Todays Content

a) Subarray Intro
b) Printing all Subarrays
c) Printing all Subarray Sums
d) Few More Subarray Questions

Subarray Concept Learnings

Continous part of an arr[] is considered as Subarray

Note1: Single ele is also considered as Subarray

Note2: Complete arr[] is also considered as Subarray

Note3: Subarray is considered from left → right

	0	1	2	3	4	5	6	7	8
arr[9] =	4	1	2	3	-1	6	9	8	12

Check if below are Subarrays:

2 3 -1 6 : Yes

4 2 : No

1 2 6 : No

-1 6 9 : Yes

8 : Yes

8 9 6 : No

4 1 2 3 -1 6 9 8 12 : Yes

2 3 6 : No

Given start index & end index print Subarray

	0	1	2	3	4	5	6	7	8
arr[9] =	4	1	2	3	-1	6	9	8	12

Ex:

s	e	output
2	4	2 3 -1
4	7	-1 6 9 8
1	6	1 2 3 -1 6 9

Idea: Iterate on subarray from [s...e]

```
void printSub(int arr[], int s, int e){
    for(int i=s; i<=e; i++){
        print(arr[i])
    }
}
```

TC: O(N) SC: O(1)

Note: Given start index & end index we can get Subarray

Given start index & length of it print Subarray

	0	1	2	3	4	5	6	7	8
ar[9] =	4	1	2	3	-1	6	9	8	12

Ex:

s	l	output	⟶ End Index
2	4	: 2 3 -1 6	5 = 2+4-1
4	3	: -1 6 9	6 = 4+3-1
1	6	: 1 2 3 -1 6 9	6 = 1+6-1

Qo: start s = 4 len = 3 : { 4, 5, 6 } e = 6 = 4+3-1 = 6

Qo.5: For a Subarray

Start index = s

Subarray length = l

End index e = s+l-1.

```
void printSub(int arr[], int s, int l) { TC: O(N) SC: O(1)
    int e = s+l-1;
    TODO


}
```

Q1: arr[7] =

0	1	2	3	4	5	6
4	2	10	3	12	-2	15

Count of Subarrays starting at index = 1? ans = 6

Subarrays:

s	e	
1	1	= {2}
1	2	= {2 10}
1	3	= {2 10 3}
1	4	= {2 10 3 12}
1	5	= {2 10 3 12 -2}
1	6	= {2 10 3 12 -2 15}

Q2: arr[7] =

0	1	2	3	4	5	6
4	2	10	3	12	-2	15

Count of Subarrays starting at index = 3?

Subarrays:

s	e	
3	3	→ {3}
3	4	→ {3 12}
3	5	→ {3 12 -2}
3	6	→ {3 12 -2 15}

Q3: Total Count of Subarrays ?

	0	1	2	3
ar[4] =	4	2	10	3

S = 0 ⇒ 4

s	e	
0	0	: [4]
0	1	: [4 2]
0	2	: [4 2 10]
0	3	: [4 2 10 3]

S = 1 ⇒ 3

s	e	
1	1	: [2]
1	2	: [2 10]
1	3	: [2 10 3]

S = 2 ⇒ 2

s	e	
2	2	: [10]
2	3	: [10 3]

S = 3 ⇒ 1

s	e	
3	3	: [3]

Total Subarrays = 4 + 3 + 2 + 1 = 10

Generalize ar[N] : {Not Needed}

$ar[N] = [a_0 \quad a_1 \quad a_2 \quad a_3 \quad \ldots \quad a_{n-2} \quad a_{n-1}]$

S = 0

s	e
0	0
0	1
0	2
0	3
⋮	⋮
0	N-1

#N Subarrays

S = 1

s	e
1	1
1	2
1	3
⋮	⋮
1	N-1

N-1 Subarrays

S = 2

s	e
2	2
2	3
2	4
⋮	⋮
2	N-1

#N-2 Subarrays

S = N-1

s	e
N-1	N-1

#1 Subarray

Total Subarrays = N + N-1 + N-2 1

= 1 + 2 + 3 + .. N-1 + N

Total Subarrays = $\frac{(N)(N+1)}{2}$ ≈ O(N²)

1Q) Given ar[N] print all Subarrays

Note: Print each subarray in new line

Constraints:

$1 <= N <= 10^2$

$1 <= ar[i] <= 10^5$

Ex: ar[4]:

0	1	2	3
2	8	-1	4

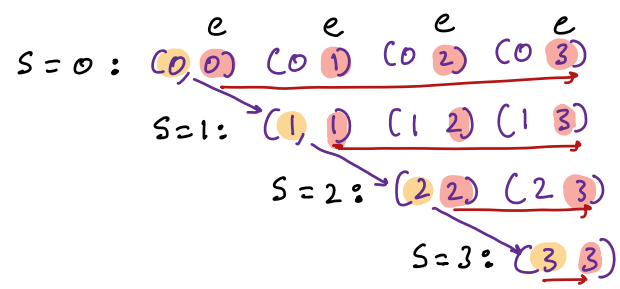
Subarrays

s	e	line	
0	0	: 1	{2}
0	1	: 2	{2 8}
0	2	: 3	{2 8 -1}
0	3	: 4	{2 8 -1 4}
1	1	: 5	{8}
1	2	: 6	{8 -1}
1	3	: 7	{8 -1 4}
2	2	: 8	{-1}
2	3	: 9	{-1 4}
3	3	: 10	{4}

Idea: Get start & End index of every subarray iterately print it.

ar[4]:

0	1	2	3
2	8	-1	4




```
void printSub(int ar[]){ TC: O(N^2) * O(N) = O(N^3)
                                SC: O(4) => O(1)
    int N = ar.length;
    for(int s=0; s<N; s++){ //start of subarray
        for(int e=s; e<N; e++){ //end of subarray
            // [s: start ind  e: end ind]
            for(int i=s; i<=e; i++){ // Print Subarray
                print(ar[i])
            }
            println();
        }
    }
}
```

28) Given ar[N] elements print each subarray sum

Note: Print each subarray sum in new line

Constraints:

$1 <= N <= 10^3$

$1 <= ar[i] <= 10^5$

Ex:

	0	1	2	3
ar[4] =	2	8	-1	4

Subarrays

s	e	line		output
0	0	: 1	{2}	→ 2
0	1	: 2	{2 8}	→ 10
0	2	: 3	{2 8 -1}	→ 9
0	3	: 4	{2 8 -1 4}	→ 13
1	1	: 5	{8}	→ 8
1	2	: 6	{8 -1}	→ 7
1	3	: 7	{8 -1 4}	→ 11
2	2	: 8	{-1}	→ -1
2	3	: 9	{-1 4}	→ 3
3	3	: 10	{4}	→ 4

Idea: Get start & end index of every subarray iterately calculate sum & print it.

```
void printSub(int ar[]){ TC: O(N^2)*O(N)=O(N^3)
                          SC: O(4) => O(1)
    int N=ar.length;
    for(int s=0; s<N; s++){ //start of subarray
        for(int e=s; e<N; e++){ //end of subarray
            // [s: start ind  e: end ind]
            // Calculate sum from [s...e]
            int sum=0
            for(int i=s; i<=e; i++){ // Cal sum
                sum=sum+ar[i];
            }
            println(sum);
        }
    }
}
```

Q10:

	0	1
ar[2] =	3	4

Subarrays

s	e		output
0	0	: 1 {3}	3
0	1	: 2 {3 4}	7
1	1	: 3 {4}	4

Idea: Calculate each subarray sum with pf[]

```
void printSum(int ar[]){   TC: O(N*N2) = O(N^2)   SC: O(N)
    int N = ar.length;
    long pSum[N];
    Psum[0] = ar[0]
    for(int i=1; i<N; i++){
        Psum[i] = Psum[i-1] + ar[i]
    }
    for(int s=0; s<N; s++){ //start of subarray
        for(int e=s; e<N; e++){ //end of subarray
            // Calculate sum from [s...e] using psum[]
            if(s==0){ // [0..e] = Psum[e]
                println(psum[e])
            }
            else
                println(psum[e] - psum[s-1])
            }
        }
    }
}
```

3Q) Given ar[N] print all Subarray sums starting at index 3

Eg: ar[10] :

0	1	2	3	4	5	6	7	8	9
3	8	4	7	9	4	3	2	10	6

S = 3

	sum
[3 3] :	7
[3 4] :	16
[3 5] :	20
[3 6] :	23
[3 7] :	25
[3 8] :	35
[3 9] :	41

[3 3] Sum = 0 + 7

[3 4] Sum = 7 + 9

[3 5] Sum = 16 + 4

[3 6] Sum = 20 + 3

[3 7] Sum = 23 + 2

[3 8] Sum = 25 + 10

[3 9] Sum = 35 + 6

Sum = 41

```
void PrintSums(int ar[]){
    int N = ar.length;
    int sum = 0;
    int s = 3
    for(int e = s; e < N; e++){
        sum = sum + ar[e]
        println(sum)
    }
}
```

Dry Run: ar[5] =

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

sum = 0

S = 3 :

e = 3 sum = sum + ar[3]
// sum = 8
print(sum)

e = 4 sum = sum + ar[4]
// sum = 12
print(sum)

e = 5 sum = sum + ar[5]
// sum = 15
print(sum)

// Subarray sums S = 1

```
void PrintSums(int ar[]){
    int N = ar.length;
    int sum = 0;
    int s = 1;
    for(int e = s; e < N; e++){
        sum = sum + ar[e]
        println(sum)
    }
}
```

Ideas: **Printing all Subarray Sums** TC: O(N²) SC: O(1)

```
void printSum(int ar[]){
    int N = ar.length;
    for(int s=0; s<N; s++){
        int sum=0;
        for(int e=s; e<N; e++){
            sum = sum+ar[e]
            //subarray sum [s..e]
            println(sum)
        }
    }
}
```

Dry Run: ar[ ] = { 3 2 -1 }

	0	1	2	Subarray
s				
0	sum=0 update sum			
	e=0 | sum=sum+ar[0] // sum=3			[0 0]
	e=1 | sum=sum+ar[1] // sum=5			[0 1]
	e=2 | sum=sum+ar[2] // sum=4			[0 2]
1	sum=0			
	e=1 | sum=sum+ar[1] // sum=2			[1 1]
	e=2 | sum=sum+ar[2] // sum=1			[1 2]
2	sum=0			
	e=2 | sum=sum+ar[2] // sum=-1			[2 2]

Questions Based on Above Ideas

a) Max Subarray Sums

```
int MaxSum(int ar[]){
    int N = ar.length;
    int ans = INT_MIN
    for(int s=0; s<N; s++){
        int sum=0;
        for(int e=s; e<N; e++){
            sum = sum+ar[e]
            //subarray sum [s..e]
            ans = Math.max(ans, sum)
        }
    }
    return ans;
}
```

b) No: of Subarrays Sums, which are even

```
int EvenSum(int ar[]){
    int N = ar.length;
    int c=0;
    for(int s=0; s<N; s++){
        int sum=0;
        for(int e=s; e<N; e++){
            sum = sum+ar[e]
            //subarraySum[s..e]
            if(sum%2==0){ c=c+1}
        }
    }
    return c;
}
```

c) No: of even length Subarrays, with sum >= B

```
void EvenlengthSum(int ar[], int B){
    int N = ar.length;
    int c=0;
    for(int s=0; s<N; s++){
        int sum=0;
        for(int e=s; e<N; e++){
            sum = sum+ar[e]
            //subarraySum[s..e]
            int l= e-s+1;
            if( l%2==0 && sum >= B){
                c=c+1
            }
        }
    }
    return c;
}
```