Problem 60: Permutation Sequence

Problem Information

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

Problem Description

The set

[1, 2, 3, ..., n]

contains a total of

n!

unique permutations.

By listing and labeling all of the permutations in order, we get the following sequence for

n = 3

:

"123"

"132"

"213"

"231"

"312"

"321"

Given

n

and

k

, return the

k

th

permutation sequence.

Example 1:

Input:

n = 3, k = 3

Output:

"213"

Example 2:

Input:

n = 4, k = 9

Output:

"2314"

Example 3:

Input:

n = 3, k = 1

Output:

"123"

Constraints:

1 <= n <= 9

1 <= k <= n!

Code Snippets

C++:

```
class Solution {
public:
string getPermutation(int n, int k) {

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

```
class Solution(object):
def getPermutation(self, n, k):
"""
:type n: int
:type k: int
:rtype: str
"""
```

JavaScript:

```
/**
* @param {number} n
* @param {number} k
* @return {string}
*/
var getPermutation = function(n, k) {

};
```

TypeScript:

```
function getPermutation(n: number, k: number): string {

};
```

C#:

```
public class Solution {
public string GetPermutation(int n, int k) {

}
}
```

C:

```
char* getPermutation(int n, int k) {

}
```

Go:

```
func getPermutation(n int, k int) string {
}
```

Kotlin:

```
class Solution {
fun getPermutation(n: Int, k: Int): String {

}
}
```

Swift:

```
class Solution {
func getPermutation(_ n: Int, _ k: Int) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn get_permutation(n: i32, k: i32) -> String {

}
}
```

Ruby:

```
# @param {Integer} n
# @param {Integer} k
# @return {String}
def get_permutation(n, k)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @param Integer $k
* @return String
*/
function getPermutation($n, $k) {

}
}
```

Dart:

```
class Solution {
String getPermutation(int n, int k) {

}
}
```

Scala:

```
object Solution {
def getPermutation(n: Int, k: Int): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec get_permutation(n :: integer, k :: integer) :: String.t
def get_permutation(n, k) do

end
end
```

Erlang:

```
-spec get_permutation(N :: integer(), K :: integer()) ->
unicode:unicode_binary().
get_permutation(N, K) ->
.
```

Racket:

```
(define/contract (get-permutation n k)
(-> exact-integer? exact-integer? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Permutation Sequence
* Difficulty: Hard
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
string getPermutation(int n, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Permutation Sequence
* Difficulty: Hard
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public String getPermutation(int n, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Permutation Sequence
Difficulty: Hard
Tags: math

Approach: Optimized algorithm based on problem constraints
Time Complexity: O(n) to O(n^2) depending on approach
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def getPermutation(self, n: int, k: int) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def getPermutation(self, n, k):
"""
:type n: int
:type k: int
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Permutation Sequence
* Difficulty: Hard
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/
```

```
/**
* @param {number} n
* @param {number} k
* @return {string}
*/
var getPermutation = function(n, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Permutation Sequence
* Difficulty: Hard
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

function getPermutation(n: number, k: number): string {

};
```

C# Solution:

```
/*
* Problem: Permutation Sequence
* Difficulty: Hard
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public string GetPermutation(int n, int k) {

}
}
```

C Solution:

```
/*
* Problem: Permutation Sequence
* Difficulty: Hard
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

char* getPermutation(int n, int k) {

}
```

Go Solution:

```
// Problem: Permutation Sequence
// Difficulty: Hard
// Tags: math
//
// Approach: Optimized algorithm based on problem constraints
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(1) to O(n) depending on approach

func getPermutation(n int, k int) string {

}
```

Kotlin Solution:

```
class Solution {
fun getPermutation(n: Int, k: Int): String {

}
}
```

Swift Solution:

```
class Solution {
func getPermutation(_ n: Int, _ k: Int) -> String {

}
}
```

Rust Solution:

```
// Problem: Permutation Sequence
// Difficulty: Hard
// Tags: math
//
// Approach: Optimized algorithm based on problem constraints
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn get_permutation(n: i32, k: i32) -> String {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @param {Integer} k
# @return {String}
def get_permutation(n, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @param Integer $k
* @return String
*/
function getPermutation($n, $k) {

}
}
```

Dart Solution:

```
class Solution {
String getPermutation(int n, int k) {

}
}
```

Scala Solution:

```
object Solution {
def getPermutation(n: Int, k: Int): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec get_permutation(n :: integer, k :: integer) :: String.t
def get_permutation(n, k) do

end
end
```

Erlang Solution:

```
-spec get_permutation(N :: integer(), K :: integer()) ->
unicode:unicode_binary().
get_permutation(N, K) ->
.
```

Racket Solution:

```
(define/contract (get-permutation n k)
(-> exact-integer? exact-integer? string?)
)
```