Question: Find sum of a subarray

```
int sumSubarray(int A[], int s, int e){
    int sum = 0;
    for(i = s; i <= e; i++){
        sum += A[i];
    }
    return sum;
}
```

```
int sumSubarray(int A[], int s, int e){
    if(s == 0) return PS[e];
    return PS[e] - PS[s-1]
}
```

Question: Print sum of all subarrays

A = 3 4 -2 (indices 0 1 2) N = 3 $\frac{3(4)}{2} = 6$

s	e		Sum
0	0	[3]	3
0	1	[3, 4]	7
0	2	[3, 4, -2]	5
1	1	[4]	4
1	2	[4, -2]	2
2	2	[-2]	-2

Brute Force

Iterate over all subarrays

```
for(start = 0; start < N; start++) {
    for(end = start; end < N; end++) {
        => // (start, end) represent a subarray
O(N) <-------- print( sumSubarray(A, start, end) );
    }
}
```

T.C: $N^2 \times O(N) = O(N^3)$

S.C: $O(1)$

Approach2: Prefix Sum

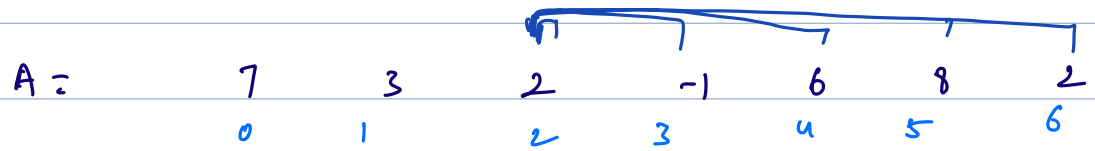
```
// Construct Prefix Sums Array -> O(N)
for(start = 0; start < N; start++) {
    for(end = start; end < N; end++) {
O(N^2)  => // (start, end) represent a subarray
        if( start == 0) {
            print( PS[end] );
        }
        else {
            print( PS[end] - PS[start-1] );
        }
    }
}
```

T.C : $O(N) + O(N^2) = O(N^2)$

S.C : $O(N) \Rightarrow$ p.s

Approach3: Carry forward

Smaller Question: Print sum of all subarrays starting at index



s	e	sum
2	2	A[2]
2	3	A[2] + A[3]
2	4	A[2] + A[3] + A[4] = 7
2	5	7 + 8 = 15
2	6	15 + 2 = 17

```
int sum = 0;
for(int end = 2; end < N; end++){
    sum = sum + A[end];
    print(sum);
}
```

0, 1, 2, 3 .... N-1

```
for [ start = 0;   start < N;   start ++ ) {
    int sum = 0;
    for [ end = start; end < N; end++ ) {
        sum = sum + A[end];
        print(sum);
    }
}
```

T.C: $O(N^2)$

S.C: $O(1)$

Dry Run

A = | 1 | 2 | 3 |
| --- | --- | --- |
| 0 | 1 | 2 |

Start	end	sum (0)	
0	0	1	
0	1	3	
0	2	6	sum = 0
1	1	2	
1	2	5	sum = 0
2	2	3	

Question: Find sum of all subarrays

Approach 1:

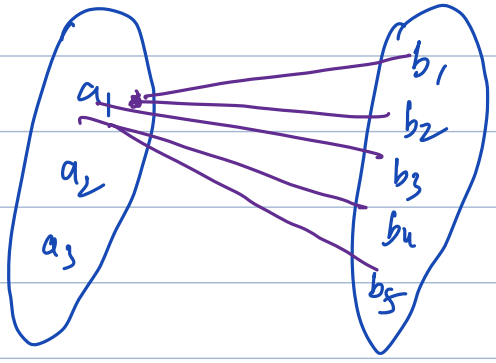
```
int totalSum = 0;
for (start = 0; start < N; start++) {
    int sum = 0;
    for (end = start; end < N; end++) {
        sum = sum + A[end];
        totalSum = totalSum + sum;
    }
}
return totalSum
```

T.C: $O(N^2)$ $N \leq 10^6$

S.C: $O(1)$

Question: Bag problem

If there are 2 bags with 3 and 5 items in each bag, find no. of ways we can select one item from Bag1 and other from Bag2



a_1 = b_1 to b_5

a_2 = b_1 to b_5

a_3 = b_1 to b_5

3 x 5 = 15 ways

Approach2 :

N = 3

S	e	Sum
0	0	A[0]
0	1	A[0] + A[1]
0	2	A[0] + A[1] + A[2]
1	1	A[1]
1	2	A[1] + A[2]
2	2	A[2]

3 x A[0] + 4 x A[1] + 3 x A[2]

3 → #subarrays

4 → #subarrays

3 → #subarrays

Task: For every element/index, find the #subarrays it occurs

A = [3, -2, 4, 1, 2, 6]

0 1 2 3 4 5

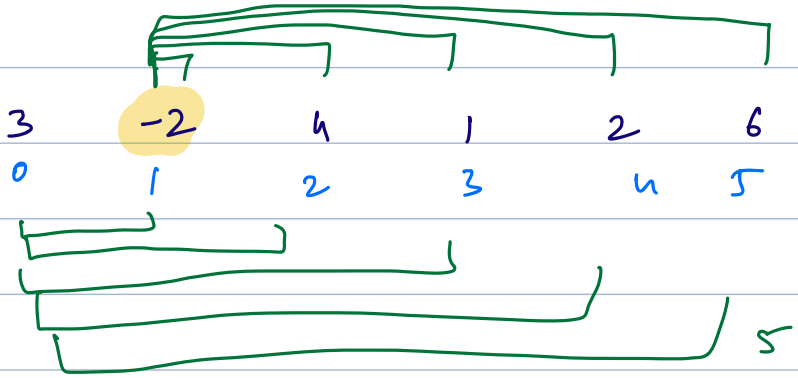
Ques: #subarrays in which index 0 occurs

6

Quiz

A = 3 -2 4 1 2 6

0 1 2 3 4 5



i = 1

= 10 subarrays

Question: # subarrays in which index 3 is part

A = 3 -2 4 1 2 6 8 11

0 1 2 3 4 5 6 7

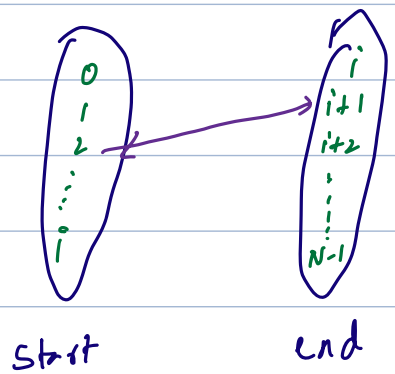
Can this element be part of any subarray which starts after index 3 ? NO

Start : [0, 3] ⇒ [0, i]

Will this element (index 3) be part of all subarrays which start before index 3 ? NO

Can this element be part of any subarray which ends before index 3 ? NO

end : [3 , N-1] [i, N-1]



[0, i] = i+1

[a, b] = b - a + 1

N-1 - i + 1 = N - i

[i, N-1]

N-1 - i + 1 = N - i

#subarrays : (i+1) * (N-i)

```
long sum = 0;
for (i = 0; i < N; i++) {
    long freq = (long)(i+1) * (N-1);
    sum = sum + A[i] * freq;
}
return sum;
```

T.C :	O(N)
S.C :	O(1)

Contribution Technique

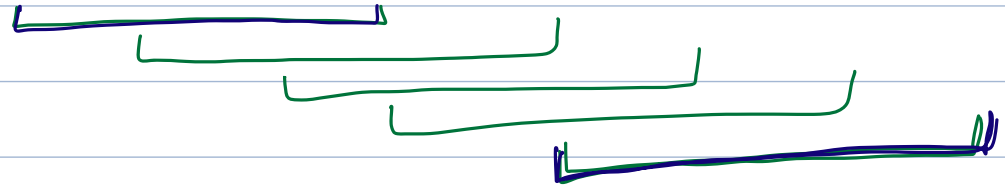
8:28

Question: Count no. of subarrays of length K

N = 8

A = 1 2 3 4 5 6 7 8

0 1 2 3 4 5 6 7



K = 4

S	C
0	3
1	4
2	5
3	6
4	7

5 subarrays

S	C
0	K-1
1	K
2	K+1
⋮	⋮
x (N-K)	N-1

[0, N-K] : N-K+1 subarrays

$[x, N-1] \Rightarrow K$

$N-1-x+1 = K$

$x = N-K$

$\overset{N-K}{\uparrow}$

$[x, N-1] =$

$N-1-x+1 = K$

$N-x = K$

$x = N-K$

1st subarray : [0, K-1]

last subarray : [N-K, N-1]

#subarrays :- N-K+1

Start indices : [0, N-K] .

Question:- Find maximum subarray sum for subarrays of length K

A :	-3	4	-2	5	3	-2	8	2	-1	4
	0	1	2	3	4	5	6	7	8	9

K = 5

s	e	sum
0	4	7
1	5	8
2	6	12
3	7	16
4	8	10
5	9	11

s+1, e+1

Brute Force

Consider all subarrays of size K, find their sum and compute max across all

```
function maxSubarrayOfLengthK(A[],N,K)
{
     ans = -infinity

     //first window
     i = 0
     j = k-1

     while(j < N)
     {
        sum = 0
        for(idx -> i to j)
        {
           sum += A[idx]
        }
        ans = max(sum, ans)

        //going to next subarray of length k
        i++
        j++
     }
     print(ans)

}
```

N-K+1 iterations; k

T.C: $(N-K+1) \times K$

$(N-K+1) \times K$

$K=1$: $(N-1+1) \times 1 = N$

$K=N$: $(N-N+1) \times N = N$

$K=\frac{N}{2}$: $\left(N-\frac{N}{2}+1\right) \times \frac{N}{2}$

TC $\left(\frac{N}{2}+1\right) \times \frac{N}{2} = O(N^2)$

S.C: $O(1)$

Approach2 : Prefix Sum

```
function maxSubarrayOfLengthK(A[],N,K)
{
     ans = -infinity
     //construct PS
     //first window
     i = 0
     j = k-1

     while(j < N)
     {
        sum = 0
        for(idx -> i to j)
        {
          sum += A[idx]
        }
        ans = max(sum, ans)

        //going to next subarray of length k
        i++
        j++
     }
     print(ans)

}
```

if(i == 0) sum = PS[j]

else sum = PS[j] - PS[i-1] ;

T.C : N-K+1 ⇓ K=1 ⇒ O(N)

S.C : O(N) ⇒ Prefix Sum

Approaches:

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

K = 6

sum = 5

5 - (-3) + 8 = 16

16 - 4 + 2 = 14

14 - (-2) + 1 = 17 => Ans

17 - 5 + 4 = 16

$\Sigma[0,5] = 5$

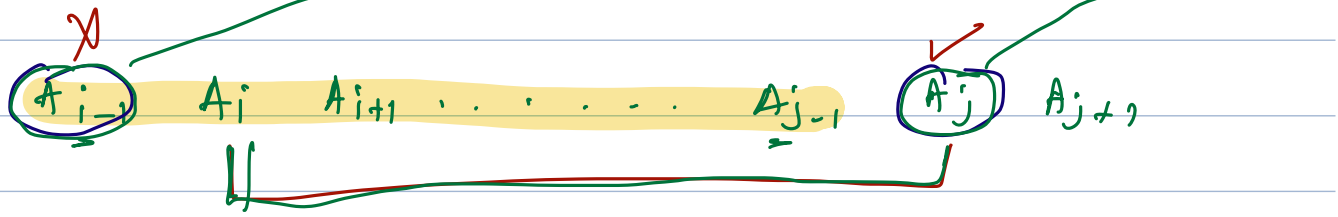
$\Sigma[1,6] = 5 - A[0] + A[5] = 16$

$\Sigma[2,7] = 16 - A[1] + A[7] = 14$

$\Sigma[3,8] = 14 - A[2] + A[8]$

$\Sigma[4,9] = 17 - A[3] + A[9] = 17 - 5 + 4$

$sum[i,j] = sum[i-1, j-1] - A[i-1] + A[j]$



Steps

1) Find sum of 1st subarray [0, k-1]
2) Iterate to remove & add an element to get sum of next subarray

```
function maxSubarrayOfLengthK(A[],N,K)
{
     ans = -infinity
     i = 0
     j = K-1

     sum = 0          // here k iterations
     for(idx -> i to j)
     {
        sum += A[idx]
     }
     ans = max(sum,ans)

     j++
     i++

     while(j<N)
     {
        sum = sum + A[j] - A[i-1]

        ans = max(sum,ans)
                                   // here N-k iterations
        i++
        j++
     }
     print(ans)
}
```

K → } ⇒ Find sum of 1st subarray of size K

N-K

T.C: K + (N-K) = N ⇒ O(N)

S.C: O(1)

Sliding window Technique

A = 1 2 3 4 5

- [1]
- [1,2]
- [1,2,3]
- [1,2,3,4]
- [1,2,3,4,5]

[5]
[4,5]
[3,4,5]
[2,3,4,5]
[1,2,3,4,5]

SF[i]: sum of elements from index i to N-1

	0	1	2	3	4
A =	1	2	3	4	5
SF =	15	14	12	9	5

decreasing