Subarrays

Jul 5, 2023

AGENDA

- Intro to subarrays
- total no. of subarrays
- generate / print all subarrays
- sum of all subarrays
- sum of all subarray sums (Contribution Technique)

Subarrays

Contigous part of an array.
Can have one or more elements.
Order should remain same.

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

[2 3 -1 6] is a subarray of length 4.

[9] is a subarray of length 1.

[4 1 2 3 -1 6 9 8 12] is a subarray containing all elements.

[4 12] → not contigous.

[3 2 1 4] → order is not same.

[2 4 1 6 -3 7 8 4]

[4, 5, 1, 9, 0, 2, 3, 5]

[5) ✓

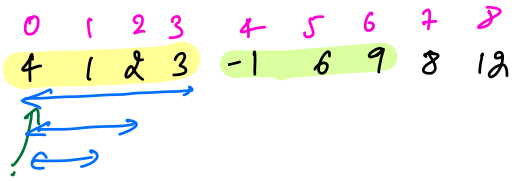
[4, 5, 1, 0] ✗

[9, 0, 2, 3] ✓

[4, 5, 1] ✓

acd

Representation of a subarray



[4 1 2 3)

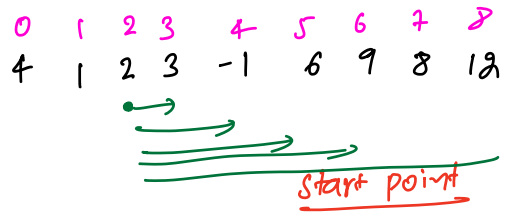
[-1 6 9]

(start index, end index) pair represents a subarray.

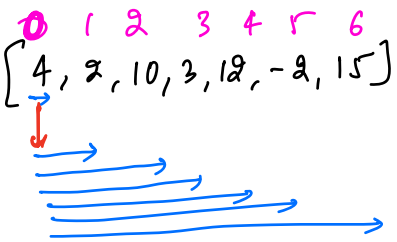
{-1, 6, 9} → (4, 6)

{4, 1, 2, 3} → (0, 3)

Count no. of subarrays



start point	end points
0	0,1,2,3,4,5,6,7,8. (N)
1	1,2,3,4....8 (N-1)
2	2,3,4,5,....8 (N-2)
⋮	⋮
N-1	8. (1)



Total no. of subarrays

$$= N + N-1 + N-2 + N-3 + \dots + 1$$

$$= \frac{N(N+1)}{2}$$

[1, 0, 7, 3]

$$\frac{4(4+1)}{2} = 10$$

1
10
107
1073

0
07
073
3

7
73

* Print a subarray

```
// start ind, end ind is given.
for(int i = start ; i<= end ; i++)
 {
        print (arr[i])
 }
```

T.C. → O(N)
S.C. → O(1)

* Print all subarrays of the array

[1, 0, 7, 3]
 i j

Generate all possible (start ind, end ind) pairs and print those.

```
for (int i=0 ; i<n ; i++)
{
     for( int j = i ; j<n; j++)
      {
            // Now I have a (i,j) pair.
            // Print this subarray.
            for(int k= i ; k<=j ; k++)
            {    print (arr[k])
            }
            print("\n")
```

0 1 2 3
[1, 0, 7, 3]

i=0, j=0
j=1
j=2
j=3
i=1, j=1
j=2
j=3
:

1
1 0
1 0 7
1 0 7 3

```
    }
}
```

T.C. → $O(N^3)$

S.C. → $O(1)$

[1, 0, 7, 3, 2, 4] → N

$\frac{N(N+1)}{2}$: no. of subarrays.

$O(N^2)$

$O(N)$ to print it.

$O(N^3)$ is the best possible soln

Q. Given an array, find sum of each subarray in the array.

[1, 0, 7, 3]

1 : 1
10 : 1
107 : 8
1073 : 11
0 : 0
07 : 7
073 : 10
7 : 7
73 : 10
3 : 3

Brute Force

```
for (int i=0; i<n; i++)
{
    for( int j=i; j<n; j++)
    {
        // Now I have a (i,j) pair.
        // find the sum of this subarray.
        sum=0
        for(int k=i ; k<=j; k++)
        {
            sum += arr[k]
        }
        print(sum)
    }
}
```

T.C. → O(N^3) ×
S.C. → O(1)

Range-sum Query [1 0 7 3]

(finding the sum of a range again and again)

Use prefix sum.

```
// Generate prefix-sum array for the
// input. Call it "Pf".

for (int i=0; i<n; i++)
{
    for( int j=i; j<n; j++)
    {
        // Now I have a (i,j) pair.
        // find the sum of this subarray.
        if(i==0)  sum = Pf[j]
          else    sum = Pf[j] - Pf[i-1]

        print(sum)

    }
}
```

T.C. → $O(N^2)$

S.C. → $O(N)$ ✗

0 1 2 3

[1 0 7 3]

[0,0] = 1
[0,1] = 1
[0,2] = 1+7 = 8
[0,3] = 8+3 = 11

[1,1] = 0
[1,2] = 0+7 = 7
[1,3] = 7+3 = 10

[2,2] = 7
[2,3] = 7+3 = 10

[3,3] = 3

Carry-forward

```
for (int i=0; i<n; i++)
{
        sum = 0
        for( int j = i; j<n; j++)
        {
                // Now I have a (i,j) pair.
                // find the sum of this subarray.
                sum += arr[j]
                print(sum)
        }
}
```

[2, 3, 7, 4] (0 1 2 3)

i=0
sum=0
j=0 sum=2
j=1 sum=5
j=2 sum=12
j=3 sum=16

i=1
sum=0
j=1 sum=3
j=2 sum=10
j=3 sum=14

Output

2, 5, 12, 16, 3, 10, 14

T.C. → $O(N^2)$

S.C. → $O(1)$

Break till 8:25 AM

Q. Find the sum of all subarray sums of an array.

[1, 0, 7, 3]

1 : 1
10 : 1
107 : 8
1073 : 11
0 : 0
07 : 7
073 : 10
7 : 7
73 : 10
3 : 3

= 58

```
ans = 0
for (int i = 0; i < n; i++)
{
        sum = 0
        for (int j = i; j < n; j++)
        {
                // Now I have a (i,j) pair.
                // find the sum of this subarray.
                sum += arr[j]
                ans += sum

        }
}
print(ans)
```

T.C. → O(N²)
S.C. → O(1)

Use the same variable

```
sum = 0
for (int i = 0; i < n; i++)
{
        for( int j = i; j < n; j++)
        {
                // Now I have a (i,j) pair.
                // find the sum of this subarray.
                sum += arr[j]

        }
}
print(sum)
```

T.C. → $O(N^2)$

S.C. → $O(1)$

Contribution technique

[1, 0, 7, 3]

1 = 1

10 = 1+0

107 = 1+0+7

1073 = 1+0+7+3

0 = 0

07 = 0+7

073 = 0+7+3

7 = 7

73 = 7+3

3 = 3

→ 1+1+0+1+0+7 + 1+0+7+3 + 0+0+7+0+7+3 + 7+7+3+3

= 1*4 + 0*6 + 7*6 + 3*4

= 4+0+42+12

= 58

[1, 0, 7, 3] Total subarray sum = 58

1 * ? + 0 * ? + 7 * ? + 3 * ?

what is the contribution of 1?

* Goal:

find the contribution of each element arr[i] in the final total sum.

[3, 7, 4] = 49

8 4 3

3
3 7
3 7 4
7
7 4
4

3+3+7+3+7+4+7+7+4+4

[2, 4, 7]
3 4 3

= 2*3 + 4*4 + 7*3
= 6 + 16 + 21
= 43

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

= 2*4 + 1*6 + 3*6 + 4*4
= 48

2
2 1
2 1 3
2 1 3 4

2 1 3 4
1 3 4
3 4
4

2 1
2 1 3
2 1 3 4
1
1 3
1 3 4

* Contribution of arr[i] in the final sum
= arr[i] * no. of subarrays in which arr[i] occurs.

How to find out
no. of subarrays in which arr[i] is present?

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

In order for your subarray to contain -2,
possible start points? 0 or 1 index

i+1 st
n-i end

possible end points? 1, 2, 3, 4, 5 index.

start: 0, 1

end: 1, 2, 3, 4, 5

Total no. of subarrays containing -2 = 2 * 5 = 10

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

$i+1 = 3$ start

$n-i = 6-2 = 4$ end pt

To contain 2,

possible start points : 0, 1, 2

possible end points : 2, 3, 4, 5

No. of subarrays = $3 * 4 = 12$

0	1	2	3	4	5
3	-2	4	-1	2	6
			i		

No. of subarrays in which arr[i] is present:

		Number
possible start points:	$0, 1, 2 \ldots i$:	$i+1$
possible end points:	$i, i+1, i+2 \ldots n-1$:	$n-i$

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

Total no. of subarrays = $(i+1) * (n-i)$

Contribution of arr[i] = $arr[i] * (i+1) * (n-i)$

Code.

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

T.C. → O(N)
S.C. → O(1)

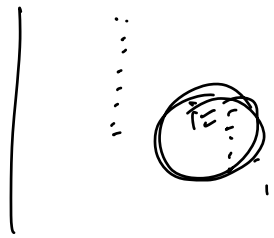
Observations

* Subarray → contigous part of an array
* No. of subarrays → $\frac{N(N+1)}{2}$
* Print all subarrays → $O(N^3)$
* Print sum of each subarray → $O(N^2)$ (Prefix sum / C.F.)
* Print total sum of all subarray sums → $O(N)$: Contribution Technique

---

[1, 1, 2, 1]
4 6 6 4

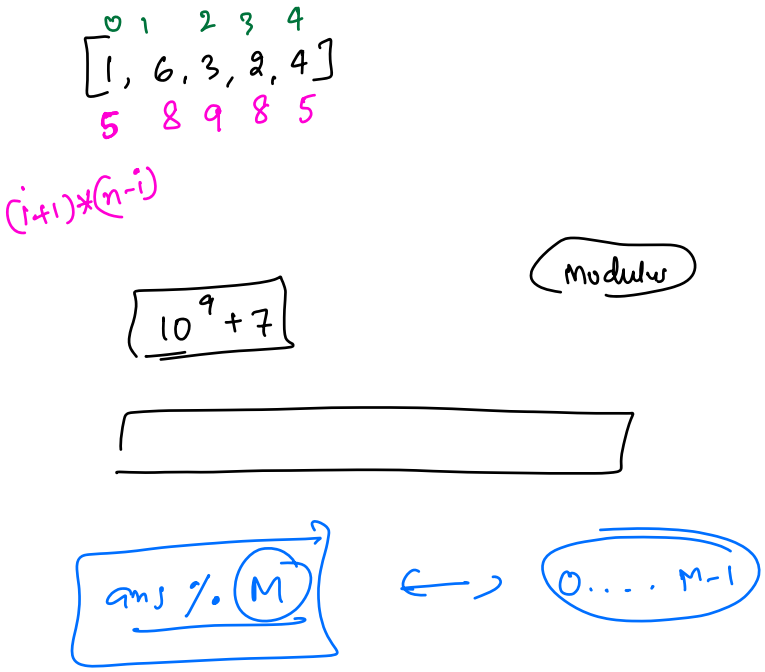
1*4 + 1*6 + 2*6 + 1*4



Compile error
Runtime error.

Unexpected output
↳ Logical error

adding print statements
Doing a dry Run.


0 1 2 3 4
[1, 6, 3, 2, 4]
5 8 9 8 5
(i+1)*(n-i)
Modulus
10^9+7
ans % M
0 . . . . M-1