

# SLIDING WINDOW

..... *Strategy*



**What is subarray** : It is contiguous part of array

1 2 3 4

(1 2) (1 2 3) (1) (2) (3) (4)

(2, 3) (2 3 4)

(3, 4)

**What is subsequence**

It is a seq. of array/string which can be obtained by deleting zero or more elements but order should same.

1	2	3	4
---	---	---	---

1

1 2

1 2 3

1 2 3 4

2

2 3

2 3 4

3

3 4

4

Subarray



1 2 3

1 4

1 2 4

1 3 4

3 4

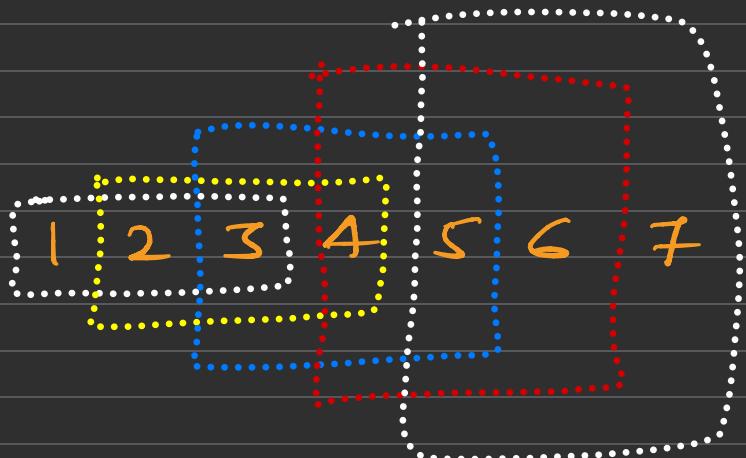
1 2 4

Sub-seq

# Types of Sliding Window (Not an algorithm) Array / String

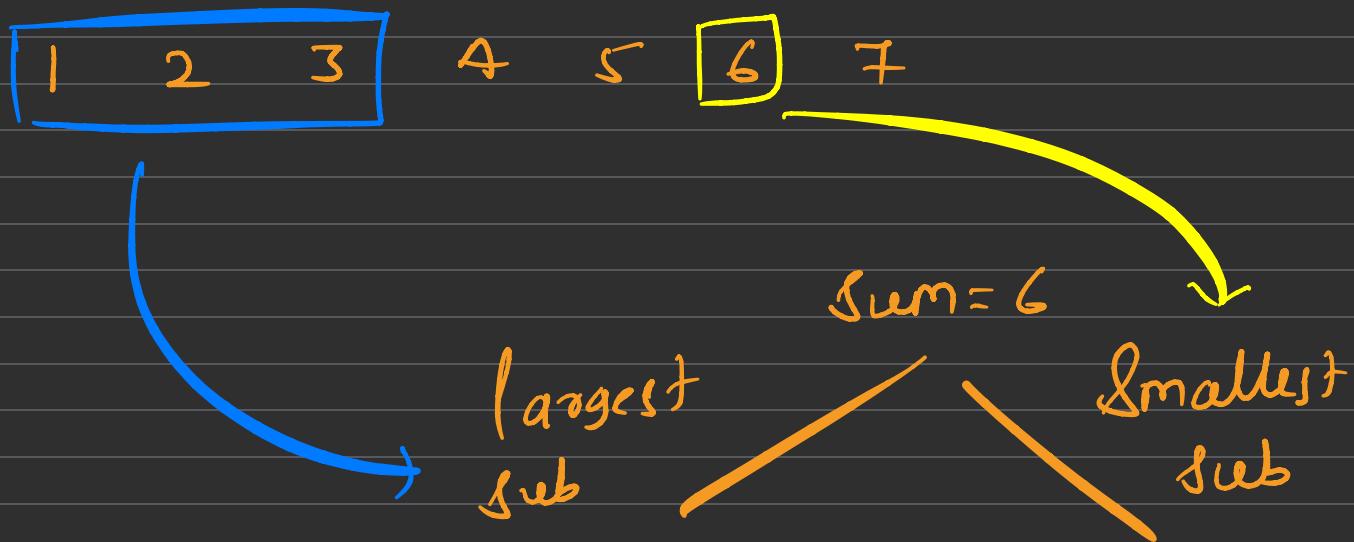
## ✓ Fixed Window Size

## Variable Window Size



$K = 3$  (Window Size)

## \* Variable Window Size



## How to approach the problem

1. Generate all the subarrays/substrings and pick the optimal one
2. Then optimise by moving the window slowly until certain conditions.
3. For variable window size, use the two pointer approach.

Given an array of integers, find all the subarrays

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

```
for ( i=0 : i<n ; i++ )  
{  
    for ( j=i : j<n ; j++ )  
    {  
        for ( k=i : k≤j : k++ )  
        {  
            cout << a(k) )  
        }  
        cout << endl  
    }  
}
```

Given a string, generate all the substrings.

String = abcdfh

Sub - strings

a

abc

ab c d

abc df

abc dfh

b

b c

b c d

b c d f

b c d f h

c

c d

c d f

c d f h

d

df

dfh

f

f h

f h

1	2	3	4
---	---	---	---

1  
1 2 3 4

2  
2 3 4

3  
3 4

4

```
for ( i=0 : i<n ; i++ )
{
    for ( j=i : j<n ; j++ )
    {
        for ( k=j : k<=j ; k++ )
        {
            cout << a(k)
        }
        cout << endl
    }
}
```

1	2	3	4
0	1	2	3

N=4

.

```
for ( i=0 : i<n ; i++ )
```

```
{
    for ( j=i : j<n ; j++ )
}
```

1 2 3 4

3 4

4

.

- given an array, print sum of all subarray
- given an array, find min from all subarray  
(print min. of each sub-array)
- $= // =$  (max)

## Max Sum Subarray of size K



Difficulty: Easy

Accuracy: 49.6%

Submissions: 194K+

Points: 2

Given an array of integers **arr[]** and a number **k**. Return the **maximum sum** of a subarray of size **k**.

Note: A subarray is a contiguous part of any given array.

### Examples:

**Input:** arr[] = [100, 200, 300, 400] , k = 2

**Output:** 700

**Explanation:**  $\text{arr}_3 + \text{arr}_4 = 700$ , which is maximum.

**Input:** arr[] = [100, 200, 300, 400] , k = 4

**Output:** 1000

**Explanation:**  $\text{arr}_1 + \text{arr}_2 + \text{arr}_3 + \text{arr}_4 = 1000$ , which is maximum.

**Input:** arr[] = [100, 200, 300, 400] , k = 1

**Output:** 400

**Explanation:**  $\text{arr}_4 = 400$ , which is maximum.

fixed window

$N=8$

7	2	1	1	4	3	9	1
0	1	2	3	4	5	6	7

$K=3$

$\text{maxSum} = 0$

$O(N-K)$



```
for (i=0 ; i <= n-K ; i++)  
    }  
    sum = 0
```

$O(K)$



```
for (j=i ; j < i+K ; j++)  
    }  
    sum = sum + a(j)
```

$O(N-K) \times O(K)$

$$\boxed{O(N \times K)} - K^2 =$$

$\uparrow$       if ( $\text{sum} > \text{maxSum}$ )

$\uparrow$       }       $\text{maxSum} = \text{sum}$

return  $\text{maxSum}$

7	2	1	1	4	3	9	1
0	1	2	3	4	5	6	7

$$N = 8 \quad K = 3$$



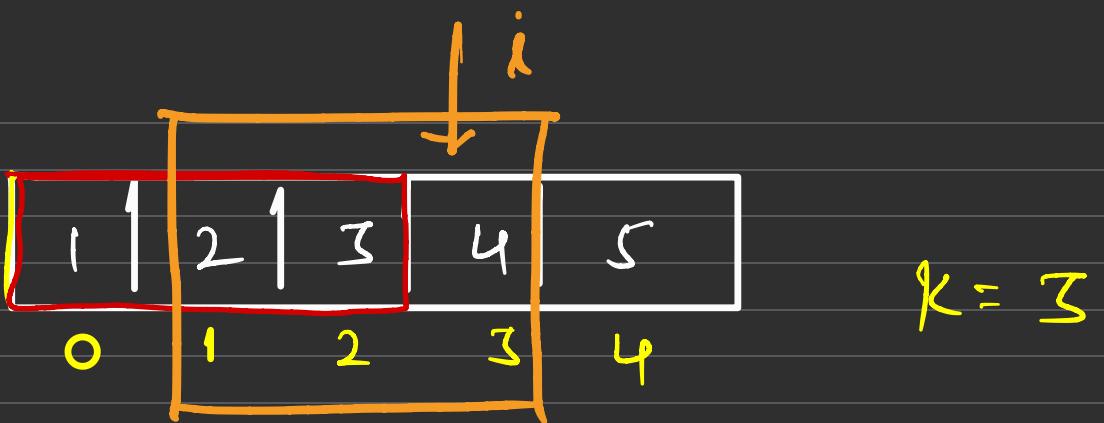
$$1 + \underline{2 + 3} = 6$$

$$\underline{2 + 3} + \underline{4} = 9$$

$$\underline{3 + 4} + \underline{5} = 12$$

$$\underline{4 + 5} + \underline{6} = 15$$

$$\text{Sum} = 9 + 5 - 2 = \underline{\underline{12}}$$



$$k = 3$$

$$i = 3 - k$$

$$= 3 - 3 = 0$$

$$\text{sum} = 6$$

$$\text{sum} = \text{sum} - a(i-k) + a(i)$$

## - Steps for fixed Window

- ① first perform operation on first k elements. ( $0, 1, \dots, k-1^{\text{th}}$  index) ( $i < k$ )
- ② start from  $k$  ( $i = k$ ) & slide forward  
but remove out of windows element  
 $\{ q[i-k] \}$  & add current element  
in window

int maxSubarraySum ( arr[], n, K )  
}      sum = 0

for ( i=0 ; i < K ; i++ ) — Step 1  
    sum += arr[i]

maxSum = sum

for ( j=K ; j < n ; j++ ) — Step 2  
}      sum = sum - arr[j-K] — Removed  
                out of window.

sum = sum + arr[i]

— Add current element

maxSum = max ( sum, maxSum )

}

Given an array **arr[]** and an integer **K**, the task is to calculate the sum of all subarrays of size **K**.

*Given an array of integer, task is to find the subarray with minimum sum of size k*

given an array, find min No. from every sub.  
of size K.

7	1	1	1	2	3	7
0	1	2	3	4	5	6

K=2

```
for (i=0 ; i <= N-K ; i++)
```

;

    num = a(i)

```
        for (j=i ; j < i+K ; j++)
```

{

```
            if (a(j) < num)
```

}

        num = a(j)

print \n / num

7	1	1	1	.	2	3	7
0	1	2	3	4	5	6	

$k = 3$

map

$\{7, 1\}$
$\{3, 1\}$
$\{2, 1\}$

1.4 perform ppn first  $k$

②  $i = k$

$i = 3486$

map  
 $\{ \text{key, value} \}$



```
void printMin (arr[], N, K)
```

```
{ map<int, int> mp
```

```
    for (i=0 ; i < K ; i++)
```

```
        mp(a(i))++
```

— Step ①

begin of  
(min) {  
key first  
value second

```
cout << (*mp.begin()).first
```

```
for (i = K ; i < n ; i++)
```

```
} int out = a(i-K)
```

```
mp(out)--
```

```
if (mp(out) == 0)
```

```
    mp.erase(out)
```

a(i-K)

x = a(i-K)

int cur = a(i)

mp[{cur}]++

cout << (\*mp.begin()).first

}

{

## Variable Sliding Window

fixed :      1 2 3 4 5 6       $k = 3$   
(len of sub)  
find max sum

variable :      1 2 3 4 5 6      sum = 5  
find largest sub. size  
find smallest sub. size

Given an array, find the maximum length of subarray with sum K.

(Sum)

1	2	3	2	4	1	1
0	1	2	3	4	5	6

K = 6

```
for (i=0 ; i<n ; i++)
```

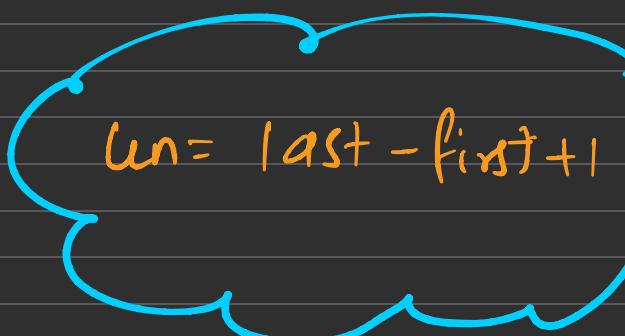
```
}      sum=0
```

```
    for (j=i ; j<n ; j++)
```

```
}      sum += a(j)
```

```
    if (sum == K)
```

```
}      len = max(len, j-i+1)
```


$$\text{len} = \text{last} - \text{first} + 1$$

return len

(  $r = \text{Expand}$ ) Add  
(  $l = \text{Shrink}$ ) Subtract

<del>K</del>	<del>K</del>	$K=6$	<del>K</del>	<del>K</del>	$l$		
1	2	3	2	4	1	1	
0	1	2	3	4	5	6	

~~X~~   ~~X~~   ~~X~~   ~~X~~   ~~X~~   ~~X~~   ~~X~~    $r$

$$\text{Sum} = \emptyset \times 368789$$

$$\text{len} = \emptyset \boxed{3}$$

~~67~~  
~~46~~

```
left = 0, right = 0  
sum = 0, len = 0  
while (right < n)  
    sum += a[right]
```

```
    if (sum > k)  
        sum = sum - a[left]  
        left++
```

```
    if (sum == k)  
        len = max (len, right - left + 1)
```

```
right++
```

```
return len
```

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

$K = 6$

$left = 0 \times 2 \times 3 \times 4 \times 5$

$right = 0 \times 1 \times 2 \times 3 \times 4 \times 5 \times 6 \neq 8 \times 9 \times 10 \neq 11$

$sum = 0 + 2 + 3 + 4 + 5 + 6 \neq 8 + 9 + 10 \neq 8$

$6 \neq 3 + 4 + 5 + 6$

Given an array, find the minimum length of subarray with sum K.

Given an array, find the maximum length of subarray with sum at least K.

Given an array, find the maximum length of subarray with sum at most K.

Given an array, find the number of subarray with sum equal to K.

cnt = 0

for (i=0 ; i < n ; i++)

}      sum = 0

for (j = i ; j < n ; j++)

}      sum += a(j)

if (sum ≥ k)

    cnt++

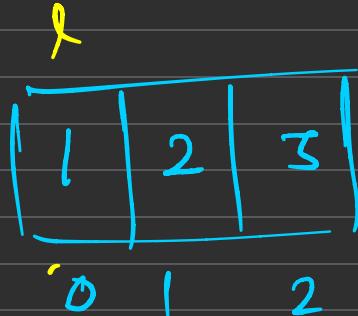
}

return cnt

1	2	3	3	4	2
0	1	2	3	4	5

K = 6

# How to Count Subarrays?



No. of subarrays

11

sum of len of  
all subarrays

