Review: Dynamic Tables

\[ n \text{ insert operations. When Table full, double size and copy} \]

Amortized cost = average over \( n \) operations

How to compute?

In class: aggregate method, sum total cost, divide by \( n \)

\[
\text{Total} = \frac{1 + 1 + 1 + 1 + 1}{1 \cdot 2 \cdot 4 \cdot 8} = n + \sum_{i=0}^{\log_2 n} 2^i \leq n + 2n = 3n
\]

Thus \( \Theta(1) \) on average

Accounting Method

- Want to bound \( \sum_{i=1}^{n} c_i \) where \( c_i \) is cost of \( i^{\text{th}} \) operation

- Pretend \( i^{\text{th}} \) op "costs" some fictional amount \( C_i \)

- Overcharge for some ops, use surplus to pay for others

Must choose \( C_i \)'s so that \( \sum_{i=1}^{n} C_i \geq \sum_{i=1}^{n} c_i \) \& \( n \) "bank balance always \( 0 \)

Then \( \sum_{i=1}^{n} C_i \) is an upper bound on total cost

Dynamic Table example

- Pretend \( i^{\text{th}} \) insert "costs" \$3, \ ie \( C_i = 3 \)

Pay \$1 for immediate insert cost
Save \$2 for later doubling cost

When table doubles

Pay \$1 to move recent item
Pay \$1 to move older item
Balance returns to \( 0 \) (or almost \( 0 \))

\[ \begin{align*}
& \text{New} \quad 1 \quad 2 \\
& \text{Old} \quad 1 \quad 2 \\
& \text{New} \quad 1 \quad 2 \\
& \text{Old} \quad 1 \quad 2 \\
& \text{New} \quad 1 \quad 2 \\
& \text{Old} \quad 1 \quad 2 \\
& \text{New} \quad 1 \quad 2 \\
& \text{Old} \quad 1 \quad 2 \\
\end{align*} \]

\[
\sum_{i=1}^{\log_2 n} 2^i \leq n + 2n = 3n
\]
why use accounting method?  
Can be more useful than aggregate method when expensive ops unknown.

**Example:** stacks w/multipops.

- **Push, cost:** 1
- **Pop, cost:** 1
- **Multipop(K), pops K items off stack (or whole stack if K > stack size)** cost $\min(K, \text{stack size})$

After $n$ ops, multipop worst-case is $O(n)$  $\Rightarrow$ non-amortized bound of $O(n^2)$

*Let's do better.*

---

**Accounting Method**

- Pretend ith op "costs" $C_i = \begin{cases} 2 & \text{for push op} \\ 0 & \text{for pop} \\ 0 & \text{for multipop} \end{cases}$

Charge $\$2$ for push op

- Pay $\$1$ now
- Save $\$1$ for possible inclusion in pop or multipop later

Easy to see that $\sum_{i=1}^{n} C_i \leq \sum_{i=1}^{n} \hat{C}_i$  $\forall n$

$\uparrow$

Total cost $\leq 2n$

---

**Alternative: Potential Method**

As before, we define fictional "amortized costs" $\hat{C}_i$  $\forall n$. We do this by defining a potential function $\phi$ on stacks, then let

$\hat{C}_i = C_i + \phi(S_i) - \phi(S_{i-1})$

actual cost + potential difference

The potential function should have the properties

- $\phi(S_0) = 0$
- $\phi(S_n) \geq 0$  $\forall n$

These properties imply that $\sum_{i=1}^{n} \hat{C}_i = \sum_{i=1}^{n} (C_i + \phi(S_i) - \phi(S_{i-1})) = \sum_{i=1}^{n} C_i + \phi(S_n) - \phi(S_0)$

$\Rightarrow \sum_{i=1}^{n} \hat{C}_i \geq \sum_{i=1}^{n} C_i$  thus $\sum \hat{C}_i$ upper bounds total cost
Potential method is like a more explicit recipe for applying accounting method

**Stack example**

Let \( \phi(s_i) \) be \# of elements in stack.

Then for any \( i \), \( \phi(s_i) - \phi(s_{i-1}) = \begin{cases} 1 \text{ for push} \\ -1 \text{ for pop} \\ -\min(\text{stack size}, k) \text{ for Multipop} \end{cases} \)

Thus \( \Delta_i = c_i + \phi(s_i) - \phi(s_{i-1}) = \begin{cases} 1+1 \times 2 = 4 \text{ for push} \\ 1-1 = 0 \text{ for pop} \\ \min(\text{size}, k) - \min(\text{size}, k) = 0 \text{ for Multipop} \end{cases} \)

Now use same argument as accounting method!