Lec-42

26/07/25 12:43 PM

[Problem Solve] — Sliding window

① Array | String

① How to identify

Input: Array | String, K

Output: sub array/substring — maximum / minimum

Sliding window: fixed size (number of) | variable size (proper)

Subarray / Substring size

---

mxk

[2, 3, 1, 4, 4, 1, 8, 7, 3, 2] K=3

0 1 2 3 4 5 6 7 8 9

① 1st window — 0 to k-1

sum = 7-2
= 5+4-3
= 6+1-4
= 6+5-1
=10
= 10+7-4
= 13+3-1
= 15+2-5
= 12

1. grow
2. shrink
3. ans calculate

k → size

ans = 15

i-k

i=3 → 0
i=4 → 1
i=5 → 2
i=6 → 3

Given an array of integers nums and an integer k, return the number of contiguous subarrays where the product of all the elements in the subarray is strictly less than k.

1+2+3+3+1 = 10

si

2 | 1 | 2 | 4 | 3

0 1 2 3 4

K=10

int p=1

while (ei < n) {
// grow
p=p*arr[i]
// shrink
while (p>=k) {
p=p/arr[si]
si++
}
// ans calculate
ans = ans + (ei - si + 1)
ei++
}

1+2+3+3+1+1

2 3 1 2 4 3

2 3 | 3 1 | 1 2 | 2 4

2 3 2 3 1 3 1 2 1 2 4

si=12 ei=4

(12) 2n-1

si si

2 | 3 | 1 | 2 | 4 | 3

0 1 2 3 4 5

K=10

R-1

R-L

R=L+1

2 | 3 | 1 | 2 | 4 | 3

P=12/2 = 6

P=8

P=24

P=12

P=3

```
public static int Product_Less_Than_K(int[] arr, int k) {
    int ei = 0, si = 0, ans = 0, p = 1;
    while (ei < arr.length) {
        // grow
        p = p * arr[ei];
        // shrink
        while (p >= k) {
            p = p / arr[si];
            si++;
        }
        // ans calculate
        ans = ans +(ei - si + 1);
        ei++;
    }
    return ans;
}
```

P=1

si<=ei

1>=0

0>=0

si si

1 | 2 | 3

ei

K=0

P=1/1=1

P=1/2=0

P=0/3=0

Kartik Bhaiya has a string consisting of only 'a' and 'b' as the characters. Kartik Bhaiya describes perfectness of a string as the maximum length substring of equal characters. Kartik Bhaiya is given a number **k** which denotes the maximum number of characters he can change. Find the maximum perfectness he can generate by changing no more than **k** characters.

a → mxx

b → mxx

(6)

b b b b b b

a b a b b a

k=2

a a a a

---

ans = 1 2 3 4 5 6 7

mxx = 1 2 3 2

K = 2

ch = 'b'

si

a a b a a a b b a a b a a

while (ei < s.length)
// grow
if (s.charAt(i) == ch)
mxx++
// shrink
while (mxx > k && si < ei)
if (s.charAt(si) == ch)
mxx--
si++
// ans calculate
ans = max(ans, ei-si+1)
ei++

Example 1: 3

```
Input: nums = [1,3,-1,-3,5,3,6,7], k = 3
Output: [3,3,5,5,6,7]
Explanation:
Window position                Max
---------------               -----
[1  3  -1] -3  5  3  6  7       3
 1 [3  -1  -3] 5  3  6  7       3
 1  3 [-1  -3  5] 3  6  7       5
 1  3  -1 [-3  5  3] 6  7       5
 1  3  -1  -3 [5  3  6] 7       6
 1  3  -1  -3  5 [3  6  7]      7
```

mxk

fun...

(3, 3, 3, 3, 6, 7)

i-k

K=4

1, 3, 1, 4, -3, -5, -7, -8, 3, 6, 7, 2

0 1 2 3 4 5 6 7 8 9 10 11

1 3 1 4 → 4

3 1 4 -3 →

1 4 -3 -5 → 4

4 -3 -5 -7 → 4

-3 -5 -7 -8 → -3

add()

q.add()

q.remove()

4-4=0

5-4=1

6-4=1

7-4=3

8-4=4

```
Deque<Integer> q = new LinkedList<>();
    int[] ans = new int[arr.length - k + 1];
    for (int i = 0; i < k; i++) {
        while (!q.isEmpty() && arr[q.getLast()] < arr[i]) {
            q.removeLast();
        }
        q.add(i);
    }
```

3 < 7

4 4 > 0

1, 3, 1, 4, -3, -5, -7, -8, 3, 6, 7, 2