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”，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

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const lines = [];
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 2) {
    const arr = lines[0].split("");
    const filter = lines[1];

    console.log(getResult(arr, filter));

    lines.length = 0;
```

JavaScript算法源码

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const lines = [];
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 2) {
    const arr = lines[0].split("");
    const filter = lines[1];

    console.log(getResult(arr, filter));

    lines.length = 0;
  }
});

const map = ["abc", "def", "ghi", "jkl", "mno", "pqr", "st", "uv", "wx", "yz"];

function getResult(arr, filter) {
  arr = arr.map((val) => map[val]);
  filter = [...filter].sort().join("");

  const res = [];
  dfs(arr, 0, [], res, filter);

  return res.join(" ");
}

function dfs(arr, index, path, res, filter) {
  if (index === arr.length) {
    const src = [...path].sort().join("");
    // 过滤这些完全包含屏蔽字符串的每一个字符的字符串
    if (!include(src, filter)) res.push(path.join(""));
    return;
  }

  for (let j = 0; j < arr[index].length; j++) {
    path.push(arr[index][j]);
    dfs(arr, index + 1, path, res, filter);
    path.pop();
  }
}

// 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
function include(src, tar) {
  if (tar.length > src.length) return false;

  let i = 0;
  let j = 0;

  while (i < src.length && j < tar.length) {
    if (src[i] == tar[j]) j++;
    i++;
  }

  return j == tar.length;
}
```

Java算法源码

```java
import java.util.*;

public class Main {
  static String[] map = {"abc", "def", "ghi", "jkl", "mno", "pqr", "st", "uv", "wx", "yz"};

  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    Integer[] arr =
        Arrays.stream(sc.next().split("")).map(Integer::parseInt).toArray(Integer[]::new);
    String filter = sc.next();

    System.out.println(getResult(arr, filter));
  }

  public static String getResult(Integer[] arr, String filter) {
    String[] newArr = Arrays.stream(arr).map(val -> map[val]).toArray(String[]::new);

    char[] cArr = filter.toCharArray();
    Arrays.sort(cArr);
    filter = new String(cArr);

    ArrayList<String> res = new ArrayList<>();
    dfs(newArr, 0, new LinkedList<>(), res, filter);
```

Java算法源码

```java
import java.util.*;

public class Main {
  static String[] map = {"abc", "def", "ghi", "jkl", "mno", "pqr", "st", "uv", "wx", "yz"};

  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    Integer[] arr =
        Arrays.stream(sc.next().split("")).map(Integer::parseInt).toArray(Integer[]::new);
    String filter = sc.next();

    System.out.println(getResult(arr, filter));
  }

  public static String getResult(Integer[] arr, String filter) {
    String[] newArr = Arrays.stream(arr).map(val -> map[val]).toArray(String[]::new);

    char[] cArr = filter.toCharArray();
    Arrays.sort(cArr);
    filter = new String(cArr);

    ArrayList<String> res = new ArrayList<>();
    dfs(newArr, 0, new LinkedList<>(), res, filter);

    StringJoiner sj = new StringJoiner(" ", "", "");
    for (String str : res) {
      sj.add(str);
    }
    return sj.toString();
  }

  public static void dfs(
      String[] arr, int index, LinkedList<Character> path, ArrayList<String> res, String filter) {
    if (index == arr.length) {
      // 过滤这些完全包含屏蔽字符串的每一个字符的字符串
      if (!include(path, filter)) {
        StringBuilder sb = new StringBuilder();
        for (Character c : path) {
          sb.append(c);
        }
        res.add(sb.toString());
      }
      return;
    }

    for (int i = 0; i < arr[index].length(); i++) {
      path.addLast(arr[index].charAt(i));
      dfs(arr, index + 1, path, res, filter);
      path.removeLast();
    }
  }

  public static boolean include(LinkedList<Character> path, String filter) {
    StringBuilder sb = new StringBuilder();
    path.stream().sorted().forEach(sb::append);
    String src = sb.toString();

    if (filter.length() > src.length()) return false;

    int i = 0;
    int j = 0;

    while (i < src.length() && j < filter.length()) {
      if (src.charAt(i) == filter.charAt(j)) j++;
      i++;
    }

    return j == filter.length();
  }
}
```

Python算法源码

```python
# 输入获取
originStr = input()
filterStr = input()

mapping = ["abc", "def", "ghi", "jkl", "mno", "pqr", "st", "uv", "wx", "yz"]


# 算法入口
def getResult(originStr, filterStr):
    newArr = list(map(lambda x: mapping[int(x)], list(originStr)))
    filterStr = "".join(sorted(filterStr))

    res = []
    dfs(newArr, 0, [], res, filterStr)

    return " ".join(res)
```

Python算法源码

```python
# 输入获取
originStr = input()
filterStr = input()

mapping = ["abc", "def", "ghi", "jkl", "mno", "pqr", "st", "uv", "wx", "yz"]


# 算法入口
def getResult(originStr, filterStr):
    newArr = list(map(lambda x: mapping[int(x)], list(originStr)))
    filterStr = "".join(sorted(filterStr))

    res = []
    dfs(newArr, 0, [], res, filterStr)

    return " ".join(res)


def dfs(arr, index, path, res, filterStr):
    if index == len(arr):
        src = "".join(sorted(path))
        if not include(src, filterStr):
            res.append("".join(path))
        return

    for i in range(len(arr[index])):
        path.append(arr[index][i])
        dfs(arr, index + 1, path, res, filterStr)
        path.pop()


def include(src, tar):
    if len(tar) > len(src):
        return False

    i = 0
    j = 0

    while i < len(src) and j < len(tar):
        if src[i] == tar[j]:
            j += 1
        i += 1

    return j == len(tar)


# 算法调用
print(getResult(originStr, filterStr))
```