Lecture 7 :- Subarrays.

Agenda

- Print subarray from [s, e] ✓
- Print each and every subarray. ✓
- Print each subarray sum. ✓
- Return all subarray sums.

Class starts at 7:05 AM

No of subarrays of arr of len n :- $\frac{n(n+1)}{2}.$

Qu1 Given arr[n], print subarray from [s, e].

Ex. arr[5] = [$\overset{0}{2}$, $\overset{1}{8}$, $\overset{2}{6}$, $\overset{3}{4}$, $\overset{4}{3}$]

s = 2, e = 4.

Output :- 6 4 3.

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

Qu2 Given an arr[n], print each and every subarray.

Ex. arr[4] = [$\overset{0}{6}$, $\overset{1}{8}$, $\overset{2}{1}$, $\overset{3}{7}$]

```
6                         8                 1           7
6  8                      8  1              1  7
6  8  1                   8  1  7
6  8  1  7
```

Approach: Traverse through all subarrays and print them.

arr[4] = [$\overset{0}{6}$, $\overset{1}{8}$, $\overset{2}{1}$, $\overset{3}{7}$]

s = 0

6 [0, 0]

6 8 [0, 1]

6 8 1 [0, 2]

6 8 1 7 [0, 3]

s = 1

8 [1, 1]

8 1 [1, 2]

8 1 7 [1, 3]

s = 2

1 [2, 2]

1 7 [2, 3]

s = 3

7 [3, 3]

s [0, n-1]	e [s, n-1]
0	[0, 3]
1	[1, 3]
2	[2, 3]
3	[3, 3]

Traversal $O(n^2)$.

$O(n)$

```
void  printEachSubarray( int[] arr) {
                  s<= arr.length-1
    for( s=0; s< arr.length; s++) {
                          e<=arr.length-1
        for(e=s; e < arr.length; e++) {

            // subarray [s,e].

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

TC: $O(n^3)$

SC: $O(1)$.

Possible to optimise? NO

Print of array | subarray → can't optimised

Qu Given arr[n], print each and every subarray sum.

arr[4] = [6, 8, 1, 7] (indices 0, 1, 2, 3)

6 6

6 8 14

6 8 1 15

6 8 1 7 22

8 8

8 1 9

8 1 7 16

1 1

1 7 8

7 7

Approach1:

```
void printsumsubarray( int[] arr) {
              s<= arr.length-1
    for( s=0; s< arr.length; s++) {
                   e<=arr.length-1
        for(e=s; e < arr.length; e++) {

            // subarray [s,e]
            int sum=0;
            for( i=s; i<=e; i++) {
                sum = sum + arr[i];
            }
            print (sum);
        }
}
```

TC: O(n^3)

SC: O(1).

Approach 2   TC: $O(n^2)$.

$arr[4] = [\overset{0}{6}, \overset{1}{8}, \overset{2}{1}, \overset{3}{7}]$

s=0

6 [0,0]   6 sum(0,0)   8   8   1   1   7   7

6 8 [0,1]   14 sum(0,1)   8 1   9   1 7 8

6 8 1 [0,2]   15 sum(0,2)   8 1 7 16

6 8 1 7 22

$subarray [s,e] \xrightarrow{sum} sum(s,e).$

Hint: Prefix sum.

$arr[4] = [\overset{0}{6}, \overset{1}{8}, \overset{2}{1}, \overset{3}{7}]$

$ps[4] = [6 \quad 14 \quad 15 \quad 22]$

sum(1,3) =

```
sum=0
for( i=1; i<=3; i++){
    sum+= arr[i]
}
```

Approach 1.

$sum(\overset{s}{\overset{\uparrow}{1}},\overset{e}{\overset{\uparrow}{3}}) = pf[3] - pf[0]. = 22 - 6 = 16$

$sum[s,e] = pf[e] - pf[s-1]$

Edge case:- $s = 0$;

$sum(0,e) = pf[e] - pf[0-1]$ { Idx out of bound exception }

$sum(0,e) = pf[e].$

```
          void  printsumsubarray (int[] arr) {
SC:O(n) ← int[] pf = prefixsumoptimal(arr);  → TC: O(n)

                 s<= arr.length-1
     for( s=0; s< arr.length; s++) {   → O(n)
                   e<=arr.length-1
          for(e=s; e < arr.length; e++) {  → O(n).

               // subarray [s,e].
               if ( s== 0) {
                    print( pf[e]);
               } else {
                    print( pf[e] - pf[s-1]);
               }
          }
     }
```

TC: $O(n^2)$. $\simeq$ $O(n) + O(n*n) = O(n) + O(n^2)$.

SC: $O(n)$.

Approach3: TC: $O(n^2)$ SC: $O(1)$ (Get rid of prefix array)

carry forward

arr[4] = [$\overset{0}{6}$, $\overset{1}{8}$, $\overset{2}{1}$, $\overset{3}{7}$]

s = 0 sum = 0

6	[0,0]	sum = 6	print(6)
6 8	[0,1]	sum = 14	print(14)
6 8 1	[0,2]	sum = 15	print(15)
6 8 1 7	[0,3]	sum = 22	print(22)

8

8 1

8 1 7

1

1 7

7.

arr[4] = [$\overset{0}{6}$, $\overset{1}{8}$, $\overset{2}{1}$, $\overset{3}{7}$]

Dry run: s = 0 , sum = 0

e = 0 —— sum = sum+ arr[e] print(6).

sum = 6.

e = 1 —— sum = sum+ arr[1] print(14)

sum = 6 + 8 = 14

e = 2 —— sum = sum+ arr[2] print(15)

sum = 14 + 1 = 15

e = 3 —— sum = 15 + 7 = 22 print(22)

s = 1 ,   sum = 0

e = 1 —— sum = sum + arr[1]   print(8)

sum = 8

e = 2 —— sum = sum + arr[2]   print(9)

sum = 8 + 1 = 9

e = 3 —— sum = 9 + arr[3] = print(16)

```
void printSumofEachsubarray(int[] arr){
    for( s = 0; s < arr.length; s++) {
        int sum = 0;
        for( e = s; e < arr.length; e++) {
            sum += arr[e];
            print(sum);
        }
    }
}
```

TC: $O(n^2)$

SC: $O(1)$

Qu   Given arr[n], return sum of every subarray sum.

arr[] = [6, 8, 1, 7]

subarray	sum
6	6
6 8	14
6 8 1	15
6 8 1 7	22
8	8
8 1	9
8 1 7	16
1	1
1 7	8
7	7

sum = 106

Break: 8:27 AM

Approach1:

```
int    printSumofEachsubarray(int[] arr){
    int sum=0;

    for( s=0; s< arr.length; s++) {

        for( e=s; e< arr.length; e++){

            sum += arr[e];

        }
    }
   return sum;
}
```

TC: $O(n^2)$

SC: $O(1)$

Approach2: SC: O(1) TC: O(n)

arr[4] = [6, 8, 1, 7]

subarray	sum
6	6
6 8	6 + 8
6 8 1	6 + 8 + 1
6 8 1 7	6 + 8 + 1 + 7
8	8
8 1	8 + 1
8 1 7	8 + 1 + 7
1	1
1 7	1 + 7
7	7

sum = 6 * 4 + 8 * 6 + 1 * 6 + 7 * 4 = 106

$$sum = \underset{\substack{\uparrow \\ arr[0]}}{6} * \underset{\substack{\uparrow \\ \text{occ of 6 in all subarrays}}}{4} + \underset{\substack{\uparrow \\ arr[1]}}{8} * \underset{\substack{\uparrow \\ \text{occ of 8 in all subarrays}}}{6} + \underset{\substack{\uparrow \\ arr[2]}}{1} * 6 + \underset{\substack{\uparrow \\ arr[3]}}{7} * 4.$$

arr[n] ——

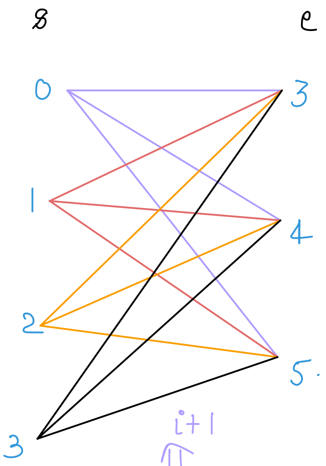
sum = arr[0] * occ0 + arr[1] * occ1 + arr[2] * occ2 - - -

- - - - - arr[n-1] * occ_{n-1}.

Challenge: find occ of every element.

$arr[6] = [\ \overset{0}{3}\ \ \overset{1}{-2}\ \ \overset{2}{4}\ \ \overset{3}{-1}\ \ \overset{4}{2}\ \ \overset{5}{6}\ ]$

case1: find occ of -1 [idx = 3]



$$\underset{[0,i]}{\overset{i+1}{\Uparrow}} \qquad \underset{[i,n-1]}{\overset{n-\cancel{1}-i+\cancel{1} = n-i}{\Uparrow}}$$

$$\text{occ of } -1 = \underset{[0,3]}{s} * \underset{[3,5]}{e}$$

$4 * 3 = 12.$

3 -2 4 -1 [0,3]

3 -2 4 -1 2 [0,4]

3 -2 4 -1 2 6 [0,5]

-2 4 -1 [1,3]

-2 4 -1 2 [1,4]

-2 4 -1 2 6 [1,5]

4 -1 [2,3]

4 -1 2 [2,4]

4 -1 2 6 [2,5]

-1 [3,3]

-1 2 [3,4]

-1 2 6 [3,5]

Generalise: occ of any el at idx i —

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

$sum = arr[0] * occ0 + arr[1] * occ1 + arr[2] * occ2$ - - . .

- . . . . $arr[n-1] * occ_{n-1}$.

$sum = arr[0] * \{(0+1) * (n-0)\} + arr[1] * \{(1+1) * (n-1)\}$ . . –

. . . . — —

Correct: find occ of every idx as part of subarray

```
int  totalsum( int[] arr) {

    int n = arr.length;

    int  sum=0;

    for( i=0;  i<n;  i++) {
        int occ =  (i+1) * (n-i);
        sum =  sum + arr[i]  *  occ;
    }

   return  sum;
}
```

TC O(n)

SC: O(1)

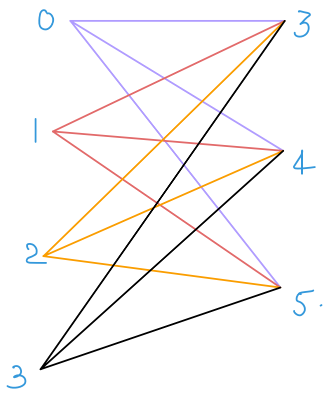
Thankyou ☺

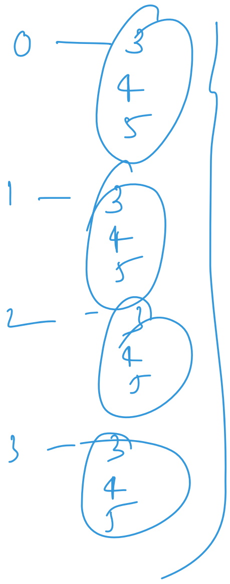
Doubts:

0 1 2 ③ 4 5

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

s [0-3] e [3-5]





$\frac{[0,3]}{④} * \frac{[3-5]}{③}$

[0,3] [3,5]

[0, i] [i, n-1]

↓ ↑

(i+1) $n-\cancel{1}-i\cancel{+1}$

(n-i)

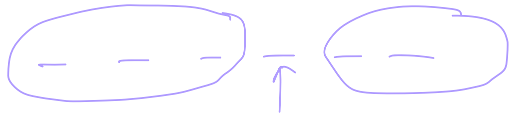
(i+1) * (n-i)

$\boxed{1*2*3*5}$

A = [1, 2, 3, ④, 5] product = 120

op [120 60 40 30 24]

A =



$$\begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \end{array}$$

A = [1, 2, 3, 4, 5]

1 * 2 * 3 * 5

α pf = [1 2 6 24 120]

2 * 3 * 4 * 5

sf = [120 120 60 20 5]

i ↑

mul(1, 4) = sf(1)

op = [ ]

4 → $\boxed{\text{mul}(0, 2)}$ * mul(4, 4)

pf[2] sf[4]

i

[1, 2, 3, 4, 5, 6]

i ↑

(0, i)

[1, 3, 6, 10, 15, 21]

↑

pf(i) pf(i-1)