Problem 2782: Number of Unique Categories

Problem Information

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

Problem Description

You are given an integer

n

and an object

categoryHandler

of class

CategoryHandler

.

There are

n

elements, numbered from

0

to

n - 1

. Each element has a category, and your task is to find the number of unique categories.

The class

CategoryHandler

contains the following function, which may help you:

boolean haveSameCategory(integer a, integer b)

: Returns

true

if

a

and

b

are in the same category and

false

otherwise. Also, if either

a

or

b

is not a valid number (i.e. it's greater than or equal to

n

or less than

0

), it returns

false

.

Return

the number of unique categories.

Example 1:

Input:

n = 6, categoryHandler = [1,1,2,2,3,3]

Output:

3

Explanation:

There are 6 elements in this example. The first two elements belong to category 1, the second two belong to category 2, and the last two elements belong to category 3. So there are 3 unique categories.

Example 2:

Input:

n = 5, categoryHandler = [1,2,3,4,5]

Output:

5

Explanation:

There are 5 elements in this example. Each element belongs to a unique category. So there are 5 unique categories.

Example 3:

Input:

n = 3, categoryHandler = [1,1,1]

Output:

1

Explanation:

There are 3 elements in this example. All of them belong to one category. So there is only 1 unique category.

Constraints:

1 <= n <= 100

Code Snippets

C++:

```
/**
* Definition for a category handler.
* class CategoryHandler {
* public:
* CategoryHandler(vector<int> categories);
* bool haveSameCategory(int a, int b);
* };
*/
class Solution {
public:
int numberOfCategories(int n, CategoryHandler* categoryHandler) {

}
```

```
};
```

Java:

```
/**
* Definition for a category handler.
* class CategoryHandler {
* public CategoryHandler(int[] categories);
* public boolean haveSameCategory(int a, int b);
* };
*/
class Solution {
public int numberOfCategories(int n, CategoryHandler categoryHandler) {

}
}
```

Python3:

```
# Definition for a category handler.
# class CategoryHandler:
# def haveSameCategory(self, a: int, b: int) -> bool:
# pass
class Solution:
def numberOfCategories(self, n: int, categoryHandler:
Optional['CategoryHandler']) -> int:
```

Python:

```
# Definition for a category handler.
# class CategoryHandler:
# def haveSameCategory(self, a, b):
# pass
class Solution(object):
def numberOfCategories(self, n, categoryHandler):
"""
:type n: int
:type categoryHandler: CategoryHandler
:rtype: int
"""
```

JavaScript:

```
/**
* Definition for a category handler.
* class CategoryHandler {
* @param {number[]} categories
* constructor(categories);
*
* @param {number} a
* @param {number} b
* @return {boolean}
* haveSameCategory(a, b);
* }
*/
/**
* @param {number} n
* @param {CategoryHandler} categoryHandler
* @return {number}
*/
var numberOfCategories = function(n, categoryHandler) {

};
```

TypeScript:

```
/**
* Definition for a category handler.
* class CategoryHandler {
* constructor(categories: number[]);
* public haveSameCategory(a: number, b: number): boolean;
* }
*/
function numberOfCategories(n: number, categoryHandler: CategoryHandler):
number {

};
```

C#:

```
/**
* Definition for a category handler.
* class CategoryHandler {
* public CategoryHandler(int[] categories);
* public bool HaveSameCategory(int a, int b);
* }
```

```
*/
public class Solution {
public int NumberOfCategories(int n, CategoryHandler categoryHandler) {

}
}
```

C:

```
/**
* Definition for a category handler.
*
* YOU HAVE TO PASS THE OBJECT ITSELF AS THE FIRST PARAMETER
*
* struct CategoryHandler {
* bool (*haveSameCategory)(struct CategoryHandler*, int, int);
* };
*/
int numberOfCategories(int n, struct CategoryHandler* categoryHandler){

}
```

Go:

```
/**
* Definition for a category handler.
* type CategoryHandler interface {
* HaveSameCategory(int, int) bool
* }
*/
func numberOfCategories(n int, categoryHandler CategoryHandler) int {

}
```

Kotlin:

```
/**
* Definition for a category handler.
* class CategoryHandler(categories: IntArray) {
* fun haveSameCategory(a: Int, b: Int): Boolean
* }
*/
```

```
class Solution {
fun numberOfCategories(n: Int, categoryHandler: CategoryHandler): Int {

}
}
```

Swift:

```
/**
* Definition for a category handler.
* class CategoryHandler {
* init(categories: [Int]) {}
* func haveSameCategory(a: Int, b: Int) -> Bool {}
* }
*/
class Solution {
func numberOfCategories(_ n: Int, _ categoryHandler: CategoryHandler) -> Int
{

}
}
```

Rust:

```
/**
* Definition for a category handler.
* impl CategoryHandler {
* pub fn new(categories: Vec<i32>) -> Self {}
* pub fn have_same_category(&self, a: i32, b: i32) -> bool {}
* }
*/
impl Solution {
pub fn number_of_categories(n: i32, category_handler: CategoryHandler) -> i32
{

}
}
```

Ruby:

```
# Definition for a category handler.
# class CategoryHandler
```

```
# def initialize(categories)
# end
# def have_same_category(a, b)
# end
# end
# @param {Integer} n
# @param {CategoryHandler} category_handler
# @return {Integer}
def number_of_categories(n, category_handler)

end
```

PHP:

```
/**
* Definition for a category handler.
* class CategoryHandler {
* function __construct($categories);
* * @param Integer $a
* * @param Integer $b
* * @return Boolean
* function haveSameCategory($a, $b);
* }
*/
class Solution {

/**
* @param Integer $n
* @param CategoryHandler $categoryHandler
* @return Integer
*/
function numberOfCategories($n, $categoryHandler) {

}
}
```

Dart:

```
/**
* Definition for a category handler.
* class CategoryHandler {
* CategoryHandler(List<int> categories);
```

```
* bool haveSameCategory(int a, int b);
* }
*/
class Solution {
int numberOfCategories(int n, CategoryHandler categoryHandler) {

}
}
```

Scala:

```
/**
* Definition for a category handler.
* class CategoryHandler(categories: Array[Int]) {
* def haveSameCategory(a: Int, b: Int): Boolean
* }
*/
object Solution {
def numberOfCategories(n: Int, categoryHandler: CategoryHandler): Int = {

}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Unique Categories
* Difficulty: Medium
* Tags: graph
*
* 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
*/

/**
* Definition for a category handler.
* class CategoryHandler {
```

```
* public:
* CategoryHandler(vector<int> categories);
* bool haveSameCategory(int a, int b);
* };
*/
class Solution {
public:
int numberOfCategories(int n, CategoryHandler* categoryHandler) {

}
};
```

Java Solution:

```
/**
* Problem: Number of Unique Categories
* Difficulty: Medium
* Tags: graph
*
* 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
*/

/**
* Definition for a category handler.
* class CategoryHandler {
* public CategoryHandler(int[] categories);
* public boolean haveSameCategory(int a, int b);
* };
*/
class Solution {
public int numberOfCategories(int n, CategoryHandler categoryHandler) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Unique Categories
```

```
Difficulty: Medium
Tags: graph

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
"""

# Definition for a category handler.
# class CategoryHandler:
# def haveSameCategory(self, a: int, b: int) -> bool:
# pass
class Solution:
def numberOfCategories(self, n: int, categoryHandler:
Optional['CategoryHandler']) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
# Definition for a category handler.
# class CategoryHandler:
# def haveSameCategory(self, a, b):
# pass
class Solution(object):
def numberOfCategories(self, n, categoryHandler):
"""
:type n: int
:type categoryHandler: CategoryHandler
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Number of Unique Categories
* Difficulty: Medium
* Tags: graph
*
* 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
*/

/**
* Definition for a category handler.
* class CategoryHandler {
* @param {number[]} categories
* constructor(categories);
*
* @param {number} a
* @param {number} b
* @return {boolean}
* haveSameCategory(a, b);
* }
*/
/**
* @param {number} n
* @param {CategoryHandler} categoryHandler
* @return {number}
*/
var numberOfCategories = function(n, categoryHandler) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Unique Categories
* Difficulty: Medium
* Tags: graph
*
* 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
*/

/**
* Definition for a category handler.
* class CategoryHandler {
* constructor(categories: number[]);
* public haveSameCategory(a: number, b: number): boolean;
```

```
* }
*/
function numberOfCategories(n: number, categoryHandler: CategoryHandler):
number {

};
```

C# Solution:

```
/*
* Problem: Number of Unique Categories
* Difficulty: Medium
* Tags: graph
*
* 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
*/

/**
* Definition for a category handler.
* class CategoryHandler {
* public CategoryHandler(int[] categories);
* public bool HaveSameCategory(int a, int b);
* }
*/
public class Solution {
public int NumberOfCategories(int n, CategoryHandler categoryHandler) {

}
}
```

C Solution:

```
/*
* Problem: Number of Unique Categories
* Difficulty: Medium
* Tags: graph
*
* 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
*/

/**
* Definition for a category handler.
*
* YOU HAVE TO PASS THE OBJECT ITSELF AS THE FIRST PARAMETER
*
* struct CategoryHandler {
* bool (*haveSameCategory)(struct CategoryHandler*, int, int);
* };
*/
int numberOfCategories(int n, struct CategoryHandler* categoryHandler){

}
```

Go Solution:

```
// Problem: Number of Unique Categories
// Difficulty: Medium
// Tags: graph
//
// 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

/**
* Definition for a category handler.
* type CategoryHandler interface {
* HaveSameCategory(int, int) bool
* }
*/
func numberOfCategories(n int, categoryHandler CategoryHandler) int {

}
```

Kotlin Solution:

```
/**
* Definition for a category handler.
* class CategoryHandler(categories: IntArray) {
```

```
* fun haveSameCategory(a: Int, b: Int): Boolean
* }
*/
class Solution {
fun numberOfCategories(n: Int, categoryHandler: CategoryHandler): Int {

}
}
```

Swift Solution:

```
/**
* Definition for a category handler.
* class CategoryHandler {
* init(categories: [Int]) {}
* func haveSameCategory(a: Int, b: Int) -> Bool {}
* }
*/
class Solution {
func numberOfCategories(_ n: Int, _ categoryHandler: CategoryHandler) -> Int
{

}
}
```

Rust Solution:

```
// Problem: Number of Unique Categories
// Difficulty: Medium
// Tags: graph
//
// 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

/**
* Definition for a category handler.
* impl CategoryHandler {
* pub fn new(categories: Vec<i32>) -> Self {}
* pub fn have_same_category(&self, a: i32, b: i32) -> bool {}
* }
```

```
*/
impl Solution {
pub fn number_of_categories(n: i32, category_handler: CategoryHandler) -> i32
{

}
}
```

Ruby Solution:

```
# Definition for a category handler.
# class CategoryHandler
# def initialize(categories)
# end
# def have_same_category(a, b)
# end
# end
# @param {Integer} n
# @param {CategoryHandler} category_handler
# @return {Integer}
def number_of_categories(n, category_handler)

end
```

PHP Solution:

```
/**
* Definition for a category handler.
* class CategoryHandler {
* function __construct($categories);
* * @param Integer $a
* * @param Integer $b
* * @return Boolean
* function haveSameCategory($a, $b);
* }
*/
class Solution {

/**
* @param Integer $n
* @param CategoryHandler $categoryHandler
```

```
* @return Integer
*/
function numberOfCategories($n, $categoryHandler) {

}
}
```

Dart Solution:

```
/**
* Definition for a category handler.
* class CategoryHandler {
* CategoryHandler(List<int> categories);
* bool haveSameCategory(int a, int b);
* }
*/
class Solution {
int numberOfCategories(int n, CategoryHandler categoryHandler) {

}
}
```

Scala Solution:

```
/**
* Definition for a category handler.
* class CategoryHandler(categories: Array[Int]) {
* def haveSameCategory(a: Int, b: Int): Boolean
* }
*/
object Solution {
def numberOfCategories(n: Int, categoryHandler: CategoryHandler): Int = {

}
}
```