CSE 6512 Randomization in Computing. Fall 2023

Exam #2 Solutions

---

1. Consider a k-uniform hypergraph $\mathcal{H}$ with less than 2^{k-1} edges. We will prove that it is 2-colorable. We color every vertex of $\mathcal{H}$ independently red or blue, each with probability $\frac{1}{2}$. The probability that the vertices of a given edge are all red or all blue is $p = 2 \times (1/2)^k$. If $\mathcal{H}$ has $< 2^{k-1}$ edges, the probability that there exists a monochromatic edge is $< p\ 2^{k-1} = 1$. As a result, there is a non-zero probability that no edge is monochromatic. This means that a proper coloring must exist.

2. Partition time into phases of length $2C(G)$ each. Let w be any node in the given graph $G(V, E)$. Probability that w is not visited in any phase is $\leq \left(\frac{1}{2}\right)$. Thus, probability that w is not visited in k successive phases is $\leq \left(\frac{1}{2}\right)^k$. Probability that there exists a node that has not been visited in k phases is $\leq n\left(\frac{1}{2}\right)^k$. This probability will be $\leq n^{-\alpha}$ if $k \geq (\alpha+1)\log n$. Thus, independent of the starting node of a random walk, the time taken to visit each node at least once is $\widetilde{O}(mn \log n)$.

3. Each processor picks a random element of B and checks if this element is in A. Checking can be done using binary search in $O(\log n)$ time. Call these two parallel steps a **phase** of the algorithm. After every phase, the processors can use the concurrent write facility to check (in $O(1)$ time) if at least one of them has found a correct answer. Repeat this phase as many times as it takes to identify a common element.

   The probability of success in any phase for a single processor is $\geq \frac{1}{n^{5/12}}$ since we know that there are $n^{7/12}$ common elements between A and B. Probability of failure in one phase, for any specific processor, is $\leq 1 - \frac{1}{n^{5/12}}$. The probability that every processor fails in a particular phase is $\leq \left(1 - \frac{1}{n^{5/12}}\right)^{n^{5/12}}$. Therefore, probability of failing in k successive phases is $\leq \left(1 - \frac{1}{n^{5/12}}\right)^{kn^{5/12}} \leq \exp(-k)$. This probability will be $\leq n^{-\alpha}$ if $k \geq \alpha \log n$. In other words, the run time of the algorithm is $\widetilde{O}(\log n)$.

4. Given two $n \times n$ matrices we can multiply them in $O(\log n)$ time using n^3 CREW PRAM processors as follows. Let A and B be the input matrices and let $C = AB$. $C_{ij} = \sum_{k=1}^{n} A_{ik}B_{kj}$. Assign n processors for each output value C_{ij}. These processors can compute $A_{i1}B_{1j}, A_{i2}B_{2j}, \ldots, A_{in}B_{nj}$ in $O(1)$ time and add them using a prefix computation in $O(\log n)$ time.

   We can use this algorithm and the repeated squaring technique to compute M^k in $O(\log n \log k)$ time.

   One of the processors can convert the integer k into binary in $O(\log k)$ time. Let this binary number be $b_q b_{q-1} \cdots b_1 b_0$, where $q = \lfloor \log k \rfloor$. The processors compute $M^2, M^4, \ldots, M^{2^q}$ in $O(\log n \log k)$ time. Followed by this they compute $\Pi_{b_i=1} M^{2^i}$. This also takes $O(\log n \log k)$ time.

5. We utilize the fact that we can search for an arbitrary element x in a sorted sequence of length n in $O(1)$ time using n^ϵ CREW PRAM processors, ϵ being any constant > 0. This search is known as n^ϵ-ary search.

   We assign n^ϵ processors to each of the keys in X. The processors associated with k_i perform a n^ϵ-ary search in Y to figure out the rank r_i of k_i in Y. As a result, we can compute the global rank of each key of X. In a similar manner we can compute the global rank of each key of Y. Once we know the ranks of the keys, we can output them in the order of their ranks. The total run time is $O(1)$.

6. Assume that we have n Common CRCW PRAM processors. Let $X = k_1, k_2, \ldots, k_n$ and $Y = l_1, l_2, \ldots, l_n$. We use two arrays $A[1:n]$ and $B[1:n]$. Here is a constant time algorithm:

   Step 1.
   for $1 \leq i \leq n$ **in parallel do**

   Processor i sets $A[i]$ and $B[i]$ to zero.

   Step 2.
   for $1 \leq i \leq n$ **in parallel do**

   Processor i tries to write k_i in $A[k_i]$;
   Processor i tries to write l_i in $B[l_i]$;

   At the end of the above step we have collected all the distinct values of X in A and all the distinct values of Y in B.

   Step 3.
   Processor 1 sets *Result* to No;
   for $1 \leq i \leq n$ **in parallel do**

   Processor i checks if $A[i] = B[i]$ and $A[i] \neq 0$. If so, it tries to set *Result* to Yes.

   Step 1 takes 2 units of time. Step 2 also takes 2 units of time. Step 3 takes 4 units of time. Thus the entire algorithm takes $O(1)$ time using n processors.