

题目描述

小王设计了一个简单的猜字谜游戏，游戏的谜面是一个错误的单词，比如nesw，玩家需要猜出谜底库中正确的单词。猜中的要求如下：

对于某个谜面和谜底单词，满足下面任一条件都表示猜中：

- 1. 变换顺序以后一样的，比如通过变换w和e的顺序，“nwes”跟“news”是可以完全对应的；
- 2. 字母去重以后是一样的，比如“woood”和“wood”是一样的，它们去重后都是“wod”

请你写一个程序帮忙在谜底库中找到正确的谜底。谜面是多个单词，都需要找到对应的谜底，如果找不到的话，返回“not found”

输入描述

- 1. 谜面单词列表，以“,”分隔
- 2. 谜底库单词列表，以“;”分隔

输出描述

- 匹配到的正确单词列表，以“;”分隔
- 如果找不到，返回“not found”

备注

- 1. 单词的数量N的范围： $0 < N < 1000$
- 2. 词汇表的数量M的范围： $0 < M < 1000$
- 3. 单词的长度P的范围： $0 < P < 20$
- 4. 输入的字符只有小写英文字母，没有其他字符

用例

输入	conection connection,today
输出	connection
说明	无

输入	bdni,woood bind,wrong,wood
输出	bind,wood
说明	无

题目解析

本题有点歧义，那就是：

对于某个谜面和谜底单词，满足下面任一条件都表示猜中：

1. 变换顺序以后一样的，比如通过变换w和e的顺序，“nwes”跟“news”是可以完全对应的；

2. 字母去重以后是一样的，比如“woood”和“wood”是一样的，它们去重后都是“wod”

那么如果两个条件都满足的话，算不算猜中呢？

如果两个条件都满足也算猜中的话，那我们直接对谜底和谜面单词进行去重+字典序排序🔗，然后对比即可。

如果两个条件都满足不算猜中，只有一个条件满足才算猜中的话，则需要对单词分别进行去重和字典序排序，然后对比两次。

这里我给出两个情况的实现。

2023.02.21 根据机考网友反馈，本题使用“两个条件都满足”的解法可以获得100%通过率

JavaScript算法题解

伏城之外

已关注

0

7

3

专栏目录

已订阅

## 题目解析

本题有点歧义，那就是：

对于某个谜面和谜底单词，满足下面任一条件都表示猜中：

1. 变换顺序以后一样的，比如通过变换w和e的顺序，“nwes”跟“news”是可以完全对应的；
2. 字母去重以后是一样的，比如“wood”和“wod”是一样的，它们去重后都是“wod”

那么如果两个条件都满足的话，算不算猜中呢？

如果两个条件都满足也算猜中的话，那我们直接对谜底和谜面单词进行去重+字典序排序<sup>Q</sup>，然后对比即可。

如果两个条件都满足不算猜中，只有一个条件满足才算猜中的话，则需要对单词分别进行去重和字典序排序，然后对比两次。

这里我给出两个情况的实现。

2023.02.21 根据机考网友反馈，本题使用“两个条件都满足”的解法可以获得100%通过率

## JavaScript算法源码

两个条件都满足也算猜中

```
1  /* JavaScript Node ACM模式 控制台输入获取 */
2  const readline = require("readline");
3
4  const rl = readline.createInterface({
5    input: process.stdin,
6    output: process.stdout,
7  });
8
9  const lines = [];
10 rl.on("line", (line) => {
11   lines.push(line);
12 })
13 if (lines.length === 2) {
14   const issues = lines[0].split(",");
15   const answers = lines[1].split(",");
16   console.log(getResult(issues, answers));
17   lines.length = 0;
18 }
19 });
20
21 function getResult(issues, answers) {
22   const ans = [];
23
24   for (let issue of issues) {
25     const str1 = [...new Set(issue)].sort().join("");
26     let find = false;
27
28     for (let answer of answers) {
29       const str2 = [...new Set(answer)].sort().join("");
30       if (str1 === str2) {
31         ans.push(answer);
32         find = true;
33         // break; // 如果一个谜面对应多个谜底，这里就不能break，如果一个谜面只对应一个谜底，那这里就要break，考试的时候都证
34       }
35     }
36
37     if (!find) {
38       ans.push("not found");
39     }
40   }
41
42   return ans.join(",");
43 }
```

唯一条件满足才算猜中

```
1  /* JavaScript Node ACM模式 控制台输入获取 */
2  const readline = require("readline");
3
4  const rl = readline.createInterface({
5    input: process.stdin,
6    output: process.stdout,
7  });
8
9  const lines = [];
10 rl.on("line", (line) => {
11   lines.push(line);
12 })
13 if (lines.length === 2) {
14   const issues = lines[0].split(",");
15   const answers = lines[1].split(",");
16   console.log(getResult(issues, answers));
17   lines.length = 0;
18 }
19 });
20
21 function getResult(issues, answers) {
```

唯一条件满足才算猜中

```
1  /* JavaScript Node ACM模式 控制台输入获取 */
2  const readline = require("readline");
3
4  const rl = readline.createInterface({
5    input: process.stdin,
6    output: process.stdout,
7  });
8
9  const lines = [];
10 rl.on("line", (line) => {
11   lines.push(line);
12
13   if (lines.length === 2) {
14     const issues = lines[0].split(",");
15     const answers = lines[1].split(",");
16     console.log(getResult(issues, answers));
17     lines.length = 0;
18   }
19 });
20
21 function getResult(issues, answers) {
22   const ans = [];
23
24   for (let issue of issues) {
25     const sorted_issue = [...issue].sort().join(""); // 排序后字符串
26     const distinct_issue = [...new Set(issue)].join(""); // 去重后字符串
27     let find = false;
28
29     for (let answer of answers) {
30       const sorted_answer = [...answer].sort().join(""); // 排序后字符串
31       const distinct_answer = [...new Set(answer)].join(""); // 去重后字符串
32
33       if (
34         sorted_issue === sorted_answer ||
35         distinct_issue === distinct_answer
36       ) {
37         ans.push(answer);
38         find = true;
39         // break; // 如果一个谜面对应多个谜底，这里就不能break，如果一个谜面只对应一个谜底，那这里就要break，考试的时候都这
40       }
41     }
42
43     if (!find) {
44       ans.push("not found");
45     }
46   }
47
48   return ans.join(",");
49 }
```

Java算法源码

两个条件都满足也算猜中

```
1  import java.util.*;
2
3  public class Main {
4    public static void main(String[] args) {
5      Scanner sc = new Scanner(System.in);
6
7      String[] issues = sc.nextLine().split(",");
8      String[] answers = sc.nextLine().split(",");
9
10     System.out.println(getResult(issues, answers));
11   }
12
13   public static String getResult(String[] issues, String[] answers) {
14     ArrayList<String> ans = new ArrayList<>();
15
16     for (String issue : issues) {
17       String str1 = getSortedAndDistinctStr(issue);
18       boolean find = false;
19
20       for (String answer : answers) {
21         String str2 = getSortedAndDistinctStr(answer);
22         if (str1.equals(str2)) {
23           ans.add(answer);
24           find = true;
25           // break; // 如果一个谜面对应多个谜底，这里就不能break，如果一个谜面只对应一个谜底，那这里就要break
26         }
27       }
28
29       if (!find) {
30         ans.add("not found");
31       }
32     }
33
34     StringJoiner sj = new StringJoiner(",","","");
35     for (String an : ans) {
36       sj.add(an);
37     }
38     return sj.toString();
39   }
40
41   private static String getSortedAndDistinctStr(String str) {
42     char[] chars = str.toCharArray();
43     Arrays.sort(chars);
44     String distinct = "";
45     for (char c : chars) {
46       if (!distinct.contains(c)) {
47         distinct += c;
48       }
49     }
50     return distinct;
51   }
52 }
```

## Java算法源码

两个条件都满足也算猜中

```
1 import java.util.*;
2
3 public class Main {
4     public static void main(String[] args) {
5         Scanner sc = new Scanner(System.in);
6
7         String[] issues = sc.nextLine().split(",");
8         String[] answers = sc.nextLine().split(",");
9
10        System.out.println(getResult(issues, answers));
11    }
12
13    public static String getResult(String[] issues, String[] answers) {
14        ArrayList<String> ans = new ArrayList<>();
15
16        for (String issue : issues) {
17            String str1 = getSortedAndDistinctStr(issue);
18            boolean find = false;
19
20            for (String answer : answers) {
21                String str2 = getSortedAndDistinctStr(answer);
22                if (str1.equals(str2)) {
23                    ans.add(answer);
24                    find = true;
25                    // break; // 如果一个谜面对应多个谜底，这里就不能break，如果一个谜面只对应一个谜底，那这里就要break。
26                }
27            }
28
29            if (!find) {
30                ans.add("not found");
31            }
32        }
33
34        StringJoiner sj = new StringJoiner(",","","");
35        for (String an : ans) {
36            sj.add(an);
37        }
38        return sj.toString();
39    }
40
41    public static String getSortedAndDistinctStr(String str) {
42        // HashSet不会记录元素加入顺序，会按照hash算法排序，因此不需要额外排序
43        HashSet<Character> set = new HashSet<>();
44        for (char c : str.toCharArray()) set.add(c);
45        return set.toString();
46    }
47 }
```

唯一条件满足才算猜中

```
1 import java.util.*;
2
3 public class Main {
4     public static void main(String[] args) {
5         Scanner sc = new Scanner(System.in);
6
7         String[] issues = sc.nextLine().split(",");
8         String[] answers = sc.nextLine().split(",");
9
10        System.out.println(getResult(issues, answers));
11    }
12
13    public static String getResult(String[] issues, String[] answers) {
14        ArrayList<String> ans = new ArrayList<>();
15
16        for (String issue : issues) {
17            String[] issueDeal = getSortedAndDistinctStr(issue);
18            boolean find = false;
19
20            for (String answer : answers) {
21                String[] answerDeal = getSortedAndDistinctStr(answer);
22
23                if (issueDeal[0].equals(answerDeal[0]) || issueDeal[1].equals(answerDeal[1])) {
24                    ans.add(answer);
25                    find = true;
26                    // break; // 如果一个谜面对应多个谜底，这里就不能break，如果一个谜面只对应一个谜底，那这里就要break。
27                }
28            }
29
30            if (!find) {
31                ans.add("not found");
32            }
33        }
34
35        StringJoiner sj = new StringJoiner(",","","");
36        for (String an : ans) {
37            sj.add(an);
38        }
39        return sj.toString();
40    }
```

唯一条件满足才算猜中


```
1 import java.util.*;
2
3 public class Main {
4     public static void main(String[] args) {
5         Scanner sc = new Scanner(System.in);
6
7         String[] issues = sc.nextLine().split(",");
8         String[] answers = sc.nextLine().split(",");
9
10        System.out.println(getResult(issues, answers));
11    }
12
13    public static String getResult(String[] issues, String[] answers) {
14        ArrayList<String> ans = new ArrayList<>();
15
16        for (String issue : issues) {
17            String[] issueDeal = getSortedAndDistinctStr(issue);
18            boolean find = false;
19
20            for (String answer : answers) {
21                String[] answerDeal = getSortedAndDistinctStr(answer);
22
23                if(issueDeal[0].equals(answerDeal[0]) || issueDeal[1].equals(answerDeal[1])) {
24                    ans.add(answer);
25                    find = true;
26                    // break; // 如果一个谜面对应多个谜底, 这里就不能break, 如果一个谜面只对应一个谜底, 那这里就要break,
27                }
28            }
29
30            if(!find) {
31                ans.add("not found");
32            }
33        }
34
35        StringJoiner sj = new StringJoiner(",","","");
36        for (String an : ans) {
37            sj.add(an);
38        }
39        return sj.toString();
40    }
41
42    public static String[] getSortedAndDistinctStr(String str) {
43        char[] arr = str.toCharArray();
44        Arrays.sort(arr);
45        String sorted_str = new String(arr); // 字典序排序后的字符串
46
47        LinkedHashSet<Character> set = new LinkedHashSet<>();
48        for (char c : str.toCharArray()) set.add(c);
49        String distinct_str = set.toString(); // 去重后的字符串
50
51        return new String[]{sorted_str, distinct_str};
52    }
53 }
```

Python算法源码

两个条件都满足也算猜中

```
1 # 输入数据
2 issues = input().split(",")
3 answers = input().split(",")
4
5
6 # 算法入口
7 def getResult(issues, answers):
8     ans = []
9
10    for issue in issues:
11        str1 = "".join(sorted(set(issue)))
12        find = False
13
14    for answer in answers:
15        str2 = "".join(sorted(set(answer)))
16
17        if str1 == str2:
18            ans.append(answer)
19            find = True
20            # break # 如果一个谜面对应多个谜底, 这里就不能break, 如果一个谜面只对应一个谜底, 那这里就要break, 考试的时候
21
22    if not find:
23        ans.append("not found")
24
25    return ",".join(ans)
26
27
28 # 算法调用
29 print(getResult(issues, answers))
```

唯一条件满足才算猜中

 伏城之外 [已关注](#)

👍 0 🗨 7



💬 3



[专栏目录](#)

[已订阅](#)

## Python算法源码

两个条件都满足也算猜中

```
1 # 输入获取
2 issues = input().split(",")
3 answers = input().split(",")
4
5
6 # 算法入口
7 def getResult(issues, answers):
8     ans = []
9
10    for issue in issues:
11        str1 = "".join(sorted(set(issue)))
12        find = False
13
14        for answer in answers:
15            str2 = "".join(sorted(set(answer)))
16
17            if str1 == str2:
18                ans.append(answer)
19                find = True
20                # break # 如果一个谜面对应多个谜底，这里就不能break，如果一个谜面只对应一个谜底，那这里就要break，考试的时候
21
22        if not find:
23            ans.append("not found")
24
25    return ",".join(ans)
26
27
28 # 算法调用
29 print(getResult(issues, answers))
```

唯一条件满足才算猜中

```
1 # 输入获取
2 issues = input().split(",")
3 answers = input().split(",")
4
5
6 # 算法入口
7 def getResult(issues, answers):
8     ans = []
9
10    for issue in issues:
11        sorted_issue = "".join(sorted(issue))
12        distinct_issue = "".join(set(issue))
13        find = False
14
15        for answer in answers:
16            sorted_answer = "".join(sorted(answer))
17            distinct_answer = "".join(set(answer))
18
19            if sorted_issue == sorted_answer or distinct_issue == distinct_answer:
20                ans.append(answer)
21                find = True
22                # break # 如果一个谜面对应多个谜底，这里就不能break，如果一个谜面只对应一个谜底，那这里就要break，考试的时候
23
24        if not find:
25            ans.append("not found")
26
27    return ",".join(ans)
28
29
30 # 算法调用
31 print(getResult(issues, answers))
```