PhoneNumber

**Statement**

Given a string containing digits from 2 to 9 inclusive, with the possibility of each digit appearing multiple times, return all possible letter combinations that the number could represent. Return the answer in any order.

The illustration below shows the mapping of digits to letters in a telephone dial pad.

**Note:** The number 11 on the telephone dial pad does not correspond to any letter, so the input string only contains digits from 22 to 99.

A screenshot of a phone

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a math test

AI-generated content may be incorrect.

Solution:

**Pattern: Subsets**

This problem lends itself naturally to the subsets pattern. To solve this problem, we can use a backtracking algorithm as the solution template to correctly generate all the possible combinations.

**Solution**

Since we need to explore different choices for each digit and generate valid combinations, we can use the subset technique with backtracking. It involves iterating through each digit in the input sequence, recursively generating combinations of letters corresponding to that digit. Each recursive step explores all possible letters for the current digit, appending each letter to the current combination and moving to the next digit. If the combination reaches the same length as the input sequence, it is added to the list of valid combinations. However, if a certain combination doesn't match the expected length, we backtrack to the previous state and explore other options. This process continues until all digits have been processed.

For example, let’s consider the input digits “23”. We start with an empty combination list and the initial index of 0. For the first digit “2”, the corresponding letters are [“a”, “b”, “c”]. We add “a” to the combination and recursively call the function with the next index of 1. For the second digit “3”, the corresponding letters are [“d”, “e”, “f”]. We add “d” to the combination and recursively call the function with the next index of 2. At this point, the length of the combination is equal to the length of the input string, so we join the combination into a string, “ad”, and append it to the list of combinations. Then, we remove the last letter “d” from the combination and move on to the next letter, “e”, in the letters corresponding to digit “3”. We repeat this process until we have explored all possible combinations.

The algorithm works as follows:

1. It first checks if the input string is empty. If it is empty, we will return an empty list immediately.
2. Creates a hash map, digits\_mapping, which maps each digit to a list of corresponding letters. For example, “2” corresponds to letters “a”, “b”, and “c”.
3. The backtrack function is then called to generate the combinations recursively. It takes the parameters, index (the current index in the digits string), path (the current combination of letters being built), digits (the input string of digits), letters (the mapping of digits to letters) and combinations (a list to store the generated combinations). Within the backtrack function, we will perform the following steps:
   1. If the length of path is equal to the length of digits, it means we have a complete combination. The path is joined into a string and appended to the combinations list.
   2. Otherwise, the function retrieves the list of possible letters corresponding to the digit at the current index. It then iterates through each letter and performs the following steps:
      1. It adds the letter to the path.
      2. It recursively calls backtrack with the updated index and path to move on to the next digit.
      3. After the recursive call, the letter is removed from the path to backtrack and explore other letter combination possibilities.
4. Finally, the combinations list is returned as a result that contains all possible letter combinations made from the numbers in the string.

The slides below help to understand the solution in a better way.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

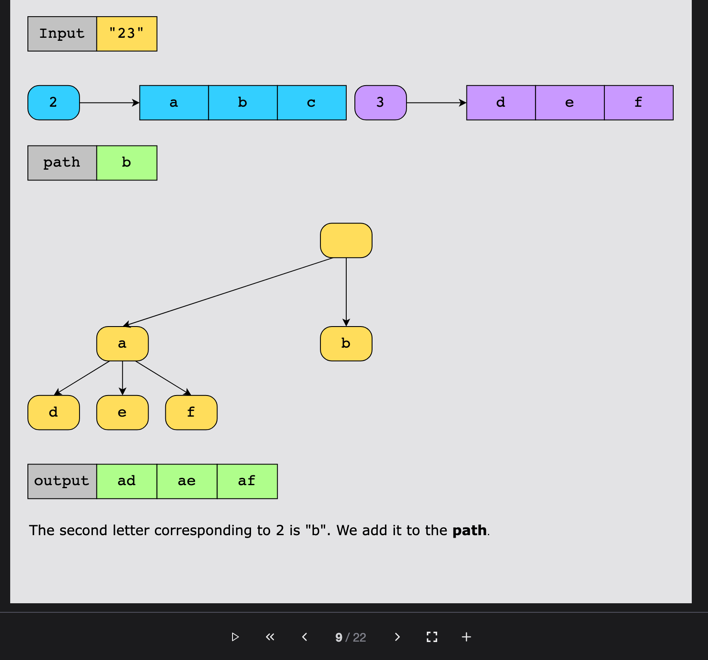
AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer screen

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer screen

AI-generated content may be incorrect.A screenshot of a computer screen

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.