ALGORITHM
THE ART OF THOUGHT

By

Prof.(Dr.) Subhasish Majumder,HOD
Department of Computer Science and Engineering
Heritage Institute of Technology, Kolkata

AMORTIZED ANALYSIS

- Time required to perform a sequence of operations is averaged over all the operations(op.)

- Amortized Analysis(AA) can be used to show that the average cost of an operation is low though a single operation may be expensive

- AA is different from Average-Case analysis as no probability is involved.

   - It guarantees the average performance of each operation in the Worst- Case

AMORTIZED ANALYSIS *cont.*

Three Major Techniques are used -

1. Aggregate Method : T(n)/x
2. Accounting Method :
    - Each type of operation may have different Amortized Cost.
3. Potential Method :
    - Define potential associated with the data structure
    - Both 2 and 3 has a concept of prepaid credit.

AGGREGATE METHOD

- Let a sequence of n operations take T(n) time in Worst –Case.
- So Amortized Cost=T(n)/n.
- This Amortized Cost will apply for each operation whatever may be the type of operations.

AGGREGATE METHOD *cont.*...

Example :

Enhance PUSH(S,X) and POP(S) by Multi pop(S,K)

<u>Multipop(S,K)</u>

1. While not Stack-Empty(S) & K!=0
2. Do POP(S)
3. K□K-1

What is the running time of Multi pop(S,K) in a stack of S objects?

□ min(S,K) is the cost

AGGREGATE METHOD *cont.*...

Let us analyse a sequence of n PUSH,POP,Multipop operations.

- ❖Worst Case cost of Multipop is O(n),since the stack size is n.
- ❖So, the Worst Case for n operations is $O(n^2)$ but this is not tight.
- ❖Any sequence of n PUSH,POP,MultiPop operation can cost at most O(n).
  - The no. of times pop can be called on a non-empty stack(including those inside Multipop) is at most the no. of PUSH operations which is O(n).
  - A.Cost O(n)/n=O(1).

AGGREGATE METHOD *cont.* ...

Incrementing a Binary Counter

Array A[0….k-1] of length k bits.

$X = \sum_{i=0}^{k-1} A[i].2^i$; LSB=0, MSB=k-1

To add 1 (modulo 2^k) to the counter, we use the procedure:

<u>INCREMENT (A)</u>

1. i←0
2. While i<length[A] & A[i]=1
3. Do A[i]←0 /*RESET*/

AGGREGATE METHOD *cont.*...

1. i□i+1
2. If [i]<length[A] THEN
3. A[i]□1 /*SET*/

❖ Same Algorithm as hardware ripple carry adder

❖ Cost of each increment operation is linear to the no. of bits flipped.

AGGREGATE METHOD *cont.* …

- Value A(2)A(1)A(0) Total Cost
- 0 00000 0
- 1 00001 1
- 2 00010 3
- 3 00011 4
- 4 00100 7
- 5 00101 8
- 6 00110 10
- 7 00111 11
- 8 01000 15

AGGREGATE METHOD *cont.…*

AGGREGATE METHOD *cont.*...

AGGREGATE METHOD *cont.*...

Some Problems

- If Multipush is included, will it remain O(1)?
  - Answer: No. why?

- Show if DECREMENT is there in K bit counter, Ø(n.k) time is required for n operation
  - Hint: Consecutive Increment-Decrement for all 1's.

$$C(i)=1 \quad \text{if } i=2^k,\ k \in Z^+$$
$$= 1 \ , \text{Otherwise}$$

A.cost = ?

AGGREGATE METHOD *cont.*…

Accounting Method

- Assign different charges to different operations.
- Some operations charges more or less than the actual cost
- When an operation is amortized exceed the actual cost the difference is assigned to credit
- Amortized cost can be split operation into actual cost and credit.
- Credit can either be deposited or used up but total credit can never be negative.

Accounting Method *cont.*…

<u>Stack Operation</u>

Actual Cost – PUSH 1

- POP (1,0)

- Multipop min(S,K)

S is the stack size.

<u>Amortized Cost</u>

-PUSH 2

-POP 0

-Multipop 0

since stack has non empty elements , credit is non negative

Accounting Method *cont.*...

- So the total Amortized cost is upper bound to total actual cost.
- So total Amortized. Cost is O(n), therefore the total actual cost is O(n).

Accounting Method *cont.*...

<u>Incrementing Bin Counter</u>

Actual cost is proportional to no. of bits flipped

- charge 2 to set bit to 1.
- charge 0 to reset bit.

So each 1 in the number at any time will be credited by 1 in it.

Now at most 1bit is set in each increment operation, so Amortized cost of an increment operation is at most 2.

Accounting Method *cont.*...

No. of 1's in the counter is never negative.

So credit is always non negative.

- For N increment operation ,

  Total Amortized Cost=O(n)

Potential Method

Potential Method *cont.*...

Potential Method *cont.*...

Potential Method *cont.*...

Potential Method *cont.*...

Potential Method *cont.*...

Potential Method *cont.*...

Potential Method *cont.* ...

TABLE_INSERT(T,x)

1. if size(T)=0
2. Then allocate tab[T] with 1s to t
3. Size[T] ← 1
4. If num[T] = Size[T]
5. Then allocate new table with 2. Size[T] slots
6. Insert all odd items to new.T
7. Free tab[T]
8. tab[T] ← new-Table
9. Size[T] ← z. Size[T]
10. Insert x into tab[T]
11. num[T] ← num[T] +1

Ø(T)=2. num[T] - Size[T]

THANK YOU