ADITYA VERMA

SLIDING WINDOW

Problem : | 2 | 3 | 5 | 2 | 9 | 7 | 1 | arr

Max^m subarray of size 3

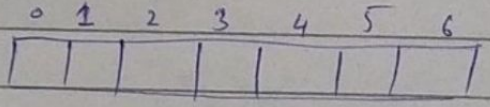
Naive

```
for (int i=0 - - - n)
    for (int j=i ; j< i+3 ; j++)
        Sum = ✓
```

Brute Force — return max^m of all sum

Improvement ??

1. Repetitive work



0 + 1 + 2
1 + 2 + 3
↳ ye work dobara yahan kar rahe hain

Ish definitely yahan optimization hoga.

~~0~~ + 1 + 2 + 3

next sum
(use of previous step)

Basically we don't need 2 loops

Identification

Pata kaise chalega ki sliding window ka concept lagega?

[arr / string] + [subarray / substring] + [largest, max, / min] + [K = window size]

↳ if ye sab given ho, toh ek baar zaroor sochna ki iss ques me sliding window protocol lag sakta hai ki nhi

2 Types
- Fixed sized window → ye hamne dekha pehle
- Variable sized window

window size [not given but asked]

asked → find out largest / smallest window
↳ based on a cond^n

Eg. Hame sum de rakha hoga, hame window sizes me se largest batana hoga jiska sum for eg 5 given ho.

Problems

Fixed - Type 1

1) Max Min sub array of size k.
2) 1st -ve in every window size of k.
3) Count occurrence of anagram
4) Max of all subarray of size k
5) Max of min for every window size

Variable - Type 2

1) Largest / smallest subarray of sum K
2) Largest substring of K distinct characters
3) Length of largest substring of no repeating characters
*4) Pick Toy
5) Minimum window substring | ***

↳ agar ye ques kaliya toh iss concept ka koi problem nhi chuleya - Std. problem.

Maximum Sum Subarray of Size K

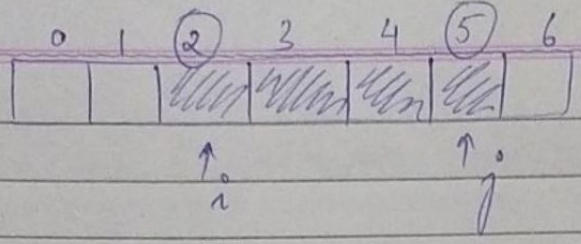
IP: size : 7
arr : 2 5 1 8 2 9 1
window size : K : 3

OP: Integer
(Max^m of all subarray sum of size 3)

Identify : 1) arr given
2) K given
3) max^m sum is o/p

Explaination:

	start	end
2 pointer le liye	i	j



If window size = 4

$j - i = 5 - 2 = 3$

$\boxed{j - i + 1} \Rightarrow 4$

If window size : k

$\boxed{k : j - i + 1}$

Code me kaise karenge

① $j - i + 1 < k$, toh j ko aage badhaao (j++)

② $j - i + 1 == k \Rightarrow$ I don't want to lose this obv.
(ye maintain karna hai window size)

calculation stage

Maintain :- i++
j++

while (j < size) {

}

Code

```
int sum = 0, maxi = Integer.MIN_VALUE;
while (j < size) {

    sum = sum + arr[j];
    if (j-i+1 < k) {
        j++;
    }
    else if (j-i+1 == k) {
        maxi = ~~max~~ Math.max(maxi, sum);

        sum = sum - arr[i];
        i++;
        j++;
    }
}
return maxi;
```

Code : Github

First Negative Number in every Window of Size K

IP: arr[]: 12 -1 -7 8 -15 30 16 28
size: 8
k: 3

O/P: -1, -1, -7, -15, -15, 0

0 ↑ DNE print zero

size - k + 1 → no of ele in o/p

Code

```
List<Integer> list = new ArrayList<Integer>();
while( j< size) {
    if (arr[j] < 0) {
        list. add (arr[j]);
    }
    if( j-i+1 < k ){
        j++;
    }
    else if ( j-i+1 == k ) {
        // calculation
        if( list. size() == 0) {
            cyso(0); // ans list.add(0);
        }
        else {
            ans add ( list.front() );
            if(arr[i] == l.front) { l.pop(front); }
        }
        // slide the window
        i++;
        j++;
    }
}
return ;
```

→ ye kam check krta challenge, kyunki agar -ve nhi hua toh kantane kalshi kaam nhi aane wala.

store -ve nos. in a list for further use

→ jab tak window size tak na pahunch jaye

yahan ans milega jab window size fit hogi

// list empty → edge case → jab list me koi -ve no. nhi hoga

// remove ve nos from list kyunki window aage badh chuki hai

12 -1 -7 8 ...

↳ iss moment pe

list [-1] [-7]

front

Count Occurrences of Anagrams

I/P :- for xx orf xa ofr #3
for : ptr

O/P : 3

Eg: aabaaba #4
aaba

Brute force : n! bekaar TC

Identification : → arr / str
→ subarr / substr
→ k : window

Socho sliding window kaise lagega

* window size : ptr.length()

For eg 1 : K = 3 ("for" length : 3)

Code :

Eg. a a b a a b a → 8
a a b a → ptr

k = ptr.length() → 4 → k = window size

```
while (j < size) {

    // calculation

    if ( < k) {
        j++ ;
    }
                    ← window size
    if ( == k) {

        ans ←
        slide the window

    }
}
```

Regular pattern (of solving these type of questions)

count

a → 3	
b → 1	

* decrease the count as you traverse in window

* Then check if a, b count = 0
  ⇒ agar hai, iska matlab hamne pura exhaust kar liya hai a, b ko

* Better Idea

a → 3	
b → 1	

↳ Count of (no. of distinct letters) = 2

Toh jaise hi a → 0 hoga, count = 2 / 1 / 0

If count = 0, that means we've used all the letters

Toh isse ye map baar baar traverse krna nhi padega.

kya kaam karna hai?

① plz : aaba → map banao

a	3
b	1

② count var : to keep track of count of distinct elements.

This is helpful cause then we would not want to traverse the map again & again

kyunki a → 0, b → 0 } ye check karne ke liye first para map me traverse karna hoga

count : map.size()

③ while (j < size) {

i = 0, j = 0 } start

a	3
b	1

if present in map : map(arr[j]) --

if (map(s[j]) == 0){

count --

3

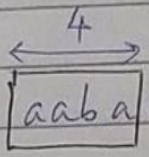
④ if (—— < k)

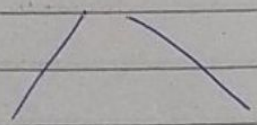
j++ ;

⑤ if (____ == k) {

1) ans ← calculation

2) slide





{ a → 0
b → 0
count → 0 }

idhar ans milega

→ count of anagrams

int ans = 0;

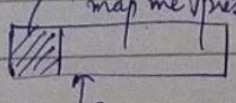
① if (count == 0) {

ans ++;

}

ans nikaalna from calculation

② check kiya ye map me present hai



hai toh count increment karo

map [s [i]] ++ ;

if (count == 1)

count ++

slide

i ++ ;

j ++ ;

⑥ return ans =

Code: GitHub

Maximum of all subarrays of size k

Fisse algorithm toh same hi lagegi (Fixed sized window)

```
i=0, j=0
while (j < size) {

    // calculation
    if (  < k) {
        j++
    }
    if (  == k) {
        ans ← calculation
        slide the window
    }
}
```

j-i+1 : window size

j-i+1

slide the window → 1) Remove the calculation for i
→ 2) i++, j++

General Format

Eg. IP : 1 3 -1 -3 5 3 6 7

OP : [3, 3, 5, 5, 6, 7]

Ise solve it on DC (Sliding Window Maximum)

Suppose [3 2 1] 5

$max^m = 3$

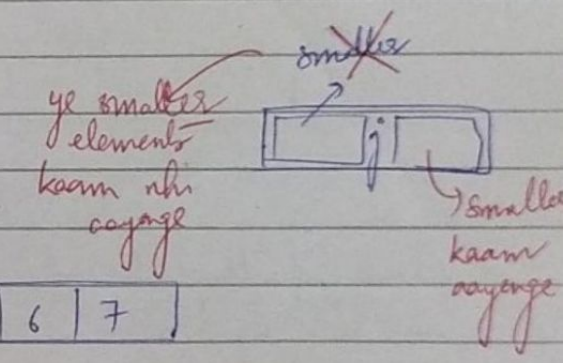
Isme $max^m = 3$

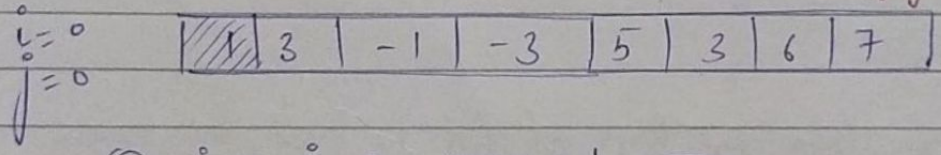
* Fir window slide kete hain : toh i wali calculation ko remove karna hoga
* Fir max var me next max^m store ho jaaye, aisa system kam chahte hain
* ~~3~~ ~~2~~ 1 ... aise

↳ window ka next max^m element store ho jaaye, aisa chahte hain kam

* Think of DS

Degree





① $i=0, j=0$ max = 1

② $i=0, j=1$ max = 3 → ab ye 1 kabhi kaam nhi aayega

~~1~~	3	-1

ans list

↑

3 push karne se pehle check

$3 > 1$

∴ pop 1

③ $i = 0, j = 2$, add -1 to list

④ Now, we've hit the window size

←k=3→

1	3	-1

↑ ye hum pop kar chuke the

list.front() se hume answer milega

Slide the window & remove i ki calculation

1	3	-1	-3	5	3	6	7

↑ j

check if this element is in list or not

(agar ye elem kabhi kaam aaya hoga toh list ke front me present hoga warna present hi nhi hoga)

window

3	-1	-3

5 ↑ j → isse pehle jitne smaller elem honge woh kaam ke nhi honge

~~3~~	~~-1~~	~~-3~~	5

↑ pop ↑ pop

list ↓ isme sirf kaam ke elements rakhenge

Code :- (java)

```
public static int[] maxSlidingWindow(int[] nums, int k) {
    int[] res = new int[nums.length - k + 1];
    int i = 0;
    Deque<Integer> deque = new LinkedList<>();
    for (int j = 0; j < nums.length; j++) {
    while (j < nums.length) {
        while (!deque.isEmpty() && deque.peekLast() < nums[j]) {
            deque.pollLast();
        }   if(j-i+1<k) {j++;}
        deque.add(nums[j]);
                                    if (j-i+1<k) {j++;};
        if(j-i+1 == k) {
            res[i] = deque.peekFirst();
        }

        if (deque.peekFirst() == nums[i]) {
            deque.pollFirst();
        }
        // slide
        i++;
        j++;
    }
    return res;
}
```

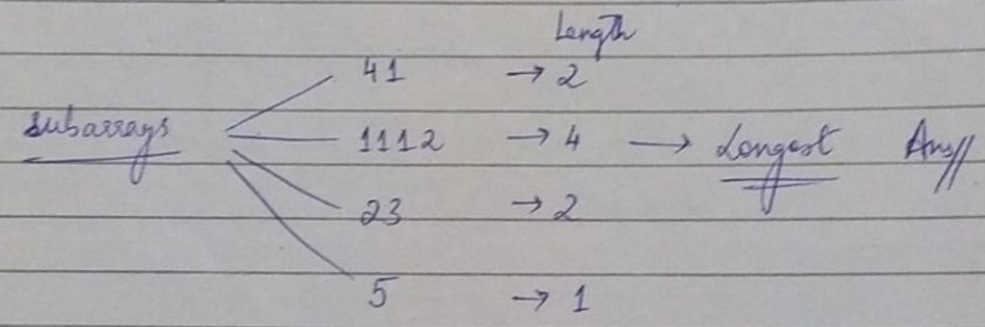
Code - GitHub Repo

Variable Size Sliding Window

Longest subarray of sum K

PS - IP - OP

IP : size : 7
arr : 4 1 1 1 2 3 5
k : 5



OP : 4

* Fixed size easy hoti kyunki window size fix rha tha Yahan pe check kena padega, window size maintain kena aana chahiye

* Identification :-

1. arr | str
2. subarr | substr
3. K → cond^n to maximize or minimize

* Explanation

Fixed	Variable
→ WS given	→ Cond^n given
→ WS = k	→ Sum <= K
→ i=0 j=0 : start	→ i=0, j=0 : start

* Code (Bilkul pehle faise approach kasenge)

```
int sum = 0 ; int max = Integer.MIN_VALUE;

while (j < size) {

    sum = sum + arr[j];
    if (sum < k) {
        j++;
    }
    else if (sum == k) {
        if(j - i + 1 > max) {
            max = j-i+1;
        }
        j++; → for next candidate
    }
    else if (sum > k) {
        while (sum > k) {
            sum = sum - arr[i];
            i++;
        }
        j++;
    }
return max;
```

* jab bhi sliding window ka size k hoga, size ko store karenge.
* size of window : j-i+1

// window is a possible candidate, ye largest ho sakta hai

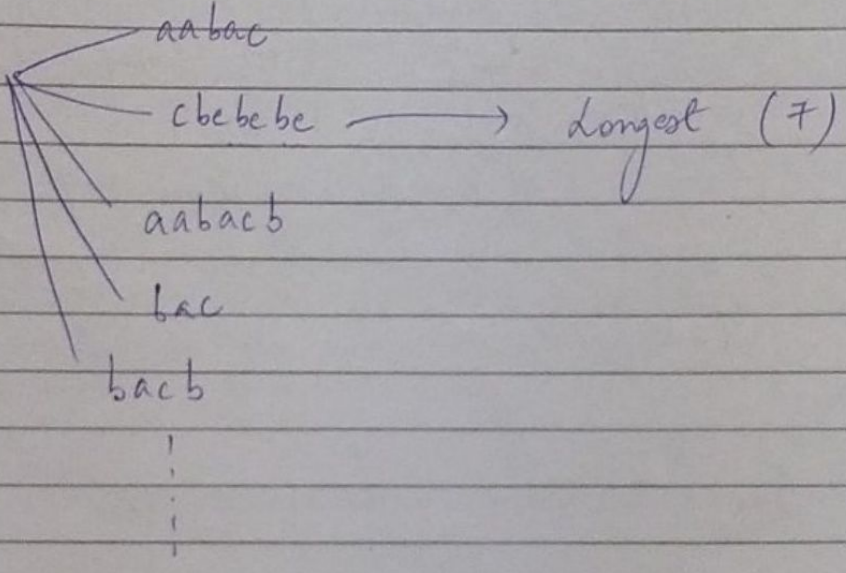
// max = Math.max (max, j-i+1);

* Variable Size WS → General Format

```
while ( j < size ) {
    // calculations
    if ( contd < k )
        j++
    else if ( contd == k ) {
        ans ← calculation
        j++
    }
    else if ( contd > k ) {
        while ( contd > k ) {
            // remove calculation for i
            i++
        }
        j++
    }
return ans
}
```

Longest Substring with K unique characters (Variable Size Window)

IP:- S : aabacbebebe
k : 3



OP :- 7

Code : (Acc to general format) int i=0, j=0, maxi

```
Map<Character, Integer> map = new HashMap<>();
while ( j < s.length() ) {

    // map me add karo
                                    → # UC
    if ( map.size() < k ) {
        j++;
    }
    else if ( map.size() == k ) {   // possible candidate
        maxi = max( maxi, j-i+1 );
        j++;
    }
```

```
else if (map.size() > k) {
    while (map.size() > k) {
        // count decrement;
        if (count == 0) {
            // remove elem from map
        }
        i++;
    }
    j++;
}
return maxi
}
```

map.size()

Code :- Github

Longest Substring ~~With~~ Without Repeating Characters

K

Pichle Ques me : ~~att~~ unique chars the
Iss ques me : all unique chars hain

I/P : S : pwwkew

o/p : 3 (wke)

* Acha iss ques me k hi nhi diya hai (Toh interesting part will be to write the cond^n)

(Ques finance kene me dimag lagaya hai)

* $k = j - i + 1$

↳ puri window me all unique chars chahiye.

* Changes from the prev. code :

① map.size() == j-i+1

② else if (map.size() < j-i+1) {

while (map.size() < j-i+1) {

pwm

iska matlab koi na koi character repeat ho rha hai

↓

Jab tak elem ko remove krte jaenge, jab tak

map.size() < j-i+1

ye false nhi ho jati

}

}

③ First if cond^n not required.

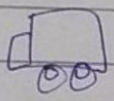
Pick Toys

* Identifying sliding window problem is the key

John in mall → Find toys → Asks Mom to buy it

Mumma gives 2 cond^n :-

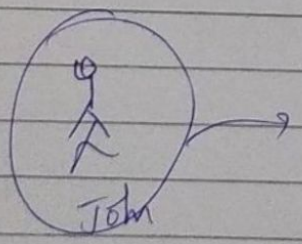
1. Pick toys line se (continuously pick toys)
2. 2 type of toys to pick → 

Jaise  Car → any no. can pick

Animal → any no. can pick

But at any moment, there should be only 2 types of toys in the bag (Those toys can be in any number)

bag



→ He will maximise the toys to buy :)

Eg. Str : aba ccab ⇒ I/P

Ques: Largest substring of 2 unique characters

↑
'k' ke saath ham already kar chuke hain

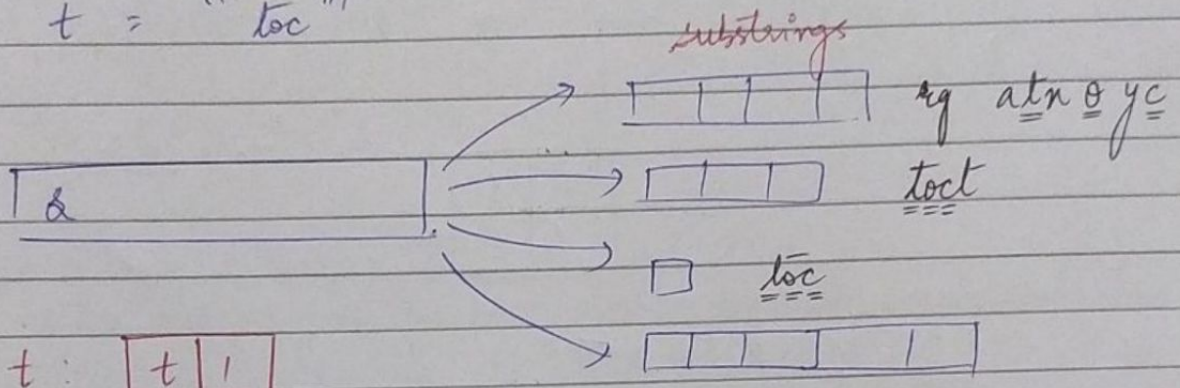
k=2

* Best Question of SW

FAMOUS PROBLEM

Minimum Window Substring

s = "timetopractice"
t = "toc"



t :

t	1
o	1
c	1

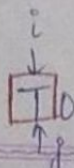
atleast ye granthy toh hona chahiye
(extra mil rhe toh koi problem nhi)

O/P : toprac

⇒ Un saari substrings me se minimum length wali batao → size print karao.

① t string ka map ✓

Eg
Dry run
s = TOTMTAPTAT
t = TTA



① 't' map :

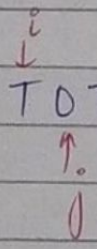
T	2
A	1

count = map.get(t)
count = 2

② If arr[j] in map ⇒ If yes
'T'
↓
decrement count

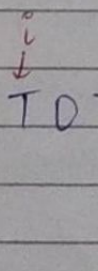
T	1
A	1

③



'O' in map ? No , So move forward

④

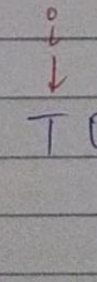


'T' present in map :

T	0
A	1

∴ count --
(count = 1)

⑤



'M' not present in map . move forward

(6) TOTMTAPTAT

T	~~0~~	-1
A	1	

'T' is present in map with count zero

Toh ab count = -1 ho jayega 'T' ka.

(kyunki hame toh min^m T 2 use krne hain, max^m kitne thi, jab T ka count negative hoga, toh hame pata chal jayega ki utne hi 'T' udane hai i wale pointer ke sedi (/ se)

→ entire T ka pata chalega, jitna 'T' ka count -ve me hoga.

(7) TOTMTAPTAT

count = 0, pe possible ans. hoga.

'A' present in map ⇒

T	-1
A	0

count = ~~0~~ 0

~~T~~OT~~T~~MTAPTAT → T : 0

ye toh hai hi abhi map me, to remove easily.

T : 1

> 0

count = ~~0~~ 1

⇒ Yahi increment i, when count = 0

↓

Jaise hi count = 1 hoga, j ko increment karenge

1) Search for an answer → j++

2) Jab ans mil gaya at count == 0

Yahi i++] while count == 0

ye cond^n rehti hai

Code → github, LC, gfg