

Today's Content

1. m^n non-repeating char for every prefix substring
2. m^n negative elements in all subarrays for size = k

6.11

queue < type que;

que.push() Insert n at rear/back end of queue

que.pop() delete ele at front end

que.front() Return ele at front end

que.back() Return ele at back end.

que.size() Return no. of ele in queue.

18 Given input of stream of characters

For every new input char print 1st non-repeating char for entire data
Note: If no non repeating character print #

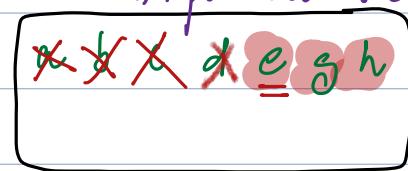
Stream: a b c c a e b e a g

Output: a a a a b b e # # g

Stream: a b c ~~d~~ d ~~e~~ e g h ~~f~~ ~~g~~ ~~h~~

Output: a a a a a a d l d d d e #

1st Note: # All possible ans char



hashmap

a: 2 # if freq > 1 = repetition.

b: 2 # if freq > 1 = repetition.

c: 2 # if freq > 1 = repetition.

d: 2 # if freq > 1 = repetition.

e: 2 # if freq > 1 = repetition.

f: 2 # if freq > 1 = repetition.

g: 2 # if freq > 1 = repetition.

Operations: Queue

Insert back

Delete front

Ans front

Pseudo Code:

void iRepeating(string s) { Tc: O(n) Sc: O(n)

queue q;

unordered_map<char, int> um;

for (int i=0; i < s.size(); i++) {

New char is st[i];

um[st[i]]++;

if (um[st[i]] == 1) { # non repeating

q.push(st[i]);

1 pop = 1 iter

N push oper

N pop oper

while (q.size() > 0 && um[q.front()] > 1) {

q.pop();

if (q.size() == 0) {

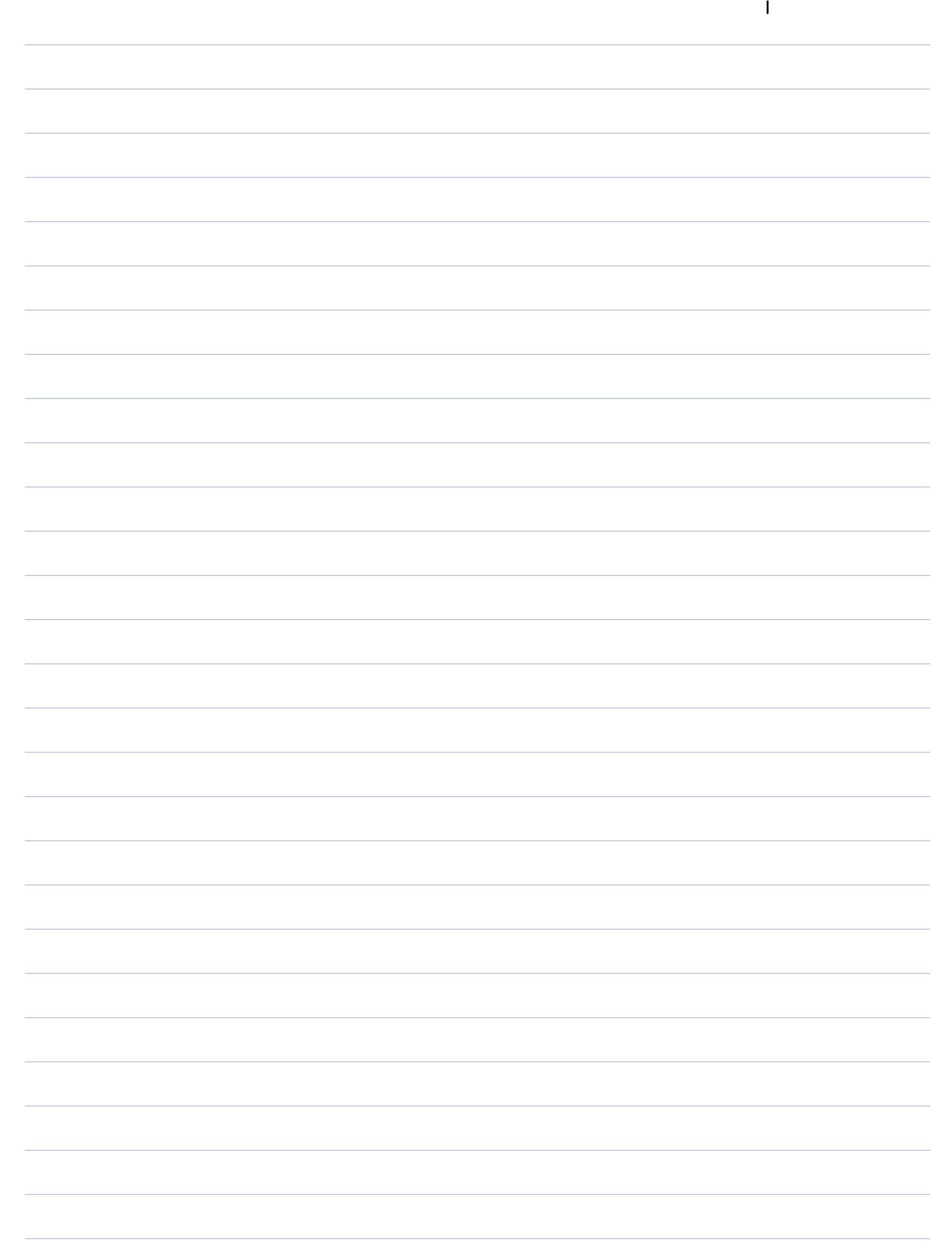
print("#");

else {

print(q.front());

}

}



28 first-re element in all subarrays of size = k.

Note: For a subarray of no-re element print 0

$$\text{arr}[] = \{ 1 \ 2 \ 3 \ 4 \ 5 \ 2 \} \quad k=4$$

#Subarrays Output:

[0 3] -1

[1 4] -1

[2 5] -2

[3 6] -4

Idea: for every subarray of len k:

Iterate & calculate 1st-re element.

Tc: $\Theta(N-k+1) * \Theta(k)$ SC: $\Theta(1)$

if $k \approx N/2$

$\Theta(N-N/2+1) * \Theta(N/2) \approx \Theta(N^2)$

Ideas : Sliding Window ?

Dry Run:

$\text{ar}[] = \{ 1 \ -1 \ -2 \ 3 \ -4 \ 5 \ 2 \ -6 \ 2 \ 8 \} \ k=5$

All possible ans

$[0 \ 4]$ $-1 \ -2 \ -4$

Def cur[]

Add cur[]

Output

-1

$[1 \ 5]$ $\text{cur}(0) = 1 \times$

$\text{cur}(5) = 5 \times$

$-1 \ -2 \ -4$

-1

$[2 \ 6]$

$\text{cur}(1) = -1$

$\text{cur}(6) = 2 \times$

$\cancel{-1} \ -2 \ -4$

$\cancel{-1}$

$[3 \ 7]$

$\text{cur}(2) = -2$

$\text{cur}(7) = -6 \checkmark$

$\cancel{-1} \ -2 \ -4 \ -6$

$\cancel{-4}$

$[4 \ 8]$

$\text{cur}(3) = 3 \times$

$\text{cur}(8) = 2 \times$

$-4 \ -6$

-4

$[5 \ 9]$

$\text{cur}(4) = -4$

$\text{cur}(9) = 8 \times$

$\cancel{-1} \ -2 \ -4 \ -6$

-6

Container : queue

Add back

Delete front

Has front

void firstNegative(vector<int> arr, int k) { TC: $O(N)$ SC: $O(k)$

queue<int> q;

for (int i=0; i < k; i++) {

if (arr[i] < 0) {

q.push(arr[i]);

}

print(q.front());

int s=1, e=k;

while (e < arr.size()) {

remove arr[s-1] add arr[e];

if (arr[s-1] == q.front()) {

q.pop();

if (arr[e] < 0) {

q.push(arr[e]);

if (q.size() == 0) {

print(0);

else {

print(q.front());

}

3

3