440. K-th Smallest in Lexicographical Order

Given two integers `n` and `k`, return *the* `k`th *lexicographically smallest integer in the range* `[1, n]`.

Example 1:

```
Input: n = 13, k = 2
Output: 10
Explanation: The lexicographical order is [1, 10, 11, 12, 13, 2, 3, 4, 5, 6, 7, 8,
9], so the second smallest number is 10.
```

Example 2:

```
Input: n = 1, k = 1
Output: 1
```

Constraints:

- `1 <= k <= n <=` 10^9

440. K-th Smallest in Lexicographical Order

```
/*
   Brute force:Get all numbers in lexicographical order
   until reaching the k-th
   Time complexity: O(n) - TLE
   Space complexity: O(1)
*/
class Solution {
   public:
      int findKthNumber(int n, int k) {
         long long cur=1;
         k--; // We have already 1.
         while(k){
            k--;
            if(cur*10>n){
               while(cur==n || cur%10==9) cur/=10;
               cur++;
            }
            else cur*=10;
         }
         return cur;
      }
};
```

440. K-th Smallest in Lexicographical Order

```
/*
    Optimization: lust look in the subtree that
    contains the k-th number
    Time complexity: O((10 log_10 n)^2) - AC
    Space complexity: O(1)
*/
class Solution {
    public:
        int count_nodes(long long cur,int n){
            int cnt=0;
            long long next=cur+1;
            while(cur<=n){
                cnt+=std::min(n*1ll-cur+1*1ll,next-cur);
                cur*=10;
                next*=10;
            }
            return cnt;
        }

        int findKthNumber(int n, int k) {
            long long cur=1;
            k--; // We have already 1.
            while(k){
                int cnt=count_nodes(cur,n);
                if(cnt<=k){
                    cur++;
                    k-=cnt;
                }
                else {
                    cur*=10;
                    k--;
                }
            }
            return cur;
        }
};
```