HW 17 - Amortized Analysis of Dynamic Array Insertion

Problem:

Given a dynamic table that doubles in size when it needs more space, find the amortized runtime for inserting n elements.

a) Aggregate Method:

Each insertion normally costs 1 unit of time. When the array runs out of space, it is doubled in size and all current elements are copied to the new array.

Let's consider inserting n elements:

- We perform n insertions.
- Every time we double, we copy all the existing elements to the new array.
- This copying cost occurs at sizes 1, 2, 4, 8, ..., up to n.

Total number of element copies:

$1 + 2 + 4 + 8 + ... + n/2 = n - 1$

Total cost = n (for insertions) + (n - 1) (for copying) = $2n - 1$

Amortized cost per insertion = $(2n - 1) / n \sim= 2$

Conclusion: Amortized cost is O(1) per insertion using the aggregate method.

b) Accounting Method:

We charge each insertion with 3 units of cost:

- 1 unit for the actual insertion

- 2 units are stored as "credit" for future copying during resize

When a resize happens:

- We copy all elements to the new array.

- Each element already paid 2 units when it was inserted.

- These credits cover the cost of copying during the resize.

Since all future copies are prepaid, no single operation ends up costing more than the credits it stored.

Conclusion: Amortized cost is O(1) per insertion using the accounting method.