CSCB63 TUT4 WEEK8
andrewyk.leung@mail.utoronto.ca
https://snoopysnipe.github.io/ta/b63w21/

Today's tutorial will be a bit different. I will teach for some time and then you will work together in breakout rooms on a problem about amortized analysis. Afterwards, a representative(s) from each group will present their solution.

Amortized Analysis
Idea: Given a data structure and its operations, we want to find the average performance of each operation in a worst case sequence of n operations

Aggregate Method
Strategy: Determine tight upper bound T(n)
Amortized cost: T(n)/n

Accounting Method
Strategy:
Give each operation \$ $\hat{c}_i$.
Each operation spends some \$ $c_i$ <= \$ $\hat{c}_i$.
If have extra money, save it for the future.
If have deficit, spend past savings.
Show that our savings >= 0 always (usually done by proving savings >= something else >= 0).
Amortized time: $O(\hat{c}_i)$

Potential Method
Strategy:
Let $D_i$ be the state of the data structure after i-th operation.
Define potential function $\Phi: D_i \to R$
Argue $\Phi(D_n) - \Phi(D_0) \geq 0$
Amortized = actual + $\Phi_i - \Phi_{i-1}$

# Binary Counter Increment

Put a $k$-bit number in an array $C$ of $k$ bits. LSB at $C[0]$.
Initially all 0's.

increment():
    $i := 0$
    while $i < C$.length and $C[i] = 1$:
        $C[i] := 0$
        $i := i + 1$
    if $i < C$.length:
        $C[i] := 1$

(For this example: modifying a bit takes $\Theta(1)$ time.)

Up to $k$ bits could be already 1. Increment takes $\Theta(k)$ time worst case. What about a sequence of $m$ increments?

Aggregate Method

ex: $k=6$

| C | | | | | | |
|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 |
| | 1 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 1 | 0 | 0 | 0 | 0 |
| | 1 | 1 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 1 | 0 | 0 | 0 |

increment

$C[i]$ is only modified $\left\lfloor \frac{n}{2^i} \right\rfloor$ times total modifications $\sum_{i=0}^{k-1} \left\lfloor \frac{n}{2^i} \right\rfloor < \sum_{i=0}^{\infty} \frac{n}{2^i}$

$$= n \sum_{i=0}^{\infty} \frac{1}{2^i}$$

[geometric series] $= n \cdot 2$

$\therefore$ $n$ increments is $O(n)$. Amortized $O(1)$.

## Accounting

Increment is \$2

changing a bit uses \$1

Initially we have \$0 saving, 0 1-bits

Increment: change a bit from 0 to 1

↳ spend \$1, save \$1

Change a bit from 1 to 0

↳ spend \$1 from savings

$\Rightarrow$ savings = # of 1-bits $\geq 0$

$\therefore O(2) \Rightarrow O(1)$

## Potential

$\Phi_i$ = # of 1-bits after i increments

$\Phi_n - \Phi_0 \geq 0$ since $\Phi_0 = 0$

After each increment:

$t$ bits changed from 1 to 0

at most 1 bit changed from 0 to 1

actual time $\leq t + 1$ since we change $t$ or $t+1$ bits

$\Phi_i - \Phi_{i-1} \leq -t + 1$ since we lose $t$ 1s and may get back a 1

$$\text{amortized} = \text{actual} + \Phi_i - \Phi_{i-1}$$
$$\leq t+1 - t + 1$$
$$= 2$$

$\therefore O(2) \Rightarrow O(1)$