Module 6 - Dictionary Tricks

Graham Cooper

June 30th, 2015

Interpolation Search

Array:

A =

0	10	17	23	57	41	58	62	73	82	91	105

Seach(23)
Binary Search, checks index ($\frac{L+R}{2}$) = floor(L + $\frac{R-L}{2}$) = 5 - chec item at index 5
Interpolation - hceck index L + ($\frac{k-A[L]}{A[R]-A[L]}(R-L)$
= L + $\frac{23-0}{100-0}$ * (R - L = L + $\frac{23}{100}(R-L)$
= 2, check index 2

Works well if keys are uniformly distributed: O(log(logn)) on average, O(n) worst case performance.

Gallop Search

Array:

A =

0	10	17	23	57	41	58	62	73	82	91	105

Search(73)
Check 0, then 10, then double steps and check 23, then double again and check 58, then double again and see 120. We know that 73 is between 120 and 58.
we do 1 + 2 + 4 + 8 + ... = $2^k \leq$ 2n
k $\in$ Θ(logn) + $\Theta(logn)$ = Θ(logn)

Self-Organizing Search

:(

Dynamic Ordering

:(Bad algorithm would be selecting L then L-1 then L then L-1.... which ends up being $\Theta(nL)$ for n accesses

An optimal alogirhtm moves L, L-1 to the front, then we access only these two and then they are 1 + 2 + 1 + 2 + ... = $\Theta(1.5n)$ run time

<u>**ADversarial Sequence**</u>:
- Keep accessing the last item,
- eg (z,y,x, ... a)^m repeat m times
- The cost of MTF
- (L + L + ... + L)^m = L^2 * m

Now a static algoritm that does not move items (L + (L-1) + ... + 1)^m = $(\frac{L(L+1)}{2}$
Opt $\leq \frac{m*L(L+1)}{2}$

Cost of MTF $\rightarrow \frac{C_{MTF}}{C_{OPT}} \approx 2$
$\exists$ a sequence for which C_{MTF} = 2 * C$_{OPT}$

Avg. Case complexity of MTF:
- Sequence generated randomly
- ie. each item j has a probability p to appear
- opt orders item is decreasing order of probabilities

C$_{opt}$ $= \sum_{j=1}^{n} P_j$
C$_{MTF}$ $= \sum_{j=1}^{n} P_j(Cost of finding P_j)$
$= \sum_{j=1}^{n} P_j(1 + number of items before P_j)$
Prob of an item i being before j:
= prob i being requested more recently
$= \frac{P_i}{P_i+P_j}$

$C_{MTF} = \sum_{j=1} nP_j(1 + \sum_{i \neq j} \frac{P_i}{P_i+P_j})$
- Do algebra $\rightarrow C_{MTF} \leq C_{OPT}$

$C_{Trans} = n * (L - \frac{1}{2})$
$C_{OPT} = n * (1.5)$

Skip Lists

- a set of lists

- each list contains a subset of key from previous list
- the first list S0 contains all items (infinity)
- each item in a list Si has a pointer to soem item in Si-1 $\implies$ Each item has tower of nodes

Searching for x

- Scan forward
- find two consectuive nodes L, R which define upper and lower bounds for n in Si
- Drop down
- Move to the next list Si-1 using pointer of L

Isert

- Randomly compute the hiehg of new item: repeatedly toss a coin until you get tails, let i the number of times the coun cam up heads
- search for k in the skip list and find the positions p0, p1, ... pi of the ites with the largest ... He moved the slide :(