Name: Raoa Faria Karim

UTA ID: 100224803

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.

We will solve this using two methods:

---

a) Using the Aggregate Method

◆ **Basic Idea:**

Each insert normally costs 1 unit. However, when the array is full, it is **resized (doubled)** and all existing elements are **copied** to the new array — this is an expensive operation.

◆ **Observation:**

The number of times the array doubles as we insert n elements is roughly $\log_2 n$.

Every doubling involves copying all current elements. The sizes at which copying happens are:

1, 2, 4, 8, ..., up to n

This forms a geometric series:

Total copies = 1 + 2 + 4 + ... + n/2 = n - 1

◆ **Total Cost:**

- Cost of n insertions = n
- Cost of copying during resizes = n - 1
- **Total = 2n - 1**

Amortized cost per insertion:

2n−1n≈2=O(1)\frac{2n - 1}{n} \approx 2 = O(1)n2n−1≈2=O(1)

---

b) Using the Accounting Method

◆ **Idea:**

We **overcharge** each insert operation with more "credits" than needed so that we can "save up" for future expensive operations (like resizing and copying).

◆ **Strategy:**

Charge **3 units** per insert:

- 1 unit for the actual insert

- 2 units saved as credit

Each time the array is resized, we copy all elements into the new array.

- Suppose we double the array from size k to size 2k.
- We copy k elements.
- But **each of those k elements had 2 credits stored** when they were inserted.
- Those credits pay for the copying!

Amortized cost per insertion:

Since every copy operation is already pre-paid, the amortized cost remains:

O(1) per insertionO(1) \text{ per insertion}O(1) per insertion

---

Final Answer:

Whether using the **aggregate** method or the **accounting** method, the **amortized cost per insertion is O(1)\boxed{O(1)}O(1)**.