# Amortized Analysis cont'd 0925

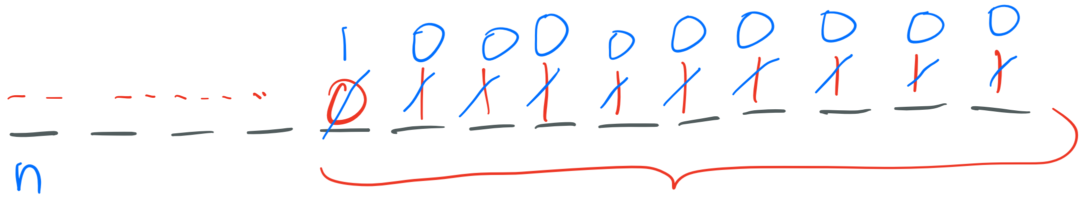
## Amortized Analysis of Binary Counter

init() – sets counter value to 0

inc() – increments counter value by 1

$$\text{init}, \underbrace{\text{inc}, \text{inc}, \text{inc}, \text{inc}, \text{inc}, \dots, \text{inc}}_{m \text{ ops}}$$

Alg: - Start from least sig bit
- flip bits while seeing 1's.
- flip rightmost 0, too.



Analysis: running time ≈ # of bits flipped.

Worst-case # of bitflips in an inc op: $n$

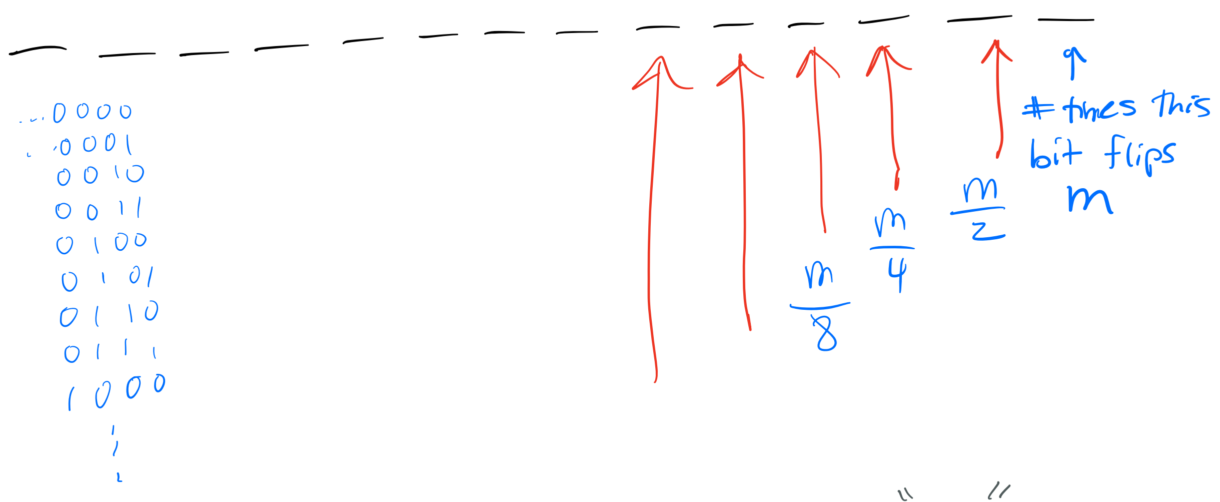
naive analysis: $O(n)$ per operation.

Can we do better when we seek the amortized run time per operation?

YES.

Aggregate method: - add up all running time

- divide by m



Total number of bit-flips in all m "inc" ops:

$$= m + \frac{m}{2} + \frac{m}{4} + \cdots$$

$$= m \cdot \sum_{i=0}^{\lg m} \frac{1}{2^i}$$

$$\leq 2m$$

$$\frac{\text{Total \# of bit flips}}{m} = \frac{2m}{m} = 2$$ bit flips per op, amortized.

# Accounting method

1 credit can pay for a bit flip.

Scheme:

- each "inc" operation will take a max of 2 credits out of the bank, and that will be sufficient to pay for all the work of the m "inc" ops

∴ Amortized running time is $\Theta(1)$ (per op)

To show 2 credits per op will pay for all the work:

inc: - is given 2 credits.

Starts at rightmost bit.

while it sees a 1:

- takes credit that is "on top of" the 1
  uses it to pay for flipping the 1 to 0.

when it sees the rightmost 0:

- pays 1 of the given credits to flip 0→1
- places 1 of the given credits "on top of" the new 1.

## Potential Method

... for you to think about.

Next topic in Amortized Analysis: Union Find