Hands on 12 HW-17

Halika Kedaveri
1002199711

2. Defining the dynamic table's behaviour

⇒ While inserting an element into the table, if table is full then it doubles in size. Doubling the size takes $O(n)$ time, where n is the current size of the table. After resizing the new element is inserted into the table.

i) Aggregate method

In this method, we calculate the total time for n insertions and divide it by n to find average time per insertion

⇒ So if the original size is 1, after insertion it doubles the size to 2 etc. After k doublings the size is 2^k

$$T_{cost} = 1 + 2 + 4 + 8 + \dots + 2^{k-1}$$
$$= O(2^0) + O(2^1) + O(2^2) + \dots O(2^{k-1})$$

$$\therefore T = O(2^{\log_2 n} - 1) = O(n-1)$$
$$= O(n)$$

⇒ The amortized runtime for inserting n elements is $O(1)$

ii) Accounting method

In this method, cost includes both actual & potential cost.

We will assign each insertion a charge that covers cost of insertion & resizing

charge $O(1)$ → each insertion operation

charge $O(n)$ → each resizing operation.

c_i be the total cost of inserting the i^{th} element.

$$A_i = c_i + c_{i-1}$$

where A_i is the amortized cost per insertion.

$\Rightarrow c_i = O(1)$

$\Rightarrow c_{i-1} = O(n)$

$$\therefore T = \underbrace{n \times O(1)}_{n \text{ insertions}} + \underbrace{\frac{n}{2} \times O(n)}_{\text{resizing operations}}$$

$$= O(n) + O\left(\frac{n^2}{2}\right) = O(n^2)$$

$\Rightarrow T_a = \frac{T}{n} = \frac{O(n)}{n} = O(1) \rightarrow$ amortized cost per operation

$\Rightarrow$ Total time $= O(n)$