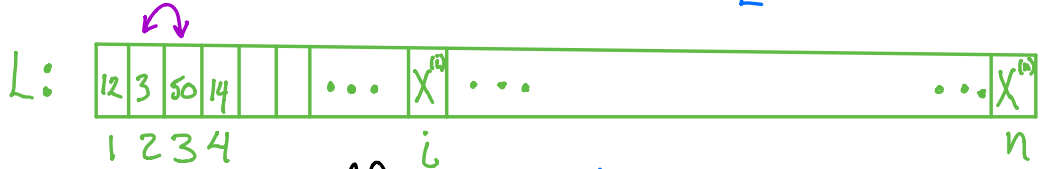
Lecture 05 Amortized Analysis II: Competitive Analysis	L05.1 6.046 2/23/2017

Usually we try to prove something about the absolute running time of an algorithm. Sometimes it is convenient instead to prove something about the running time relative to that of another algorithm. This is especially helpful if the comparison is to the "best possible" algorithm. This is the essence of competitive analysis.

The Problem: Self-Organizing Lists

List L contains n elements

- Only one operation: ACCESS(x) ← key of an element in the list L
  - Find element with key x
  - Which costs $\text{rank}_L(x)$



- ACCESS(14) returns the fourth element
- Transposing 3 & 50 costs "1"

- After every ACCESS we can re-order the list
  - transpose of adjacent elements costs "1"
  - multiple times for larger re-arrangements

GOAL: Choose a sequence of transpositions that optimizes performance of ACCESS by minimizing cost $C_A(S)$ $\forall S$, in on-line manner.

Cost; Algorithm A; Input sequence of keys, S

On-line algorithm: Must respond to sequence of inputs "immediately" as each is presented, before seeing more of input sequence (e.g. game of Tetris).

Off-line algorithm: Can see whole sequence of inputs in advance and make possibly better choices (imagine playing Tetris this way!).

Worst-case analysis: The adversary always chooses the key of the last element ($x^{(n)}$) of the list L.

$$C_A(S) = \Omega(|S| \cdot n)$$

← worst-case, any alg. does poorly, even ignoring cost of any re-ordering

Average-case analysis: Suppose key x is accessed with probability $p(x)$. Then

$$E[C_A(S)] = |S| \sum_{x \in L} p(x) \cdot \text{rank}_L(x)$$

expectation value of the cost of input sequence S

which is minimized when L is sorted in decreasing order of $p(x)$.

INTUITION: Make the most frequent/likely inputs less expensive, while accepting the consequence that other, less frequent/likely inputs will be more expensive. It's still a win.

Heuristic: Keep a count of number of times each element is accessed, and adjust L in decreasing order of count. ← Adversary could still produce poor worst-case performance

Practical: Empirically it is found that the "move-to-front" (MTF) heuristic yields good results.

- After accessing X, move X to head of list L
  - Cost of $\text{rank}_L(x)$ to access
  - Cost of $\text{rank}_L(x) - 1$ to perform transpositions
  - Total cost = $2 \cdot \text{rank}_L(x) - 1$

To analyze performance will use competitive analysis.

COMPETITIVE ANALYSIS (Sleator & Tarjan, 1985)

Definition: An on-line algorithm is α-competitive if there exists a constant k such that for any sequence S of operations

$$C_A(S) \leq \alpha \cdot C_{OPT}(S) + k,$$

where OPT is the optimal off-line algorithm.

Theorem: MTF is 4-competitive for self-organizing lists.

Proof: Let L_i be MTF list after i^{th} access, and let L_i^* be OPT list after i^{th} access.

Let c_i = MTF cost for i^{th} operation
$= 2 \cdot \text{rank}_{L_{i-1}}(x_i) - 1$

Let c_i^* = OPT cost for i^{th} operation
$= \text{rank}_{L_{i-1}^*}(x_i) + t_i$ ← # transpositions in OPT algorithm

Amortized Analysis: Potential Function

Let $\Phi(L_i) = 2 \cdot (\#\text{inversions between } L_i \text{ \& } L_i^*)$
$= 2 \left| \{(x,y): x <_{L_i} y \text{ and } y <_{L_i^*} x\} \right|$

"$\text{rank}_{L_i}(x) < \text{rank}_{L_i}(y)$"

"$\text{rank}_{L_i^*}(y) < \text{rank}_{L_i^*}(x)$"

Example:

L_i: E C A D B

L_i^*: C A B D E

Check all pairs in L_i for inversions in L_i^*:

{ (E,C), (E,A), (E,D), (E,B), (C,A), (C,D), (C,B), (A,D), (A,B), (D,B) }

✓ = inversion
✗ = not
5 inversions
$\Rightarrow \Phi = 10$

takes 5 transpositions to turn L_i into L_i^*

Properties: $\phi(L_0) = 0$ if MTF & OPT start with same list order

$\phi(L_i) \geq 0$ always, because smallest number of inversions is 0.

A transpose creates or destroys 1 inversion, so $\Delta\phi = \pm 2$

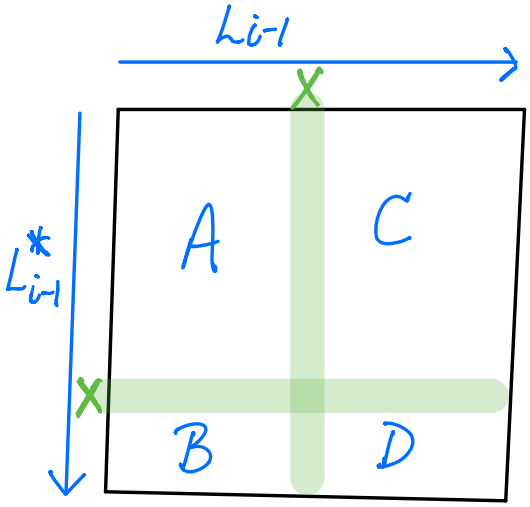
Once we access x in both L_{i-1} & L^*_{i-1}, all other elements fall into one of 4 categories.

A: elements before x in L_{i-1} & L^*_{i-1}

B: elements before x in L_{i-1} but after x in L^*_{i-1}

C: elements after x in L_{i-1} but before x in L^*_{i-1}

D: elements after x in L_{i-1} & L^*_{i-1}



L_{i-1}: A U B | x | C U D

$\text{rank}_{L_{i-1}}(x) = r$

L^*_{i-1}: A U C | x | B U D

$r^* = \text{rank}_{L^*_{i-1}}(x)$

On access: $r = |A| + |B| + 1$ ← when MTF moves x to front, creates $|A|$ inversions and destroys $|B|$ inversions

$r^* = |A| + |C| + 1$ ← each transpose by OPT creates ≤1 inversion

$$\phi(L_i) - \phi(L_{i-1}) \leq 2(|A| - |B| + t_i)$$

First examine per-Access cost for i^{th} Access:

$$\hat{C}_i = C_i + \phi(L_i) - \phi(L_{i-1})$$

($\hat{C}_i$: amortized cost; C_i: actual cost)

$$\leq (2r-1) + 2(|A| - |B| + t_i)$$

$$= 2r - 1 + 2(|A| - (r-1-|A|) + t_i) \qquad r = |A| + |B| + 1$$

$$= 2r - 1 + 4|A| - 2r + 2 + 2t_i$$

$$= 4|A| + 1 + 2t_i$$

$$\leq 4(r^* + t_i) \qquad r^* = |A| + |C| + 1 \geq |A| + 1$$

$$= 4C_i^*$$

Now examine total cost of $|S|$ operations:

$$C_{MTF} = \sum_{i=1}^{|S|} C_i = \sum_{i=1}^{|S|} (\hat{C}_i + \phi(L_{i-1}) - \phi(L_i))$$

(telescoping series)

$$\leq \left(\sum_{i=1}^{|S|} 4C_i^*\right) + \phi(L_0) - \phi(L_{|S|})$$

$$\phi(L_0) = 0 \qquad \phi(L_{|S|}) \geq 0$$

$$\leq 4C_{OPT} \quad \checkmark$$

NOTE:

(1) We never found the optimal algorithm, yet successfully argued about how MTF competes with it.

(2) If cost of transposition is free, then MTF is 2-competitive.

(3) If $L_0 \neq L_0^*$, then $\phi(L_0)$ might be $\Theta(n^2)$ worst case, but $C_{MTF}(S) \leq 4 \cdot C_{OPT}(S) + \Theta(n^2)$ is still 4-competitive, because our competitive analysis treats n as a fixed constant and analyzes costs as $|S| \to \infty$.