Contribution Technique

Given an array of size N, Find the sum of all the subarrays.

① $N \leq 10^3$

② $N \leq 10^5$

A : [1 3 2]

1 → 1
3 → 3
2 → 2
1 3 → 4
3 2 → 5
1 3 2 → 6
21

Approach 1:

```
int ans = 0
for (L=0; L<N; L++){
   for (R=L; R<N; R++){
      sum=0;
      for (i=L; i<=R; i++){
         sum += arr[i];
      }
      ans += sum;
   }
}
```

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

* Not valid for $N \leq 10^3$ or $N \leq 10^5$.

Approach 2:

L=0 R=0, R=1, R=2

[1 3 2]

sum = ~~(1+3)~~ 1+3+2

ans : 1 + (1+3) + (1+3+2)

```
int ans = 0;
for (L=0; L<N; L++){
   sum=0;
   for (R=L; R<N; R++){
      sum += arr[R];
      ans += sum;
   }
}
```

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

* valid for $N \leq 10^3$ but not $N \leq 10^5$

* An array has $\frac{N(N+1)}{2}$ no. of subarrays.

For $N \leq 10^5$, add column wise ⇒ check the no. of times i appears and add all.

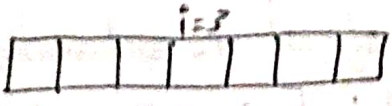
[1 3 2]

1			→ 1
	3		→ 3
		2	→ 2
1	3		→ 4
	3	2	→ 5
1	3	2	→ 6
↓	↓	↓	21

$3 \times 1 + 4 \times 3 + 3 \times 2 = 21$

i^{th} element

To find how many times A appear, use this formula :



i^{th} element

No. of times A appear = ~~(N+1) * (i~~ (i+1) × (N-i)

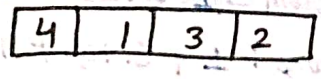
Approach 3 :

```
int ans = 0;
for (int i=0; i<N; i++) {
    ans += (arr[i] * (i+1) * (N-i));
}
```

T.C : O(n)

Contribution

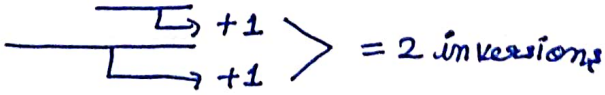
- Atomic Item contribution
- Extending End contribution
- Every [start / End] contribution

Q. 

Find the sum of no. of (#) inversion of each subarray ? Constraint : $N \leq 10^3$

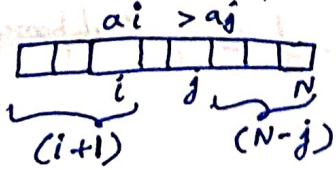
What is inversion ?
When (i, j) exist such that $i < j$ and $arr[i] > arr[j]$.

For exa: 1 3 2



= 2 inversions

* Kitne subarrays me voh particular inversion occur hoga.



Total subarrays contains (i, j) pair
= (i+1) * (N - j) [If $a_i > a_j$]

Approach :

```
void solve() {
    int n;
    cin >> n;
    int arr[n];
    for (int i=0; i<n; i++) {
        cin >> arr[i];
    }
```

```
int ans = 0;
for (int i = 0; i < n; i++) {
  for (int j = i+1; j < n; j++) {
    if (arr[i] > arr[j]) {
      ans += (i+1) * (n-j);
    }
  }
}
cout << ans << endl;
}
```

Ex: Q: For every subarray find its product and sum them.

a b c d

a ab abc → (c+bc+abc) x d = cd+bcd+abcd+d

b bc

c

a + b+ab + c+bc+abc

a] b] c] d]

d

a b c +cxd

+ +bxc +bxcxd

axb +axbxc +axbxcxd

Code :-

```
void solve() {
  int n;
  cin>>n;
  int arr[n];
  for (int i=0; i<n; i++) {
    cin >> arr[i];
  }
  int ans = 0;
  int lastans = 0;
  for (int i=0; i<n; i++) {
    lastans += lastans * arr[i] + arr[i];
    ans += lastans;
  }
  cout << ans << endl;
}
```

Q. Max. Subarray sum

Code:-

```
int ans = 0;
int lastans = 0;
for (int i=0; i<n; i++){
  lastans = max (lastans + arr[i], arr[i]);
  ans = max (ans, lastans);
}
cout << ans << endl;
```