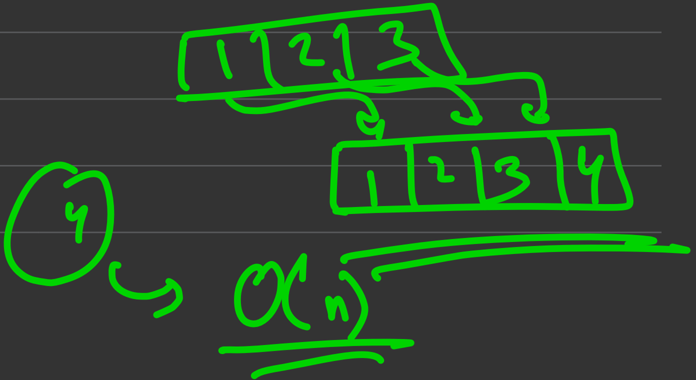
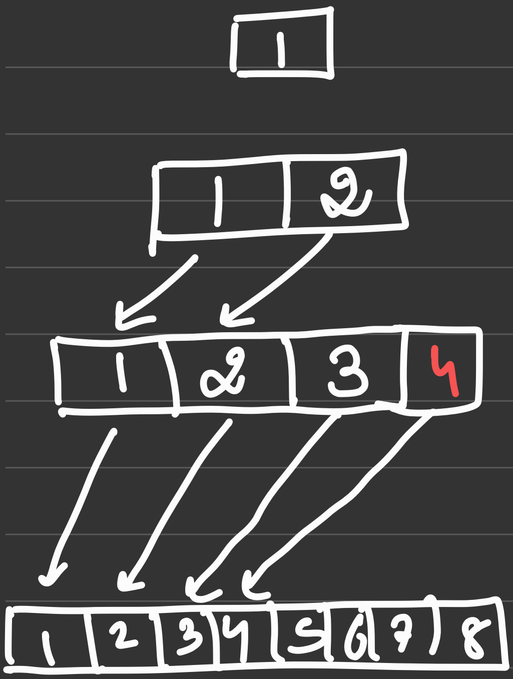
↳ There are problem with arrays →

1) We need a size to init array & that size is fixed

2) During runtime we can't inc or dec length of array.



↳ If we try to create a new array of size $\underline{\underline{n}}+1$ → Then for one insertion we will take $O(n)$ $\underline{\underline{\text{time}}}$



| Capacity | Size |
|---|---|
| 1 | 0 |
| 2 | 1 |
| 4 | 3 |
| 4 | 4 |
| 8 | 5 |
| 8 | 6 |
| 8 | 7 |
| 8 | 8 |

1, 2, 3 . . . . 9

operations

$1 \rightarrow 1$

$2 \rightarrow 2 \rightarrow 2^0+1$

$3 \rightarrow 3 \rightarrow 2^1+1$

$4 - 1$

$5 - 5 \rightarrow 2^2+1$

$6 - 1$

$7 - 1$

$8 - 1$

$9 - 9 \rightarrow 2^3+1$

avg time $\rightarrow$ $\dfrac{\text{total no. of operations}}{\text{no. of insertions}}$

$$\frac{1 + 2 + 3 + 1 + 5 + 1 + 1 + 1 + 9 \ldots\ldots}{n}$$

$$\frac{\left(\overbrace{1 + 1 + 1 + 1 \ldots\ldots}^{n}\right) + 2^0 + 2^1 + 2^2 \ldots\ldots 2^{\log n}}{n}$$

$2^{\log_2 n} \rightarrow n$

$$\frac{n + 1 \times (2^{\log n} - 1)}{n}$$

$$\frac{n + n - 1}{n} \Rightarrow \frac{2n-1}{n} \rightarrow \underline{\underline{\text{const}}}$$

avg → per insertion you took <u>const time</u>

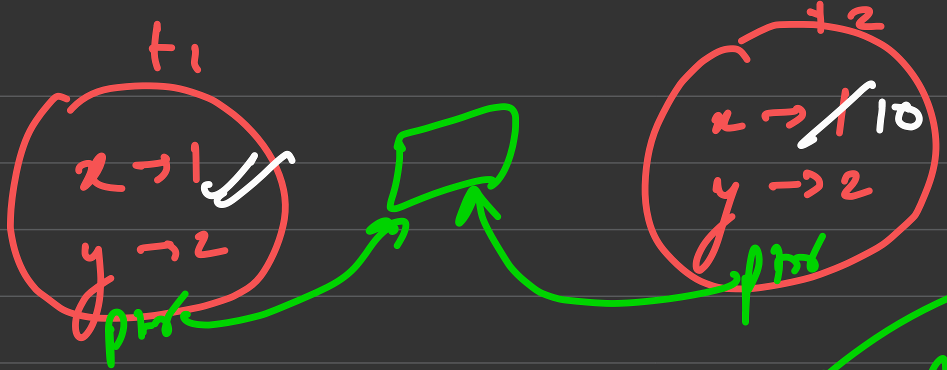
↳ array → insertion → $\underline{\underline{\Theta(1)}}$

amortized analysis

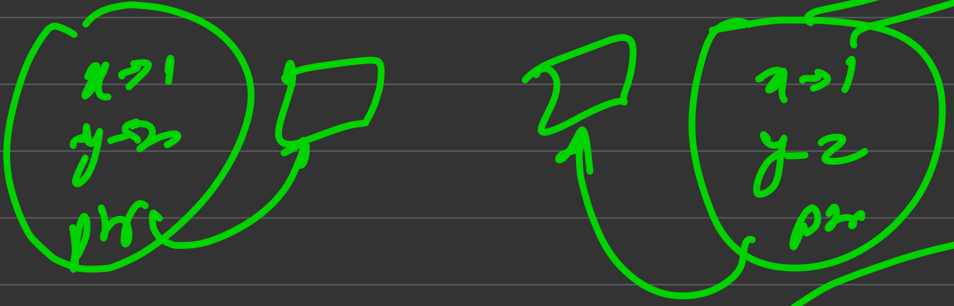
C++ → Vector

Java → ArrayList

Python → List

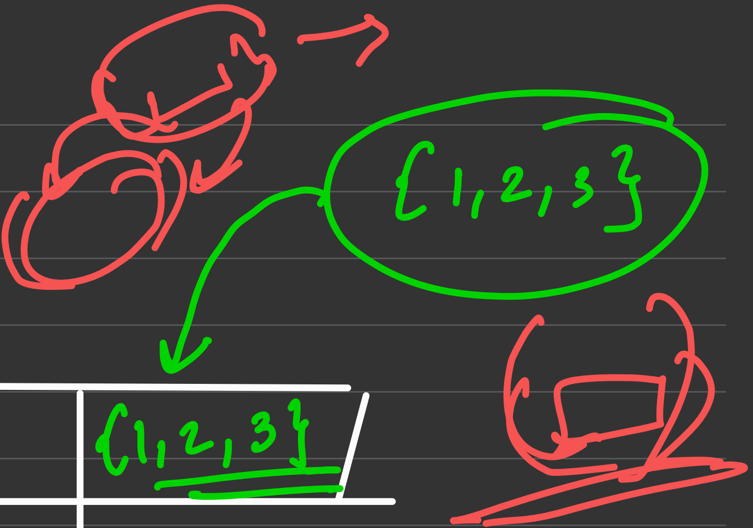


Shallow copy



Deep copy

Push_back →



obj

v. push_back({1, 2, 3})

v. push_back(obj)