春节刷题计划（一）| 当Kotlin遇上LeetCode

2022-01-28 朱涛

《朱涛 · Kotlin编程第一课》 课程介绍 >

讲述：朱涛

时长 07:56　大小 7.28M

你好，我是朱涛。

时光飞逝，不知不觉间，我们就已经完成了基础篇的学习，并且也已经完成了三个实战项目，但这终归是不够过瘾的。想要完全掌握 Kotlin 的基础语法，我们还需要更多的练习。我相信，你现在的心情就像是一个手握屠龙刀的勇士，热切希望找一些对手来验证自己的学习成果。

其实，我自己在学习一门新的编程语言的时候，有一个高效的方法，也分享给你。这里我以 Kotlin 为例，假设我现在是一个新手，想快速掌握 Kotlin 的话，我会这样做：

- 第一步，我会**去 Google 搜索一些语言特性对比的文章**。比如，我熟悉 Java，想学 Kotlin，我就会去搜“from Java to Kotlin”，然后去看一些 Java、Kotlin 语法对比的文章。这时候，我大脑里就会建立起 Java 与 Kotlin 的语法联系。

- 第二步，我会打开Kotlin 官方文档，花几个小时的时间粗略看一遍，对 Kotlin 的语法有个大致印象。
- 最后一步，我会**打开 LeetCode 之类的网站**，开始用 Kotlin 刷题。在刷题的过程中，我也会先从模拟类的题目开始，之后再到数组、链表、Map、二叉树之类的数据结构。整个过程由易到难，刚开始的时候，我会选择“简单题”，等熟练以后，再选择“中等题”，心情好的时候，我偶尔会做个“困难题”挑战一下。

当然，对于你来说，第一步和第二步都已经不是问题了，通过前面十几节课程的学习，你已经有了牢固的 Kotlin 基础。现在欠缺的，只是大量的练习而已。

说回来，其实我认为，我这种学习编程的方法是个一举多得的，比如它可以让我们：

- **快速掌握一门新的编程语言**。
- **夯实基本功**。通过刷算法题，可以进一步巩固自己的数据结构与算法知识，这对于以后的工作也会有很大的帮助。所谓软件的架构，其实一定程度上就是在选择不同的数据结构与算法解决问题。而基本功的扎实程度，也决定了一名开发者的能力上限。
- **面试加分**。众所周知，顶级的 IT 公司面试的时候，都是要做算法题的。假如你是一名 Android 或 Java 工程师，如果你能用 Kotlin 写出漂亮的题解，那将会是大大加分项。

另外，由于语法的简洁性，你会发现，用 Kotlin 做算法题，**比 Java 要“爽”很多**。同样的一道题目，用 Java 你可能要写很多代码，但 Kotlin 却只需要简单的几行。

所以接下来的春节假期呢，我就会带你来一起刷题，希望你在假期放松休息、陪伴家人之余，也不要停下学习的脚步。好，那么今天，我们就先来看几个简单的题目，就当作是热身了。

热身 1：移除字符串当中的“元音字母”

这是 LeetCode 的 1119 号题。题意大致是这样的：程序的输入是一个字符串 s。题目要求我们移除当中的所有元音字母 a、e、i、o、u，然后返回。

这个问题，如果我们用 Java 来实现的话，大致会是这样的：

复制代码

```
public String removeVowels(String s) {
    StringBuilder builder = new StringBuilder();
```

```
    char[] array = s.toCharArray();

    for(char c: array) {
        // 不是元音字母，才会拼接
        if(c != 'a' && c != 'e' && c != 'i' && c !='o' && c != 'u') {
            builder.append(c);
        }
    }

    return builder.toString();
}
```

但如果是用 Kotlin，我们一行代码就可以搞定：

复制代码

```
fun removeVowels(s: String): String =
        s.filter { it !in setOf('a', 'e', 'i', 'o', 'u') }
```

这里，我们是使用了字符串的扩展函数 filter，轻松就实现了前面的功能。这个题目很简单，同时也比较极端，下面我们来看一个更复杂的例子。

热身 2：最常见的单词

这是 LeetCode 的 819 号题。题意大致如下：程序的输入是一段英语文本（paragraph），一个禁用单词列表（banned）返回出现次数最多、同时不在禁用列表中的单词。

这个题目其实跟我们第 2 次的实战项目“英语词频统计”有点类似，我们之前实现的是完整的单词频率，并且降序。这个题目只需要我们找到频率最高的单词，不过就是多了一个**单词黑名单**而已。

那么，这个题目如果我们用 Java 来实现，肯定是要不少代码的，但如果用 Kotlin，简单的几行代码就可以搞定了：

复制代码

```
fun mostCommonWord1(paragraph: String, banned: Array<String>) =
            paragraph.toLowerCase()
                .replace("[^a-zA-Z ]".toRegex(), " ")
                .split("\\s+".toRegex())
                .filter { it !in banned.toSet() }
                .groupBy { it }
```

```
                .mapValues { it.value.size }
                .maxBy { it.value }
                ?.key?:throw IllegalArgumentException()
```

可以看到，这段代码我们只需要在原有 TextProcessor 的基础上，做一点修改，就完成了。

另外，你可能也发现了，我们前面的两个例子都是用函数式思维来解决的，这其实是因为这两个问题用命令式会更复杂。而对于一些其他的问题，我们其实仍然可以选择命令式来解决。比如：手写排序算法。

热身 3：用 Kotlin 实现冒泡排序

冒泡排序，是计算机里最基础的一种排序算法。如果你忘了它的实现方式，也没关系，我做了一个动图，让你可以清晰地看到算法的执行过程。

那么针对冒泡排序法，如果我们用 Kotlin 来实现，命令式的方式会更加直观一些，就像下面这样：

复制代码

```
fun sort(array: IntArray): IntArray {
    for (end in (array.size - 1) downTo 1) {
        for (begin in 1..end) {
            if (array[begin - 1] > array[begin]) {
```

```
                val temp = array[begin - 1]
                array[begin - 1] = array[begin]
                array[begin] = temp
            }
        }
    }
    return array
}
```

这里，我们需要格外注意的是，在逆序遍历数组的时候，我们是使用了**逆序**的 Range：“(array.size - 1) downTo 1”，而如果这里是用“1…(array.size - 1)”的话，其实是会出问题的。因为 Kotlin 当中的 Range 要求必须是右边不能小于左边，比如“1…3”是可以的，而“3…1”是不行的。

好了，到这里，相信你对用 Kotlin 刷算法题已经有了一定认识了。正如 Kotlin 官方所宣传的那样，Kotlin 是一门多范式的编程语言，对于不同的问题，我们完全可以选择不同范式来进行编程。说到底就是：**怎么爽就怎么来**。

小作业

好，最后也再给你留一个小作业，请你用 Kotlin 来完成LeetCode 的 165 号题《版本号判断》。

注意：LeetCode 中文站使用的 Kotlin 版本，仍然停留在 1.3.10。如果你是使用 Kotlin 1.6 解题，代码在 IDE 当中编译通过了，而 LeetCode 显示编译出错，那么你就需要修改一下对应的实现。或者，你也可以将新版本的库函数一起拷贝到 Solution 当中去。

这道题目我会在下节课给出答案解析，我们下节课再见。

上一篇 加餐四 | 什么是“空安全思维”？

下一篇 春节刷题计划（二）| 一题三解，搞定版本号判断

精选留言 (10)

写留言

爱学习的小羊

2022-03-21

我这个算是半java半kotlin编程了吧

```
fun compareVersion(version1: String, version2: String): Int {
        val nums1 = version1.split(".")
        val nums2 = version2.split(".")
        for (i in 0..maxOf(nums1.size,nums2.size)){
            var a = 0
            var b = 0
            if (i < nums1.size) a = nums1[i].toInt()
            if (i<nums2.size) b = nums2[i].toInt()
            val data = a - b
            when {
                data > 0 -> return 1
                data < 0 -> return -1
            }

        }
        return 0
    }
```

作者回复: 这样也挺好看的

梁中华

2022-03-19

```
fun mostCommonWord1(paragraph: String, banned: Array<String>) =
        paragraph.toLowerCase()
            .replace("[^a-zA-Z ]".toRegex(), " ")
```

```
            .split("\\s+".toRegex())
            .filter { it !in banned.toSet() }
            .groupBy { it }
            .mapValues { it.value.size }
           // .maxBy { it.value }  //这里编译不过，我改了下
           // ?.key?:throw IllegalArgumentException()
        .toList()   //先转成List才能用MaxBy
       .maxByOrNull { it.second }
```

作者回复: 嗯，因为新版本里隐藏了，LeetCode 能用。


PoPlus

2022-03-04

涛哥，校招生还推荐用 kotlin 来写算法吗，感觉有些简化的太过了 😄

作者回复: 放心，校招考察的是思维，哪个语言其实没那么重要，语言跟岗位相关会更好，关联不大也不会有太大问题。只要别过渡依赖API即可。

举个例子，面试官让你实现：“字符串匹配算法”，这时候我们就不能用contains、indexOf之类的API，而是要老老实实去实现KMP之类的算法。让面试官看到你的算法功底、解题思路，这就够了。


浅色的风

2022-03-02

是不是java思维

```
fun compareVersion(version1: String, version2: String): Int {
        val listV1 = version1.split(".").toList()
        val listV2 = version2.split(".").toList()

        val result = if (listV1.size > listV2.size) 1 else -1

        val maxList = if (listV1.size > listV2.size) listV1 else listV2
        val minList = if (listV1.size <= listV2.size) listV1 else listV2

        for(i in 0..minList.size - 1){
            if(listV1[i].toInt() > listV2[i].toInt()){
                return 1
            }else if(listV1[i].toInt() < listV2[i].toInt()){
```

```
            return -1
        }
    }

    for(j in minList.size..maxList.size - 1){
        if(maxList[j].toInt() > 0){
            return result
        }
    }

    return 0

}
```

作者回复: 这思路也不差。


白乾涛

2022-03-02

我知道为啥找不到 maxBy 了，估计是因为这个方法在 1.6 版本中隐藏了

```
@DeprecatedSinceKotlin(warningSince = "1.4", errorSince = "1.5", hiddenSince = "1.6")
public inline fun <T, R : Comparable<R>> Sequence<T>.maxBy(selector: (T) -> R): T? {
    return maxByOrNull(selector)
}
```

作者回复: 是的，有的时候，找不到API了也有可能是废弃了。

共 2 条评论 >

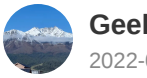
Geek_Adr

2022-02-07

```
// 我能找到的最大程度的函数式
    fun compareVersion(version1: String, version2: String): Int {
        return (version1.split(".").map { it.toInt() }
                to version2.split(".").map { it.toInt() })
            .run {
                (0 until maxOf(first.size, second.size))
                    .fold(0) { acc, i ->
                        if (acc != 0) acc
```

```
                else first.getOrElse(i) { 0 }.compareTo(second.getOrElse(i) { 0 })
            }
        }
    }
```

作者回复: 思路很不错，这种方式不必定义额外的函数，大家可以参考。



Geek_Adr
2022-02-07

```
// 函数式 176ms 击败19%
    fun compareVersion(version1: String, version2: String): Int {
        return (version1.split(".").map { it.toInt() }
                to version2.split(".").map { it.toInt() })
            .run {
                (0 until max(first.size, second.size))
                    .fold(0) { acc, i ->
                        if (acc != 0) acc
                        else first.getOrElse(i) { 0 } - second.getOrElse(i) { 0 }
                    }
            }.let {
                when {
                    it > 0 -> 1
                    it < 0 -> -1
                    else -> 0
                }
            }
    }

// java  156ms 击败70%
    fun compareVersion(version1: String, version2: String): Int {
        val v1 = version1.split(".").map { it.toInt() }
        val v2 = version2.split(".").map { it.toInt() }
        var idx = 0
        while (idx < v1.size && idx < v2.size) {
            if (v1[idx] > v2[idx]) {
                return 1
            } else if (v1[idx] < v2[idx]) {
                return -1
            }
```

```
            idx++
        }
        while (idx < v1.size) {
            if (v1[idx++] > 0) {
                return 1
            }
        }
        while (idx < v2.size) {
            if (v2[idx++] > 0) {
                return -1
            }
        }
        return 0
    }
```

作者回复: 这个题解也非常值得参考。

再次感谢这位同学。


郑峰

2022-01-30

```Kotlin
fun compareVersion(version1: String, version2: String): Int {
  val v1 = version1.split(".").map { it.toInt() }
  val v2 = version2.split(".").map { it.toInt() }

  for (i in 0 until maxOf(v1.size, v2.size)) {
    val diff = (v1.getOrElse(i) { 0 } - v2.getOrElse(i) { 0 })
    if (diff != 0) return if (diff > 0) 1 else -1
  }
  return 0;
}

```

作者回复: 简洁、干净、优雅！


$Kotlin

2022-01-28

```
fun compareVersion(version1: String, version2: String): Int {
    val versionList1 = version1.split(".").toMutableList()
    val versionList2 = version2.split(".").toMutableList()
    if (versionList1.size > versionList2.size) {
        repeat(versionList1.size - versionList2.size) {
            versionList2.add("0")
        }
    } else {
        repeat(versionList2.size - versionList1.size) {
            versionList1.add("0")
        }
    }
    for (index in 0..versionList1.size-1) {
        val v1Int = versionList1[index].toInt()
        val v2Int = versionList2[index].toInt()
        if (v1Int > v2Int) {
            return 1
        } else if (v1Int < v2Int) {
            return -1
        }
    }
    return 0
}
```

作者回复: 代码写的不错，思路很清晰，能加上一些代码注释就更好了。

jim

2022-01-28

春节还更新吗？

编辑回复: 春节期间老师会带大家刷题哈，我知道大家都很着急学协程，但是心急吃不了热豆腐嘛，先跟着老师一起练习和复习，把基础夯实好了，春节后协程就到啦，这门课又跑不了~