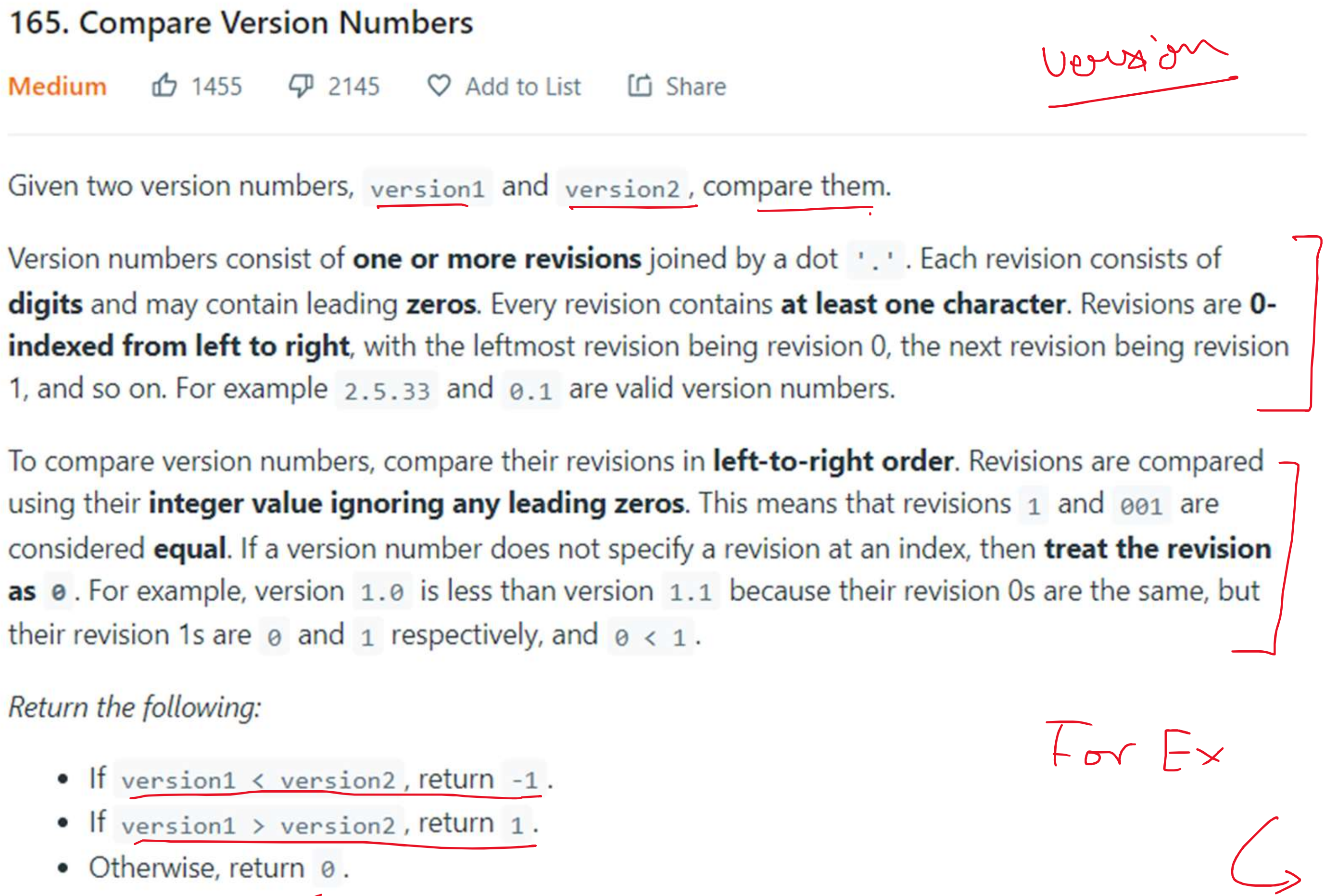
165. Compare Version Numbers

https://leetcode.com/problems/compare-version-numbers/

February 25, 2022

-Priyanshu Arya



165. Compare Version Numbers

Given two version numbers, `version1` and `version2`, compare them.

Version numbers consist of **one or more revisions** joined by a dot `'.'`. Each revision consists of **digits** and may contain leading **zeros**. Every revision contains **at least one character**. Revisions are **0-indexed from left to right**, with the leftmost revision being revision 0, the next revision being revision 1, and so on. For example `2.5.33` and `0.1` are valid version numbers.

To compare version numbers, compare their revisions in **left-to-right order**. Revisions are compared using their **integer value ignoring any leading zeros**. This means that revisions `1` and `001` are considered **equal**. If a version number does not specify a revision at an index, then **treat the revision as `0`**. For example, version `1.0` is less than version `1.1` because their revision 0s are the same, but their revision 1s are `0` and `1` respectively, and `0 < 1`.

Return the following:

- If `version1 < version2`, return `-1`.
- If `version1 > version2`, return `1`.
- Otherwise, return `0`. ✓

Version

2.5.33 ← ✓

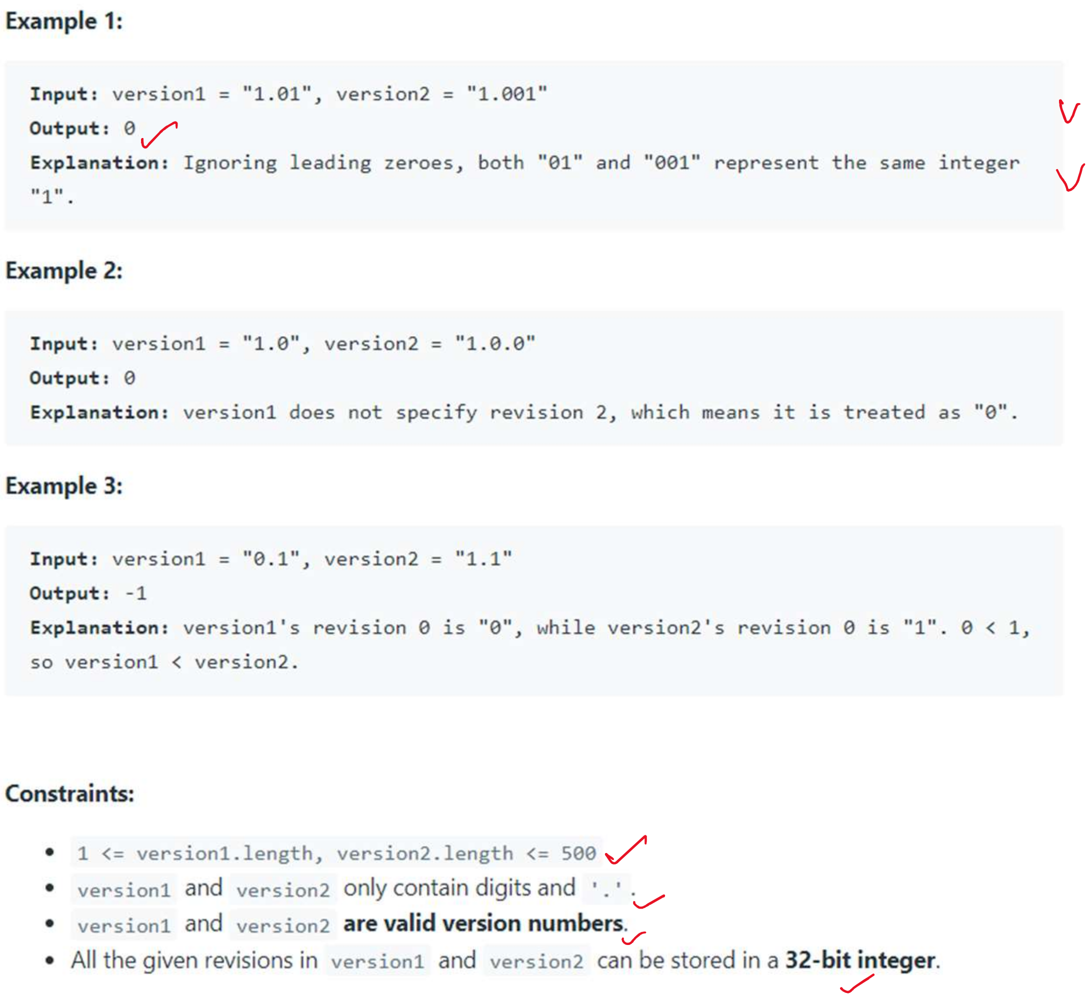
0.1 ← ✓

If after . starts with 0 then it will get ignored

For Ex

2.05.0033

↳ 2.5.33



Example 1:

```
Input: version1 = "1.01", version2 = "1.001"
Output: 0 ✓
Explanation: Ignoring leading zeroes, both "01" and "001" represent the same integer
"1".
```

v1=1.01 , 1.1 > return 0
v2=1.001 , 1.1

Example 2:

```
Input: version1 = "1.0", version2 = "1.0.0"
Output: 0
Explanation: version1 does not specify revision 2, which means it is treated as "0".
```

1.0 → 1 > return 0
1.0.0 → 1

Example 3:

```
Input: version1 = "0.1", version2 = "1.1"
Output: -1
Explanation: version1's revision 0 is "0", while version2's revision 0 is "1". 0 < 1,
so version1 < version2.
```

0.1 → 0.1 > return -1
1.1 → 1.1

Constraints:

- `1 <= version1.length, version2.length <= 500` ✓
- `version1` and `version2` only contain digits and `'.'`. ✓
- `version1` and `version2` **are valid version numbers**. ✓
- All the given revisions in `version1` and `version2` can be stored in a **32-bit integer**. ✓

Intution

→ Split version from '.' & store in List

ver1 = 1.2.03.4

1	2	03	4

ver2 = 3.2.5.04.6

3	2	5	04	6

→ Iterate in arr (max on both array)

Compare 1 or 3   return -1 ✓

→ Compare all indexes if found greater any of them return according

Algorithm

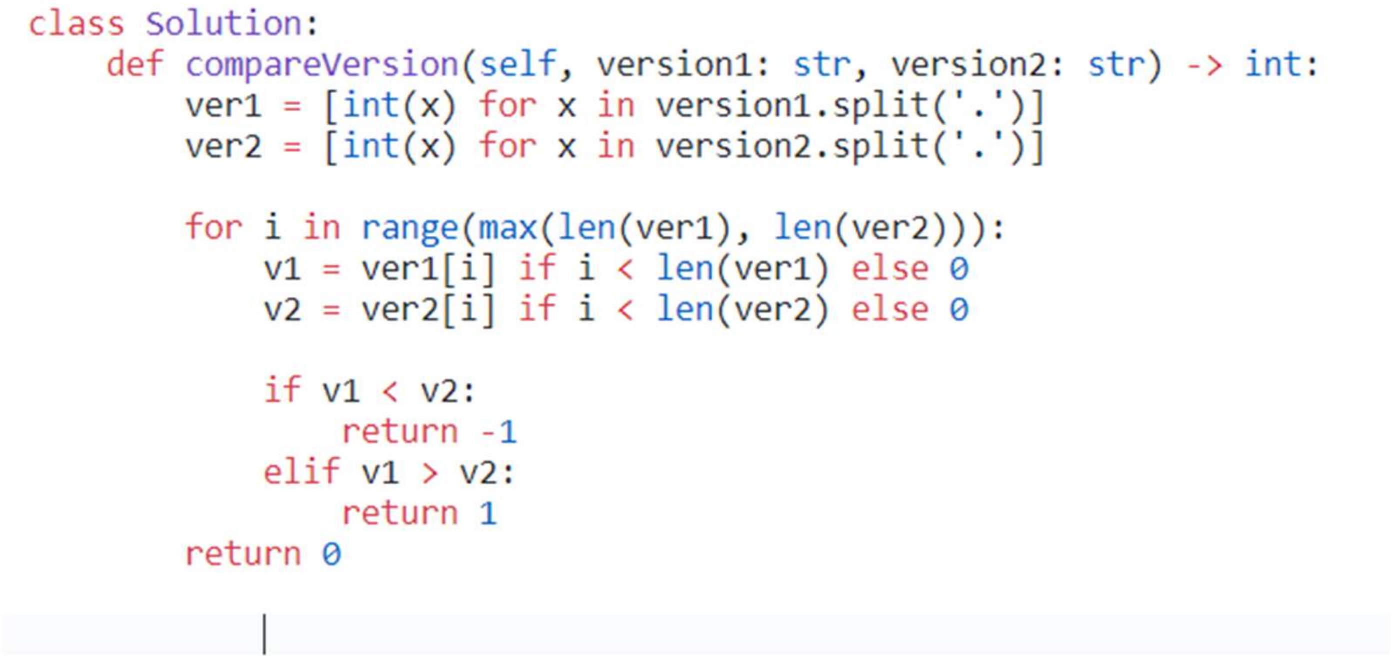
```
CompareVersion(version1, version2):

    ver1 = [int(x) for x in version1.split('.')]
    ver2 = [int(x) for x in version2.split('.')]

    for i in range(max(len(ver1)), len(ver2)):
        if i < len(ver1):
            v1 = ver1[i]
        else:
            v1 = 0
        if i < len(ver2):
            v2 = ver2[i]
        else
            v2 = 0
        if v1 > v2:
            return 1
        if v1 < v2:
            return -1
    return 0
```

Time Complexity : $O(n)$

n: max dof in version1 or version2

Space Complexity : $O(\underline{2}n) \rightarrow O(n)$

we made 2 List for storing splitted elements



```python
class Solution:
    def compareVersion(self, version1: str, version2: str) -> int:
        ver1 = [int(x) for x in version1.split('.')]
        ver2 = [int(x) for x in version2.split('.')]

        for i in range(max(len(ver1), len(ver2))):
            v1 = ver1[i] if i < len(ver1) else 0
            v2 = ver2[i] if i < len(ver2) else 0

            if v1 < v2:
                return -1
            elif v1 > v2:
                return 1
        return 0
```

Thank you

If you like Please share this and feel free to connect for any queries.
GitHub: https://github.com/priyanshu-arya/DSA/tree/master/Leetcode%201
Discord: https://discord.gg/qPer56TP
Mail: priyanshuarya2482000@gmail.com