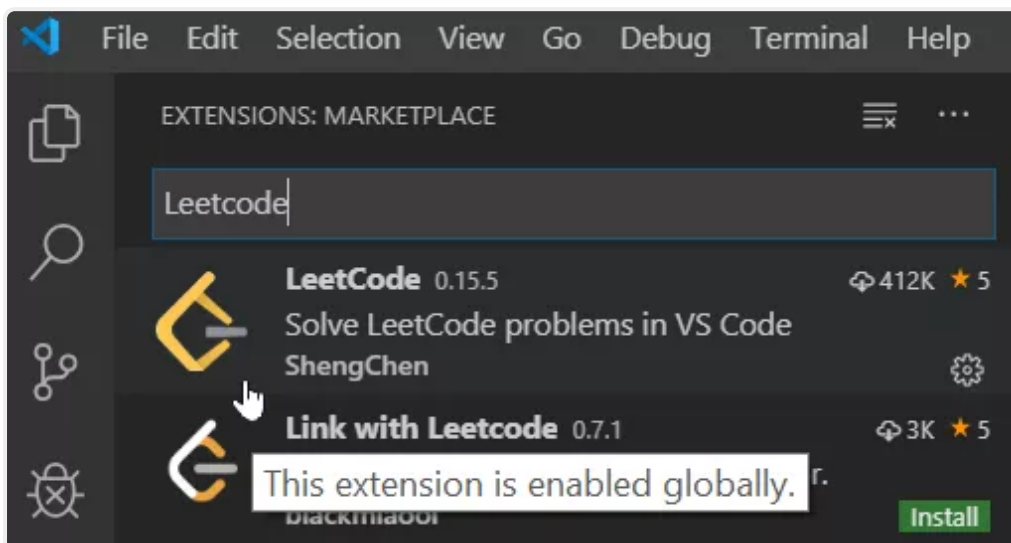
```
leetcode136 > leetcode136.csproj
1 <Project Sdk="Microsoft.NET.Sdk">
2
3   <PropertyGroup>
4     <OutputType>Exe</OutputType>
5     <TargetFramework>netcoreapp3.0</TargetFramework>
6   </PropertyGroup>
7
8 </Project>
9
```

而我们的项目名，即 `project-name` 是 `leetcode136`，于是 `program` 的具体值为：

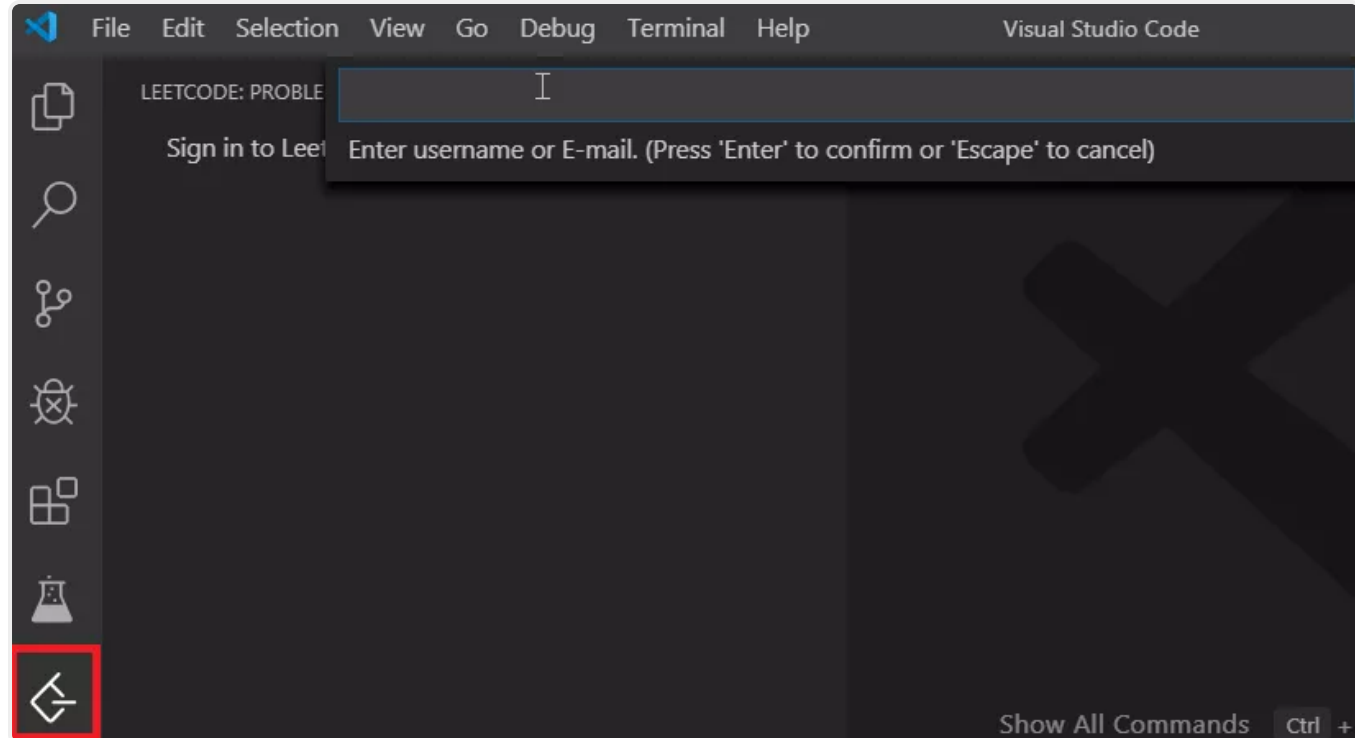
```
"${workspaceFolder}/bin/Debug/netcoreapp3.0/leetcode136.dll"
```

安装 LeetCode 插件

在扩展中搜索安装作者是 `ShengChen` 的 `LeetCode` 插件。



然后点击左下角的 LeetCode Logo 进行登录：

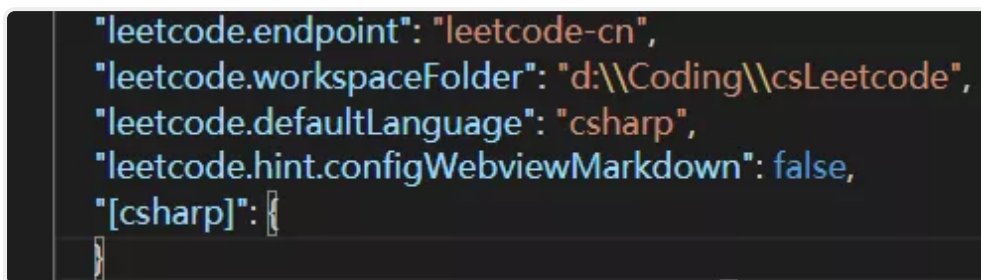


推荐选择 力扣leetcode-cn.com:

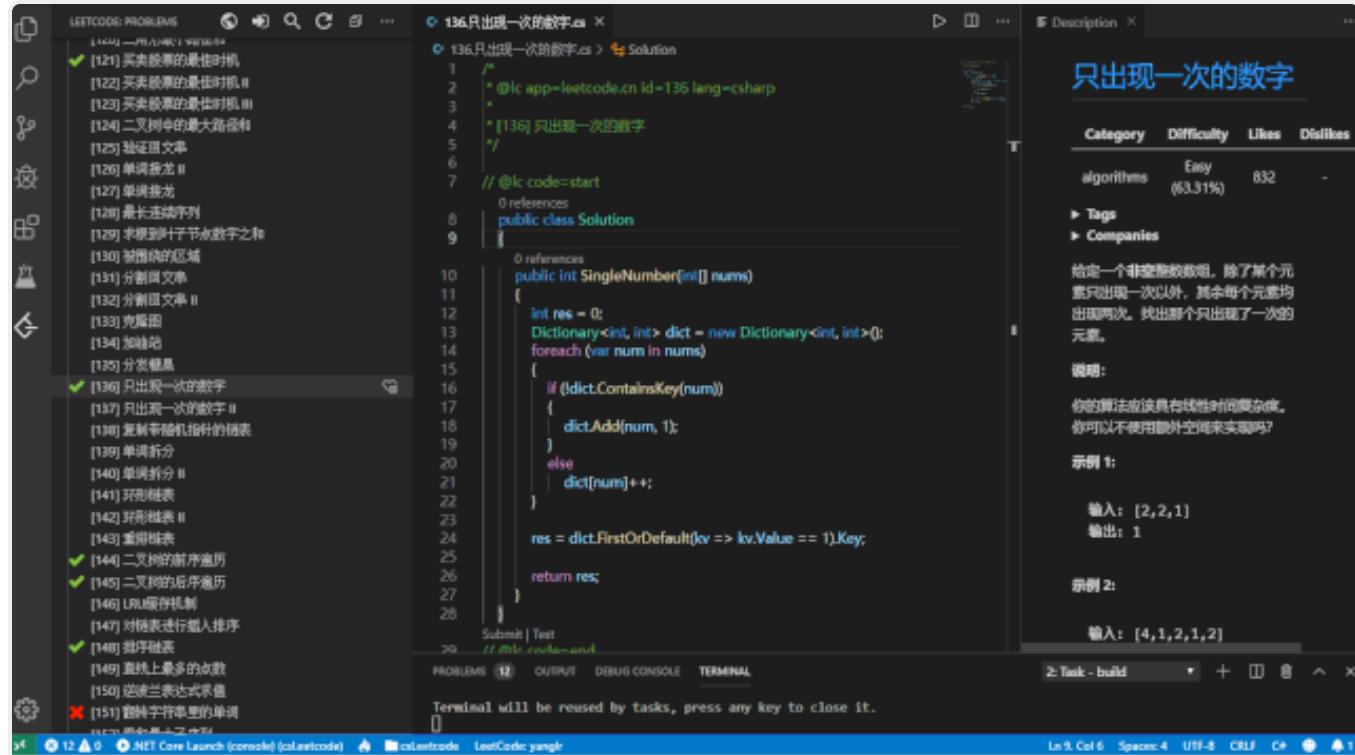


在国内使用，网络相对稳定。

然后在 LeetCode 插件的配置文档中将 `leetcode.defaultLanguage` 设置为 `csharp`。

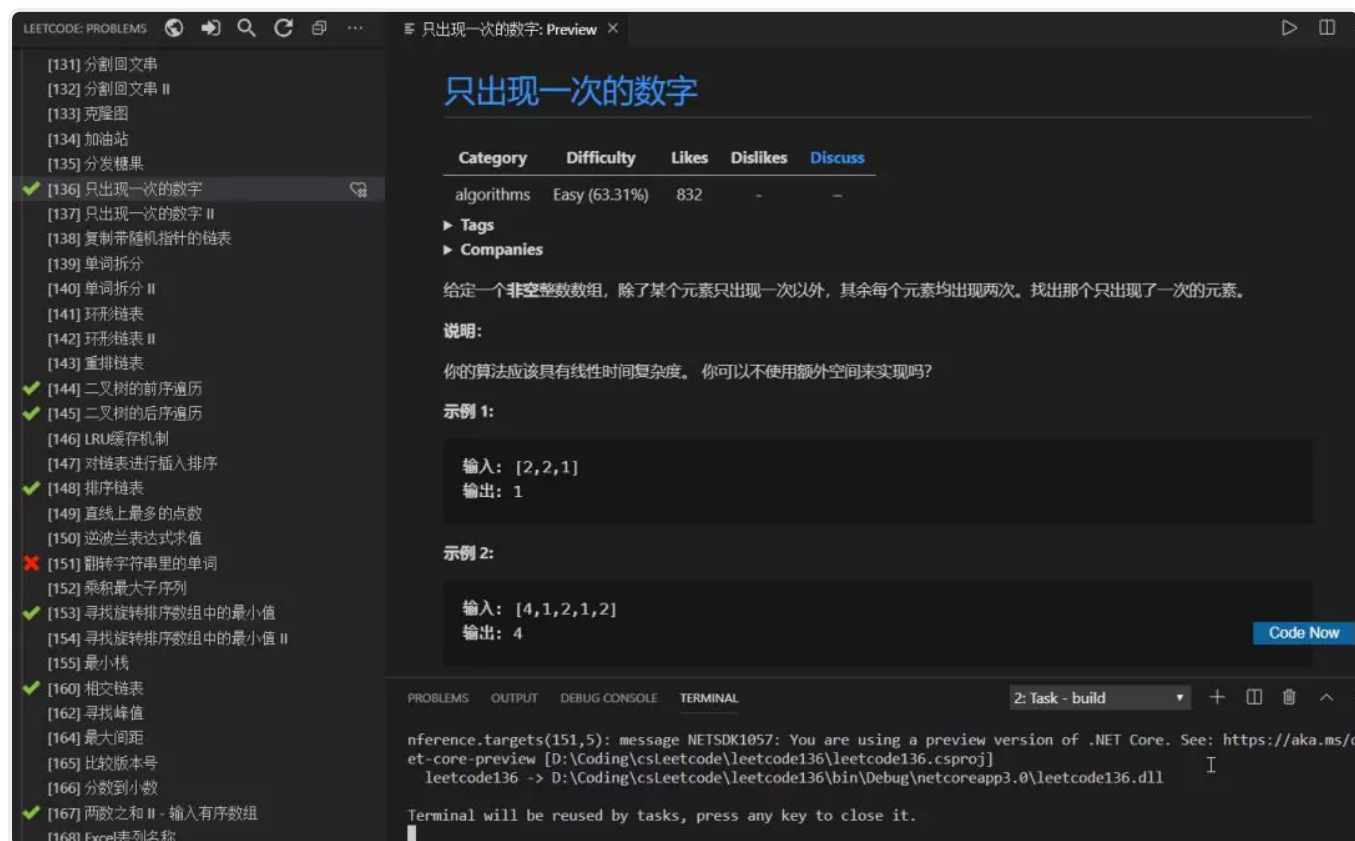


其完整操作过程如下:



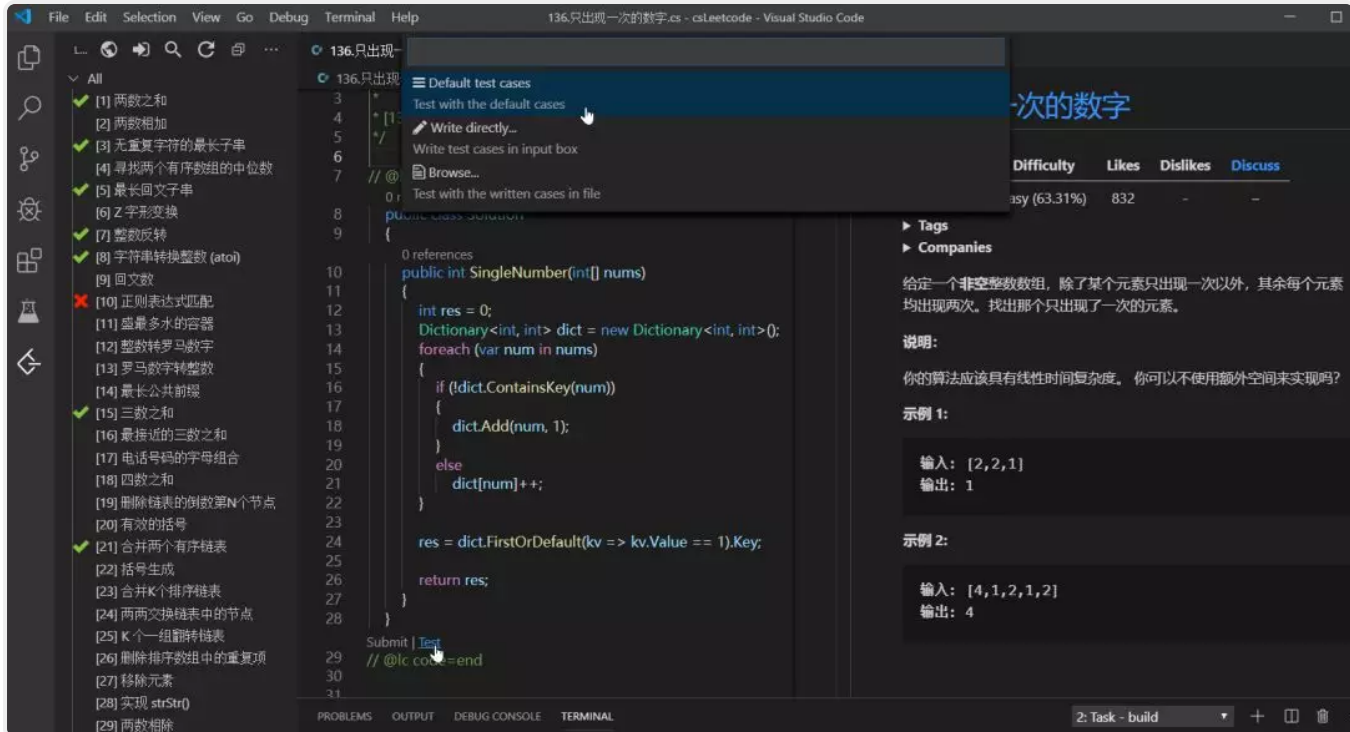
接下来就可以愉快地使用LeetCode刷题了。

我们在左侧题库列表中选择 **All**，找到刚才的例子No.136，双击问题标题，可以看到问题描述：



如果需要提交代码，只需点右下角的"Code Now"按钮。

当我们写完代码后，可以点击里面的代码下方的 **Test** 按钮进行测试，这便等价于网页版的按钮"执行代码"。



点 `Test` 后有图中3种选项，我一般是使用第一个。除非部分 Test Case 无法通过，才会使用第2个或第3个选项。

此时呢，如果对代码比较有信心，可以直接点"Test"按钮左侧的"Submit"按钮提交代码了。

关于VS Code的LeetCode 插件，文章 [LeetCode for VS Code: 春招 Offer 收割利器](#) 中有更详细的说明。

接下来的 **LeetCode面试题系列连载** 中每篇文章将会提供解题思路、算法复杂度的简要分析、已AC代码、提交的答案排名等，敬请期待。

参考资料:

.NET Core and Visual Studio Code

<https://code.visualstudio.com/docs/languages/dotnet>

使用VS Code 开发.NET CORE 程序指南

<https://www.cnblogs.com/xboo/p/11431222.html>

End

作者简介: *Bravo Yeung*, 计算机硕士, 知乎干货答主(获73K 赞同, 34K 感谢, 210K 收藏)。曾在国内 Top3 互联网视频直播公司短暂工作过, 后加入一家外企做软件开发至今。

欢迎在留言区留下你的观点，一起讨论提高。如果今天的文章让你有新的启发，学习能力的提升上有新的认识，欢迎转发分享给更多人。

欢迎各位读者加入 **.NET技术交流群**，在公众号后台回复“**加群**”或者“**学习**”即可。



- 微信后台回复“ebook”，给你：一份全网最强的电子书下载指南。
- 回复“运营工具箱”，给你：一整套运营工具，学会后极大地提到运营效率！



你点的每一个**"在看"**，
我都当成了喜欢！

