Technische Universität München
Institut für Informatik
Theoretical Computer Science

WS 2017/18

Fundamental Algorithms 9

Exercise 1 (Parallel sort)

BucketSortPRAM is a proposed parallel implementation of the BucketSort algorithm in a PRAM, n processor model. `copy` is a sequential algorithm which takes an array A and an array of arrays B as arguments and copies the elements of B into A in order. The function `index` distributes the array elements *evenly* into buckets, i.e. elements are assigned to buckets with (roughly) the same probability. Moreover, larger elements are assigned to larger buckets, i.e. $\texttt{index}(a) < \texttt{index}(b)$ implies $a < b$. nb denotes the number of buckets used for the sorting.

Algorithm 1: BucketSortPRAM

Input: A: Array$[1..n]$
Result: A is sorted
$B \leftarrow$ Array$[1..nb]$;
for $i = 1$ **to** n *in parallel* **do** `insert(`$B[\texttt{index}(A[i])]$, $A[i]$`)`;
for $i = 1$ **to** nb *in parallel* **do** `BubbleSort(`$B[i]$`)`;
`copy(`A, B`)`;

1. For the two parallel loops in BucketSortPRAM, state for both arrays A and B whether there is concurrent or exclusive read / write access to their elements.

2. Implement a parallel version of `copy` in an EREW PRAM model. You may use the following functions:
   - `len`(A): Returns the length of the array A in constant time.
   - `pfill`(A, v): Uses $\frac{n}{2}$ processors to set all entries of A to v in $O(\log n)$ time (binary fan-out).
   - `pPrefAdd`(A): Computes the addition-prefixes of A, i.e. for the returned array B we have that $B[i] = \sum_{k=1}^{i} A[i]$. The computation uses $\frac{n}{2}$ processors and $O(\log n)$ time.

   What is the parallel complexity (depending on nb and l, the maximum length of any array in B), and how many processors can your algorithm use?