Vectors and Strings

Constructed
functions
optimizations

Vectors → STL → Containers

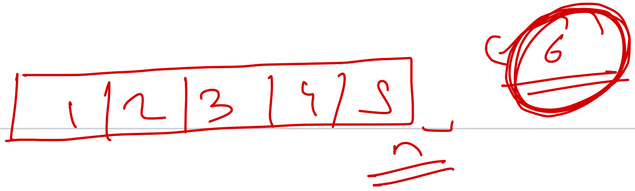
STL ↓ Standard Template Library

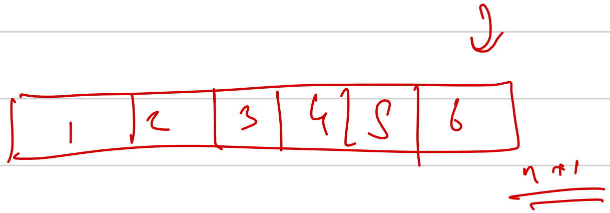
Vectors are dynamic arrays.
↳ compiler and env dependent

array → we can't resize it
↳ size should be defined.

→ Vectors create atleast the amount of space reqd to store the elements. And if demand for space increases it resizes.

→ Also internally vectors use inbuilt arrays for implementation array





generally

everytime vector doubles the size of array.

1, 2, 3, 4, 5, 6, 7, 8, 9, 10 $\longrightarrow$

	size	capacity	no. of op	
[ ] $\rightarrow$	0	0		
[1]	1	1	$\rightarrow$ 1 $\leftarrow$	1
[1, 2]	2	2	$\rightarrow$ 2 $\rightarrow (1+2^1)$ $\leftarrow$	2
[1, 2, 3, 4]	3	4	$\rightarrow$ 3 $\rightarrow (1+2^1)$ $\leftarrow$	3
	4	4	$\rightarrow$ 1 $\leftarrow$	4
[. . . . . . .] $\rightarrow$	5	8	$\rightarrow$ 5 $\rightarrow (1+2^2)$ $\leftarrow$	5
	6	8	$\rightarrow$ 1 $\leftarrow$	6
	7	8	$\rightarrow$ 1 $\leftarrow$	7
	8	8	$\rightarrow$ 1 $\leftarrow$	8
	9	16	$\rightarrow$ 9 $\rightarrow (1+2^3)$ $\leftarrow$	9
			⋮	

avg TC $\rightarrow$ $\dfrac{\text{\# of operation}}{\text{total insertion}}$

amortized algo analysis

$$\frac{(1+2+3+1+5+1+1+1+9+\dots)}{n}$$

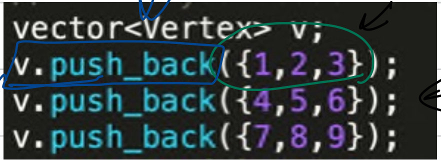
$$\frac{\left(1+(1+2^0)+(1+2^1)+1+(1+2^2)+1+1+1+(1+2^3)\dots\right)}{n}$$

$$\frac{\left(\overbrace{1+1+1+1\dots\dots}^{n}\right)+\left(\overbrace{2^0+2^1+2^2+2^3\dots}^{\log_2 n}\right)}{n}$$

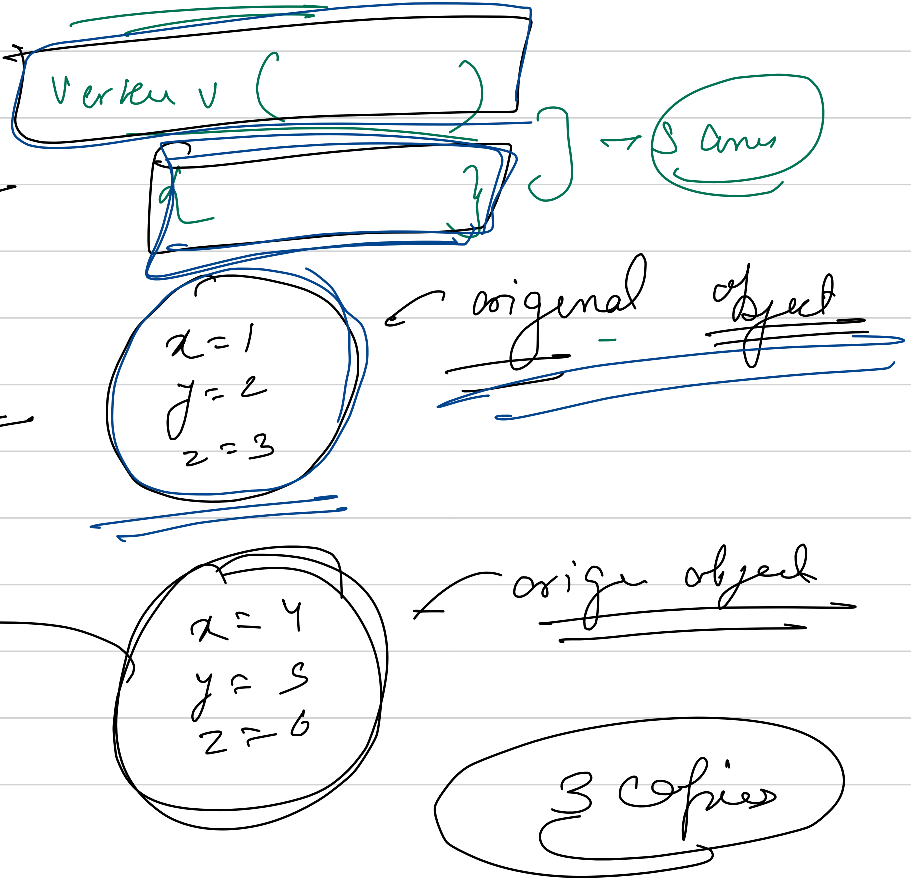
$$\frac{n+1\times\left(2^{\log_2 n}-1\right)}{n}$$

$$\rightarrow\frac{(n+n-1)}{n}=\frac{2n-1}{n}\rightarrow\boxed{\text{constant}}$$

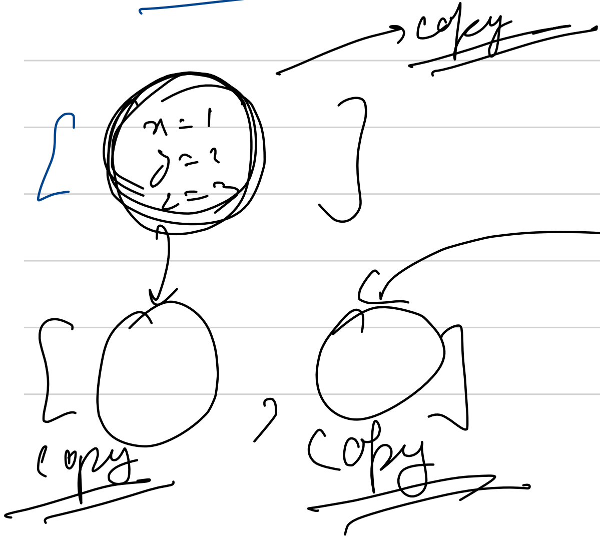
Push back → O(1)



```
vector<Vertex> v;
v.push_back({1,2,3});
v.push_back({4,5,6});
v.push_back({7,8,9});
```



copies the object



emplace_back

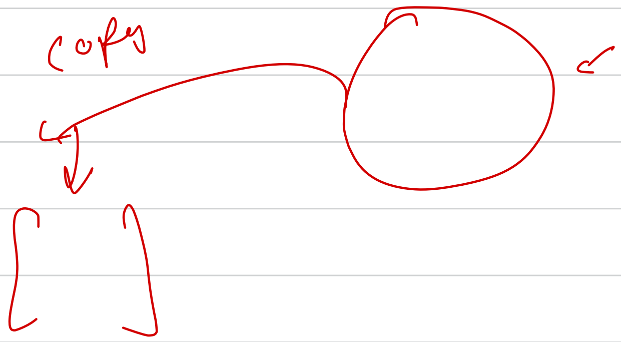
v. emplace_back(1, 2, 3)

{1, 2, 3}

directly creating obj inside vector

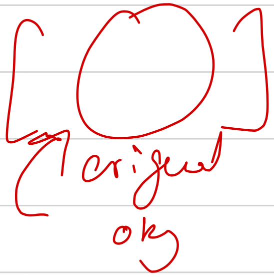
push-back

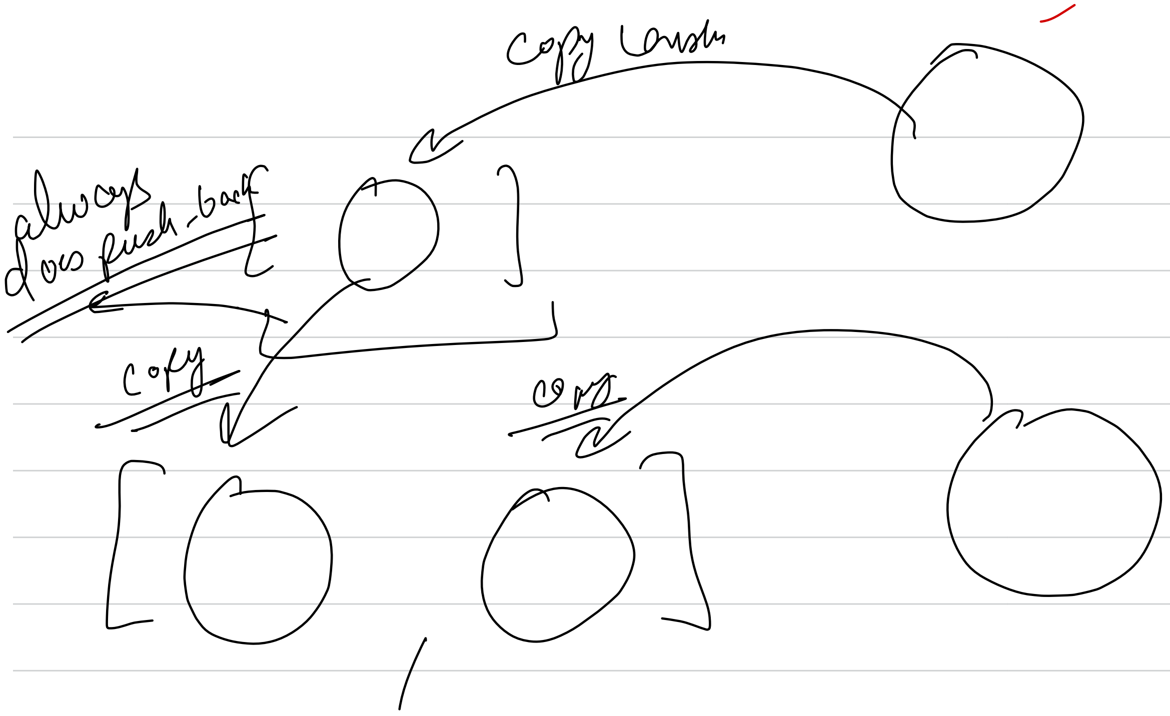
for user defined objects, push-back does call by value.



emplace-back

for user defined objects emplace-back does call by reference.




copy Lonsh
always does push_back
copy
copy