Question: Given a dynamic table that doubles in size when it needs more scape. find the amortised runtime for inserting n elements:

a) use the aggregate method

b) use the accounting method

a) Aggregate Method

The aggregate method computes the total cost of nnn operations and then averages it over all nnn operations.

Cost of Insertions:

1. **Normal Insertions**:
    - There are nnn normal insertions. Each takes O(1)O(1)O(1), so the total cost of normal insertions is O(n)O(n)O(n).
2. **Resizing Costs**:
    - The first resize occurs after 1 insertion (cost O(1)O(1)O(1)).
    - The second resize occurs after 2 insertions (cost O(2)O(2)O(2)).
    - The third resize occurs after 4 insertions (cost O(4)O(4)O(4)).
    - The fourth resize occurs after 8 insertions (cost O(8)O(8)O(8)).
    - This pattern continues until we double up to the size nnn.

So the total cost of resizing is:

O(1)+O(2)+O(4)+O(8)+⋯+O(n/2)=O(n−1)≈O(n)O(1) + O(2) + O(4) + O(8) + \dots + O(n/2) = O(n - 1) \approx O(n)O(1)+O(2)+O(4)+O(8)+⋯+O(n/2)=O(n−1)≈O(n)

Total Cost:

- The total cost for nnn insertions is the sum of normal insertions and resizing: O(n)+O(n)=O(n)O(n) + O(n) = O(n)O(n)+O(n)=O(n)
- Since we are inserting nnn elements, the amortized cost per insertion is: O(n)n=O(1)\frac{O(n)}{n} = O(1)nO(n)=O(1)

b) Accounting Method

The accounting method assigns "credits" to each operation in such a way that the total cost of the operations is evenly distributed, allowing us to compute an amortized cost. This method involves charging more for operations that don't take as much time and saving credits for operations that are more expensive.

Let's assign a **credit of 2** to each insertion:

- For a regular insertion, the operation costs O(1)O(1)O(1), but we charge 2 credits.
- The extra credit can be "saved" for when resizing occurs.

Breakdown of Charges:

- For each normal insertion (that doesn't require resizing), we charge 2 credits, and this covers the O(1)O(1)O(1) cost of the insertion and some credit for resizing.
- When the array doubles in size, we incur a copying cost. Specifically, at the time of the resize, we have to copy all the elements in the array, which costs O(k)O(k)O(k) for kkk elements.
- By charging 2 credits for each insertion, we accumulate enough credits to cover the resizing costs, because each doubling requires copying all elements that were previously inserted.

Credits for Resizing:

- Suppose the current array size is 2i2^i2i. At this point, the total number of operations performed up to that point is 2i2^i2i, and we have already charged 2 credits for each of these 2i2^i2i insertions. This gives us a total of 2×2i=2i+12 \times 2^i = 2^{i+1}2×2i=2i+1 credits.
- The cost of the resize is 2i2^i2i, which is less than the 2i+12^{i+1}2i+1 credits accumulated, so the extra credit will be used to pay for the resizing.

Total Cost:

- Each insertion costs 2 credits, and there are nnn insertions, so the total cost is 2n2n2n credits.
- Since each insertion is paid for with 2 credits, the amortized cost per insertion is: 2nn=2\frac{2n}{n} = 2n2n=2