**Problem Name:** Lexicographical Numbers

**Tags:** Depth-first-search Trie

**Company:** ByteDance Pony.ai Microsoft

**Level:** Medium

**Language:** C++

**Problem Statement:** Given an integer n, return all the numbers in the range [1, n] sorted in lexicographical order.

**Input Format:**

The first line of each input contains integer n

# Output Format:

An array of size n containing lexicographical number

# Constraints:

* 1 <= n <= 5 \* 104

# Example:

**Input:** n = 13

**Output:** [1,10,11,12,13,2,3,4,5,6,7,8,9]

# Solution:

**Explanation:** In this problem we begin with number, i=1. We change the value of i using two ways on priority:

1. **multiplying by 10**: It is given higher priority because for example: for i=10, 100 (10 \* 10) will always before 11 (10+1).
2. **incrementing by 1**: It is given lower priority. Also, incrementing the value can only be performed till last digit is 9. Because, for example: i=119, if we increment it we 120, but by this process 120 gets added to result before 12 (which would have come by incrementing 10).

# Code:

 #include <bits/stdc++.h>

 using namespace std;

 void fun(vector<int>&res, int i, int n){

        if(i>n)

            return;

        res.push\_back(i);

        fun(res,i\*10,n);

        if((i+1)%10!=0)

            fun(res,i+1,n);

    }

    vector<int> lexicalOrder(int n) {

        vector<int> res;

        fun(res,1,n);

        return res;

    }

    int main(){

        int n;

        cin>>n;

        vector<int> result(n);

        result = lexicalOrder(n);

        for(int i=0; i<n; i++){

            cout<<result[i]<<" ";

        }

    }

**Time Complexity:** O(N)

**Space Complexity:** O(1)