30、比较两个版本号的大小，考点 or 实现——逻辑分析

题目描述

输入两个版本号 version1 和 version2，每个版本号由多个子版本号组成。

子版本号之间由“.”隔开，由大小写字母、数字组成，并且至少有一个字符。

按从左到右的顺序比较子版本号，比较规则如下：

- 子版本号前面的0不参与比较，比如 001 和 1 是相等的。
- 小写字母 > 大写字母 > 数字
- 空字符和0相等，比如 1 和 1.0 相等

比较结果

如果 version1 > version2 ，返回 1

如果 version1 < version2 ，返回-1

其他情况返回0

输入描述

第一行输入version1

第二行输入version2

输出描述

输出version1和version2的比较结果

用例

输入	5.2 5.1a
输出	1
说明	无

输入	5.6.1 5.6.2a
输出	-1
说明	无

输入	5.6.8.a 5.6.8.0a
输出	0
说明	无

题目解析

我的解题思路如下：

首先把版本号的每个子版本的前导0去掉，这里我用的是正则表达式 /^0+/ 去匹配前导0，并替换为''。

需要注意的是，如果子版本就是0或者由多个0组成，则按上面替换逻辑，会得到一个空串子版本，为了避免这种情况，在替换后，我们需要判断子版本是否为空串，如果为空串，则给一个'0'。

去掉每个子版本的前导0后，我们需要求出两个版本号的子版本个数，将最多的个数赋值给len。

然后for循环遍历两个版本号0~len-1序号的子版本，

注意，如果某个版本号不存在对应序号子版本，则默认取0作为子版本。

然后比较两个版本号的同序号子版本，比较规则是：小写字母 > 大写字母 > 数字

由于这里的数字就是数字字符串，因此该比较规则就是 字典序 。

```
> '111111' < 'A'
< true
> 'Z'< 'a'
< true
```

我们直接用比较运算符比较两个子版本字符串即可。

2023.04.29 根据网友指正，本题输入的两个版本，如果对应序号的子版本，都是纯数字的话，此时不应该按照字典序来比较，比如下面例子：

```
10
6
```

很明显的，10版本要比6版本大，但是由于采用字典序比较，因此10会比6小。

因此，当对应序号子版本都是纯数字字符串时，我们应该进行数值比较。而只要对应序号的子版本有一个是非纯数字，则按照字典序比较。

JavaScript算法源码

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const lines = [];
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 2) {
    const v1 = lines[0];
    const v2 = lines[1];

    console.log(getResutlt(v1, v2));
    lines.length = 0;
  }
});

function getResutlt(v1, v2) {
  const arr1 = convert(v1);
  const arr2 = convert(v2);
```

```
  const isDigit = /^\d+$/;

  const len = Math.max(arr1.length, arr2.length);

  for (let i = 0; i < len; i++) {
    let a = arr1[i] || 0;
    let b = arr2[i] || 0;

    if (isDigit.test(a) && isDigit.test(b)) {
      a = Number(a);
      b = Number(b);
    }

    if (a > b) return 1;
    else if (a < b) return -1;
  }

  return 0;
}

function convert(version) {
  return version.split(".").map((str) => {
    str = str.replace(/^0+/, "");
    return str == "" ? "0" : str;
  });
}
```

Java算法源码

```java
import java.util.Arrays;
import java.util.Scanner;

public class Main {
  // 输入获取
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    String v1 = sc.next();
    String v2 = sc.next();

    System.out.println(getResult(v1, v2));
  }

  // 算法入口
  public static int getResult(String v1, String v2) {
    String[] arr1 = convert(v1);
    String[] arr2 = convert(v2);

    int n = Math.max(arr1.length, arr2.length);

    for (int i = 0; i < n; i++) {
      String tmp1 = arr1.length > i ? arr1[i] : "0";
      String tmp2 = arr2.length > i ? arr2[i] : "0";

      try {
        int i1 = Integer.parseInt(tmp1);
        int i2 = Integer.parseInt(tmp2);
        if (i1 != i2) return i1 > i2 ? 1 : -1;
      } catch (Exception e) {
        int res = tmp1.compareTo(tmp2);
        if (res != 0) return res > 0 ? 1 : -1;
      }
    }

    return 0;
  }

  public static String[] convert(String version) {
    // 注意split方法入参会被当成正则，因此这里不能直接使用"."，因为"."在正则中是元字符，有特殊含义，我们应该使用转义后的"\\."
    return Arrays.stream(version.split("\\."))
        .map(
            sub -> {
              String s = sub.replaceAll("^0+", ""); // 去除前导0
              return "".equals(s) ? "0" : s; // 如果是"0"，去除前导0后就变为了""，需要做特殊处理
            })
        .toArray(String[]::new);
  }
}
```

Python算法源码

```python
# 输入获取
v1 = input()
v2 = input()


# 去除前导0
def rmLeadZero(x):
    tmp = x.lstrip("0")
    return "0" if tmp == "" else tmp


# 将大版本按"."切割为子版本列表
def convert(version):
    return list(map(rmLeadZero, version.split(".")))


# 算法入口
def getResult():
    arr1 = convert(v1)
    arr2 = convert(v2)

    n = max(len(arr1), len(arr2))

    for i in range(n):
        # 空字符和0相等，比如 1 和 1.0 相等
        tmp1 = arr1[i] if len(arr1) > i else "0"
        tmp2 = arr2[i] if len(arr2) > i else "0"

        if tmp1.isdigit() and tmp2.isdigit():
            tmp1 = int(tmp1)
            tmp2 = int(tmp2)

        if tmp1 > tmp2:
            return 1
        elif tmp1 < tmp2:
            return -1

    return 0


# 算法调用
print(getResult())
```