Amortized Analysis

sequence of operations on a data structure

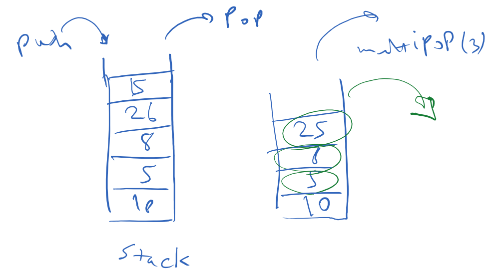
↳ what's average cost per operation

example: Stack with multipop operation

push 10, push 5, ~ 8, 26, 15

pop → 15

multipop (3) → 26, 8, 5




```
MULTIPOP(S, k)
  while S is not empty and k > 0
      POP(S)
      k = k - 1
```

Analysis:

PUSH → $O(1)$ each → sequence of n ops. $O(n)$

Pop → $O(1)$ each → $O(n)$ sequence of n ops

multipop → # PoP operations

iterations = min ($\underbrace{\text{size of } S}_{s}$, k)

Cost of one multipop = min(s, k) → worst case $O(n)$

↳ sequence of n ops. : worst case $O(n^2)$

Aggregate analysis

In a sequence of n operations

↳ $\leq n$ #PUSHes ⟹ $\leq n$ PoPs directly or indirectly via Multipop

total cost = $O(n)$

⟹ average cost for one operation = $O(1)$

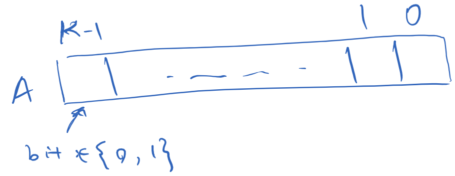
General approach of aggregate analysis:

sequence of n operations $\longrightarrow T(n)$

amortized cost $\longrightarrow \frac{T(n)}{n}$

example: Binary Counter

k-bit counter $A[0 \ldots k-1]$



value of counter $= \sum_{i=0}^{k-1} A[i] \cdot 2^i$

INCREMENT(A, k)
$i = 0$
while $i < k$ and $A[i] == 1$
$A[i] = 0$
$i = i + 1$
if $i < k$
$A[i] = 1$

inc ↓

value	A
0	000 ← bit flip
1	001
2	010
3	011
4	100
5	101
6	110
7	111
0	000

- cost of increment $= \Theta(\#$ of bits flipped$) \longrightarrow$ worst case $O(k)$

- sequence of n increments $\longrightarrow$ worst case $O(nk)$

bit	how often flips	sequence of n inc.
0	every time	n
1	every other time	$\lfloor n/2 \rfloor$
2	$1/4$ the time	$\lfloor n/4 \rfloor$
⋮	⋮	
i	$1/2^i$ the time	$\lfloor n/2^i \rfloor$
	⋮	
$i \geq k$	never	0

Total cost $= \sum_{i=0}^{k-1} \lfloor n/2^i \rfloor$

$\sum_{i=0}^{k-1} \frac{1}{2^i} = 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} \leq 2$

$$\text{Total cost} = \sum_{i=0}^{k-1} \lfloor n/2^i \rfloor$$

$$= n \sum_{i=0}^{k-1} \lfloor 1/2^i \rfloor$$

$$< n \left(\frac{1}{1-(1/2)} \right)$$

$$= 2n$$

$$\sum_{i=0}^{k-1} \frac{1}{2^i} = 1 + 1/2 + 1/4 + 1/8 \leq 2$$

total cost $= O(n)$

amortized cost per operation $= O(1)$

Accounting Method

Intentionally assign different charges to different operations.

compared to actual cost $\begin{cases} < \\ > \end{cases}$

Amortized cost = amount we charge

goal: total amortized cost $\geq$ total actual cost

↳ upper bound for amortized cost will work for actual cost too

per op: amortized cost $\geq$ actual cost ⟶ store diff on objects in data structure as credit

" < " ⟶ use previous credit to pay for diff.

c_i = actual cost for ith op

$\hat{c}_i$ = amortized " "

$$\sum_{i=1}^{n} \hat{c}_i \geq \sum_{i=1}^{n} c_i$$

note: $\sum \hat{c}_i - \sum c_i \geq 0$ invariant

Stack example:

ops.	c_i (actual)	$\hat{c}_i$ (amortized cost)
PUSH	1	2
POP	1	0
MULTIPOP	min (K, S)	0

PUSH → $\hat{c} = 2$ → \$2 { use \$1 to pay for PUSH
store \$1 as credit → to be used for POP/MULTIPOP of that item

total amortized cost of n operation = O(n)

Binary Counter example:

charge \$2 to set a bit to 1
→ { Spend \$1 for setting a bit to 1
Store \$1 as credit for when resetting that bit to 0

amortized cost for increment ≤ 2

potential method:

data structure D

D_i : data structure after ith operation

D_0 , initial data structure

c_i , $\hat{c}_i$

potential function (Φ) $\Phi : D_i \longrightarrow \mathbb{R}$

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$$

$$= c_i + \Delta \Phi(D_i)$$

→ increase in potential

n

Total amortized cost $= \sum_{i=1} \hat{c}_i$

$$= \sum_{i=1}^{n} \left(c_i + \Phi(D_i) - \Phi(D_{i-1}) \right)$$

$$= \sum_{i=1}^{n} c_i + \underbrace{\Phi(D_n) - \Phi(D_0)}_{\geq 0}$$

stack example:

operations	actual cost c_i	$\Delta\Phi = \Phi(D_i) - \Phi(D_{i-1})$	$\hat{c}_i$
push	1	$(s+1) - s = 1$	$1+1=2$
pop	1	$(s-1) - s = -1$	$1-1=0$
multipop	$\min(k,s)$	$(s - \min(k,s)) - s = -\min(k,s)$	$\min(k,s) - \min(k,s) = 0$

Consider Φ = # items on stack (s)

binary counter example:

- Consider Φ = # of 1's after ith increment $= b_i$
- assume that ith operation resets t_i bits to 0

$c_i \leq \underset{\text{resets}}{t_i} + \underset{\text{set}}{1}$

$$\begin{cases} b_i = 0 \longrightarrow b_{i-1} = k = t_i \Longrightarrow b_i = b_{i-1} - t_i \\ b_i > 0 \longrightarrow b_i = b_{i-1} - t_i + 1 \end{cases}$$

$$b_i \leq b_{i-1} - t_i + 1$$

$$\Delta\Phi(D_i) = b_i - b_{i-1}$$

$$\leq b_{i-1} - t_i + 1 - b_{i-1}$$

$$= 1 - t_i$$

$$\hat{c}_i = c_i + \Delta\Phi(D_i)$$

$$\leq (t_i + 1) + (1 - t_i)$$

$$= 2$$

$\hat{c}_i \leq 2 \longrightarrow$ amortized cost of n operations $= O(n)$

check out example of dynamic tables