Number of Students Unable to Eat Lunch

The school cafeteria offers circular and square sandwiches at lunch break, referred to by numbers 0 and 1 respectively. All students stand in a queue. Each student either prefers square or circular sandwiches.

The number of sandwiches in the cafeteria is equal to the number of students. The sandwiches are placed in a stack. At each step:

If the student at the front of the queue prefers the sandwich on the top of the stack, they will take it and leave the queue.
Otherwise, they will leave it and go to the queue's end
This continues until none of the queue students want to take the top sandwich and are thus unable to eat.

You are given two integer arrays students and sandwiches where sandwiches[i] is the type of the ith sandwich in the stack (i = 0 is the top of the stack) and students[j] is the preference of the jth student in the initial queue (j = 0 is the front of the queue). Return the number of students that are unable to eat.

Input: students = [1,1,0,0], sandwiches = [0,1,0,1]
Output: 0
Explanation:
- Front student leaves the top sandwich and returns to the end of the line making students = [1,0,0,1].
- Front student leaves the top sandwich and returns to the end of the line making students = [0,0,1,1].
- Front student takes the top sandwich and leaves the line making students = [0,1,1] and sandwiches = [1,0,1].
- Front student leaves the top sandwich and returns to the end of the line making students = [1,1,0].
- Front student takes the top sandwich and leaves the line making students = [1,0] and sandwiches = [0,1].
- Front student leaves the top sandwich and returns to the end of the line making students = [0,1].
- Front student takes the top sandwich and leaves the line making students = [1] and sandwiches = [1].
- Front student takes the top sandwich and leaves the line making students = [] and sandwiches = [].
Hence all students are able to eat.

Input: students = [1,1,1,0,0,1], sandwiches = [1,0,0,0,1,1]
Output: 3

Input: students [1,1,0,0], sandwiches = [0,1,0,1]

0 - Circuler
1 - Square

students[1,1,0,0] 0 -> 0 -> 1 -> 1(front) sandwiches[0,1,0,1] 1 -> 0 -> 1 -> 0 (front)

students[1,0,0,1] -> 1 -> 0 -> 0 -> 1 (front) sandwiches[0<-1<-0<-1]

students[0,0,1,1] -> -> 1 -> 1 -> 0 -> 0 sandwiches[0,1,0,1]

students[0,1,1] -> 1 -> 1 -> 0 sandwiches[1,0,1]

students[1,1,0] -> 0 -> 1 -> 1 sandwiches[1,0,1]

students[1,0] -> 0 -> 1 sandwiches[0,1]

students[0,1] -> 1 -> 0 sandwiches[0,1]

students[1] -> 1 sandwiches[1]

students[] -> sandwiches[]

output: 0

****

students [1,1,1,0,0,1], sandwiches[1,0,0,0,1,1]

students [1,1,0,0,1], sandwiches[0,0,0,1,1]

students [1,0,0,1,1], sandwiches[0,0,0,1,1]

students [0,0,1,1,1], sandwiches[0,0,0,1,1]

students [0,1,1,1], sandwiches[0,0,1,1]
counter = 0

students [1,1,1], sandwiches[0,1,1]

students [1,1,1], sandwiches[0,1,1] : couter = 1
students [1,1,1], sandwiches[0,1,1] : couter = 2
students [1,1,1], sandwiches[0,1,1] : counter= 3

output : 3

TimeComplexity O(2n)
4 <-> 4

n

[1]+students[3] <-> sandwishes[3]

[2]+ 2+1 <-> 3

[3] + 1 +2 <-> 3

[4] + 2 <-> 2

[5] + 1+1 <-> 2

[5+1] + 1 <-> 1

Design Hit Counter

Design a hit counter which counts the number of hits
received in the past 5 minutes (i.e., the past 300 seconds).

Your system should accept a timestamp parameter
(in seconds granularity), and you may assume that calls are
being made to the system in chronological order (i.e., timestamp is monotonically increasing). Several hits may arrive roughly at the same time.

Implement the HitCounter class:

HitCounter() Initializes the object of the hit counter system.
void hit(int timestamp) Records a hit that happened at timestamp
(in seconds). Several hits may happen at the same timestamp.
int getHits(int timestamp) Returns the number of hits in the past
5 minutes from timestamp (i.e., the past 300 seconds).

Input
["HitCounter", "hit", "hit", "hit", "getHits", "hit", "getHits", "getHits"]
[[], [1], [2], [3], [4], [300], [300], [301]]
Output
[null, null, null, null, 3, null, 4, 3]

Explanation
HitCounter hitCounter = new HitCounter();
hitCounter.hit(1); // hit at timestamp 1.
hitCounter.hit(2); // hit at timestamp 2.
hitCounter.hit(3); // hit at timestamp 3.
hitCounter.getHits(4); // get hits at timestamp 4, return 3.
hitCounter.hit(300); // hit at timestamp 300.
hitCounter.getHits(300); // get hits at timestamp 300, return 4.
hitCounter.getHits(301); // get hits at timestamp 301, return 3.

1sec -> 2 sec -> 3sec -> 4sec

gitHits(300)// output: 4 5*60 = 300sec : 4
300-4 = 296sec
300-3 = 297Sec
300-2 = 298Sec
300-1 = 299Sec

4 -> 3 -> 2 -> 1

gitHits(301)// output: 3
301-4 = 297sec
301-3 = 298Sec
301-2 = 299Sec
301-1 = 300 (X)
4 -> 3 -> 2 -> 1 => 301 - 1 = 300 >= 300

gitHits(303)// output: 1
303-4 = 299sec
303-3 = 300Sec
303-2 = 301Sec
303-1 = 302Sec
4 => 303-4 299 >=300

1sec -> 2 sec -> 2 sec -> 2 sec -> 3sec -> 4sec
gitHits(300)// output: 6 5*60 = 300sec : 4
300-4 = 296sec
300-3 = 297Sec
300-2 = 298Sec (3)
300-1 = 299Sec

1sec -> 2 sec -> 2 sec -> 2 sec -> 3sec -> 4sec
gitHits(304)// output: 6 5*60 = 300sec : 4