CSE 5311 Notes 4: Self-Organizing Linear Search

(Last updated 6/18/14 4:21 PM)

What is the optimal way to organize a *static* list for searching?

1. By decreasing access probability - optimal static/fixed ordering.

2. Key order - if misses will be "frequent", to avoid searching entire list.

Other Considerations:

1. Access probabilities may change (or may be unknown).

2. Set of keys may change.

These lead to the pursuit of *dynamically optimal* (online) data structures whose adaptive behavior is within a constant factor of an optimal (offline) strategy.

Online - must process each request before the next request is revealed.

Offline - given the entire sequence of requests before any processing. ("knows the future")

ASIDE – THE SKI RENTAL PROBLEM (Reference: S. Phillips and J. Westbrook, "On-Line Algorithms: Competitive Analysis and Beyond")

Cost of skiing:

Buy skis = $\$C$

Rent skis one time = \$1

You have never skied $\Rightarrow$ You don't know how many times (i) you will ski.

If you knew i . . .

$i \le C \Rightarrow$ Rent $\qquad$ $C \le i \Rightarrow$ Buy

So, the optimal (offline) strategy yields a cost of $\min(i,c)$.

Deterministic (online) strategy: Rent $C - 1$ times, then buy.

Cost for different scenarios:

$i < C \qquad C < i \qquad i = C$

Maximum ratio of online to offline = *Competitive Ratio*

Can *CR* be improved?

Let strategy A_j be:

Rent up to *j* - 1 times

Buy on *j*th trip

Worst-case *CR* for a particular A_j:

j ≤ *C*: Suppose *i* = *j*

$$CR = \frac{i-1+C}{i}$$

$$2 \le CR \le C$$

j > *C*: Suppose *i* = *C*

$$CR = \frac{j-1+C}{C} \ge 2$$

But, the expected *CR* can be improved by randomizing the choice of *j* to weaken the *adversary*:

Knows the algorithms (strategies)

Knows the distribution for *j*

Does not know *j*

(S. Ben-David et.al., "On the Power of Randomization in Online Algorithms")

Determining the optimal distribution (optimal mixed strategy from game theory):

Let $A_j(k)$ = cost of A_j for skiing *k* times

π_j = probability that A_j is chosen

$$\begin{bmatrix} A_1(1) & A_2(1) & \cdots & A_C(1) \\ A_1(2) & A_2(2) & \cdots & A_C(2) \\ \vdots & \vdots & \ddots & \vdots \\ A_1(C) & A_2(C) & \cdots & A_C(C) \end{bmatrix} \begin{bmatrix} \pi'_1 \\ \pi'_2 \\ \vdots \\ \pi'_C \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ \vdots \\ C \end{bmatrix}$$

Solve for $\pi'_1 \cdots \pi'_C$

$$CR = \alpha = \frac{1}{\sum_{j=1}^{C} \pi'_j} \qquad \left(\sum_{j=1}^{C} \pi'_j \ne 1 \right)$$

$$\pi_j = \alpha \pi'_j$$

Closed form:

$$\alpha = \frac{1}{\left(1+\frac{1}{C-1}\right)^{C}-1} + 1 \approx 1.582 = \frac{e}{e-1} \text{ as } C \to \infty$$

$$\pi_i = \frac{\alpha-1}{C}\left(\frac{C}{C-1}\right)^{i}$$

Concepts of Self-Organizing Linear Search

Have list adapt to give better performance.

Advantages:

Simple to code.

Convenient for situations with relatively small # of elements that does not justify more elaborate mechanism.

Useful for some user interfaces.

Access Distributions for Probabilistic Analysis:

Uniform - Theoretically convenient

80-20 (or 90-10) Rule

Zipf - n items, $P_i = \frac{1}{iH_n}$, $H_n = \sum_{k=1}^{n} \frac{1}{k}$

Since distribution may be unknown or changing, we are dealing with

Locality (temporary heavy accesses)

vs.

Convergence (obtaining optimal ordering)

Implementation Approaches

Move-to-front (good locality)

Transpose (Slow to converge. Alternating request anomaly.)

Count - Number of accesses is stored in each record (or use CLRS problem 5-1 to reduce bits).

Sort records in decreasing count order.

Move-ahead-k: more aggressive than transpose

PROBABILISTIC ANALYSIS

Markov Model (iterative solver available on course webpage: `soList.c`)

Permutation of list ⇔ state

Operation on list ⇔ transition with probability

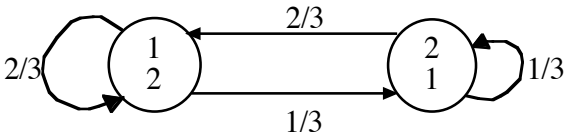
Analysis:

1. System of linear equations ⇒ probability of being in each state
2. Probability of each state ⇒ expected # of probes for retrieval

Move-to-front on list with 2 elements and Zipf's

$$P_1 = \frac{1}{1H_2} = \frac{1}{1\left(\frac{1}{1}+\frac{1}{2}\right)} = \frac{2}{3}$$

$$P_2 = \frac{1}{2H_2} = \frac{1}{2\left(\frac{1}{1}+\frac{1}{2}\right)} = \frac{1}{3}$$



1. Find probabilities for being in each of the two states.

Use $P\begin{pmatrix}1\\2\end{pmatrix} = \frac{2}{3}P\begin{pmatrix}1\\2\end{pmatrix} + \frac{2}{3}P\begin{pmatrix}2\\1\end{pmatrix}$ $P\begin{pmatrix}i\\j\end{pmatrix}$ is the probability of the list having element i before element j.

$$P\begin{pmatrix}2\\1\end{pmatrix} = \frac{1}{3}P\begin{pmatrix}1\\2\end{pmatrix} + \frac{1}{3}P\begin{pmatrix}2\\1\end{pmatrix}$$

and $1 = P\begin{pmatrix}1\\2\end{pmatrix} + P\begin{pmatrix}2\\1\end{pmatrix}$

to obtain $P\begin{pmatrix}1\\2\end{pmatrix} = \frac{2}{3}$ $P\begin{pmatrix}2\\1\end{pmatrix} = \frac{1}{3}$

2. Expected # of probes =

$$P\begin{pmatrix}1\\2\end{pmatrix}(1\bullet P_1 + 2\bullet P_2) + P\begin{pmatrix}2\\1\end{pmatrix}(1\bullet P_2 + 2\bullet P_1)$$
$$= \frac{2}{3}\left(1\bullet\frac{2}{3} + 2\bullet\frac{1}{3}\right) + \frac{1}{3}\left(1\bullet\frac{1}{3} + 2\bullet\frac{2}{3}\right) = \frac{13}{9}$$

Can generalize to get large, *sparse* system of equations for any n and any distribution. Consider n=3.

1. Find probabilities for being in each of the 3!=6 states.

Use $P\begin{pmatrix}i\\j\\k\end{pmatrix} = p_i P\begin{pmatrix}i\\j\\k\end{pmatrix} + p_i P\begin{pmatrix}j\\i\\k\end{pmatrix} + p_i P\begin{pmatrix}j\\k\\i\end{pmatrix}$ for five of the six states

and $1 = P\begin{pmatrix}1\\2\\3\end{pmatrix} + P\begin{pmatrix}1\\3\\2\end{pmatrix} + P\begin{pmatrix}2\\1\\3\end{pmatrix} + P\begin{pmatrix}2\\3\\1\end{pmatrix} + P\begin{pmatrix}3\\1\\2\end{pmatrix} + P\begin{pmatrix}3\\2\\1\end{pmatrix}$

to obtain the probability of being in each of the six states.

2. Expected # of probes =

$$P\begin{pmatrix}1\\2\\3\end{pmatrix}(1\bullet P_1 + 2\bullet P_2 + 3\bullet P_3) + P\begin{pmatrix}1\\3\\2\end{pmatrix}(1\bullet P_1 + 2\bullet P_3 + 3\bullet P_2) + P\begin{pmatrix}2\\1\\3\end{pmatrix}(1\bullet P_2 + 2\bullet P_1 + 3\bullet P_3)$$
$$+P\begin{pmatrix}2\\3\\1\end{pmatrix}(1\bullet P_2 + 2\bullet P_3 + 3\bullet P_1) + P\begin{pmatrix}3\\1\\2\end{pmatrix}(1\bullet P_3 + 2\bullet P_1 + 3\bullet P_2) + P\begin{pmatrix}3\\2\\1\end{pmatrix}(1\bullet P_3 + 2\bullet P_2 + 3\bullet P_1)$$

Rivest obtained closed form for move-to-front

$$\text{Expected \# of probes} = \frac{1}{2} + \sum_{\substack{1\le i\le n\\ 1\le j\le n}} \frac{P_i P_j}{P_i + P_j}$$

For uniform probabilities (all $P_i = \frac{1}{n}$), the closed form gives:

$$\frac{1}{2} + n^2 \frac{\frac{1}{n}\bullet\frac{1}{n}}{\frac{1}{n}+\frac{1}{n}} = \frac{1}{2} + \frac{n}{2} = \frac{n+1}{2}$$

(Aside: For Zipf's, gives 40% more probes than optimal static ordering.)

ANALYSIS OF COUNT STRATEGY VS. OPTIMAL FIXED/STATIC ORDER

1. Suppose all elements are *equally likely* (uniform distribution).

   Q. Worst case sequence?

   A. Rotate requests such that last list element is always requested.

   Q. Compare expected number of probes per request.

   A. Count uses n probes

   Optimal fixed uses $\frac{n+1}{2}$ probes Count = 2 • Optimal - 1

2. Suppose $P_1 \geq P_2 \geq P_3 \geq \cdots \geq P_n$

   Suppose total number of requests is m:

   $mP_1 \geq mP_2 \geq mP_3 \geq \cdots \geq mP_n$

   Q. Worst case sequence?

   A. Example: m=1000 n=4 P_1=0.4 P_2=0.3 P_3=0.2 P_4=0.1

400	300	200	100
300	200	100	0
200	100	0	0
100	0	0	0

   1. Perform mnP_n requests cycling among all n elements (n probes each).
   2. Perform $m(n-1)(P_{n-1} - P_n)$ requests cycling among n - 1 elements (n - 1 probes each).
   3. Perform $m(n-2)(P_{n-2} - P_{n-1})$ requests cycling among n - 2 elements (n - 2 probes each).

   . . .

   Q. Compare expected number of probes per request.

   A. Optimal fixed has expected probes = $\sum_{i=1}^{n} iP_i$

Count uses twice the expected number of probes as the optimal fixed order in the worst case, so Count is statically optimal (within a constant of the optimal fixed ordering):

$$\text{Total Probes} = mn^2 P_n + \sum_{i=1}^{n-1} (P_i - P_{i+1}) i^2 m$$

$$\text{Expected Probes} = n^2 P_n + \sum_{i=1}^{n-1} (P_i - P_{i+1}) i^2 = \sum_{i=1}^{n} P_i i^2 - \sum_{i=2}^{n} P_i (i-1)^2$$

$$= \sum_{i=1}^{n} P_i i^2 - \sum_{i=2}^{n} P_i \left(i^2 - 2i + 1 \right)$$

$$= \sum_{i=1}^{n} P_i (2i - 1) = 2 \sum_{i=1}^{n} i P_i - \sum_{i=1}^{n} P_i = 2 \bullet optimal - 1$$

Count is not dynamically optimal (not guaranteed to be within a constant of the best offline strategy).

MOVE-TO-FRONT (MTF) ONLINE STRATEGY VS. OPTIMAL OFFLINE STRATEGY

(Similar to CLRS problem 17-5. From D.D.Sleator and R.E. Tarjan, "Amortized Efficiency of List Update and Paging Rules", *CACM 28 (2)*, Feb. 1985, `http://dl.acm.org.ezproxy.uta.edu/citation.cfm?doid=2786.2793`)

Key Differences:

MTF is given requests *one-at-a-time* and thus applies an "obvious" online strategy

1) Go down the list to the requested item

2) Bring it to the front of the list

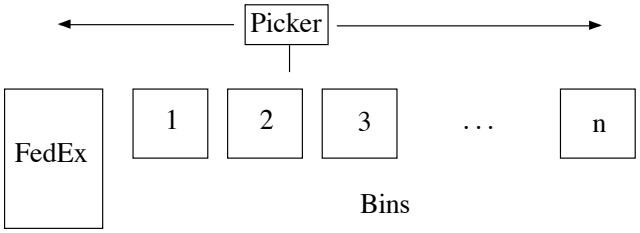
OPT is given the entire sequence of accesses (e.g. offline) *before* any searching or adjustments are applied. (There is no charge for the computation to determine the actions. This "robot scheduling" problem is NP-hard [1].)

For a request, the following "models" the cost of the processing that occurs

1) Go down the list to the requested item (charge 1 unit per item examined)

2) Optionally, bring requested item *closer* to the front of the list (no charge, "free")

3) Optionally, perform "paid" transpositions on adjacent elements (charge 1 unit each)



From amortized complexity:

$$\hat{c}_i = c_i + \Phi(i) - \Phi(i-1)$$

$$\sum_{i=1}^{m} c_i + \Phi(m) - \Phi(0) = \sum_{i=1}^{m} \hat{c}_i$$

If $\Phi(m) - \Phi(0)$ is never negative, then an upper bound on $\sum_{i=1}^{m} \hat{c}_i$ is an upper bound on $\sum_{i=1}^{m} c_i$.

Potential Function Used to Compare Two Strategies:

To simplify the analysis, assume both strategies start with same list. The lists may differ while processing the sequence.

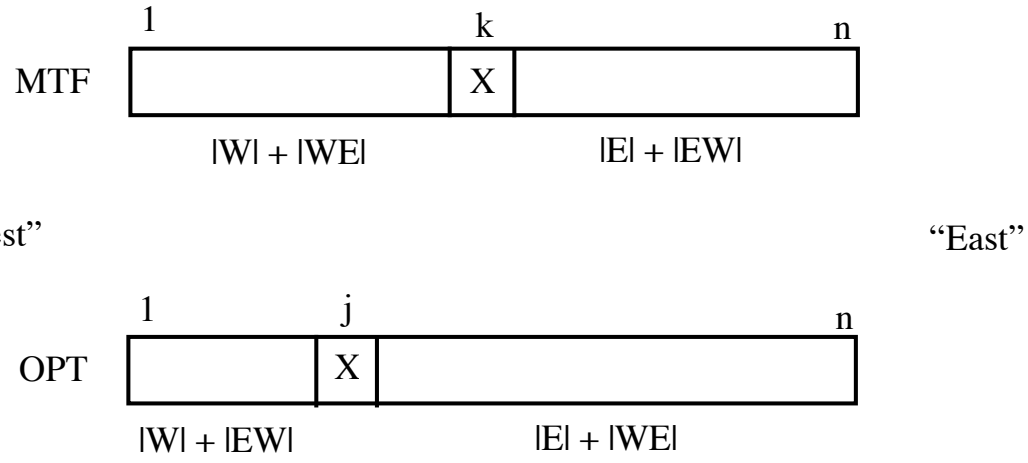
The potential (Φ) is the number of *inversions* in the MTF list with respect to the OPT list. (i.e. count the number of pairs whose order is different in the two lists.) Intuitively, this is the cost for MTF to correct its list to correspond to OPT.

Example: Lists 2, 3, 4, 5, 1 and 3, 5, 4, 1, 2 have 5 inversions.

Defining Φ this way gives $\Phi(0) = 0$ and $\Phi(i) \geq 0$, so $\Phi(m) - \Phi(0)$ is never negative.

To compare MTF to OPT (without detailing OPT), the amortized cost ($\hat{c}_i$) of each MTF operation is bounded by the actual cost of the corresponding OPT operation.

Before processing ACCESS to some X



W = Items "west" of X in both lists E = Items "east" of X in both lists

WE = Items west of X in MTF, but east of X in OPT

EW = Items east of X in MTF, but west of X in OPT

Observations:

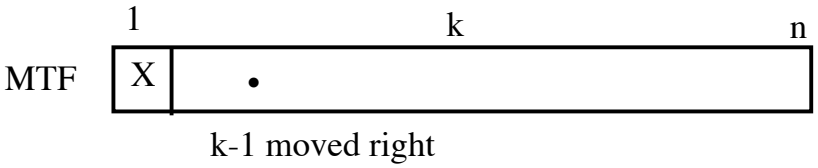
$\|W\| + \|EW\| + 1 = j$	(i)	From OPT diagram
$k = \|W\| + \|WE\| + 1$		From MTF diagram
$k - 1 - \|WE\| = \|W\|$	(ii)	Algebra on previous step
$k - \|WE\| + \|EW\| = j$		Add (i) and (ii), then simplify
$k - \|WE\| \leq j$		From previous step and fact that $\|EW\| \geq 0$

Amortized cost of ACCESS of X (by MTF)

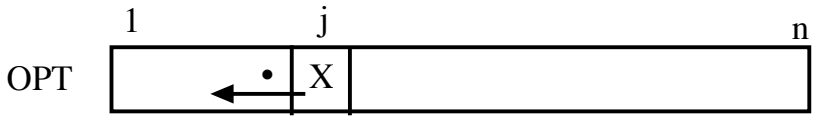
Search + |W| new inversions - |WE| lost inversions

$$k + (k - 1 - |WE|) - |WE| = 2(k - |WE|) - 1 \leq 2j - 1$$

BUT, OPT list also changes (but details are not needed)



k-1 moved right



f free transpositions

p paid transpositions

Actual cost of OPT = j + p

"Correction" to Δ in $\Phi \leq$ -f for lost inversions + p for new inversions

So, after accessing and updating both lists, the amortized cost of MTF ACCESS is

$$\hat{c}_i \leq k + k - 1 - |WE| - |WE| - f + p \leq 2j - 1 - f + p \leq 2j - 1 + 2p \leq 2OPT - 1$$

Based on the observation that an upper bound on $\sum_{i=1}^{m} \hat{c}_i$ is an upper bound on $\sum_{i=1}^{m} c_i$, MTF is dynamically optimal.

[1] C. Ambühl, "Offline List Update is NP-Hard", *Lecture Notes in Computer Science 1879*, Springer-Verlag, 2000.

Aside: Randomization may be used to obtain strategies with better competitive ratios.

BIT:

Every list element has a bit that is set randomly.

Each ACCESS complements bit, but only does MTF if bit is 1.

Achieves expected $CR \leq 1.75$. Other methods achieve 1.6.

(Recent related paper: S. Angelopoulos and P. Schweitzer, "Paging and List Update under Bijective Analysis", *JACM* 60, *2*, 2013, `http://dl.acm.org.ezproxy.uta.edu/citation.cfm?doid=2450142.2450143`.)