SLIDING WINDOW

Strategy

What is subarray

: It is contigious 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

X

1 2 3

1 3 4

1 4

3 4

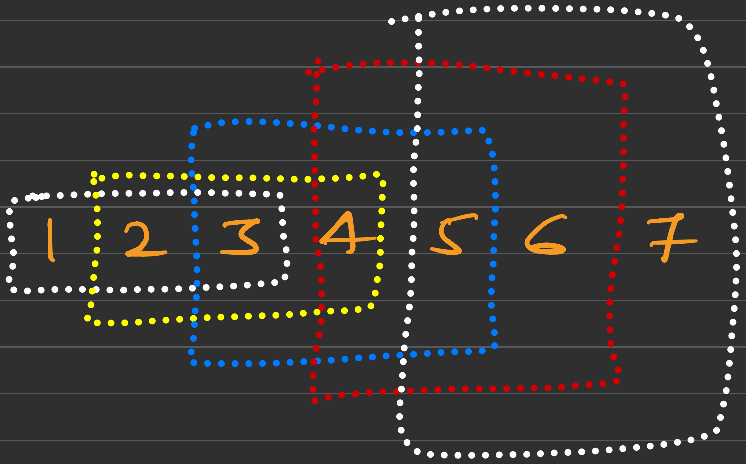
1 2 4

Sub-seq

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

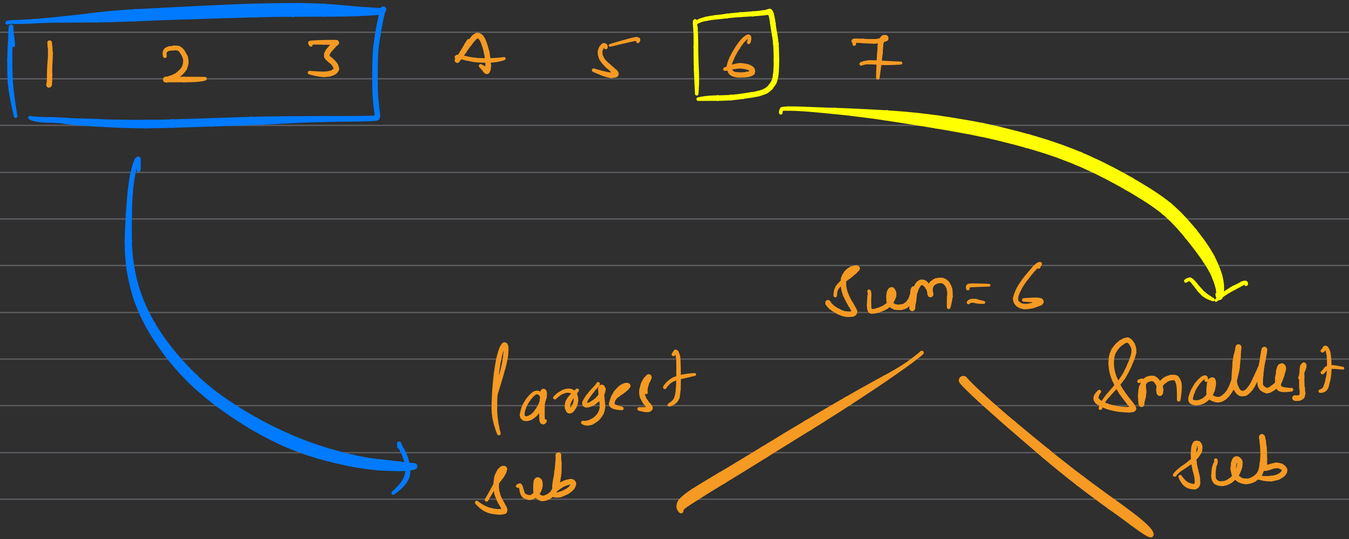
✓ Fixed Window Size

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++)
        {     println ( a(k))
        }
        println("\n")
    }
}
```

Given a string, generate all the substrings.

String = abcdfh

Sub-strings

a
abc
abcd
abcdf
abcdfh

b
bc
bcd
bcdf
bcdfh

c
cd
cdf
cdfh

d
df
dfh

f
fh

h

1	2	3	4

1
1 2
1 2 3
1 2 3 4

2
2 3
2 3 4

3
3 4

4

```
for ( i=0 : i<n ;i++)
{
    for ( j= i : j <n ;j++)
    {
        for ( k= i : k ≤ j : k++)
        {
            println ( a(k))
        }
        println("\n")
    }
}
```

1	2	3	4
0	1	2	3

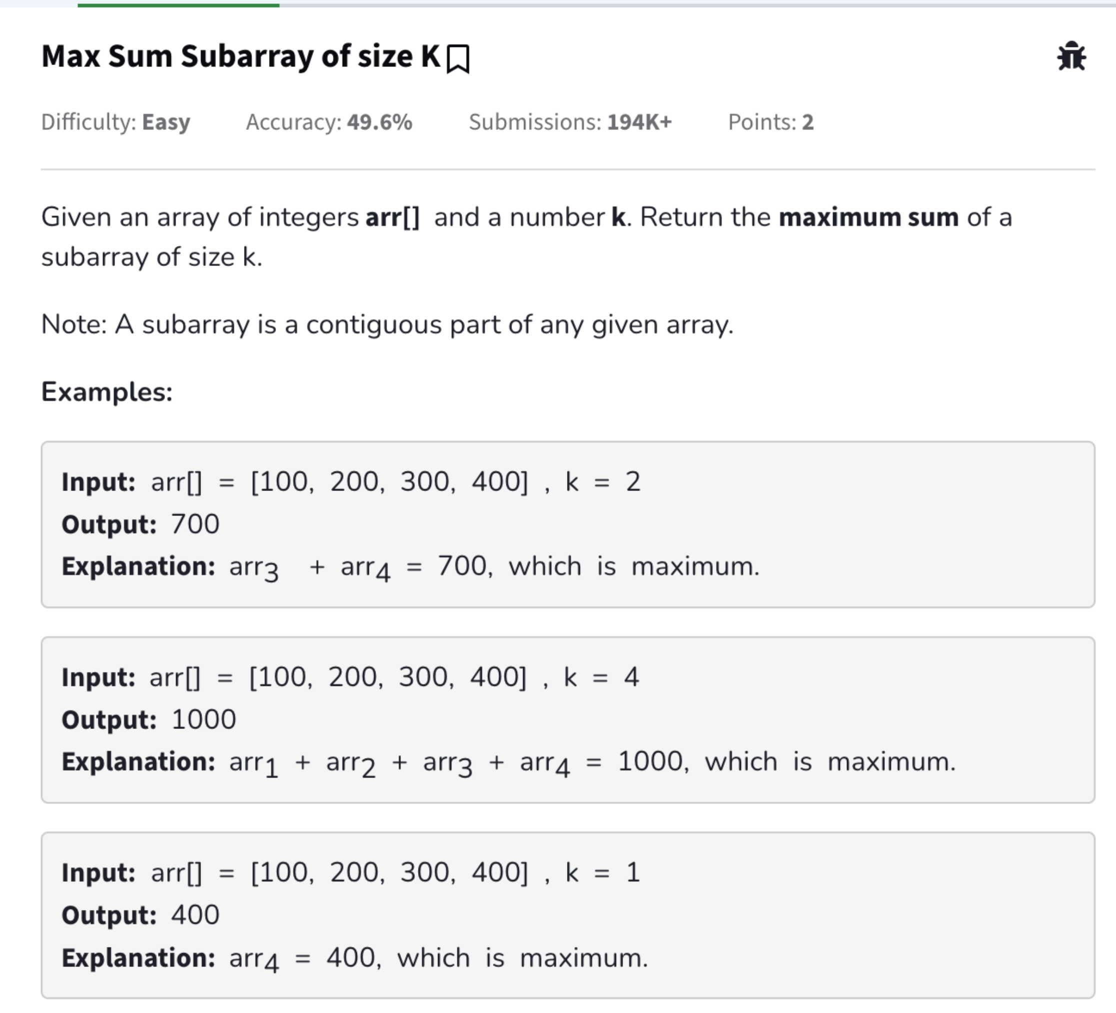
N=4

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

1 2 3 4
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.

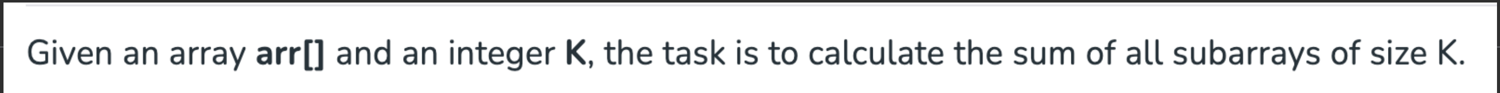
Note: A subarray is a contiguous part of any given array.

Examples:

Input: arr[] = [100, 200, 300, 400] , k = 2
Output: 700
Explanation: $arr_3 + arr_4 = 700$, which is maximum.

Input: arr[] = [100, 200, 300, 400] , k = 4
Output: 1000
Explanation: $arr_1 + arr_2 + arr_3 + arr_4 = 1000$, which is maximum.

Input: arr[] = [100, 200, 300, 400] , k = 1
Output: 400
Explanation: $arr_4 = 400$, which is maximum.



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