Lecture 1

6.851: Advanced Data Structures
Prof. Erik Demaine & TA Oren Weimann

Goal: maintain data to support fast queries, maybe fast updates, using little space

Classics:
- linked lists
- binary search trees
- graphs
- strings
- hashing

Advanced view:
- self-adjusting, MTF,
- splay trees, Tango trees
- dynamic connectivity
- suffix trees/trays, wildcards
- perfect, Cuckoo hashing

Advanced models:
- integers: predecessor (like BST), priority Q's, lower
- caching: cache-oblivious *
- temporal: persistence, retroactive
- succinct: \( o(n) \) or even \( \emptyset \) extra space

Requirements:
- attending lectures
- scribing one or two lectures
- lightweight homework (1 page/week)
- research-oriented project (theory, experiment, survey)
Self-adjusting data structures: linked lists

DS: linked list of $n$ elements, say $\{1, 2, ..., n\}$
query: access$(x)$: find node containing elt. $x$
(using only linked list)

Models:

1. worst case: $n$
2. stochastic: independent random choices
   $p_1 + p_2 + ... + p_n = 1$; $p_i = \Pr\{\text{access}(i)\}$
   $\Rightarrow$ store $1 \rightarrow 2 \rightarrow ... \rightarrow n$ s.t. $p_1 \geq p_2 \geq ... \geq p_n$
   $\Rightarrow$ cost $\sum_{i=1}^{n} i \cdot p_i \leftarrow \text{stochastic OPT}$
      but what if we don't know $p_i$'s?
3. general: arbitrary request sequence $x_1, x_2, ..., x_m$
   - define frequency $f_i = \#\text{access}(i)$'s; $p_i = f_i / m$
   - static OPT: $\sum_{i=1}^{n} i \cdot f_i$ where $f_1 \geq f_2 \geq ... \geq f_n$
      (at least as good as stochastic OPT)
   - but dynamic OPT can exploit dependence
   - startup model: arbitrary initial order
      or initially empty; access(new element) $\Rightarrow$ append

3a. one-finger model: access$(x)$ starts at head
    pay 1 per back, forward, swap (with fore neighbor)

3b. const.-finger model: all fingers start at head
    standard pointer copying: $f_1$.fore $= f_2$, etc.
    at the end, DS must be a list
Natural access(x) algorithms:

1. **Frequency Counting (FC):** store & order by f_i's
   - stochastic OPT
   - \( \leq 2 \cdot \text{static OPT} \)
   - \( \neq O(1) \cdot \text{dynamic OPT} \)
   - [Bünger - SICOMP 1979]
   - [Bentley & McGeoch - CACM 1985]
   - [Sleator & Tarjan - CACM 1985]

2. **Transpose:** move x one position to front
   - \( \leq \text{stochastic MTF} \)
   - \( \neq O(1) \cdot \text{static OPT} \)
   - [Rivest - CACM 1976]
   - [Bentley & McGeoch - CACM 1985]
   - \[\text{access: } 1, 2, \ldots, n-1, n, n-2, n, n-2, n, n-2, \ldots\]
   - \[\Rightarrow \text{list: } 1, 2, \ldots, n-1, n-2, n\]
   - Transpose \( \sim n^{3/2} + m \cdot n \)
   - Static OPT \( \sim n^{3/2} + \frac{3}{2} \cdot m \)
   - Roughly \( n \times \)

3. **Move To Front (MTF):** move x all the way front
   - \( \approx \pi/2 \cdot \text{stochastic OPT} \)
   - \( \approx \pi/2 \cdot \text{stochastic OPT} \)
   - \( \leq 2 \cdot \text{static OPT} \)
   - \( \leq O(1) \cdot \text{one-finger OPT} \)
   - \( \neq O(1) \cdot \text{const.-finger OPT} \)
   - For any online strategy
     - \( \sim \sqrt{n} \cdot \log n \)
Sleator & Tarjan cost model: \((\approx \text{one-finger})\)
- pay \(i\) to access \(x\) at position \(i\)
- free to move \(x\) anywhere closer to front
- pay 1 to swap any two adjacent elements

MTF is 2-competitive: [Sleator & Tarjan - CACM 1985]
- run MTF & OPT on same sequence, in parallel, starting from same initial list order
- potential \(\Phi = \#\text{ inversions between MTF & OPT lists}\)
  \[= \# \text{ pairs } (i, j) \text{ where } i \overset{\text{MTF}}{\leq} j \text{ & } j \overset{\text{OPT}}{\leq} i \text{ or vice versa} \]
- access \((x)\): consider \(x\) in MTF list:
  \[
  \begin{array}{cccccc}
  \langle & \langle & \langle & \langle & \langle & x \rangle \\
  \rangle & \rangle & \rangle & \rangle & \rangle & \text{MTF} \\
  k
  \end{array}
  \]
  - label elts. in front of \(x\) in MTF list as \(<\) (same) or \(>\) (inverted) \(x\) in OPT list
  - \(\Delta \Phi\) from MTF\((x)\) = \(#<\)'s - \(#>\)'s
  - actual cost = \(#<\)'s + \(#>\)'s + 1
  \[\Rightarrow \text{amortized cost} = 2 \cdot \text{#<}'s + 1 \leq 2 \cdot \text{OPT}(x) - 1 \]
  \(<\) position of \(x\) in OPT list
  - if OPT pays for swap, \(\Delta \Phi \leq 1\)
  - if OPT makes free swap, \(\Delta \Phi = -1\)
  \[\Rightarrow \text{MTF} \leq 2 \cdot \text{OPT position cost} - m\]
  \[+ \text{OPT paid swaps} - \text{OPT free swaps} \]
  \[< 2 \cdot \text{OPT} - m. \]
\[\square\]
Best one-finger strategies:
- no deterministic algorithm is $\leq 2$-competitive \[\text{[Karp \& Raghavan]}\]
- bit algorithm is $1.75$-competitive in expectation (against oblivious adversary) \[\text{[Reingold, Westbrook, Sleator - Algorithmica 1994]}\]
  - assign random bit to each element, once at beginning
  - upon access, flip bit, and MTF if 1.
- best algorithm is $1.6$-competitive in expectation \[\text{[Albers, von Stengel, Werchner - IPL 1995]}\]
- no algorithm is $<1.5$-competitive

**OPEN:** close gap for randomized one-finger

- offline OPT is NP-hard \[\text{[Ambuehl - ESA 2000]}\]

**OPEN:** approximability of one-finger offline?
Order By Next Request (OBNR): [Munro-ESA 2000]
- offline (omniscient) access algorithm
- access(x): find x, say at position i
continue scan to \[ \lfloor i \rfloor = 2 \log_2 i \]
hyperceiling
Sort these elts. by next request time (and put x at front, say)

Example:

<table>
<thead>
<tr>
<th>x</th>
<th>cost</th>
<th>list</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1 2 3 4 5 6 7 8 ...</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2 1 3 4 5 6 7 8 ...</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3 4 1 2 5 6 7 8 ...</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>4 3 1 2 5 6 7 8 ...</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>5 6 7 8 1 2 3 4 ...</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>6 5 7 8 1 2 3 4 ...</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>7 8 5 6 1 2 3 4 ...</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>8 7 5 6 1 2 3 4 9 ...</td>
</tr>
<tr>
<td>9</td>
<td>16</td>
<td>9 10 11 12 13 14 15 16 1 2 ...</td>
</tr>
</tbody>
</table>

etc.

Total cost per permutation \( \sim n \cdot (\frac{1}{2} \cdot 2 + \frac{1}{4} \cdot 4 + \ldots) \)
\[ = n \log n \]
MTF costs \( \sim n^{2/2} \) \( \Rightarrow \) ratio \( \sim \frac{n}{\log n} \)
Ditto for any online algorithm on some permutation
Munro cost model: (roughly const. fingers)
- pay \(i\) to scan first \(i\) elements
- free to permute these elements arbitrarily
- in OBNR, can actually sort in \(O(i)\) work:

**Invariant:** elements \(2^b+1, 2^b+2, \ldots, 2^{b+1}-1\) sorted

\[\Rightarrow\] sorting = merging blocks of sizes \(2^0, 2^1, \ldots, 2^i\)

**OBNR analysis:** amortized cost \((x) \leq 1 + 4 \lceil \log w(x) \rceil\)

where \(w(x) = \#\) distinct \(y\)'s accessed since last access \((x)\) including \(x\) itself — **working-set bound**

**Proof:** charge cost \(\lceil \log \rceil\) for access \((y)\) to elements in penultimate block, of size \((\lceil \log \rceil+1)/4\)

(special case: \(i=1 \Rightarrow\) just pay cost of 1)

- if element \(x\) in block \(b\) gets charged
  then either \(x\) advances to block \(b+1\)
  or it is followed by \(2^{b+1}\) elements
  \[\Rightarrow\] not charged in block \(b\) before next access \((y)\)

\[\Rightarrow\] \(y\) charged at most once in each block

- \(y\) advances to block \(\lceil \log w(x) \rceil\) before access \((y)\)

\[\Rightarrow\] amortized cost \(4 \lceil \log w(x) \rceil\).

- improve 4 to 2.6641... by changing \(2 \Rightarrow 4.24429\)... [Munro]
- lower bound of \(\Omega(1+\log w(x))\) [Demaine & Harmon 2006]

**Open:** nail/improve constant