PW SKILLS

# Sliding Window

# Utility of sliding window

- Subarray
- Substrings
- Types : K sized / Variable Sized

arr = {10, 1, 20, 3, 40, 31, 89}

# Basic Algorithm :

**Q : Maximum sum Subarray of size k.**

arr = { 10, 20, 1, 3, -40, 80, 10 }  K=3

Method 1 : Brute Force

↓

Check for all K sized subarrays

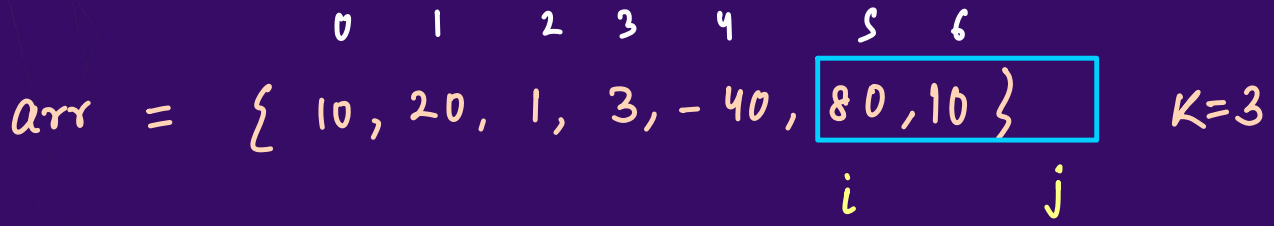
↓

n-K+1

T.C. = $O(n*K)$ ~ $O(n^2)$ if K is almost like 'n'

# Basic Algorithm :

**Q : Maximum sum Subarray of size k.**




Two consecutive windows of size 'K' have 'K-1' elements common

maxSum = 31 43 So

sum = 31 24
-36
43 So

$$Sum = sum - arr[i-1] + arr[j]$$

# Basic Algorithm :

PW SKILLS

**Q : Maximum sum Subarray of size k.**

```
int maxSum = 0;
int i=0,j=k-1,sum=0;
for(int a=0;a<=k-1;a++){ // k times
    sum += arr[a];
}
i++; j++;
while(j<n){ // n-k times
    sum = sum - arr[i-1] + arr[j];
    maxSum = Math.max(maxSum,sum);
    i++; j++;
}
System.out.println(maxSum);
```

Optimised Approach

↓

T.C. = O(n)

# Ques:

PW SKILLS

**Q1 : Number of Subarrays of size K and Average greater than or equal to Threshold**

arr = { 2, 2, 2, 2, 5, 5, 8 }  K = 3,  t = 4

{2, 5, 5} , {5, 5, 8}

1 1

4 6

ans = 2

1

count

avg = sum/K

[Leetcode 1343]

# Ques:

PW SKILLS

## Q2 : Minimum size subarray sum.

arr = { 2, 3, 1, 2, 4, 3 }  target = 7
i  j

{2, 3, 1, 2}

{3, 1, 2, 4}

{1, 2, 4}

{2, 4, 3}

{4, 3}  → ans = 2

Brute Force Soln

↓

T.C. = $O(n^2)$

[Leetcode 209]

# Ques:

PW SKILLS

## Q2 : Minimum size subarray sum.

arr = { 2, 3, 1, 2, 4, 3 }  target = 7
i  j

sum = ~~0~~ ~~8~~ 6

len = 4

minLen = 4

[Leetcode 209]

# Ques:

PW SKILLS

**Q2 : Minimum size subarray sum.**

arr = { 4, 1, 2, 3, 2, 1 }  target = 7

i  j

[Leetcode 209]

# Ques:

## Q2 : Minimum size subarray sum.

```java
public int minSubArrayLen(int target, int[] arr) {
        int n = arr.length, minLen = Integer.MAX_VALUE;
        int i = 0, j = 0, sum = 0;
        while(j<n && sum<target){ // first window
            sum += arr[j++];
        }
        j--;
        // sliding window
        while(i<n && j<n){
            int len = j-i+1;
            if(sum>=target) minLen = Math.min(minLen,len);
            sum -= arr[i];
            i++; j++;
            while(j<n && sum<target){
                sum += arr[j++];
            }
            j--;
        }
        if(minLen==Integer.MAX_VALUE) return 0;
        return minLen;
    }
```

[Leetcode 209]

# Homework:

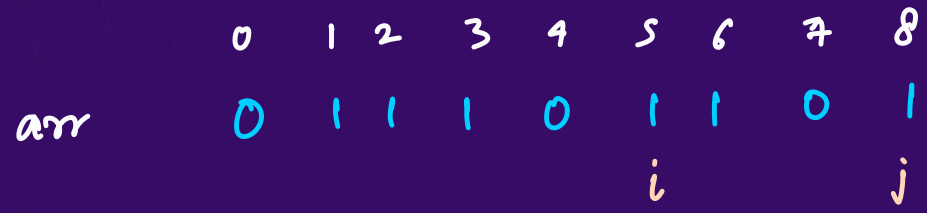
**Q : Subarray Product Less than K.**

**[Leetcode 713]**

# Ques:

PW SKILLS

**Q3 : Longest subarray of 1's after deleting one element.**




zeroes = 0 1

maxlen = 0 5

len = 8 3

[Leetcode 1493]

# Ques:

## Q3 : Longest subarray of 1's after deleting one element.


```
while(j<n){
            if(arr[j]==1) j++;
            else{ // arr[j]==0
                if(zeroes==0){
                    j++;
                    zeroes++;
                }
                else{ // zeroes==1
                    int len = j-i-1;
                    maxLen = Math.max(maxLen,len);
                    j++;
                    while(i<n && arr[i]==1) i++;
                    i++;
                }
            }
        }
```

[Leetcode 1493]

# Homework:

**Q : Max consecutive ones III.**

**[Leetcode 1004]**

# Ques:

## Q4 : Grumpy Bookstore owner

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| cust | 1 | 0 | 1 | 2 | 1 | 1 | 7 | 5 |
| grumpy | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

K = 3

1 + 1 + 1 + 7 = 10 people are satisfied

0 + 2 + 1 + 5 = 8 people not satisfied

[Leetcode 1052]

# Ques:

## Q4 : Grumpy Bookstore owner

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| cust | 1 | 0 | 1 | 2 | 1 | 1 | 7 | 5 |
| grumpy | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |

K=3

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| cust | 1 | 0 | 1 | 2 | 1 | 1 | 7 | 5 |
| grumpy | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

K=3 ✗ wrong

[Leetcode 1052]

# Ques:

## Q4 : Grumpy Bookstore owner

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| cust | 1 | 0 | 1 | 2 | 1 | 1 | 7 | 5 |
| grumpy | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |

K = 3

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| cust | 1 | 0 | 1 | 2 | 1 | 1 | 7 | 5 |
| grumpy | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

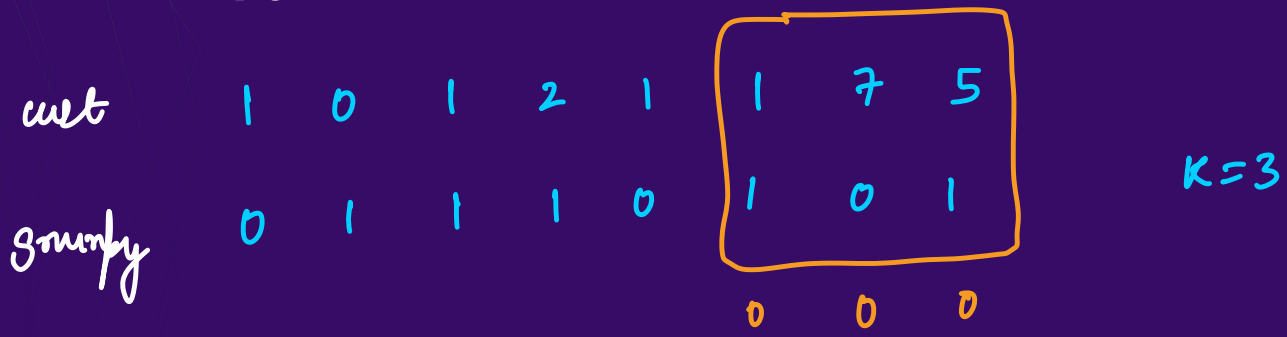
K = 3

1 + 1 + 1 + 7 + 5 = 15 people satisfied

[Leetcode 1052]

# Ques:

## Q4 : Grumpy Bookstore owner




unsatisfied Count = 1 3 8 3 x 6

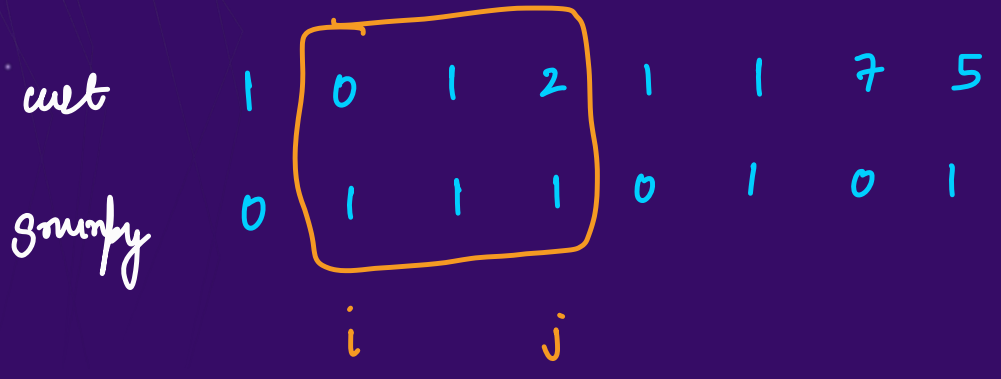
maxUnsatisfied = 0 1 8 6

a = 5 , b = 7

[Leetcode 1052]

# Ques:

## Q4 : Grumpy Bookstore owner




K=3

unsatisfied = 3

[Leetcode 1052]

# Ques: hint → hashmap → map<type, freq>

PW SKILLS

## Q5 : Fruit into Baskets [Pick Toys]

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

Q// Find the length of largest subarray that contains atmost 2 distinct elements

2 baskets

Create hashmap, add elements & reduce the window if map.size() > 2

i to j

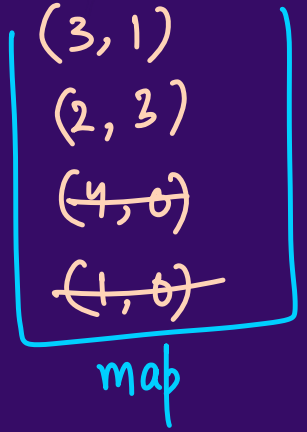
[Leetcode 904]

# Ques:

## Q5 : Fruit into Baskets

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 4 | 1 | 4 | 1 | 2 | 3 | 2 | 2 |
| | | | | | | | i | | | j |

T.C. = O(n)




maxlen = 0 1 2 3 4 5

len = 1 2 3 2 3 4 3 4 4 3 4

[Leetcode 904]

# Ques:

PW SKILLS

**Q6 : Count number of nice subarrays.**

arr = 2 9 6 3 8 K=2

→ 0 1 0 1 0

9, 6, 3

2, 9, 6, 3

9, 6, 3, 8

2, 9, 6, 3, 8

Count = 4

[Leetcode 1248]

# Ques: Method-1 : Sliding window

PW SKILLS

## Q6 : Count number of nice subarrays.

arr =

| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | a | i | | | | | | j | | b | |

K = 3

K2 = प्रॉब्लम 3

Count = 0

[Leetcode 1248]

# Ques:

PW SKILLS

## Q6 : Count number of nice subarrays.

```
public int numberOfSubarrays(int[] arr, int k) {
    int i = 0, j = 0, a = 0, b = 0, n = arr.length, k2 = 0;
    int count = 0;
    while(i<n && arr[i]%2==0) i++;
    while(j<n && k2<k) if(arr[j++]%2!=0) k2++;
    if(k2<k) return 0;
    j--;
    b = j + 1;
    while(b<n && arr[b]%2==0) b++;
    b--;
    while(b<n){ // sliding window
        count += (i-a+1)*(b-j+1); // math
        a = i + 1;
        i++;
        while(i<n && arr[i]%2==0) i++;
        j = b + 1;
        b = j + 1;
        while(b<n && arr[b]%2==0) b++;
        b--;
    }
    return count;
}
```

Sliding window approach

T.C. = O(n)

A.S. = O(1)

[Leetcode 1248]

# Ques: Method-2 : Prefix-Sum & Hashmap <ele, first idx>

PW SKILLS

## Q6 : Count number of nice subarrays.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| arr = | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| pre = | 0 | 1 | 1 | 2 | 2 | 2 | 3 | 3 |
| | | a | | b | | | | i |

K = 2

ans = 10

(2,3)
(1,1)
(0,0)
map

→ arr[i] - K = a

→ arr[i] - K + 1 = b

cont = 0 2 X 0 8 10

b - a + 1 / b - a

[Leetcode 1248]

# Ques:

## Q6 : Count number of nice subarrays.

```java
public int numberOfSubarrays(int[] arr, int k) {
    int n = arr.length, count = 0;
    for(int i=0;i<n;i++) arr[i] %= 2;
    for(int i=1;i<n;i++) arr[i] += arr[i-1];
    Map<Integer,Integer> map = new HashMap<>();
    for(int i=0;i<n;i++){
        if(!map.containsKey(arr[i])) map.put(arr[i],i);
        int a = 0;
        if(map.containsKey(arr[i]-k)) a = map.get(arr[i]-k);
        int b = 0;
        if(map.containsKey(arr[i]-k+1)) b = map.get(arr[i]-k+1);
        if(arr[i]==k) count += (b-a+1);
        if(arr[i]>k) count += (b-a);
    }
    return count;
}
```

T.C. = O(n)

S.C. = O(n)

[Leetcode 1248]

# THANK YOU