Made with Goodnotes

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.

Made with Goodnotes

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

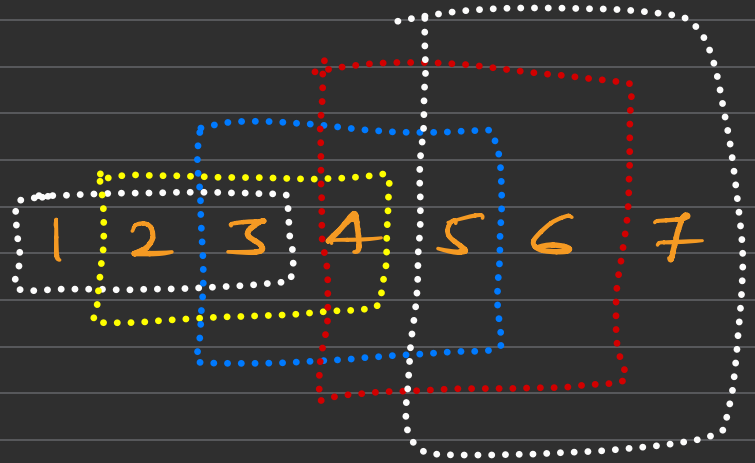
Sub-seq

Made with Goodnotes

Types of Sliding Window (Not an algorithm)

Array / String

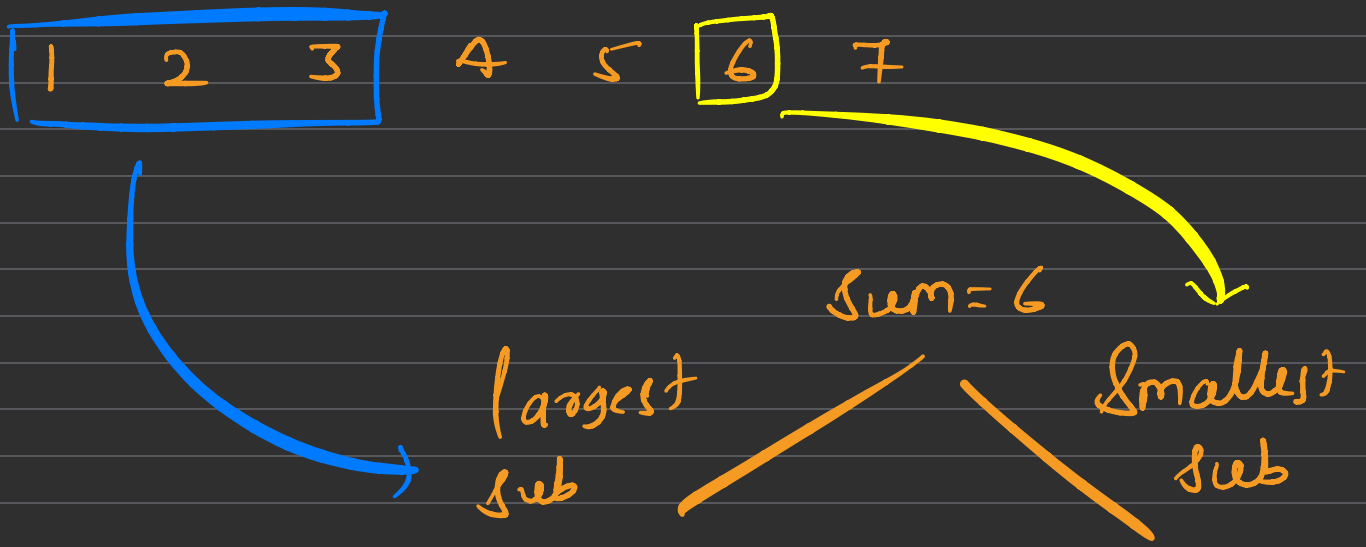
- ✓ Fixed Window Size
- Variable Window Size



K = 3 (Window Size)

Made with Goodnotes

* Variable Window Size



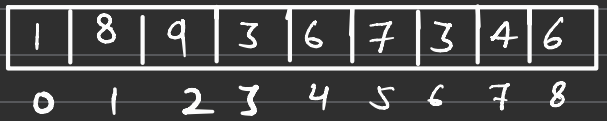
Made with Goodnotes

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.

Made with Goodnotes

Given an array of integers, find all the subarrays




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

Made with Goodnotes

Given a string, generate all the substrings.

String = abcdfh

Sub-strings

a	b	c	d	f	h
abc	bc	cd	df	fh	
abcd	bcd	cdf	dfh		
abcdf	bcdf	cdfh			
abcdfh	bcdfh				

Made with Goodnotes

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")
    }
}
```

N=4

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

1 2 3 4

2 3 4

3 4

4

```
}
}
```

Made with Goodnotes

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

Made with Goodnotes

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.

Made with Goodnotes

fixed window

N = 8

K = 3

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

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)`

`}`

`if (sum > maxSum)`

`{ maxSum = sum`

`}`

`}`

`return maxSum`

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

$O(N \times K) - K^2$

Made with Goodnotes

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

N = 8 K = 3

1 2 3 4 5 6 K=3

1 + 2 + 3 = 6

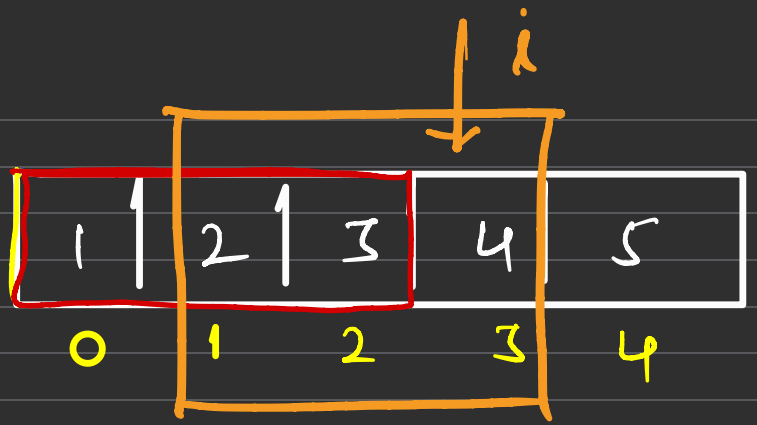
2 + 3 + 4 = 9

3 + 4 + 5 = 12

4 + 5 + 6 = 15

Sum = 9 + 5 − 2 = 12

Made with Goodnotes



$k = 3$

$i = 3 - k$

$= 3 - 3 = 0$

Sum = 6

$$Sum = Sum - a(i-k) + a(i).$$

Made with Goodnotes

_ Steps for fixed Window

① first perform operation on first k elements. (0, 1 .... $k-1^{th}$ index) ($i < k$)

② Start from k ($i = k$) & slide forward but remove out of Windows element { a[i-k] } & add current element in window

Made with Goodnotes

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

    for (i=0 ; i<K ; i++)                    ——— Step 1
        sum += a(i)

    maxSum = sum

    for (i= K ;  i<n ; i++)                  ——— Step 2
    {     sum = sum - a(i-K)                 ——— Removed out of Win.

          sum = sum + a(i)                   ——— Add curr element

    maxSum = max (sum, maxSum)
}
```

Made with Goodnotes

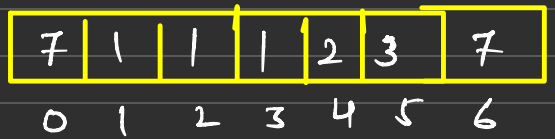
Given an array **arr[]** and an integer **K**, the task is to calculate the sum of all subarrays of size K.

Made with Goodnotes

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

Made with Goodnotes

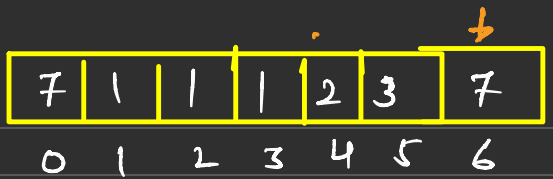
given an array, find min no. from every sub. of size K.



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)
    println( num
```

Made with Goodnotes



k = 3

map

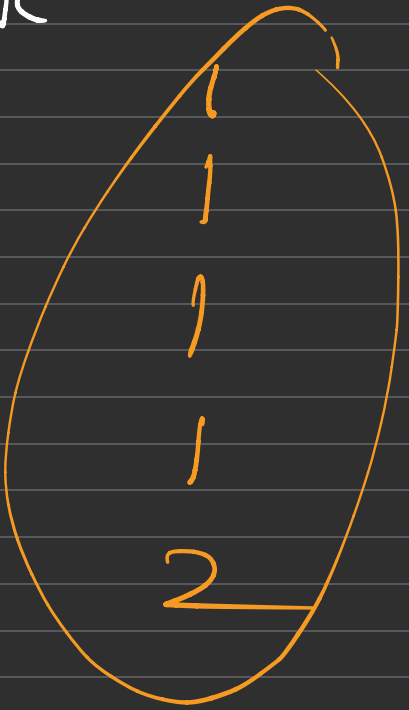
{7, 1}
{3, 1}
{2, 1}

map
{key, value}

1. Perform pattern first k

② i = k

i = ~~3 4 5~~ 6



Made with Goodnotes

```
void printMin(arr[], N, K)
{
    map<int, int> mp
    for (i=0 ; i<K ; i++)
        mp[a(i)]++
```

— step ①

begin (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))

Made with Goodnotes

```
int cur = a[i]
mp[cur]++
cout << ((*mp.begin()).first
}
}
```

Made with Goodnotes