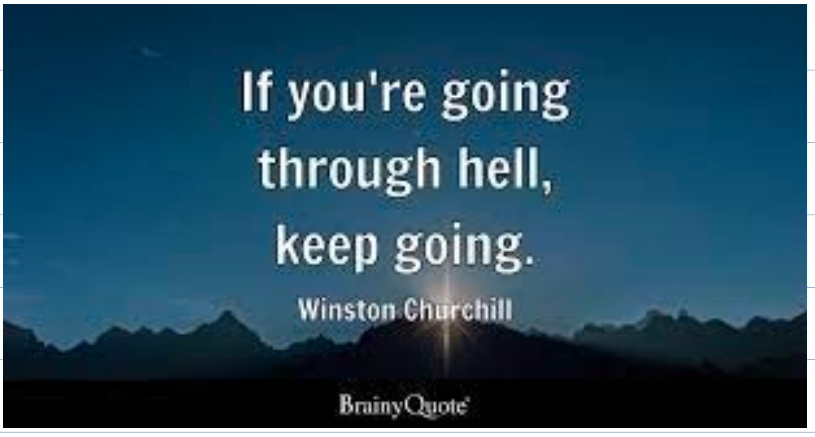
Subarray Content

→ 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

Assignment → Bulbs question.
Homework → Even Subarrays.



Subarray Basics :-

- → Continuous part of an array.
- → Single element is a 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} ✗

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

T.C → O(N)

S.C → O(1)

// How many Subarrays.

Ex1:- $arr[4] = \begin{array}{cccc} 0 & 1 & 2 & 3 \\ -1 & 3 & 2 & 3 \end{array} \rightarrow \frac{\cancel{(4)}^{2}\,(4+1)}{\cancel{2}} \Rightarrow \underline{10}$

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

=> 10 Subarrays.

// 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 at N-2	Start at 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	⋮	⋮		
⋮				
0 - N-1	1 - N-1	2 - N-1		
N	N - 1	N - 2	2	1

$$N + N-1 + N-2 + \dots 2 + 1 \Rightarrow \boxed{\frac{n(n+1)}{2}}$$

no. of subarrays in a given arr

// Printing all Subarrays

0	1	2	3
2	4	5	6

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

3

3

3

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

0 to 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

6

Max Subarray Sum :-

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

[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

// Print all subarray sums

```
int maxSum = -∞   --> Integer.MIN_VALUE
for (s=0; s<n; s++) {
    for (e=s; e<n; e++) {
        int sum=0
        for (i=s; i<=e; i++) {
            sum += arr[i];
        }
        print(sum)
    }
}
```

```
if (sum > maxSum)
{ maxSum = sum
}
```

T.C → $O(N^3)$
S.C → $O(1)$

prefix sum (s,e) → if (s==0) pf[e]
→ else pf[e] - pf[s-1]

T.C → $O(N^2)$
S.C → $O(N)$

Kadane's Algorithm :- T.C → $O(N)$
S.C → $O(1)$.

↓

Advance first lecture.

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

sum = 8, 10, 19, 29, 2, 11, 21, 9, 19, 10

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

T.C → $O(n^2)$

S.C → $O(1)$

// Printing all subarrays sum starting at index 2.

	0	1	2	3	4	5	6	7
Ex:- arr[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)
}
```

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.

```
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 -> $O(1)$.

Break 10:00 pm - 10:10 pm.

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

Sum of Subarray sums :-

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

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

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

    }
}
```

T.C → O(n^2)

S.C → O(1)

Ques) In how many subarray index 3 is present?

$$
\begin{array}{lcccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 \\
arr = & 3 & -2 & 4 & -1 & 2 & 6
\end{array}
$$

s	e
0	3
1	4
2	5
3	

$\Rightarrow 4 * 3 = \underline{12}$

Possible subarrays.

$(i+1) * (n-i) \Rightarrow 4 * 3 \Rightarrow \underline{12}$

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

$\Rightarrow 2 * 5 = 10$

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

$\Rightarrow 1 \times 6 = 6$

Generalize

0, 1, 2, 3 . . . . i n-1

s	e
0	i
1	i+1
2	i+2
3	:
:	
i	n-1
i+1	n-i

$\Rightarrow (i+1)(n-i)$

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

// 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)

(*) Even Subarrays :-

A → {2, 4, 6, 9}   2, 4, 8, 12,
$\underline{6}$

→ One or more subarrays of even length.

First & Last of all subarrays should be even.

A = {0, 1, 2, 3, . . n-1}
↓
even.

adhoc
↑

→ n-1 should be even.

→ 0^{th} should be even.

→ arr. leng should be even.

-: Bulbs :-

0, 1, 0, 1

→ 1, 0, 1, 0

→ 1, 1, 0, 1

→ 1, 1, 1, 0 → 1 1 1 1

0, 0, 0, 0 → 1

0 1 0 1 1 → ans

Bulb = 1 ← on
0 ← off

0 0 1 1 0 0 1 0 1

0 1 0 1 0 1

O → not worst case

worst case → $O(n^2)$

$O(1)$

```
for (i=0; i<n; i++){
    if (arr[i]==k)
        return True
    }
}
```