Problem 2053: Kth Distinct String in an Array

Problem Information

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

Problem Description

A

distinct string

is a string that is present only

once

in an array.

Given an array of strings

arr

, and an integer

k

, return

the

k

th

distinct string

present in

arr

. If there are

fewer

than

k

distinct strings, return

an

empty string

""

.

Note that the strings are considered in the

order in which they appear

in the array.

Example 1:

Input:

arr = ["d","b","c","b","c","a"], k = 2

Output:

"a"

Explanation:

The only distinct strings in arr are "d" and "a". "d" appears 1

st

, so it is the 1

st

distinct string. "a" appears 2

nd

, so it is the 2

nd

distinct string. Since k == 2, "a" is returned.

Example 2:

Input:

arr = ["aaa","aa","a"], k = 1

Output:

"aaa"

Explanation:

All strings in arr are distinct, so the 1

st

string "aaa" is returned.

Example 3:

Input:

arr = ["a","b","a"], k = 3

Output:

""

Explanation:

The only distinct string is "b". Since there are fewer than 3 distinct strings, we return an empty string "".

Constraints:

1 <= k <= arr.length <= 1000

1 <= arr[i].length <= 5

arr[i]

consists of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
string kthDistinct(vector<string>& arr, int k) {

}
};
```

Java:

```
class Solution {
public String kthDistinct(String[] arr, int k) {

}
}
```

Python3:

```
class Solution:
def kthDistinct(self, arr: List[str], k: int) -> str:
```

Python:

```
class Solution(object):
def kthDistinct(self, arr, k):
"""
:type arr: List[str]
:type k: int
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string[]} arr
* @param {number} k
* @return {string}
*/
var kthDistinct = function(arr, k) {

};
```

TypeScript:

```
function kthDistinct(arr: string[], k: number): string {

};
```

C#:

```
public class Solution {
public string KthDistinct(string[] arr, int k) {
```

```
}
}
```

C:

```
char* kthDistinct(char** arr, int arrSize, int k) {

}
```

Go:

```
func kthDistinct(arr []string, k int) string {

}
```

Kotlin:

```
class Solution {
fun kthDistinct(arr: Array<String>, k: Int): String {

}
}
```

Swift:

```
class Solution {
func kthDistinct(_ arr: [String], _ k: Int) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn kth_distinct(arr: Vec<String>, k: i32) -> String {

}
}
```

Ruby:

```
# @param {String[]} arr
# @param {Integer} k
# @return {String}
def kth_distinct(arr, k)

end
```

PHP:

```
class Solution {

/**
* @param String[] $arr
* @param Integer $k
* @return String
*/
function kthDistinct($arr, $k) {

}
}
```

Dart:

```
class Solution {
String kthDistinct(List<String> arr, int k) {

}
}
```

Scala:

```
object Solution {
def kthDistinct(arr: Array[String], k: Int): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec kth_distinct(arr :: [String.t], k :: integer) :: String.t
def kth_distinct(arr, k) do
```

```
end
end
```

Erlang:

```
-spec kth_distinct(Arr :: [unicode:unicode_binary()], K :: integer()) ->
unicode:unicode_binary().
kth_distinct(Arr, K) ->
.
```

Racket:

```
(define/contract (kth-distinct arr k)
(-> (listof string?) exact-integer? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Kth Distinct String in an Array
* Difficulty: Easy
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
string kthDistinct(vector<string>& arr, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Kth Distinct String in an Array
* Difficulty: Easy
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public String kthDistinct(String[] arr, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Kth Distinct String in an Array
Difficulty: Easy
Tags: array, string, hash

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:
def kthDistinct(self, arr: List[str], k: int) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def kthDistinct(self, arr, k):
"""
:type arr: List[str]
:type k: int
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Kth Distinct String in an Array
* Difficulty: Easy
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {string[]} arr
* @param {number} k
* @return {string}
*/
var kthDistinct = function(arr, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Kth Distinct String in an Array
* Difficulty: Easy
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function kthDistinct(arr: string[], k: number): string {

};
```

C# Solution:

```
/*
* Problem: Kth Distinct String in an Array
* Difficulty: Easy
```

```
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public string KthDistinct(string[] arr, int k) {

}
}
```

C Solution:

```
/*
* Problem: Kth Distinct String in an Array
* Difficulty: Easy
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

char* kthDistinct(char** arr, int arrSize, int k) {

}
```

Go Solution:

```
// Problem: Kth Distinct String in an Array
// Difficulty: Easy
// Tags: array, string, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func kthDistinct(arr []string, k int) string {
```

```
}
```

Kotlin Solution:

```
class Solution {
fun kthDistinct(arr: Array<String>, k: Int): String {

}
}
```

Swift Solution:

```
class Solution {
func kthDistinct(_ arr: [String], _ k: Int) -> String {

}
}
```

Rust Solution:

```
// Problem: Kth Distinct String in an Array
// Difficulty: Easy
// Tags: array, string, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn kth_distinct(arr: Vec<String>, k: i32) -> String {

}
}
```

Ruby Solution:

```
# @param {String[]} arr
# @param {Integer} k
# @return {String}
def kth_distinct(arr, k)
```

```
end
```

PHP Solution:

```
class Solution {

/**
* @param String[] $arr
* @param Integer $k
* @return String
*/
function kthDistinct($arr, $k) {

}
}
```

Dart Solution:

```
class Solution {
String kthDistinct(List<String> arr, int k) {

}
}
```

Scala Solution:

```
object Solution {
def kthDistinct(arr: Array[String], k: Int): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec kth_distinct(arr :: [String.t], k :: integer) :: String.t
def kth_distinct(arr, k) do

end
end
```

Erlang Solution:

```
-spec kth_distinct(Arr :: [unicode:unicode_binary()], K :: integer()) ->
unicode:unicode_binary().
kth_distinct(Arr, K) ->
.
```

Racket Solution:

```
(define/contract (kth-distinct arr k)
(-> (listof string?) exact-integer? string?)
)
```