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 $\gg \frac{T(n)}{n}$.

Example - Stack.

- Two operations: PUSH (x).
  POP (k).

  i.e. PUSHING element x onto stack and POPPING k elements from stack.

Note: - 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 $O(n)$.

$\therefore$ Averge 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 $\phi : \{D_0, D_1, D_2, D_3, \ldots\} \to \mathbb{R}$ 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 $\phi \geq 0$,

Total cost $\leq$ Total Amortized cost.

Proof

$$\text{Total cost} = \sum_{i=1}^{n} c_i$$

$$= \sum_{i=1}^{n} \hat{c}_i + \underbrace{\phi(D_{i-1}) - \phi(D_i)}_{\text{telescoping sum}}$$

$$= \left(\sum_{i=1}^{n} \hat{c}_i\right) + \left(\phi(D_0) - \phi(D_n)\right)$$

$$= \text{Total Amortized cost} + \left(\phi(D_0) - \phi(D_n)\right).$$

$\blacksquare$

Example. Stack. (empty at start).

Let ϕ = # elements on the stack.

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

Push: Amortized cost = Actual cost + $\Delta\phi$

$= 1 + 1$

$= 2$.

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

$= k + (-k)$

$= 0$!!!

$\therefore$ Seq. of n operations takes

total time $\leq$ total amortized cost

$\leq n \cdot 2$.

$\boxed{\text{////}}$

<u>Example</u>: Binary Counter (starting with 000..0).

- k-bit counter.
- INCREMENT (this is the only operation).

$$
\begin{array}{r}
0111 \cdots 111 \\
+ \qquad\qquad 1 \\
\hline
1000 \cdots 000
\end{array}
$$

$\therefore$ worst case cost of INCREMENT is k.

However we'll show, using potential function method, that average cost per operation is $O(1)$.

Let ϕ = # 1's in the counter.

ϕ (start-state) $= 0$, $\phi \geq 0$.

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

$$
\begin{array}{r}
*****0\overbrace{1111\cdots1}^{b} \\
+ \qquad\qquad\qquad 1 \\
\hline
*****1\,0000\cdots0
\end{array}
$$

$\therefore$ 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-1}th "

$\therefore$ Averge cost per increment $= 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots$

$$\leq 2. \quad \text{(as before)}$$