List Update Problem

Vamsi Addanki

University of Vienna

vamsi.addanki@univie.ac.at

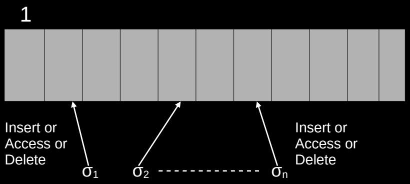
February 8, 2021

Overview

1. Preliminaries
   - Model
   - Potential function
   - Amortized costs
   - competitiveness
2. Deterministic online algorithms
   - Transpose
   - Frequency count
   - Move-to-Front
3. c-competitiveness lower bound

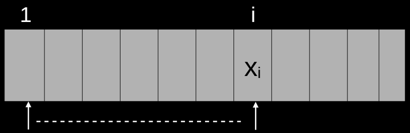
Preliminaries

- Maintain a dictionary
- Unsorted linear list L
- Request sequence σ
    - *access* an item in the list
    - *insert* an item into the list
    - *delete* an item in the list



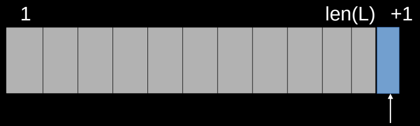
Preliminaries

- *access:* A list update algorithm starts at the front of the list and searches **linearly** through the items until the desired item is found
- cost is i, where i is the position of requested item



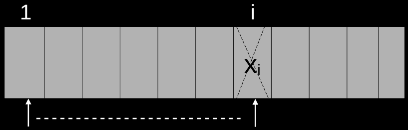
Preliminaries

- *insert:* To insert a new item, the algorithm first **scans the entire list** to verify that the item is not already present and then inserts the item at the **end of the list**
- cost is $len(L) + 1$



Preliminaries

- *delete:* To delete an item, the algorithm scans the list to search for the item and then deletes it
- cost is i, where i is the position of requested item



Preliminaries

- *free exchanges:* **After access or insert** of an item, moving it to the front costs 0.
- *paid exchanges:* At any time, adjacent items in L may be exchanged at cost 1. For example, $swap(L[i], L[i+1])$ costs 1.

Preliminaries

Two models,

- *Static list accessing:* only *access* operation is allowed
- *Dynamic list accessing:* any operation is allowed

Preliminaries

Problem: Minimize the cost of servicing a request sequence σ

- List update algorithm ALG
- Optimal offline algorithm OPT (for the same problem as ALG)
- $ALG(\sigma)$ is the cost incurred by ALG to service request sequence σ

Potential function Φ

- S_{ALG} and S_{OPT} denote the set of possible configurations of ALG and OPT
- $\Phi : \ S_{ALG} \times S_{OPT} \rightarrow \mathbb{R}$

Amortized costs

Let ALG_i denote the cost incurred by ALG during the i^{th} event and denote **amortized cost** of ALG for i^{th} event as a_i

- $a_i = ALG_i + \Phi_i - \Phi_{i-1}$

c-competitive

Definition (c-competitive)

$$\forall len(L),\ \forall\sigma,\ \exists\alpha \mid ALG(\sigma) \leq c \cdot OPT(\sigma) + \alpha$$

In order to prove ALG is c-competitive, it is enough to show[1]

- $\forall i,\ a_i \leq c \cdot OPT_i$
- $\exists b \mid \forall i,\ \Phi_i \geq b$

[1] $a_i = ALG_i + \Phi_i - \Phi_{i-1}$

Deterministic online algorithms

- Transpose
- Frequency count
- Move-to-Front

Transpose

- After accessing or inserting an item at index i, $swap(L[i], L[i-1])$

Frequency count

- Let f_i denote the frequency count of an item k in L
- Whenever i is requested, f_k++;
- Maintain L in nonincreasing order of frequency count

Move-to-Front (MTF)

- Upon a request to item at index i, move it to front of L

Move-to-Front (MTF)

Theorem

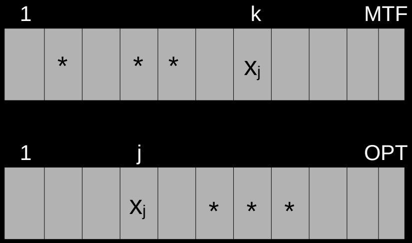
The Move-to-Front algorithm is 2-competitive.

Move-to-Front (MTF)

Proof:

- Consider a request sequence $\sigma = \sigma(1), \sigma(2), ..., \sigma(m)$
- Assume that σ consists of only *access*
- w.l.o.g assume that $L_{MTF} = L_{OPT}$ at the beginning
- Consider a potential function Φ defined as number of inversions in L_{MTF} compared to L_{OPT}
  - *inversion* is a pair $x, y \mid L_{MTF}^{-1}[x] < L_{MTF}^{-1}[y],\ L_{OPT}^{-1}[x] > L_{OPT}^{-1}[y]$
  - $\Phi(t)$ is the potential after serving $\sigma(t)$
  - $\Phi(0) = 0$

Move-to-Front (MTF)



Move-to-Front (MTF)

Proof (continued...):

- for any $t \mid 1 \leq t \leq m$, let x denote the item requested by $\sigma(t)$
- $k = \#$ of items that precede x in L_{MTF} and L_{OPT}
- $l = \#$ of items that precede x in L_{MTF} but follows x in L_{OPT}
- $MTF(t) = k + l + 1$
- $OPT(t) \geq k + 1$
- When MTF serves $\sigma(t)$ and moves x to front, l inversions are destroyed and at most k new inversions are created.
- Thus, $MTF(t) + \Phi(t) - \Phi(t-1) \leq MTF(t) + k - l$

Move-to-Front (MTF)

Proof (continued...):

$$MTF(t) + \Phi(t) - \Phi(t-1) \leq MTF(t) + k - l = 2 \cdot k + 1$$

$$MTF(t) + \Phi(t) - \Phi(t-1) \leq 2 \cdot OPT(t) - 1 \tag{1}$$

$$\sum_{t=1}^{m} MTF(t) + \Phi(m) - \Phi(0) \leq \sum_{t=1}^{m} 2 \cdot OPT(t) - m$$

Move-to-Front (MTF)

Proof (continued...):

$$MTF(\sigma) \leq 2 \cdot OPT(\sigma) - m + \Phi(m) \tag{2}$$

Since $\Phi(m)$ is non-negative, the proof follows from the definition of $c - competitive$ Def. 1

Move-to-Front (MTF)

Proof (continued...): Extension to allow insertion and deletion:

- On an insertion, $MTF(t) = OPT(t) = n + 1$, where $n = len(L_{MTF}) = len(L_{OPT})$ before insertion
  - at most n new inversions are created
- On a deletion, l inversions are removed and no new inversion is created

non-competitiveness of Transpose and FC

Theorem

Both Transpose and Frequency count are not c-competitive for any constant c.

non-competitiveness of Transpose and FC

Proof (Transpose):

- let L_0 denote our list initially
- consider a request sequence that asks for $L_0[len(L)]$ and $L_0[len(L-1)]$ repeatedly
- for each pair of such requests, Transpose incurs a cost of $2 \cdot len(L)$
- OPT would simply move these two items to the front of the list, there by incurring a cost of 3 for each pair of requests
- similarly for a request sequence of size n, the cost ratio, online to opt is, $\frac{n \cdot len(L)}{(\frac{n}{2}-1) \cdot 3 + 2 \cdot len(L)}$

non-competitiveness of Transpose and FC

Proof (Transpose):

- $\lim\limits_{n\to\infty} \frac{n \cdot len(L)}{(\frac{n}{2}-1)\cdot 3 + 2\cdot len(L)} = \frac{2\cdot len(L)}{3}$
- since $len(L)$ has no a priori bound in dyanamic list accessing problem, Transpose is non-competitive

non-competitiveness of Transpose and FC

Proof (FC):

- let $x_1, x_2, ..., x_l$ be the items in the list
- let k be any integer such that $k > l$
- σ is such that, first x_1 is requested k times, then x_2 is requested $k - 1$ times and so on
  - x_i is requested $k + 1 - i$ times
  - notice, x_i are requested in the decreased order of relative frequencies
- upon requesting x_i, FC moves it to i^{th} position and never changes it's position

non-competitiveness of Transpose and FC

Proof (FC):

- $FC(\sigma) = \sum_{i=1}^{l} i \cdot (k+1-i) = \frac{k \cdot l \cdot (l+1)}{2} + \frac{l \cdot (1-l^2)}{3}$
- now, to prove a lower bound, we need some upper bound on $OPT(\sigma)$
- Assume the list is initially in the worst-case for MTF
  - x_i is located at last position when it is first requested (cost of l), moved to front and stays there for $k-i$ subsequent requests

$$MTF(\sigma) \leq \sum_{i=1}^{l} [l + (k-i)] = l \cdot (l+k) - \frac{l \cdot (l+1)}{2}$$

$$\implies \frac{FC(\sigma)}{OPT(\sigma)} \geq \frac{\frac{k \cdot l \cdot (l+1)}{2} + \frac{l \cdot (1-l^2)}{3}}{l \cdot (l+k) - \frac{l \cdot (l+1)}{2}}$$

non-competitiveness of Transpose and FC

Proof (FC):

$$\implies \frac{FC(\sigma)}{OPT(\sigma)} \geq \frac{\frac{k \cdot l \cdot (l+1)}{2} + \frac{l \cdot (1-l^2)}{3}}{l \cdot (l+k) - \frac{l \cdot (l+1)}{2}}$$

$$\implies \lim_{k \to \infty} = \frac{l+1}{2}$$

Since $l = len(L)$, which has no a priori bound, FC is non-competitive

Lower bound for any deterministic online algorithm

Theorem

Let A be a deterministic online algorithm for the list update problem. If A is c-competitive, then $c \geq 2$

Lower bound for any deterministic online algorithm

Proof:

- let $x_1, x_2, ..., x_l$ be the items in the list
- construct a request sequence of size m such that each request is made to the item that is at the last position in L_A
- $A(\sigma) = m \cdot n$
- OPT will first sort items in the nonincreasing order of request frequencies and then serves σ without any exchanges
  - in rearranging OPT incurs a cost of at most $n \cdot \frac{n-1}{2}$
  - in serving σ, OPT incurs a cost of at most $m \cdot \frac{n+1}{2}$
- $OPT(\sigma) \leq \frac{m \cdot (n+1)}{2} + \frac{n \cdot (n-1)}{2}$

Lower bound for any deterministic online algorithm

Proof [2]:

- $OPT(\sigma) \leq \frac{m\cdot(n+1)}{2} + \frac{n\cdot(n-1)}{2}$
- $OPT(\sigma) \leq \frac{m\cdot(n+1)}{2}$ (for $m >> n$)

$$A(\sigma) \geq \frac{2 \cdot n}{n+1} \cdot OPT(\sigma)$$

We conclude the proof, since the RHS in the above inequality approaches 2 as n increases and the bound holds for any n.

[2] $A(\sigma) = m \cdot n$

References

Daniel D. Sleator and Robert E. Tarjan. 1985. Amortized efficiency of list update and paging rules. Commun. ACM 28, 2 (Feb. 1985), 202–208. DOI:https://doi.org/10.1145/2786.2793

Albers S. Competitive online algorithms. BRICS; 1996 Sep. Available online: https://imada.sdu.dk/~joan/dm19/albers.pdf

The End