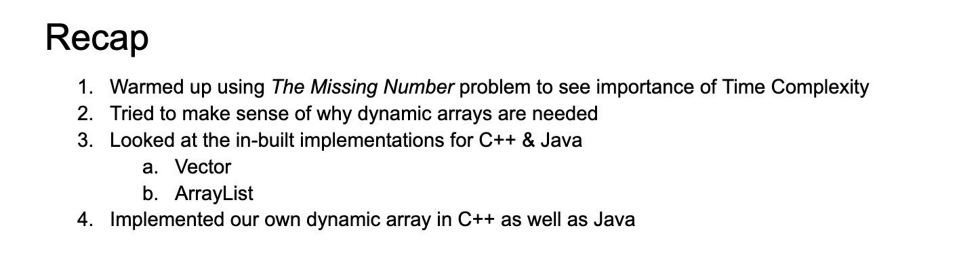
Hey folks, let's start at 9:02PM

L21
Arrays - Practice 1

*Please don't forget to join the doubt session this Sunday (11th Feb) at **9 PM**.*

Join Discord - https://bit.ly/ly-discord



Recap

1. Warmed up using *The Missing Number* problem to see importance of Time Complexity
2. Tried to make sense of why dynamic arrays are needed
3. Looked at the in-built implementations for C++ & Java
    a. Vector
    b. ArrayList
4. Implemented our own dynamic array in C++ as well as Java

cap = 8
size = 8

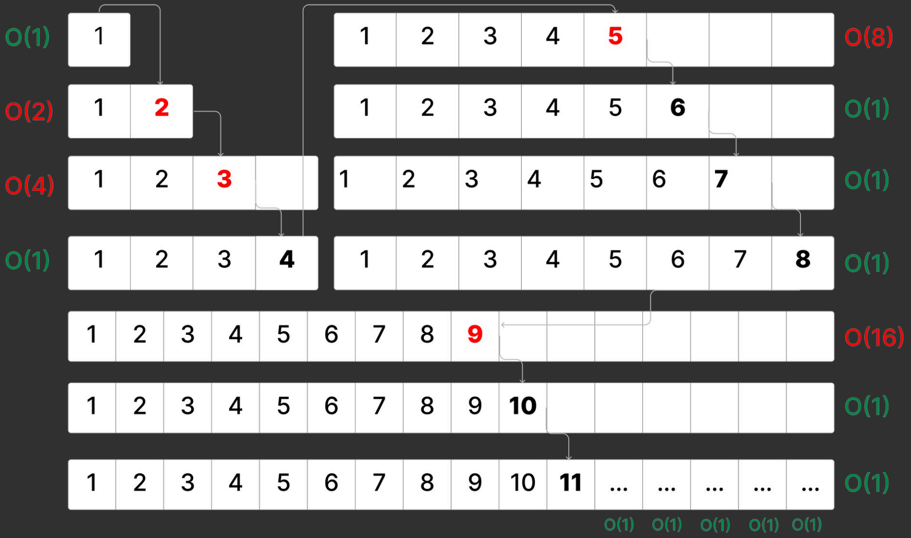
Time Complexity Analysis

$2 * cap + cap$

$\Rightarrow 3 * cap$

cap = 16
size = 16

5	1	2	3	4	3	7	16	25	17	-	-	-	-	-	-


O(1)
1
O(2)
1 2
O(4)
1 2 3
O(1)
1 2 3 4
1 2 3 4 5
O(8)
1 2 3 4 5 6
O(1)
1 2 3 4 5 6 7
O(1)
1 2 3 4 5 6 7 8
O(1)
1 2 3 4 5 6 7 8 9
O(16)
1 2 3 4 5 6 7 8 9 10
O(1)
1 2 3 4 5 6 7 8 9 10 11 ... ...
O(1)
O(1) O(1) O(1) O(1) O(1)

Observation ⇒ Sometimes only 1 operation is req. to push_back/add, but sometimes you've to do 3*N steps.

If you do N (where N is a large numbers) push_back/add operations:

cap = $2^0, 2^1, 2^2, 2^3, 2^4, 2^5$ — — .

whenever the size/N is a power of 2, then it takes 3*N steps for the next push_back/add operation, otherwise 1 step only

$1, 2, 4, 8, 16, 32, 64 \leftarrow 100 \Rightarrow \log_2 100 = \lceil 6. \ldots 7$

$$2^K \le N$$

$$K' \le \log_2 N$$

⇙

$0, 1, 2, 3 \ldots\ldots \log_2 N$

$$\underbrace{\{1 + 1 + 1 + 1 \cdots \cdots 1\}}_{N} + 3 * \underbrace{\{2^0 + 2^1 + 2^2 + 2^3 + 2^4 \cdots\}}_{\log_2 N} \Rightarrow \frac{2^0 * \left(2^{\log_2 N} - 1\right)}{2-1}$$

$$N + 3 * (N - 1)$$

$$\Rightarrow 4 * N - 3 \Rightarrow O(N)$$

Total $\Rightarrow$ $4N - 3$

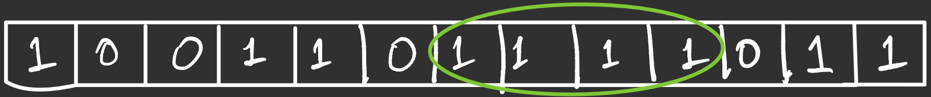
Avg. steps per operation $= \dfrac{4N-3}{N} \Rightarrow 4 - \dfrac{3}{N} \; (<4)$

$\Rightarrow \sim O(1)$

Amortised time complexity

Let’s do some practice now

Max Consecutive Ones



1	0	0	1	1	0	1	1	1	1	0	1	1

N = 13

cur = 2

ans = 4

Max Sum Subarray of Size K

N = 10

K = 4

10	1	5	3	2	2	8	9	3	5

[10, 1, 5, 3] ⇒ 19

[1, 5, 3, 2] ⇒ 11

[5, 3, 2, 2] ⇒ 12

[3, 2, 2, 8] ⇒ 15

[2, 2, 8, 9] ⇒ 21

[2, 8, 9, 3] ⇒ 22

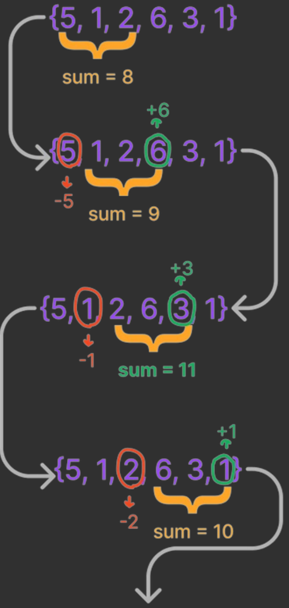
[8, 9, 3, 5] ⇒ 25

$O(K)$ per subarray.

$N-K+1$ subarrays

$$(N-K+1) * K \Rightarrow N * K - K^2 + 1$$

$$\Rightarrow O(N * K)$$



{5, 1, 2, 6, 3, 1}

sum = 8

+6

{5, 1, 2, 6, 3, 1}

-5

sum = 9

+3

{5, 1, 2, 6, 3, 1}

-1

sum = 11

+1

{5, 1, 2, 6, 3, 1}

-2

sum = 10

Max Subarray Sum = 11

5	1	2	6	3	1

Thank You!

Reminder: Going to the gym & observing the trainer work out can help you know the right technique, but you'll muscle up only if you lift some weights yourself.

So, PRACTICE, PRACTICE, PRACTICE!