Sailokeswari Polarapu
UTA ID : 1002190695

Design and Analysis of Algorithms
Chapter 17
Home work

1). Given a dynamic table that doubles in size when it needs more space. Find the amoritized runtime for inserting n elements.

a). use the aggregate method.

To insert 'n' elements using the aggregate method with the Cost of i^{th} operation

This can be done in two ways

For Case 1:

If we don't take and need to allocate new memory then

1	2	3	4	5	6	7	8	.....
O(1)	O(1)	O(1)	O(1)	O(1)	O(1)	O(1)	O(1)	

So the Sequence of n inserts

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

So, replace O(1) in above example

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

thus the amoritized runtime is, O(n) for inserting n elements is O(1)

For case 2:

If we allocate new memory

$i = 2^k + 1$, $k = 1, 2, 3, \ldots$

to include the Capacity & double the size of array.

then we need to allocate new memory

For inserting the element n in the new array

$$\begin{cases} \text{Running time} = 2^k + 1 & \text{if } i = 2^k + 1, \text{ Case 1} \\ = 1 & \text{otherwise, Case 2} \end{cases}$$

b). Use the accounting method.

Using the accounting method, charge 2 units for each insertion

When the table doubles in size from m to 2m, credit m units

The credit exactly pay for the copy cost of O(m).

Total Credit is $m + 2m + 4m + \cdots$

$n/2 * m = O(n)$

Pseudo Code:

```
initialize table with capacity = 1
    for i=1 to n:
    if table is new table with size 2* current size
    insert element i into table
        initialize Charges = 0
        initialize Credits = 0
    for i=1 to n:
            Charges +=2
            Credits +=m
```

Total Charges $= 2 * n = O(n)$

Total Credits $= m + 2m + \cdots n/2 + m = O(n)$

Cost per insertion = total / n

$= O(n)/n$

$= O(1)$

Therefore, runtime per insertion $= O(1)$

Total time for inserting n elements is O(n).