Competitive Analysis

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 algorithm is "best possible." This is the essence of competitive analysis.

The Problem: Self-organizing lists

List $L$ containing $n$ elements
. only one operation $\to$ Access($x$)
  . Find element with key $x$
  . Cost is rank$_L(x)$

$\begin{array}{ccccccc}
& & & & & & x \rightarrow \\
1 & 2 & 3 & \cdots & & & \\
\end{array}$

- Interesting property - after every access can re-order list
  - transposing adjacent elements for cost of "1"
  - can do multiple times for larger re-arrangements
"On-line" algorithm — must respond to sequence of inputs immediately as each is presented (e.g., the game Tetris)

"Off-line" algorithm — can see the whole sequence of inputs in advance and make potentially better choices (imagine if Tetris worked this way!)

Worst-case analysis for "on-line" self-organizing lists:
- Adversary always accesses the tail (n-th) element in list

\[ C_A(s) = \Omega(|s| \cdot n) \quad \text{in worst case} \]

Average-case analysis
- Let \( p(x) \) be probability of accessing element \( x \)

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

- Can be minimized when elements ordered by decreasing \( p(x) \)

Heuristic: Keep a count of the number of times each element accessed, and adjust \( h \) in decreasing order of count. Adversary could still make this difficult
Practical: Empirically find "move-to-front" (MTF) heuristic yields good results.

- After accessing \( x \), move \( x \) to head of list for total cost = \( 2 \cdot \text{rank}_L(x) - 1 \)
  - \( \text{1} \cdot \text{rank}_L(x) \) to access
  - \( \text{1} \cdot \text{rank}_L(x) - 1 \) to perform transpositions

To show this is a good idea, will do a competitive analysis.

Definition: An on-line algorithm \( A \) is \( \alpha \)-competitive if exists constant \( k \) such that for every sequence \( S \) of operations

\[
C_A(S) \leq \alpha \cdot C_{\text{opt}}(S) + k
\]

where \( C_{\text{opt}} \) is the optimal off-line algorithm "6-d's algorithm"

(if you are going to compare yourself to something, the optimal full-knowledge algorithm is a good one.)
Theorem: MTF is 4-competitive for self-organizing lists

Proof: Let \( L_i \) be list after \( i \)'th access, and let \( L_i^* \) be \( \text{OPT} \) list after \( i \)'th access.

Let \( C_i = \text{MTFs cost for } i \text{'th operation} = 2 \cdot \text{rank}_{L_i} (x) - 1 \)
Let \( C_i^* = \text{OPT cost for } i \text{'th operation} = \text{rank}_{L_i^*} (x) + b_i \)

Amortized analysis: Potential function

\[
\Phi(L_i) = 2 \cdot \# \text{ inversions between } L_i \& L_i^*
\]

\[
= 2 \cdot \left| \{(x,y) : x <_{L_i} y \text{ and } y <_{L_i^*} x \} \right|
\]

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

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

Example:

\( L_i: \overset{\checkmark}{E} C A D B \)

\( L_i^*: C A B D E \)

Check all pairs in \( L_i \) for inversion in \( L_i^* \):

\[
\begin{align*}
& (E, C) \quad (E, A) \quad (E, D) \quad (E, B) \\
& (C, A) \quad (C, D) \quad (C, B) \quad (A, D) \quad (A, B) \quad (D, B)
\end{align*}
\]

\( \checkmark = \text{ inversion} \)
\( \times = \text{ not} \)

\( \Rightarrow 5 \text{ inversions} \Rightarrow \Phi = 10 \)
Properties:
\[ I(L_i) = 0 \] if MTF and OPT start w/same list
\[ I(L_i) \geq 0 \] always, because smallest \# of inversions possible is 0
A transpose creates/destroys 1 inversion, so \( \Delta I = \pm 1 \)

Once we access \( x \) in both \( L_i \) and \( L_i^* \), all other elements fall into one of 4 categories:
A: elements before \( x \) in \( L_i \) and \( L_i^* \)
B: elements before \( x \) in \( L_i \) but after \( x \) in \( L_i^* \)
C: elements after \( x \) in \( L_i \) but before \( x \) in \( L_i^* \)
D: elements after \( x \) in \( L_i \) and \( L_i^* \)

On access \( r = |A| + |B| + 1 \) ← when MTF moves \( x \) to front, creates \( |A| \) inversions & destroys \( |B| \) inversions
\( r^* = |A| + |C| + 1 \) ← each transpose by OPT creates \( \leq 1 \) inversion

\[ I(L_i) - I(L_i^*) \leq 2(|A| - |B| + t_i) \]
\[
\hat{C}_i = C_i + \mathbb{I}(L_i) - \mathbb{I}(L_{i-1})
\]
\[
\leq 2r + 2\left(|A| - |B| + t_i\right) \quad r = |A| + |B| + 1
\]
\[
= 2r + 2\left(|A| - (r-1-|A|) + t_i\right)
\]
\[
= 2r + 4|A| - 2r + 2 + 2t_i
\]
\[
= 4(|A| + 2 + 2t_i) \quad r^* = |A| + |C| + 1 \geq |A| + 1
\]
\[
= 4C_i^*
\]  
Re-arrangement of usual form
\[
C_{\text{MTF}}(S) = \sum_{i=1}^{\lfloor \frac{|S|}{|i|} \rfloor} C_i = \sum_{i=1}^{\lfloor \frac{|S|}{|i|} \rfloor} \left(\hat{C}_i + \mathbb{I}(L_{i-1}) - \mathbb{I}(L_i)\right)
\]
\[
\leq \left(\sum_{i=1}^{\lfloor |S|/|i| \rfloor} 4C_i^*\right) + \mathbb{I}(L_{L_0}) - \mathbb{I}(L_{L_{|S|}})
\]
\[
\leq 4C_{\text{opt}}(S)
\]

Note:
1. We never found the optimal algorithm, yet successfully argued about how MTF competed with it.
2. If cost of transpose is free, then MTF is 2-competitive.
3. If \( L_0 \neq L_0^* \), then \( \mathbb{I}(L_0) \) might be \( \Theta(n^2) \) worst case, but \( C_{\text{MTF}}(S) \leq 4C_{\text{opt}}(S) + \Theta(n^2) \) is still 4-competitive, because \( n^2 \) is constant as \( |S| \to \infty \).