386. Lexicographical Numbers

Given an integer `n`, return all the numbers in the range `[1, n]` sorted in lexicographical order.

You must write an algorithm that runs in `O(n)` time and uses `O(1)` extra space.

Example 1:

```
Input: n = 13
Output: [1,10,11,12,13,2,3,4,5,6,7,8,9]
```

Example 2:

```
Input: n = 2
Output: [1,2]
```

Constraints:

- `1 <= n <= 5 * 10`^{4}

386. Lexicographical Numbers

```
/*
    Recursion preorder traversal
    Time complexity: O(n)
    Space complexity: O(log10(n))
*/
class Solution {
    public:
        std::vector<int> lexicalOrder(int n) {
            std::vector<int> ans;

            auto solve=[&](int cur,auto& self)->void{
                if(cur>n) return;
                ans.push_back(cur);
                cur*=10;
                for(int i=0;i<=9;++i) self(cur+i,self);
            };

            for(int i=1;i<=9;++i) solve(i,solve);
            return ans;
        }
};
```

386. Lexicographical Numbers

```
/*
    Iterative
    Time complexity: o(n)
    Space complexity: O(1)
*/
class Solution {
    public:
        std::vector<int> lexicalOrder(int n) {
            std::vector<int> ans;
            int cur=1;
            while(ans.size()<n){
                ans.push_back(cur);
                if(cur*10<=n) cur*=10;
                else{
                      while(cur==n || cur%10==9) cur/=10;
                      cur++;
               }
            }
            return ans;
        }
};
```