Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen
Prof. Dr. Ernst W. Mayr
Chris Pinkau

Winter Semester 2012/13
Problem Set 1
October 23, 2012

Parallel Algorithms

Due date: October 30, 2012 **before** *class!*

Problem 1 (10 Points)

Recall the definition of the Landau notation for $f, g : \mathbb{N} \to \mathbb{N}$:

$$
\begin{array}{lll}
f = \mathcal{O}(g) & :\Longleftrightarrow & \exists c > 0 \,\exists n_0 \in \mathbb{N}\, \forall n \geq n_0 : f(n) \leq c \cdot g(n), \\
f = \Omega(g) & :\Longleftrightarrow & g = \mathcal{O}(f), \\
f = \Theta(g) & :\Longleftrightarrow & f = \mathcal{O}(g) \wedge f = \Omega(g), \\
f = o(g) & :\Longleftrightarrow & \forall c > 0 \,\exists n_0 \in \mathbb{N}\, \forall n \geq n_0 : f(n) \leq c \cdot g(n), \\
f = \omega(g) & :\Longleftrightarrow & g = o(f).
\end{array}
$$

Remark: Depending on the author, you will see the notations $f = \mathcal{O}(g)$ or $f \in \mathcal{O}(g)$, respectively. Both notations are tolerated, just be consistent with yours!

(a) For strictly positive functions f, g, i.e. $f(n), g(n) > 0$ for all $n \in \mathbb{N}$, show or disprove:

- (i) $f = \Theta(g)$ if and only if there exist $c_1, c_2 > 0$ such that $c_1 \leq \frac{f(n)}{g(n)} \leq c_2$ for almost all $n \in \mathbb{N}$. ("almost all" is equivalent to "except for finitely many").
- (ii) $f = o(g)$ if and only if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.

(b) Show that polynomial growth is dominated by exponential growth, i.e. for every $d > 0, b > 1$ it holds that $n^d = o(b^n)$.

(c) For each of the following pairs of functions f, g determine whether $f = o(g), g = o(f)$ or $f = \Theta(g)$.

- (i) $f(n) = n^2, \quad g(n) = 2n^2 + 100\sqrt{n}$,
- (ii) $f(n) = 1000n, \quad g(n) = n \log n$,
- (iii) $f(n) = 2^{2^{n+1}}, \quad g(n) = 2^{2^n}$,
- (iv) $f(n) = n^n, \quad g(n) = 2^{2^n}$.

Problem 2 (10 Points)

Given an array $A = A(1) \ldots A(n)$ of $n = 2^k$ numbers, the task is to compute the sum $A(1) + \cdots + A(n)$. Briefly explain the algorithmic model you use for solving this problem and give an example of an efficient parallel algorithm (and its running time) when using a hypercube network.
What if your network has less than n processors?

Problem 3 (10 Points)

Given an n-dimensional hypercube, find and prove the following:

(i) the number of vertices,

(ii) the number of edges,

(iii) the diameter,

(iv) the bisection width (the bisection width is the minimal number of edges which have to be cut to create two networks with $n/2$ vertices each).

Problem 4 (10 Points)

Given a tree network, find a numbering of the vertices/gates, such that for every two sibling vertices the number of their common parent vertex can be easily computed.