

Problem 753: Cracking the Safe

Problem Information

Difficulty: Hard

Acceptance Rate: 58.28%

Paid Only: No

Tags: Depth-First Search, Graph, Eulerian Circuit

Problem Description

There is a safe protected by a password. The password is a sequence of `n` digits where each digit can be in the range `[0, k - 1]`.

The safe has a peculiar way of checking the password. When you enter in a sequence, it checks the **most recent** n **digits** that were entered each time you type a digit.

* For example, the correct password is "345" and you enter in "012345": * After typing '0', the most recent '3' digits is "0", which is incorrect. * After typing '1', the most recent '3' digits is "01", which is incorrect. * After typing '2', the most recent '3' digits is "012", which is incorrect. * After typing '3', the most recent '3' digits is "123", which is incorrect. * After typing '4', the most recent '3' digits is "234", which is incorrect. * After typing '5', the most recent '3' digits is "345", which is correct and the safe unlocks.

Return any string of **minimum length** that will unlock the safe **at some point** of entering it.

Example 1:

Input: $n = 1$, $k = 2$ **Output:** "10" **Explanation:** The password is a single digit, so enter each digit. "01" would also unlock the safe.

Example 2:

Input: $n = 2$, $k = 2$ **Output:** "01100" **Explanation:** For each possible password: - "00" is typed in starting from the 4th digit. - "01" is typed in starting from the 1st digit. - "10" is typed in starting from the 3rd digit. - "11" is typed in starting from the 2nd digit. Thus "01100" will

unlock the safe. "10011", and "11001" would also unlock the safe.

****Constraints:****

$1 \leq n \leq 4$ $1 \leq k \leq 10$ $1 \leq kn \leq 4096$

Code Snippets

C++:

```
class Solution {
public:
    string crackSafe(int n, int k) {
        }
    };
}
```

Java:

```
class Solution {
public String crackSafe(int n, int k) {
        }
    }
}
```

Python3:

```
class Solution:
    def crackSafe(self, n: int, k: int) -> str:
```