CSE 6512 Randomization in Computing. Fall 2023

Exam #2 (model) Solutions

---

1. Let us calculate the expected number of Hamiltonian paths in a random tournament $T(V, E)$ where every edge of $G(V, E)$ has a random orientation, chosen independently with probability $\frac{1}{2}$.

   Let π be a random permutation of $\{1, \ldots, n\}$. What is the probability that the sequence of nodes $\pi(1), \pi(2), \ldots, \pi(n)$ corresponds to a Hamiltonian path? This sequence will be Hamiltonian path if $(\pi(i), \pi(i+1))$ is a directed edge in the tournament, for $1 \leq i \leq (n-1)$. The probability of this is $\frac{1}{2^{n-1}}$. As a result, it follows that the expected number of Hamiltonian paths in a random tournament is $\frac{n!}{2^{n-1}}$. Therefore, there exists a tournament that has at least $\frac{n!}{2^{n-1}}$ Hamiltonian paths.

2. Here we use the fact that two sorted sequences of length N can be merged in $O(\log \log N)$ time using N CREW PRAM processors. We recursively merge the sequences $X_1, X_2, \ldots, X_{(k/2)}$ to get Y_1 using $\sum_{i=1}^{(k/2)} |X_i|$ processors. At the same time we recursively merge $X_{(k/2)+1}, X_{(k/2)+2}, \ldots, X_k$ to get Y_2 using $\sum_{i=(k/2)+1}^{k} |X_i|$ processors. The total number of processors used is clearly n.

   Followed by the above reursive steps, we merge Y_1 and Y_2 using n processors. This merging will take $O(\log \log n)$ time. Let $T(k)$ be the time needed to merge k sorted sequences using n processors. Then, we have:

   $$T(k) = T(k/2) + O(\log \log n).$$

   The above recurrence relation can be solved to get: $T(k) = O(\log k \log \log n)$.

3. In this problem we use the fact that we can sort a sequence of n integers in the range $[1, n \log^c n]$ in $\widetilde{O}(\log n)$ time using $\frac{n}{\log n}$ Arbitrary CRCW PRAM processors, where c is any constant.

   Let $X_1, X_2, \ldots, X_k$ be the input sequences. We start by sorting each of these sequences using the above algorithm. This will take $\widetilde{O}(\log n)$ time using $\frac{nk}{\log n}$ processors. Consider a complete binary tree where each leaf has one of these sorted sequences. We traverse this tree level by level starting at the level immediately above the leaves. At any node in the tree we compute the intersection of the two children.

   We can compute the intersection of two sorted sequences of length at most q each (where $q \leq n$) using $\frac{q}{\log n}$ processors in $\widetilde{O}(\log n)$ time. This means that the computations in each level of the tree can be computed in $\widetilde{O}(\log n)$ time using $\frac{nk}{\log n}$ processors.

   As a result, the entire algorithm runs in time $\widetilde{O}(\log n \log k)$ using $\frac{nk}{\log n}$ processors.

   Extra Credit: There are two cases to consider.

   Case 1: $k \leq \log n$: In this case use the previous algorithm.

Case 2: $k > \log n$: Note that the number of nodes in any level of the tree is one half of the number of nodes in the level below. We use the above technique of computing intersections for the bottom $2 \log \log n$ levels. The number of nodes in the $2 \log \log n^{\text{th}}$ level from the bottom is $\frac{n}{\log^2 n}$. From this level on, use the $\widetilde{O}\left(\frac{\log n}{\log \log n}\right)$ time algorithm for integer sorting (and hence intersections). The total time is $\widetilde{O}\left(\log n \log \log n + \frac{\log n}{\log \log n} \log k\right)$. Number of processors needed at any level of the tree is no more than $\frac{nk}{\log n}$.

4. Here is an algorithm: Pick a random sample S from A of size $n^{1/4}$. This can be done in $O(1)$ time using $n^{1/4}$ processors. Find the median x of S and output this element. We can find x, for example, using the optimal randomized selection algorithm we have discussed in class. Clearly, this algorithm runs in $\widetilde{O}(\log n)$ time.

   Analysis: Let $s = |S| = n^{1/4}$. Using one of the sampling lemmas we have stated in class, if $r = rank(x, A)$, then, $Pr\left[|r - \frac{s}{2}\frac{n}{s}| > \frac{n}{\sqrt{s}}\sqrt{4\alpha \log n}\right] \leq n^{-\alpha}$. In other words, $Pr\left[|r - \frac{n}{2}| > n^{7/8}\sqrt{4\alpha \log n}\right] \leq n^{-\alpha}$. This implies that $rank(x, A)$ lies in the interval $[\frac{3}{8}n, \frac{5}{8}n]$ with high probability.

5. Consider the computation of $C[i, j]$, for any $1 \leq i, j \leq n$. Using n processors we can compute the following n elements in $O(1)$ time: $A[i, 1] + B[1, j], A[i, 2] + B[2, j], \ldots, A[i, n] + B[n, j]$. Subsequently, we can compute the minimum of these n elements in $\widetilde{O}(1)$ time using n processors, using one of the selection algorithms we have discussed in class.

   Thus, we can compute the tropical product of two matrices in $\widetilde{O}(1)$ time using n^3 arbitrary CRCW PRAM processors.

6. Processor 1 sets *Result* to *false*. We first sort X using the sorting algorithm discussed in class in $\widetilde{O}(\log n)$ time using n arbitrary CRCW PRAM processors. Followed by this, we assign one input key per processor. In particular, processor i gets k_i (for $1 \leq i \leq n$). For $1 \leq i \leq n$, in parallel, processor i performs a binary search in X to see if $(r - k_i) \in X$; if so processor i tries to write *true* in *Result*. This step takes $O(\log n)$ time and at the end of this step, *Result* will have the correct answer. Clearly, the total run time of the algorithm is $\widetilde{O}(\log n)$.