Cracking the Safe

There is a box protected by a password. The password is `n` digits, where each letter can be one of the first `k` digits `0, 1, ..., k-1`.

You can keep inputting the password, the password will automatically be matched against the last `n` digits entered.

For example, assuming the password is `"345"`, I can open it when I type `"012345"`, but I enter a total of 6 digits.

Please return any string of minimum length that is guaranteed to open the box after the entire string is inputted.

Example 1:

```
Input: n = 1, k = 2
Output: "01"
Note: "10" will be accepted too.
```

Example 2:

```
Input: n = 2, k = 2
Output: "00110"
Note: "01100", "10011", "11001" will be accepted too.
```

Note:

1. `n` will be in the range `[1, 4]`.
2. `k` will be in the range `[1, 10]`.
3. `k^n` will be at most `4096`.

Solution 1

This is kinda greedy approach.
So straight up we can tell that we have k^n combinations of the lock.
So the best way to generate the string is reusing last n-1 digits of previous lock << I can't really prove this but I realized this after writing down some examples.

Hence, we'll use dfs to generate the string with goal is when our string contains all possible combinations.

```
class Solution {
    public String crackSafe(int n, int k) {
        StringBuilder sb = new StringBuilder();
        int total = (int) (Math.pow(k, n));
        for (int i = 0; i < n; i++) sb.append('0');

        Set<String> visited = new HashSet<>();
        visited.add(sb.toString());

        dfs(sb, total, visited, n, k);

        return sb.toString();
    }

    private boolean dfs(StringBuilder sb, int goal, Set<String> visited, int n, int
k) {
        if (visited.size() == goal) return true;
        String prev = sb.substring(sb.length() - n + 1, sb.length());
        for (int i = 0; i < k; i++) {
            String next = prev + i;
            if (!visited.contains(next)) {
                visited.add(next);
                sb.append(i);
                if (dfs(sb, goal, visited, n, k)) return true;
                else {
                    visited.remove(next);
                    sb.delete(sb.length() - 1, sb.length());
                }
            }
        }
        return false;
    }

}
```

written by invalid original link here

Solution 2

https://www.youtube.com/watch?v=iPLQgXUiU14

This video helped a lot, combined with spending some time to reprove it on my own, in my own words!

written by harsh700ca original link here

Solution 3

I tested the solution with all possible test cases, the result seems correct.

Accepted Python Code

```
class Solution(object):
    def crackSafe(self, n, k):
        """
        :type n: int
        :type k: int
        :rtype: str
        """
        ans = "0" * (n - 1)
        visits = set()
        for x in range(k ** n):
            current = ans[-n+1:] if n > 1 else ''
            for y in range(k - 1, -1, -1):
                if current + str(y) not in visits:
                    visits.add(current + str(y))
                    ans += str(y)
                    break
        return ans
```

Test:

```
import collections
so = Solution()
for n in range(1, 5):
    for k in range(1, 11):
        if k ** n > 4096: continue
        ans = so.crackSafe(n, k)
        vset = set(ans[x : x + n] for x in range(len(ans) - n + 1))
        print 'k =', k, ' n =', n
        print len(ans), len(vset), sorted(collections.Counter(ans).items()), '\n'
```

See also: http://bookshadow.com/weblog/2017/12/24/leetcode-open-the-lock/

written by 在线疯狂 original link here