Amortized Analysis

- Used for analysis of data structures that support multiple operations (INSERT, DELETE, MIN etc).

- The analysis shows that a sequence of $n$ operations takes time $T(n)$, so average cost per operation is $\frac{T(n)}{n}$.

- However, any of the operations, by itself, may take time $\approx \frac{T(n)}{n}$.

Example - Stack

- Two operations: $\text{Push}(x)$, $\text{Pop}(k)$,

  i.e. pushing element $x$ onto stack and popping $k$ elements from stack.

Note: $\text{Pop}(k)$ could take $O(k)$ time.

- However, in a sequence of $n$ operations, each element is pushed and popped at most once, hence total time is $O(n)$.

  - Average cost per operation is $O(1)$!
Potential Function Method

- Let $D_i$ be the state of the data structure after $i$ operations.
- $D_0 = $ initial state.
- $C_i = $ cost of $i$th operation.
- Define $\phi: \{D_0, D_1, D_2, D_3, \ldots \} \rightarrow \mathbb{R}$ to be the "potential function."

Amortized cost of $i$th operation $\hat{C}_i$ is defined as

\[
\hat{C}_i = C_i + \Delta \phi \quad \text{change in potential}
\]

\[
= C_i + (\phi(D_i) - \phi(D_{i-1}))
\]

Theorem. For a sequence of $n$ operations,

- Total cost $\leq$ Total Amortized cost $+ (\phi(D_0) - \phi(D_n))$.
- If (as often is the case) $\phi(D_0) = 0$ and $\phi > 0$,

Total cost $\leq$ Total Amortized cost.
Proof

Total cost = \[ \sum_{i=1}^{n} C_i \]

= \[ \sum_{i=1}^{n} C_i + \phi(D_{i-1}) - \phi(D_i) \]

= \( \left( \sum_{i=1}^{n} C_i \right) + \left( \phi(D_0) - \phi(D_n) \right) \)

= Total Amortized Cost + \( \phi(D_0) - \phi(D_n) \)

Example: Stack (empty at start).

Let \( \phi = \text{# elements on the stack} \).

Clearly \( \phi(\text{Start state}) = 0, \phi \geq 0 \).

Push: Amortized Cost = Actual cost + \( \Delta \phi \)

\[ \Delta \phi = 1 + 1 \]
\[ = 2 \]

Pop(k) Amortized Cost = Actual cost + \( \Delta \phi \)

\[ \Delta \phi = k + (-k) \]
\[ = 0 \]

!!!
Seq. of $n$ operations takes

\[
total \text{ time} \leq \text{ total amortized cost} \leq n \cdot 2.
\]

Example: Binary Counter (starting with 000-0).
- $k$-bit counter.
- INCREMENT (this is the only operation)

\[
\begin{array}{c}
0111 + 1 \\
\hline
1000 - 000.
\end{array}
\]

\[\text{Worst case cost of INCREMENT is } k.\]
However we'll show, using potential function method, that average cost per operation is $O(1)$.

Let $\phi = \# \text{ 1's in the counter}$.

$\phi \text{ (start-state)} = 0, \quad \phi \geq 0$. 

Suppose the counter has seq. of b trailing 1's.

\[
\begin{array}{c}
\text{****01111...1} \\
+ \text{1} \\
\hline
\text{**00000...0}
\end{array}
\]

Amortized cost = Actual cost + \Delta \phi

= (b + 1) + (-b + 1)

= 2.

Alternate proof

Note that the least significant bit changes for every increment. Second " every second " third " every fourth " i-th " every 2^i-th "

\[
\text{Average cost per increment } \leq 2. \quad \text{(as before)}
\]