CSL 757 Model Centric Algorithm Design

Major , Sem II 2016-17, Max 70, Time 2 1/2 hrs

Note (i) Write your answers neatly and precisely. You won't get a second chance to explain what you have written.
(ii) Every algorithm must be accompanied by proof of correctness and a formal analysis of running time and processor bound. Feel free to quote any result from the lectures without proof - for any anything new, you must prove it first.

1. Suppose we have a system with D disks where D is a parameter that allows simultaneous I-O of D blocks of size B - one from each disk. Here the size of internal memory $M \geq D \cdot B$. Initially N/DB blocks of data reside in each of the disks. Finally we would like to have the sorted elements ordered in some predefined way across the disks.

   Discuss an implementation of k-way mergesort for this system for an appropriate k where the goal is to minimize the number of parallel I-O block transfers. More specifically, comment on the following
   (i) What would be a reasonable lower bound for this model ?

   Since a D-disk system can be simulated by a single disk using D fold slow down, the lower bound is $\Omega(N/(DB) \cdot \frac{\log N/B}{\log M/B}$.
   (ii) What are main issues if we were to adapt the mergesort for the single disk model ? **(15 marks)**

   By striping the data across D disks, i.e., ordering the blocks as 1, 2, 3 ..D and wrapping around to the first disk as $D+1, D+2 \ldots$, we can run a DB way mergesort using $N/DB \cdot \frac{\log N/DB}{\log M/DB}$ I-Os which is the same as a single disk system with block size $B' = BD$. Note that this is not as good as the lower bound. We are not able to run a M/B way mergesort that will match the lower bound. (To run a M/B mergesort, we need to output M/B blocks of output from every DB blocks of input).

2. In the setting of the streaming model, where the length of the stream is not known is advance, how do you maintain a uniformly chosen subset of $k \geq 1$ elements. Recall that reservoir sampling is used to choose a single element uniformly at random without the knowledge of the length of the stream. Your method should provably pick any of the k-subsets with equal probability**(15 marks)**

   The algorithm given was correct, i.e., choose the latest element with probability k/i and exchange with one of the previously sampled element. However, the proof of uniformly chosen sample doesn't follow from the uniform sampling of elements. You have to inductively show that the subset maintained has probability $\frac{1}{\binom{i}{k}}$ for all i.This easily follows from the recurrence

   $$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

   Note that, if $k = 2$, by showing that every element is chosen with probability $2/i$, how does it follow that every pair is equally likely. (The elements cannot be independently chosen and the dependence has to be proved using conditional probability).

3. Consider an $N \times N \times N$ three dimensional mesh. Design an efficient sorting algorithm for N^3 numbers where each has $O(1)$ memory processor holds one element initially and finally. You can choose any pre-defined indexing scheme for the sorted permutation. Justify why your algorithm is optimal or near-optimal. **(20 marks)**

   We can adapt shearsort by choosing to sort every plane in snakelike sorted fashion (using shearsort

again) and then doing the vertical sorts. Each iteration will clean up half the planes so that the overall sorting takes $n \log^2 n$ steps. It is possible to sort faster using better adptations of shear-sort.

The obvious lower bound is the distance bound $3N$.

4. **Sliding window model of streaming** Consider a variation of the conventional streaming model where we are interested to compute a function of only the last N entries (instead of from the beginning of the stream).

   Given a 0-1 bit stream, design an algorithm to keep track of the number of 1's in the last N inputs using space s. Your answer can be approximate (in a multiplicative or additive manner) as a function of s. For instance for $s = N$, we can get an exact answer. **(20 marks)**

   We will maintain s counters each with time stamps $n_1, n_2, \ldots n_s$ where $1 \leq n_1 \leq n_2 \ldots n_s \leq N$ such that the count of 1's in each counter are almost equal (within 1). With each bit arrival, the time stamps are updated (increased by 1) and if necessary the number of 1's across the counters are redistributed to maintain the almost equal invariant. Once the time stamp of a counter expires (exceeds N) it is discarded and cycled back.

   The time stamp of a bucket corresponds to the most recent 1 in it. Moreover, the counts of each bucket $c_1 \leq c_2 \ldots c_s$ and maintained in a way such that $c_i = \{x, 2x\}$ for some appropriate integer x. This can be done by clubbing consecutive counters. For the first s values we can maintain explicit records - for the next 1, we club together the last two 1's and create a slot for the latest 1. This invariant can be maintained inductively.

   To answer a count query we add the counts and it can be easily shown that this is within multiplicative factor $1 + 1/s$ (note that total storage is $O(s \log N)$ bits.