Agenda

ANALYSIS OF ALGORITHMS:
ONLINE PROBLEMS AND AMORTIZED ANALYSIS
ADAPTIVE DATA STRUCTURES:
- DICTIONARY DATA STRUCTURE
- SELF-ORGANIZING LISTS
- COMPETITIVE PERFORMANCE

Sundar B. CS&IS, BITS, Pilani @Pilani

Dictionary Data Structure

- The way the list can be best arranged will depend on the input distribution:
  - i.e. an offline problem scenario in which the designer knows the input sequence ahead of time
    - (s)he can decide a data structure that is best for the sequence.
  - e.g. a BST where frequently accessed items are near the root
    - Recall the exercise on Dynamic Programming algorithm for Optimal BST

Dictionary Data Structure

- But when operations are online, one needs adaptive data structures:
  - these are referred to as self-organizing lists:
    - e.g. can you rearrange the BST if you know the frequency of one or more input items?

Self-Organizing Lists: Abstract Model

- Assume the following dictionary model:
  - A dictionary stores its elements as an unsorted list
    - find scans the list sequentially, i.e. to locate the i^{th} item, the cost is i.
    - Similarly, insert would cost i+1 for the i^{th} item.
- Suppose accesses are independent of each other and suppose the probability of accessing item i is given, say, p_i
  - An optimum algorithm will arrange items in non-increasing order by probability :
    - let us refer to this algorithm as DP (for decreasing probability).

Self-Organizing Lists

- Self-Organizing Strategies:
  - Move-to-Front (MF):
    - On access/insertion, move the item to the front, without changing the relative order of other items.
  - Transpose (T):
    - On access/insertion, exchange it with the preceding item
  - Frequency Count (FC):
    - Maintain the list in non-increasing order by frequency count. Increase count on access/insertion.

S-O Lists: Performance

- Suppose
  - the list size is fixed,
  - accesses are independent of each other, and
  - the probability of accessing item i is given, say, p_i
    - [Note: The last assumption is relevant only for DP.]
- What would be the competitive performance of the online algorithms?

S-O Lists: Performance of FC

- How competitive is FC w.r.t. DP ?
    - $E_{FC} \ / \ E_{DP} \cong 1$
  - Intuitive argument (based on the Law of Large Numbers):
    - Consider a long sequence of operations i.e.
      - #operations >> size of list
    - FC would have put the most frequent items in the beginning of the list
      - *according to the frequency (at this point)*
    - Now if you run DP on this sequence of operations on this list, how would they (FC and DP) compare?

S-O Lists: Performance of MF and T - Results

- Under assumptions similar to those of the last slide:
  - $E_{MF}(p) / E_{DP}(p) <= 2$.
  - $E_T(p) <= E_{MF}(p)$
    - But MF performs much better in practice:
      - because it soon converges to its asymptotic behavior given a random initial list
      - and it behaves close to a static decreasing frequency algorithm.
    - When tested on real data:
      - MF beats T consistently
      - MF is competitive with FC and sometimes better
        - *because MF is tuned for data with high locality*