Problem 402: Remove K Digits

Problem Information

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

Problem Description

Given string num representing a non-negative integer

num

, and an integer

k

, return

the smallest possible integer after removing

k

digits from

num

.

Example 1:

Input:

num = "1432219", k = 3

Output:

"1219"

Explanation:

Remove the three digits 4, 3, and 2 to form the new number 1219 which is the smallest.

Example 2:

Input:

num = "10200", k = 1

Output:

"200"

Explanation:

Remove the leading 1 and the number is 200. Note that the output must not contain leading zeroes.

Example 3:

Input:

num = "10", k = 2

Output:

"0"

Explanation:

Remove all the digits from the number and it is left with nothing which is 0.

Constraints:

1 <= k <= num.length <= 10

5

num

consists of only digits.

num

does not have any leading zeros except for the zero itself.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

```
}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Remove K Digits
* Difficulty: Medium
* Tags: string, greedy, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Remove K Digits
* Difficulty: Medium
* Tags: string, greedy, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
}
```

Python3 Solution:

```
"""
Problem: Remove K Digits
Difficulty: Medium
Tags: string, greedy, stack

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Remove K Digits
* Difficulty: Medium
* Tags: string, greedy, stack
*
* Approach: String manipulation with hash map or two pointers
* 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 {string}
*/
```

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

};
```

TypeScript Solution:

```
/**
* Problem: Remove K Digits
* Difficulty: Medium
* Tags: string, greedy, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

};
```

C# Solution:

```
/*
* Problem: Remove K Digits
* Difficulty: Medium
* Tags: string, greedy, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
}
```

C Solution:

```c
/*
* Problem: Remove K Digits
* Difficulty: Medium
* Tags: string, greedy, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
```

Go Solution:

```go
// Problem: Remove K Digits
// Difficulty: Medium
// Tags: string, greedy, stack
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func removeKdigits(num string, k int) string {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
```

```
}
```

Rust Solution:

```
// Problem: Remove K Digits
// Difficulty: Medium
// Tags: string, greedy, stack
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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