Arrays – Subarrays

Subarray — Contiguous chunk | part of an array

✓ complete array ≡ subarray

✓ single element ≡ subarray

✓ φ array ≡ subarray

A = | 3 | 8 | 7 | 6 |

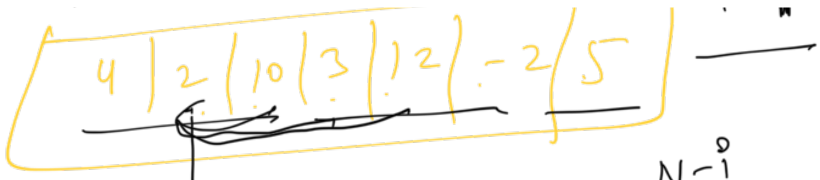
A1 =	7	6	Yes
A2 =	8	7	6	Y
A3 =	9	7	6	N
A4 =	3	Y

A5 =	3	8	7	6	
A6 =	6	7	N

Qus

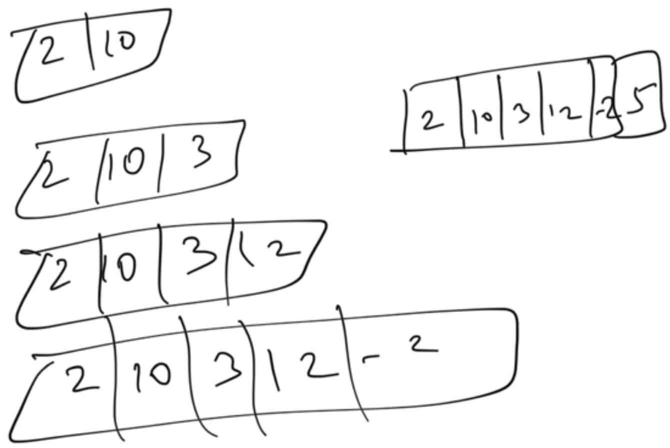
N = 7

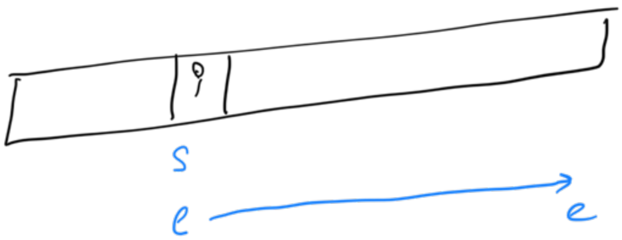
i =



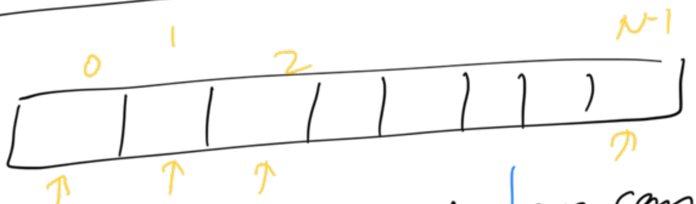
N - i

2





|subarray_i| = N - i

A = 

Q) How many subarrays can exist

non empty

subarrays (non empty)

= # subArr start 0

+ # subA 1.

+ # sub 2.

+ |sub 3|

+ |sub 4|

+ |sub N-1|

$|subA| = |subA_0| + |subA_1| \ldots |subA_{N-1}|$

$N-0$ $\quad N-1$ $\quad N-(N-1) = 1$

$= N + (N-1) + (N-2) + (N-3) \ldots 1.$

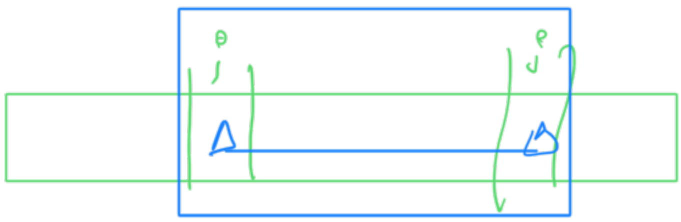
$1 + 2 + 3 + 4 \ldots N$

$= 1+2+\dots$

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

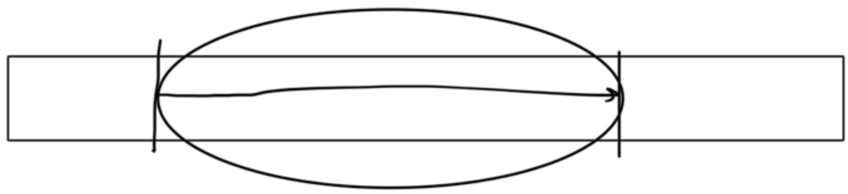
$|A| = 20$

Q Print all values of a subarray starting with index = 5 and ending with index = 12

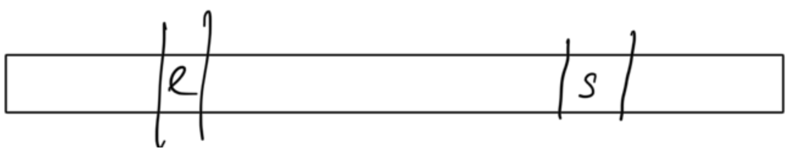


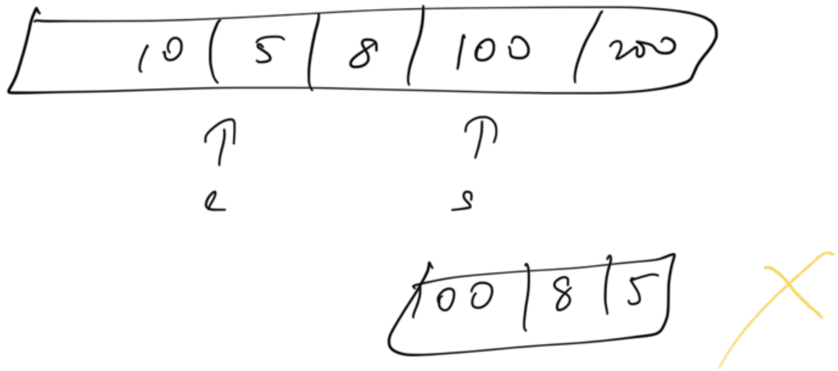
start = $i = 12$

end = $j = 5$

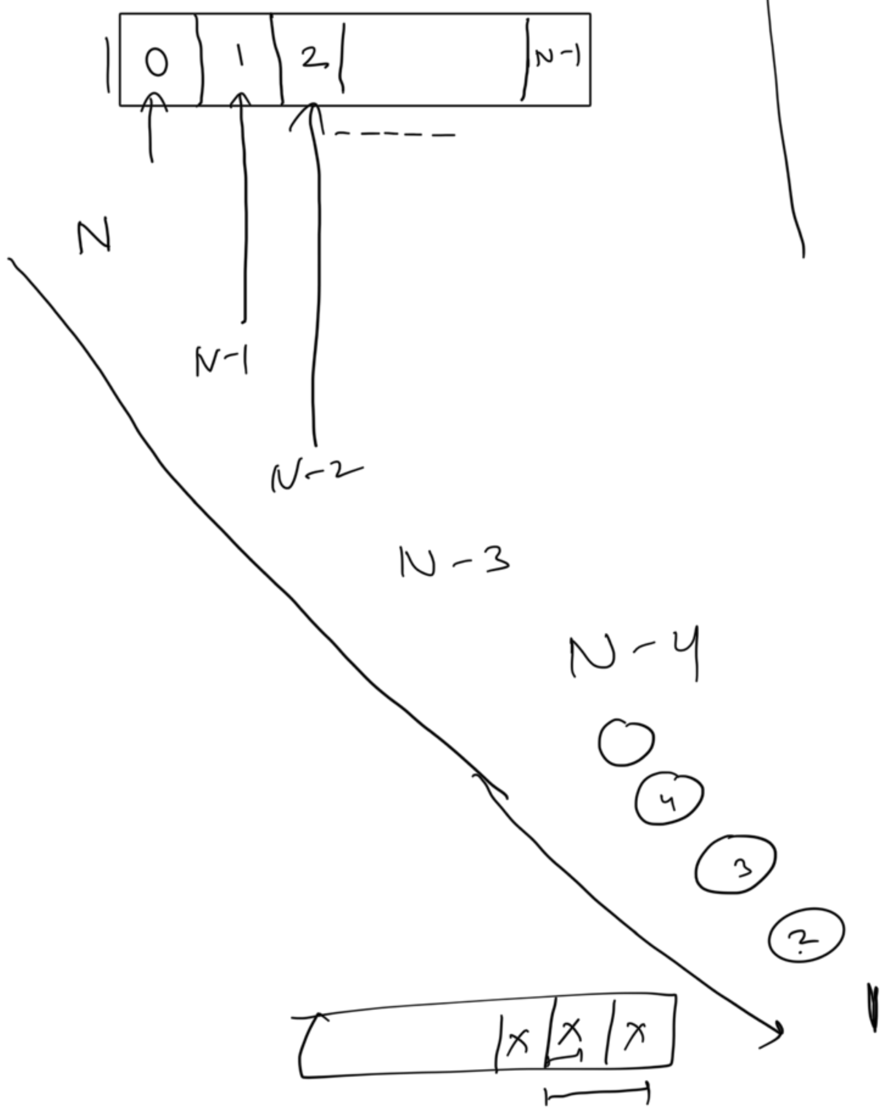


start > end





$i \Rightarrow N-i$ elements



$1 + 2 + 3$ — — N 1

(Q) $A = [\ 2 \quad 8 \quad 5 \quad 4\ ]$

$\quad\quad\quad\ \ 0 \quad 1 \quad 2 \quad 3$

Print all non empty subarrays :)

[start, end]

1 loop

O(N) ← for (i = 0, i < N, i++) ≡ loop for start

{

end O(N) ← for (j = i , j < N, j++)

{

for (k = i, k <= j, k++)

print(arr[k]);

O(N)

Print (st, end)

}

}

O(N^3)

T.C = O(N

auxillary S.C = O(1)

Start ≡ for (i=0, i<N, i++)

{

for (j=i, j<N, j++)

end ≡

}

$$1\,T = \frac{4\times 5}{2} = 10 \qquad \frac{4\times 5}{2} = 10$$

i → i

| 2 | 10 | 8 | 18 | = N = 4

	2	10	8	18
i=1	10	8	18	
i=2	8	18		
i=3	8			

```
for (i=0, i<N, i++)
{
    for(j=i ; j<N, j++)
    {
        for (k=i , k<=j, k++)
        {
        }
    }
}
```

$O(N^2)$ subarray

$\times$

$O(N)$ print

$O(N^3)$:-|

Q A = [ ]

Print sum of every subarray

A = | 2 | 10 | 8 | 4 |

i, j →

10 → 10

10 8 → 18

10 8 4 → 22

2 → 2

2 10 → 12

2 10 8 → 20

2 10 8 4 → 24

2 10 0 1

8 → 8

8 4 → 12

4 → 4

Claim: $O(N^2)$ subarrays

Claim 2: Print all subarray → $O(N)$ time

T.C. can't be better than $O(n^3)$

Claim 3:

sum

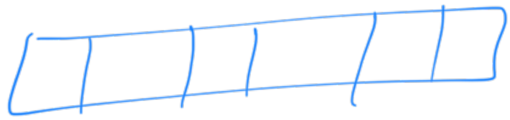
I can't solve sum/subarray in less $O(n^2)$ time

~~No longer true~~

Print an element sum

$O(1)$ time

Claim: You can't print all subarray sums in less than $O(n^2)$ time



Boiled Info ≡ Sum

Carry forward.??

	0	1	2	3
A	2	10	8	4

2	2
2 10	12
2 10 8	20
2 10 8 4	24
10	10
10 8	18
10 8 4	22
8	8
8 4	12
4	4

0	1	2	3
2	10	8	4

☆ $i=0$

2	$a[0] \to O(1)$	$j=0$
2 10	$a[0] + a[1] \to O(1)$	1
2 10 8	$a[0] + a(1) + a(2) \to O(1)$	2
2 10 8 4	$a(0) + a(1) + a(2) + a(3)$ $O(1)$	$j=3$

☆ $i=1$

10	$a(1) \to O(1)$	$j=1$
10 8	$O(1)$ $a(1) + a(2)$	2
10 8 4	$O(1)$ $a(1) + a(2) + a(3)$	3

☆ $i=2$

8	$a(2) \to O(1)$	$j=2$
8 4	$a(2) + a(3) \to O(1)$	$j=3$

☆ $i=3$

4 $\quad a(3) \rightarrow O(1)$ $\quad j = 3$

Pseudocode

2	8	4	10

i

start index $\Leftarrow$ for ($i = 0$, $i < N$, $i{+}{+}$)

{

$\boxed{Sum = 0}$

end index — for ($j = i$, $j < N$, $j{+}{+}$)

{

Sum = sum + a(j)

print (sum)

}

}

$T.C = O(N^2)$

auxilliary $S.C = O(1)$

$O(N^3)$

$N = 500$

$O(N^2)$

$N = 10{,}000$ $\quad$ 1sec

Q Print every possible subarray

$O(N^2) \times O(N)$

$= O(N^3)$

Q Print the sum of every subarray

$O(N^2)$

Q Print the sum of all the subarray sums

A = | 2 | 8 | 10 | 4 |

2 = 2	8 = 8	10 = 10	4 = 4
2 8 = 10	8 10 = 18	10 4 = 14	
2 8 10 = 20	8 10 4 = 22		
2 8 10 4 = 24			

Subarr Sum

= 132 ☆

2	8	10	4

$i = 0$

$i = 1$

$2 + 8 + 10 + 4$

2

10

20

24

n.w

n

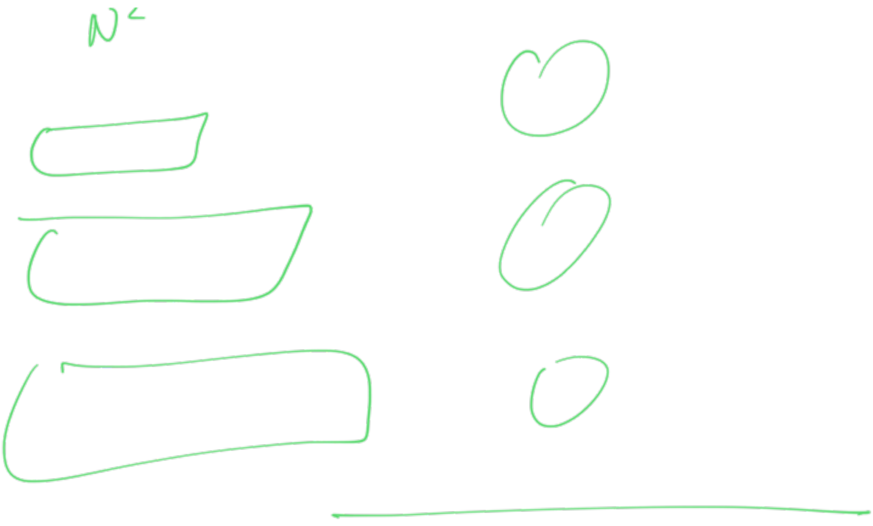
n

```
sum = 0
for( i=0, i<N, i++)
{
    tempsum = 0
    for( j=i, j<N, j++)
    {
        tempsum = ts + a[j]
        sum = sum + temps
    }
}
print(sum)
```

T.C = $O(N^2)$

8:58a

9:00a

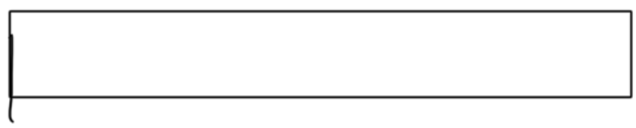


$$Ans = \Sigma(\text{subarraySum})$$

Claim 1: You have to print 1 global sum

Claim 2: O(1) time will be taken to print any global sum

Claim 3: Baseline ≡ constant time



Boiled ≡ sum up the subarray

$A = \begin{array}{|c|c|c|c|} \hline 8 & 4 & 10 & 2 \\ \hline \end{array}$

$\quad\;\; 0 \quad 1 \quad 2 \quad 3$

$i=0$

✓	8	$a(0)$
✓	8 4	$a(0) + a(1)$
✓	8 4 10	$a(0) + a(1) + a(2)$
✓	8 4 10 2	$a(0) + a(1) + a(2) + a(3)$

$i=1$

4	$a(1)$
4 10	$a(1) + a(2)$
4 10 2	$a(1) + a(2) + a(3)$

$i=2$

10	$a(2)$
10 2	$a(2) + a(3)$

$i=3$

2	$a(3)$

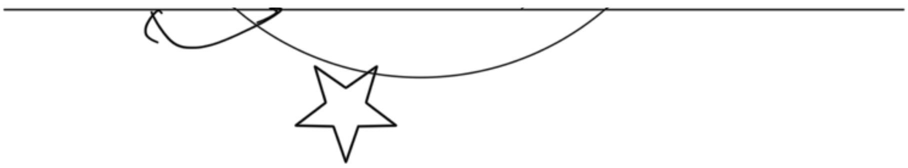
$1 \times 4 = 4 \qquad 4 \times a(0) \longrightarrow N$ time

$2 \times 3 = 6 \qquad 6 \times a(1)$

$3 \times 2 = 6 \qquad 6 \times a(2)$

$4 \times 1 = 4 \qquad 4 \times a(3) \longrightarrow N$ time

$\# s = (i+1) \times (N-i)$



$$\text{Ans} = \begin{cases} c_0 \times a(0) \\ + \\ c_1 \times a(1) \\ + \\ c_2 \times a(2) \\ + \\ c_3 \times a(3) \\ \vdots \\ c_{N-1} \times a(n-1) \end{cases}$$

$2+2+2+2$
$= \underline{4} \times \underline{2}$

Linear Combination

$c_i \equiv$ # subarray i contributes

2

2 8

2 8 10

= 2

= 10

= 20

2 8 10 4 = 24

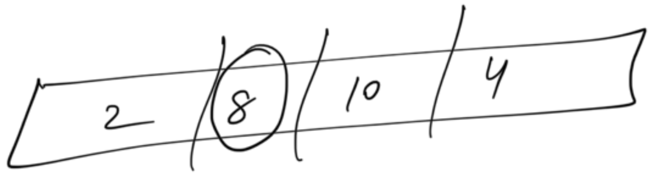
56

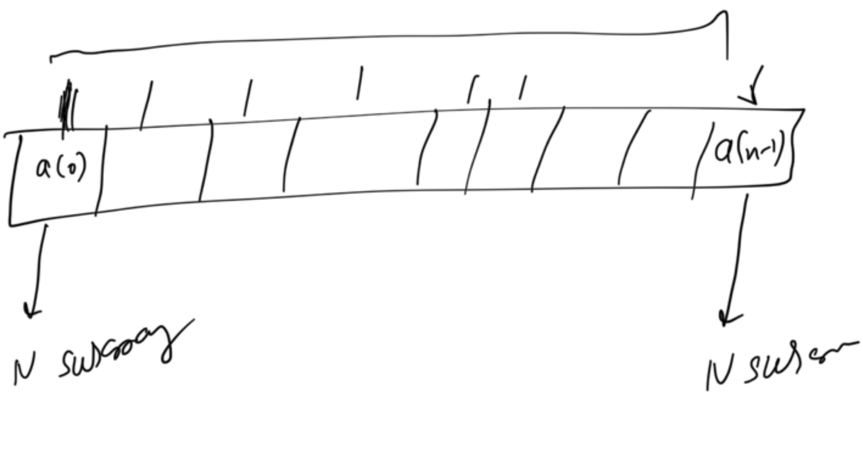
2 + 2 + 8 + 2 + 8 + 10
+ 2 + 8 + 10 + 4

= 56

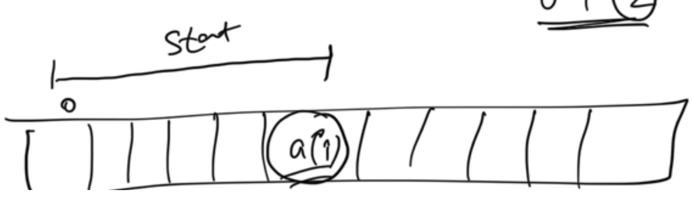
How to logically figure out in how many subarrays do we have a number

a_i?





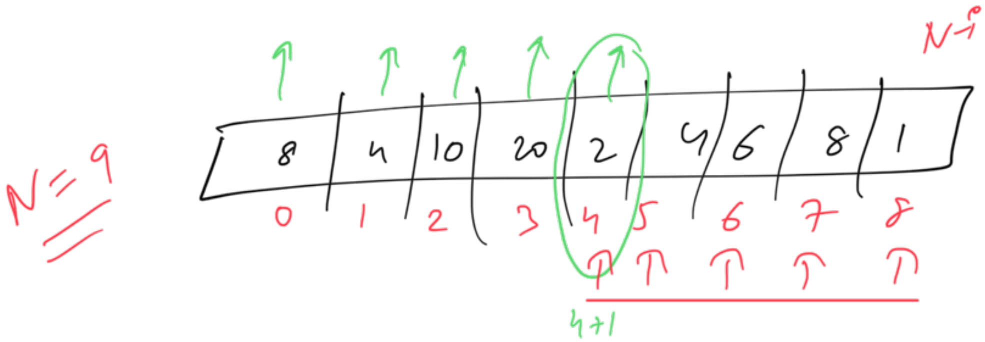
0 1 2



⊢ end ⟶

start pts ≡ $(i+1)$

Any subarray that starts till me and ends beyond me (myself included) will have my contribution

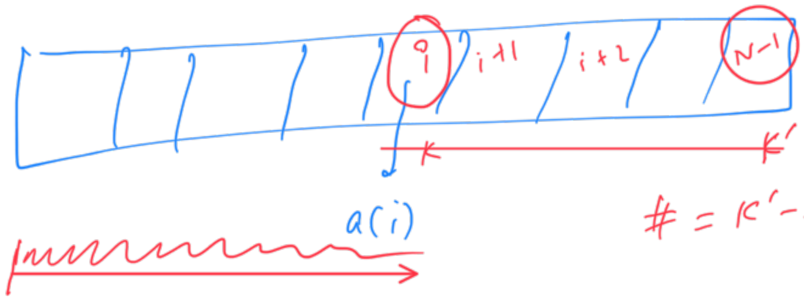


prospective starting pts $= (i+1) =$

prospective ending pts $= (N-i) =$

[illegible] $= (i+1) \times (N-i)$

sum



$\# = k' - k + 1$

$j - i + 1$

$N - 1 - i + 1$

$N - i \cancel{- 1} \cancel{+1}$

$= N - i$

$i+1$

subarray =

$(i+1) \times (N-i)$

```
sum = 0
for (i=0, i<N, i++)
{
    sum = sum + [(i+1) x (N-i) x a(i)]
}
```

$T.C = O(N)$

$\text{aux } S.C = O(1)$

$A = \begin{array}{|c|c|c|c|c|c|} \hline 3 & -2 & 4 & -1 & 2 & 6 \\ \hline \end{array}$

$\underline{\underline{i=1}}$

$N - i = 6-1$
$= 5$

$(i+1) = 2$

$(N-i) = 5$