思想：算法就是每次对应取出相同位置的小数点之前所有的字符，把他们转换为数字比较，若不同则可直接得到答案，若相同，再对应往下取，如果一个数字已经没有小数点了，则默认取出0，和另一个比较，这样也解决了末尾无效0的情况。

165. Compare Version Numbers

Description | Hints | Submissions | Discuss | Solution

Pick One

Compare two version numbers *version1* and *version2*.
If `version1 > version2` return `1;` if `version1 < version2` return `-1;` otherwise return `0`.

You may assume that the version strings are non-empty and contain only digits and the `.` character.
The `.` character does not represent a decimal point and is used to separate number sequences.
For instance, `2.5` is not "two and a half" or "half way to version three", it is the fifth second-level revision of the second first-level revision.

Example 1:

```
Input: version1 = "0.1", version2 = "1.1"
Output: -1
```

Example 2:

```
Input: version1 = "1.0.1", version2 = "1"
Output: 1
```

Example 3:

```
Input: version1 = "7.5.2.4", version2 = "7.5.3"
Output: -1
```

```java
public class L165 {
       public int compareVersion(String version1, String version2) {
           String [] arr1 = version1.split("\\.");
           String [] arr2 = version2.split("\\.");

           int i = 0;
           while (i < arr1.length || i < arr2.length) {
                if(i < arr1.length && i < arr2.length) {
                    if (Integer.parseInt(arr1[i]) < Integer.parseInt(arr2[i])) {
                        return -1;
                    }else if (Integer.parseInt(arr1[i]) > Integer.parseInt(arr2[i])) {
                        return 1;
                    }
                }else if (i < arr1.length) {
                    if(Integer.parseInt(arr1[i]) != 0)
                    return 1;
                }else if (i < arr2.length) {
                    if(Integer.parseInt(arr2[i]) != 0)
                        return -1;
                }
                i++;
        }
          return 0;
       }
}
```