SKILLS

Sliding Window

Lecture-35

Raghav Garg

Utility of sliding window

- **Subarray** → prefix sum? Sliding window? PS, SW dono
- **Substrings**
- **Largest / Smallest sum**
- In a window / subarray of given size 'K'

Ques : Maximum sum Subarray of size k

0	1	2	3	4	5	6	7	8
7	1	2	5	8	4	9	3	6

arr = { 7, 1, 2, 5, 8, 4, 9, 3, 6 }

n = 9

K = 4

{7,1,2} , {1,2,5}, {2,5,8} , {5,8,4} , {8,4,9}

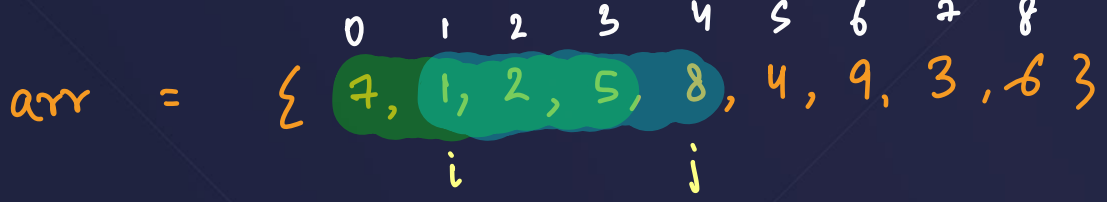
{4,9,3} , {9,3,6}

i = 0 to i = n-k

↳ for(int j = i ; j < i-k ; j++)

Ques : Maximum sum Subarray of size k

Let us use the sliding window algorithm



blue sum = green sum + 8 - 7 ✓

curr window sum = prev window sum + arr[i] - arr[i-1]

	0	1	2	3	4	5	6	7	8
arr = {	7,	1,	2,	5,	8,	4,	9,	3,	6 }
			i			j			

$n = 9$

$K = 4$

maxSum = INT_MIN

prevSum = 15

i = 1

j = K

currSum = 16

```
while (j < n){
    int currSum = prevSum + arr[j] - arr[i-1];
    if (maxSum < currSum){
        maxSum = currSum;
        maxIdx = i;
    }
    prevSum = currSum;
    i++;
    j++;
}
```

Ques : Grumpy Bookstore owner [Leetcode - 1052]

```
// [1,0,1,2,1,1,7,5]  → customers
// [0,1,0,1,0,1,0,1]   3 min
// [0,1,0,1,0,0,0,0]
```

grumpy modified

0 0 0 1 0 1 0 1

Brute force : try every window

→ T.C. = $O((n-k)*n)$

= $O(n^2)$

Sliding window algo

Hint : Find that window that has most 'loss' of satisfaction ✓

↳ != the window with the least satisfaction ✗

Sliding Window Algo

```
// [1,0,1,2,1,1,7,5] arr
// [0,1,0,1,0,1,0,1]  3 min
```

grv

i j

maxLoss

maxIdx

loss of satisfaction , → los

green los = blue los + arr[j] − arr[i-1]

- green los → currLoss
- blue los → prevLoss
- arr[j] → only if grv[j]==1
- arr[i-1] → only if grv[i-1]==1

customers =

[9,10,4,5]

i

j

grumpy =

[1,0,1,1]

minutes =

1

```
int prevLoss = 0;
for(int i=0;i<k;i++){
    if(grumpy[i]==1) prevLoss += arr[i];
}
int maxLoss = prevLoss;
int maxIdx = 0;
int i = 1;
int j = k;
```

prevLoss = 0 9

maxLoss = 9

maxIdx = 0

i = 1

j = 1

customers =

0	1	2	3	4
[9	10	4	5]	
			i	
			j	

grumpy =

[1,0,1,1]

minutes =

1

```
while(j<n){
    int currLoss = prevLoss;
    if(grumpy[j]==1) currLoss += arr[j];
    if(grumpy[i-1]==1) currLoss -= arr[i-1];
    if(maxLoss<currLoss){
        maxLoss = currLoss;
        maxIdx = i;
    }
    prevLoss = currLoss;
    i++;
    j++;
}
// filling 0s in the grumpy array window
for(int i=maxIdx;i<maxIdx+k;i++){
    grumpy[i] = 0;
}
// sum of satisfaction
int sum = 0;
for(int i=0;i<n;i++){
    if(grumpy[i]==0) sum += arr[i];
}
return sum;
```

i < maxIdx + k

currLoss = 9 0 4 5 1

prevLoss = 0 9 0 4 1

maxLoss = 9

maxIdx = 0

i = 1 2 3 4

j = 1 2 3 4

Ques : First negative number in every window of size k

arr = { 2, -3, 4, 4, -7, -1, 4, -2, 6 } K=3

ans = { -3, -3, -7, -7, -7, -1, -2 } → n-K+1

Brute Force : O(nK) ×

Optimised → Sliding Window → O(n) ✓

	0	1	2	3	4	5	6	7	8
arr =	2	-3	4	4	-7	-1	4	-2	6

K = 4

i = start of window

ans = {-3, -3, -7, -7, -7, -1}

```
while(j<n){ // n iterations
    if( p >= i) ans[i] = arr[p];
    else {
        for[p=i ; p<=j ; p++){ // K iterations
            if(arr[p]<0) break;
        }
        ans[i] = arr[p];
    }
    i++;
    j++;
}
```

T.C. = O(n*k) ✗

T.C. = O(n) ✓

Ques : Minimum size subarray sum [Leetcode - 209]

↳ I have to find the window size

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

i

sum = 0 2 5 6 8 10 8 4 6 10

minlen = 7 4

→ 2x = { 1, 2, 3, 4, 5 } target = 121

$n + n-1 + n-2 \cdots 1 = \frac{n(n+1)}{2} \Rightarrow T.C. = O(n^2)$

nums = { 1, 2, 4, 6, 3, 4, 3 } target = 10

i j

Sum = 0 1 3 7 13 12 10 6 9 13 7
107

len = 0 1 3 2 3 3

minLen = ~~INT_MAX~~ 1 3 2

```
while (j<n){
    Sum += arr[j];
    while (sum >= target) {
        len = j-i+1;
        minLen = min(minLen, len);
        Sum -= arr[i];
        i++;
    }
    j++;
}
```

- T.C. = O(n)

Atmo → $\leq 2^{*}n$

Ques : Max consecutive ones III [Leetcode - 1004]

nums = { 1, 0, 1, 1, 0, 0 ✓, 1, 1, 1, 1, 0 ✓ }
i j

K = 2 flips = 0 1 2 1 2

1) If nums[j] == 1 → j++

2) if (nums[i] == 0)
↳ if (flips < K) flips++

3) else
len = j - i
maxLen = max(maxLen, len)

→ 'i' ko pehle 0 ke aage le aao i++ bas

maxLen = INT_MIN ~~& &~~

len = ~~& & &~~ 6

Ques : Max consecutive ones III [Leetcode - 1004]

nums = { 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1 } k = 3

i

j

1) nums[j] == 1 → j++

2) nums[j] == 0

↳ a) flips < k → flips++, j++

b) flips == k

- len nikal lo
- i ko flipped zeroes me se pehle wale ke jo aage

flips = 0 1 2 3

maxlen = ~~1~~ ~~2~~ 3 ~~4~~ ~~5~~ 10

len = ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ 8

ray of 1's after deleting one

[Leetcode - 1493]

```
int zeroPos = -1;
int i = 0;
int j = 0;
int maxLen = 0;
int count = 0;
while(j<n){
    int prev = zeroPos;
    if(nums[j]==0){
        count++;
        zeroPos = j;
    }
    if(count<=1) j++;
    else{
        maxLen = Math.max(maxLen,j-i);
        i = prev + 1;
        count--;
        j++;
    }
}
maxLen = Math.max(maxLen,j-i);
return maxLen-1;
```

nums = { 0, 1, 1, 1, 0, 1, 1, 0, 1 }

i zp j

p

maxLen = 0 ✗ 6

count = 0 ✗ 2 ✗ 2 1

prev = ✗ 1 0

zeroPos = -✗ 0

Ques : Subarray Product Less than K [Leetcode - 713]

no. of subarrays nikalne hai

nums = { 10, 5, 2, 6 }

i j

K = 100

count = 0 2 5

product = 1 10 50 ~~100~~ 10 60

```
pro *= nums[j]
while (pro <= K){
    count += (j-i)
    pro /= nums[i];
    i++;
}
j++;
```

```
if(k<=1) return 0;
int n = nums.size();
int i = 0;
int j = 0;
int count = 0;
int product = 1;
while(j<n){
    product *= nums[j];
    while(product>=k){
        count += (j-i);
        product /= nums[i];
        i++;
    }
    j++;
}
```

{10, 5, 19, 4} K=100

i j

count = 0 1 2

pro = 1 10 50 950 95 380 76

while (i<n){

Thank you!