**[Letter Combinations of a Phone Number](https://leetcode.com/problems/letter-combinations-of-a-phone-number/)**

**import** java.util.\*;

**public** **class** LetterCombination {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

System.***out***.println(*letterCombinations*("23"));

}

**public** **static** List<String> letterCombinations(String digits) {

List<String> result = **new** ArrayList<>();

**if**(digits == **null** || digits.length() == 0)

**return** result;

Queue<String> queue = **new** LinkedList<>();

String[] mapping = {"0","1","abc","def","ghi","jkl","mno","pqrs","tuv","wxyz"};

queue.offer("");

**for**(**int** i = 0 ; i < digits.length() ; i++) {

**int** length = queue.size();

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

String temp = queue.poll();

**for**(**char** c : mapping[Character.*getNumericValue*(digits.charAt(i))].toCharArray()) {

queue.offer(temp + c);

}

}

}

**return** **new** ArrayList<>(queue);

}

}

Time complexity : O(3^*N* \* 4^*M*) where N is the number of digits in the input that maps to 3 letters (e.g. 2, 3, 4, 5, 6, 8) and M is the number of digits in the input that maps to 4 letters (e.g. 7, 9), and N+M is the total number digits in the input.

Space complexity : O( 3^N \* 4^M), since one has to keep 3^N \* 4^Msolutions.