# Reconstruct Original Digits from English

Given a **non-empty** string containing an out-of-order English representation of digits `0-9`, output the digits in ascending order.

**Note:**

1. Input contains only lowercase English letters.
2. Input is guaranteed to be valid and can be transformed to its original digits. That means invalid inputs such as "abc" or "zerone" are not permitted.
3. Input length is less than 50,000.

**Example 1:**

```
Input: "owoztneoer"

Output: "012"
```

**Example 2:**

```
Input: "fviefuro"

Output: "45"
```

## Solution 1

The idea is:

for zero, it's the only word has letter 'z',
for two, it's the only word has letter 'w',
......
so we only need to count the unique letter of each word, Coz the input is always valid.

Code:

```java
public String originalDigits(String s) {
    int[] count = new int[10];
    for (int i = 0; i < s.length(); i++){
        char c = s.charAt(i);
        if (c == 'z') count[0]++;
        if (c == 'w') count[2]++;
        if (c == 'x') count[6]++;
        if (c == 's') count[7]++; //7-6
        if (c == 'g') count[8]++;
        if (c == 'u') count[4]++;
        if (c == 'f') count[5]++; //5-4
        if (c == 'h') count[3]++; //3-8
        if (c == 'i') count[9]++; //9-8-5-6
        if (c == 'o') count[1]++; //1-0-2-4
    }
    count[7] -= count[6];
    count[5] -= count[4];
    count[3] -= count[8];
    count[9] = count[9] - count[8] - count[5] - count[6];
    count[1] = count[1] - count[0] - count[2] - count[4];
    StringBuilder sb = new StringBuilder();
    for (int i = 0; i <= 9; i++){
        for (int j = 0; j < count[i]; j++){
            sb.append(i);
        }
    }
    return sb.toString();
}
```

written by markieff original link here

## Solution 2

```
public class Solution {
    public String originalDigits(String s) {
        if(s==null || s.length()==0) return "";
        int[] count = new int[128];
        for(int i=0;i<s.length();i++)  count[s.charAt(i)]++;
        int[] num = new int[10];
        num[0] = count['z'];
        num[2] = count['w'];
        num[6] = count['x'];
        num[8] = count['g'];
        num[7] = count['s']-count['x'];
        num[5] = count['v']-count['s']+count['x'];
        num[4] = count['u'];
        num[3] = count['h']-count['g'];
        num[1] = count['o']-count['z']-count['w']-count['u'];
        num[9] = count['i']-count['x']-count['g']-count['v']+count['s']-count['x'
];
        String ret = new String();
        for(int i=0;i<10;i++)
            for(int j=num[i];j>0;j--) ret += String.valueOf(i);
        return ret;
    }
}
```

written by YuTingLiu original link here

## Solution 3

The **even** digits all have a unique letter while the**odd** digits all don't:

`zero`: Only digit with `z`
`two`: Only digit with `w`
`four`: Only digit with `u`
`six`: Only digit with `x`
`eight`: Only digit with `g`

The odd ones for easy looking, each one's letters all also appear in other digit words: `one`, `three`, `five`, `seven`, `nine`

written by StefanPochmann original link here