Problem 2343: Query Kth Smallest Trimmed Number

Problem Information

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

Problem Description

You are given a

0-indexed

array of strings

nums

, where each string is of

equal length

and consists of only digits.

You are also given a

0-indexed

2D integer array

queries

where

queries[i] = [k

i

, trim

i

]

. For each

queries[i]

, you need to:

Trim

each number in

nums

to its

rightmost

trim

i

digits.

Determine the

index

of the

k

i

th

smallest trimmed number in

nums

. If two trimmed numbers are equal, the number with the

lower

index is considered to be smaller.

Reset each number in

nums

to its original length.

Return

an array

answer

of the same length as

queries

,

where

answer[i]

is the answer to the

i

th

query.

Note

:

To trim to the rightmost

x

digits means to keep removing the leftmost digit, until only

x

digits remain.

Strings in

nums

may contain leading zeros.

Example 1:

Input:

nums = ["102","473","251","814"], queries = [[1,1],[2,3],[4,2],[1,2]]

Output:

[2,2,1,0]

Explanation:

1. After trimming to the last digit, nums = ["2","3","1","4"]. The smallest number is 1 at index 2.
2. Trimmed to the last 3 digits, nums is unchanged. The 2

nd

smallest number is 251 at index 2. 3. Trimmed to the last 2 digits, nums = ["02","73","51","14"]. The 4

th

smallest number is 73. 4. Trimmed to the last 2 digits, the smallest number is 2 at index 0. Note that the trimmed number "02" is evaluated as 2.

Example 2:

Input:

nums = ["24","37","96","04"], queries = [[2,1],[2,2]]

Output:

[3,0]

Explanation:

1. Trimmed to the last digit, nums = ["4","7","6","4"]. The 2

nd

smallest number is 4 at index 3. There are two occurrences of 4, but the one at index 0 is considered smaller than the one at index 3. 2. Trimmed to the last 2 digits, nums is unchanged. The 2

nd

smallest number is 24.

Constraints:

1 <= nums.length <= 100

1 <= nums[i].length <= 100

nums[i]

consists of only digits.

All

nums[i].length

are

equal

.

1 <= queries.length <= 100

queries[i].length == 2

1 <= k

i

<= nums.length

1 <= trim

i

<= nums[i].length

Follow up:

Could you use the

Radix Sort Algorithm

to solve this problem? What will be the complexity of that solution?

Code Snippets

C++:

```
class Solution {
public:
vector<int> smallestTrimmedNumbers(vector<string>& nums, vector<vector<int>>&
queries) {

}
};
```

Java:

```
class Solution {
public int[] smallestTrimmedNumbers(String[] nums, int[][] queries) {

}
}
```

Python3:

```
class Solution:
def smallestTrimmedNumbers(self, nums: List[str], queries: List[List[int]])
-> List[int]:
```

Python:

```
class Solution(object):
def smallestTrimmedNumbers(self, nums, queries):
"""
:type nums: List[str]
:type queries: List[List[int]]
:rtype: List[int]
"""
```

JavaScript:

```
/**
* @param {string[]} nums
* @param {number[][]} queries
* @return {number[]}
```

```
*/
var smallestTrimmedNumbers = function(nums, queries) {

};
```

TypeScript:

```
function smallestTrimmedNumbers(nums: string[], queries: number[][]):
number[] {

};
```

C#:

```
public class Solution {
public int[] SmallestTrimmedNumbers(string[] nums, int[][] queries) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* smallestTrimmedNumbers(char** nums, int numsSize, int** queries, int
queriesSize, int* queriesColSize, int* returnSize) {

}
```

Go:

```
func smallestTrimmedNumbers(nums []string, queries [][]int) []int {

}
```

Kotlin:

```
class Solution {
fun smallestTrimmedNumbers(nums: Array<String>, queries: Array<IntArray>):
IntArray {
```

```
}
}
```

Swift:

```
class Solution {
func smallestTrimmedNumbers(_ nums: [String], _ queries: [[Int]]) -> [Int] {

}
}
```

Rust:

```
impl Solution {
pub fn smallest_trimmed_numbers(nums: Vec<String>, queries: Vec<Vec<i32>>) ->
Vec<i32> {

}
}
```

Ruby:

```
# @param {String[]} nums
# @param {Integer[][]} queries
# @return {Integer[]}
def smallest_trimmed_numbers(nums, queries)

end
```

PHP:

```
class Solution {

/**
* @param String[] $nums
* @param Integer[][] $queries
* @return Integer[]
*/
function smallestTrimmedNumbers($nums, $queries) {

}
}
```

Dart:

```
class Solution {
List<int> smallestTrimmedNumbers(List<String> nums, List<List<int>> queries)
{

}
}
```

Scala:

```
object Solution {
def smallestTrimmedNumbers(nums: Array[String], queries: Array[Array[Int]]):
Array[Int] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec smallest_trimmed_numbers(nums :: [String.t], queries :: [[integer]]) ::
[integer]
def smallest_trimmed_numbers(nums, queries) do

end
end
```

Erlang:

```
-spec smallest_trimmed_numbers(Nums :: [unicode:unicode_binary()], Queries ::
[[integer()]]) -> [integer()].
smallest_trimmed_numbers(Nums, Queries) ->
.
```

Racket:

```
(define/contract (smallest-trimmed-numbers nums queries)
(-> (listof string?) (listof (listof exact-integer?)) (listof
exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Query Kth Smallest Trimmed Number
* Difficulty: Medium
* Tags: array, string, sort, queue, heap
*
* 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> smallestTrimmedNumbers(vector<string>& nums, vector<vector<int>>&
queries) {

}
};
```

Java Solution:

```
/**
* Problem: Query Kth Smallest Trimmed Number
* Difficulty: Medium
* Tags: array, string, sort, queue, heap
*
* 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[] smallestTrimmedNumbers(String[] nums, int[][] queries) {

}
}
```

Python3 Solution:

```
"""
Problem: Query Kth Smallest Trimmed Number
Difficulty: Medium
Tags: array, string, sort, queue, heap

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 smallestTrimmedNumbers(self, nums: List[str], queries: List[List[int]])
-> List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def smallestTrimmedNumbers(self, nums, queries):
"""
:type nums: List[str]
:type queries: List[List[int]]
:rtype: List[int]
"""
```

JavaScript Solution:

```
/**
* Problem: Query Kth Smallest Trimmed Number
* Difficulty: Medium
* Tags: array, string, sort, queue, heap
*
* 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[]} nums
* @param {number[][]} queries
* @return {number[]}
```

```
*/
var smallestTrimmedNumbers = function(nums, queries) {

};
```

TypeScript Solution:

```
/**
* Problem: Query Kth Smallest Trimmed Number
* Difficulty: Medium
* Tags: array, string, sort, queue, heap
*
* 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 smallestTrimmedNumbers(nums: string[], queries: number[][]):
number[] {

};
```

C# Solution:

```
/*
* Problem: Query Kth Smallest Trimmed Number
* Difficulty: Medium
* Tags: array, string, sort, queue, heap
*
* 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[] SmallestTrimmedNumbers(string[] nums, int[][] queries) {

}
}
```

C Solution:

```
/*
* Problem: Query Kth Smallest Trimmed Number
* Difficulty: Medium
* Tags: array, string, sort, queue, heap
*
* 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* smallestTrimmedNumbers(char** nums, int numsSize, int** queries, int
queriesSize, int* queriesColSize, int* returnSize) {

}
```

Go Solution:

```
// Problem: Query Kth Smallest Trimmed Number
// Difficulty: Medium
// Tags: array, string, sort, queue, heap
//
// 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 smallestTrimmedNumbers(nums []string, queries [][]int) []int {

}
```

Kotlin Solution:

```
class Solution {
fun smallestTrimmedNumbers(nums: Array<String>, queries: Array<IntArray>):
IntArray {

}
}
```

Swift Solution:

```
class Solution {
func smallestTrimmedNumbers(_ nums: [String], _ queries: [[Int]]) -> [Int] {

}
}
```

Rust Solution:

```
// Problem: Query Kth Smallest Trimmed Number
// Difficulty: Medium
// Tags: array, string, sort, queue, heap
//
// 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 smallest_trimmed_numbers(nums: Vec<String>, queries: Vec<Vec<i32>>) ->
Vec<i32> {

}
}
```

Ruby Solution:

```
# @param {String[]} nums
# @param {Integer[][]} queries
# @return {Integer[]}
def smallest_trimmed_numbers(nums, queries)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[] $nums
* @param Integer[][] $queries
* @return Integer[]
*/
function smallestTrimmedNumbers($nums, $queries) {
```

```
}
}
```

Dart Solution:

```
class Solution {
List<int> smallestTrimmedNumbers(List<String> nums, List<List<int>> queries)
{

}
}
```

Scala Solution:

```
object Solution {
def smallestTrimmedNumbers(nums: Array[String], queries: Array[Array[Int]]):
Array[Int] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec smallest_trimmed_numbers(nums :: [String.t], queries :: [[integer]]) ::
[integer]
def smallest_trimmed_numbers(nums, queries) do

end
end
```

Erlang Solution:

```
-spec smallest_trimmed_numbers(Nums :: [unicode:unicode_binary()], Queries ::
[[integer()]]) -> [integer()].
smallest_trimmed_numbers(Nums, Queries) ->
.
```

Racket Solution:

```
(define/contract (smallest-trimmed-numbers nums queries)
(-> (listof string?) (listof (listof exact-integer?)) (listof
exact-integer?))
)
```