COT 6936: Topics in Algorithms

Giri Narasimhan

ECS 254A / EC 2443; Phone: x3748

giri@cs.fiu.edu

http://www.cs.fiu.edu/~giri/teach/COT6936_S10.html

https://online.cis.fiu.edu/portal/course/view.php?id=427

Amortized Analysis

- In amortized analysis, we are looking for the time complexity of a sequence of n operations, instead of the cost of a single operation.
- Cost of a sequence of n operations = n S(n), where S(n) = worst case cost of each of the n operations
- ***Amortized Cost*** = T(n)/n, where T(n) = worst case total cost of the n operations in the sequence.
- Amortized cost can be small even when some operations in that sequence are expensive. Often, the worst case may not occur in every operation. The cost of expensive operations may be 'paid for' by charging to other less expensive operations.

Problem 1: Stack Operations

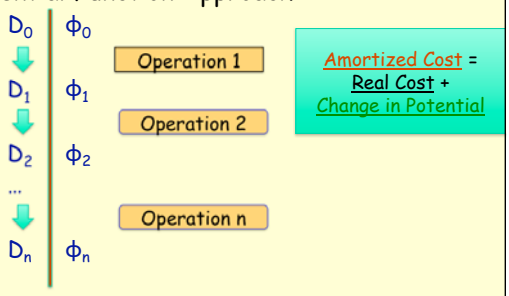
- Data Structure: **Stack**
- Operations:
  - *Push(s,x)* : Push object *x* into stack *s*.
    - Cost: T(push)= O(1).
  - *Pop(s)* : Pop the top object in stack *s*.
    - Cost: T(pop)=O(1).
  - *MultiPop(s,k)* ; Pop the top *k* objects in stack *s*.
    - Cost: T(mp) = O(size(s)) worst case
- *Assumption:* Start with an empty stack
- **Approach 1:** *Simple analysis:* For N operations, maximum size of stack = N. Cost of *MultiPop* = O(N). Total cost of N operations ≤ N x T(mp) = $O(N^2)$. Amortized cost = O(N).

Amortized analysis: Stack Operations

- Intuition: Worst case cannot happen all the time!
- Idea: *Push* operations "pay" for every Pop and *MultiPop* operation.
- Approach 2:
  - Pay 2 dollars for each *Push* operation
    - one to pay for operation itself, and
    - another for "future use" (we pin it to the object on the stack).
  - When we do *Pop* or *MultiPop* operations, instead of paying from our pocket, we pay for operations with extra dollar pinned to objects being popped.
- So total cost of N operations ≤ 2 x N
- Amortized cost = T(N)/N = 2 = O(1). NOT O(N)

Amortized Analysis: Approach 3

- Potential Function Approach



Approach 3: Stack Operations

- Potential Function
  ϕ = # of items in stack
- Analysis

Operation	Real Cost	Change in ϕ	Amortized Cost
Push	1	1	2
Pop	1	-1	0
MultiPop	k	-k	0

- Amortized Cost of any operation ≤ 2
- Total Amortized Cost of N operations = O(N)

Problem 2: Binary Counter

- Data Structure: binary counter b.
- Operations: Inc(b).
  - Cost of Inc(b) = number of bits flipped in the operation.
- What's the total cost of N operations when this counter counts up to integer N?
- *Approach 1: simple analysis*
  - The size of the counter is log(N). The worst case will be that every bit is flipped in an operation, so for N operations, the total cost under the worst case is O(Nlog(N))

Approach 2: Binary Counter

- Intuition: Worst case cannot happen all the time!

000000
000001
000010
000011
000100
000101
000110
000111

Bit 0 flips every time, bit 1 flips every other time, bit 2 flips every fourth time, etc. We can conclude that for bit k, it flips every 2^k time.

So the total bits flipped in N operations, when the counter counts from 1 to N, will be = ?

$$T(N) = \sum_{k=0}^{\log N} \frac{N}{2^k} < N \sum_{k=0}^{\infty} \frac{1}{2^k} = 2N$$

NOT O(log N)

So the amortized cost will be T(N)/N = 2 = O(1).

Approach 3: Binary Counter

- Use recurrence relations
  - For k bit counters, the total cost is
    t(k) = 2 x t(k-1) + 1
  - So for N operations, T(N) = t(log(N)).
    t(k) = ?
  - T(N) can be proved to be bounded by 2N.

Amortized Analysis: Approach 4

- Potential Function
  - ϕ = # of 1's in binary counter
- Analysis
  - Real Cost = k
  - Change in Potential Function = -(k-1) + 1
    = -k + 2
  - Amortized Cost = 2
  - Total Amortized Cost of n increments = O(n)

Amortized Analysis: Potential Method

- For n operations, data structure goes through states: $D_0, D_1, D_2, \ldots, D_n$ with costs $c_1, c_2, \ldots, c_n$
- Define potential function $\Phi(D_i)$: represents potential energy of data structure after i^{th} operation.
- The amortized cost of i^{th} operation is defined by:

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

- The total amortized cost is

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{N} (c_i + \Phi(D_i) - \Phi(D_{i-1})) = \Phi(D_n) - \Phi(D_0) + \sum_{i=1}^{n} c_i$$

$$\sum_{i=1}^{n} c_i = -(\Phi(D_n) - \Phi(D_0)) + \sum_{i=1}^{n} \hat{c}_i$$

Recipe for Amortized Analysis

- Design a potential function
- Estimate Real Cost
- Estimate Amortized Cost

Hardest Part