# 题目

给你一个字符串word ，该字符串由数字和小写英文字母组成。

请你用空格替换每个不是数字的字符。例如，"a123bc34d8ef34"将会变成" 123  34 8  34"。注意，剩下的这些整数为（相邻彼此至少有一个空格隔开）："123"、"34"、"8" 和 "34"。

返回对word完成替换后形成的 不同 整数的数目。

只有当两个整数的不含前导零的十进制表示不同， 才认为这两个整数也不同。

示例 1：

输入：word = "a123bc34d8ef34"

输出：3

解释：不同的整数有 "123"、"34" 和 "8" 。注意，"34" 只计数一次。

示例 2：

输入：word = "leet1234code234"

输出：2

示例 3：

输入：word = "a1b01c001"

输出：1

解释："1"、"01" 和 "001" 视为同一个整数的十进制表示，因为在比较十进制值时会忽略前导零的存在。

提示：

1 <= word.length <= 1000

word 由数字和小写英文字母组成

# 分析

## 方法一：哈希表

**思路：**

通过遍历字符串的方式来提取数字，并使用一个集合来存储不同的整数。

代码：

class Solution {

public:

int numDifferentIntegers(string word) {

unordered\_set<string> integers; // 用于存储不同的整数

int i = 0;

while (i < word.length()) {

if (isdigit(word[i])) { // 如果是数字，则提取整数

string num;

while (i < word.length() && isdigit(word[i])) {

num += word[i];

i++;

}

// 去除前导零

int start = 0;

while (start < num.length() && num[start] == '0') {

start++;

}

if (start == num.length()) { // 整数为0时

integers.insert("0");

} else {

integers.insert(num.substr(start));

}

} else { // 如果不是数字，则继续向后遍历

i++;

}

}

return integers.size();

}

};

这段代码遍历了字符串 word，并根据字符是否为数字来提取整数。然后，它使用一个 unordered\_set<string> 来存储不同的整数。在提取整数时，如果整数有前导零，则会去除这些前导零，确保整数的不同表示形式被视为相同的整数。最后，返回集合的大小，即不同整数的数目。

**或：**

1、对合法数字加unordered\_set<>，即可去重复

2、合法数字需要处理

非数字时，用"-"表示

对于前置0，如果数字本身是"0"时又添加新数字，就删去"0"再加入数字

3、使用字符串保存，int会溢出

4、最后统计数量

**代码：**

class Solution {

public:

int numDifferentIntegers(string word) {

word += "-";

unordered\_set<string> ans;

string val = "-";

for (auto c : word) {

if (!isdigit(c)) {

if (val == "-") continue;

ans.insert(val);

val = "-";

}

else {

val = (val == "-") ? "0" : val;

val = (val == "0") ? "" : val;

val += c;

}

}

return ans.size();

}

};

或：

注意：不能直接统计整形，很容易溢出，转换为string保存则不用担心溢出，但是要考虑多个前导0的影响（用stringstream可以很简单的把字符串把空格分开读入）

class Solution {

public:

int numDifferentIntegers(string word) {

unordered\_set<string> arr; //统计每个不同的字符串个数

for (auto &ch : word) if (ch > '9' || ch < '0') ch = ' '; // 不熟数数字 就改为空格

stringstream ss(word); //按空格 分开读入

string t;

while (ss >> t) {

int i = 0;

for (; i < t.size() && t[i] == '0'; ++i) {} //去掉前导 0的影响

arr.insert(t.substr(i));

}

return arr.size();

}

};