**难度: 简单**

稀疏数组搜索。有个排好序的字符串数组，其中散布着一些空字符串，编写一种方法，找出给定字符串的位置。

示例1:

输入: words = ["at", "", "", "", "ball", "", "", "car", "", "","dad", "", ""], s = "ta"

输出：-1

说明: 不存在返回-1。

示例2:

输入：words = ["at", "", "", "", "ball", "", "", "car", "", "","dad", "", ""], s = "ball"

输出：4

提示:

words的长度在[1, 1000000]之间

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/sparse-array-search-lcci

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

**思路1(没有使用压缩):**

1). 创建一个map用来存取String和Integer

2). 循环把String数组进行存储

3). 对map进行判断,如果为null ,return -1 否则直接返回下标

|  |
| --- |
| public int findString(String[] words, String s) {  HashMap<String, Integer> map = new HashMap<>();  for (int i = 0; i < words.length; i++) {  map.put(words[i],i);  }  Integer result = map.get(s);  if (result == null) {  return -1;  }  return result;  } |

**思路2(使用稀疏数组):**

1). 先统计一下不为空的字符串

2). 创建一个二维数组大小就为 [ 不为空的字符串个数 ][2]

说明: 2列是因为 第0个放数据,第1个放下标 (数组从0开始)

3). 再次判断不为空的字符串,把不为空的数据和下标全部放到二维数组中

4). 判断二维数组中是否有对应的字符串 如果有,返回下标,没有的话,直接返回-1

|  |
| --- |
| public int findString2(String[] words, String s) {  int hit = 0;  for (int i=0; i < words.length; i++) {  //统计不等于空的字符串  if (!words[i].equals("")) {  hit++;  }  }    String[][] remaining = new String[hit][2];  int index = 0;  for (int i = 0; i < words.length; i++) {  //存储不等于空的字符串  if (!words[i].equals("")) {  remaining[index][0] = words[i];  remaining[index][1] = i+"";  index++;  }  }  //查找压缩过后的二维数组  for (int i = 0; i < hit; i++) {  if (s.equals(remaining[i][0])) {  return Integer.parseInt(remaining[i][1]);  }  }  return -1;  } |