Problem 1505: Minimum Possible Integer After at Most K Adjacent Swaps On Digits

Problem Information

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

Problem Description

You are given a string

num

representing

the digits

of a very large integer and an integer

k

. You are allowed to swap any two adjacent digits of the integer

at most

k

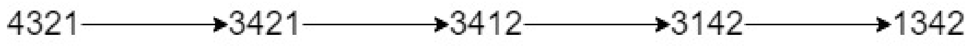
times.

Return

the minimum integer you can obtain also as a string

.

Example 1:



Input:

num = "4321", k = 4

Output:

"1342"

Explanation:

The steps to obtain the minimum integer from 4321 with 4 adjacent swaps are shown.

Example 2:

Input:

num = "100", k = 1

Output:

"010"

Explanation:

It's ok for the output to have leading zeros, but the input is guaranteed not to have any leading zeros.

Example 3:

Input:

num = "36789", k = 1000

Output:

"36789"

Explanation:

We can keep the number without any swaps.

Constraints:

1 <= num.length <= 3 * 10

4

num

consists of only

digits

and does not contain

leading zeros

.

1 <= k <= 10

9

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

}

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

```
end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Minimum Possible Integer After at Most K Adjacent Swaps On Digits
* Difficulty: Hard
* Tags: string, tree, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Minimum Possible Integer After at Most K Adjacent Swaps On Digits
* Difficulty: Hard
* Tags: string, tree, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public String minInteger(String num, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Possible Integer After at Most K Adjacent Swaps On Digits
Difficulty: Hard
Tags: string, tree, greedy

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum Possible Integer After at Most K Adjacent Swaps On Digits
* Difficulty: Hard
* Tags: string, tree, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Possible Integer After at Most K Adjacent Swaps On Digits
* Difficulty: Hard
* Tags: string, tree, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

function minInteger(num: string, k: number): string {

};
```

C# Solution:

```
/*
* Problem: Minimum Possible Integer After at Most K Adjacent Swaps On Digits
* Difficulty: Hard
* Tags: string, tree, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

public class Solution {
public string MinInteger(string num, int k) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Possible Integer After at Most K Adjacent Swaps On Digits
* Difficulty: Hard
* Tags: string, tree, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

char* minInteger(char* num, int k) {

}
```

Go Solution:

```
// Problem: Minimum Possible Integer After at Most K Adjacent Swaps On Digits
// Difficulty: Hard
// Tags: string, tree, greedy
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func minInteger(num string, k int) string {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Minimum Possible Integer After at Most K Adjacent Swaps On Digits
// Difficulty: Hard
// Tags: string, tree, greedy
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

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

}
}
```

Ruby Solution:

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

```
end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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