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 Ø 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

a.k.a. linear search

DS: linked list of n elements, say $\{1, 2, \dots, n\}$

query: access(x): find node containing elt. x (using only linked list)

Models:

① worst case: n

② stochastic: independent random choices

$p_1 + p_2 + \dots + p_n = 1$; $p_i = \Pr\{\text{access}(i)\}$

$\Rightarrow$ store $\boxed{1} \to \boxed{2} \to \dots \to \boxed{n}$ s.t. $p_1 \geq p_2 \geq \dots \geq p_n$

$\Rightarrow$ cost $\sum_{i=1}^{n} i \cdot p_i$ ← stochastic OPT

— but what if we don't know p_i's?

③ general: arbitrary request sequence $x_1, x_2, \dots, x_m$

- define frequency f_i = #access(i)'s; $p_i = f_i / m$
- static OPT: $\sum_{i=1}^{n} i \cdot f_i$ where $f_1 \geq f_2 \geq \dots \geq f_n$

  (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

- $\sim$ stochastic OPT [Bitner – SICOMP 1979]
- $\le 2 \cdot$ static OPT [Bentley & McGeoch – CACM 1985]
- $\ne O(1) \cdot$ dynamic OPT [Sleator & Tarjan – CACM 1985]

② Transpose: move x one position to front

- $\le$ stochastic MTF [Rivest – CACM 1976]
- $\ne O(1) \cdot$ static OPT [Bentley & McGeoch – CACM 1985]

access: $1, 2, \ldots, n-1, n, n-2, n, n-2, n, n-2, \ldots$

$\Rightarrow$ list: $1, 2, \ldots, n-1, n-2, n$

Transpose $\sim n^2/2 + m \cdot n$

static OPT $\sim n^2/2 + \frac{3}{2} \cdot m$

roughly $n \times$

③ Move To Front (MTF): move x all the way front

- $\ge \pi/2 \cdot$ stochastic OPT [Gonnet, Munro, Suwanda – SICOMP 1981]
- $\le \pi/2 \cdot$ stochastic OPT [Chung, Hajela, Seymour – JCSS 1988]
- $\le 2 \cdot$ static OPT [Bentley & McGeoch – CACM 1985]
- $\le O(1) \cdot$ one-finger OPT [Sleator & Tarjan – CACM 1985]
- $\ne O(1) \cdot$ const.-finger OPT [Munro – ESA 2000]

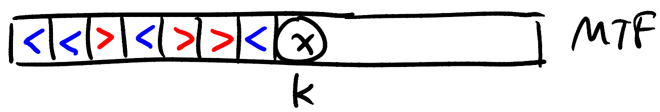
for any online strategy — $n/\lg n \times$

Sleator & Tarjan cost model: ($\approx$ one-finger)

- pay i to access x at position i
- free to move x anywhere closer to front
- pay 1 to swap <u>any</u> 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 Φ = # inversions between MTF & OPT lists
  = # pairs (i, j) where $i <_{MTF} j$ & $j <_{OPT} i$ or vice versa
- access(x): consider x in MTF list:



  - 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

  $\Rightarrow$ amortized cost $= 2 \cdot \underbrace{\#<\text{'s}}_{<\text{ position of } x \text{ in OPT list}} + 1 \le 2 \cdot OPT(x) - 1$
- if OPT pays for swap, $\Delta\Phi \le 1$
- if OPT makes free swap, $\Delta\Phi = -1$

$\Rightarrow$ MTF $\le$ 2 · OPT position cost − m
+ OPT paid swaps − OPT free swaps
< 2 · OPT − m. $\square$

Best one-finger strategies:

- no deterministic algorithm is <2-competitive [Karp & Raghavan]
- Bit algorithm is 1.75-competitive in expectation (against oblivious adversary) [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 [Albers, von Stengel, Werchner — IPL 1995]
- no algorithm is <1.5-competitive

$\boxed{\text{OPEN:}}$ close gap for randomized one-finger

- offline OPT is NP-hard [Ambuehl — ESA 2000]

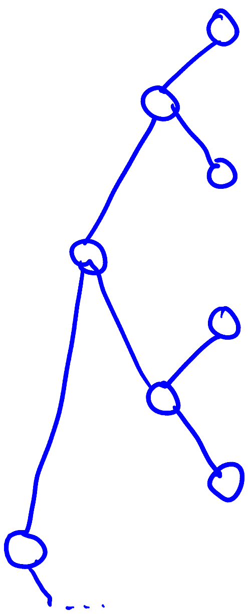
$\boxed{\text{OPEN:}}$ approximability of one-finger offline?

Order By Next Request (OBNR): [Munro–ESA 2000]

- offline (omniscient) access algorithm
- access(x): find x, say at position i
  continue scan to $\underbrace{\lceil\lceil i \rceil\rceil = 2^{\lceil \lg i \rceil}}_{\text{hyperceiling}}$

  sort these elts. by next request time
  (and put x at front, say)

Example:



x	cost	list
1	1	1 2 3 4 5 6 7 8 ···
2	2	2 1 3 4 5 6 7 8 ···
3	4	3 4 1 2 5 6 7 8 ···
4	2	4 3 1 2 5 6 7 8 ···
5	8	5 6 7 8 1 2 3 4 ···
6	2	6 5 7 8 1 2 3 4 ···
7	4	7 8 5 6 1 2 3 4 ···
8	2	8 7 5 6 1 2 3 4 9 ···
9	16	9 10 11 12 13 14 15 16 1 2 ···

etc.

Total cost per permutation $\sim n \cdot (\frac{1}{2} \cdot 2 + \frac{1}{4} \cdot 4 + \cdots)$
$= n \lg n$

MTF costs $\sim n^2/2 \Rightarrow$ ratio $\sim n/\lg n$
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 $2^0, 2^1, \ldots, \lceil r_i \rceil$

OBNR analysis: amortized cost$(x) \le 1 + 4\lceil \lg w(x) \rceil$

where $w(x)$ = # distinct y's accessed since last access(x) including x itself — WORKING-SET BOUND

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

(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 $\lceil \lg w(x) \rceil$ before access(y)

$\Rightarrow$ amortized cost $4\lceil \lg w(x) \rceil$. $\square$

- improve 4 to 2.6641… by changing $2 \to 4.24429\ldots$ [Munro]
- lower bound of $\Omega(1 + \lg w(x))$ [Demaine & Harmon 2006]

$\boxed{\text{OPEN:}}$ nail/improve constant