Problem 2424: Longest Uploaded Prefix

Problem Information

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

Problem Description

You are given a stream of

n

videos, each represented by a

distinct

number from

1

to

n

that you need to "upload" to a server. You need to implement a data structure that calculates the length of the

longest uploaded prefix

at various points in the upload process.

We consider

i

to be an uploaded prefix if all videos in the range

1

to

i

(

inclusive

) have been uploaded to the server. The longest uploaded prefix is the

maximum

value of

i

that satisfies this definition.

Implement the

LUPrefix

class:

LUPrefix(int n)

Initializes the object for a stream of

n

videos.

void upload(int video)

Uploads

video

to the server.

int longest()

Returns the length of the

longest uploaded prefix

defined above.

Example 1:

Input

["LUPrefix", "upload", "longest", "upload", "longest", "upload", "longest"] [[4], [3], [], [1], [], [2], []]

Output

[null, null, 0, null, 1, null, 3]

Explanation

LUPrefix server = new LUPrefix(4); // Initialize a stream of 4 videos. server.upload(3); // Upload video 3. server.longest(); // Since video 1 has not been uploaded yet, there is no prefix. // So, we return 0. server.upload(1); // Upload video 1. server.longest(); // The prefix [1] is the longest uploaded prefix, so we return 1. server.upload(2); // Upload video 2. server.longest(); // The prefix [1,2,3] is the longest uploaded prefix, so we return 3.

Constraints:

1 <= n <= 10

5

1 <= video <= n

All values of

video

are

distinct

.

At most

2 * 10

5

calls

in total

will be made to

upload

and

longest

.

At least one call will be made to

longest

.

Code Snippets

C++:

```
class LUPrefix {
public:
LUPrefix(int n) {

}

void upload(int video) {

}

int longest() {

}
};

/**
* Your LUPrefix object will be instantiated and called as such:
* LUPrefix* obj = new LUPrefix(n);
* obj->upload(video);
* int param_2 = obj->longest();
*/
```

Java:

```
class LUPrefix {

public LUPrefix(int n) {

}

public void upload(int video) {

}

public int longest() {

}
}
```

```
/**
* Your LUPrefix object will be instantiated and called as such:
* LUPrefix obj = new LUPrefix(n);
* obj.upload(video);
* int param_2 = obj.longest();
*/
```

Python3:

```
class LUPrefix:

def __init__(self, n: int):


def upload(self, video: int) -> None:


def longest(self) -> int:



# Your LUPrefix object will be instantiated and called as such:
# obj = LUPrefix(n)
# obj.upload(video)
# param_2 = obj.longest()
```

Python:

```
class LUPrefix(object):

def __init__(self, n):
"""
:type n: int
"""


def upload(self, video):
"""
:type video: int
:rtype: None
"""
```

```
def longest(self):
"""
:rtype: int
"""



# Your LUPrefix object will be instantiated and called as such:
# obj = LUPrefix(n)
# obj.upload(video)
# param_2 = obj.longest()
```

JavaScript:

```
/**
* @param {number} n
*/
var LUPrefix = function(n) {

};

/**
* @param {number} video
* @return {void}
*/
LUPrefix.prototype.upload = function(video) {

};

/**
* @return {number}
*/
LUPrefix.prototype.longest = function() {

};

/**
* Your LUPrefix object will be instantiated and called as such:
* var obj = new LUPrefix(n)
* obj.upload(video)
* var param_2 = obj.longest()
```

```
*/
```

TypeScript:

```
class LUPrefix {
constructor(n: number) {

}

upload(video: number): void {

}

longest(): number {

}
}

/**
* Your LUPrefix object will be instantiated and called as such:
* var obj = new LUPrefix(n)
* obj.upload(video)
* var param_2 = obj.longest()
*/
```

C#:

```
public class LUPrefix {

public LUPrefix(int n) {

}

public void Upload(int video) {

}

public int Longest() {

}
}
```

```
/**
* Your LUPrefix object will be instantiated and called as such:
* LUPrefix obj = new LUPrefix(n);
* obj.Upload(video);
* int param_2 = obj.Longest();
*/
```

C:

```
typedef struct {

} LUPrefix;


LUPrefix* lUPrefixCreate(int n) {

}

void lUPrefixUpload(LUPrefix* obj, int video) {

}

int lUPrefixLongest(LUPrefix* obj) {

}

void lUPrefixFree(LUPrefix* obj) {

}

/**
* Your LUPrefix struct will be instantiated and called as such:
* LUPrefix* obj = lUPrefixCreate(n);
* lUPrefixUpload(obj, video);

* int param_2 = lUPrefixLongest(obj);

* lUPrefixFree(obj);
*/
```

Go:

```
type LUPrefix struct {

}


func Constructor(n int) LUPrefix {

}


func (this *LUPrefix) Upload(video int) {

}


func (this *LUPrefix) Longest() int {

}


/**
* Your LUPrefix object will be instantiated and called as such:
* obj := Constructor(n);
* obj.Upload(video);
* param_2 := obj.Longest();
*/
```

Kotlin:

```
class LUPrefix(n: Int) {

fun upload(video: Int) {

}

fun longest(): Int {

}

}
```

```
/**
* Your LUPrefix object will be instantiated and called as such:
* var obj = LUPrefix(n)
* obj.upload(video)
* var param_2 = obj.longest()
*/
```

Swift:

```
class LUPrefix {

init(_ n: Int) {

}

func upload(_ video: Int) {

}

func longest() -> Int {

}
}

/**
* Your LUPrefix object will be instantiated and called as such:
* let obj = LUPrefix(n)
* obj.upload(video)
* let ret_2: Int = obj.longest()
*/
```

Rust:

```
struct LUPrefix {

}


/**
* `&self` means the method takes an immutable reference.
```

```
* If you need a mutable reference, change it to `&mut self` instead.
*/
impl LUPrefix {

fn new(n: i32) -> Self {

}

fn upload(&self, video: i32) {

}

fn longest(&self) -> i32 {

}
}

/**
* Your LUPrefix object will be instantiated and called as such:
* let obj = LUPrefix::new(n);
* obj.upload(video);
* let ret_2: i32 = obj.longest();
*/
```

Ruby:

```
class LUPrefix

=begin
:type n: Integer
=end
def initialize(n)

end


=begin
:type video: Integer
:rtype: Void
=end
def upload(video)
```

```
end


=begin
:rtype: Integer
=end
def longest()

end


end

# Your LUPrefix object will be instantiated and called as such:
# obj = LUPrefix.new(n)
# obj.upload(video)
# param_2 = obj.longest()
```

PHP:

```
class LUPrefix {
/**
* @param Integer $n
*/
function __construct($n) {

}

/**
* @param Integer $video
* @return NULL
*/
function upload($video) {

}

/**
* @return Integer
*/
function longest() {

}
```

```
}

/**
* Your LUPrefix object will be instantiated and called as such:
* $obj = LUPrefix($n);
* $obj->upload($video);
* $ret_2 = $obj->longest();
*/
```

Dart:

```
class LUPrefix {

LUPrefix(int n) {

}

void upload(int video) {

}

int longest() {

}
}

/**
* Your LUPrefix object will be instantiated and called as such:
* LUPrefix obj = LUPrefix(n);
* obj.upload(video);
* int param2 = obj.longest();
*/
```

Scala:

```
class LUPrefix(_n: Int) {

def upload(video: Int): Unit = {

}

def longest(): Int = {
```

```
}

}

/**
* Your LUPrefix object will be instantiated and called as such:
* val obj = new LUPrefix(n)
* obj.upload(video)
* val param_2 = obj.longest()
*/
```

Elixir:

```
defmodule LUPrefix do
@spec init_(n :: integer) :: any
def init_(n) do

end

@spec upload(video :: integer) :: any
def upload(video) do

end

@spec longest() :: integer
def longest() do

end
end

# Your functions will be called as such:
# LUPrefix.init_(n)
# LUPrefix.upload(video)
# param_2 = LUPrefix.longest()

# LUPrefix.init_ will be called before every test case, in which you can do
some necessary initializations.
```

Erlang:

```
-spec lu_prefix_init_(N :: integer()) -> any().
lu_prefix_init_(N) ->
.

-spec lu_prefix_upload(Video :: integer()) -> any().
lu_prefix_upload(Video) ->
.

-spec lu_prefix_longest() -> integer().
lu_prefix_longest() ->
.


%% Your functions will be called as such:
%% lu_prefix_init_(N),
%% lu_prefix_upload(Video),
%% Param_2 = lu_prefix_longest(),

%% lu_prefix_init_ will be called before every test case, in which you can do
some necessary initializations.
```

Racket:

```
(define lu-prefix%
(class object%
(super-new)

; n : exact-integer?
(init-field
n)

; upload : exact-integer? -> void?
(define/public (upload video)
)
; longest : -> exact-integer?
(define/public (longest)
)))

;; Your lu-prefix% object will be instantiated and called as such:
;; (define obj (new lu-prefix% [n n]))
;; (send obj upload video)
;; (define param_2 (send obj longest))
```

Solutions

C++ Solution:

```
/*
* Problem: Longest Uploaded Prefix
* Difficulty: Medium
* Tags: tree, graph, hash, search, queue, heap
*
* Approach: DFS or BFS traversal
* Time Complexity: O(n) where n is number of nodes
* Space Complexity: O(h) for recursion stack where h is height
*/

class LUPrefix {
public:
LUPrefix(int n) {

}

void upload(int video) {

}

int longest() {

}
};

/**
* Your LUPrefix object will be instantiated and called as such:
* LUPrefix* obj = new LUPrefix(n);
* obj->upload(video);
* int param_2 = obj->longest();
*/
```

Java Solution:

```
/**
* Problem: Longest Uploaded Prefix
* Difficulty: Medium
```

```
* Tags: tree, graph, hash, search, queue, heap
*
* Approach: DFS or BFS traversal
* Time Complexity: O(n) where n is number of nodes
* Space Complexity: O(h) for recursion stack where h is height
*/

class LUPrefix {

public LUPrefix(int n) {

}

public void upload(int video) {

}

public int longest() {

}
}

/**
* Your LUPrefix object will be instantiated and called as such:
* LUPrefix obj = new LUPrefix(n);
* obj.upload(video);
* int param_2 = obj.longest();
*/
```

Python3 Solution:

```
"""
Problem: Longest Uploaded Prefix
Difficulty: Medium
Tags: tree, graph, hash, search, queue, heap

Approach: DFS or BFS traversal
Time Complexity: O(n) where n is number of nodes
Space Complexity: O(h) for recursion stack where h is height
"""
```

```
class LUPrefix:

def __init__(self, n: int):


def upload(self, video: int) -> None:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class LUPrefix(object):

def __init__(self, n):
"""
:type n: int
"""


def upload(self, video):
"""
:type video: int
:rtype: None
"""


def longest(self):
"""
:rtype: int
"""



# Your LUPrefix object will be instantiated and called as such:
# obj = LUPrefix(n)
# obj.upload(video)
# param_2 = obj.longest()
```

JavaScript Solution:

```javascript
/**
* Problem: Longest Uploaded Prefix
* Difficulty: Medium
* Tags: tree, graph, hash, search, queue, heap
*
* Approach: DFS or BFS traversal
* Time Complexity: O(n) where n is number of nodes
* Space Complexity: O(h) for recursion stack where h is height
*/

/**
* @param {number} n
*/
var LUPrefix = function(n) {

};

/**
* @param {number} video
* @return {void}
*/
LUPrefix.prototype.upload = function(video) {

};

/**
* @return {number}
*/
LUPrefix.prototype.longest = function() {

};

/**
* Your LUPrefix object will be instantiated and called as such:
* var obj = new LUPrefix(n)
* obj.upload(video)
* var param_2 = obj.longest()
*/
```

TypeScript Solution:

```
/**
* Problem: Longest Uploaded Prefix
* Difficulty: Medium
* Tags: tree, graph, hash, search, queue, heap
*
* Approach: DFS or BFS traversal
* Time Complexity: O(n) where n is number of nodes
* Space Complexity: O(h) for recursion stack where h is height
*/

class LUPrefix {
constructor(n: number) {

}

upload(video: number): void {

}

longest(): number {

}
}

/**
* Your LUPrefix object will be instantiated and called as such:
* var obj = new LUPrefix(n)
* obj.upload(video)
* var param_2 = obj.longest()
*/
```

C# Solution:

```
/*
* Problem: Longest Uploaded Prefix
* Difficulty: Medium
* Tags: tree, graph, hash, search, queue, heap
*
* Approach: DFS or BFS traversal
* Time Complexity: O(n) where n is number of nodes
* Space Complexity: O(h) for recursion stack where h is height
*/
```

```
public class LUPrefix {

public LUPrefix(int n) {

}

public void Upload(int video) {

}

public int Longest() {

}
}

/**
* Your LUPrefix object will be instantiated and called as such:
* LUPrefix obj = new LUPrefix(n);
* obj.Upload(video);
* int param_2 = obj.Longest();
*/
```

C Solution:

```
/*
* Problem: Longest Uploaded Prefix
* Difficulty: Medium
* Tags: tree, graph, hash, search, queue, heap
*
* Approach: DFS or BFS traversal
* Time Complexity: O(n) where n is number of nodes
* Space Complexity: O(h) for recursion stack where h is height
*/




typedef struct {

} LUPrefix;
```

```
LUPrefix* lUPrefixCreate(int n) {

}

void lUPrefixUpload(LUPrefix* obj, int video) {

}

int lUPrefixLongest(LUPrefix* obj) {

}

void lUPrefixFree(LUPrefix* obj) {

}

/**
* Your LUPrefix struct will be instantiated and called as such:
* LUPrefix* obj = lUPrefixCreate(n);
* lUPrefixUpload(obj, video);

* int param_2 = lUPrefixLongest(obj);

* lUPrefixFree(obj);
*/
```

Go Solution:

```
// Problem: Longest Uploaded Prefix
// Difficulty: Medium
// Tags: tree, graph, hash, search, queue, heap
//
// Approach: DFS or BFS traversal
// Time Complexity: O(n) where n is number of nodes
// Space Complexity: O(h) for recursion stack where h is height

type LUPrefix struct {

}
```

```
func Constructor(n int) LUPrefix {

}


func (this *LUPrefix) Upload(video int) {

}


func (this *LUPrefix) Longest() int {

}


/**
* Your LUPrefix object will be instantiated and called as such:
* obj := Constructor(n);
* obj.Upload(video);
* param_2 := obj.Longest();
*/
```

Kotlin Solution:

```
class LUPrefix(n: Int) {

fun upload(video: Int) {

}

fun longest(): Int {

}

}

/**
* Your LUPrefix object will be instantiated and called as such:
* var obj = LUPrefix(n)
```

```
* obj.upload(video)
* var param_2 = obj.longest()
*/
```

Swift Solution:

```
class LUPrefix {

init(_ n: Int) {

}

func upload(_ video: Int) {

}

func longest() -> Int {

}
}

/**
* Your LUPrefix object will be instantiated and called as such:
* let obj = LUPrefix(n)
* obj.upload(video)
* let ret_2: Int = obj.longest()
*/
```

Rust Solution:

```
// Problem: Longest Uploaded Prefix
// Difficulty: Medium
// Tags: tree, graph, hash, search, queue, heap
//
// Approach: DFS or BFS traversal
// Time Complexity: O(n) where n is number of nodes
// Space Complexity: O(h) for recursion stack where h is height

struct LUPrefix {
```

```
}


/**
* `&self` means the method takes an immutable reference.
* If you need a mutable reference, change it to `&mut self` instead.
*/
impl LUPrefix {

fn new(n: i32) -> Self {

}

fn upload(&self, video: i32) {

}

fn longest(&self) -> i32 {

}
}

/**
* Your LUPrefix object will be instantiated and called as such:
* let obj = LUPrefix::new(n);
* obj.upload(video);
* let ret_2: i32 = obj.longest();
*/
```

Ruby Solution:

```
class LUPrefix

=begin
:type n: Integer
=end
def initialize(n)

end
```

```
=begin
:type video: Integer
:rtype: Void
=end
def upload(video)

end


=begin
:rtype: Integer
=end
def longest()

end



end

# Your LUPrefix object will be instantiated and called as such:
# obj = LUPrefix.new(n)
# obj.upload(video)
# param_2 = obj.longest()
```

PHP Solution:

```
class LUPrefix {
/**
* @param Integer $n
*/
function __construct($n) {

}

/**
* @param Integer $video
* @return NULL
*/
function upload($video) {

}
```

```
/**
* @return Integer
*/
function longest() {

}
}

/**
* Your LUPrefix object will be instantiated and called as such:
* $obj = LUPrefix($n);
* $obj->upload($video);
* $ret_2 = $obj->longest();
*/
```

Dart Solution:

```
class LUPrefix {

LUPrefix(int n) {

}

void upload(int video) {

}

int longest() {

}
}

/**
* Your LUPrefix object will be instantiated and called as such:
* LUPrefix obj = LUPrefix(n);
* obj.upload(video);
* int param2 = obj.longest();
*/
```

Scala Solution:

```
class LUPrefix(_n: Int) {

def upload(video: Int): Unit = {

}

def longest(): Int = {

}

}

/**
* Your LUPrefix object will be instantiated and called as such:
* val obj = new LUPrefix(n)
* obj.upload(video)
* val param_2 = obj.longest()
*/
```

Elixir Solution:

```
defmodule LUPrefix do
@spec init_(n :: integer) :: any
def init_(n) do

end

@spec upload(video :: integer) :: any
def upload(video) do

end

@spec longest() :: integer
def longest() do

end
end

# Your functions will be called as such:
# LUPrefix.init_(n)
# LUPrefix.upload(video)
# param_2 = LUPrefix.longest()
```

```
# LUPrefix.init_ will be called before every test case, in which you can do
some necessary initializations.
```

Erlang Solution:

```
-spec lu_prefix_init_(N :: integer()) -> any().
lu_prefix_init_(N) ->
.

-spec lu_prefix_upload(Video :: integer()) -> any().
lu_prefix_upload(Video) ->
.

-spec lu_prefix_longest() -> integer().
lu_prefix_longest() ->
.


%% Your functions will be called as such:
%% lu_prefix_init_(N),
%% lu_prefix_upload(Video),
%% Param_2 = lu_prefix_longest(),

%% lu_prefix_init_ will be called before every test case, in which you can do
some necessary initializations.
```

Racket Solution:

```
(define lu-prefix%
(class object%
(super-new)

; n : exact-integer?
(init-field
n)

; upload : exact-integer? -> void?
(define/public (upload video)
)
; longest : -> exact-integer?
```

```
(define/public (longest)
)))

;; Your lu-prefix% object will be instantiated and called as such:
;; (define obj (new lu-prefix% [n n]))
;; (send obj upload video)
;; (define param_2 (send obj longest))
```