Problem 967: Numbers With Same Consecutive Differences

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

Given two integers n and k, return

an array of all the integers of length

n

where the difference between every two consecutive digits is

k

. You may return the answer in

any order

.

Note that the integers should not have leading zeros. Integers as

02

and

043

are not allowed.

Example 1:

Input:

n = 3, k = 7

Output:

[181,292,707,818,929]

Explanation:

Note that 070 is not a valid number, because it has leading zeroes.

Example 2:

Input:

n = 2, k = 1

Output:

[10,12,21,23,32,34,43,45,54,56,65,67,76,78,87,89,98]

Constraints:

2 <= n <= 9

0 <= k <= 9

Code Snippets

C++:

```
class Solution {
public:
vector<int> numsSameConsecDiff(int n, int k) {
}
};
```

Java:

```
class Solution {
public int[] numsSameConsecDiff(int n, int k) {

}
}
```

Python3:

```
class Solution:
def numsSameConsecDiff(self, n: int, k: int) -> List[int]:
```

Python:

```
class Solution(object):
def numsSameConsecDiff(self, n, k):
"""
:type n: int
:type k: int
:rtype: List[int]
"""
```

JavaScript:

```
/**
* @param {number} n
* @param {number} k
* @return {number[]}
*/
var numsSameConsecDiff = function(n, k) {

};
```

TypeScript:

```
function numsSameConsecDiff(n: number, k: number): number[] {

};
```

C#:

```
public class Solution {
public int[] NumsSameConsecDiff(int n, int k) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* numsSameConsecDiff(int n, int k, int* returnSize) {

}
```

Go:

```
func numsSameConsecDiff(n int, k int) []int {

}
```

Kotlin:

```
class Solution {
fun numsSameConsecDiff(n: Int, k: Int): IntArray {

}
}
```

Swift:

```
class Solution {
func numsSameConsecDiff(_ n: Int, _ k: Int) -> [Int] {

}
}
```

Rust:

```
impl Solution {
pub fn nums_same_consec_diff(n: i32, k: i32) -> Vec<i32> {
}
}
```

Ruby:

```
# @param {Integer} n
# @param {Integer} k
# @return {Integer[]}
def nums_same_consec_diff(n, k)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @param Integer $k
* @return Integer[]
*/
function numsSameConsecDiff($n, $k) {

}
}
```

Dart:

```
class Solution {
List<int> numsSameConsecDiff(int n, int k) {

}
}
```

Scala:

```
object Solution {
def numsSameConsecDiff(n: Int, k: Int): Array[Int] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec nums_same_consec_diff(n :: integer, k :: integer) :: [integer]
def nums_same_consec_diff(n, k) do

end
end
```

Erlang:

```
-spec nums_same_consec_diff(N :: integer(), K :: integer()) -> [integer()].
nums_same_consec_diff(N, K) ->
.
```

Racket:

```
(define/contract (nums-same-consec-diff n k)
(-> exact-integer? exact-integer? (listof exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Numbers With Same Consecutive Differences
* Difficulty: Medium
* Tags: array, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
vector<int> numsSameConsecDiff(int n, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Numbers With Same Consecutive Differences
* Difficulty: Medium
* Tags: array, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int[] numsSameConsecDiff(int n, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Numbers With Same Consecutive Differences
Difficulty: Medium
Tags: array, search

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def numsSameConsecDiff(self, n: int, k: int) -> List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def numsSameConsecDiff(self, n, k):
"""
:type n: int
:type k: int
```

```
:rtype: List[int]
"""
```

JavaScript Solution:

```
/**
* Problem: Numbers With Same Consecutive Differences
* Difficulty: Medium
* Tags: array, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number} n
* @param {number} k
* @return {number[]}
*/
var numsSameConsecDiff = function(n, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Numbers With Same Consecutive Differences
* Difficulty: Medium
* Tags: array, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function numsSameConsecDiff(n: number, k: number): number[] {

};
```

C# Solution:

```
/*
* Problem: Numbers With Same Consecutive Differences
* Difficulty: Medium
* Tags: array, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int[] NumsSameConsecDiff(int n, int k) {

}
}
```

C Solution:

```
/*
* Problem: Numbers With Same Consecutive Differences
* Difficulty: Medium
* Tags: array, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* numsSameConsecDiff(int n, int k, int* returnSize) {

}
```

Go Solution:

```
// Problem: Numbers With Same Consecutive Differences
// Difficulty: Medium
// Tags: array, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func numsSameConsecDiff(n int, k int) []int {

}
```

Kotlin Solution:

```
class Solution {
fun numsSameConsecDiff(n: Int, k: Int): IntArray {

}
}
```

Swift Solution:

```
class Solution {
func numsSameConsecDiff(_ n: Int, _ k: Int) -> [Int] {

}
}
```

Rust Solution:

```
// Problem: Numbers With Same Consecutive Differences
// Difficulty: Medium
// Tags: array, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn nums_same_consec_diff(n: i32, k: i32) -> Vec<i32> {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @param {Integer} k
# @return {Integer[]}
def nums_same_consec_diff(n, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @param Integer $k
* @return Integer[]
*/
function numsSameConsecDiff($n, $k) {

}
}
```

Dart Solution:

```
class Solution {
List<int> numsSameConsecDiff(int n, int k) {

}
}
```

Scala Solution:

```
object Solution {
def numsSameConsecDiff(n: Int, k: Int): Array[Int] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec nums_same_consec_diff(n :: integer, k :: integer) :: [integer]
def nums_same_consec_diff(n, k) do
```

```



end
end
```

Erlang Solution:

```
-spec nums_same_consec_diff(N :: integer(), K :: integer()) -> [integer()].
nums_same_consec_diff(N, K) ->
.
```

Racket Solution:

```
(define/contract (nums-same-consec-diff n k)
(-> exact-integer? exact-integer? (listof exact-integer?))
)
```