Daily Coding Problem #81

Problem

This problem was asked by Yelp.

Given a mapping of digits to letters (as in a phone number), and a digit string, return all possible letters the number could represent. You can assume each valid number in the mapping is a single digit.

For example if {“2”: [“a”, “b”, “c”], 3: [“d”, “e”, “f”], …} then “23” should return [“ad”, “ae”, “af”, “bd”, “be”, “bf”, “cd”, “ce”, “cf"].

Solution

There is a relatively straight forward substructure to this problem.

Let's assume that we knew the result of the function with the same digits except the first character. Then, we could reconstruct the final result by: for each character the first digit maps to, prepend that character to each permutation from the recursive call.

For example, if the digits are '12', and the mapping is {'1': ['a', 'b', 'c'], '2': ['d', 'e', 'f']} then without the first digit, the result would be ['d', 'e', 'f']. If we prepend 'a', 'b', and 'c', to each permutation, we would get 'ad', 'ae', 'af', 'bd', 'be', 'bf', 'cd', 'ce', 'cf'.

```python
def get_permutations(digits, mapping):
    digit = digits[0]

    if len(digits) == 1:
        return mapping[digit]

    result = []
    for char in mapping[digit]:
        for perm in get_permutations(digits[1:], mapping):
            result.append(char + perm)
    return result
```