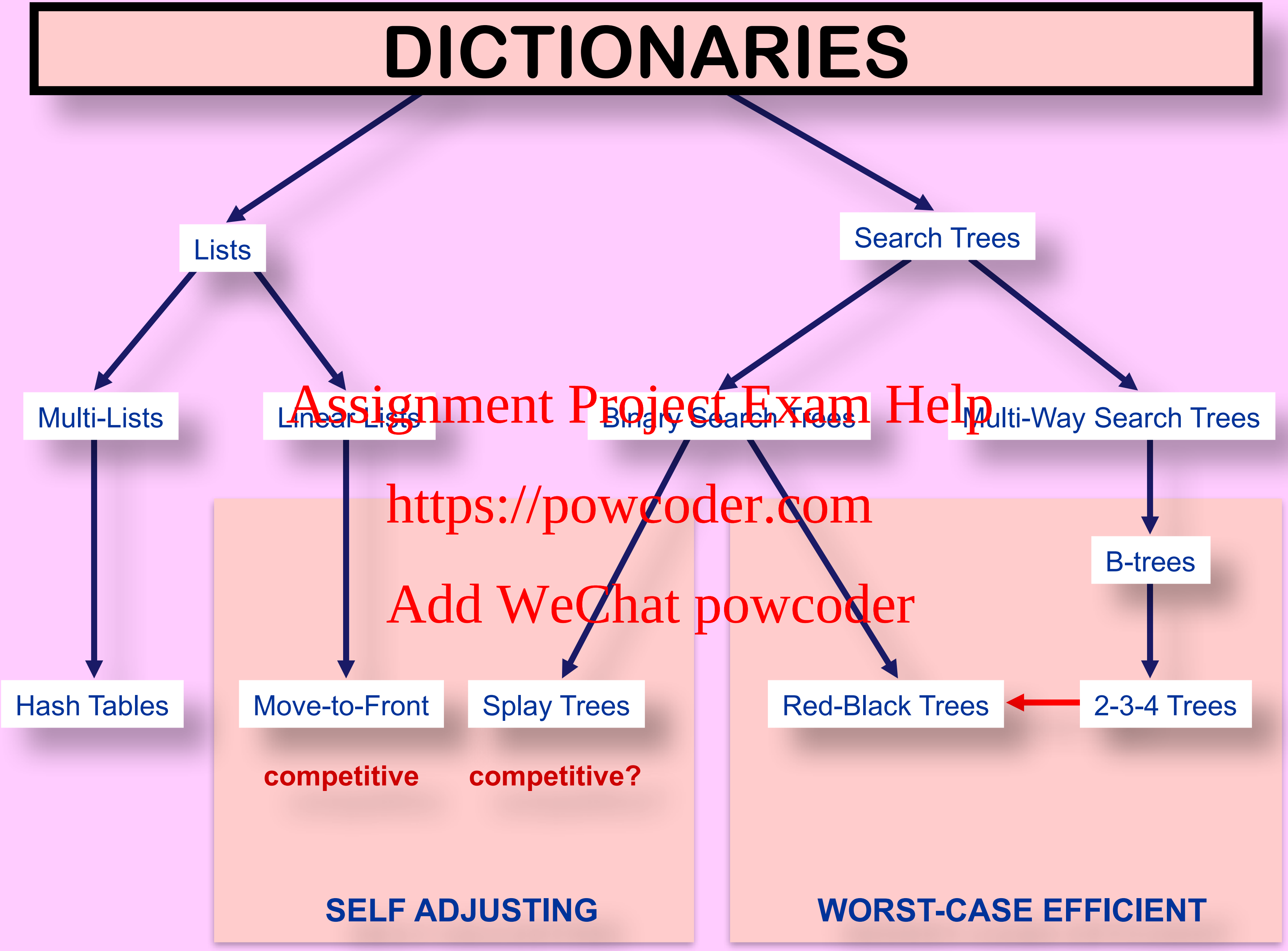
EECS 4101/5101

Prof. Andy Mirzaian

Move to Front Self Adjusting Linear Lists


DICTIONARIES
Lists
Search Trees
Multi-Lists
Linear Lists
Binary Search Trees
Multi-Way Search Trees
B-trees
Hash Tables
Move-to-Front
Splay Trees
Red-Black Trees
2-3-4 Trees
competitive
competitive?
SELF ADJUSTING
WORST-CASE EFFICIENT

TOPICS in this slide

- Linear List with Move-to-Front

- Competitiveness of Move-to-Front:
  - Static Dictionary
  - Dynamic Dictionary
  - Expected Case

References:

✂Lecture Note 2

Move-to-Front

Introduction

- **DICTIONARY:** Maintain a set D of items (each with a unique identifier called key) that supports the following three operations:

  Search (x, D): Locate item x in D (also called access)

  Insert (x, D): Insert item x in D (no duplicate keys)

  Delete (x, D): Delete item x from D

- s = a sequence of m dictionary operations.
- We usually assume D is initially empty (except for static dictionaries, i.e., search only).
- We often assume there is a linear ordering defined on keys (e.g., integers) so that we can make key comparisons.

- **D as a SEQUENTIAL LINEAR LIST:**

  We need i probes to sequentially access the item at position i on list D.

- **SELF-ADJUSTING FEATURE:**

  Exchange: swap the positions of a pair of adjacent items on D.
  Free exchange: exchanging the accessed/inserted item with its preceding item on D.
  Paid exchange: all other types of exchanges.
  Costs: 1 for each item probe
  1 for each paid exchange
  0 for each free exchange

Some on-line heuristics

- **Move-to-Front (MF):**
  After each dictionary operation perform maximum number of free exchanges (no paid exchanges).
  This moves the accessed/inserted item to the front of the list without affecting the relative order of the other items on the list.

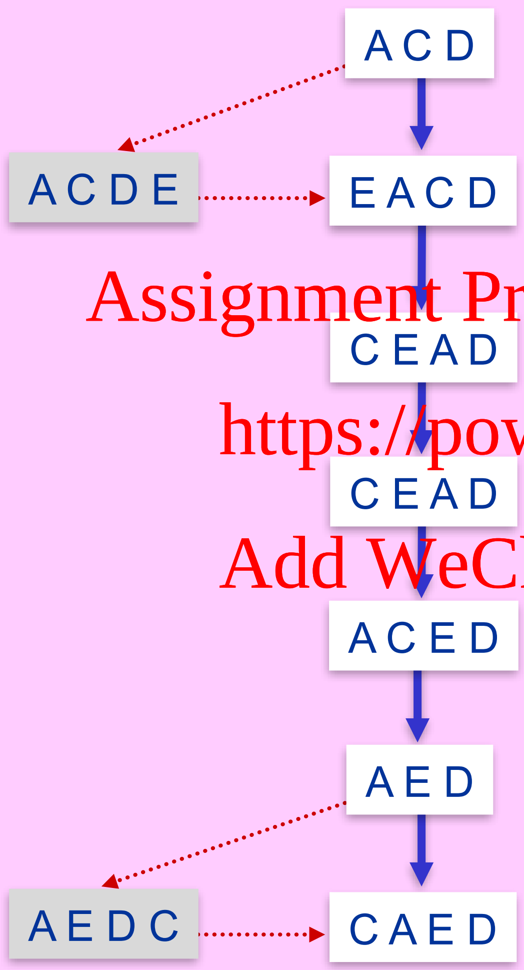
- **Transpose (T):**

  After each dictionary operation perform one free exchange if possible (no paid exchanges).

  This moves the accessed/inserted item one position closer to the front (if not at the front already), without affecting the relative order of the other items on the list.

- **Frequency Count (FC):**
  Maintain a frequency count for each item, initially zero. (This requires extra space.) Increase the count of an item each time it is accessed/inserted; reduce its count to zero when deleted. Perform exchanges as necessary to maintain the list in non-increasing order of frequency counts.

Move-to-Front Example



List	Operation	Cost
	Insert E	4
	Insert C	3
	Search B	4
	Search A	3
	Delete C	2
	Insert C	4

Move-to-Front Applications

- **Cache based Memory Management:**

  - Small fast Cache versus large slower main memory.

  - Cache I/O page-in-page-out rules.

  - Least Recently Used (LRU) paging rule = Move-to-Front.

  [Sleator, Tarjan, "Amortized efficiency of list update and paging rules," Communications of ACM, 28(2), pp: 202-208, 1985.]

- **Adaptive Data Compression:**

  - Adaptive coding based on MF compares favorably with Hoffman coding.

  [Bentley, Sleator, Tarjan, Wei, "A locally adaptive data compression scheme," Communications of ACM, 29(4), pp: 320-330, 1986.]

Static Dictionary

Static Dictionary: Search only

- Initial list $D_0 = [\ x_1,\ x_2,\ \ x_3,\ \ldots,\ x_n\ ]$.
  s = a sequence of m successful searches.
  k_i = the frequency count (number of search requests in s) for item x_i, i = 1..n.
  $m = k_1 + k_2 + \ldots + k_n$ total number of search requests in s.

- **Decreasing Frequency (DF):**
  This off-line strategy initially arranges the list items in non-increasing order of access frequencies, and does not perform any exchanges. We do not charge DF the cost of the initial rearrangement! Without loss of generality assume:

$$k_1 \quad k_2 \quad \ldots \quad k_n$$

- **FACT:** Among off-line strategies that perform no exchanges, DF is Optimum.

$$C_{DF}(s) = \sum_{i=1}^{n} i \cdot k_i = \sum_{i=1}^{n} \sum_{j=1}^{k_i} i \, .$$

$$C_{MF}(s) = \sum_{i=1}^{n} \sum_{j=1}^{k_i} t_{ij} \, , \qquad t_{ij} = \text{position of } x_i \text{ during its } j^{th} \text{ access.}$$

- Challenging Question: What is the optimum off-line strategy with exchanges?

Static Dictionary: Example sequence

Initial list D_0 = [A, B, C].

s = a sequence of m=9 searches to A, B, C.

$k_A = 4 \; > \; k_B = 3 \; > \; k_C = 2$.

$C_{DF}(s) = 1 \cdot k_A + 2 \cdot k_B + 3 \cdot k_C = 1 \cdot 4 + 2 \cdot 3 + 3 \cdot 2 = 16$.

$C_{MF}(s)$ = ? Depends on s.

- s_1 = A A A A B B B C C

  $C_{MF}(s_1) = 1+1+1+1 + 2+1+1 + 3+1 = 12 \qquad < C_{DF}(s_1)$

- s_2 = C C B B B A A A A

  $C_{MF}(s_2) = 3+1 + 3+1+1 + 3+1+1+1 = 15 \qquad < C_{DF}(s_2)$

- s_3 = C B A C B A B A A

  $C_{MF}(s_3) = 3+3+3 + 3+3+3 + 2+2+1 = 23 \qquad > C_{DF}(s_3)$

Static Dictionary: MF Efficiency

THEOREM 1: $C_{MF}(s) \le 2C_{DF}(s) - m.$

Proof: MF's List: $\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ x_i\ \ldots\ldots\ldots\ldots\ldots\ldots$

t_{ij} = cost of j^{th} access to x_i

$s = \;.\;.\;.\;x_i\;.\;.\;.\;x_i\;.\;.\;.\;x_i\;.\;.\;.\;\underbrace{x_i\;.\;.\;.\;.\;.\;.\;.\;.\;.\;x_i}_{}\;.\;.\;.\;x_i\;.\;.\;.\;x_i\;.\;.\;.$

j^{th} access to x_i

subsequence s_{ij}

$(j-1)^{st}$ access to x_i

$A_{ij} = |\{ x_h \mid h > i,\ x_h \text{ accessed during } s_{ij} \}|$

$B_{ij} = |\{ x_l \mid l < i,\ x_l \text{ accessed during } s_{ij} \}| \le i - 1$

$t_{ij} = A_{ij} + B_{ij} + 1 \le A_{ij} + i$

$$\sum_{j=1}^{k_i} A_{ij} \le k_{i+1} + k_{i+2} + \cdots + k_n$$

continued

Static Dictionary: MF Efficiency

THEOREM 1: $C_{MF}(s) \le 2C_{DF}(s) - m.$

Proof Cont'd: $t_{ij} \le A_{ij} + i, \qquad \sum_{j=1}^{k_i} A_{ij} \le k_{i+1} + k_{i+2} + \cdots + k_n$

$$C_{MF}(s) = \sum_{i=1}^{n}\sum_{j=1}^{k_i} t_{ij} \le \sum_{i=1}^{n}\sum_{j=1}^{k_i} (A_{ij} + i) = \sum_{i=1}^{n}\sum_{j=1}^{k_i} i + \sum_{i=1}^{n}\sum_{j=1}^{k_i} A_{ij} = C_{DF}(s) + \sum_{i=1}^{n}\sum_{j=1}^{k_i} A_{ij}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{k_i} A_{ij} \le \sum_{i=1}^{n} (k_{i+1} + k_{i+2} + \cdots + k_n) = \begin{pmatrix} k_2 + k_3 + k_4 + \cdots + k_m \\ \quad + k_3 + k_4 + \cdots + k_m \\ \qquad + k_4 + \cdots + k_m \\ \cdots\cdots \\ \cdots\cdots \\ + k_m \end{pmatrix}$$

$$= \sum_{i=1}^{n} (i-1)k_i = \sum_{i=1}^{n} i \cdot k_i - \sum_{i=1}^{n} k_i = C_{DF}(s) - m$$

$$\therefore C_{MF}(s) \le 2C_{DF}(s) - m$$

QED

Accounting Interpretation

An amortized interpretation of $C_{MF}(s) \le 2C_{DF}(s) - m$ is $\hat{c}_{MF} \le 2c_{DF} - 1$, where

$\hat{c}_{MF}$ = the amortized cost of a search operation by MF,

c_{DF} = the actual cost of the same operation by DF.

$h > i$

MF's List: ....x_h .... x_i ....

access cost to x_i

Excess access cost due to x_h, $h > i$, having jumped in front of x_i.

These jumps also cause inversions on MF's list compared to DF's fixed list.

Transfer excess charge from x_i to x_h (back in time to) when x_h was last accessed and jumped ahead of x_i.

So, x_h gets charged its normal access cost h, and a transferred charge of at most h-1 (for lower indexed items x_i, i=1..h-1, that were jumped over).

Amortized access cost to x_h: $\hat{c}_{MF} \le h + (h-1) = 2h-1 = 2c_{DF} - 1$.

We will generalize this idea to dynamic dictionaries.

Dynamic Dictionary

The Setup

- s = a sequence of m search, insert, delete operations on an initially empty list.
- A = an arbitrary algorithm that executes s.

- $F_A(s)$ = total # of free exchanges made by A on s.
- $X_A(s)$ = total # of paid exchanges made by A on s.

- $C^-_A(s)$ = total cost by A on s, excluding paid exchanges.
- $C_A(s)$ = total cost by A on s, including paid exchanges.

FACT: The following hold:

1. $C_A(s) = C^-_A(s) + X_A(s)$.

2. $F_A(s) \le C^-_A(s) - m$.

3. $X_{MF}(s) = X_T(s) = X_{FC}(s) = 0$.

4. $C_{MF}(s) = C^-_{MF}(s)$.

MF is 2-Competitive

THEOREM 2: $C_{MF}(s) \le 2C_A(s) - m$, for all A and s.

Proof: We will prove $C_{MF}(s) \le 2C^{-}_A(s) + X_A(s) - F_A(s) - m$.

$\hat{c}_{MF}$ = amortized cost charged to MF by a single operation, excluding A's exchanges.
c_A = actual cost of A on the same operation, excluding A's exchanges.

CLAIM:

$$\hat{c}_{MF} \begin{cases} \le 2c_A - 1 & \text{for search or insert} \\ \le c_A \le 2c_A - 1 & \text{for delete} \\ \le +1 & \text{for a paid exchange by A} \\ = -1 & \text{for a free exchange by A} \end{cases}$$

Proof by the potential function method. Potential:

Φ(MF, A) = Number of inversions in MF's list with respect to A's list.

Inversion = any (not necessarily adjacent) pair of items (x,y), where x appears before y in MF's list, but x appears after y in A's list.

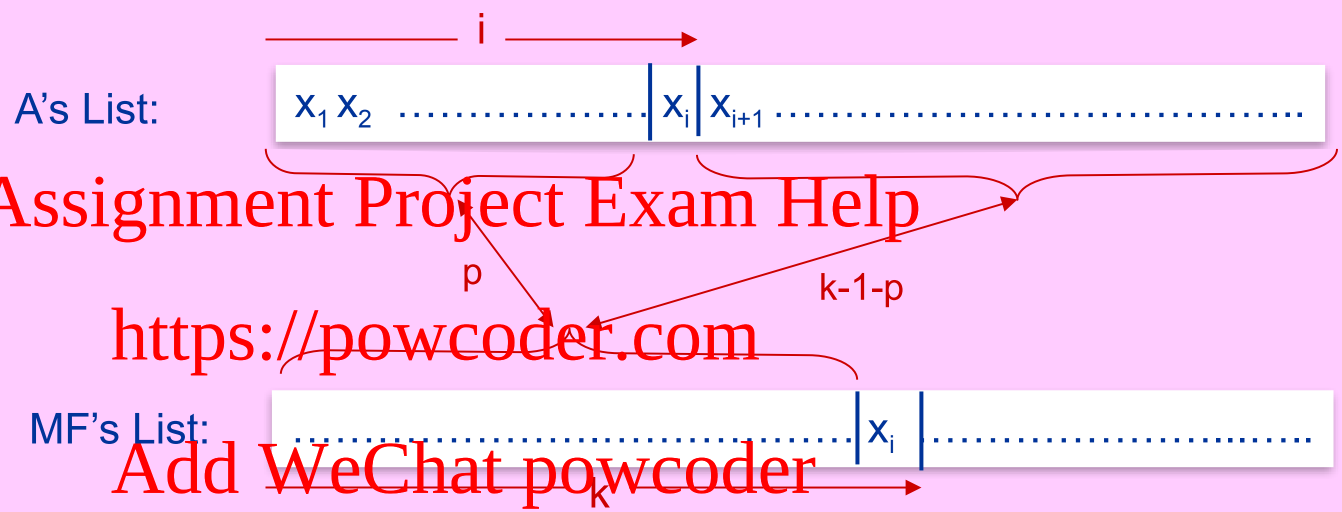
Example: Φ([3,6,2,5,4,1] , [1,2,3,4,5,6]) = 10.

continued

MF is 2-Competitive

THEOREM 2: $C_{MF}(s) \le 2C_A(s) - m$, for all A and s.

Proof of CLAIM Cont'd:



Not yet accounting for exchanges made by A:

Search: $\hat{c}_{MF} = c_{MF} + \Delta\Phi = k + [\ p - (k\text{-}1\text{-}p)\ ] = 2p + 1 \le 2(i-1)+1 = 2i - 1 = 2c_A - 1.$

Insert: The same as search with i = k = L+1 (L = length of the list before insertion.)

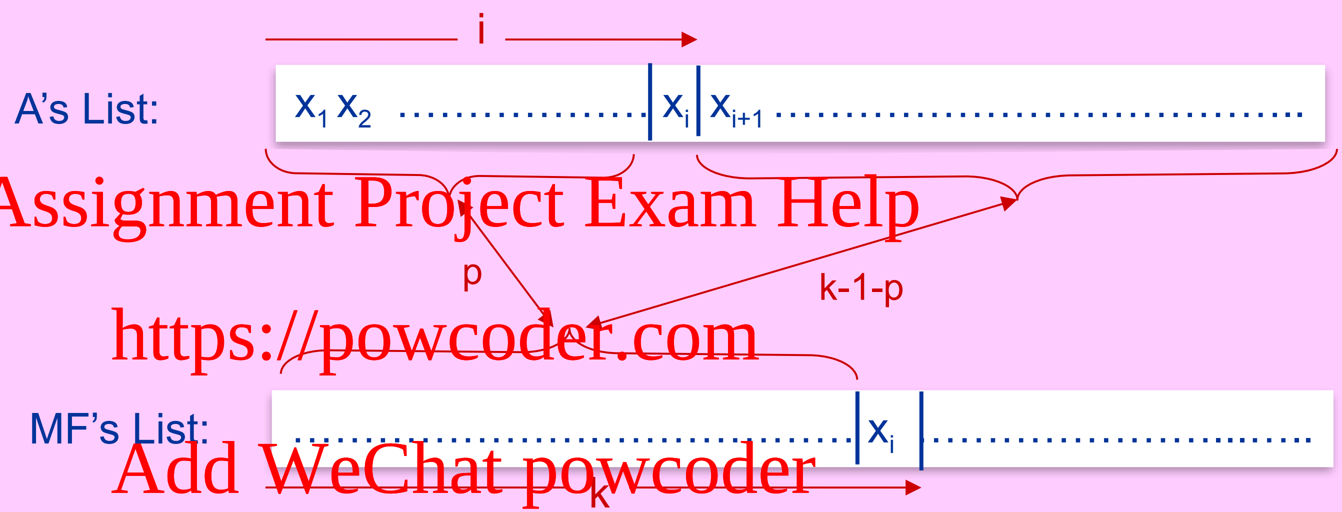
Delete: $\hat{c}_{MF} = c_{MF} + \Delta\Phi = k + [\ -(i\text{-}1\text{-}p) - (k\text{-}1\text{-}p)\ ] = 2(p+1) - i \le i = c_A \le 2c_A - 1.$

continued

MF is 2-Competitive

THEOREM 2: $C_{MF}(s) \le 2C_A(s) - m$, for all A and s.

Proof of CLAIM Cont'd:



Accounting for exchanges made by A:

Paid exchange by A: $\hat{c}_{MF} = c_{MF} + \Delta\Phi \le 0 + 1 = 1.$

Free exchange by A: $\hat{c}_{MF} = c_{MF} + \Delta\Phi = 0 - 1 = -1.$

QED

Expected Case

Static Dictionary: Expected Case

- Initial list $D_0 = [\ x_1, x_2, x_3, \ldots, x_n\ ]$. Search only.
- p_i = search probability for item x_i, $i = 1..n$.
- $p_i > 0$, $p_1 + p_2 + \ldots + p_n = 1$.

- As in Decreasing Frequency (DF), assume initial list is arranged in non-increasing order of access probability.

$$p_1 \quad p_2 \quad \ldots \quad p_n > 0.$$

- E_A = expected cost of a single search operation by algorithm A.

Static Dictionary: Expected Case

THEOREM 3: The following hold (when $p_1 \;\; p_2 \;\; \ldots \;\; p_n > 0$):

(a) $$E_{DF} = \sum_{i=1}^{n} i \cdot p_i$$

(b) $$E_{MF} = 1 + 2 \sum_{1 \le i < j \le n} \frac{p_i p_j}{p_i + p_j}$$

(c) $$E_{MF} \le 2E_{DF} - 1.$$

Proof:

(a) This follows directly from the definition of expectation.

(c) This follows from (a) and (b) and using the fact that $p_i/(p_i + p_j) \le 1$.

(b) See Next page.

continued

Static Dictionary: Expected Case

Proof of (b): $E_{MF} = \sum_{i=1}^{n} L_i \cdot p_i \qquad (L_i = \text{expected position of } x_i \text{ on the list})$

0/1 random variable $Y_{ij} = \begin{cases} 1 & \text{if } x_j \text{ appears before } x_i \\ 0 & \text{otherwise} \end{cases}$

$p_{ij} = \text{Prob}[Y_{ij} = 1] = \text{Prob}[x_j \text{ is accessed} \mid x_i \text{ or } x_j \text{ is accessed}] = \dfrac{p_j}{p_i + p_j}$

conditional probability

$E[Y_{ij}] = 1 \cdot p_{ij} + 0 \cdot (1 - p_{ij}) = p_{ij}$

by the linearity of expectation

$$L_i = 1 + E\left[\sum_{\substack{j=1 \\ j \neq i}}^{n} Y_{ij}\right] = 1 + \sum_{\substack{j=1 \\ j \neq i}}^{n} E[Y_{ij}] = 1 + \sum_{\substack{j=1 \\ j \neq i}}^{n} p_{ij} = 1 + \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{p_j}{p_i + p_j}$$

$$E_{MF} = \sum_{i=1}^{n} \left(\left(1 + \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{p_j}{p_i + p_j} \right) \cdot p_i \right) = \sum_{i=1}^{n} p_i + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{p_i p_j}{p_i + p_j} = 1 + 2 \sum_{1 \le i < j \le n} \frac{p_i p_j}{p_i + p_j}.$$

QED

Exercises

1. Consider the Move-to-Front (MF) and the Transpose (T) heuristics on linear-lists. Show a sequence s of n dictionary operations search/insert/delete that starts with the empty set, such that the ratio $C_T(s)/C_{MF}(s)$ is asymptotically as high as possible. What is that asymptotic ratio as a function of n?

2. Show that the Transpose (T) and the Frequency Count (FC) on-line heuristics are not competitive for:
   (a) Static dictionaries.
   (b) Dynamic dictionaries.

3. Theorem 2 shows that the total cost of the Move-to-Front (MF) heuristic is at most twice the cost of the best off-line strategy (over any sequence s of m dictionary operations on an initially empty list). Show this ratio bound is tight in the worst case. That is, for any sufficiently large m, give an adversarial sequence s of m dictionary operations on an initially empty list such that if algorithm A is the optimal off-line strategy for sequence s, then the asymptotic cost ratio is $C_{MF}(s) / C_A(s) = 2 - o(1)$ (i.e., the ratio approaches 2).

4. Theorem 2 shows that the amortized running time of the Move-to-Front (MF) heuristic on linear lists is within a constant (i.e., 2) multiplicative factor of the best off-line strategy. Does the same result hold (possibly with a somewhat larger multiplicative constant factor) for the Move-Half-Way-to-Front (MHWF) strategy? Prove your claim. (The latter strategy moves the accessed/inserted item half-way towards the front. That is, if the item was at position i, then it is moved to position $\lceil i/2 \rceil$, without affecting the relative order of the other elements.)
[Hint: First compare MHWF with MF, then apply transitivity.]

5. In the no-free-exchange model we assume that both “free” and “paid” exchanges have unit cost each. Prove that in this model the move-to-front algorithm has a competitive ratio of 4.
[Hint: You'll need to adjust the potential function.]

6. Randomized Move-to-Front: This algorithm flips a coin and moves the accessed or inserted item to front with probability ½. Now to be competitive, the **expected** cost (taken over all its random choices) of the algorithm running on the sequence should be within a constant factor of the cost of the optimum off-line algorithm on the same sequence. Prove that in this sense, the randomized move-to-front algorithm is competitive.

7. This question concerns Theorem 3 on the Move-to-Front heuristic. Consider the following (non-increasing) access probability distributions (for a suitable normalization parameter α that depends on n but not on i).

(a) Uniform distribution: $p_i = \alpha$, $i=1..n$.

(b) Exponential distribution: $p_i = \alpha\cdot 2^{-i}$, $i=1..n$.

(c) Harmonic distribution: $p_i = \alpha/i$, $i=1..n$.

For each of these access probability distributions answer the following two questions:

(i) What should α be so that the given formula becomes a valid probability distribution.

(ii) Evaluate the ratio E_{MF} / E_{DF} and derive its exact limit as n goes to infinity. How does it compare with the upper bound 2 given in Theorem 3(c)?

END