Ch. 17

a) Aggregate Method

~~A~~ To insert 'n' elements, and the cost of i^{th} operation

Case 1 - Here there is no ~~need~~ requirement to assign new memory

1	2	3	4	5	6	7	8	9
$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$

Sequence for the n inserts will be

$$O(n) + O(2n) = O(n)$$

putting the value $O(1)$ in the eqn, we get

$$O(1) + O(2n) = O(1)$$

Therefore amortize runtime is $O(n)$ for the insertion of n element $= O(1)$

~~For~~ ~~ca~~

Case 2 -

Need to allocate new memory.

$i = 2^k + 1$, where $K = 1, 2, 3, 4, 5, 6 \ldots$

to include the capacity ~~it~~ double the size of array.

then we require to allocate new memory.

So to insert n element in new ~~memory~~ array.

Running time $= 2^k + 1$ if $i = 2^{k+1}$, case 1

$= 1$, otherwise, case 2

b) Accounting method

Operation which cause capacity to include are expensive

i	1	2	3	4	5
$t(i)$	1	2	3	1	5

Size is change from 4 to 5 the size gets doubled and numbers are copied from old to new ones.

$\therefore$ No. of Consecutive in $t(i) = \cancel{2^k+1}$

$$= 2^k+1 - (2^{k-1}+1) - 1$$

or

$$2^{k-1} - 1$$

$$= \frac{2^k+1}{2^{k-1}+1} \simeq 2 \quad \text{if } k = \text{large}$$