Maximum Sum Sub array Of size K

1. Identification
2. Prob Statement, Input, Output
3. Abstract → Code
4. Code

Input size 7

arr[] : 2 5 1 8 2 9 1.

Given an array of integers Arr of size N and a number K Return The maximum sum of subarray of size K

2 5 1
5 1 8
⋮
2 9 1 — Max Sum

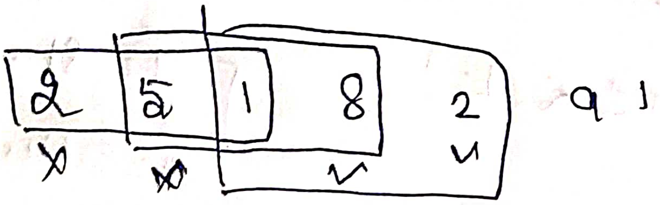
arr[] : 100, 200, $\boxed{300, 400}$

K = 2

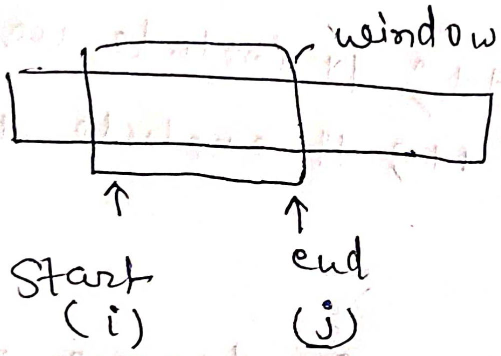
Output 700 (Arr3 + Arr4)
Max

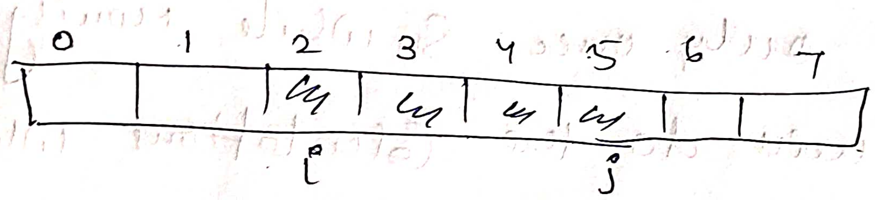
Identification —
- arr + substring
- Sub array (substring
- window size

⇒ Sliding window problem



- every time take 1 and remove 1 from 1st index
- Maintain the size of window.



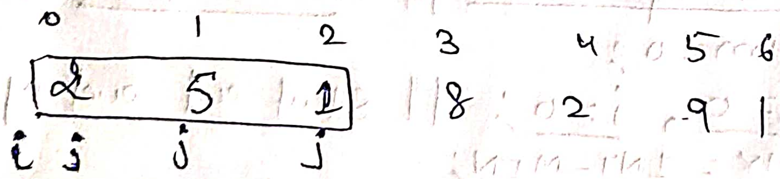


windowsize = 4

$j - i = 5 - 2 = 3$

- So $\boxed{j-i+1}$ Represent window size

window size = $\boxed{j - i + 1}$



$j-i+1 < K \Rightarrow$ increase upto $\boxed{j-i+1 == K}$

```
{ j++;
}
```

- Once we get size of window then maintain its size every time.

```
when((j-i+1) == K) → cond
{
    Calculation // to maintain its size
}
```

```
int i=0, j=0;

  If ( j-i+1 < k)          [ i= start of window
                             j= end of window
    {  j++;
    }

  If ( j-i+1 == K)

    {
  Maintain      { j++;  // include next element to window
  window        { i++;  // exclude 1st element from wind
  size
    }
```

→ Because we will traverse over each element only once so while moving j we will do the calculations which is sum in our case.

Code (size of array, window size(K), arr)

```
int sum=0;
int i=0, j=0;  // start and end of window.
int mx = INT-MIN;
While (j< arr.size())  // j Goes upto end.
{
   sum = sum + arr[j];  // it is valid when sub-
                        // array is size of K
   [ If(j-i+1 < K)
        j++;       ]  Making window

   else If (j-i+1 == k)
  {  Mx = max(mx, sum);
//NOTE: Sum always Contains the sum of
//        elements in the window

    Sum = sum - arr[i];

          i++;
          j++;

}
```

Final Code

```
    int  MaxSumwindow(arr, n, k)
  {  int sum = 0
     int i=0; // start of window
     int j=0; // end of window
     int ~~max~~ mx = INT_MIN;
  while (j < arr.size())
                n
  {  sum = sum + arr[j];
     If (j-i+1 < K)   j++

     else If ((j - i + 1) == k)

        mx = max (mx, sum);

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