

题目描述

数字0、1、2、3、4、5、6、7、8、9分别关联 a~z 26个英文字母。

- 0 关联 “a”, “b”, “c”
- 1 关联 “d”, “e”, “f”
- 2 关联 “g”, “h”, “i”
- 3 关联 “j”, “k”, “l”
- 4 关联 “m”, “n”, “o”
- 5 关联 “p”, “q”, “r”
- 6 关联 “s”, “t”
- 7 关联 “u”, “v”
- 8 关联 “w”, “x”
- 9 关联 “y”, “z”

例如7关联”u”, “v”, 8关联”x”, “w”, 输入一个字符串例如“78”, 和一个屏蔽字符串“ux”, 那么“78”可以组成多个字符串例如: “ux”, “uw”, “vx”, “vw^Q”, 过滤这些完全包含屏蔽字符串的每一个字符的字符串, 然后输出剩下的字符串。

输入描述

无

输出描述

无

用例

输入	78 ux
输出	uw vx vw
说明	ux完全包含屏蔽字符串ux, 因此剔除

题目解析

本题可以使用 [深度优先^Q搜索DFS](#)。

本题类似于[LeetCode - 17 电话号码的字母组合_伏城之外的博客-CSDN博客](#)

另外本题中, 判断一个字符串A完全包含另一个字符串B全部字符, 的方法请看

[华为OD机试 - 密室逃生游戏_伏城之外的博客-CSDN博客](#)

JavaScript算法源码

```
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 arr = lines[0].split("");
15     const filter = lines[1];
16
17     console.log(getResult(arr, filter));
18
19     lines.length = 0;
```

JavaScript算法源码

```
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
14 if (lines.length === 2) {
15   const arr = lines[0].split("");
16   const filter = lines[1];
17
18   console.log(getResult(arr, filter));
19
20   lines.length = 0;
21 }
22
23 const map = ["abc", "def", "ghi", "jkl", "mno", "pqr", "st", "uv", "wx", "yz"];
24
25 function getResult(arr, filter) {
26   arr = arr.map((val) => map[val]);
27   filter = [...filter].sort().join("");
28
29   const res = [];
30   dfs(arr, 0, [], res, filter);
31
32   return res.join(" ");
33 }
34
35 function dfs(arr, index, path, res, filter) {
36   if (index === arr.length) {
37     const src = [...path].sort().join("");
38     // 过滤这些完全包含目标字符串的每一个字符的字符串
39     if (!include(src, filter)) res.push(path.join(""));
40     return;
41   }
42
43   for (let j = 0; j < arr[index].length; j++) {
44     path.push(arr[index][j]);
45     dfs(arr, index + 1, path, res, filter);
46     path.pop();
47   }
48 }
49
50 // 该方法用于判断src字符串是否完全包含tar字符串的每一个字符
51 function include(src, tar) {
52   if (tar.length > src.length) return false;
53
54   let i = 0;
55   let j = 0;
56
57   while (i < src.length && j < tar.length) {
58     if (src[i] === tar[j]) j++;
59     i++;
60   }
61
62   return j === tar.length;
63 }
```

Java算法源码

```
1  import java.util.*;
2
3  public class Main {
4    static String[] map = {"abc", "def", "ghi", "jkl", "mno", "pqr", "st", "uv", "wx", "yz"};
5
6    public static void main(String[] args) {
7      Scanner sc = new Scanner(System.in);
8
9      Integer[] arr =
10         Arrays.stream(sc.next().split("")).map(Integer::parseInt).toArray(Integer[]::new);
11      String filter = sc.next();
12
13      System.out.println(getResult(arr, filter));
14    }
15
16    public static String getResult(Integer[] arr, String filter) {
17      String[] newArr = Arrays.stream(arr).map(val -> map[val]).toArray(String[]::new);
18
19      char[] cArr = filter.toCharArray();
20      Arrays.sort(cArr);
21      filter = new String(cArr);
22
23      ArrayList<String> res = new ArrayList<>();
24      dfs(newArr, 0, new LinkedList<>(), res, filter);
25    }
26
27    private static void dfs(String[] newArr, int index, LinkedList<String> path, ArrayList<String> res, String filter) {
28      if (index === newArr.length) {
29        String src = path.stream().sorted().collect(Collectors.joining(""));
30        // 过滤这些完全包含目标字符串的每一个字符的字符串
31        if (!include(src, filter)) res.add(path.stream().collect(Collectors.joining(" ")));
32        return;
33      }
34
35      for (String str : newArr[index]) {
36        path.add(str);
37        dfs(newArr, index + 1, path, res, filter);
38        path.removeLast();
39      }
40    }
41
42    private static boolean include(String src, String tar) {
43      if (tar.length > src.length) return false;
44
45      int i = 0, j = 0;
46
47      while (i < src.length && j < tar.length) {
48        if (src.charAt(i) === tar.charAt(j)) j++;
49        i++;
50      }
51
52      return j === tar.length;
53    }
54 }
```

Java算法源码

```
1 import java.util.*;
2
3 public class Main {
4     static String[] map = {"abc", "def", "ghi", "jkl", "mno", "pqr", "st", "uv", "wx", "yz"};
5
6     public static void main(String[] args) {
7         Scanner sc = new Scanner(System.in);
8
9         Integer[] arr =
10             Arrays.stream(sc.next().split("")).map(Integer::parseInt).toArray(Integer[]::new);
11         String filter = sc.next();
12
13         System.out.println(getResult(arr, filter));
14     }
15
16     public static String getResult(Integer[] arr, String filter) {
17         String[] newArr = Arrays.stream(arr).map(val -> map[val]).toArray(String[]::new);
18
19         char[] cArr = filter.toCharArray();
20         Arrays.sort(cArr);
21         filter = new String(cArr);
22
23         ArrayList<String> res = new ArrayList<>();
24         dfs(newArr, 0, new LinkedList<>(), res, filter);
25
26         StringJoiner sj = new StringJoiner(" ", "", "");
27         for (String str : res) {
28             sj.add(str);
29         }
30         return sj.toString();
31     }
32
33     public static void dfs(
34         String[] arr, int index, LinkedList<Character> path, ArrayList<String> res, String filter) {
35         if (index == arr.length) {
36             // 过滤这些完全包含筛选字符串的每一个字符的字符串
37             if (!include(path, filter)) {
38                 StringBuilder sb = new StringBuilder();
39                 for (Character c : path) {
40                     sb.append(c);
41                 }
42                 res.add(sb.toString());
43             }
44             return;
45         }
46
47         for (int i = 0; i < arr[index].length(); i++) {
48             path.addLast(arr[index].charAt(i));
49             dfs(arr, index + 1, path, res, filter);
50             path.removeLast();
51         }
52     }
53
54     public static boolean include(LinkedList<Character> path, String filter) {
55         StringBuilder sb = new StringBuilder();
56         path.stream().sorted().forEach(sb::append);
57         String src = sb.toString();
58
59         if (filter.length() > src.length()) return false;
60
61         int i = 0;
62         int j = 0;
63
64         while (i < src.length() && j < filter.length()) {
65             if (src.charAt(i) == filter.charAt(j)) j++;
66             i++;
67         }
68
69         return j == filter.length();
70     }
71 }
```

Python算法源码

```
1 # 输入获取
2 originStr = input()
3 filterStr = input()
4
5 mapping = ["abc", "def", "ghi", "jkl", "mno", "pqr", "st", "uv", "wx", "yz"]
6
7
8 # 算法入口
9 def getResult(originStr, filterStr):
10     newArr = list(map(lambda x: mapping[int(x)], list(originStr)))
11     filterStr = "".join(sorted(filterStr))
12
13     res = []
14     dfs(newArr, 0, [], res, filterStr)
15
16     return " ".join(res)
```



伏城之外 已关注



0



2



3



专栏目录



已订阅

Python算法源码

```
1 # 输入获取
2 originStr = input()
3 filterStr = input()
4
5 mapping = ["abc", "def", "ghi", "jkl", "mno", "pqr", "st", "uv", "wx", "yz"]
6
7
8 # 算法入口
9 def getResult(originStr, filterStr):
10     newArr = list(map(lambda x: mapping[int(x)], list(originStr)))
11     filterStr = "".join(sorted(filterStr))
12
13     res = []
14     dfs(newArr, 0, [], res, filterStr)
15
16     return " ".join(res)
17
18
19 def dfs(arr, index, path, res, filterStr):
20     if index == len(arr):
21         src = "".join(sorted(path))
22         if not include(src, filterStr):
23             res.append("".join(path))
24         return
25
26     for i in range(len(arr[index])):
27         path.append(arr[index][i])
28         dfs(arr, index + 1, path, res, filterStr)
29         path.pop()
30
31
32 def include(src, tar):
33     if len(tar) > len(src):
34         return False
35
36     i = 0
37     j = 0
38
39     while i < len(src) and j < len(tar):
40         if src[i] == tar[j]:
41             j += 1
42         i += 1
43
44     return j == len(tar)
45
46
47 # 算法调用
48 print(getResult(originStr, filterStr))
```