Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn
M. Ahmadi, A. R. Molla, O. Saukh February 4, 2016

Algorithm Theory, Winter Term 2015/16
Problem Set 14

hand in (hard copy or electronically) by 10:15, Thursday February 11, 2016,

Note: The points of this problem set are not required to be admitted to the exam. However, any points you achieve for this problem set count as bonus points towards getting 50% of the achievable points of all other problem sets.

Exercise 1: All-prefix-sums on Multi-dimensional Matrices (6 points)

In this exercise, we consider a generalization of the the all-prefix-sums problem discussed in the lecture. We study the following all-prefix-sums problem defined on a d-dimensional array.
We are given a $n \times n \times \cdots \times n$ array $\mathcal{A}$ with entries $a_{i_1,i_2,\dots,i_d}$ (for $i_j \in \{1,\dots,n\}$). The goal is to calculate the all-prefix-sums array $\mathcal{S}$ with entries $s_{i_1,i_2,\dots,i_d}$ which are defined as follows

$$s_{i_1,i_2,\dots,i_d} := \sum_{j_1=1}^{i_1}\sum_{j_2=1}^{i_2}\cdots\sum_{j_d=1}^{i_d} a_{j_1,j_2,\dots,j_d}$$

Note that for $d = 1$, the problem is the usual all-prefix-sums problem from the lecture.

(a) (3 points) To warm up, we first consider the case $d = 2$. Give an efficient algorithm to solve the 2-dimensional all-prefix-sums problem. What are the work T_1 and span T_∞ of your solution.

Hint: *The problem can be solved by performing $2n$ standard all-prefix-sums computations.*

(b) (2 points) Generalize the above algorithm to $d \geq 2$ dimensions. What are work T_1 and span T_∞ of the resulting parallel algorithm?

(c) (1 points) What is the minimum number of processors needed such that asymptotically, the maximum possible speed-up can be achieved?

Exercise 2: Merging Two Sorted Arrays (6 points)

You are given two sorted arrays $A = [a_1,\dots,a_n]$ and $B = [b_1,\dots,b_n]$, each of size n. The goal is to merge them into one sorted array $C = [c_1,\dots,c_{2n}]$ of length $2n$.

(a) (1.5 points) We first consider the following subproblem. Given an index $i \in \{1,\dots,n\}$, we want to find the final position $j \in \{1,\dots,2n\}$ of the value a_i in the array C. Give a fast sequential algorithm to compute j. What is the (sequential) running time of your algorithm?

(b) (1.5 points) Use the above algorithm to construct a parallel merging algorithm. The work T_1 of your algorithm should be at most $O(n \log n)$ and the span T_∞ should be (asymptotically) as small as possible. What is the span T_∞ of your algorithm?

(c) (3 points) We now want to solve the merging problem in constant time (in parallel). Show that by using $O(n)$ processes, the subproblem considered in (a) can be solved in $O(1)$ time. Use this to get a constant-time parallel algorithm to merge the two sorted arrays. How many processors do you need to achieve a constant-time algorithm?