Tutorial

October 29, 2013

Tutorial

Tutorial

Amortized Analysis

Adversary Argument

Other Problems

Basic

Amortized analysis is a strategy for analyzing
a sequence of operations to show that
the average cost per operation is small,
even though a single operation within the sequence
might be expensive.

Key points:

- $\neq$ average-case analysis; no probability here (Quick-Sort, HashTable)
- worst-case analysis; upper-bound
- on operation sequence; not on separate operations
- cheap ops (often) vs. expensive ops (occasionally)

Basic

Methods:

- summation method
  - the op sequence is known and easy to analyze
  - sum and then average
- accounting method
  - impose an extra charge on inexpensive ops and use it to pay for expensive ops later on

Example for Summation Method

Example (Binary Counter)

- start from 0
- Increment
- measure: *bit flip*
- cost sequence: $1, 2, 1, 3, 1, 2, 1, 4, 1, 2, 1, 3, 1, 2, 1, 5, \ldots$
- think globally about each bit:

$$\sum_{i=0}^{\lfloor \log n \rfloor} \lfloor \frac{n}{2^i} \rfloor < n \sum_{i=0}^{\infty} \lfloor \frac{1}{2^i} \rfloor = 2n$$

- average cost per op: 2

Accounting Method

Accounting method:
amortized cost = actual cost + accounting cost

$$\hat{c}_i = c_i + a_i, a_i >=< 0.$$

$$\forall n, \sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \sum_{i=1}^{n} a_i$$

- upper bound: the *total amortized cost* of a sequence of ops must be an upper bound on the *total actual cost* of the sequence

$$\forall n, \sum_{i=1}^{n} \hat{c}_i \geq \sum_{i=1}^{n} c_i \Rightarrow \forall n, \sum_{i=1}^{n} a_i \geq 0$$

- put the accounting cost on specific objects

Example for Accounting Method

Example (Binary Counter)

To show that $\hat{c}_i = 2$:

- low-level bit: $0 \to 1 (c = 2 = 1 : set + 1 : accouting)$
- put the accounting (1) on the bit 1; number of 1s = sum of accounting ≥ 0
- $1 \to 0 (c = 0)$
- Increment $\hat{c}_i = 2 : (0 \to 1) + (1 \to 0)$; *at most* $0 \to 1$

Remarks

- *you cannot say:* cost per operation = 2
- *you should say:* average cost per operation over any operation sequence ≤ 2
- constant vs. variable
- Decrement? how to support Decrement in $O(1)$?

Table Expansion

Example (Table Expansion)

- "double when it is full"
- TABLE-INSERT
- measure: *elementary insert*

- Summation Method:
  a sequence of n TABLE-INSERT
  - $$c_i = \begin{cases} i & \text{if } i-1 \text{ is an exact power of 2} \\ l & \text{o.w.} \end{cases}$$
  - $$\sum_{i=1}^{n} c_i \leq n + \sum_{j=0}^{\lfloor \lg n \rfloor} 2^j < n + 2n = 3n.$$

Table Expansion

Example (Table Expansion)

- "double when it is full"
- TABLE-INSERT
- measure: *elementary insert*

- Accounting Method:
  why $\hat{c}_i = 3$*? why not 2?*
  - check $\hat{c}_i = 2 = 1(insert) + 1(move)$
  - problem: $2 \Rightarrow 4(\text{after expansion}, \sum a_i = 0) \Rightarrow 8$
  - so, $\hat{c}_i = 3 = 1(insert) + 1(\text{move itself}) + 1(\text{give a hand})$
  - Meta-method: trial and error

Insertion Cost (4.2.7)

Example (Insertion Cost (4.2.7))

- INSERT
- measure: create (1; ignore here) + merge ($2m$)

Summation Method:

$$\sum_{i=1}^{\lfloor \lg n \rfloor} \frac{n}{2^i} 2^i = n \lg n \text{ (Errata: not } i = 0 \text{ in class)}$$

Accounting Method:

- $\hat{c}_i = \lg n$
- consider each inserted element
- to check $\sum a_i = \sum_i (\lg n - D_{a_i}) \geq 0$

Union Find

Example (Union Find)

- Make-Set
- wUnion
- cFind: beautiful code (P_{289}, P_{508})
- $\hat{c}$ *are different*
- worst-case

Tutorial

Amortized Analysis

Adversary Argument

Other Problems

Basic

- lower bound

$$T(A) = \max_I T(A, I)$$

$$T(P) = \min_A \max_I T(A, I)$$

- two directions: SORTING

$$n! \Rightarrow n^2(cards) \Rightarrow n \lg n(\text{John von Neumann@1948}) \Rightarrow ???$$

$$n \Rightarrow n \lg n$$

- adversary argument
  - alg vs. adversary
  - *alg says:* I have a dream
  - *adversary says:* pay something necessary
  - adversary strategy: also an alg; How to construct it?
  - finding min and max; finding the second largest; finding median; horse racing; matrix search

Finding the Median

Example (Finding the Median)

- decision tree
  - tree vs. alg
  - path vs. execution on input
  - internal node vs. comparison
  - leaf vs. output
- $\forall$ leaf, x^*: every other keys are comparable to x^*. why?
- critical comparison (first):
  $\forall y, \exists y$ vs. $z : y > z \geq x^*$ or $y < z \leq x^*$
  - path, internal node
  - *alg:* how do I know?
  - *adversary:* no, you don't know. It is my *private classification.*
  - *alg:* so you can cheat!
  - *adversary:* no, look at this path; critical, critical, non-critical ... (example)

Finding the Median

Example (Finding the Median)

Adversary: *to enfore critical comparisons + non-critical operations as many as possible*

- critical: $n-1$
- non-critical:
  - $L; N; S$
  - $\textsc{Compare}(x, y)$ until $|L| = \frac{n-1}{2}$ or $|R| = \frac{n-1}{2}$

$$N, N; N, S; S, N; N, L; L, N;$$

$$L, S; S, L;$$

$$L, L; S, S(\textit{maybe critical}; \text{however no new } L, S)$$

- *alg:* do $|L| = \frac{n-1}{2}$ at least; via $\textsc{Compare}$
- *adversary:* $1L$ per compare at most; all non-critical
- *alg:* at least $\frac{n-1}{2}$ non-critical comparisons

Horse Racing

Example (Horse Racing)

- 25;5;3
- 7 rounds
- adversary argument
  - < 5: why?
  - $= 5$: which is the fastest?
  - $= 6$: to know the fastest, you must run the five first ranked why?
  - $= 6$: which is the second? (a_2, b_1)

Matrix Search

Example (Matrix Search (P_{246}, 5.24))

- $M[n][n]$; row; column
- $n^2 \Rightarrow T(n) = 3T(\frac{n}{2}) + O(1) = n^{\lg 3} \Rightarrow 2n - 1$
- $2n - 1$: check the below-left corner element
- worst case: check the two diagonals $i + j = n - 1, i + j = n$
- adversary $\textsc{Compare}(x, M[i, j])$:

$$i + j \leq n - 1 : x > M[i, j]; i + j > n - 1 : x < M[i, j]$$

- *alg:* at least check every element in two diagonals
- *adversary:* eliminates 1 at most per comparison
- *alg:* at least $2n - 1$ comparisons

Tutorial

Amortized Analysis

Adversary Argument

Other Problems

Other Problems

- 4.2.3: sorted, distinct; $O(\lg n)$; $a_i = i$; binary search; $T(n) = T(\frac{n}{2}) + O(1)$
- 4.2.5: search for two numbers; $x = a + b$;
  extension: sorted;
  Ex: 1,2,3,5,8,9;2,3,4,6,10,12 [proof or counterexample]
- 4.2.6: closed address hashing

$$Q_k = (\frac{1}{n})^k (1 - \frac{1}{n})^{n-k} \binom{n}{k}$$

- 4.1.4,4.1.5,4.1.6: binomial tree
- 2.1.2: $H \leq N - 1$