ITMO

(L-7)

Page No.
Date: / /

O(n) O(n)
↳ loose ↳ tight
→ (saving today for future)

Amortised analysis → expected time required to perform a sequence of DS operations avgd over all (n) operations. Performed. Avg cost of operation may be small when avgd over n operation even then a single operation is expensive. AA guarantees the Avg performance of each operation in worst case

① Aggregate Method. → TT/n
② Accounting Method. → used prepaid (cost involved each item)
③ Potential Method. ↳ cost of one can be used by other (low cost)

↳ ex-) stack operations.

```
multiPop(S,k)
{ while(!S.empty() && k!=0)
    { S.pop()
      k--;
    }
```

Pop = O(1)
Push = O(1)

↳ O(min(S,k))
↳ size of stack

worst case = O(n)

n operations (push, pop, multipop).
in worst → O(n)
worst case → O(n²)

AC ⇒ O(n)/n = O(1)

In aggregate analysis every opn is having the same amortized cost, here we determine the upperbound $T(n)$ on total cost of sequence of n-operations, then the avg cost per operation will be $T(n)/n$.
↳ Amt cost for every operation.

* k-bit binary counter

$A[k-1]$ ... $A[0]$
0000
0001
0010
0011
0100

```
i ← 0
while( i < length[A] and A[i] == 1)
    do A[i] ← 0
       i ← i+1
if i < length[A]
    A[i] = 1
```

O(k) → TC

n time worst complexity O(nk)

$\sum_{i=1}^{k-1} \frac{n}{2^i}$ → TC over all operation

$= n$

$O(n)$ → TC over n operations

Am.

In this method we assign diff. charges to diff operations with some op. charged more or less, than they actually cost. The amount we charge for op is its amortised cost, when Amortised cost > Actual cost, diff is assigned to the object in form of credit. That credit can later be used for the operation where Amortised cost is less than actual cost of operation. Thus the total Amortised cost of sequence of n operations must be upper bound of total actual cost of sequence.

Let actual cost of i^{th} operation c_i
and charged cost $\hat{c}_i$

Then

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

$$AC = \hat{c}_i - c_i$$

Push = 1 Pop = 1 multipop = O(N, size)

Let's assign amortised cost

Push = 2 pop = 0 multipop = 0

it should not be (-ve).

+2 / 1 Pop + push

here, if we push an element in the stack, we are using one unit of cost, and remaining 1 unit is stored in form of credit to it. The credit then can be used as prepayment for cost of pop operation. Since every element is having credit with it, so amount of ~~target~~ credit is non-negative.

So, for any sequence of n (operations) total amortised cost is upper bound on actual cost

$= O(n)$ only.

L-9

		A. cost	
increment	$0 \to 1$	(2)	1 used 1 credit stored with object
	$1 \to 0$	(0)	

(3) Potential Method

credit ≡ Potential energy
↳ store with DS.
store with object

In potential method prepaid work is stored in form of PE that can be released to pay for future operations.

Ans

Initial DS = D_0

n operations are performed

C_i → actual cost of ith operation

D_i → DS after ith operation

Φ → Potential function that maps each DS D_i to some real values which represents the potential associated with D_i.

$$\hat{C}_i = C_i + [\Phi(D_i) - \Phi(D_{i-1})]$$

(after, before; increase in potential → non-ve) ≥ 0

for n operations →

$$\sum_{i=1}^{n} \hat{C}_i = \sum_{i=1}^{n} C_i + [\Phi(D_n) - \Phi(D_0)]$$

– baki cancel out ho jayenge.

Stack → potential = no. of ele in stack

Since stack can't be negative, so stack D_i for ith op is non-negative

ith operation on stack →

$$\Phi(D_i) - \Phi(D_{i-1}) = k+1-k$$

(Potential)

Push = 1 + 1

Pop = 0 (1 + (-1)).

increment.

Let us suppose potential after ith op is B_i which indicates no of 1's on counter after ith operation.

Actual cost C_i → t_{i+1} → no of 1 → req for resetting the bit, and t_{i+1} setting the bit 1

01111 → before (b_{i-1})
01000 → after (b_i)

1 of 1 handle

setting bit to 1

Suppose ith op resets the t_i bit, so actual cost of op is $t_i + 1$

$$b_i = b_{i-1} - t_i + 1$$

$$\Phi = \underbrace{(b_{i-1} - t_i + 1)}_{\text{after}} - \underbrace{(b_{i-1})}_{\text{before}}$$

$$\Phi = 1 - t_i$$

$$AC = t_i + 1 + (1 - t_i)$$

$$= 2$$