Description | Editorial | Solutions (11.1K) | Submissions

17. Letter Combinations of a Phone Number

Medium | 16.4K | 876

Companies

Given a string containing digits from `2-9` inclusive, return all possible letter combinations that the number could represent. Return the answer in **any order**.

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

Example 1:

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

Example 2:

```
Input: digits = ""
```

Java | Auto

```java
class Solution {
    public List<String> letterCombinations(String digits) {
        List<String> ans = new ArrayList<>();
        if (digits.length() == 0)
        return ans;

        HashMap<Character , String> hm = new HashMap<>();
        hm.put('2' , "abc");
        hm.put('3' , "def");
        hm.put('4' , "ghi");
        hm.put('5' , "jkl");
        hm.put('6' , "mno");
        hm.put('7' , "pqrs");
        hm.put('8' , "tuv");
        hm.put('9' , "wxyz");

        backtrack(digits, 0, hm, new StringBuilder(), ans);
        return ans;
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

digits =
"23"

Output

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

Console | Run | Submit

Description | Editorial | Solutions (12.8K) | Submissions

34. Find First and Last Position of Element in Sorted Array

Medium 17.9K 437

Companies

Given an array of integers `nums` sorted in non-decreasing order, find the starting and ending position of a given `target` value.

If `target` is not found in the array, return `[-1, -1]`.

You must write an algorithm with `O(log n)` runtime complexity.

Example 1:

```
Input: nums = [5,7,7,8,8,10], target = 8
Output: [3,4]
```

Example 2:

```
Input: nums = [5,7,7,8,8,10], target = 6
Output: [-1,-1]
```

Example 3:

```
Input: nums = [], target = 0
Output: [-1,-1]
```

Constraints:

Java | Auto

```java
class Solution {
    public int first(int[] nums, int target) {
        int ans = -1;
        int low = 0, high = nums.length - 1;
        while (low <= high) {
            int mid = low + (high - low) / 2;
            if (nums[mid] == target) {
                ans = mid;
                high = mid - 1;
            } else if (nums[mid] < target) {
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }
        return ans;
    }

```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

nums =
[5,7,7,8,8,10]

target =
8

Console | Run | Submit