CMPT307: Amortized Analysis

Week 9-3

Xian Qiu

Simon Fraser University

xianq@sfu.ca

Amortized Analysis

consider a sequence of n Push, Pop, Multipop operations on an initial empty stack

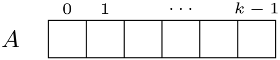
- ▷ Push(S, x) pushes x onto stack S $O(1)$
- ▷ Pop(S) pops the top of S and returns the popped object $O(1)$
- ▷ Multipop(S, k) removes k top objects from S $O(n)$

what is the total running time for the n operations?

- ▷ Pop, Multipop can pop at most the number of Push operations, thus $O(n)$
- ▷ amortized cost of each operation is $\frac{O(n)}{n} = O(1)$

Incrementing Binary Counter

binary counter $A[0..k-1]$ range: $\{0, \ldots, 2^k - 1\}$



$\textsc{Increment}(A)$

```
i = 0;
while i < k and A[i] == 1 do
    A[i] = 0;
    i = i + 1;
if i < k then
    A[i] = 1;
```

each $\textsc{Increment}$ takes $O(k)$ in worst-case

Example

in initial $A[i] = 0$ and consider n INCREMENT operations

$A[3]$	$A[2]$	$A[1]$	$A[0]$	value
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

$A[0]$ flips n times

$A[1]$ flips $\left\lfloor \frac{n}{2} \right\rfloor$ times

$A[2]$ flips $\left\lfloor \frac{n}{2^2} \right\rfloor$ times

$\vdots$

$A[k-1]$ flips $\left\lfloor \frac{n}{2^{k-1}} \right\rfloor$ times

$$T(n) = \sum_{i=0}^{k-1} \left\lfloor \frac{n}{2^i} \right\rfloor < n \sum_{i=0}^{\infty} \frac{1}{2^i} = 2n \quad \Rightarrow \frac{T(n)}{n} = O(1)$$

Remarks

- amortized analysis present a worst-case bound for n operations
- average-case analysis makes assumptions on input, and get "expected" costs
- amortized cost is not as exact as average-case cost
  amortized analysis makes no assumption about distribution
- when considering different operations, it is often easier to use amortized cost

approaches for amortized analysis

- aggregate analysis (already seen)
- accounting method
- potential method

Accounting Method

- assign different charges $\hat{c}_i$ to different operations
- amortized cost of operation i is $\hat{c}_i$
- let c_i be the real cost and require that

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

asymptotic

we may ignore constant factors

Incrementing Binary Counter

$T(n)$ is proportional to the total number of flips

- setting to one: $A[i] = 1$, charge 2 dollars
  - one dollar for cost and one dollar for credit
- recover: $A[i] = 0$, charge 0 dollars
  - pay cost using credit

feasibility: total charge is (asymptotically) no less than $T(n)$

amortized cost: INCREMENT runs "setting to one" at most once

Potential Method

- perform n operations, starting with initial data structure D_0
- apply the ith operation to D_{i-1} yields D_i
- c_i = actual cost of the ith operation
- define potential function $\Phi(D_i) \in \mathbb{R}$ s.t.

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

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

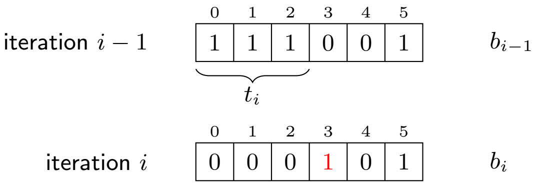
$$= \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$

- may set $\Phi(D_0) = 0$, hence require that $\Phi(D_n) \geq 0$
  often show $\Phi(D_i) \geq 0$ for all i, as n is not known
- $\Phi(D_i) - \Phi(D_{i-1})$ indicates overcharge or undercharge of c_i

$b_i = \#$1s in the counter after the ith INCREMENT $= \Phi(D_i)$

$t_i = \#$ bits to be reset to zero at the ith INCREMENT

- ▷ if $b_i > 0$, then $b_i = b_{i-1} - t_i + 1$



- if $b_i = 0$, reset k bits, hence $b_{i-1} = t_i = k$

iteration $i-1$

0	1	2	3	4	5
1	1	1	1	1	1

iteration i

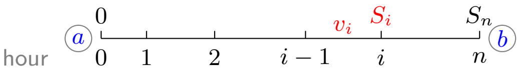
0	1	2	3	4	5
0	0	0	0	0	0

in any case: $\Phi(D_i) - \Phi(D_{i-1}) \le b_{i-1} - t_i + 1 - b_{i-1} = 1 - t_i$

amortized cost $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) \le t_i + 1 + (1 - t_i) = 2$

since $\Phi(D_i) \ge 0$, it holds that $\sum_{i=1}^{n} \hat{c}_i \ge \sum_{i=1}^{n} c_i$

a vehicle takes n hours from a to b

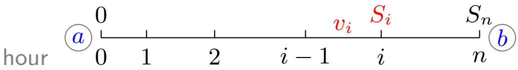


- v_i = average speed during $i-1$ and i
- S_i = distance from a to the position at hour i
- what is the average speed $\bar{v}$?

aggregate analysis: $\bar{v} = \frac{\text{dist}(a,b)}{n} = \frac{S_n}{n}$

accounting method

- if $v_i > \bar{v}$, assign speed $v_i - \bar{v}$ to "slow hours"
- if $v_i < \bar{v}$, get speed $\bar{v} - v_i$ from "fast hours"



potential method: $\Phi_i - \Phi_{i-1}$ = overcharge or undercharge

$$\begin{aligned}\Phi_i &= \text{expected distance} - \text{actual distance at hour } i \\ &= \bar{v} \cdot i - S_i\end{aligned}$$

$$\Phi_i - \Phi_{i-1} = \bar{v} - (S_i - S_{i-1}) = \bar{v} - v_i$$

$$\hat{v} = v_i + \Phi_i - \Phi_{i-1} = \bar{v}$$

but we do not know $\bar{v}$ and S_i

- guess $\bar{v}$ and estimate S_i, then define potential function
- show that Φ_i works well