Subarray Content

→ Subarray Basics
→ Printing Subarray
→ Generating All Subarrays
→ Printing all subarray sums.

- Approach 1 • Approach 2
- Max Subarray Sum
- Sum of all subarray sums

Doubt session

Todo → Arrays class

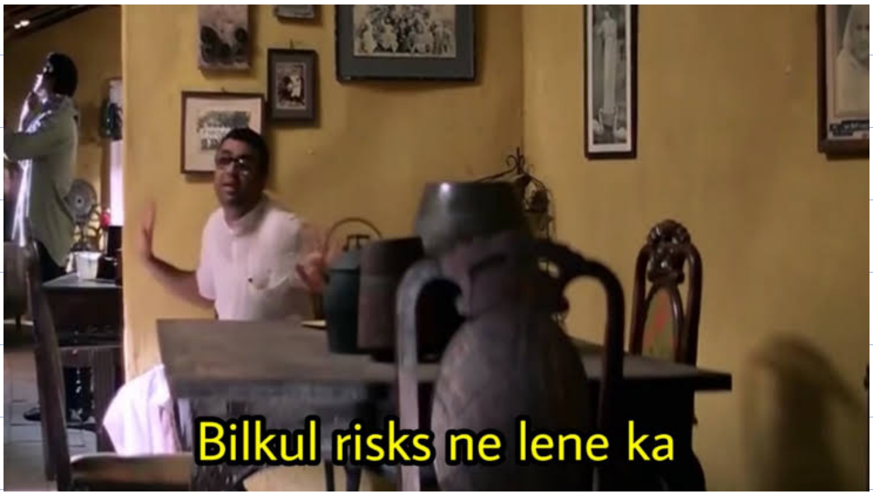
Assignment → Bulbs question.

Homework → Even Subarrays.

Due to connectivity issues in last class:-

Jio fibre
Airtel xtreme

me →



Subarray Basics :-

- Continuous part of an array.
- Single element is an subarray
- Complete array is subarray.

Ex :-

	0	1	2	3	4	5	6	7	8	9
arr[10] =	-2	4	6	3	8	1	4	3	2	-10

indices : [3 4 5 7 8] ✗

indices : [4 5 6 7 8] ✓

indices : [2 6] ✗

Quiz 1, 2, 3

```
- Print Sub ( arr[], s, e) {
    // [subarray] [s e]
      for (i = s; i <= e; i++) {
          Print(arr[i]
      }
  }
```

T.C → O(N)

S.C → O(1)

// How many Subarrays.

Ex 1:- arr[4] =

$$\begin{array}{cccc} 0 & 1 & 2 & 3 \\ -1 & 3 & 2 & 3 \end{array}$$

0 - 0 : -1
0 - 1 : -1 3
0 - 2 : -1 3 2
0 - 3 : -1 3 2 3

1 - 1 : 3
1 - 2 : 3 2
1 - 3 : 3 2 3

2 - 2 : 2
2 - 3 : 2 3
3 - 3 : 3

$\Rightarrow$ 10 Subarrays

$$\frac{\cancel{4}^{2}(4+1)}{\cancel{2}} \Rightarrow \underline{10}$$

// Given N elements, How many subarrays?

arr[N] = { 0, 1, 2, 3, 4 ... i, i+1 ... N-2, N-1 }

Start → 0	Start - 1	Start - 2	Start N-2	Start N-1
0 - 0	1 - 1	2 - 2	(N-2) - (N-2)	(N-1) - (N-1)
0 - 1	1 - 2	2 - 3	(N-2) - (N-1)	
0 - 2	1 - 3	2 - 4		
0 - 3	⋮	⋮		
⋮	1 - N-1	2 - N-1		
0 - N-1				
n	n - 1	n - 2	2	1

Sum of subarrays starting at diff. indices :-

$$n + (n-1) + (n-2) + \dots 2 + 1$$

$$\Rightarrow \frac{n(n+1)}{2}$$

no. of subarrays in a given array

// Printing all subarrays

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

0	1	2	3
2	4	5	6

s=0 e → 0 → 3

s	e
0	0
0	1
0	2
0	3
1	1
1	2
1	3
2	2
2	3
3	3

T.C → $O(N^3)$

S.C → $O(1)$

2

2 4

2 4 5

2 4 5 6

4

4 5

4 5 6

5

5 6

Max Subarray Sum :-

$$\begin{array}{cccccc} & 0 & 1 & 2 & 3 & \\ arr[4] = & [8 & 2 & 9 & 10] & \end{array}$$

[0 - 0] = [8] → 8

[0 - 1] = [8 2] → 10

[0 - 2] = [8 2 9] → 19

[0 - 3] = [8 2 9 10] → 29

[1 - 1] = [2] → 2

[1 - 2] = [2 9] → 11

[1 - 3] = [2 9 10] → 21

[2 - 2] = [9] → 9

[2 - 3] = [9 10] → 19

[3 - 3] = [10] → 10

29

Max Subarray Sum.

// Print all subarray sums

$\to$ Integer.MIN_VALUE

int maxSum = $-\infty$.

```
for(s=0; s<n; s++){

    for(e=s; e<n; e++){
        int sum = 0
        for(i=s; i<=e; i++){
            sum = sum + arr[i]
        }
        print(sum)      if(sum > maxSum)
                        { __
                        }
    }
}
```

$T.C \to O(N^3)$

$S.C \to O(1)$

Prefix Sum $(s,e) \to$ if $(s==0)$ pf[e]

else pf[e] - pf[s-1]

$T.C \to O(N^2)$

$S.C \to O(N)$

Kadane's Algorithm $\to O(n)$

$\downarrow$

Advance first lecture

// using prefix :-

// create prefix :- pf[n].

```
int maxSum = -∞.
for (s=0; s<n; s++) {
    for (e=s; e<n; e++) {
        int sum = 0

        if (s==0) {
            sum = pf[e]
        } else { sum = pf[e]-pf[s-1]
        }
    }
}
```

// Printing all subarrays sum starting at index 2.

Ex:- arr[7]:

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

[2-2] → 2

[2-3] → 1

[2-4] → 7

[2-5] → 15

[2-6] → 17

[2-7] → 20

```
sum = 0;
for(j=2; j<n; j++) {
    sum = sum + arr[j]
    print (sum)
}
```

sum = $\cancel{0}$ $\cancel{2}$ $\cancel{1}$ $\cancel{7}$ $\cancel{15}$ $\cancel{17}$ 20

j = $\cancel{2}$ $\cancel{3}$ $\cancel{4}$ $\cancel{5}$ $\cancel{6}$ $\cancel{7}$ 8

print

2
1
7
15
17
20

Print all subarray sums starting at index i.

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

// Printing all subarray sums using carry forward?

2 4 6 7

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

T.C → $O(n^2)$

S.C → $O(1)$

Time → 10:16 pm

10:22 pm

// Print max subarray sums.

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

Advance → Kadane's Algorithm

T.C → $O(n)$, S.C → $O(1)$

Sum of Subarray Sums :-

$$arr[4] = [8 \quad 2 \quad 9 \quad 10]$$

[0 - 0]	= [8]	→	8
[0 - 1]	= [8 2]	→	10
[0 - 2]	= [8 2 9]	→	19
[0 - 3]	= [8 2 9 10]	→	29
[1 - 1]	= [2]	→	2
[1 - 2]	= [2 9]	→	11
[1 - 3]	= [2 9 10]	→	21
[2 - 2]	= [9]	→	9
[2 - 3]	= [9 10]	→	19
[3 - 3]	= [10]	→	10
	138		138

$(i+1) * (n-i)$

$(0+1) * (4-0) \Rightarrow 4$

4 6 6 4

// Print sum of all subarray sums.

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

T.C → $O(n^2)$, S.C → $O(1)$.

Ques) In how many subarray index 2 is present?

	0	1	2	3	4	5
arr[] =	2	-2	4	-1	2	6

s	e
0	3
1	4
2	5
3	

⇒ 12

Ques) In how many subarrays index 1 is present?

	0	1	2	3	4	5
arr[] =	3	-2	4	-1	2	6

s	e
0	1
1	2
	3
	4
	5

$\Rightarrow$ $\underline{10}$

Ques) In how many subarrays index 0 is present?

	0	1	2	3	4	5
arr[] =	3	-2	4	-1	2	6

s	e
0	0
	1
	2
	3
	4
	5

$\Rightarrow 1 \times 6 \Rightarrow \underline{6}$

Generalize

0, 1, 2, ... i, ... n-1

s	e
0	i
1	i+1
2	:
:	:
i	n-1
(i+1)	n-i

Total subarrays in which ith index would be present → $(i+1) * (n-i)$

```
// Sum of all subarray sums:-
   sum = 0
for (i=0; i<n; i++) {
     // freq of i
     freq = (i+1) * (n-i)
        sum = sum + freq * arr[i]
```

}
}

return sum

T.C → O(N)
S.C → O(1).