Problem 1850: Minimum Adjacent Swaps to Reach the Kth Smallest Number

Problem Information

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

Problem Description

You are given a string

num

, representing a large integer, and an integer

k

.

We call some integer

wonderful

if it is a

permutation

of the digits in

num

and is

greater in value

than

num

. There can be many wonderful integers. However, we only care about the

smallest-valued

ones.

For example, when

num = "5489355142"

:

The 1

st

smallest wonderful integer is

"5489355214"

.

The 2

nd

smallest wonderful integer is

"5489355241"

.

The 3

rd

smallest wonderful integer is

"5489355412"

.

The 4

th

smallest wonderful integer is

"5489355421"

.

Return

the

minimum number of adjacent digit swaps

that needs to be applied to

num

to reach the

k

th

smallest wonderful

integer

.

The tests are generated in such a way that

k

th

smallest wonderful integer exists.

Example 1:

Input:

num = "5489355142", k = 4

Output:

2

Explanation:

The 4

th

smallest wonderful number is "5489355421". To get this number: - Swap index 7 with index 8: "5489355

14

2" -> "5489355

41

2" - Swap index 8 with index 9: "54893554

12

" -> "54893554

21

"

Example 2:

Input:

num = "11112", k = 4

Output:

4

Explanation:

The 4

th

smallest wonderful number is "21111". To get this number: - Swap index 3 with index 4: "111

12

" -> "111

21

" - Swap index 2 with index 3: "11

12

1" -> "11

21

1" - Swap index 1 with index 2: "1

12

11" -> "1

21

11" - Swap index 0 with index 1: "

12

111" -> "

21

111"

Example 3:

Input:

num = "00123", k = 1

Output:

1

Explanation:

The 1

st

smallest wonderful number is "00132". To get this number: - Swap index 3 with index 4: "001

23

" -> "001

32

"

Constraints:

2 <= num.length <= 1000

1 <= k <= 1000

num

only consists of digits.

Code Snippets

C++:

```
class Solution {
public:
int getMinSwaps(string num, int k) {

}
};
```

Java:

```
class Solution {
public int getMinSwaps(String num, int k) {

}
}
```

Python3:

```
class Solution:
def getMinSwaps(self, num: str, k: int) -> int:
```

Python:

```
class Solution(object):
def getMinSwaps(self, num, k):
```

```
"""
:type num: str
:type k: int
:rtype: int
"""
```

JavaScript:

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

};
```

TypeScript:

```
function getMinSwaps(num: string, k: number): number {

};
```

C#:

```
public class Solution {
public int GetMinSwaps(string num, int k) {

}
}
```

C:

```
int getMinSwaps(char* num, int k) {

}
```

Go:

```
func getMinSwaps(num string, k int) int {

}
```

Kotlin:

```
class Solution {
fun getMinSwaps(num: String, k: Int): Int {

}
}
```

Swift:

```
class Solution {
func getMinSwaps(_ num: String, _ k: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn get_min_swaps(num: String, k: i32) -> i32 {

}
}
```

Ruby:

```
# @param {String} num
# @param {Integer} k
# @return {Integer}
def get_min_swaps(num, k)

end
```

PHP:

```
class Solution {

/**
* @param String $num
* @param Integer $k
* @return Integer
*/
function getMinSwaps($num, $k) {
}
}
```

Dart:

```
class Solution {
int getMinSwaps(String num, int k) {

}
}
```

Scala:

```
object Solution {
def getMinSwaps(num: String, k: Int): Int = {

}
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec get_min_swaps(Num :: unicode:unicode_binary(), K :: integer()) ->
integer().
get_min_swaps(Num, K) ->
.
```

Racket:

```
(define/contract (get-min-swaps num k)
(-> string? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Adjacent Swaps to Reach the Kth Smallest Number
* Difficulty: Medium
* Tags: array, string, greedy
*
* 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 getMinSwaps(string num, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Adjacent Swaps to Reach the Kth Smallest Number
* Difficulty: Medium
* Tags: array, string, greedy
*
* 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 getMinSwaps(String num, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Adjacent Swaps to Reach the Kth Smallest Number
Difficulty: Medium
Tags: array, string, greedy

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 getMinSwaps(self, num: str, k: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum Adjacent Swaps to Reach the Kth Smallest Number
* Difficulty: Medium
* Tags: array, string, greedy
*
* 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 {string} num
* @param {number} k
* @return {number}
*/
```

```
var getMinSwaps = function(num, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Adjacent Swaps to Reach the Kth Smallest Number
* Difficulty: Medium
* Tags: array, string, greedy
*
* 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 getMinSwaps(num: string, k: number): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Adjacent Swaps to Reach the Kth Smallest Number
* Difficulty: Medium
* Tags: array, string, greedy
*
* 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 GetMinSwaps(string num, int k) {

}
}
```

C Solution:

```c
/*
* Problem: Minimum Adjacent Swaps to Reach the Kth Smallest Number
* Difficulty: Medium
* Tags: array, string, greedy
*
* 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
*/

int getMinSwaps(char* num, int k) {

}
```

Go Solution:

```go
// Problem: Minimum Adjacent Swaps to Reach the Kth Smallest Number
// Difficulty: Medium
// Tags: array, string, greedy
//
// 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 getMinSwaps(num string, k int) int {

}
```

Kotlin Solution:

```kotlin
class Solution {
fun getMinSwaps(num: String, k: Int): Int {

}
}
```

Swift Solution:

```swift
class Solution {
func getMinSwaps(_ num: String, _ k: Int) -> Int {

}

}
```

Rust Solution:

```rust
// Problem: Minimum Adjacent Swaps to Reach the Kth Smallest Number
// Difficulty: Medium
// Tags: array, string, greedy
//
// 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 get_min_swaps(num: String, k: i32) -> i32 {

}
}
```

Ruby Solution:

```ruby
# @param {String} num
# @param {Integer} k
# @return {Integer}
def get_min_swaps(num, k)

end
```

PHP Solution:

```php
class Solution {

/**
* @param String $num
* @param Integer $k
* @return Integer
*/
function getMinSwaps($num, $k) {

}
}
```

Dart Solution:

```
class Solution {
int getMinSwaps(String num, int k) {

}
}
```

Scala Solution:

```
object Solution {
def getMinSwaps(num: String, k: Int): Int = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec get_min_swaps(Num :: unicode:unicode_binary(), K :: integer()) ->
integer().
get_min_swaps(Num, K) ->
.
```

Racket Solution:

```
(define/contract (get-min-swaps num k)
(-> string? exact-integer? exact-integer?)
)
```