# Problem 1562: Find Latest Group of Size M

## Problem Information

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

## Problem Description

Given an array

arr

that represents a permutation of numbers from

1

to

n

.

You have a binary string of size

n

that initially has all its bits set to zero. At each step

i

(assuming both the binary string and

arr

are 1-indexed) from

1

to

n

, the bit at position

arr[i]

is set to

1

.

You are also given an integer

m

. Find the latest step at which there exists a group of ones of length

m

. A group of ones is a contiguous substring of

1

's such that it cannot be extended in either direction.

Return

the latest step at which there exists a group of ones of length

exactly

m

.

If no such group exists, return

-1

.

Example 1:

Input:

arr = [3,5,1,2,4], m = 1

Output:

4

Explanation:

Step 1: "00

1

00", groups: ["1"] Step 2: "0010

1

", groups: ["1", "1"] Step 3: "

1

0101", groups: ["1", "1", "1"] Step 4: "1

1

101", groups: ["111", "1"] Step 5: "111

1

1", groups: ["11111"] The latest step at which there exists a group of size 1 is step 4.

Example 2:

Input:

arr = [3,1,5,4,2], m = 2

Output:

-1

Explanation:

Step 1: "00

1

00", groups: ["1"] Step 2: "

1

0100", groups: ["1", "1"] Step 3: "1010

1

", groups: ["1", "1", "1"] Step 4: "101

1

1", groups: ["1", "111"] Step 5: "1

1

111", groups: ["11111"] No group of size 2 exists during any step.

Constraints:

n == arr.length

1 <= m <= n <= 10

5

1 <= arr[i] <= n

All integers in

arr

are

distinct

.

## Code Snippets

**C++:**

```
class Solution {
public:
int findLatestStep(vector<int>& arr, int m) {

}
};
```

**Java:**

```
class Solution {
public int findLatestStep(int[] arr, int m) {

}
}
```

**Python3:**

```
class Solution:
def findLatestStep(self, arr: List[int], m: int) -> int:
```

**Python:**

```
class Solution(object):
def findLatestStep(self, arr, m):
"""
:type arr: List[int]
:type m: int
:rtype: int
"""
```

**JavaScript:**

```
/**
* @param {number[]} arr
* @param {number} m
* @return {number}
*/
var findLatestStep = function(arr, m) {

};
```

**TypeScript:**

```
function findLatestStep(arr: number[], m: number): number {

};
```

**C#:**

```
public class Solution {
public int FindLatestStep(int[] arr, int m) {

}
}
```

**C:**

```
int findLatestStep(int* arr, int arrSize, int m) {

}
```

**Go:**

```
func findLatestStep(arr []int, m int) int {

}
```

**Kotlin:**

```
class Solution {
fun findLatestStep(arr: IntArray, m: Int): Int {

}
}
```

**Swift:**

```
class Solution {
func findLatestStep(_ arr: [Int], _ m: Int) -> Int {

}
}
```

**Rust:**

```
impl Solution {
pub fn find_latest_step(arr: Vec<i32>, m: i32) -> i32 {

}
}
```

**Ruby:**

```
# @param {Integer[]} arr
# @param {Integer} m
# @return {Integer}
def find_latest_step(arr, m)

end
```

**PHP:**

```
class Solution {
```

```
/**
* @param Integer[] $arr
* @param Integer $m
* @return Integer
*/
function findLatestStep($arr, $m) {

}
}
```

**Dart:**

```
class Solution {
int findLatestStep(List<int> arr, int m) {

}
}
```

**Scala:**

```
object Solution {
def findLatestStep(arr: Array[Int], m: Int): Int = {

}
}
```

**Elixir:**

```
defmodule Solution do
@spec find_latest_step(arr :: [integer], m :: integer) :: integer
def find_latest_step(arr, m) do

end
end
```

**Erlang:**

```
-spec find_latest_step(Arr :: [integer()], M :: integer()) -> integer().
find_latest_step(Arr, M) ->
.
```

**Racket:**

```
(define/contract (find-latest-step arr m)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```

## Solutions

### C++ Solution:

```
/*
* Problem: Find Latest Group of Size M
* Difficulty: Medium
* Tags: array, string, tree, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public:
int findLatestStep(vector<int>& arr, int m) {

}
};
```

### Java Solution:

```
/**
* Problem: Find Latest Group of Size M
* Difficulty: Medium
* Tags: array, string, tree, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public int findLatestStep(int[] arr, int m) {

}
```

```
}
```

### Python3 Solution:

```
"""
Problem: Find Latest Group of Size M
Difficulty: Medium
Tags: array, string, tree, hash, search

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

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

### Python Solution:

```
class Solution(object):
def findLatestStep(self, arr, m):
"""
:type arr: List[int]
:type m: int
:rtype: int
"""
```

### JavaScript Solution:

```
/**
* Problem: Find Latest Group of Size M
* Difficulty: Medium
* Tags: array, string, tree, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/
```

```
/**
* @param {number[]} arr
* @param {number} m
* @return {number}
*/
var findLatestStep = function(arr, m) {

};
```

### TypeScript Solution:

```
/**
* Problem: Find Latest Group of Size M
* Difficulty: Medium
* Tags: array, string, tree, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

function findLatestStep(arr: number[], m: number): number {

};
```

### C# Solution:

```
/*
* Problem: Find Latest Group of Size M
* Difficulty: Medium
* Tags: array, string, tree, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

public class Solution {
public int FindLatestStep(int[] arr, int m) {

}
```

```
}
```

### C Solution:

```
/*
* Problem: Find Latest Group of Size M
* Difficulty: Medium
* Tags: array, string, tree, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

int findLatestStep(int* arr, int arrSize, int m) {

}
```

### Go Solution:

```
// Problem: Find Latest Group of Size M
// Difficulty: Medium
// Tags: array, string, tree, hash, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func findLatestStep(arr []int, m int) int {

}
```

### Kotlin Solution:

```
class Solution {
fun findLatestStep(arr: IntArray, m: Int): Int {

}
}
```

### Swift Solution:

```
class Solution {
func findLatestStep(_ arr: [Int], _ m: Int) -> Int {

}
}
```

### Rust Solution:

```
// Problem: Find Latest Group of Size M
// Difficulty: Medium
// Tags: array, string, tree, hash, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

impl Solution {
pub fn find_latest_step(arr: Vec<i32>, m: i32) -> i32 {

}
}
```

### Ruby Solution:

```
# @param {Integer[]} arr
# @param {Integer} m
# @return {Integer}
def find_latest_step(arr, m)

end
```

### PHP Solution:

```
class Solution {

/**
* @param Integer[] $arr
* @param Integer $m
* @return Integer
*/
function findLatestStep($arr, $m) {
```

```
}
}
```

**Dart Solution:**

```
class Solution {
int findLatestStep(List<int> arr, int m) {

}
}
```

**Scala Solution:**

```
object Solution {
def findLatestStep(arr: Array[Int], m: Int): Int = {

}
}
```

**Elixir Solution:**

```
defmodule Solution do
@spec find_latest_step(arr :: [integer], m :: integer) :: integer
def find_latest_step(arr, m) do

end
end
```

**Erlang Solution:**

```
-spec find_latest_step(Arr :: [integer()], M :: integer()) -> integer().
find_latest_step(Arr, M) ->
.
```

**Racket Solution:**

```
(define/contract (find-latest-step arr m)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```