17. Letter Combinations of a Phone Number

Jan. 23, 2019 | 247.7K views

Average Rating: 3.87 (84 votes)

Given a string containing digits from `2-9` inclusive, return all possible letter combinations that the number could represent.

A mapping of digit to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.

Example:

```
Input: "23"
Output: ["ad", "ae", "af", "bd", "be", "bf", "cd", "ce", "cf"].
```

Note:

Although the above answer is in lexicographical order, your answer could be in any order you want.

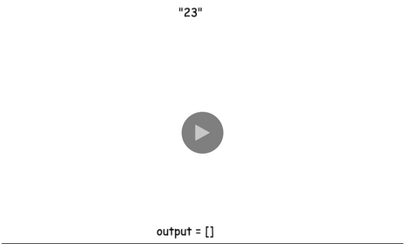
Solution

Approach 1: Backtracking

Backtracking is an algorithm for finding all solutions by exploring all potential candidates. If the solution candidate turns to be *not* a solution (or at least not the *last* one), backtracking algorithm discards it by making some changes on the previous step, *i.e.* *backtracks* and then try again.

Here is a backtrack function `backtrack(combination, next_digits)` which takes as arguments an ongoing letter combination and the next digits to check.

- If there is no more digits to check that means that the current combination is done.
- If there are still digits to check :
  - Iterate over the letters mapping the next available digit.
    - Append the current letter to the current combination `combination = combination + letter`.
    - Proceed to check next digits : `backtrack(combination + letter, next_digits[1:])`.



Java Python Copy

```
class Solution {
  Map<String, String> phone = new HashMap<String, String>() {{
    put("2", "abc");
    put("3", "def");
    put("4", "ghi");
    put("5", "jkl");
    put("6", "mno");
    put("7", "pqrs");
    put("8", "tuv");
    put("9", "wxyz");
  }};

  List<String> output = new ArrayList<String>();

  public void backtrack(String combination, String next_digits) {
    // if there is no more digits to check
    if (next_digits.length() == 0) {
      // the combination is done
      output.add(combination);
    }
    // if there are still digits to check
    else {
      // iterate over all letters which map
      // the next available digit
      String digit = next_digits.substring(0, 1);
      String letters = phone.get(digit);
      for (int i = 0; i < letters.length(); i++) {
```

Complexity Analysis

- Time complexity : $\mathcal{O}(3^N \times 4^M)$ where `N` is the number of digits in the input that maps to 3 letters (*e.g.* `2, 3, 4, 5, 6, 8`) and `M` is the number of digits in the input that maps to 4 letters (*e.g.* `7, 9`), and `N+M` is the total number digits in the input.
- Space complexity : $\mathcal{O}(3^N \times 4^M)$ since one has to keep $3^N \times 4^M$ solutions.

Rate this article:

Previous Next

Comments: 107

Sort By

Type comment here... (Markdown is supported)

Preview Post

chicurel ★ 289 March 11, 2019 11:06 AM Report

Why the solution is presented as backtracking if there's no 'back' tracking here? I think that should be mentioned just as a recursive solution?

282 Share Reply

SHOW 18 REPLIES

spide0606 ★ 76 March 20, 2019 7:40 AM

Runtime: 1 ms, faster than 100.00% of Java online submissions.

class Solution {
public List letterCombinations(String digits) {

Read More

76 Share Reply

SHOW 7 REPLIES

vicpark ★ 37 March 7, 2019 3:13 AM Report

16ms, faster than 100.00% of Python2 submissions

```
class Solution(object):
    def letterCombinations(self, digits):
```

Read More

36 Share Reply

SHOW 3 REPLIES

MichaelTan ★ 24 May 20, 2019 8:00 AM

I think the complexity is actually O(N * 4^N) because you actually have to construct the strings, each of which is length n.

24 Share Reply

SHOW 1 REPLY

vt1on ★ 499 February 10, 2019 2:27 AM Report

4ms, faster than 100% of C++ solutions:

```
vector<string> letterCombinations(string digits) {
        if (digits.empty()) return {};
```

Read More

12 Share Reply

SHOW 4 REPLIES

humy ★ 34 January 31, 2019 9:03 AM Report

32 ms, faster than 100.00% of Python3

```
class Solution:
    def letterCombinations(self, digits):
```

Read More

10 Share Reply

SHOW 2 REPLIES

conhic ★ 17 May 17, 2019 1:03 AM

C# solution, one liner.

```
public class Solution {
    private readonly Dictionary<char, List<char>> _phone = new Dictionary<char, Li
```

Read More

16 Share Reply

SHOW 6 REPLIES

haha000 ★ 9 May 27, 2019 9:30 AM Report

Shouldn't the time complexity be at least O((N+M) x (3^N x 4^N)), it takes at least (N+M) time complexity to construct each of the combination, let alone the inefficient way of concatenating strings in the solution (which takes actually O((N+M)^2) for each combination construction)?

7 Share Reply

SHOW 1 REPLY

namasteDuniya ★ 140 January 24, 2019 7:05 AM

Does there exist solution which can be done in better Time complexity than O(4^n), for simplicity I am not considering 3,4 combinations separate in complexity. Any suggestions or answers will be appreciated. Thanks.

7 Share Reply

SHOW 4 REPLIES

fsvara ★ 6 July 13, 2019 2:09 AM Report

Very concise Python 3:

```
import itertools as it
class Solution:
```

Read More

6 Share Reply

SHOW 4 REPLIES