Q Given an array of size N and Q queries of the format s & e.

start index ⇑ s

⇑ e end index

Return the sum of elements from index s to e. (including both)

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

Q: 4

s	e	
1	3	12
2	7	12
4	8	9
0	2	5

Basic: for every query (q) ⇓ loop from s to e

```
for (i=1; i<=q; i++)
{
    // s & e
    for (j=s; j<=e)
    {
    }
}
```

O(q*N)

scores after every over from 41 → 50

41	42	43	44	45	46	47	48	49	50
288	312	330	349	360	383	394	406	436	439

406 after finish of 48th

cumulative data

Last 5 overs? (46-50)

439 - 360 = 79

score[50] - score[45]

[42 to 45] = 360 − 288

including both = 72

beginning

prefix sum at i → sum of all elements from 0-i

pf[i] = Σ 0-i

	0	1	2	3	4	5	6	7	8	9
	-3	6	2	4	5	2	8	-9	3	1
pfsum[i]:	-3	3	5	9	14	16	24	15	18	19

pf[5] = A[0] + A[1] + A[2] + A[3] + A[4] + A[5]

pf[6] = A[0] + A[1] + A[2] + A[3] + A[4] + A[5] + A[6]

pf[6] = pf[5] + A[6]

pf[i] =	pf[i-1]	+ arr[i]
0-i	0-(i-1)	

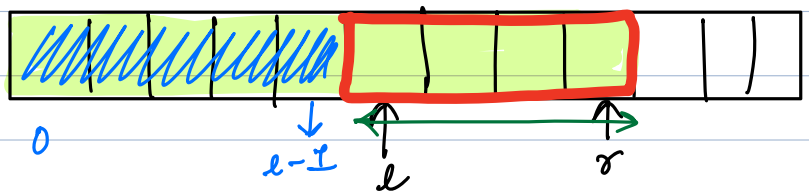
pf[0] = arr[0]

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

```
pf[0] = arr[0];
for(i=1; i<n; i++)
    pf[i] = pf[i-1] + arr[i];
```

l → r

pf sum



$$sum(l \text{ to } r) = pf[r] - pf[l-1]$$

l = 0 ?

↓

if (l == 0) pf[r]

	0	1	2	3	4	5	6	7	8	9
A:	-3	6	2	4	5	2	8	-9	3	1
Q: 4	-3	3	5	9	14	16	24	15	18	19

s	e
1	3
2	7
4	8
0	2

→ pf[3] - pf[0] = 9 - (-3) = 12 — O(1)

pf[7] - pf[1] = 15 - 3 = 12 — O(1)

pf[8] - pf[3] = 18 - 9 = 9 — O(1)

pf[2] = 5 — O(1)

build pf sum

T.C: O(q + N) → answer q queries

S.C: O(N) → reduce by using the same array

subarray

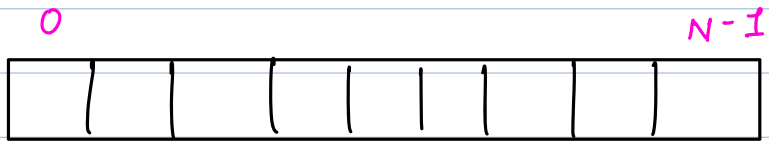
range sum

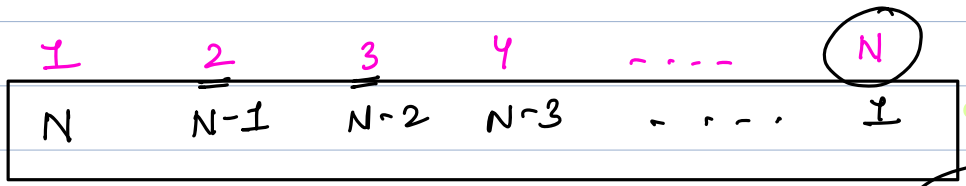
pf sum

subarray ? contiguous part of an array

1 or N

→ Total subarrays in an array of size N = $n(n+1)/2$





$n(n+1)/2$

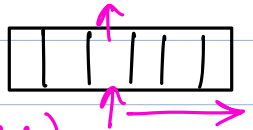
start & end

Q = array of size N. Find the sum of each subarrays.

0	1	2	3
9	-1	2	3

0,0	0,1	0,2	0,3
9	8	10	13
1,1	1,2	1,3	
-1	1	4	
	2,2	2,3	
	2	5	
		3,3	
		3	

Basic :- Go to each subarray & calculate sum




```
for( i=0 ; i<n ; i++)
{
    for( j=i ; j<n ; j++)
    {
        // i-j
        sum=0;
        for( k=i ; k<=j ; k++)
            sum+=arr[k];
        print(sum)
    }
}
```

T.C: $O(n^3)$

// pfsum array

n^3 — 3 loops

2 loop → n^2

$n^2 \log_2 n$ — sort

```
for( i=0→n-1)
    for(j=i→n-1)
    {
        // i-j = ___
        if(i==0) sum = pf[j];
        sum = pf[j] - pf[i-1];
    }
}
```

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

S.C: $O(N)$

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

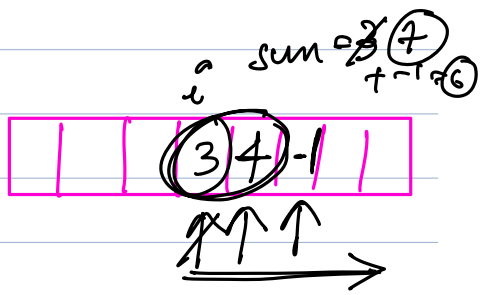
subarr of each sum — start j from 2

[2,2] = arr[2]

sum = [2,3] = arr[2] + arr[3]

[2,4] = arr[2] + arr[3] + arr[4]

sum




```
for( i=0; i<n; i++)
{
    sum = 0;
    for(j=i; j<n; j++)
    {
        sum += arr[j];
        print(sum);
    }
}
```

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

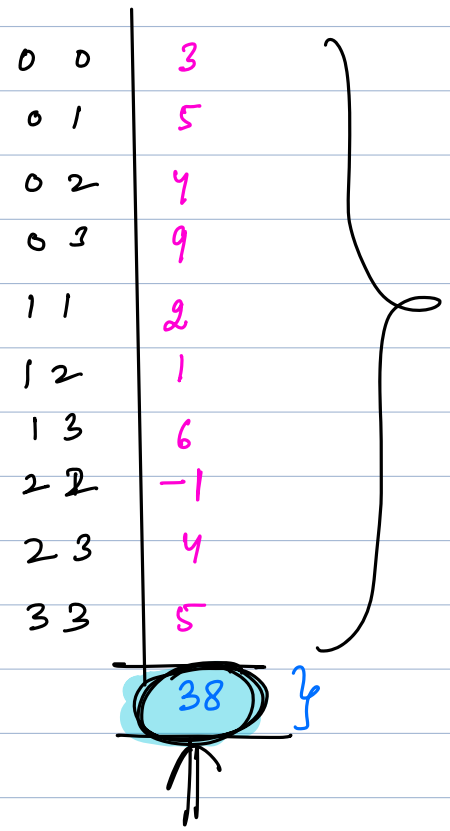
S.C: $O(1)$

$\propto n^2$

$\propto n^2$ sums

Q array of size N. Find total sum of all subarrays sum.

0	1	2	3
3	2	-1	5



Basic :- $O(n^3)$ ⇓ pf sum $O(n^2)$ S.C: $O(N)$ ⇓ $O(n^2)$ ⇓ $O(n)$

0	1	2
-1	3	4

A:

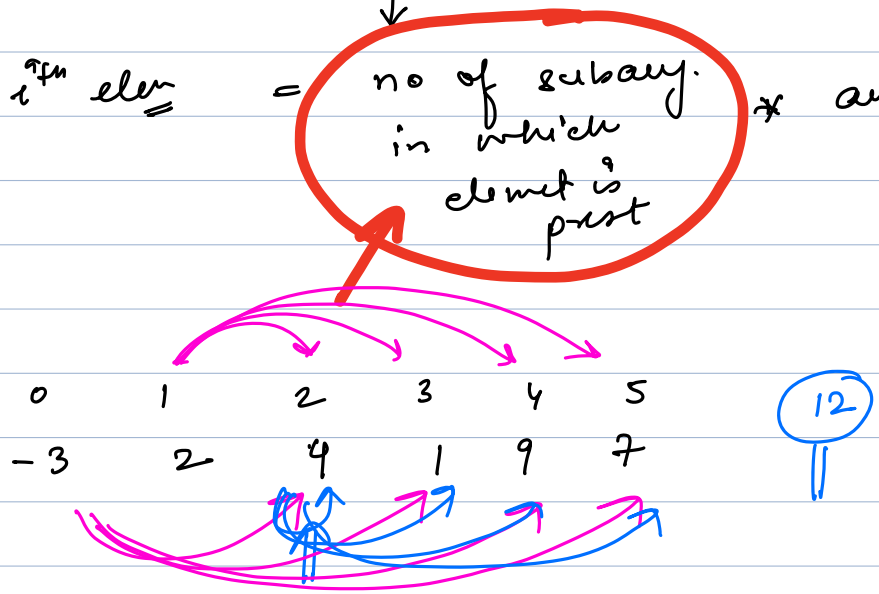
0,0	-1	arr[0]
0,1	2	+ arr[0] + arr[1]
0,2	6	+ arr[0] + arr[1] + arr[2]
1,1	3	+ arr[1]
1,2	7	+ arr[1] + arr[2]
2,2	4	+ arr[2]

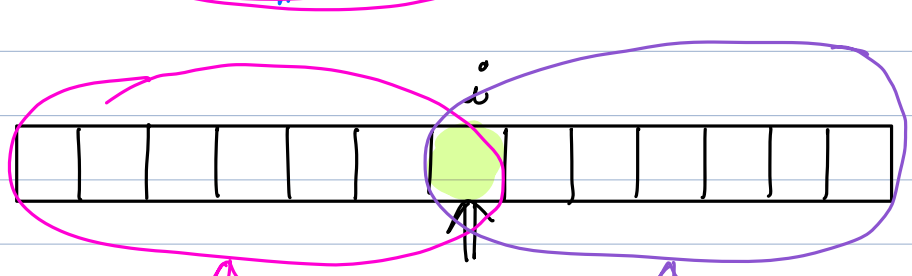
21 = (3 * arr[0] + 4 * arr[1] + 3 * arr[2]) / (-3 + 12 + 3 * 4)

= 21

How to find contribution of each element ?

cont i^{th} elem = no of subarr. in which element is present * arr[i]





a — b

b-a+1

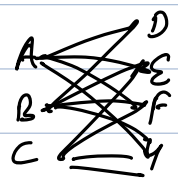
3 — 7

7-3+1 = 5

start pt → 0 - i → i+1

end pt → i → n-1 → (n-1) - (i) + 1 → n-i

Total no of subarr for i^{th} = $(i+1) * (n-i)$ — O(1)



$$ans = \sum_{i=0}^{n} (i+1) * (n-i) * arr[i]$$

contribution technique

T.C : O(n)

Q Find the max subarray sum of subarrays of size k

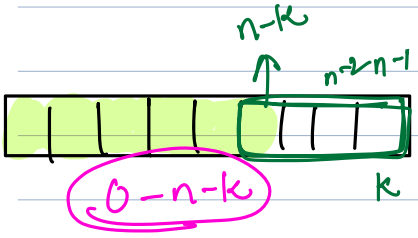
0	1	2	3	4	5	6	7	8	9
-3	4	-2	5	3	-2	8	2	-1	4

N = 10

k = 5

7

6



			size
0	4	7	
1	5	8	
2	6	12	
3	7	16	
4	8	10	
5	9	11	

subarr of size k?

N - k + 1

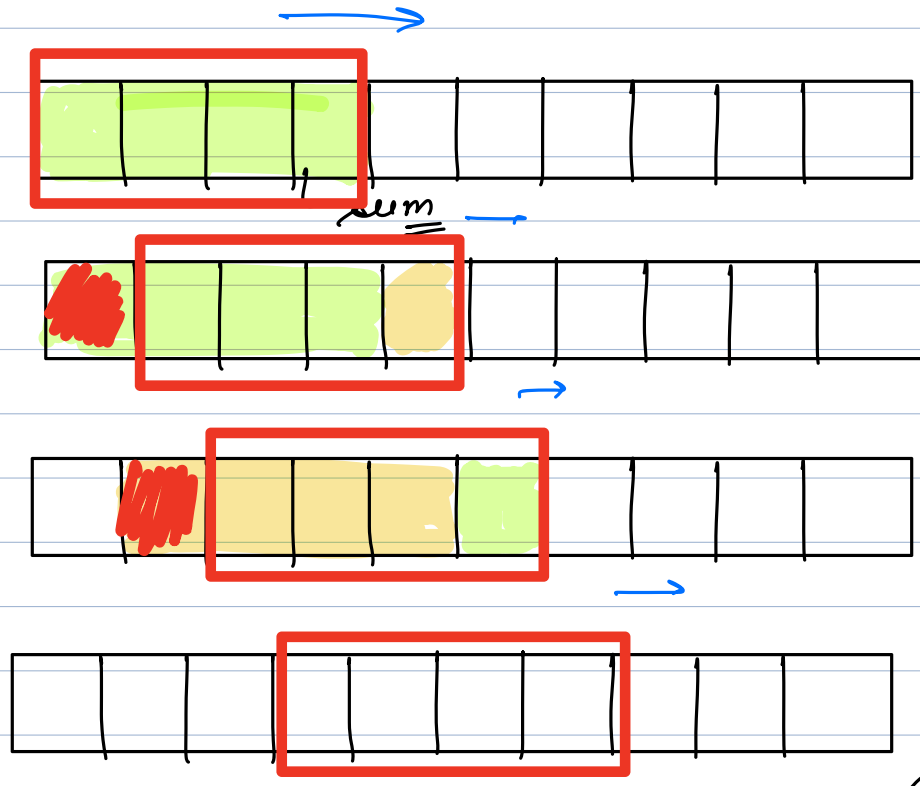
```
{ s = 0
  e = k-1
  while (e < n)
  {
      for (i=s; i<=e; i++)
          sum += arr[i];
      ans = max(ans, sum);
      s++; e++;
  }
```

$O((n-k+1) * k)$

k = 1 → O(n)

k = n/2 → $(n - n/2 + 1) * n/2 = n^2/4$ → $O(n^2)$

k = n → O(n)



subarray size is fixed
↓
sliding window

sum — outgoing + incoming

k=4

0 — 3 → traverse = sum

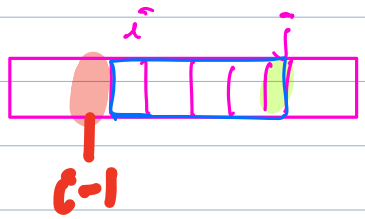
1 — 4 → sum + arr[4] − arr[0] = sum

2 — 5 → sum + arr[5] − arr[1] = sum

i-1 → j-1

i → j

sum + arr[j] − arr[i-1]



Sliding window

```
// build first window
for(i=0; i<k; i++)
{
    sum += arr[i];
}
ans = max(ans, sum);
s = 1;
e = k;
while(e < n)
{
    sum = sum - arr[s-1] + arr[e];
    s++; e++;
    ans = max(ans, sum);
}
```

0 - k-1 → 1 → k

s - e

n^2 → $n \log n$ → sort + BS

prefix sum → subarr sum

$O(n^3) \rightarrow O(n^2)$

contribution

sliding window

Arrays → BIT-1 → Rec → Hashmaps