Question: https://leetcode.com/problems/letter-combinations-of-a-phone-number/

So the solution is pretty simple, create a dict manually where key is the digit, and its value contains list of all characters which we can obtain from that digit.

Now use DFS:

For 1st digit we have n possibilities of characters, now for each of those characters we have another n possibilities for next digit. Like this we will get a tree.

Once there is no more charecter add the obtained combinations in list, i.e., the result.

Code:

class Solution {

Map<Integer, List<Character>> dict = new HashMap<>();

public List<String> letterCombinations(String digits) {

dict.put(1, new ArrayList<Character>());

List<String> res = new ArrayList<String>();

if(digits.length()<1) return res;

Character v='a';

for(int i=2; i<=9; i++){

List<Character> li = new ArrayList<Character>();

for(int j=0; j<3; j++){

li.add(v);

v++;

}

if(i==9||i==7){

li.add(v);

v++;

}

dict.put(i, li);

}

helper(digits, 0, res, "");

return res;

}

public void helper(String digits, int pos, List<String> res, String s){

if(pos>=digits.length()) res.add(s);

else{

for(Character c: dict.get(digits.charAt(pos)-'0')){

helper(digits, pos+1, res, s+c);

}

}

return;

}

}

Github Link :<https://lnkd.in/ecwtJeaz>