PIA Due tonight !!

Amortized Analysis

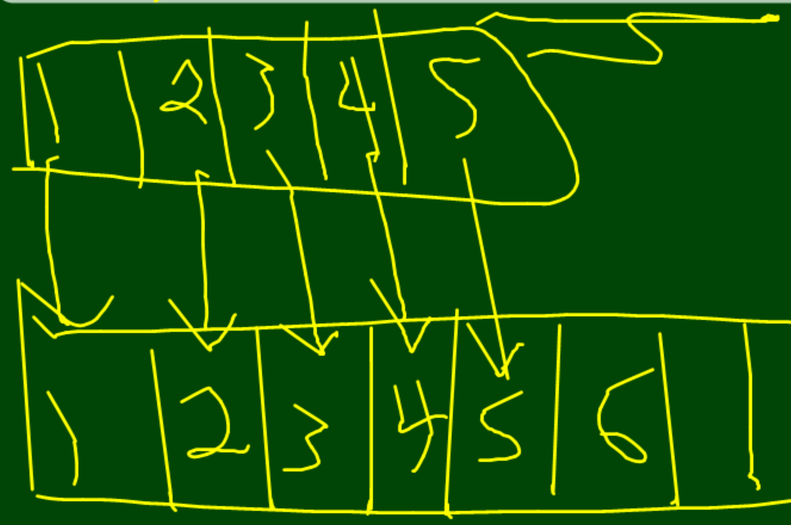
Analyzing `push` for an `ArrayStack`

Best Case

$\Theta(1)$

Worst Case

$\Theta(n)$



Analyzing `push` for an `ArrayStack`

This is where "amortized analysis" comes in. Sometimes, we have a **very rare** expensive operation that we can "charge" to other operations.

Intuition: Rent, Tuition

You pay one big sum for a long period of time, but you can afford it because it happens very rarely.

Back to `ArrayStack`

Say we have a full `Stack` of size n. Then, consider the next n `pushes`:

Analyzing `push` for an `ArrayStack`

What happens if we change our resize rule to each of the following:

- $n \to n + 1$

- $n \to \frac{3n}{2}$

- $n \to 5n$

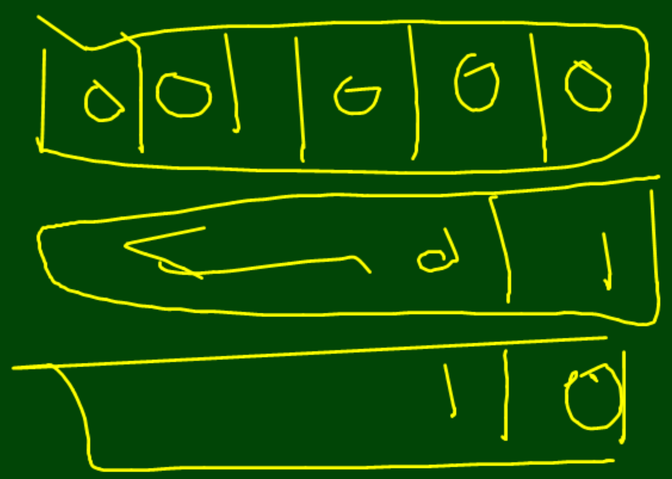
Binary Counter

Time Complexity of A Binary Counter

We would like to analyze an n-bit binary counter with the single method `increment()`. For example,

$$0000 \xrightarrow{1} 0001 \xrightarrow{2} 0010 \xrightarrow{1} 0011 \xrightarrow{3} 0100 \xrightarrow{1} 0101 \xrightarrow{2} 0110 \xrightarrow{1} 0111 \xrightarrow{4} 1000$$

Asymptotic Time Complexity of `increment`



Binary Counter

Time Complexity of A Binary Counter

We would like to analyze an n-bit binary counter with the single method `increment()`. For example,

$$0000 \xrightarrow{1} 0001 \xrightarrow{2} 0010 \xrightarrow{1} 0011 \xrightarrow{3} 0100 \xrightarrow{1} 0101 \xrightarrow{2} 0110 \xrightarrow{1} 0111 \xrightarrow{4} 1000$$

Amortized Time Complexity of `increment`

As always, the first step is to split the operations into "chunks". Where's a good splitting point?

$$\underbrace{10\cdots0}_{n} \rightarrow \underbrace{10\cdots0}_{n+1}$$

Looking at the ones we've already calculated, we get:

n	$n=0$	$n=1$	$n=2$	$n=3$
$T(n)$	1	2	4	8

Time Complexity of A Binary Counter

We would like to analyze an n-bit binary counter with the single method `increment()`.

$$0000 \xrightarrow{1} 0001 \xrightarrow{2} 0010 \xrightarrow{1} 0011 \xrightarrow{3} 0100 \xrightarrow{1} 0101 \xrightarrow{2} 0110 \xrightarrow{1} 0111 \xrightarrow{4} 1000$$

Amortized Time Complexity of `increment`

n	$n=0$	$n=1$	$n=2$	$n=3$
$T(n)$	1	2	4	8

We go by induction on n. Let $P(n)$ be the statement

"incrementing the counter from $\underbrace{10\cdots0}_{n}$ to $\underbrace{10\cdots0}_{n+1}$ changes 2^n bits"

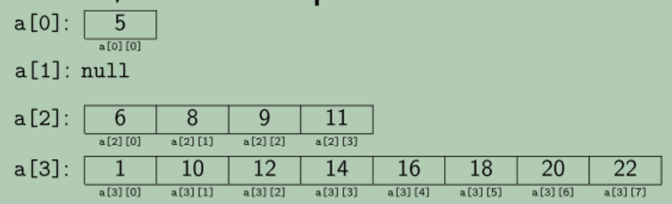
for all $n \in \mathbb{N}$.

A New Data Structure!

Amortized Sorted Array Dictionary

Consider the following data structure:

- We have an array of **sorted** arrays of `int`s. The ith array has size 2^i. So, for example:



- The single method `add(`**val**`)` which works as follows:
  - Make a new array 

  - Until we find a `null` array:
    - If `a[i]` is `null`, set `a[i] = temp`.
    - Otherwise, `temp = merge(a[i], temp); a[i] = null;` and loop.

Let n be # of arrays.

$$m = \sum_{k=0}^{n-1} 2^k = 2^n - 1 = O(2^n)$$