Cracking the safe

### 753. 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.

题意分析：问题可以转化为 找出一个字符串，这个字符串包含0到k-1这么多个数字的n位排列

我们只需要着重看两位。用dfs即可

每次我们尝试往Str加入一位数字。如果加入的这个数字与前面n-1为数字，这个排列已经出现过，这个数字没必要加入去。如果没出现，就可以继续往后面dfs。至于为什么这种解法是最优。连楼主都不知道

**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**;

}

}