Advanced Data Structures
Lecture 6

Lecture by Dr. Shay Mozes
Typeset by Steven Karas

2016-12-15
Last edited 20:37:42 2016-12-15

Sometihng
Two "equation sheets" for exam.

1 Competitive Analysis

Competitive Analysis compares our algorithm against one that has perfect knowledge. Is used in the Advanced Algorithms course to evaluate online algorithms.

Worst case $\Theta(n)$

Average case Access each element with uniform probability. $\Theta(n)$

Stochastic Access each element i with probability p_i such that $p_1 \geq p_2 \geq ... \geq p_n$. $\sum_{i=1}^{n} i \cdot p_i$. In practical terms, the probability each element is accessed is known to us.

Static Sequence of queries $\sigma = x_1, ..., x_m$. Define f_i be the number of queries for element i. However, the data structure cannot be changed at runtime.

Dynamic We are allowed to change the order of the list at runtime. The dynamic optimal solution is the most efficient solution given the order of queries.

Sleator Trajan Cost Model When we serach for the element x in the place i, we pay i. To move the element x earlier in the list is free. Switching between neighbors costs us 1.

2 Self Balancing Linked Lists

A list of n elements $1, 2, ..., n$.

Static Optimal An static optimal strategy is to order the elements by f_i descending.

2.1 Access

Walk the list to find the element.

Possible strategies

Move to Front Move each element to the front when accessed.

Transpose Promote by 1 on each access.

Frequency Count Each time an element is accessed.

An algorithm A is k-competitive, if there is a b such that any sequence σ of queries it holds that $A(\sigma) \leq k \cdot OPT + b$.

2.2 Frequency Count

Frequency count is 2-competitive: $FC \leq 2 \cdot staticOPT$. Proof is left as an exercise. However, FC is not competitive to dynamic optimum: $FC \neq O(1) \cdot dynamicOPT$

Counterexample for dynamic optimum

$$\underbrace{1, ..., 1}_{n \text{ times}}, \underbrace{2, ..., 2}_{n-1 \text{ times}}, ..., n$$

$dynamicOPT$ solves this in $\Theta(n^2)$, whereas FC solves this in $\sum_{i=1}^{n}(n+1-i) \cdot i = \Theta(n^3)$

2.3 Transpose

Transpose is not competitive to $staticOPT$. The counterexample is $n, n-1, n, ...$ and so on m times. Thus:

$$Tr : \Theta(m \cdot n)$$
$$staticOPT = \Theta(m)$$

2.4 Move to Front

Move to Front is 2-competitive wrt both $staticOPT$ and $dynamicOPT$.

2-competitive with $staticOPT$ Let $Cost_{ij}$ be the number of times element i has been accessed while searching for element j.

$$totalCost = \sum_{j} \sum_{j \neq i} Cost_{ij}$$

Assuming that $f_1 \geq ... \geq f_n$:

$$Cost_{ij}^{OPT} = \begin{cases} f_j & i < j \\ 0 & else \end{cases}$$

⇐ Please note that MTF has a very common worst case: a reverse scan

$$staticOpt = \sum_{i \neq j} Cost_{ij} = \sum_{i<j} Cost_{ij}^{OPT} + \sum_{i>j} Cost_{ij}^{OPT} = \sum_{i<j} f_j$$

$Cost_{ij}^{MTF} \leq f_j$ because we search f_j times for j. Also, $Cost_{ji} \leq f_j$ because after we search for i, we will not run into j until we search for j again.

$$MTF = \sum_{i<j} Cost_{ij}^{MTF} + \sum_{j>i} Cost_{ij}^{MTF} \leq \sum_{i<j} f_j + \sum_{j>i} Cost_{ji}^{MTF}$$

$$\leq \sum_{i<j} f_j + \sum_{i<j} f_j = 2 \cdot staticOPT$$

2-competitive with *dynamicOPT* The potential will quantify how different the list of MTF is from that of *dynamicOPT*. We will use the number of substitutions as this metric. A substitution is a pair i, j whose appearances in MTF and *dynamicOPT* are reversed.

$$\Phi_0 = 0$$

$$\Phi_k \geq 0$$

Φ_k = number of substitutions between MTF and *dynamicOPT* after k queries

On access(x): MTF actual cost = the number of elements preceding x + 1. $\Delta\Phi_{MTF}$ = the number of elements that used to be swapped in OPT, less the number of elements

OPT actual cost $\geq$ number of elements before x and the number of "paid swaps" + 1. Each "paid swap" increases the potential by at most 1. If OPT moves x to the front, the potential does not increase. $\Delta\Phi_{OPT} \leq$ the number of "paid swaps"

amortized-cost-MFT: mtf-actual + $\Delta\Phi_{MTF}$ + $\Delta\Phi_{OPT}$ which is $\leq$ twice the swapped blues and the paid swaps + $1 \leq 2 \cdot actualOPT$

$$MTF \leq \sum_{x \in \sigma} amort(op(x)) \leq 2 \sum_{x \in \sigma}$$

Visual proof done on whiteboard

There is no deterministic algorithm that is better than 2-competitive.

2.5 Bitflip MTF

1.75-competitive on average.

The best is 1.6-competitive, and the lower bound has been proven to be 1.5.

3 Splay Trees

Worst case access is $\Omega(\log n)$ and $O(\log n)$.

Stochastic optimal Choose each node such that the sum of probabilities of the subtrees is more or less equal. However, this is only an approximation of the optimal. We can build the optimal from the bottom up using dynamic programming.

Let $cost_T = \sum_i p_i \cdot d(i)$ where $d(i)$ is the depth of the element i in the tree. Let $cost_{ij} = \min_{i \leq r \leq j} (cost_{i,r-1} + cost_{r+1,j}) + \sum_{k=i}^{j} p_k$.

Implementation of the DP is to compute the $cost_{ij}$ for all $i, j \in [1...n]^2$ which is $O(n^2)$ pairs, and fill a table starting from i, i. Computing $cost_{ij}$ costs us $O(n)$, which gives us a total cost of $O(n^3)$.

Knuth 71 $O(n^2)$ algorithm for construction. Knuth proved that $r_{i,j-1} \leq r_{ij} \leq r_{i,j+1}$. Colloquially, adding an element on one side of the tree should never move the root in the other direction.

Visual explanation of the intuition on the whiteboard.