28/11/2023

QUEUE CLASS - 3

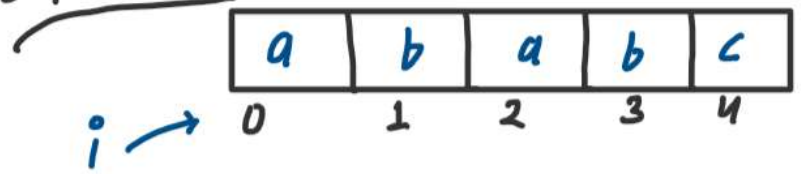
1. First Unique Character in a String (Leetcode-387)

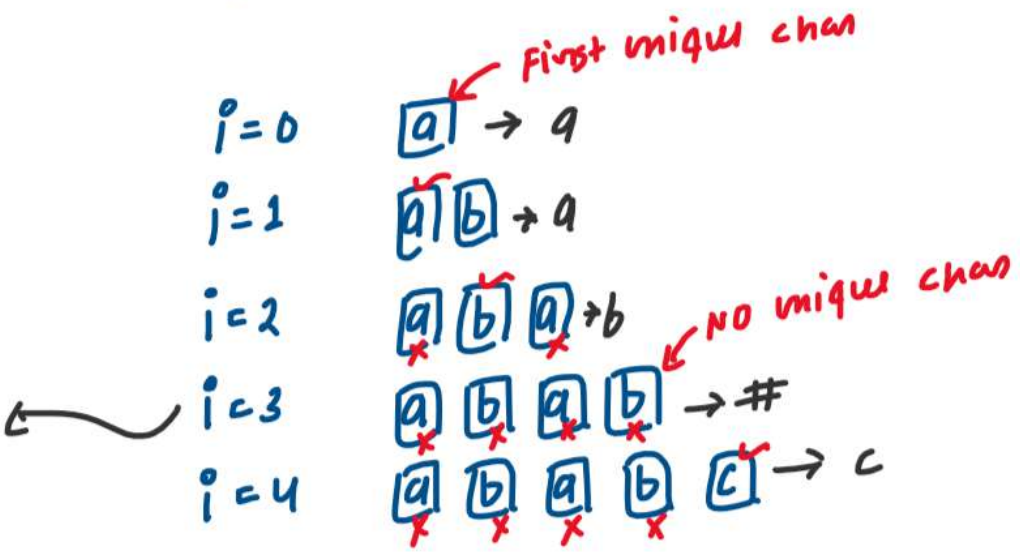
Ex1

String s = "ababc"

Output = a a b # c

Explanation





Ans

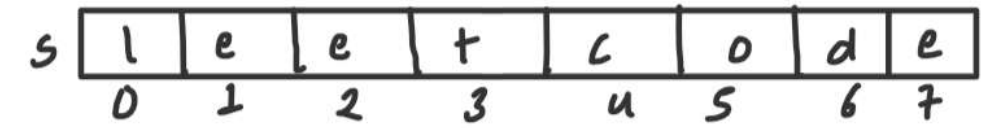
a a b # c

Ex2

String s = "leetcode"

Output = 1

Explanation



i=0 l → 1

i=1 l e → 1

i=2 l e e → 1

i=3 l e e t → 1

i=4 l e e t c → 1

i=5 l e e t c o → 1

i=6 l e e t c o d → 1

i=7 l e e t c o d e → 1

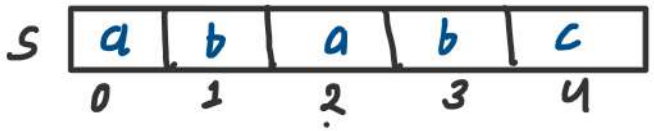
First unique character is l in whole string

Ans 1

DRY RUN with building Logic

L → R

EX1



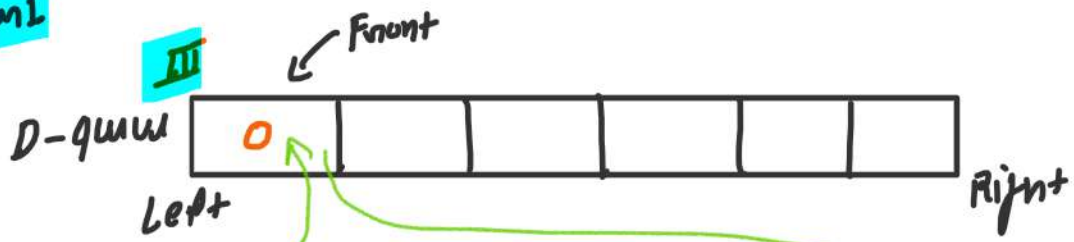
I ch = a

II FREQUENCY TABLE

Char	count
a	1

Situation1

III



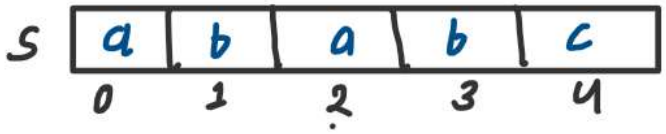
IV Ans => unique character hai ya nahi check karenge

(I) pick q.front() => 0 index

(II) Go to the freq. Table to check the Ith Index char is unique or not

print = a

If num 2

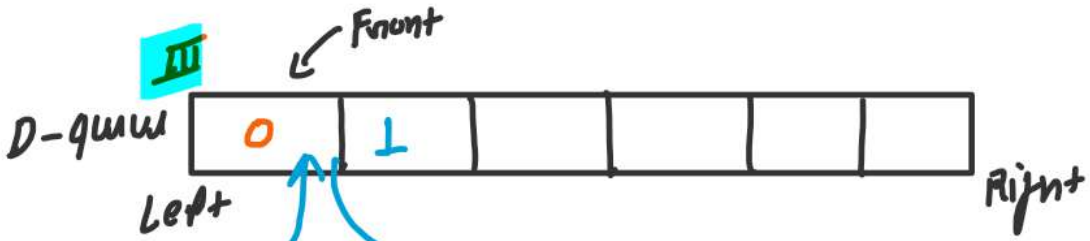


I ch = a b

II FREQUENCY TABLE

Char	count
a	1
b	1

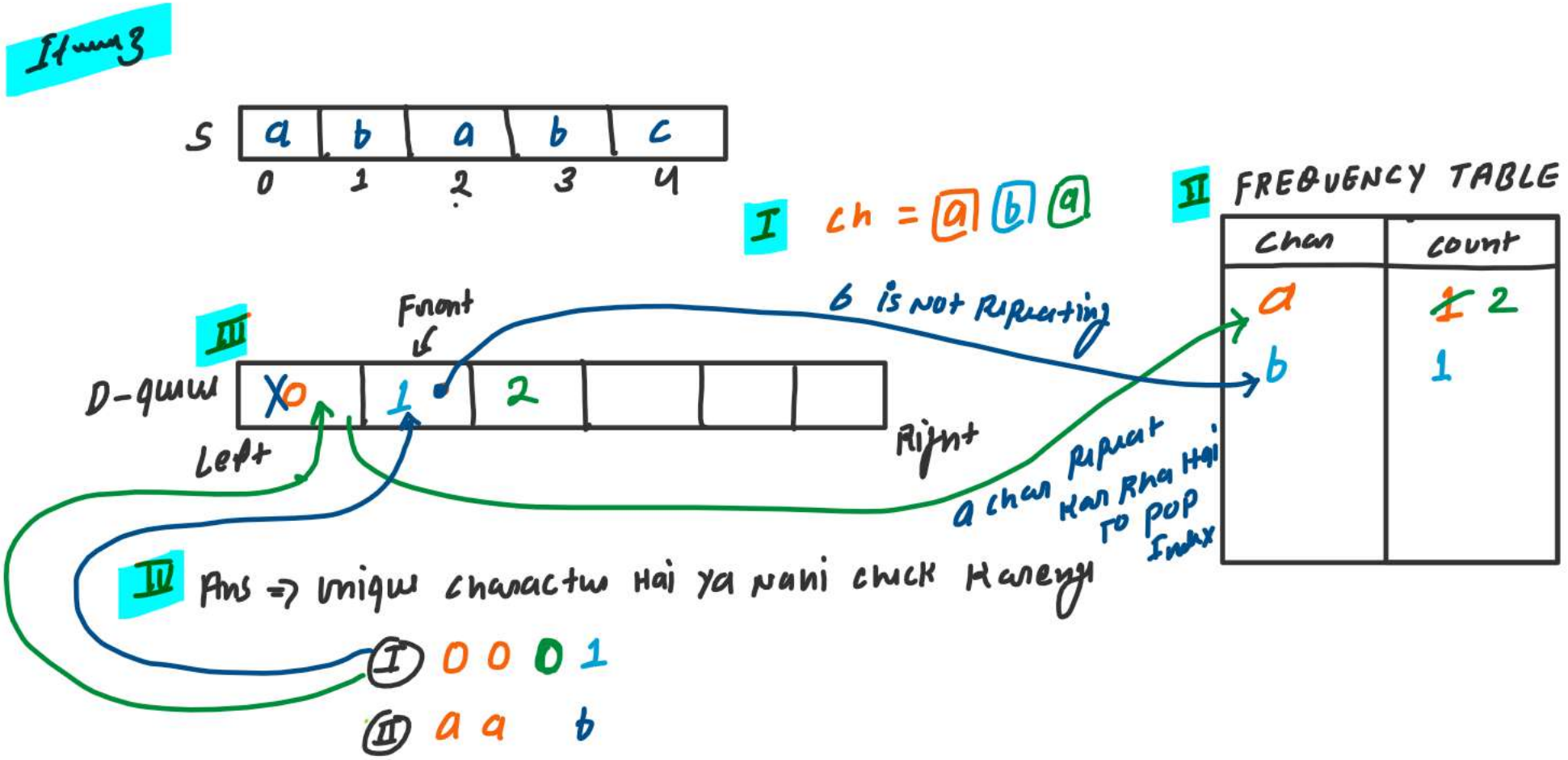
III

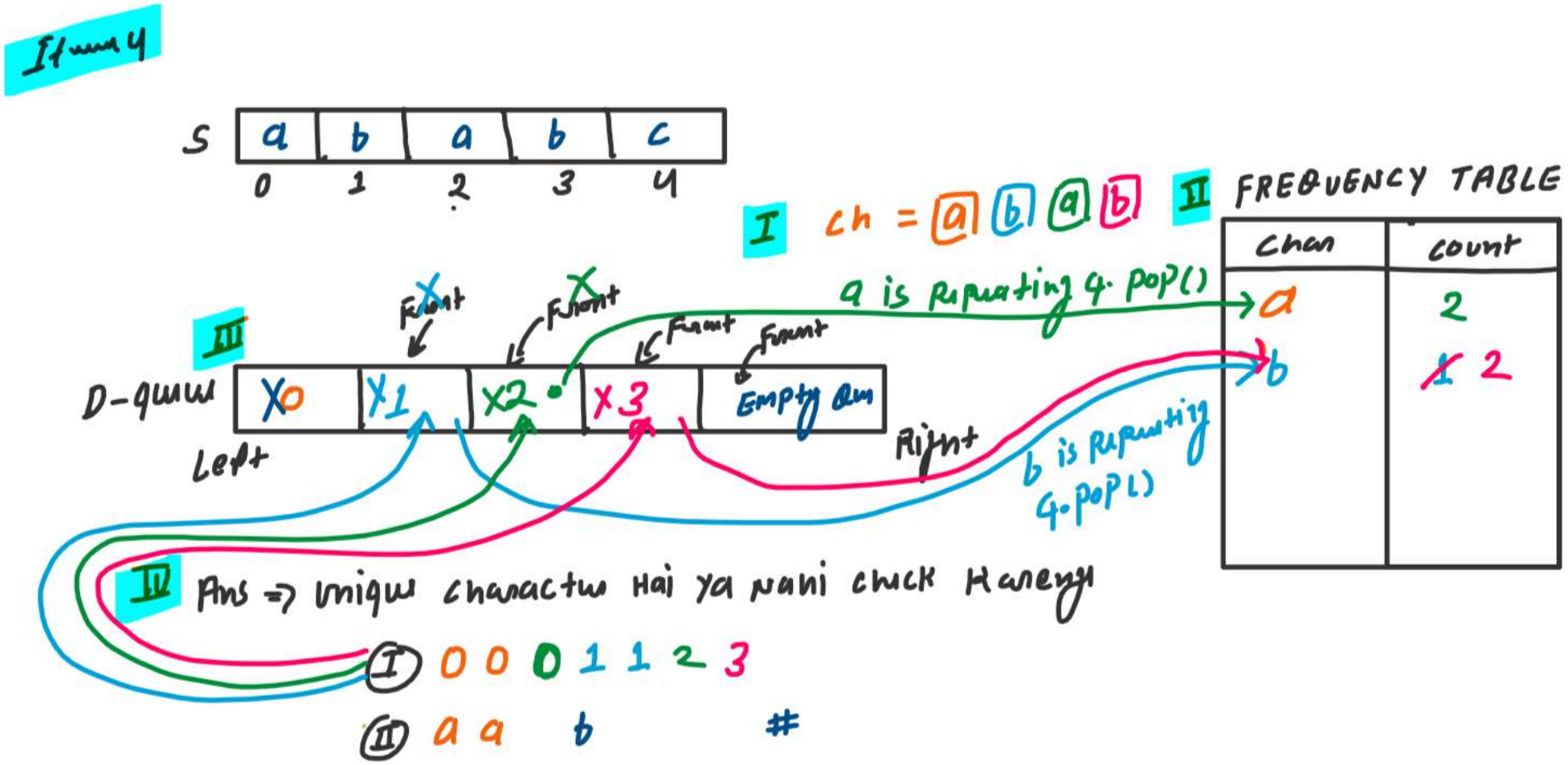


IV Ans => unique character hai ya nahi check karenge

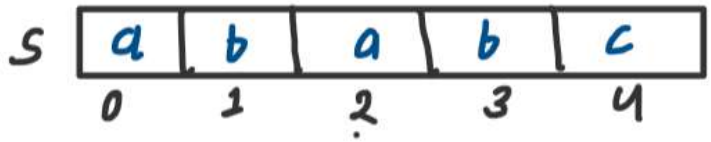
I 0 0

II a a


Ilwng3
S
a b a b c
0 1 2 3 4
I ch = a b a
II FREQUENCY TABLE
Char
count
a
2
b
1
b is not Repeating
III
Front
D-queue
X 0
1
2
Left
Right
a char Repeat Kar Rha Hai To POP Index
IV Ans => unique character hai ya nahi check Karenge
I 0 0 0 1
II a a b


If num 4
S
a b a b c
0 1 2 3 4
I ch = a b a b
II FREQUENCY TABLE
Char
count
a
2
b
2
III
D-queue
X0 X1 X2 X3 Empty Qu
Front
Front
Front
Front
Left
Right
a is Repeating q.pop()
b is Repeating q.pop()
IV Ans => unique character hai ya nahi check karenge
I 0 0 0 1 1 2 3
II a a b #

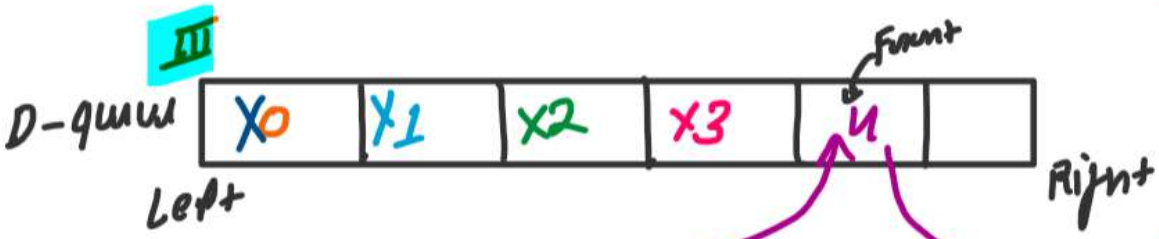
Ques 5



I ch = a b a b c

II FREQUENCY TABLE

Char	count
a	2
b	2
c	1



IV Ans => unique character hai ya nahi check karenge

(I) 0 0 0 1 1 2 3 4

(II) a a b # c

Final output **a a b # c**

```cpp
// 1. First Non Repeating/Unique Character in a String (Leetcode-387)
void firstUniqueChar(string s){
    deque<int> q;
    int freq[26] = {0};

    for (int i = 0; i < s.length(); i++)
    {
        // I -> picking current character from string
        char ch = s[i];

        // II -> storing the frequency of current character to array
        freq[ch - 'a']++;

        // III -> inserting the index of current character to queue
        q.push_back(i);


        // IV -> Ans find kro: Unique character hai ya nhi
        while(!q.empty()){
            // I -> picking the front value from queue
            char frontChar = s[q.front()];

            // II -> going to freq table to check the frontChar is unique or not
            if(freq[frontChar - 'a'] > 1){
                // Character is repeating-> no ans
                q.pop_front();
            }
            else{
                // Character is not repeating-> ans
                cout<< frontChar << " ";
                break;
            }
        }

        // Agar koi bhi ans nhi mila hai
        // iska mtlb queue empty ho chuka hai
        if(q.empty()){
            cout<< '#' << " ";
        }
    }
}
```

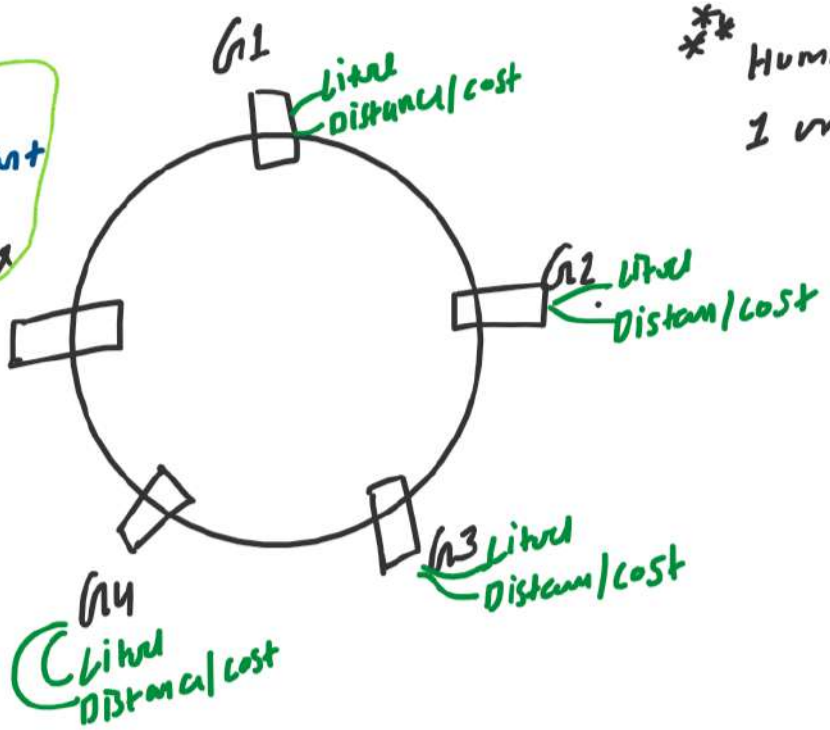
```cpp
// LEETCODE PROBLEM SOLUTION CODE
class Solution {
public:
    int firstUniqChar(string s){
        deque<int> q;
        int freq[26] = {0};
        int ans = -1;

        for (int i = 0; i < s.length(); i++)
        {
            // I -> picking current character from string
            char ch = s[i];

            // II -> storing the frequency of current character to array
            freq[ch - 'a']++;

            // III -> inserting the index of current character to queue
            q.push_back(i);


            // IV -> Ans find kro: Unique character hai ya nhi
            while(!q.empty()){
                // I -> picking the front value from queue
                char frontChar = s[q.front()];

                // II -> going to freq table to check the frontChar is unique or not
                if(freq[frontChar - 'a'] > 1){
                    // Character is repeating-> no ans
                    q.pop_front();
                }
                else{
                    // Character is not repeating-> ans
                    ans = q.front();
                    break;
                }
            }

            // Agar koi bhi ans nhi mila hai
            // iska mtlb queue empty ho chuka hai
            if(q.empty()){
                ans = -1;
            }
        }
        return ans;
    }
};
```

Time Complexity: *O(N^2), Where N is length of string*
Space Complexity: *O(N), Where queue stores N number of characters*

2. Gas Station (Leetcode-134) ***

Find Kya Karna Hai?

Man lo car G1 se chalna start karti Hai or wo G1 par AAkar RUK jati Hai TO Ans G1 ka Index otherwise -1 Hoga



** Humari car 1 litre me 1 unit distance cover karegi only.

Ex 1

Input

gas

1	2	3	4	5
0	1	2	3	4

distance

3	4	5	1	2
0	1	2	3	4

Index = 4

gas = 5 + 3 = 8

dist = 2

bal = 8 - 2

= 6

Index = 0

gas = 1 + 6 = 7

dist = 3

bal = 7 - 3 = 4

Index = 1

gas = 2 + 4 = 6

dist = 4

bal = 6 - 4 = 2

Index = 2

gas = 3 + 2 = 5

dist = 5

bal = 5 - 5 = 0

Ans = 3

BRUTE FORCE APPROACH

Index = 0 ✗ gas = 1, dist = 3 — can can not start from Index 0

Index = 1 ✗ gas = 2, dist = 4

Index = 2 ✗ gas = 3, dist = 5

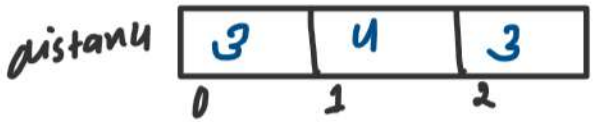
Index = 3 gas = 4, dist = 1, bal = 4 - 1 = 3

can can start from Index (3) and Ans can be Equ to (3)

TC = $O(N^2)$

Ex2

Input



Output = -1

Can index 2 se start hue but Index 1 pun RUK jati Hai Iska mtlb Gas Nani Rani Hogi TO Ans will be -1

BRUTE FORCE

Index = 0
X
Gas = 2
dist = 3

Index = 1
X
Gas = 3
dist = 4

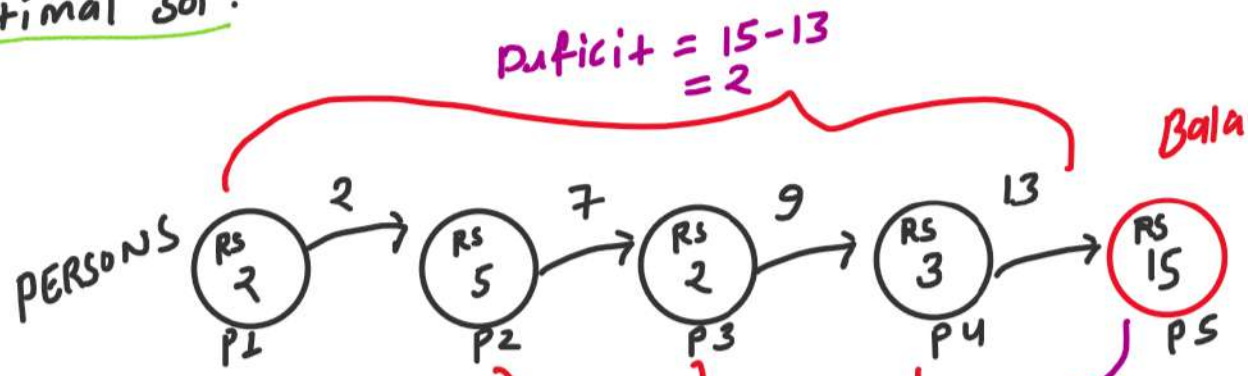
Index = 2
Gas = 4
dist = 3
bal = 4-3 = 1

Index = 0
Gas = 2+1
dist = 3
bal = 0

Index = 1

Loop Stop at index 1

Optimal Soln:



Manlo P1 ko RS na liye jaye
Tab bhi PS ki Help Nahi ho sakti Hai
OR Le liye jaye
Tab bhi Nahi ho sakti Hai

Manlo PS ke pass RS 15 Already hai
To wo khud se Aphi Help kar sakta Hai

- PS ko RS 15 ki Need Hai
- P1, P2, P3, and P4 RS. Collect karke PS ki Help karna Chahte Hai

if (Help)
↳ PS Happy

Balance − Deficit ≥ 0
↳ PS Happy

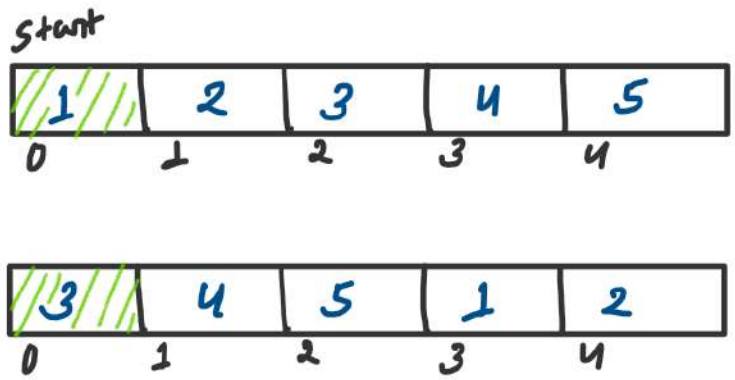
if (!Help)
↳ PS Not Happy

Bal − Def < 0
↳ PS Not Happy

T.C. = O(N)

Ex 1

Input




```
if(bal < 0)
    ↳ deficit += bal;   ← Absolute
      start = index+1;
      bal = 0;
```

Situation 1

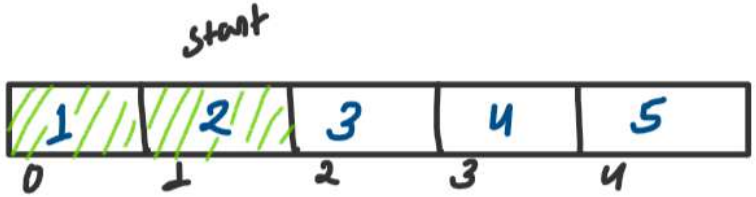
deficit = 0
balance = 0
start = 0

Index = 0

Gas = 1
dist = 3
bal = 1-3 = -2 ↳

deficit = 0+2
= 2
start = 1
bal = 0

Input




```
if(bal < 0)                    ← Absolutu
  ↳ dificit += bal;
    start = indux+1;
    bal = 0;
```

Itmatim 2

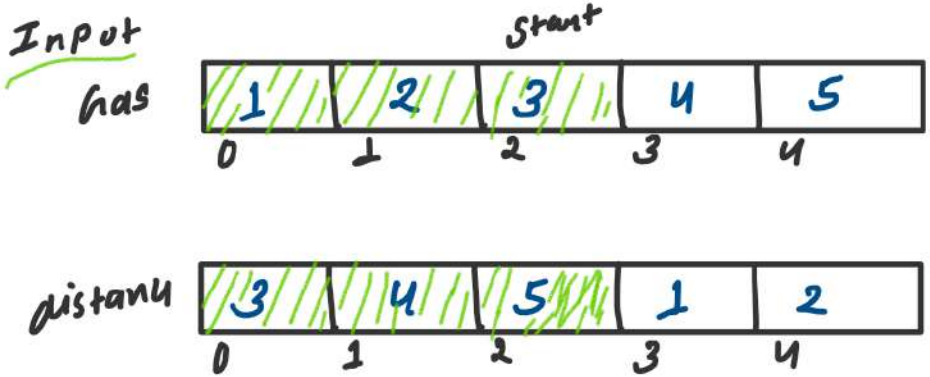
dificit = 2
balonca = 0
Start = 1

Indux = 1

gas = 2
dist = 4
bal = 2-4 = -2 ↳

```
dificit = 2+2
        = 4
start = 2
bal = 0
```

Input




```
if(bal < 0)                    ← Absolute
  ↳ deficit += bal;
    start = index+1;
    bal = 0;
```

Situation 3

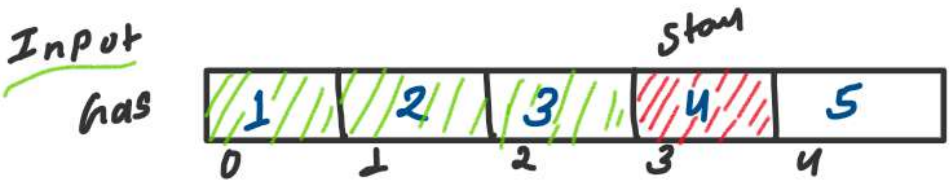
deficit = 4
balance = 0
Start = 2

Index = 2

Gas = 3
dist = 5
bal = 3 - 5 = -2

↳ deficit = 4 + 2 = 6
Start = 3
bal = 0

Input




```
if(bal >=0)
  ↳ bal += bal
    index++
```

Intuition

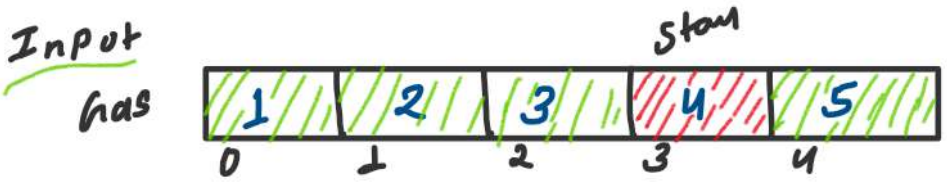
deficit = 6
balance = 0
Start = 3

Index = 3

Gas = 4
dist = 1
bal = 4-1 = 3

```
( bal += bal
  bal = 0+3
      = 3 )
```

Input




```
if(bal >=0)
  ↳ bal += bal
    index++
```

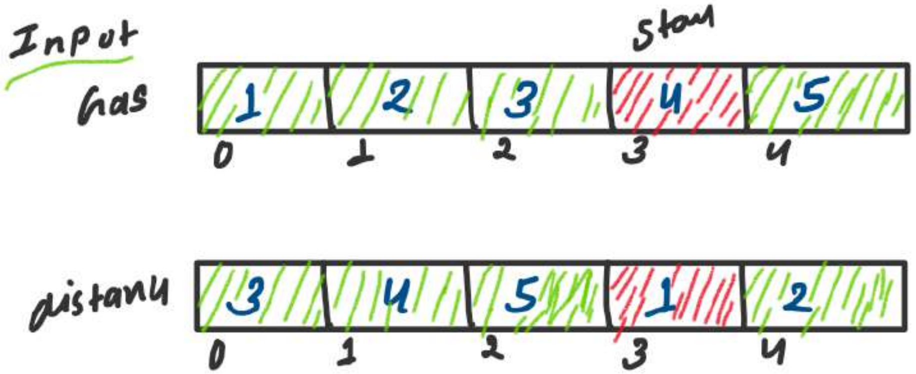
Situations

deficit = 6
balance = 3
Start = 3

Index = 4

gas = 5
dist = 2
bal = 5-2 = 3

```
bal += bal
bal = 3+3
    = 6
```



Iteration 6

deficit = 6
balance = 6
Start = 3

Index = 6
↳ STOP the loop

```
if ( bal - deficit >= 0 )
    ↳ return start;
Else if (bal - deficit < 0)
    ↳ return -1;
```

↳ CAR jaha se chalna Start kari thi AND usi point par Aakar Ruk gayi hai
↳ Iska mtlb Circle complete ho chuka hai

↳ Circle complete nahi Huaa hai

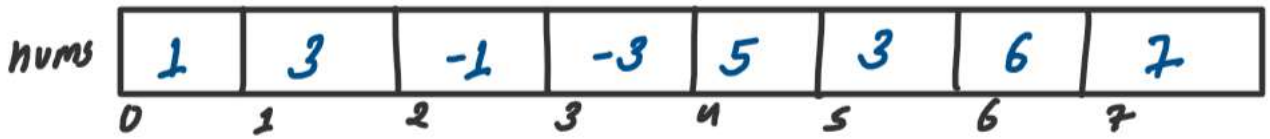
$T.C = O(N)$
↳ $N \Rightarrow$ length of array

```cpp
// 2. Gas Station (Leetcode-134)
// Optimal Solution Without Using Queue

class Solution {
public:
    int canCompleteCircuit(vector<int>& gas, vector<int>& cost) {
        int deficit = 0;
        int balance = 0;
        // Car start index se drive karna start karegi
        int start = 0;

        // Har ek index ko check karenge ki wo ans/starting index hai ya nahi
        for(int index = 0; index < gas.size(); index++){
            balance += gas[index] - cost[index];

            if(balance < 0){
                deficit += abs(balance);
                start = index + 1;
                balance = 0;
            }
        }

        // Car circle complete kar chuki hai
        if(balance - deficit >= 0){
            return start;
        }
        else{
        // Car circle complete nahi kar payi hai
            return -1;
        }
    }
};
```

Time Complexity: *O(N), Where N is size of array*
Space Complexity: *O(1), no extra space used*

3. Sliding Window Maximum (Leetcode-239)

Ex

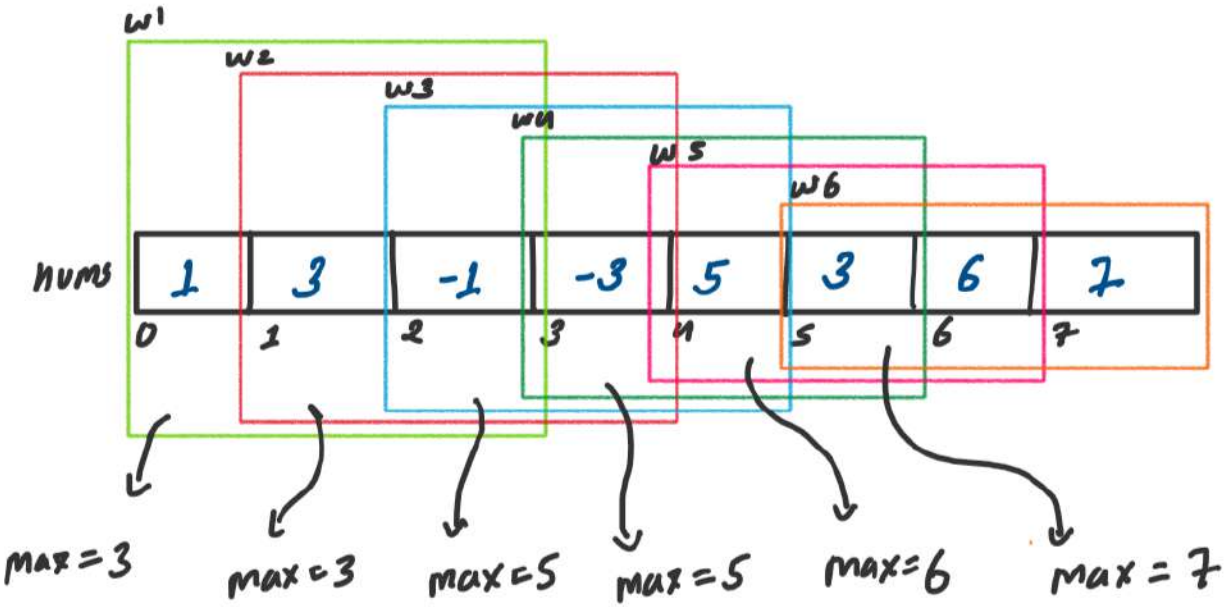


K=3

Output

3	3	5	5	6	7

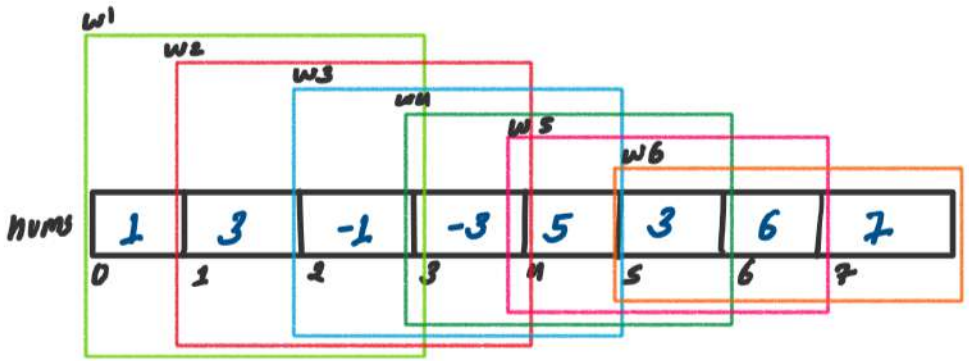
Explanation



K=3

N=size=8

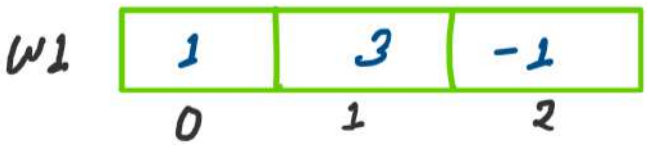
DRY RUN

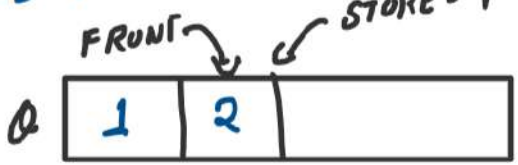


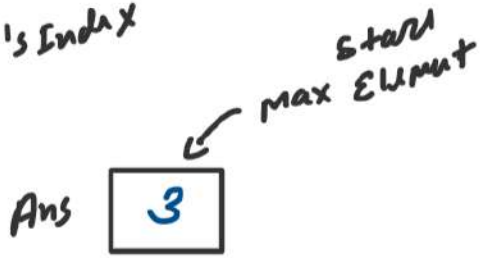
K=3

N=size=8

STEP1 Process first window of size 'K'

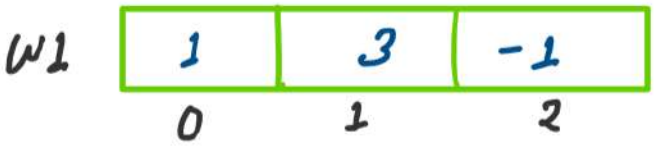


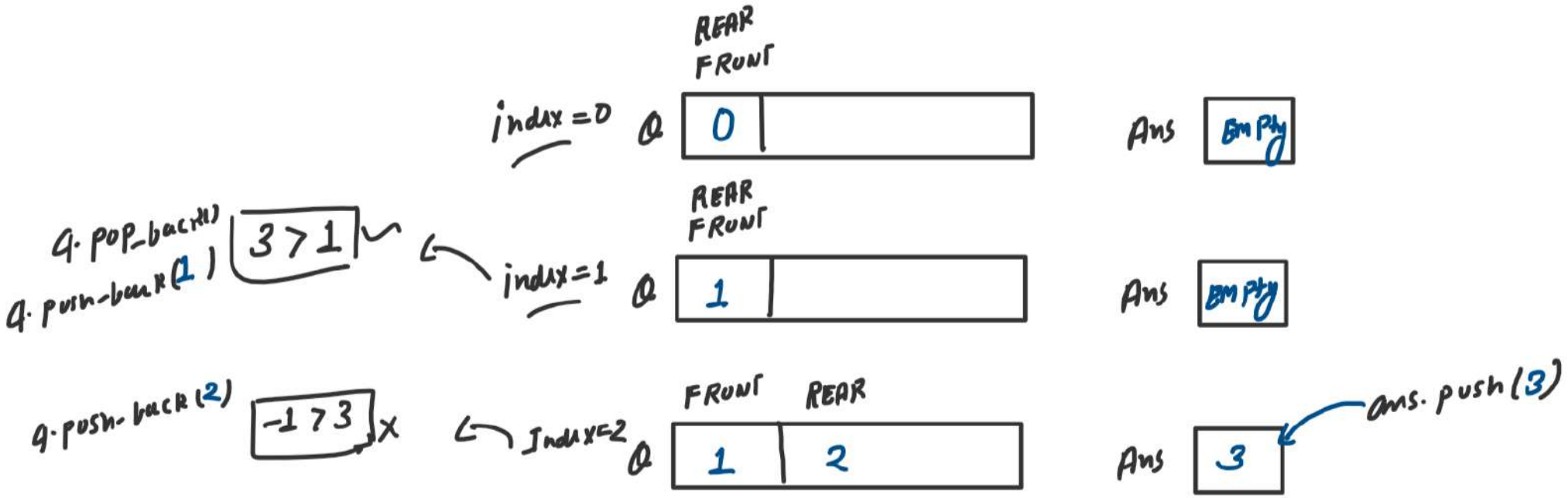




YEH kaise fill ho gya?

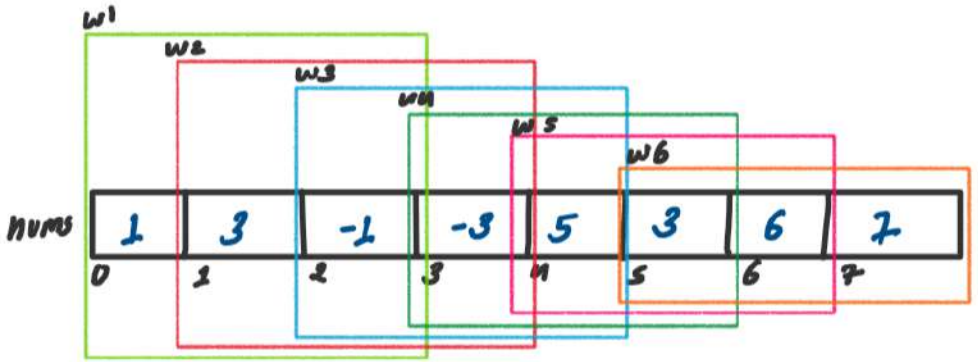
STEP1 Process first window of size 'K'





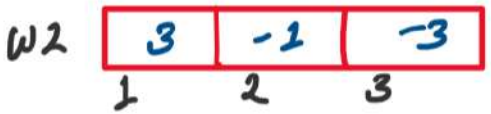
STEP2 Process Remaining windows

↳ REMOVE Invalid



K = 3

N = size = 8



REMOVE (in queue)

① Out of Range [0, 1, 2]

② Chotta Element Removed

index - q.front() >= K

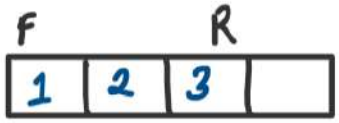
↳ q.pop_front();

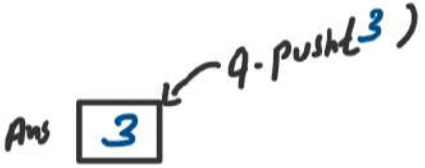
q.pop_back()

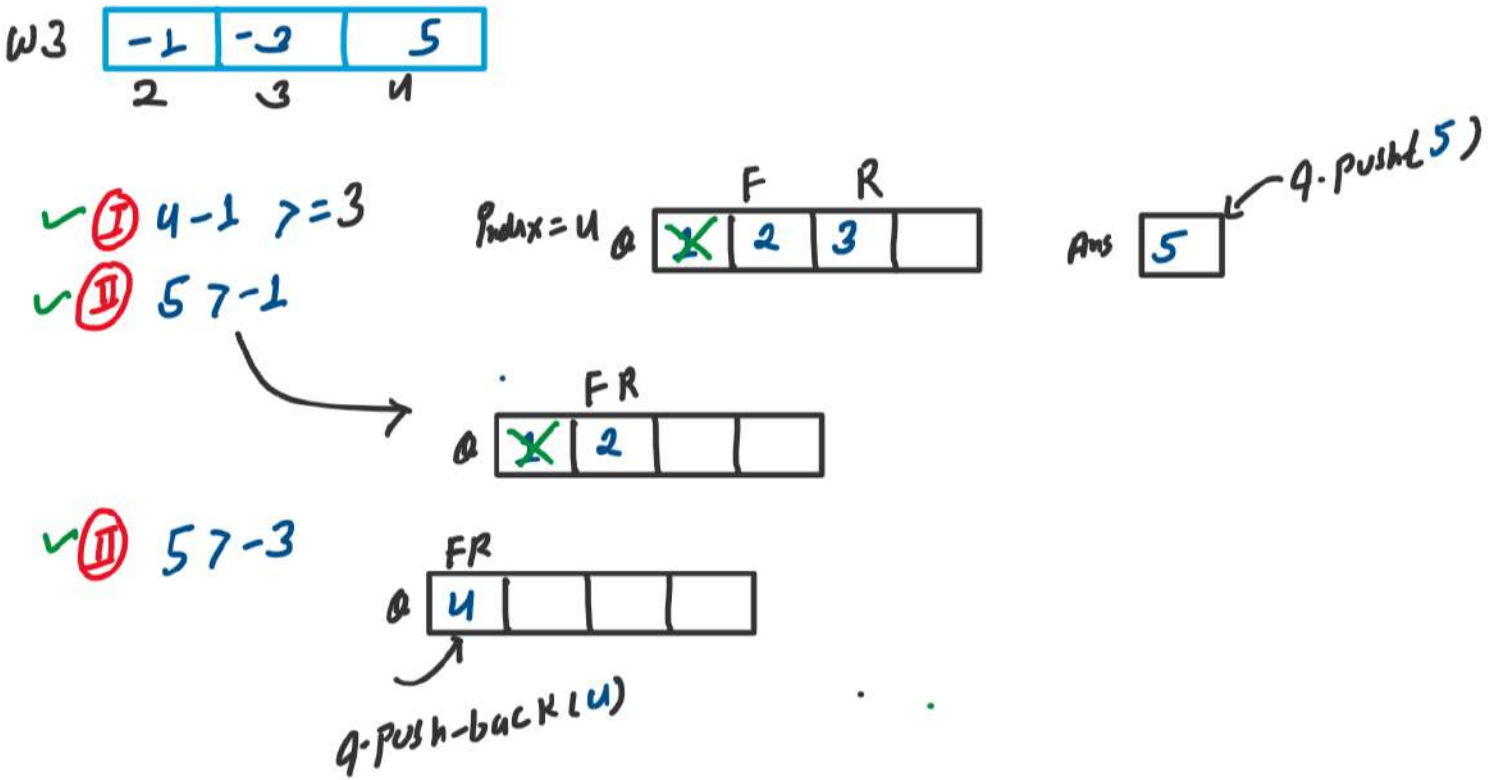
✗ ① 3 - 1 >= 3

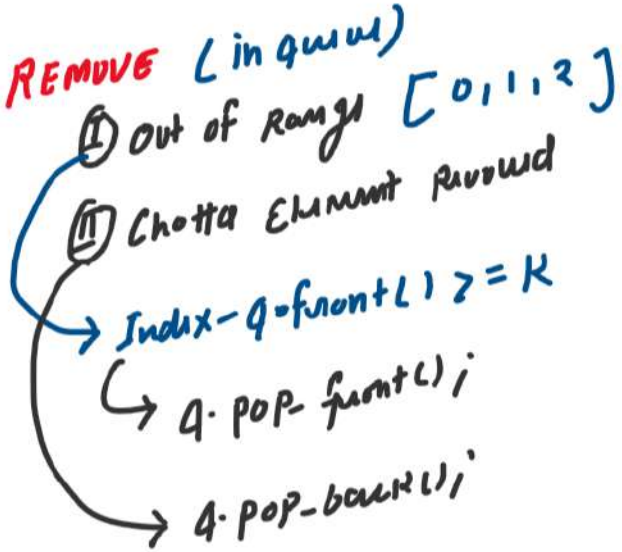
✗ ② -3 > -1

index = 3



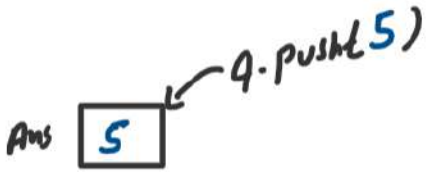



W3
-1
-2
5
2
3
4
I 4-1 >=3
II 5 7-1
F R
Index=4
Q
2
3
Ans
5
q.push(5)
FR
2
II 5 7-3
4
q.push_back(4)


REMOVE (in queue)
I Out of Range [0,1,2]
II Chotta Element Removed
Index-q.front() >= K
q.pop_front();
q.pop_back();

X (I) 5-4 >=3

X (II) 3 >= 5



REMOVE (in queue)

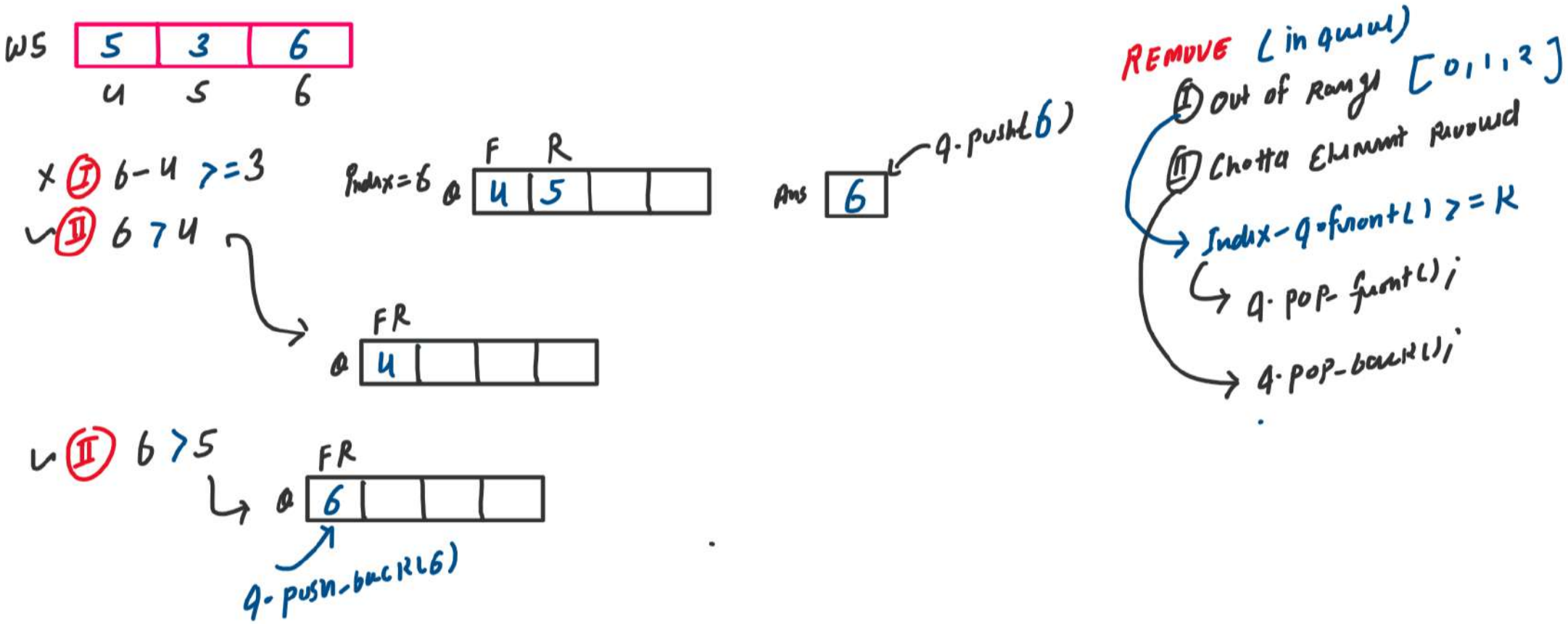
(I) Out of Range [0,1,2]

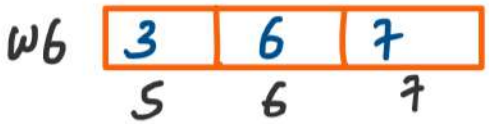
(II) Chotta Element Ruvuvid

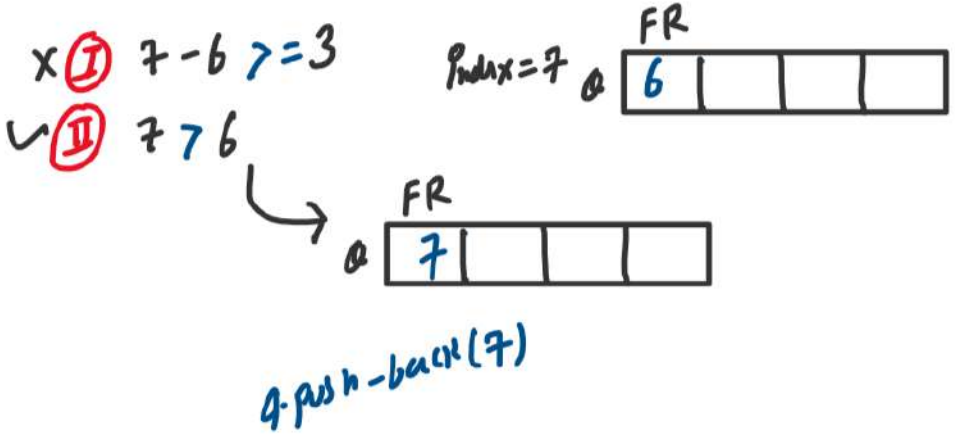
Index - q.front() >= K

q.pop_front();

q.pop_back();

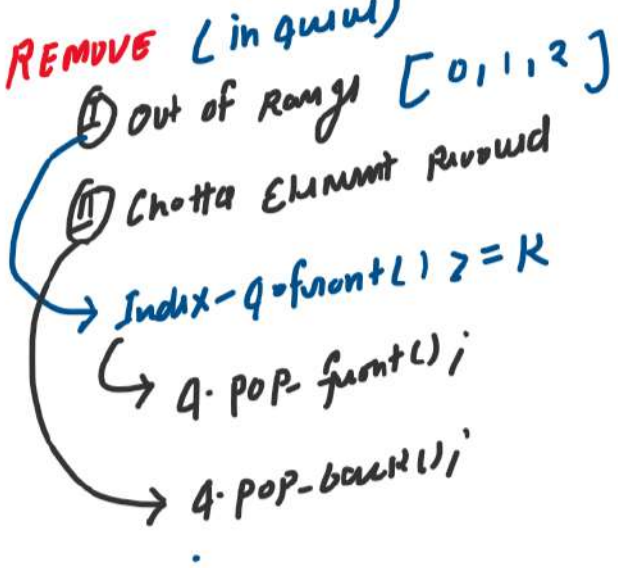

WS
5
3
6
4
5
6
x (I) 6-4 >=3
(II) 6 > 4
Index=6
Q
F R
4
5
Ans
6
q.push(6)
FR
Q
4
(II) 6 > 5
FR
Q
6
q.push_back(6)
REMOVE (in queue)
(I) Out of Range [0,1,2]
(II) Chotta Element Ruvuvid
Index-q.front() >= K
q.pop_front();
q.pop_back();


W6
3
6
7
5
6
7


X (I) 7-6 >=3
(II) 7 7 6
FR
7
q.push_back(7)

Index=7
FR
6


Ans
7
q.push(7)


REMOVE (in queue)
(I) Out of Range [0,1,2]
(II) Chotta Element Removed
Index-q.front() >= K
q.pop_front();
q.pop_back();

```cpp
// 3. Sliding Window Maximum (Leetcode-239)

class Solution {
public:
    vector<int> maxSlidingWindow(vector<int>& nums, int k) {
        deque<int> q; // store the max element's index
        vector<int> ans; // store the max element

        // Step 1: process the first window for "k time"
        for(int index = 0; index < k; index++){
            int element = nums[index];

            // Agar queue me element chotta hai
            while(!q.empty() && element > nums[q.back()]){
                q.pop_back();
            }

            // Yanha tabhi pahuch skta hu
            // Ya to queue me element chotta nhi hai
            // Ya queue empty ho chuka hai
            q.push_back(index);
        }

        // Step 2: process remaning windows
    }
};
```

```cpp
    // Step 2: process remaning windows
    for(int index = k; index < nums.size(); index++){
        // Purani window ka ans store kardo
        ans.push_back(nums[q.front()]);

        // Remove
        // I -> remove the out of range index from queue
        if(!q.empty() && index - q.front() >= k){
            q.pop_front();
        }

        // II -> remove chotta index from queue
        // Agar queue me element chotta hai
        while(!q.empty() && nums[index] > nums[q.back()]){
            q.pop_back();
        }

        // Addition
        // Yanha tabhi pahuch skta hu
        // Ya to queue me element chotta nhi hai
        // Ya queue empty ho chuka hai
        q.push_back(index);
    }

    // Last window ka ans store karlo
    ans.push_back(nums[q.front()]);

    return ans;
```

Time Complexity: *O(N),*
where N is size of array

Space Complexity: *O(K),*
where K is the size of the window