[Sliding Window]

1. Identification :- We might have to use sliding window if :-

⇒ arr or string → + Subarray or Substring → + largest max^m, min^m etc

+ K = window size (given) → maybe given in Q or need to be calculated.

2. Types :-

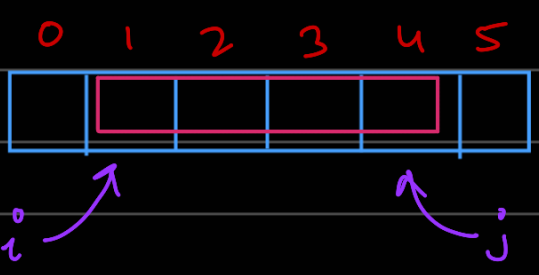
Sliding Window

- **Fixed**
  - → fixed window size.
  - K → fixed size
- **Variable**
  - → No window size is given
  - K → variable size
  - → We might have to calculate largest window / smallest window. acc^ to some cond^ given.
    - Need to calculate 'k'.

I. Fixed size window Problems :-

1. Max^m sum in subarray of size 'k' :-

let $k = 4$

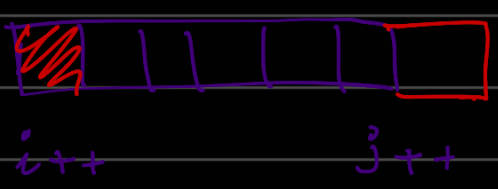


size $\Rightarrow (j - i) + 1$

→ Start 'i' & 'j' from 0th index

→ Start moving 'j' untill the desired window size is achieved. Maintain this size now.
↳ While moving 'j' → we can calculate the sum

→ Move the sliding window according to the cond^n in the question.
↳ Now you have to perform some operations on the current window



pseudo code ⇒

```
while( j < size)
{
    sum = sum + arr[j] ;
}
```

keep calculating sum while moving 'j'

move 'j' only unless the desired window size is achieved.

```
if( j-i+1 < k ){
    j++;
}
```

Move the window, but first remove the 'i' the element from the prev. sum.

```
else if( (j-i)+1 == k) {
    max = Math.Max(sum, max);
    sum = sum - arr[i];
    i++;
    j++;
}
return max;
```
Ans

(2.) First (-)ve integer in every window of size 'k'

pseudo code ⇒

```
int ans;
int[] queue   → to store the (-)ve no(s) in FIFO order
while (j < size) {
    if(arr[i]<0)
    { queue.add(arr[i])
    }
```

<- calculations - - - ->
<- - - with 'j' - - - - - ->

```
if ((j-i)+1 < k) {
    j++;
}
else if ((j-i)+1 == k)
{
    1→ Calculate ans.
```

<--- ---->

2 → Slide the window

```
if (queen.size() == 0)
    ans.add(0);
else {
    ans.add(queen[0])
    if (arr[i] < 0) {
    }    queen.pop(0)

    i++;
} j++;
return ans;
```