1002223241

chapter : 17

① Inserting n elements using
@ Aggregate method
→ The table doubles in size when it runs out of space.
→ So, if the original size is 1, after insertion it doubles to size 2, after 2 more insertions, it doubles to size 4 etc.
→ In general, after k doublings, the size is 2^k.

* Pseudo code :
→ initialize table with capacity = 1.
for i = 1 to n;
if if table is full:
new table = create new table with size 2 * current size copy element from old table to new table
table = new table
insert element i into table
let k = log (n+1) - 1
Total cost = O(n) * k
= O(n log n)
Amortize cost per insertion
= O(log n)

Runtime per insertion is
O(log n) //

1002223241

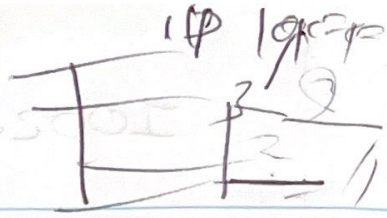
TOTAL time is $O(n) * \log(n+1)$,

(b) 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 $m + 2m + 4m + \ldots \frac{n}{2} m$
$= O(n)$

* Pseudo code:

→ initialize table with capacity = 1
For i = 0 to n:
if table is Full:
newtable = create new table with size 2 * current size
copy elements form old table to new table *, table = newtable
insert element i into table
initialize charges = 0
initialize credits = 0
for i = 1 to n:
charges += 2
if table doubled in size from m to 2m:
credit += m

1002223241



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

TOTAL credits = $n + 2n + \ldots + n/2\, n$

$= O(n)$

Amortized cost per insertion

= Total / n = $O(n)/n$

= 1 //

Runtime per insertion $O(1)$ //

TOTAL time = $O(n)$