My idea:

1. Use map to save digit in keyboard
2. Use backtrack (recursive) to generate all candidate.
3. Go from step 0
4. End when step == digits.length
5. With each step, push\_back value and next step, after that pop\_back value.

World idea:

class Solution {

public:

    void solve(string res, vector<string>& ans, string digits, int i, string mapping[]){

        if(i >= digits.length()){

            ans.push\_back(res);

            return;

        }

        int index = digits[i] - '0';

        string val = mapping[index];

        for(int j = 0; j<val.length(); j++){

            res.push\_back(val[j]);

            solve(res, ans, digits, i+1, mapping);

            res.pop\_back();

        }

    }

    vector<string> letterCombinations(string digits){

        vector<string> ans;

        if(digits.length() == 0)    return ans;

        string res = "";

        string mapping[10] = {"", "", "abc" , "def", "ghi", "jkl", "mno", "pqrs", "tuv", "wxyz"};

        solve(res, ans, digits, 0, mapping);

        return ans;

    }

};