

## Amortized Analysis

We'll start with an example:

expansion of dynamic arrays.

Dynamic array has

size - allocated capacity

num - number of elements  
currently in the array

What is an optimal strategy for  
expanding a dynamic array?

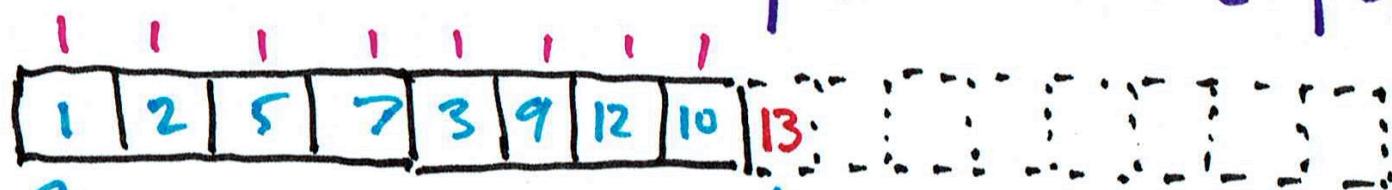
When an insertion would overflow  
the allocated array, we should

1. Allocate an array of size  $2 \cdot \text{size}$ .
2. Copy the elements to the new array.
3. Insert the element into the new array.

Why is this optimal?

First way to think about it...

With every insertion, "bank" some time credits to spend on expansion.



insertion costs 1 time unit; bank 1 additional charge 2 per insertion!

Spend my credits to copy original data to expanded array

So far, so good. I can cover the cost of expansion with stored credits.

Keep going...



Problem! I only have half the credits I need.

We should have charged 3 time units per insertion and banked 2.

For the first expansion we will have overcharged, but we'll be good after that.

2	2	2				
a	b	c	d			

Six credits. More than what we need. Discard extra.

A diagram illustrating a linked list structure. It consists of seven rectangular boxes arranged horizontally, representing nodes. Each box contains a lowercase letter: 'a', 'b', 'c', 'd', 'e', 'f', and 'g'. Above each letter is a small vertical tick mark. The first six boxes ('a' through 'f') are connected by black horizontal lines, forming a chain. The seventh box ('g') is positioned to the right of this chain and is not connected to it. Below the boxes, there are three colored horizontal bars: a pink bar under 'a', a blue bar under 'b', and a green bar under 'c'.

Six credits. Enough to cover copy.  
Spend all my credit.

Diagram illustrating a linked list structure with 15 nodes. The nodes are represented by boxes, and the connections between them are shown by arrows pointing to the right. The first 11 nodes contain the following characters: g, h, i, j, k, l, i, m, n, o, p. The remaining four nodes are empty. A red arrow points from the node containing 'i' to the node containing 'm'. Below the list, the word "loops!" is written in pink.

So what?

If we think of insertion as costing three time units, we can cover all future expansion costs with credits. Expansion costs nothing!

Since "3" is a constant, insertion is still  $O(1)$  irrespective of how many expansions are required.

Another way to say this ...

Let  $n$  be size. The time  
to perform an expansion is  $O(n)$ .

However, I must have done  $n$   
insertions if an expansion  
is necessary.

Divide the cost of expansion by  
the number of insertions:

$$O(n)/n = O(1)$$