

## Leetcode 17. Letter Combinations of a Phone Number

**Success** Details >

Runtime: **0 ms**, faster than **100.00%** of Java online submissions for Letter Combinations of a Phone Number.

Memory Usage: **37.6 MB**, less than **93.16%** of Java online submissions for Letter Combinations of a Phone Number.

Next challenges: [Generate Parentheses](#) [Combination Sum](#) [Binary Watch](#)

Show off your acceptance: [f](#) [t](#) [in](#)

Time Submitted	Status	Runtime	Memory	Language
11/21/2020 12:48	Accepted	0 ms	37.6 MB	java

```
10    {"p", "q", "r", "s"},
11    {"t", "u", "v"},
12    {"w", "x", "y", "z"}
13    };
14    List<String> res = new ArrayList<>();
15    backtrace(digits, 0, keyboard, new StringBuilder(), res);
16    return res;
17
18    void backtrace(String digits, int i, String[][] keyboard,
19    StringBuilder tmp, List<String> res) {
20        if (i == digits.length()) {
21            res.add(tmp.toString());
22            return;
23        }
24        int index = digits.charAt(i) - '2';
25        for (int j = 0; j < keyboard[i][index].length(); j++) {
26            tmp.append(keyboard[i][index].charAt(j));
27            backtrace(digits, i + 1, keyboard, tmp, res);
28            tmp.deleteCharAt(tmp.length() - 1);
29        }
30    }
```

Testcase Run Code Result Debuqquer

**Accepted** Runtime: 0 ms

Your input: "23"

Output: ["ad","ae","af","bd","be","bf","cd","ce","cf"]

Expected: ["ad","ae","af","bd","be","bf","cd","ce","cf"]

Console How to create a testcase

Run Code Submit

Main algorithm: backTrace

$T(n) = O(kn)$ ,  $k$  is the max number of letters per button,  $n$  is the size of input string

Space complexity:  $O(1) + O(n) + O(kn) = O(kn)$

$O(1)$ : pre-defined digit-letter mapping

$O(n)$ : temporary result during backTrace

$O(kn)$ : final result