Today: Competitive Analysis
- Self-organizing lists
- Move-To-Front heuristic
- "Competitive" Analysis.

General Theme:
- Often it is good to ask not "How poorly does my algorithm do?" but rather "Is it much worse than other algorithms on the same input?".
- Especially good when algorithm works with partial knowledge
  - e.g. does not know the future
  - e.g. remembers little of the past

- In such cases all algorithms perform poorly in "worst-case"; but some tend to perform poorly only in the worst-case, while others perform poorly everywhere. ... will be good to distinguish.
- Competitive analysis will do this.
**Self-Organizing Lists** (Toy Data Structure) (Real Stuff More Complex...)

- Static database of $N$ elements

- Only operation: $\text{Access}(x)$: returns record with key $x$.

- Data Structure: list $L$ containing all $n$ elements; order allowed to change by swaps of adjacent elements.

- Cost of $\text{Access}(x) = \text{position of } x \text{ in } L$.

$$\triangleq \text{rank}_L(x)$$

**Goal:** Maintain order so as to minimize total cost of accesses for sequence $x_1, x_2, x_3, \ldots, x_m$

**Catch:** Don't know sequence in advance!
Example

- $n = 6$  
  $\text{Set} = \{3, 4, 12, 14, 17, 50\}$

$L = \begin{array}{ccccccc}
12 & 3 & 50 & 14 & 17 & 4 \\
\end{array}$

- $\text{rank}_L(14) = 4$
- $\text{rank}_L(3) = 2 \Rightarrow \text{cost of } \text{Access}(3) = 2$;

- Cost of swapping adjacent elements = 1.

$L' = \begin{array}{ccccccc}
12 & 50 & 3 & 14 & 17 & 4 \\
\end{array}$

$\text{Cost}(L \rightarrow L') = 1$

---

Food For Thought:

How would you use swaps to minimize the cost of $\text{Access}$ on the sequence

$12, 12, 12, 12, 12, \ldots \ 12, 4, 4, 4, \ldots \ 4$
When we consider the task of "optimizing" the access costs, by careful use of swaps, it is natural to ask the question: What information do we have when we make the decision to swap elements?

**OFFLINE VERSION**

- Entire sequence

  12, 12, ... 12, 4, 4, ... 4

  given in advance.

- In this case, for this example at least, it is clear that after the sequence of **Access to [12]** finishes, we should swap **4** till it is at the front of the list, and then all remaining **Accesses** will cost 4 units per **Access**.
Online Version

Here you get to see the sequence one element at a time.

- Access (12);

What would you do to the list?

- Access (14);

What would you do to the list?

- Could swap 4 to the front, but next Access may be to 12.
- Could leave 4 where it is, but next (100) Accesses may be to 4.

Indeed chasing this example carefully, we get...

for every algorithm A,

the exist a sequence \( x_1, \ldots, x_m \)

such that \( \text{cost}_A(x_1, \ldots, x_m) = m \cdot n \)
So ... every online algorithm has the same worst-case performance? 

i.e., all algorithms equally good? bad? 

Doesn’t seem satisfactory.

A heuristic: MOVE-TO-FRONT (MTF) rule.

- Every time you Access \( x \), move \( x \) to front of \( L \) using swaps.

- Cost = \( 2 \cdot \text{rank}_L(x) \).

Seems to work very well "in practice".

Can we justify?

Will do so, by comparing against cost of other (even offline) algorithms on some.
**Competitive Analysis**

Online algorithm $A$ is $\alpha$-competitive if $\exists \, \alpha \geq 1$ s.t. for every sequence $S = (x_1, \ldots, x_m)$,

$$\text{cost}_A(S) \leq \alpha \cdot \text{OPT}(S) + k$$

where $\text{OPT}(S)$ is minimum cost of accessing $S$, even for offline algorithm.

In other words, "$A$ is competitive" implies that for every sequence, it performs not much worse than any other algorithm.

**Theorem:** MTF is $4$-competitive

i.e., on every sequence $S = x_1, x_2, \ldots, x_m$

$$\text{cost}_{\text{MTF}}(S) \leq 4 \cdot \text{OPT} \cdot \text{cost}(S)$$
How to prove this?

<table>
<thead>
<tr>
<th>Time</th>
<th>MTF</th>
<th>OPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>L₀</td>
<td>L₀</td>
</tr>
<tr>
<td>1</td>
<td>rank₁,₀(x₁) L₁</td>
<td>L₁</td>
</tr>
<tr>
<td>2</td>
<td>rank₁,₁(x₂) L₂</td>
<td>L₂</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>i</td>
<td>rank₁,₀(xᵢ) Lᵢ</td>
<td>Lᵢ</td>
</tr>
</tbody>
</table>

Wish to compare

\[2 \text{rank}_{L_{i-1}}(x_i) \text{ vs. } \text{rank}_{L_{i-1}^*}(x_i) + ε_i\]

But for any one i, Lᵢ is maybe large. Rhs maybe 1.
Why? At time \( t-1 \), OPT may swap \( x_i \) to front with many swaps and thus make \( \text{Rank}_{L_i} \( x_i \) = 1."

How to change OPT for this "prospective" move?

Idea: Potential function \( \Phi \).

\( \Phi_i = \text{measures "distance" between } L_i \) and \( L^* \).

**Defn:** \( \tilde{\Phi}_i (L_i, L^*_i) = \Phi_i - I \) where \( I \) is the number of pairs \( (x, y) \) such that \( x \in L_i \), \( y \in L^*_i \), and \( y < x \) in the list \( L^*_i \).

\( \tilde{\Phi}_i = \text{"# inversions"} \).
Intuition:
if $\phi_i$ large: $L_i \approx L_i^*$
very different;
but $Access(x_{i+1})$ will likely
make them "more" similar
if $\phi_i$ small: $L_i \approx L_i^*$
very similar, so $Access(x_{i+1})$
will cost same amount for
$OPT$ and $MTF$.

Example

$\phi(L, L^*) = 3$
Lemma: After \( i^{th} \) operation

\[
2 \cdot \text{rank}_{L_{i-1}}(x_i) + 2 \cdot (\Phi_i - \Phi_{i-1}) \\
\leq 4 \cdot \left[ \text{rank}_{L_{i-1}}(x_i) + \varepsilon_i \right]
\]

Will prove this soon, but why does

Lemma \implies \text{Theorem}?

Note: 1. \( \Phi_0 = 0 \)

2. \( \Phi_i \geq 0 \) \( \forall i \)

3. Have

\( H_i \cdot \text{cost}(x_i) + 2(\Phi_i - \Phi_{i-1}) \leq 4 \cdot \text{OPT}(x_i) \)

Summing over \( i \) ...

\[
\sum_{i=1}^{m} \text{cost}(x_i) + 2 \sum_{i=1}^{m} (\Phi_i - \Phi_{i-1}) \leq 4 \sum_{i=1}^{m} \text{OPT}(x_i)
\]
\[
\text{cost}_{\text{MTF}}(S) + 2 \overline{\Phi}_m - 2 \overline{\Phi}_0 \leq 4 \text{OPT}(S)
\]

\[
\nu \leq 0
\]

\[
\downarrow
\]

\[
\text{cost}_{\text{MTF}}(S) \leq 4 \text{OPT}(S) \quad \text{[Theorem!]} \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad
Claim 1:

\[2 \text{rank}_{L_{i-1}}(x_i) + 2\left(\overline{\Phi}_{i-\frac{1}{2}} \right) \leq 4 \cdot \text{rank}_{L_{i-1}}(x_i)\]

Claim 2:

\[2\left(\overline{\Phi}_{i} - \overline{\Phi}_{i-\frac{1}{2}}\right) \leq 4 \cdot t_i\]

First note that Claim 1 + Claim 2

\[\Rightarrow\] Lemma (just add the inequalities)

So let's prove the claims.

Proof of Claim 2: Obvious .... in fact \( t_i \) swaps in \( \frac{t_i}{2} \) is the only one that inverts by \( t_i \).

\[\Rightarrow \overline{\Phi}_i \leq \overline{\Phi}_{i-\frac{1}{2}} + t_i\]

which is stronger than what we want.
Proof of Claim 1: (finally we're getting to the heart of the analysis).

\[
\begin{align*}
L_{i-1} & \xrightarrow{\square} 1 \xrightarrow{\square} \cdots \xrightarrow{\square} x_i \\
L_i \xrightarrow{\square} 1 \xrightarrow{\square} \cdots \xrightarrow{\square} \cdots \\
\end{align*}
\]

\[
\text{rank}_{L_{i-1}} (x_i) = k \\
\text{rank}_{L_{i-1}}^* (x_i) = l \\
\overline{\Phi}_{i-1} - \overline{\Phi}_{i-1} = ?
\]

\[
\begin{align*}
L_i & \xrightarrow{\square} 1 \xrightarrow{\square} \cdots \\
\end{align*}
\]

* Only inversions that change are of the form \((x_i, y)\)*
<table>
<thead>
<tr>
<th>Location of $y$</th>
<th>Effect on $\Phi_{i-\frac{1}{2}} - \Phi_i$</th>
<th># of such $y$'s</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y &lt; \ell_{i-1}$</td>
<td>$+1$</td>
<td>$\leq \ell$</td>
</tr>
<tr>
<td>$y &lt; L^*_{i-1}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$y &lt; \ell_{i-1}$</td>
<td>$-1$</td>
<td>$\geq \ell_{k-\ell}$</td>
</tr>
</tbody>
</table>

$y > L^*_{i-1}$

$y > \ell_{i-1}$

No effect

Doesn't matter

$y < L^*_{i-1}$

No effect

Doesn't matter

$y < \ell_{i-1}$

No effect

Doesn't matter

$y > \ell_{i-1}$
Thus \[ \Phi_{i-\frac{1}{2}} - \Phi_{i-1} \leq 2l - k \]

\[ \Rightarrow 2k + 2(\Phi_{i-\frac{1}{2}} - \Phi_{i-1}) \leq 2k + 4l - 2k \]

\[ = 4l = 4 \text{rank}_{L_{i-1}} (x_i) \]

\[ \mathbb{Q} (\text{of Claim 1}) \]

Conclusions:

1. Competitive analysis often much more useful than "worst-case" analysis (imagine applying it to the stock market...)

2. Potential functions can help establish competitive bounds.
Addendum

Weak vs. strictly-competitiveness:

In our definition of $\alpha$-competitive, we allowed a constant $k \geq 0$

s.t. for every sequence $S$

$$\text{cost}_A(S) \leq \alpha \cdot \text{cost}_{\text{opt}}(S) + k.$$ 

In the literature, this is also called weak competitiveness.

A stronger requirement would ask for this $k$ to be zero.

If an algorithm satisfies this with $k = 0$, then we call it

strictly competitive.

In lecture, we showed that MTF was strictly 4-competitive.