Problem 2424: Longest Uploaded Prefix

Problem Information

Difficulty: Medium
Acceptance Rate: 54.42%
Paid Only: No
Tags: Hash Table, Binary Search, Union Find, Design, Binary Indexed Tree, Segment Tree, Heap (Priority Queue), Ordered Set

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 <= 105` * `1 <= video <= n` * All values of `video` are **distinct**. * At most `2 * 105` 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()
```