```
class Solution {
public:
    vector<string> letterCombinations(string digits) {
        static const vector<string> lookup = {" ", "", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv",
"wxyz"};

        if (empty(digits)) {
            return {};
        }
        int total = 1;
        for (const auto& digit : digits) {
            total *= size(lookup[digit - '0']);
        }
        vector<string> result;
        for (int i = 0; i < total; ++i) {
            int base = total;
            string curr;
            for (const auto& digit : digits) {
                const auto& choices = lookup[digit - '0'];
                base /= size(choices);
                curr.push_back(choices[(i / base) % size(choices)]);
            }
            result.emplace_back(move(curr));
        }
        return result;
    }
};

// Time:  O(n * 4^n)
// Space: O(1)
// iterative solution
class Solution2 {
public:
    vector<string> letterCombinations(string digits) {
        static const vector<string> lookup = {" ", "", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv",
"wxyz"};

        if (empty(digits)) {
            return {};
        }
        vector<string> result = {""};
        for (int i = size(digits) - 1; i >= 0; --i) {
            const auto& choices = lookup[digits[i] - '0'];
            int m = size(choices), n = size(result);
            result.resize(m * n);
            for (int j = m * n - 1; j >= 0; --j) {
                result[j] = choices[j / n] + result[j % n];
            }
        }
        return result;
    }
};

// Time:  O(n * 4^n)
// Space: O(n)
// recursive solution
class Solution3 {
public:
    vector<string> letterCombinations(string digits) {
        if (empty(digits)) {
            return {};
        }
        vector<string> result;
        string curr;
        letterCombinationsRecu(digits, &curr, &result);
        return result;
    }

private:
    void letterCombinationsRecu(const string &digits, string *curr, vector<string> *result) {
        static const vector<string> lookup = {" ", "", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv",
"wxyz"};

        if (size(*curr) == size(digits)) {
            result->emplace_back(*curr);
            return;
        }
        for (const auto& c: lookup[digits[size(*curr)] - '0']) {
            curr->push_back(c);
            letterCombinationsRecu(digits, curr, result);
            curr->pop_back();
        }
    }
};
```