Parallel Computation

Final Report

Dec. 30, 2003 $\sim$ Jan. 27, 2004

Problem 1 (4.14) A number s_0 and four sequences of numbers $\{a_1, a_2, \ldots, a_n\}$, $\{b_1, b_2, \ldots, b_n\}$, $\{d_1, d_2, \ldots, d_n\}$, and $\{e_1, e_2, \ldots, e_n\}$ are given. Suppose we have the following recurrence

$$s_i = (a_i s_{i-1} + b_i)/(e_i s_{i-1} + d_i).$$

Show how the sequence $s_1, s_2, \ldots, s_n$ can be computed in parallel in $O(\lg n)$ time under the PRAM model.

Problem 2 (4.18) In a Fibonacci sequence, the first two numbers are equal to 1, and each subsequent number is the sum of the previous two. Use prefix computation to generate the first n Fibonacci numbers.

Problem 3 (4.20) A sequence of pairs (l_i, r_i), $i = 1, 2, \ldots, n$, is given, representing intervals on a horizontal line, with l_i and r_i the left and right endpoints, respectively, of interval J_i. The left endpoint l_i is *contained* in interval J_k if $l_k \leq l_i \leq r_k$. It is required to determine, for each l_i, $i = 1, 2, \ldots, n$, the number of intervals in which it is contained. Show how prefix computation can be used to solve this problem.

Problem 4 (4.21) Given a polynomial $g(x)$ of degree n, it is required to evaluate g at points $h\epsilon$ where ϵ is an arbitrary real and h is an integer taking consecutive values $h = a, a+1, \ldots, b$, for two given integers a and b, $a < b$. Suggest an algorithm for solving this problem using prefix computation.

Problem 5 (4.22) A two-dimensional array I of numbers is given. It is required to find a rectangular sub-array of I with the largest sum among all such sub-array of I. Use prefix computation to solve this problem on a PRAM. Analyze the running time of your algorithm as a function of the number of rows and columns of I.

Problem 6 (4.25) In Section 4.9.1, a lower bound of $\Omega(n \lg n)$ was derived on the number of operations required to perform GPG when $\prec$ and $*$ are $<$ and $+$, respectively. Can similar lower bounds be derived using other linear and other binary operations?

Problem 7 (4.29) Extend the problem of computing maximal points to a set of points in a three-dimensional space. Then show how the problem can be solved using GPC.

Problem 8 Suppose we have designed a PRAM algorithm with running time $O(\frac{n}{p} \lg \lg p + \lg p)$ where n is the problem size and p is the number of processors used in solving this problem. How many processors should we use in order to minimize the required time for this PRAM algorithm?

Problem 9 (5.5) A binary sequence of length n consisting of a (possibly empty) string of 0's followed by a (possibly empty) string of 1's.

(a) Given a PRAM with N processors, $1 \leq N \leq n$, describe an algorithm for determining the number of 1's in the sequence.

(b) Repeat **(a)** for the case where the sequence consisting of a string of 0's, followed by a string of 1's, followed by a string of 0's.

Problem 10 (5.10) Let X and Y be two sorted sequences of numbers of length n and m, respectively.

(a) Develop a RAM algorithm for finding the kth smallest element among all elements of X and Y.

(b) Use the algorithm in **(a)** to obtain a PRAM algorithm for merging X and Y into a third sorted sequence Z of length $n + m$.

Problem 11 (5.13) Can you find an algorithm for selecting the kth smallest element of a sequence of n numbers $S = \{s_1, s_2, \ldots, s_n\}$ that uses $O(n/\log^{\epsilon} n)$ processors and runs in $O(\log^{\epsilon} n)$ time, for some $0 \leq \epsilon \leq 1$?

Problem 12 (5.17) Use algorithm RAM SELECT to design a sequential divide-and-conquer algorithm for computing the convex hull of a set of n points in the plane.

Problem 13 (5.22) For a set S of n points in the plane, it is required to determine which two points are closest to one another. Describe and analyze a RAM algorithm for solving this problem based on the divide-and-conquer approach.

Problem 14 (5.23) Solve the following recurrence

$$\begin{aligned} q(1) &= 1, \\ q(b) &= 3q(b/2) + ab, \end{aligned}$$

where a is a constant.

Problem 15 (5.24) Given two sequences $\{a_1, a_2, \ldots, a_n\}$, where $a_1 = 0$, and $\{b_1, b_2, \ldots, b_n\}$, a closed-form solution to the first-order linear recurrence

$$\begin{aligned} x_1 &= b_1, \\ x_i &= a_i x_{i-1} + b_i, \quad i = 2, 3, \ldots, n, \end{aligned}$$

is given by

$$x_i = \sum_{j=0}^{i} \left(\prod_{k=j+1}^{i} a_k \right) b_j$$

for $i = 2, 3, \ldots, n$. Design a parallel divide-and-conquer approach for computing $x_2, x_3, \ldots, x_n$.

Problem 16 (5.26) Given a bipartite graph G, it is required to color the edges of G with a minimum number of colors such that each edge is assigned a color and no two edges adjacent to the same vertex receive the same color. Design a parallel divide-and-conquer algorithm to solve this problem.

Problem 17 (6.1) Suppose that we are given m directed and rooted binary trees with a total of n vertices. Each vertex has a pointer to its parent. The trees are identified by their roots, but no vertex (unless it is a root) knows the identity of the tree to which it belongs. Design an algorithm that allows each vertex to know the identity of its tree, and analyze the processor and time requirements of your algorithm.

Problem 18 (6.7) A method is given in Section 6.3.3 for determining the level of a vertex in a rooted tree. Suggest alternative ways to perform this computation. (Hints: Try using function $pos, preorder$, and $parent$.)

Problem 19 (6.8) Suppose that each vertex in a rooted directed tree DT knows its parent. Design a PRAM algorithm (different from the one given in Section 6.3.3) for enumerating the descendants of each vertex in DT (i.e., to evaluate $des(v_j)$). (Hints: Use pointer jumping.)

Problem 20 (6.10) Let a directed and rooted *complete binary tree* with n leaves be given. Each vertex stores a number and points to its parent, with the root's pointer equal to *nil*. It is required to perform a prefix computation on the tree such that each vertex stores the minimum of all numbers in its subtree. Design a parallel algorithm in $O(\lg \lg n)$ time. Generalize your algorithm so that it runs in $O(\lg n)$ time on arbitrary trees.

Problem 21 (6.14) Prove by induction on the number of nodes, that the algorithm of Section 6.3.2 builds an Euler tour and not several small cycles.

Problem 22 (6.17) As defined in Section 4.10.3, the *inorder* traversal of a rooted binary tree visits the vertices in the following order: The left subtree is traversed recursively in inorder, then the root is visited, and finally, the right subtree is traversed recursively in inorder. Given a binary tree T with n nodes, show that how the Euler tour technique can be used to number the vertices of T in the order of the inorder traversal of T. Analyze the running time and processor requirements of your algorithm.

Problem 23 (6.18) Suppose that the vertices of a rooted *complete binary tree* T have been assigned inorder numbers, as in the previous problem. Show that the lowest common ancestor of any pair of vertices of T (as defined in Section 6.3.4) can be obtained in constant time by one processor. Here we assume that *any* reasonable word operation takes constant time.

Problem 24 (6.19) The algorithm of Section 6.3.4 solves the interval minima problem in $O(\lg n)$ time using $O(n)$ processors.

(a) Show that this algorithm is not cost optimal by describing an $O(n)$-time sequential algorithm to solve the interval minima problem.

(b) Obtain a cost-optimal PRAM algorithm for this problem — that is, an algorithm which runs in $O(\lg n)$ time using $O(n/\lg n)$ processors — based on the following approach:

   (i) Divide the sequence S into $n/\lg n$ subsequences, and each of size $\lg n$.

   (ii) Solve the interval minima problem for each subsequence sequentially, using the optimal algorithm developed in (a).

   (iii) Compute the prefix and suffix minima of each subsequence of S; let a_i be the smallest element of the ith subsequence.

   (iv) Apply the algorithm of Section 6.3.4 to the sequence $\{a_1, a_2, \ldots, a_{n/\lg n}\}$.