, 11

CS 240 - Data Structures and Data Management **Spring 2017**

Lecture 10, 11: June 2 - June 9, 2017

Lecturer: Taylor Smith *Notes By: Harsh Mistry*

10.1 Optimal Static Ordering

A list of elements ordered by non-increasing probability of access has minimum expected access cost.

- $L = \langle x_1, \dots, x_n \rangle$
  Expected access cost in L is
  $E(L) = \sum_{i=1}^{n} P(x_i)T(x_i) = \sum_{i=1}^{n} P(x_i) \cdot i$ $P(X_i)$ - access probability for x_i
  $T(x_i)$ - position of x_i in L
- Example : P(a) = 0.3 , P(b) = 0.5, P(c) = 0.2
  $L = \langle a, b, c \rangle \implies E(L) = 0.3 + 0.5 \times 2 + 0.2 \times 3 = 1.9$

Proof: Proof by contradiction

- $L = \langle x_1, \dots, x_k, x_{k+1}, \dots, x_n \rangle$
  Suppose the access cost of L is optimal and there is k such that $P(x_k) < P(x_{k+1})$

$$E(L) = P(x_k) \cdot k + P(x_{k+1}) \cdot (k+1) + \sum_{i \neq k, k+1} P(x_i) \cdot i$$

- Create another list L' by swapping x_k and x_{k+1}
  $L' = \langle x_1, \dots, x_{k+1}, x_k, \dots, x_n \rangle$

$$E(L') = P(x_{k+1}) \cdot k + P(x_k) \cdot (k+1) + \sum_{i \neq k, k+1} P(x_i) \cdot i$$

- $E(L') - E(L) = P(x_k) - P(x_{k+1}) < 0 \implies E(L') < E(L)$

■

10.2 Dynamic Ordering

If you don't know the probabilities ahead of time, you can

- Move-To-Front(MTF): Upon a successful search, move the accessed item to the front of the list
- Transpose: Upon a successful search, swap the accessed item with the item immediately preceding it

10.2.1 Performance of dynamic ordering

- Both can be implemented in arrays or linked lists.
- Transpose does not adapt quickly to changing access patterns.
- MTF Works well in practice.
- Theoretically MTF is competitive:
- No more than twice as bad as the optimal offline ordering.

10.3 Skip Lists

- Randomized data structure for dictionary ADT
- A hierarchy of ordered linked lists
- A skip list for a set S of items is a series of lists $S_0, S_1, \ldots, S_h$ such that:
  - Each list S_i contains the special keys $-\infty$ and $+\infty$
  - List S_0 contains the keys of S in non-decreasing order
  - Each list is a subsequence of the previous one, basically each list is a subset of previous list.
  - List S_h contains only two special keys
- A two-dimensional collection of positions: levels and towers

Search In Skip Lists

```
skip-search(L,k)
L : A skip list, k : a key
  p = topmost left position of L
  S = stack of positions initially containing p
  while below(p) != null do
    p = below(p)
    while key(after(p)) < k do
      p = after(p)
    end while
    S.push(p)
  end while
  return S
```

- S contains positions of the largest key less than k at each level
- after(top(s)) will have key k, iff k is in L

Sumamry of Skip List

- Expected space usage $O(n)$
- Expected height : $O(\log n)$
  A skip list with n items has height at most 3 log n with probability at least $1 = 1/n^2$

- Skip-Search $O(\log n)$
- Skip-Insert $O(\log n)$
- Skip-Delete $O(\log n)$
- Skip lists are fast and simple to implement in practice